```
;********************************************************************************
; UNIX386.ASM (RETRO UNIX 386 Kernel) - v0.2.1.0
;--------------------------------------------------------------------------------
; NASM version 2.11 (unix386.s)
;
; RETRO UNIX 386 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.2) by ERDOGAN TAN (Beginning: 24/12/2013)
;
; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
; (v0.1 - Beginning: 11/07/2012)
;
; [ Last Modification: 04/02/2016 ]
;
; Derived from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document>
;
; Derived from 'UNIX v7/x86' source code by Robert Nordier (1999)
; UNIX V7/x86 source code: see www.nordier.com/v7x86 for details.
;
;********************************************************************************

; 24/12/2013

; Entering protected mode:
; Derived from 'simple_asm.txt' source code file and
; 'The world of Protected mode' tutorial/article by Gregor Brunmar (2003)
; (gregor.brunmar@home.se)
; http://www.osdever.net/tutorials/view/the-world-of-protected-mode
;

; "The Real, Protected, Long mode assembly tutorial for PCs"
; by Michael Chourdakis (2009)
; http://www.codeproject.com/Articles/45788/
; http://www.michaelchourdakis.com
;

; Global Descriptor Table:
; Derived from 'head.s" source code of Linux v1.0 kernel
; by Linus Torvalds (1991-1992)
;

KLOAD equ 10000h ; Kernel loading address
      ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
KCODE equ 08h      ; Code segment descriptor (ring 0)
KDATA equ 10h      ; Data segment descriptor (ring 0)
; 19/03/2015
UCODE equ 1Bh ; 18h + 3h  (ring 3)
UDATA equ 23h ; 20h + 3h  (ring 3)
; 24/03/2015
TSS   equ 28h      ; Task state segment descriptor (ring 0)
; 19/03/2015
CORE  equ 400000h  ; Start of USER's virtual/linear address space
                   ; (at the end of the 1st 4MB)
ECORE equ 0FFC00000h ; End of USER's virtual address space (4GB - 4MB)
                   ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)

; 27/12/2013
KEND    equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)

; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
;--------- CMOS TABLE LOCATION ADDRESS'S -----------------------------------
CMOS_SECONDS EQU    00H          ; SECONDS (BCD)
CMOS_MINUTES EQU    02H          ; MINUTES (BCD)
CMOS_HOURS   EQU    04H          ; HOURS (BCD)
CMOS_DAY_WEEK      EQU    06H          ; DAY OF THE WEEK  (BCD)
CMOS_DAY_MONTH     EQU    07H          ; DAY OF THE MONTH (BCD)
CMOS_MONTH   EQU    08H          ; MONTH (BCD)
CMOS_YEAR    EQU    09H          ; YEAR (TWO DIGITS) (BCD)
CMOS_CENTURY EQU    32H          ; DATE CENTURY BYTE (BCD)
CMOS_REG_A   EQU    0AH          ; STATUS REGISTER A
CMOS_REG_B   EQU    00BH         ; STATUS REGISTER B  ALARM
CMOS_REG_C   EQU    00CH         ; STATUS REGISTER C  FLAGS
CMOS_REG_D   EQU    0DH          ; STATUS REGISTER D  BATTERY
CMOS_SHUT_DOWN     EQU    0FH          ; SHUTDOWN STATUS COMMAND BYTE
;--------------------------------------
;     CMOS EQUATES FOR THIS SYSTEM     ;
;--------------------------------------------------------------------------------
CMOS_PORT    EQU    070H         ; I/O ADDRESS OF CMOS ADDRESS PORT
CMOS_DATA    EQU    071H         ; I/O ADDRESS OF CMOS DATA PORT
NMI          EQU    10000000B    ; DISABLE NMI INTERRUPTS MASK -
                                 ; HIGH BIT OF CMOS LOCATION ADDRESS

; Memory Allocation Table Address
; 05/11/2014
; 31/10/2014
MEM_ALLOC_TBL      equ    100000h            ; Memory Allocation Table at the end of
                                 ; the 1st 1 MB memory space.
                                 ; (This address must be aligned
                                 ;  on 128 KB boundary, if it will be
                                 ;  changed later.)
                                 ; ((lower 17 bits of 32 bit M.A.T.
                                 ;   address must be ZERO)).
                                 ; ((((Reason: 32 bit allocation
                                 ;     instructions, dword steps)))
                                 ; (((byte >> 12 --> page >> 5)))
;04/11/2014
PDE_A_PRESENT      equ    1            ; Present flag for PDE
PDE_A_WRITE  equ    2            ; Writable (write permission) flag
PDE_A_USER   equ    4            ; User (non-system/kernel) page flag
;
PTE_A_PRESENT      equ    1            ; Present flag for PTE (bit 0)
PTE_A_WRITE  equ    2            ; Writable (write permission) flag (bit 1)
PTE_A_USER   equ    4            ; User (non-system/kernel) page flag (bit 2)
```

```
                                PTE_A_ACCESS    equ      32           ; Accessed flag (bit 5) ; 09/03/2015

                                ; 17/02/2015 (unix386.s)
                                ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
                                DPT_SEGM equ 09000h  ; FDPT segment (EDD v1.1, EDD v3)
                                ;
                                HD0_DPT      equ 0     ; Disk parameter table address for hd0
                                HD1_DPT      equ 32          ; Disk parameter table address for hd1
                                HD2_DPT      equ 64          ; Disk parameter table address for hd2
                                HD3_DPT      equ 96          ; Disk parameter table address for hd3


                                ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
                                ;      (HDPT: Programmer's Guide to the AMIBIOS, 1993)
                                ;
                                FDPT_CYLS    equ 0 ; 1 word, number of cylinders
                                FDPT_HDS     equ 2 ; 1 byte, number of heads
                                FDPT_TT             equ 3 ; 1 byte, A0h = translated FDPT with logical values
                                                    ; otherwise it is standard FDPT with physical values
                                FDPT_PCMP    equ 5 ; 1 word, starting write precompensation cylinder
                                                    ; (obsolete for IDE/ATA drives)
                                FDPT_CB             equ 8 ; 1 byte, drive control byte
                                                     ; Bits 7-6 : Enable or disable retries (00h = enable)
                                                     ; Bit 5      : 1 = Defect map is located at last cyl. + 1
                                                     ; Bit 4 : Reserved. Always 0
                                                     ; Bit 3 : Set to 1 if more than 8 heads
                                                     ; Bit 2-0 : Reserved. Alsways 0
                                FDPT_LZ             equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
                                FDPT_SPT     equ 14 ; 1 byte, sectors per track

                                ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
                                ; (11 bytes long) will be used by diskette handler/bios
                                ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).

                                [BITS 16]       ; We need 16-bit intructions for Real mode

                                [ORG 0]
                                      ; 12/11/2014
                                      ; Save boot drive number (that is default root drive)
00000000 8816[1A6B]                   mov    [boot_drv], dl ; physical drv number

                                      ; Determine installed memory
                                      ; 31/10/2014
                                      ;
00000004 B801E8                       mov    ax, 0E801h ; Get memory size
00000007 CD15                         int    15h       ; for large configurations
00000009 7308                         jnc    short chk_ms
0000000B B488                         mov    ah, 88h    ; Get extended memory size
0000000D CD15                         int    15h
                                      ;
                                      ;mov   al, 17h      ; Extended memory (1K blocks) low byte
                                      ;out   70h, al ; select CMOS register
                                      ;in    al, 71h ; read data (1 byte)
                                      ;mov   cl, al
                                      ;mov   al, 18h ; Extended memory (1K blocks) high byte
                                      ;out   70h, al ; select CMOS register
                                      ;in    al, 71h ; read data (1 byte)
                                      ;mov   ch, al
                                      ;
0000000F 89C1                         mov    cx, ax
00000011 31D2                         xor    dx, dx
                                chk_ms:
00000013 890E[F06D]                   mov    [mem_1m_1k], cx
00000017 8916[F26D]                   mov    [mem_16m_64k], dx
                                      ; 05/11/2014
                                      ;and   dx, dx
                                      ;jz    short L2
0000001B 81F90004                       cmp     cx, 1024
0000001F 7315                         jnb    short L0
                                              ; insufficient memory_error
                                              ; Minimum 2 MB memory is needed...
                                      ; 05/11/2014
                                      ; (real mode error printing)
00000021 FB                           sti
00000022 BE[696C]                     mov    si, msg_out_of_memory
00000025 BB0700                       mov    bx, 7
00000028 B40E                         mov    ah, 0Eh      ; write tty
                                oom_1:
0000002A AC                           lodsb
0000002B 08C0                         or     al, al
0000002D 7404                         jz     short oom_2
0000002F CD10                         int    10h
00000031 EBF7                         jmp    short oom_1
                                oom_2:
00000033 F4                             hlt
00000034 EBFD                         jmp    short oom_2

                                L0:
                                %include 'diskinit.inc' ; 07/03/2015
                              <1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC
                              <1> ; Last Modification: 04/02/2016
                              <1>
                              <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
                              <1>
                              <1> ; ///////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////
                              <1>
                              <1>       ; 10/12/2014 - 02/02/2015 - dsectrm2.s
                              <1> ;L0:
                              <1>       ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
                              <1>       ; Detecting disk drives... (by help of ROM-BIOS)
00000036 BA7F00               <1>       mov    dx, 7Fh
                              <1> L1:
00000039 FEC2                 <1>       inc    dl
0000003B B441                 <1>       mov    ah, 41h ; Check extensions present
```

```
                                <1>                     ; Phoenix EDD v1.1 - EDD v3
0000003D BBAA55                 <1>       mov   bx, 55AAh
00000040 CD13                   <1>       int   13h
00000042 721A                   <1>       jc    short L2
                                <1>
00000044 81FB55AA               <1>       cmp   bx, 0AA55h
00000048 7514                   <1>       jne   short L2
0000004A FE06[1D6B]             <1>       inc   byte [hdc]   ; count of hard disks (EDD present)
0000004E 8816[1C6B]             <1>         mov     [last_drv], dl  ; last hard disk number
00000052 BB[A06A]               <1>       mov   bx, hd0_type - 80h
00000055 01D3                   <1>       add   bx, dx
00000057 880F                   <1>       mov   [bx], cl ; Interface support bit map in CX
                                <1>                      ; Bit 0 - 1, Fixed disk access subset ready
                                <1>                      ; Bit 1 - 1, Drv locking and ejecting ready
                                <1>                      ; Bit 2 - 1, Enhanced Disk Drive Support
                                <1>                           ;            (EDD) ready (DPTE ready)
                                <1>                      ; Bit 3 - 1, 64bit extensions are present
                                <1>                           ;            (EDD-3)
                                <1>                      ; Bit 4 to 15 - 0, Reserved
00000059 80FA83                 <1>       cmp   dl, 83h       ; drive number < 83h
0000005C 72DB                   <1>       jb    short L1
                                <1> L2:
                                <1>       ; 23/11/2014
                                <1>       ; 19/11/2014
0000005E 30D2                   <1>       xor   dl, dl  ; 0
                                <1>       ; 04/02/2016 (esi -> si)
00000060 BE[1E6B]               <1>       mov   si, fd0_type
                                <1> L3:
                                <1>       ; 14/01/2015
00000063 8816[1B6B]             <1>       mov   [drv], dl
                                <1>       ;
00000067 B408                   <1>       mov   ah, 08h ; Return drive parameters
00000069 CD13                   <1>       int   13h
0000006B 7210                   <1>       jc    short L4
                                <1>             ; BL = drive type (for floppy drives)
                                <1>             ; DL = number of floppy drives
                                <1>             ;
                                <1>             ; ES:DI = Address of DPT from BIOS
                                <1>             ;
0000006D 881C                   <1>       mov   [si], bl ;  Drive type
                                <1>                    ; 4 = 1.44 MB, 80 track, 3 1/2"
                                <1>       ; 14/01/2015
0000006F E8A202                 <1>       call  set_disk_parms
                                <1>       ; 10/12/2014
00000072 81FE[1E6B]             <1>       cmp   si, fd0_type
00000076 7705                   <1>       ja    short L4
00000078 46                     <1>       inc   si ; fd1_type
00000079 B201                   <1>       mov   dl, 1
0000007B EBE6                   <1>       jmp   short L3
                                <1> L4:
                                <1>       ; Older BIOS (INT 13h, AH = 48h is not available)
0000007D B27F                   <1>       mov   dl, 7Fh
                                <1>       ; 24/12/2014 (Temporary)
0000007F 803E[1D6B]00           <1>       cmp   byte [hdc], 0 ; EDD present or not ?
00000084 0F879000               <1>         ja      L10       ; yes, all fixed disk operations
                                <1>                       ; will be performed according to
                                <1>                       ; present EDD specification
                                <1> L6:
00000088 FEC2                   <1>       inc   dl
0000008A 8816[1B6B]             <1>         mov     [drv], dl
0000008E 8816[1C6B]             <1>         mov     [last_drv], dl ; 14/01/2015
00000092 B408                   <1>       mov   ah, 08h ; Return drive parameters
00000094 CD13                   <1>       int   13h    ; (conventional function)
00000096 0F828201               <1>         jc      L13       ; fixed disk drive not ready
0000009A 8816[1D6B]             <1>         mov     [hdc], dl ; number of drives
                                <1>       ;; 14/01/2013
                                <1>       ;;push cx
0000009E E87302                 <1>       call  set_disk_parms
                                <1>       ;;pop  cx
                                <1>       ;
                                <1>       ;;and  cl, 3Fh       ; sectors per track (bits 0-6)
000000A1 8A16[1B6B]             <1>         mov     dl, [drv]
000000A5 BB0401                 <1>       mov   bx, 65*4 ; hd0 parameters table (INT 41h)
000000A8 80FA80                 <1>       cmp   dl, 80h
000000AB 7603                   <1>       jna   short L7
000000AD 83C314                 <1>       add   bx, 5*4       ; hd1 parameters table (INT 46h)
                                <1> L7:
000000B0 31C0                   <1>       xor   ax, ax
000000B2 8ED8                   <1>       mov   ds, ax
000000B4 8B37                   <1>         mov     si, [bx]
000000B6 8B4702                 <1>         mov     ax, [bx+2]
000000B9 8ED8                   <1>       mov   ds, ax
000000BB 3A4C0E                 <1>         cmp     cl, [si+FDPT_SPT] ; sectors per track
000000BE 0F855601               <1>         jne     L12 ; invalid FDPT
000000C2 BF0000                 <1>       mov   di, HD0_DPT
000000C5 80FA80                 <1>       cmp   dl, 80h
000000C8 7603                   <1>       jna   short L8
000000CA BF2000                 <1>       mov   di, HD1_DPT
                                <1> L8:
                                <1>       ; 30/12/2014
000000CD B80090                 <1>       mov   ax, DPT_SEGM
000000D0 8EC0                   <1>       mov   es, ax
                                <1>       ; 24/12/2014
000000D2 B90800                 <1>       mov   cx, 8
000000D5 F3A5                   <1>       rep   movsw  ; copy 16 bytes to the kernel's DPT location
000000D7 8CC8                   <1>       mov   ax, cs
000000D9 8ED8                   <1>       mov   ds, ax
                                <1>       ; 02/02/2015
000000DB 8A0E[1B6B]             <1>         mov     cl, [drv]
000000DF 88CB                   <1>       mov   bl, cl
000000E1 B8F001                 <1>       mov   ax, 1F0h
000000E4 80E301                 <1>       and   bl, 1
000000E7 7406                   <1>       jz    short L9
000000E9 C0E304                 <1>       shl   bl, 4
000000EC 2D8000                 <1>       sub   ax, 1F0h-170h
```

```
                                <1> L9:
000000EF AB                     <1>       stosw  ; I/O PORT Base Address (1F0h, 170h)
000000F0 050602                 <1>       add    ax, 206h
000000F3 AB                     <1>       stosw  ; CONTROL PORT Address (3F6h, 376h)
000000F4 88D8                   <1>       mov    al, bl
000000F6 04A0                   <1>       add    al, 0A0h
000000F8 AA                     <1>       stosb  ; Device/Head Register upper nibble
                                <1>       ;
000000F9 FE06[1B6B]             <1>       inc    byte [drv]
000000FD BB[A06A]               <1>       mov    bx, hd0_type - 80h
00000100 01CB                   <1>       add    bx, cx
00000102 800F80                 <1>         or     byte [bx], 80h  ; present sign (when lower nibble is 0)
00000105 A0[1D6B]               <1>       mov    al, [hdc]
00000108 FEC8                   <1>       dec    al
0000010A 0F840E01               <1>         jz     L13
0000010E 80FA80                 <1>       cmp    dl, 80h
00000111 0F8673FF               <1>         jna    L6
00000115 E90401                 <1>         jmp    L13
                                <1> L10:
00000118 FEC2                   <1>       inc    dl
                                <1>       ; 25/12/2014
0000011A 8816[1B6B]             <1>       mov    [drv], dl
0000011E B408                   <1>       mov    ah, 08h ; Return drive parameters
00000120 CD13                   <1>       int    13h    ; (conventional function)
00000122 0F82F600               <1>         jc     L13
                                <1>       ; 14/01/2015
00000126 8A16[1B6B]             <1>       mov    dl, [drv]
0000012A 52                     <1>       push   dx
0000012B 51                     <1>       push   cx
0000012C E8E501                 <1>       call   set_disk_parms
0000012F 59                     <1>       pop    cx
00000130 5A                     <1>       pop    dx
                                <1>       ; 04/02/2016 (esi -> si)
00000131 BE[3485]               <1>       mov    si, _end ; 30 byte temporary buffer address
                                <1>                       ; at the '_end' of kernel.
00000134 C7041E00               <1>       mov    word [si], 30
00000138 B448                   <1>       mov    ah, 48h       ; Get drive parameters (EDD function)
0000013A CD13                   <1>       int    13h
0000013C 0F82DC00               <1>         jc     L13
                                <1>       ; 04/02/2016 (ebx -> bx)
                                <1>       ; 14/01/2015
00000140 29DB                   <1>       sub    bx, bx
00000142 88D3                   <1>       mov    bl, dl
00000144 80EB80                 <1>       sub    bl, 80h
00000147 81C3[206B]             <1>       add    bx, hd0_type
0000014B 8A07                   <1>       mov    al, [bx]
0000014D 0C80                   <1>       or     al, 80h
0000014F 8807                   <1>       mov    [bx], al
00000151 81EB[1E6B]             <1>       sub    bx, hd0_type - 2 ; 15/01/2015
00000155 81C3[6A6B]             <1>       add    bx, drv.status
00000159 8807                   <1>       mov    [bx], al
                                <1>       ; 04/02/2016 (eax -> ax)
0000015B 8B4410                 <1>       mov    ax, [si+16]
0000015E 854412                 <1>       test   ax, [si+18]
00000161 7412                   <1>       jz     short L10_A0h
                                <1>                     ; 'CHS only' disks on EDD system
                                <1>                     ;  are reported with ZERO disk size
00000163 81EB[6A6B]             <1>       sub    bx, drv.status
00000167 C1E302                 <1>       shl    bx, 2
0000016A 81C3[4E6B]             <1>       add    bx, drv.size ; disk size (in sectors)
0000016E 8907                   <1>       mov    [bx], ax
00000170 8B4412                 <1>       mov    ax, [si+18]
00000173 8907                   <1>       mov    [bx], ax
                                <1>
                                <1> L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
                                <1>        ; for CHS disks (28/02/2015)
                                <1>       ; 30/12/2014
00000175 BF0000                 <1>       mov    di, HD0_DPT
00000178 88D0                   <1>       mov    al, dl
0000017A 83E003                 <1>       and    ax, 3
0000017D C0E005                 <1>       shl    al, 5 ; *32
00000180 01C7                   <1>       add    di, ax
00000182 B80090                 <1>       mov    ax, DPT_SEGM
00000185 8EC0                   <1>       mov    es, ax
                                <1>       ;
00000187 88E8                   <1>       mov    al, ch ; max. cylinder number (bits 0-7)
00000189 88CC                   <1>       mov    ah, cl
0000018B C0EC06                 <1>       shr    ah, 6  ; max. cylinder number (bits 8-9)
0000018E 40                     <1>       inc    ax     ; logical cylinders (limit 1024)
0000018F AB                     <1>       stosw
00000190 88F0                   <1>       mov    al, dh ; max. head number
00000192 FEC0                   <1>       inc    al
00000194 AA                     <1>       stosb         ; logical heads (limits 256)
00000195 B0A0                   <1>       mov    al, 0A0h ; Indicates translated table
00000197 AA                     <1>       stosb
00000198 8A440C                 <1>       mov    al, [si+12]
0000019B AA                     <1>       stosb         ; physical sectors per track
0000019C 31C0                   <1>       xor    ax, ax
                                <1>       ;dec   ax      ; 02/01/2015
0000019E AB                     <1>       stosw         ; precompensation (obsolete)
                                <1>       ;xor   al, al  ; 02/01/2015
0000019F AA                     <1>       stosb         ; reserved
000001A0 B008                   <1>       mov    al, 8   ; drive control byte
                                <1>                      ; (do not disable retries,
                                <1>                     ; more than 8 heads)
000001A2 AA                     <1>       stosb
000001A3 8B4404                 <1>       mov    ax, [si+4]
000001A6 AB                     <1>       stosw         ; physical number of cylinders
                                <1>       ;push  ax      ; 02/01/2015
000001A7 8A4408                 <1>       mov    al, [si+8]
000001AA AA                     <1>       stosb         ; physical num. of heads (limit 16)
000001AB 29C0                   <1>       sub    ax, ax
                                <1>       ;pop   ax      ; 02/01/2015
000001AD AB                     <1>       stosw         ; landing zone (obsolete)
000001AE 88C8                   <1>       mov    al, cl  ; logical sectors per track (limit 63)
```

```
000001B0 243F                   <1>       and    al, 3Fh
000001B2 AA                     <1>       stosb
                                <1>       ;sub   al, al  ; checksum
                                <1>       ;stosb
                                <1>       ;
000001B3 83C61A                 <1>       add    si, 26   ; (BIOS) DPTE address pointer
000001B6 AD                     <1>       lodsw
000001B7 50                     <1>       push   ax      ; (BIOS) DPTE offset
000001B8 AD                     <1>       lodsw
000001B9 50                     <1>       push   ax      ; (BIOS) DPTE segment
                                <1>       ;
                                <1>       ; checksum calculation
000001BA 89FE                   <1>       mov    si, di
000001BC 06                     <1>       push   es
000001BD 1F                     <1>       pop    ds
                                <1>       ;mov   cx, 16
000001BE B90F00                 <1>       mov    cx, 15
000001C1 29CE                   <1>       sub    si, cx
000001C3 30E4                   <1>       xor    ah, ah
                                <1>       ;del   cl
                                <1> L11:
000001C5 AC                     <1>       lodsb
000001C6 00C4                   <1>       add    ah, al
000001C8 E2FB                   <1>       loop   L11
                                <1>       ;
000001CA 88E0                   <1>       mov    al, ah
000001CC F6D8                   <1>       neg    al     ; -x+x = 0
000001CE AA                     <1>       stosb         ; put checksum in byte 15 of the tbl
                                <1>       ;
000001CF 1F                     <1>       pop    ds     ; (BIOS) DPTE segment
000001D0 5E                     <1>       pop    si     ; (BIOS) DPTE offset
                                <1>       ;
                                <1>       ; 23/02/2015
000001D1 57                     <1>       push   di
                                <1>       ; ES:DI points to DPTE (FDPTE) location
                                <1>       ;mov   cx, 8
000001D2 B108                   <1>       mov    cl, 8
000001D4 F3A5                   <1>       rep    movsw
                                <1>       ;
                                <1>       ; 23/02/2015
                                <1>       ; (P)ATA drive and LBA validation
                                <1>       ; (invalidating SATA drives and setting
                                <1>       ; CHS type I/O for old type fixed disks)
000001D6 5B                     <1>       pop    bx
000001D7 8CC8                   <1>       mov    ax, cs
000001D9 8ED8                   <1>       mov    ds, ax
000001DB 268B07                 <1>       mov    ax, [es:bx]
000001DE 3DF001                 <1>       cmp    ax, 1F0h
000001E1 7418                   <1>       je     short L11a
000001E3 3D7001                 <1>       cmp    ax, 170h
000001E6 7413                   <1>       je     short L11a
                                <1>       ; invalidation
                                <1>       ; (because base port address is not 1F0h or 170h)
000001E8 30FF                   <1>       xor    bh, bh
000001EA 88D3                   <1>       mov    bl, dl
000001EC 80EB80                 <1>       sub    bl, 80h
000001EF C687[206B]00           <1>       mov    byte [bx+hd0_type], 0 ; not a valid disk drive !
000001F4 808F[6C6B]F0           <1>         or       byte [bx+drv.status+2], 0F0h ; (failure sign)
000001F9 EB14                   <1>       jmp    short L11b
                                <1> L11a:
                                <1>       ; LBA validation
000001FB 268A4704               <1>       mov    al, [es:bx+4] ; Head register upper nibble
000001FF A840                   <1>       test   al, 40h ; LBA bit (bit 6)
00000201 750C                   <1>       jnz    short L11b ; LBA type I/O is OK! (E0h or F0h)
                                <1>       ; force CHS type I/O for this drive (A0h or B0h)
00000203 28FF                   <1>       sub    bh, bh
00000205 88D3                   <1>       mov    bl, dl
00000207 80EB80                 <1>       sub    bl, 80h ; 26/02/2015
0000020A 80A7[6C6B]FE           <1>         and      byte [bx+drv.status+2], 0FEh ; clear bit 0
                                <1>                           ; bit 0 = LBA ready bit
                                <1>       ; 'diskio' procedure will check this bit !
                                <1> L11b:
0000020F 3A16[1C6B]             <1>       cmp    dl, [last_drv] ; 25/12/2014
00000213 7307                   <1>         jnb      short L13
00000215 E900FF                 <1>         jmp      L10
                                <1> L12:
                                <1>       ; Restore data registers
00000218 8CC8                   <1>       mov    ax, cs
0000021A 8ED8                   <1>       mov    ds, ax
                                <1> L13:
                                <1>       ; 13/12/2014
0000021C 0E                     <1>       push   cs
0000021D 07                     <1>       pop    es
                                <1> L14:
0000021E B411                   <1>       mov    ah, 11h
00000220 CD16                   <1>       int    16h
00000222 7406                   <1>       jz     short L15 ; no keys in keyboard buffer
00000224 B010                   <1>       mov    al, 10h
00000226 CD16                   <1>       int    16h
00000228 EBF4                   <1>       jmp    short L14
                                <1> L15:
                                <1> ; //////
                                <1>       ; 24/11/2014
                                <1>       ; 19/11/2014
                                <1>       ; 14/11/2014
                                <1>       ; Temporary code for disk searching code check
                                <1>       ;
                                <1>       ; This code will show existing (usable) drives and also
                                <1>       ; will show EDD interface support status for hard disks
                                <1>       ; (If status bit 7 is 1, Identify Device info is ready,
                                <1>       ; no need to get it again in protected mode...)
                                <1>       ;
                                <1>       ; 13/11/2014
0000022A BB0700                 <1>       mov    bx, 7
0000022D B40E                   <1>       mov    ah, 0Eh
```

```
525 0000022F A0[1E6B]           <1>       mov   al, [fd0_type]
526 00000232 20C0               <1>       and   al, al
527 00000234 743D               <1>       jz    short L15a
528 00000236 88C2               <1>       mov   dl, al
529 00000238 B046               <1>       mov   al, 'F'
530 0000023A CD10               <1>       int   10h
531 0000023C B044               <1>       mov   al, 'D'
532 0000023E CD10               <1>       int   10h
533 00000240 B030               <1>       mov   al, '0'
534 00000242 CD10               <1>       int   10h
535 00000244 B020               <1>       mov   al, ' '
536 00000246 CD10               <1>       int   10h
537 00000248 E8B200             <1>       call  L15c
538 0000024B B020               <1>       mov   al, ' '
539 0000024D CD10               <1>       int   10h
540                             <1>       ;
541 0000024F A0[1F6B]           <1>       mov   al, [fd1_type]
542 00000252 20C0               <1>       and   al, al
543 00000254 741D               <1>       jz    short L15a
544 00000256 88C2               <1>       mov   dl, al
545 00000258 B046               <1>       mov   al, 'F'
546 0000025A CD10               <1>       int   10h
547 0000025C B044               <1>       mov   al, 'D'
548 0000025E CD10               <1>       int   10h
549 00000260 B031               <1>       mov   al, '1'
550 00000262 CD10               <1>       int   10h
551 00000264 B020               <1>       mov   al, ' '
552 00000266 CD10               <1>       int   10h
553 00000268 E89200             <1>       call  L15c
554 0000026B B020               <1>       mov   al, ' '
555 0000026D CD10               <1>       int   10h
556 0000026F B020               <1>       mov   al, ' '
557 00000271 CD10               <1>       int   10h
558                             <1> L15a:
559 00000273 A0[206B]           <1>       mov   al, [hd0_type]
560 00000276 20C0               <1>       and   al, al
561 00000278 7479               <1>       jz    short L15b
562 0000027A 88C2               <1>       mov   dl, al
563 0000027C B048               <1>       mov   al, 'H'
564 0000027E CD10               <1>       int   10h
565 00000280 B044               <1>       mov   al, 'D'
566 00000282 CD10               <1>       int   10h
567 00000284 B030               <1>       mov   al, '0'
568 00000286 CD10               <1>       int   10h
569 00000288 B020               <1>       mov   al, ' '
570 0000028A CD10               <1>       int   10h
571 0000028C E86E00             <1>       call  L15c
572 0000028F B020               <1>       mov   al, ' '
573 00000291 CD10               <1>       int   10h
574                             <1>       ;
575 00000293 A0[216B]           <1>       mov   al, [hd1_type]
576 00000296 20C0               <1>       and   al, al
577 00000298 7459               <1>       jz    short L15b
578 0000029A 88C2               <1>       mov   dl, al
579 0000029C B048               <1>       mov   al, 'H'
580 0000029E CD10               <1>       int   10h
581 000002A0 B044               <1>       mov   al, 'D'
582 000002A2 CD10               <1>       int   10h
583 000002A4 B031               <1>       mov   al, '1'
584 000002A6 CD10               <1>       int   10h
585 000002A8 B020               <1>       mov   al, ' '
586 000002AA CD10               <1>       int   10h
587 000002AC E84E00             <1>       call  L15c
588 000002AF B020               <1>       mov   al, ' '
589 000002B1 CD10               <1>       int   10h
590                             <1>       ;
591 000002B3 A0[226B]           <1>       mov   al, [hd2_type]
592 000002B6 20C0               <1>       and   al, al
593 000002B8 7439               <1>       jz    short L15b
594 000002BA 88C2               <1>       mov   dl, al
595 000002BC B048               <1>       mov   al, 'H'
596 000002BE CD10               <1>       int   10h
597 000002C0 B044               <1>       mov   al, 'D'
598 000002C2 CD10               <1>       int   10h
599 000002C4 B032               <1>       mov   al, '2'
600 000002C6 CD10               <1>       int   10h
601 000002C8 B020               <1>       mov   al, ' '
602 000002CA CD10               <1>       int   10h
603 000002CC E82E00             <1>       call  L15c
604 000002CF B020               <1>       mov   al, ' '
605 000002D1 CD10               <1>       int   10h
606                             <1>       ;
607 000002D3 A0[236B]           <1>       mov   al, [hd3_type]
608 000002D6 20C0               <1>       and   al, al
609 000002D8 7419               <1>       jz    short L15b
610 000002DA 88C2               <1>       mov   dl, al
611 000002DC B048               <1>       mov   al, 'H'
612 000002DE CD10               <1>       int   10h
613 000002E0 B044               <1>       mov   al, 'D'
614 000002E2 CD10               <1>       int   10h
615 000002E4 B033               <1>       mov   al, '3'
616 000002E6 CD10               <1>       int   10h
617 000002E8 B020               <1>       mov   al, ' '
618 000002EA CD10               <1>       int   10h
619 000002EC E80E00             <1>       call  L15c
620 000002EF B020               <1>       mov   al, ' '
621 000002F1 CD10               <1>       int   10h
622                             <1>       ;
623                             <1> L15b:
624 000002F3 B00D               <1>       mov   al, 0Dh
625 000002F5 CD10               <1>       int   10h
626 000002F7 B00A               <1>       mov   al, 0Ah
627 000002F9 CD10               <1>       int   10h
628                             <1>       ;;xor  ah, ah
629                             <1>       ;;int  16h
```

```
630                                     <1>       ;
631 000002FB EB77                       <1>         jmp     L16  ; jmp short L16
632                                     <1>         ;
633                                     <1> L15c:
634 000002FD 88D6                       <1>       mov   dh, dl
635 000002FF C0EE04                     <1>       shr   dh, 4
636 00000302 80C630                     <1>       add   dh, 30h
637 00000305 80E20F                     <1>       and   dl, 15
638 00000308 80C230                     <1>       add   dl, 30h
639 0000030B 88F0                       <1>       mov   al, dh
640 0000030D CD10                       <1>       int   10h
641 0000030F 88D0                       <1>       mov   al, dl
642 00000311 CD10                       <1>       int   10h
643 00000313 C3                         <1>       retn
644                                     <1>       ;
645                                     <1>       ; end of temporary code for disk searching code check
646                                     <1>
647                                     <1> ; //////
648                                     <1>
649                                     <1> set_disk_parms:
650                                     <1>       ; 04/02/2016 (ebx -> bx)
651                                     <1>       ; 10/07/2015
652                                     <1>       ; 14/01/2015
653                                     <1>       ;push  bx
654 00000314 28FF                       <1>       sub   bh, bh
655 00000316 8A1E[1B6B]                 <1>       mov   bl, [drv]
656 0000031A 80FB80                     <1>       cmp   bl, 80h
657 0000031D 7203                       <1>       jb    short sdp0
658 0000031F 80EB7E                     <1>       sub   bl, 7Eh
659                                     <1> sdp0:
660 00000322 81C3[6A6B]                 <1>       add   bx, drv.status
661 00000326 C60780                     <1>       mov   byte [bx], 80h ; 'Present' flag
662                                     <1>       ;
663 00000329 88E8                       <1>       mov   al, ch ; last cylinder (bits 0-7)
664 0000032B 88CC                       <1>       mov   ah, cl ;
665 0000032D C0EC06                     <1>       shr   ah, 6  ; last cylinder (bits 8-9)
666 00000330 81EB[6A6B]                 <1>       sub   bx, drv.status
667 00000334 D0E3                       <1>       shl   bl, 1
668 00000336 81C3[246B]                 <1>       add   bx, drv.cylinders
669 0000033A 40                         <1>       inc   ax  ; convert max. cyl number to cyl count
670 0000033B 8907                       <1>       mov   [bx], ax
671 0000033D 50                         <1>       push  ax ; ** cylinders
672 0000033E 81EB[246B]                 <1>       sub   bx, drv.cylinders
673 00000342 81C3[326B]                 <1>       add   bx, drv.heads
674 00000346 30E4                       <1>       xor   ah, ah
675 00000348 88F0                       <1>       mov   al, dh ; heads
676 0000034A 40                         <1>       inc   ax
677 0000034B 8907                       <1>       mov   [bx], ax
678 0000034D 81EB[326B]                 <1>         sub     bx, drv.heads
679 00000351 81C3[406B]                 <1>         add     bx, drv.spt
680 00000355 30ED                       <1>       xor   ch, ch
681 00000357 80E13F                     <1>       and   cl, 3Fh      ; sectors (bits 0-6)
682 0000035A 890F                       <1>       mov   [bx], cx
683 0000035C 81EB[406B]                 <1>         sub     bx, drv.spt
684 00000360 D1E3                       <1>       shl   bx, 1
685 00000362 81C3[4E6B]                 <1>       add   bx, drv.size ; disk size (in sectors)
686                                     <1>       ; LBA size = cylinders * heads * secpertrack
687 00000366 F7E1                       <1>       mul   cx
688 00000368 89C2                       <1>       mov   dx, ax ; heads*spt
689 0000036A 58                         <1>       pop   ax ; ** cylinders
690 0000036B 48                         <1>       dec   ax ; 1 cylinder reserved (!?)
691 0000036C F7E2                       <1>       mul   dx ; cylinders * (heads*spt)
692 0000036E 8907                       <1>       mov   [bx], ax
693 00000370 895702                     <1>       mov   [bx+2], dx
694                                     <1>       ;
695                                     <1>       ;pop  bx
696 00000373 C3                         <1>       retn
697                                     <1>
698                                     <1> ;align 2
699                                     <1>
700                                     <1> ;cylinders :  dw 0, 0, 0, 0, 0, 0
701                                     <1> ;heads     :  dw 0, 0, 0, 0, 0, 0
702                                     <1> ;spt    :  dw 0, 0, 0, 0, 0, 0
703                                     <1> ;disk_size :  dd 0, 0, 0, 0, 0, 0
704                                     <1>
705                                     <1> ;last_drv:
706                                     <1> ;     db  0
707                                     <1> ;drv_status:
708                                     <1> ;     db  0,0,0,0,0,0
709                                     <1> ;     db 0
710                                     <1>
711                                     <1>
712                                     <1> ; End Of DISK I/O SYSTEM STRUCTURE INITIALIZATION /// 06/02/2015
713                                     <1>
714                                     <1> L16:
715
716                                             ; 10/11/2014
717 00000374 FA                                 cli    ; Disable interrupts (clear interrupt flag)
718                                             ; Reset Interrupt MASK Registers (Master&Slave)
719                                             ;mov   al, 0FFh     ; mask off all interrupts
720                                             ;out   21h, al             ; on master PIC (8259)
721                                             ;jmp   $+2  ; (delay)
722                                             ;out   0A1h, al     ; on slave PIC (8259)
723                                             ;
724                                             ; Disable NMI
725 00000375 B080                               mov    al, 80h
726 00000377 E670                               out    70h, al             ; set bit 7 to 1 for disabling NMI
727                                             ;23/02/2015
728 00000379 90                                 nop                ;
729                                             ;in    al, 71h             ; read in 71h just after writing out to 70h
730                                                                ; for preventing unknown state (!?)
731                                             ;
732                                             ; 20/08/2014
733                                             ; Moving the kernel 64 KB back (to physical address 0)
734                                             ; DS = CS = 1000h
```

```
                                        ; 05/11/2014
0000037A 31C0                           xor    ax, ax
0000037C 8EC0                           mov    es, ax ; ES = 0
                                        ;
0000037E B90040                         mov    cx, (KEND - KLOAD)/4
00000381 31F6                           xor    si, si
00000383 31FF                           xor    di, di
00000385 F366A5                         rep    movsd
                                        ;
00000388 06                             push   es ; 0
00000389 68[8D03]                       push   L17
0000038C CB                             retf
                                        ;
                                  L17:
                                        ; Turn off the floppy drive motor
0000038D BAF203                           mov     dx, 3F2h
00000390 EE                               out     dx, al ; 0 ; 31/12/2013

                                        ; Enable access to memory above one megabyte
                                  L18:
00000391 E464                           in     al, 64h
00000393 A802                           test   al, 2
00000395 75FA                             jnz     short L18
00000397 B0D1                           mov    al, 0D1h     ; Write output port
00000399 E664                           out    64h, al
                                  L19:
0000039B E464                           in     al, 64h
0000039D A802                           test   al, 2
0000039F 75FA                             jnz     short L19
000003A1 B0DF                           mov    al, 0DFh     ; Enable A20 line
000003A3 E660                           out    60h, al
                                  ;L20:
                                        ;
                                        ; Load global descriptor table register

                                          ;mov     ax, cs
                                          ;mov     ds, ax

000003A5 2E0F0116[4068]                   lgdt    [cs:gdtd]

000003AB 0F20C0                           mov     eax, cr0
                                        ; or   eax, 1
000003AE 40                             inc     ax
000003AF 0F22C0                         mov     cr0, eax

                                        ; Jump to 32 bit code

000003B2 66                             db 66h                  ; Prefix for 32-bit
000003B3 EA                             db 0EAh             ; Opcode for far jump
000003B4 [BA030000]                     dd StartPM          ; Offset to start, 32-bit
                                                            ; (1000h:StartPM = StartPM + 10000h)
000003B8 0800                           dw KCODE            ; This is the selector for CODE32_DESCRIPTOR,
                                                            ; assuming that StartPM resides in code32

                                  [BITS 32]

                                  StartPM:
                                        ; Kernel Base Address = 0 ; 30/12/2013
000003BA 66B81000                       mov ax, KDATA           ; Save data segment identifier
000003BE 8ED8                             mov ds, ax              ; Move a valid data segment into DS register
000003C0 8EC0                                mov es, ax              ; Move data segment into ES register
000003C2 8EE0                                mov fs, ax              ; Move data segment into FS register
000003C4 8EE8                                mov gs, ax              ; Move data segment into GS register
000003C6 8ED0                             mov ss, ax              ; Move data segment into SS register
000003C8 BC00000900                       mov esp, 90000h         ; Move the stack pointer to 090000h

                                  clear_bss: ; Clear uninitialized data area
                                        ; 11/03/2015
000003CD 31C0                           xor  eax, eax ; 0
000003CF B9CD050000                     mov  ecx, (bss_end - bss_start)/4
                                        ;shr  ecx, 2 ; bss section is already aligned for double words
000003D4 BF[006E0000]                   mov  edi, bss_start
000003D9 F3AB                           rep  stosd

                                  memory_init:
                                        ; Initialize memory allocation table and page tables
                                        ; 16/11/2014
                                        ; 15/11/2014
                                        ; 07/11/2014
                                        ; 06/11/2014
                                        ; 05/11/2014
                                        ; 04/11/2014
                                        ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
                                        ;
                                  ;     xor    eax, eax
                                  ;     xor    ecx, ecx
000003DB B108                           mov    cl, 8
000003DD BF00001000                     mov    edi, MEM_ALLOC_TBL
000003E2 F3AB                           rep    stosd           ; clear Memory Allocation Table
                                                               ; for the first 1 MB memory
                                        ;
000003E4 668B0D[F06D0000]               mov    cx, [mem_1m_1k]        ; Number of contiguous KB between
                                                               ; 1 and 16 MB, max. 3C00h = 15 MB.
000003EB 66C1E902                       shr    cx, 2           ; convert 1 KB count to 4 KB count
000003EF 890D[70700000]                 mov    [free_pages], ecx
000003F5 668B15[F26D0000]               mov    dx, [mem_16m_64k]  ; Number of contiguous 64 KB blocks
                                                               ; between 16 MB and 4 GB.
000003FC 6609D2                         or     dx, dx
000003FF 7413                           jz     short mi_0
                                        ;
00000401 6689D0                         mov    ax, dx
00000404 C1E004                         shl    eax, 4          ; 64 KB -> 4 KB (page count)
00000407 0105[70700000]                 add    [free_pages], eax
0000040D 0500100000                     add    eax, 4096       ; 16 MB = 4096 pages
00000412 EB07                           jmp    short mi_1
```

```
840                                 mi_0:
841 00000414 6689C8                        mov   ax, cx
842 00000417 66050001                      add   ax, 256               ; add 256 pages for the first 1 MB
843                                 mi_1:
844 0000041B A3[6C700000]                  mov   [memory_size], eax ; Total available memory in pages
845                                                                 ; 1 alloc. tbl. bit = 1 memory page
846                                                                 ; 32 allocation bits = 32 mem. pages
847                                        ;
848 00000420 05FF7F0000                    add   eax, 32767      ; 32768 memory pages per 1 M.A.T. page
849 00000425 C1E80F                        shr   eax, 15               ; ((32768 * x) + y) pages (y < 32768)
850                                                                 ;  --> x + 1 M.A.T. pages, if y > 0
851                                                                 ;  --> x M.A.T. pages, if y = 0
852 00000428 66A3[80700000]                mov   [mat_size], ax        ; Memory Alloc. Table Size in pages
853 0000042E C1E00C                        shl   eax, 12               ; 1 M.A.T. page = 4096 bytes
854                                        ;                     ; Max. 32 M.A.T. pages (4 GB memory)
855 00000431 89C3                          mov   ebx, eax        ; M.A.T. size in bytes
856                                        ; Set/Calculate Kernel's Page Directory Address
857 00000433 81C300001000                  add   ebx, MEM_ALLOC_TBL
858 00000439 891D[68700000]                mov   [k_page_dir], ebx  ; Kernel's Page Directory address
859                                                                 ; just after the last M.A.T. page
860                                        ;
861 0000043F 83E804                        sub   eax, 4          ; convert M.A.T. size to offset value
862 00000442 A3[78700000]                  mov   [last_page], eax   ; last page ofset in the M.A.T.
863                                        ;                     ; (allocation status search must be
864                                                                 ; stopped after here)
865 00000447 31C0                          xor   eax, eax
866 00000449 48                            dec   eax             ; FFFFFFFFh (set all bits to 1)
867 0000044A 6651                          push  cx
868 0000044C C1E905                        shr   ecx, 5          ; convert 1 - 16 MB page count to
869                                                                 ; count of 32 allocation bits
870 0000044F F3AB                          rep   stosd
871 00000451 6659                          pop   cx
872 00000453 40                            inc   eax             ; 0
873 00000454 80E11F                        and   cl, 31          ; remain bits
874 00000457 7412                          jz    short mi_4
875 00000459 8907                          mov   [edi], eax      ; reset
876                                 mi_2:
877 0000045B 0FAB07                        bts   [edi], eax      ; 06/11/2014
878 0000045E FEC9                          dec   cl
879 00000460 7404                          jz    short mi_3
880 00000462 FEC0                          inc   al
881 00000464 EBF5                          jmp   short mi_2
882                                 mi_3:
883 00000466 28C0                          sub   al, al          ; 0
884 00000468 83C704                        add   edi, 4          ; 15/11/2014
885                                 mi_4:
886 0000046B 6609D2                        or    dx, dx          ; check 16M to 4G memory space
887 0000046E 7421                          jz    short mi_6      ; max. 16 MB memory, no more...
888                                        ;
889 00000470 B900021000                    mov   ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
890                                        ;
891 00000475 29F9                          sub   ecx, edi        ; displacement (to end of 16 MB)
892 00000477 7406                          jz    short mi_5      ; jump if EDI points to
893                                                                 ;         end of first 16 MB
894 00000479 D1E9                          shr   ecx, 1          ; convert to dword count
895 0000047B D1E9                          shr   ecx, 1          ; (shift 2 bits right)
896 0000047D F3AB                          rep   stosd           ; reset all bits for reserved pages
897                                                                 ; (memory hole under 16 MB)
898                                 mi_5:
899 0000047F 6689D1                        mov   cx, dx          ; count of 64 KB memory blocks
900 00000482 D1E9                          shr   ecx, 1          ; 1 alloc. dword per 128 KB memory
901 00000484 9C                            pushf                 ; 16/11/2014
902 00000485 48                            dec   eax             ; FFFFFFFFh (set all bits to 1)
903 00000486 F3AB                          rep   stosd
904 00000488 40                            inc   eax             ; 0
905 00000489 9D                            popf                  ; 16/11/2014
906 0000048A 7305                          jnc   short mi_6
907 0000048C 6648                          dec   ax              ; eax = 0000FFFFh
908 0000048E AB                            stosd
909 0000048F 6640                          inc   ax              ; 0
910                                 mi_6:
911 00000491 39DF                          cmp   edi, ebx        ; check if EDI points to
912 00000493 730A                          jnb   short mi_7      ; end of memory allocation table
913                                        ;                     ; (>= MEM_ALLOC_TBL + 4906)
914 00000495 89D9                          mov   ecx, ebx        ; end of memory allocation table
915 00000497 29F9                          sub   ecx, edi        ; convert displacement/offset
916 00000499 D1E9                          shr   ecx, 1          ; to dword count
917 0000049B D1E9                          shr   ecx, 1          ; (shift 2 bits right)
918 0000049D F3AB                          rep   stosd           ; reset all remain M.A.T. bits
919                                 mi_7:
920                                        ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
921 0000049F BA00001000                    mov   edx, MEM_ALLOC_TBL
922                                        ;sub  ebx, edx        ; Mem. Alloc. Tbl. size in bytes
923                                        ;shr  ebx, 12               ; Mem. Alloc. Tbl. size in pages
924 000004A4 668B0D[80700000]              mov   cx, [mat_size]        ; Mem. Alloc. Tbl. size in pages
925 000004AB 89D7                          mov   edi, edx
926 000004AD C1EF0F                        shr   edi, 15               ; convert M.A.T. address to
927                                                                 ; byte offset in M.A.T.
928                                                                 ; (1 M.A.T. byte points to
929                                                                 ;         32768 bytes)
930                                                                 ; Note: MEM_ALLOC_TBL address
931                                                                 ; must be aligned on 128 KB
932                                                                 ; boundary!
933 000004B0 01D7                          add   edi, edx        ; points to M.A.T.'s itself
934                                        ; eax = 0
935 000004B2 290D[70700000]                sub   [free_pages], ecx ; 07/11/2014
936                                 mi_8:
937 000004B8 0FB307                        btr   [edi], eax      ; clear bit 0 to bit x (1 to 31)
938                                        ;dec  bl
939 000004BB FEC9                          dec   cl
940 000004BD 7404                          jz    short mi_9
941 000004BF FEC0                          inc   al
942 000004C1 EBF5                          jmp   short mi_8
943                                 mi_9:
944                                        ;
```

```
                                        ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
                                        ;            (allocate pages for system page tables)
                                        ; edx = MEM_ALLOC_TBL
000004C3 8B0D[6C700000]                 mov    ecx, [memory_size] ; memory size in pages (PTEs)
000004C9 81C1FF030000                   add    ecx, 1023     ; round up (1024 PTEs per table)
000004CF C1E90A                         shr    ecx, 10             ; convert memory page count to
                                                              ; page table count (PDE count)
                                        ;
000004D2 51                             push   ecx           ; (**) PDE count (<= 1024)
                                        ;
000004D3 41                             inc    ecx           ; +1 for kernel page directory
                                        ;
000004D4 290D[70700000]                 sub    [free_pages], ecx ; 07/11/2014
                                        ;
000004DA 8B35[68700000]                 mov    esi, [k_page_dir] ; Kernel's Page Directory address
000004E0 C1EE0C                         shr    esi, 12             ; convert to page number
                                    mi_10:
000004E3 89F0                           mov    eax, esi      ; allocation bit offset
000004E5 89C3                           mov    ebx, eax
000004E7 C1EB03                         shr    ebx, 3        ; convert to alloc. byte offset
000004EA 80E3FC                         and    bl,  0FCh     ; clear bit 0 and bit 1
                                                             ;   to align on dword boundary
000004ED 83E01F                         and    eax, 31             ; set allocation bit position
                                                             ;  (bit 0 to bit 31)
                                        ;
000004F0 01D3                           add    ebx, edx      ; offset in M.A.T. + M.A.T. address
                                        ;
000004F2 0FB303                         btr    [ebx], eax    ; reset relevant bit (0 to 31)
                                        ;
000004F5 46                             inc    esi           ; next page table
000004F6 E2EB                           loop   mi_10         ; allocate next kernel page table
                                                              ; (ecx = page table count + 1)
                                        ;
000004F8 59                             pop    ecx           ; (**) PDE count (= pg. tbl. count)
                                        ;
                                        ; Initialize Kernel Page Directory and Kernel Page Tables
                                        ;
                                        ; Initialize Kernel's Page Directory
000004F9 8B3D[68700000]                 mov    edi, [k_page_dir]
000004FF 89F8                           mov    eax, edi
00000501 0C03                           or     al, PDE_A_PRESENT + PDE_A_WRITE
                                                             ; supervisor + read&write + present
00000503 89CA                           mov    edx, ecx      ; (**) PDE count (= pg. tbl. count)
                                    mi_11:
00000505 0500100000                     add    eax, 4096     ; Add page size (PGSZ)
                                                              ; EAX points to next page table
0000050A AB                             stosd
0000050B E2F8                           loop   mi_11
0000050D 29C0                           sub    eax, eax      ; Empty PDE
0000050F 66B90004                       mov    cx, 1024      ; Entry count (PGSZ/4)
00000513 29D1                           sub    ecx, edx
00000515 7402                           jz     short mi_12
00000517 F3AB                           rep    stosd         ; clear remain (empty) PDEs
                                        ;
                                        ; Initialization of Kernel's Page Directory is OK, here.
                                    mi_12:
                                        ; Initialize Kernel's Page Tables
                                        ;
                                        ; (EDI points to address of page table 0)
                                        ; eax = 0
00000519 8B0D[6C700000]                 mov    ecx, [memory_size] ; memory size in pages
0000051F 89CA                           mov    edx, ecx      ; (***)
00000521 B003                           mov    al, PTE_A_PRESENT + PTE_A_WRITE
                                                             ; supervisor + read&write + present
                                    mi_13:
00000523 AB                             stosd
00000524 0500100000                     add    eax, 4096
00000529 E2F8                           loop   mi_13
0000052B 6681E2FF03                     and    dx, 1023      ; (***)
00000530 740B                           jz     short mi_14
00000532 66B90004                       mov    cx, 1024
00000536 6629D1                         sub    cx, dx        ; from dx (<= 1023) to 1024
00000539 31C0                           xor    eax, eax
0000053B F3AB                           rep    stosd         ; clear remain (empty) PTEs
                                                             ; of the last page table
                                    mi_14:
                                        ;  Initialization of Kernel's Page Tables is OK, here.
                                        ;
0000053D 89F8                           mov    eax, edi      ; end of the last page table page
                                                              ; (beginging of user space pages)
0000053F C1E80F                         shr    eax, 15             ; convert to M.A.T. byte offset
00000542 24FC                           and    al, 0FCh      ; clear bit 0 and bit 1 for
                                                             ; aligning on dword boundary
00000544 A3[7C700000]                   mov    [first_page], eax
00000549 A3[74700000]                   mov    [next_page], eax ; The first free page pointer
                                                              ; for user programs
                                                              ; (Offset in Mem. Alloc. Tbl.)
                                        ;
                                        ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
                                        ;

                                        ; Enable paging
                                        ;
0000054E A1[68700000]                     mov     eax, [k_page_dir]
00000553 0F22D8                         mov    cr3, eax
00000556 0F20C0                         mov    eax, cr0
00000559 0D00000080                     or     eax, 80000000h      ; set paging bit (bit 31)
0000055E 0F22C0                         mov    cr0, eax
                                          ;jmp    KCODE:StartPMP

00000561 EA                             db 0EAh              ; Opcode for far jump
00000562 [68050000]                       dd StartPMP              ; 32 bit offset
00000566 0800                           dw KCODE             ; kernel code segment descriptor
```

```
1050
1051
1052                                    StartPMP:
1053                                          ; 06/11//2014
1054                                          ; Clear video page 0
1055                                          ;
1056                                          ; Temporary Code
1057                                          ;
1058 00000568 B9E8030000                     mov    ecx, 80*25/2
1059 0000056D BF00800B00                     mov    edi, 0B8000h
1060 00000572 31C0                           xor    eax, eax     ; black background, black fore color
1061 00000574 F3AB                           rep    stosd
1062
1063                                          ; 19/08/2014
1064                                          ; Kernel Base Address = 0
1065                                          ; It is mapped to (physically) 0 in the page table.
1066                                          ; So, here is exactly 'StartPMP' address.
1067                                          ;
1068                                          ;;mov  ah, 4Eh      ; Red background, yellow forecolor
1069                                          ;;mov  esi, msgPM
1070                                          ;; 14/08/2015 (kernel version message will appear
1071                                          ;;            when protected mode and paging is enabled)
1072 00000576 B40B                           mov    ah, 0Bh ; Black background, light cyan forecolor
1073 00000578 BE[806B0000]                   mov    esi, msgKVER
1074 0000057D BF00800B00                     mov    edi, 0B8000h ; 27/08/2014
1075                                          ; 20/08/2014
1076 00000582 E896010000                     call   printk
1077
1078                                          ; 'UNIX v7/x86' source code by Robert Nordier (1999)
1079                                          ; // Set IRQ offsets
1080                                          ;
1081                                          ;  Linux (v0.12) source code by Linus Torvalds (1991)
1082                                          ;
1083                                                                ;; ICW1
1084 00000587 B011                           mov    al, 11h                ; Initialization sequence
1085 00000589 E620                           out    20h, al                ;      8259A-1
1086                                          ; jmp  $+2
1087 0000058B E6A0                           out    0A0h, al           ;      8259A-2
1088                                                                ;; ICW2
1089 0000058D B020                           mov    al, 20h                ; Start of hardware ints (20h)
1090 0000058F E621                           out    21h, al                ;      for 8259A-1
1091                                          ; jmp  $+2
1092 00000591 B028                           mov    al, 28h                ; Start of hardware ints (28h)
1093 00000593 E6A1                           out    0A1h, al           ;      for 8259A-2
1094                                                                ;
1095 00000595 B004                           mov    al, 04h                ;; ICW3
1096 00000597 E621                           out    21h, al                ;      IRQ2 of 8259A-1 (master)
1097                                          ; jmp  $+2
1098 00000599 B002                           mov    al, 02h                ;      is 8259A-2 (slave)
1099 0000059B E6A1                           out    0A1h, al           ;
1100                                                                ;; ICW4
1101 0000059D B001                           mov    al, 01h                ;
1102 0000059F E621                           out    21h, al                ;      8086 mode, normal EOI
1103                                          ; jmp  $+2
1104 000005A1 E6A1                           out    0A1h, al           ;      for both chips.
1105
1106                                          ;mov   al, 0FFh     ; mask off all interrupts for now
1107                                          ;out   21h, al
1108                                          ;; jmp       $+2
1109                                          ;out   0A1h, al
1110
1111                                          ; 02/04/2015
1112                                          ; 26/03/2015 System call (INT 30h) modification
1113                                          ;  DPL = 3 (Interrupt service routine can be called from user mode)

1114                                          ;
1115                                          ;; Linux (v0.12) source code by Linus Torvalds (1991)
1116                                          ;  setup_idt:
1117                                          ;
1118                                            ;; 16/02/2015
1119                                          ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
1120                                          ; 21/08/2014 (timer_int)
1121 000005A3 BE[4C680000]                   mov    esi, ilist
1122 000005A8 8D3D[006E0000]                 lea    edi, [idt]
1123                                          ; 26/03/2015
1124 000005AE B930000000                     mov    ecx, 48             ; 48 hardware interrupts (INT 0 to INT 2Fh)
1125                                          ; 02/04/2015
1126 000005B3 BB00000800                     mov    ebx,  80000h
1127                                    rp_sidt1:
1128 000005B8 AD                             lodsd
1129 000005B9 89C2                           mov    edx, eax
1130 000005BB 66BA008E                       mov    dx, 8E00h
1131 000005BF 6689C3                         mov    bx, ax
1132 000005C2 89D8                           mov    eax, ebx     ; /* selector = 0x0008 = cs */
1133                                                                    ; /* interrupt gate - dpl=0, present */
1134 000005C4 AB                             stosd  ; selector & offset bits 0-15
1135 000005C5 89D0                           mov    eax, edx
1136 000005C7 AB                             stosd  ; attributes & offset bits 16-23
1137 000005C8 E2EE                           loop   rp_sidt1
1138 000005CA B110                           mov    cl, 16        ; 16 software interrupts (INT 30h to INT 3Fh)
1139                                    rp_sidt2:
1140 000005CC AD                             lodsd
1141 000005CD 21C0                           and    eax, eax
1142 000005CF 7413                           jz     short rp_sidt3
1143 000005D1 89C2                           mov    edx, eax
1144 000005D3 66BA00EE                       mov    dx, 0EE00h   ; P=1b/DPL=11b/01110b
1145 000005D7 6689C3                         mov    bx, ax
1146 000005DA 89D8                           mov    eax, ebx     ; selector & offset bits 0-15
1147 000005DC AB                             stosd
1148 000005DD 89D0                           mov    eax, edx
1149 000005DF AB                             stosd
1150 000005E0 E2EA                           loop   rp_sidt2
1151 000005E2 EB16                           jmp    short sidt_OK
1152                                    rp_sidt3:
1153 000005E4 B8[470A0000]                   mov    eax, ignore_int
```

```
1154 000005E9 89C2                      mov   edx, eax
1155 000005EB 66BA00EE                  mov   dx, 0EE00h   ; P=1b/DPL=11b/01110b
1156 000005EF 6689C3                    mov   bx, ax
1157 000005F2 89D8                      mov   eax, ebx     ; selector & offset bits 0-15
1158                                rp_sidt4:
1159 000005F4 AB                        stosd
1160 000005F5 92                        xchg  eax, edx
1161 000005F6 AB                        stosd
1162 000005F7 92                        xchg  edx, eax
1163 000005F8 E2FA                      loop  rp_sidt4
1164                                sidt_OK:
1165 000005FA 0F011D[46680000]          lidt  [idtd]
1166                                    ;
1167                                    ; TSS descriptor setup ; 24/03/2015
1168 00000601 B8[00700000]              mov   eax, task_state_segment
1169 00000606 66A3[3A680000]            mov   [gdt_tss0], ax
1170 0000060C C1C010                    rol   eax, 16
1171 0000060F A2[3C680000]              mov   [gdt_tss1], al
1172 00000614 8825[3F680000]            mov   [gdt_tss2], ah
1173 0000061A 66C705[66700000]68-       mov   word [tss.IOPB], tss_end - task_state_segment
1174 00000622 00
1175                                    ;
1176                                    ; IO Map Base address (When this address points
1177                                    ; to end of the TSS, CPU does not use IO port
1178                                    ; permission bit map for RING 3 IO permissions,
1179                                    ; access to any IO ports in ring 3 will be forbidden.)
1180                                    ;
1181                                    ;mov  [tss.esp0], esp ; TSS offset 4
1182                                    ;mov  word [tss.ss0], KDATA ; TSS offset 8 (SS)
1183 00000623 66B82800                  mov   ax, TSS  ; It is needed when an interrupt
1184                                                   ; occurs (or a system call -software INT- is requested)
1185                                                   ; while cpu running in ring 3 (in user mode).
1186                                                   ; (Kernel stack pointer and segment will be loaded
1187                                                   ; from offset 4 and 8 of the TSS, by the CPU.)
1188 00000627 0F00D8                      ltr   ax  ; Load task register
1189                                    ;
1190                                esp0_set0:
1191                                    ; 30/07/2015
1192 0000062A 8B0D[6C700000]            mov   ecx, [memory_size] ; memory size in pages
1193 00000630 C1E10C                    shl   ecx, 12 ; convert page count to byte count
1194 00000633 81F900004000              cmp   ecx, CORE ; beginning of user's memory space (400000h)
1195                                                    ; (kernel mode virtual address)
1196 00000639 7605                      jna   short esp0_set1
1197                                    ;
1198                                    ; If available memory > CORE (end of the 1st 4 MB)
1199                                    ; set stack pointer to CORE
1200                                    ;(Because, PDE 0 is reserved for kernel space in user's page directory)
1201                                    ;(PDE 0 points to page table of the 1st 4 MB virtual address space)
1202 0000063B B900004000                mov   ecx, CORE
1203                                esp0_set1:
1204 00000640 89CC                      mov   esp, ecx ; top of kernel stack (**tss.esp0**)
1205                                esp0_set_ok:
1206                                    ; 30/07/2015 (**tss.esp0**)
1207 00000642 8925[04700000]            mov   [tss.esp0], esp
1208 00000648 66C705[08700000]10-         mov     word [tss.ss0], KDATA
1209 00000650 00
1210                                    ; 14/08/2015
1211                                    ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
1212                                    ;
1213                                    ;cli  ; Disable interrupts (for CPU)
1214                                    ;    (CPU will not handle hardware interrupts, except NMI!)
1215                                    ;
1216 00000651 30C0                      xor   al, al       ; Enable all hardware interrupts!
1217 00000653 E621                      out   21h, al            ; (IBM PC-AT compatibility)
1218 00000655 EB00                      jmp   $+2          ; (All conventional PC-AT hardware
1219 00000657 E6A1                      out   0A1h, al     ;  interrupts will be in use.)
1220                                                       ; (Even if related hardware component
1221                                                       ;  does not exist!)
1222                                    ; Enable NMI
1223 00000659 B07F                      mov   al, 7Fh            ; Clear bit 7 to enable NMI (again)
1224 0000065B E670                      out   70h, al
1225                                    ; 23/02/2015
1226 0000065D 90                        nop
1227 0000065E E471                      in    al, 71h            ; read in 71h just after writing out to 70h
1228                                                       ; for preventing unknown state (!?)
1229                                    ;
1230                                    ; Only a NMI can occur here... (Before a 'STI' instruction)
1231                                    ;
1232                                    ; 02/09/2014
1233 00000660 6631DB                    xor   bx, bx
1234 00000663 66BA0002                  mov   dx, 0200h    ; Row 2, column 0  ; 07/03/2015
1235 00000667 E8920F0000                call  set_cpos
1236                                    ;
1237                                    ; 06/11/2014
1238                                    ; Temporary Code
1239                                    ;
1240 0000066C E8C2110000                call  memory_info
1241                                    ; 14/08/2015
1242                                    ;call getch ; 28/02/2015
1243                                drv_init:
1244 00000671 FB                        sti   ; Enable Interrupts
1245                                    ; 06/02/2015
1246 00000672 8B15[206B0000]            mov   edx, [hd0_type] ; hd0, hd1, hd2, hd3
1247 00000678 668B1D[1E6B0000]          mov   bx, [fd0_type] ; fd0, fd1
1248                                    ; 22/02/2015
1249 0000067F 6621DB                    and   bx, bx
1250 00000682 751B                      jnz   short di1
1251                                    ;
1252 00000684 09D2                      or    edx, edx
1253 00000686 7529                      jnz   short di2
1254                                    ;
1255                                setup_error:
1256 00000688 BE[A56C0000]              mov   esi, setup_error_msg
1257                                psem:
```

```
1258 0000068D AC                        lodsb
1259 0000068E 08C0                      or      al, al
1260                                    ;jz     short haltx ; 22/02/2015
1261 00000690 7426                      jz      short di3
1262 00000692 56                        push    esi
1263 00000693 31DB                      xor     ebx, ebx ; 0
1264                                                     ; Video page 0 (bl=0)
1265 00000695 B407                      mov     ah, 07h ; Black background,
1266                                                     ; light gray forecolor
1267 00000697 E83E0E0000                call    write_tty
1268 0000069C 5E                        pop     esi
1269 0000069D EBEE                      jmp     short psem
1270
1271                                di1:
1272                                    ; supress 'jmp short T6'
1273                                    ;  (activate fdc motor control code)
1274 0000069F 66C705[9E070000]90-       mov     word [T5], 9090h ; nop
1275 000006A7 90
1276                                    ;
1277                                    ;mov    ax, int_0Eh  ; IRQ 6 handler
1278                                    ;mov    di, 0Eh*4    ; IRQ 6 vector
1279                                    ;stosw
1280                                    ;mov    ax, cs
1281                                    ;stosw
1282                                    ;; 16/02/2015
1283                                      ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
1284                                    ;
1285 000006A8 E8D8200000                CALL    DSKETTE_SETUP; Initialize Floppy Disks
1286                                    ;
1287 000006AD 09D2                      or      edx, edx
1288 000006AF 7407                        jz      short di3
1289                                di2:
1290 000006B1 E814210000                call          DISK_SETUP   ; Initialize Fixed Disks
1291 000006B6 72D0                        jc      short setup_error
1292                                di3:
1293 000006B8 E84A110000                call    setup_rtc_int; 22/05/2015 (dsectrpm.s)
1294                                    ;
1295 000006BD E8A6600000                call    display_disks ; 07/03/2015  (Temporary)
1296                                ;haltx:
1297                                    ; 14/08/2015
1298                                    ;call  getch ; 22/02/2015
1299 000006C2 FB                        sti     ; Enable interrupts (for CPU)
1300                                    ; 14/08/2015
1301 000006C3 B9FFFFFF0F                mov     ecx, 0FFFFFFFh
1302                                md_info_msg_wait:
1303 000006C8 51                        push    ecx
1304 000006C9 B001                      mov     al, 1
1305 000006CB 8A25[96700000]            mov     ah, [ptty] ; active (current) video page
1306 000006D1 E8CE5D0000                call    getc_n
1307 000006D6 59                        pop     ecx
1308 000006D7 7502                      jnz     short md_info_msg_ok
1309 000006D9 E2ED                      loop    md_info_msg_wait
1310                                md_info_msg_ok:
1311                                    ; 30/06/2015
1312 000006DB E801310000                call    sys_init
1313                                    ;
1314                                    ;jmp    cpu_reset ; 22/02/2015
1315                                hang:
1316                                    ; 23/02/2015
1317                                    ;sti                ; Enable interrupts
1318 000006E0 F4                        hlt
1319                                    ;
1320                                    ;nop
1321                                    ;; 03/12/2014
1322                                    ;; 28/08/2014
1323                                    ;mov    ah, 11h
1324                                    ;call  getc
1325                                    ;jz      _c8
1326                                    ;
1327                                    ; 23/02/2015
1328                                    ; 06/02/2015
1329                                    ; 07/09/2014
1330 000006E1 31DB                      xor     ebx, ebx
1331 000006E3 8A1D[96700000]            mov     bl, [ptty]   ; active_page
1332 000006E9 89DE                      mov     esi, ebx
1333 000006EB 66D1E6                    shl     si, 1
1334 000006EE 81C6[98700000]            add     esi, ttychr
1335 000006F4 668B06                    mov     ax, [esi]
1336 000006F7 6621C0                    and     ax, ax
1337                                    ;jz     short _c8
1338 000006FA 74E4                      jz      short hang
1339 000006FC 66C7060000                mov     word [esi], 0
1340 00000701 80FB03                    cmp     bl, 3        ; Video page 3
1341                                    ;jb     short _c8
1342 00000704 72DA                      jb      short hang
1343                                    ;
1344                                    ; 02/09/2014
1345 00000706 B40E                      mov     ah, 0Eh             ; Yellow character
1346                                                                ; on black background
1347                                    ; 07/09/2014
1348                                nxtl:
1349 00000708 6653                      push    bx
1350                                    ;
1351                                    ;xor    bx, bx       ; bl = 0 (video page 0)
1352                                                         ; bh = 0 (video mode)
1353                                                         ; Retro UNIX 386 v1 - Video Mode 0
1354                                                         ; (PC/AT Video Mode 3 - 80x25 Alpha.)
1355 0000070A 6650                      push    ax
1356 0000070C E8C90D0000                call    write_tty
1357 00000711 6658                      pop     ax
1358 00000713 665B                      pop     bx ; 07/09/2014
1359 00000715 3C0D                      cmp     al, 0Dh             ; carriage return (enter)
1360                                    ;jne    short _c8
1361 00000717 75C7                      jne     short hang
1362 00000719 B00A                      mov     al, 0Ah             ; next line
```

```
1363 0000071B EBEB                      jmp    short nxtl
1364
1365                                  ;_c8:
1366                                  ;     ; 25/08/2014
1367                                  ;     cli                         ; Disable interrupts
1368                                  ;     mov    al, [scounter + 1]
1369                                  ;     and    al, al
1370                                  ;     jnz    hang
1371                                  ;     call   rtc_p
1372                                  ;     jmp     hang
1373
1374
1375                                        ; 27/08/2014
1376                                        ; 20/08/2014
1377                                  printk:
1378                                          ;mov    edi, [scr_row]
1379                                  pkl:
1380 0000071D AC                            lodsb
1381 0000071E 08C0                          or     al, al
1382 00000720 7404                          jz     short pkr
1383 00000722 66AB                          stosw
1384 00000724 EBF7                          jmp    short pkl
1385                                  pkr:
1386 00000726 C3                            retn
1387
1388                                  ; 25/07/2015
1389                                  ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
1390                                  ; 17/02/2015
1391                                  ; 06/02/2015 (unix386.s)
1392                                  ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
1393                                  ;
1394                                  ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
1395                                  ;
1396                                  ;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) ------------------------------------
1397                                  ;     THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
1398                                  ;     THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
1399                                  ;     IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
1400                                  ;                                                                            :
1401                                  ;     THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
1402                                  ;     POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.        :
1403                                  ;     THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
1404                                  ;     OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE           :
1405                                  ;     DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                   :
1406                                  ;     THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH            :
1407                                  ;     INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A           :
1408                                  ;     ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.        :
1409                                  ;-------------------------------------------------------------------------------
1410                                  ;
1411
1412                                  timer_int:  ; IRQ 0
1413                                  ;int_08h:   ; Timer
1414                                        ; 14/10/2015
1415                                        ; Here, we are simulating system call entry (for task switch)
1416                                        ; (If multitasking is enabled,
1417                                        ; 'clock' procedure may jump to 'sysrelease')
1418 00000727 1E                          push  ds
1419 00000728 06                          push  es
1420 00000729 0FA0                        push  fs
1421 0000072B 0FA8                        push  gs
1422 0000072D 60                          pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
1423 0000072E 66B91000                    mov    cx, KDATA
1424 00000732 8ED9                          mov    ds, cx
1425 00000734 8EC1                          mov    es, cx
1426 00000736 8EE1                          mov    fs, cx
1427 00000738 8EE9                          mov    gs, cx
1428                                      ;
1429 0000073A 0F20D9                      mov   ecx, cr3
1430 0000073D 890D[DC070000]              mov   [cr3reg], ecx ; save current cr3 register value/content
1431                                      ;
1432 00000743 3B0D[68700000]              cmp   ecx, [k_page_dir]
1433 00000749 741F                        je    short T3
1434                                      ;
1435                                      ; timer interrupt has been occurred while OS is in user mode
1436 0000074B A3[48740000]                mov   [u.r0], eax
1437 00000750 89E1                        mov   ecx, esp
1438 00000752 83C130                      add   ecx, ESPACE ; 4 * 12 (stack frame)
1439 00000755 890D[40740000]              mov   [u.sp], ecx ; kernel stack pointer at the start of interrupt
1440 0000075B 8925[44740000]              mov   [u.usp], esp ; kernel stack points to user's registers
1441                                      ;
1442 00000761 8B0D[68700000]              mov   ecx, [k_page_dir]
1443 00000767 0F22D9                      mov   cr3, ecx
1444                                  T3:
1445 0000076A FB                          sti                         ; INTERRUPTS BACK ON
1446 0000076B 66FF05[E4700000]            INC   word [TIMER_LOW]    ; INCREMENT TIME
1447 00000772 7507                        JNZ   short T4            ; GO TO TEST_DAY
1448 00000774 66FF05[E6700000]            INC   word [TIMER_HIGH]   ; INCREMENT HIGH WORD OF TIME
1449                                  T4:                             ; TEST_DAY
1450 0000077B 66833D[E6700000]18          CMP   word [TIMER_HIGH],018H    ; TEST FOR COUNT EQUALING 24 HOURS
1451 00000783 7519                        JNZ   short T5            ; GO TO DISKETTE_CTL
1452 00000785 66813D[E4700000]B0-         CMP   word [TIMER_LOW],0B0H
1453 0000078D 00
1454 0000078E 750E                        JNZ   short T5            ; GO TO DISKETTE_CTL
1455
1456                                  ;-----      TIMER HAS GONE 24 HOURS
1457                                        ;;SUB  AX,AX
1458                                        ;MOV  [TIMER_HIGH],AX
1459                                        ;MOV  [TIMER_LOW],AX
1460 00000790 29C0                        sub   eax, eax
1461 00000792 A3[E4700000]                mov   [TIMER_LH], eax
1462                                      ;
1463 00000797 C605[E8700000]01            MOV   byte [TIMER_OFL],1
1464
1465                                  ;-----      TEST FOR DISKETTE TIME OUT
1466
1467                                  T5:
```

```
                                        ; 23/12/2014
0000079E EB1D                           jmp    short T6           ; will be replaced with nop, nop
                                                                  ; (9090h) if a floppy disk
                                                                  ; is detected.
                                        ;mov   al,[CS:MOTOR_COUNT]
000007A0 A0[EB700000]                   mov    al, [MOTOR_COUNT]
000007A5 FEC8                           dec    al
                                        ;mov   [CS:MOTOR_COUNT], al      ; DECREMENT DISKETTE MOTOR CONTROL
000007A7 A2[EB700000]                   mov    [MOTOR_COUNT], al
                                        ;mov   [ORG_MOTOR_COUNT], al
000007AC 750F                           JNZ    short T6           ; RETURN IF COUNT NOT OUT
000007AE B0F0                           mov    al,0F0h
                                        ;AND   [CS:MOTOR_STATUS],al      ; TURN OFF MOTOR RUNNING BITS
000007B0 2005[EA700000]                 and    [MOTOR_STATUS], al
                                        ;and   [ORG_MOTOR_STATUS], al
000007B6 B00C                           MOV    AL,0CH             ; bit 3 = enable IRQ & DMA,
                                                                  ; bit 2 = enable controller
                                                                  ;     1 = normal operation
                                                                  ;     0 = reset
                                                                  ; bit 0, 1 = drive select
                                                                  ; bit 4-7 = motor running bits
000007B8 66BAF203                       MOV    DX,03F2H           ; FDC CTL PORT
000007BC EE                             OUT    DX,AL              ; TURN OFF THE MOTOR
                                T6:
                                        ;inc   word [CS:wait_count]      ; 22/12/2014 (byte -> word)
                                                                  ; TIMER TICK INTERRUPT
                                        ;;inc  word [wait_count] ;;27/02/2015
                                        ;INT   1CH                ; TRANSFER CONTROL TO A USER ROUTINE
                                        ;;;;cli
                                        ;call  u_timer                   ; TRANSFER CONTROL TO A USER ROUTINE
000007BD FF15[D8070000]                 call   [x_timer] ; 14/05/2015
                                T7:
                                        ; 14/10/2015
000007C3 B020                           MOV    AL,EOI             ; GET END OF INTERRUPT MASK
000007C5 FA                             CLI                       ; DISABLE INTERRUPTS TILL STACK CLEARED
000007C6 E620                           OUT    INTA00,AL          ; END OF INTERRUPT TO 8259 - 1
                                        ;
000007C8 A1[DC070000]                   mov    eax, [cr3reg]             ; previous value/content of cr3 register
000007CD 0F22D8                         mov    cr3, eax  ; restore cr3 register content
                                        ;
000007D0 61                             popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
                                        ;
000007D1 0FA9                           pop    gs
000007D3 0FA1                           pop    fs
000007D5 07                             pop    es
000007D6 1F                             pop    ds
000007D7 CF                             iretd  ; return from interrupt


                                ; ////////////////

                                ; 14/05/2015 - Multi tasking 'clock' procedure (sys emt)
                                x_timer:
000007D8 [E0070000]                     dd     u_timer                   ; 14/05/2015
                                        ;dd    clock

                                ; 14/10/2015
000007DC 00000000               cr3reg: dd 0

                                        ; 06/02/2015
                                        ; 07/09/2014
                                        ; 21/08/2014
                                u_timer:
                                ;timer_int:  ; IRQ 0
                                        ; 06/02/2015
                                        ;push  eax
                                        ;push  edx
                                        ;push  ecx
                                        ;push  ebx
                                        ;push  ds
                                        ;push  es
                                        ;mov   eax, KDATA
                                        ;mov   ds, ax
                                        ;mov   es, ax
000007E0 FF05[AC700000]                 inc    dword [tcount]
000007E6 BB[F26B0000]                   mov    ebx, tcountstr + 4
000007EB 66A1[AC700000]                 mov    ax, [tcount]
000007F1 B90A000000                     mov    ecx, 10
                                rp_divtcnt:
000007F6 31D2                           xor    edx, edx
000007F8 F7F1                           div    ecx
000007FA 80C230                         add    dl, 30h
000007FD 8813                           mov    [ebx], dl
000007FF 6609C0                         or     ax, ax
00000802 7403                           jz     short print_lzero
00000804 4B                             dec    ebx
00000805 EBEF                           jmp    short rp_divtcnt
                                print_lzero:
00000807 81FB[EE6B0000]                 cmp    ebx, tcountstr
0000080D 7606                           jna    short print_tcount
0000080F 4B                             dec    ebx
00000810 C60330                         mov    byte [ebx], 30h
00000813 EBF2                           jmp    short print_lzero
                                print_tcount:
00000815 56                             push   esi
00000816 57                             push   edi
00000817 BE[CA6B0000]                   mov    esi, timer_msg ; Timer interrupt message
                                        ; 07/09/2014
0000081C 66BB0100                       mov    bx, 1        ; Video page 1
                                ptmsg:
00000820 AC                             lodsb
00000821 08C0                           or     al, al
00000823 740F                           jz     short ptmsg_ok
00000825 56                             push   esi
00000826 6653                           push   bx
```

```
1573 00000828 B42F                      mov     ah,  2Fh ; Green background, white forecolor
1574 0000082A E8AB0C0000                call  write_tty
1575 0000082F 665B                      pop   bx
1576 00000831 5E                        pop   esi
1577 00000832 EBEC                      jmp   short ptmsg
1578                                    ;; 27/08/2014
1579                                    ;mov     edi, 0B8000h + 0A0h ; Row 1
1580                                    ;call  printk
1581                                    ;
1582                                ptmsg_ok:
1583                                    ; 07/09/2014
1584 00000834 6631D2                    xor   dx, dx       ; column 0, row 0
1585 00000837 E8C20D0000                call  set_cpos     ; set cursor position to 0,0
1586                                    ; 23/02/2015
1587                                    ; 25/08/2014
1588                                    ;mov   ebx, scounter       ; (seconds counter)
1589                                    ;dec   byte [ebx+1]        ; (for reading real time clock)
1590                                ;     dec   byte [scounter+1]
1591                                ;;    jns   short timer_eoi           ; 0 -> 0FFh ?
1592                                ;     jns   short u_timer_retn
1593                                    ; 26/02/2015
1594                                ;     call  rtc_p
1595                                ;     mov   ebx, scounter       ; (seconds counter)
1596                                ;     mov   byte [ebx+1], 18    ; (18.2 timer ticks per second)
1597                                ;     dec   byte [ebx]          ; 19+18+18+18+18 (5)
1598                                ;     jnz   short timer_eoi           ; (109 timer ticks in 5 seconds)
1599                                ;     jnz   short u_timer_retn ; 06/02/2015
1600                                ;     mov   byte [ebx], 5
1601                                ;     inc   byte [ebx+1] ; 19
1602                                ;;timer_eoi:
1603                                ;;    mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
1604                                ;;    out   20h, al      ; 8259 PORT
1605                                    ;
1606                                ;u_timer_retn:  ; 06/02/2015
1607 0000083C 5F                        pop   edi
1608 0000083D 5E                        pop   esi
1609                                    ;pop  es
1610                                    ;pop  ds
1611                                    ;pop  ebx
1612                                    ;pop  ecx
1613                                    ;pop  edx
1614                                    ;pop  eax
1615                                    ;iret
1616 0000083E C3                        retn  ; 06/02/2015
1617
1618                                    ; 28/08/2014
1619                                irq0:
1620 0000083F 6A00                      push      dword 0
1621 00000841 EB48                      jmp   short which_irq
1622                                irq1:
1623 00000843 6A01                      push      dword 1
1624 00000845 EB44                      jmp   short which_irq
1625                                irq2:
1626 00000847 6A02                      push      dword 2
1627 00000849 EB40                      jmp   short which_irq
1628                                irq3:
1629                                    ; 20/11/2015
1630                                    ; 24/10/2015
1631 0000084B 2EFF15[393F0000]          call  dword [cs:com2_irq3]
1632 00000852 6A03                      push  dword 3
1633 00000854 EB35                      jmp   short which_irq
1634                                irq4:
1635                                    ; 20/11/2015
1636                                    ; 24/10/2015
1637 00000856 2EFF15[353F0000]          call  dword [cs:com1_irq4]
1638 0000085D 6A04                      push      dword 4
1639 0000085F EB2A                      jmp   short which_irq
1640                                irq5:
1641 00000861 6A05                      push      dword 5
1642 00000863 EB26                      jmp   short which_irq
1643                                irq6:
1644 00000865 6A06                      push      dword 6
1645 00000867 EB22                      jmp   short which_irq
1646                                irq7:
1647 00000869 6A07                      push      dword 7
1648 0000086B EB1E                      jmp   short which_irq
1649                                irq8:
1650 0000086D 6A08                      push      dword 8
1651 0000086F EB1A                      jmp   short which_irq
1652                                irq9:
1653 00000871 6A09                      push      dword 9
1654 00000873 EB16                      jmp   short which_irq
1655                                irq10:
1656 00000875 6A0A                      push      dword 10
1657 00000877 EB12                      jmp   short which_irq
1658                                irq11:
1659 00000879 6A0B                      push      dword 11
1660 0000087B EB0E                      jmp   short which_irq
1661                                irq12:
1662 0000087D 6A0C                      push      dword 12
1663 0000087F EB0A                      jmp   short which_irq
1664                                irq13:
1665 00000881 6A0D                      push      dword 13
1666 00000883 EB06                      jmp   short which_irq
1667                                irq14:
1668 00000885 6A0E                      push      dword 14
1669 00000887 EB02                      jmp   short which_irq
1670                                irq15:
1671 00000889 6A0F                      push      dword 15
1672                                    ;jmp  short which_irq
1673
1674                                    ; 19/10/2015
1675                                    ; 29/08/2014
1676                                    ; 21/08/2014
1677                                which_irq:
```

```
1678 0000088B 870424                    xchg   eax, [esp]  ; 28/08/2014
1679 0000088E 53                        push   ebx
1680 0000088F 56                        push   esi
1681 00000890 57                        push   edi
1682 00000891 1E                        push   ds
1683 00000892 06                        push   es
1684                                    ;
1685 00000893 88C3                      mov    bl, al
1686                                    ;
1687 00000895 B810000000                mov    eax, KDATA
1688 0000089A 8ED8                      mov    ds, ax
1689 0000089C 8EC0                      mov    es, ax
1690                                    ; 19/10/2015
1691 0000089E FC                        cld
1692                                      ; 27/08/2014
1693 0000089F 8105[786B0000]A000-         add     dword [scr_row], 0A0h
1694 000008A7 0000
1695                                    ;
1696 000008A9 B417                      mov    ah, 17h      ; blue (1) background,
1697                                                  ; light gray (7) forecolor
1698 000008AB 8B3D[786B0000]              mov     edi, [scr_row]
1699 000008B1 B049                      mov    al, 'I'
1700 000008B3 66AB                      stosw
1701 000008B5 B052                      mov    al, 'R'
1702 000008B7 66AB                      stosw
1703 000008B9 B051                      mov    al, 'Q'
1704 000008BB 66AB                      stosw
1705 000008BD B020                      mov    al, ' '
1706 000008BF 66AB                      stosw
1707 000008C1 88D8                      mov    al, bl
1708 000008C3 3C0A                      cmp    al, 10
1709 000008C5 7208                      jb     short iix
1710 000008C7 B031                            mov    al, '1'
1711 000008C9 66AB                            stosw
1712 000008CB 88D8                            mov    al, bl
1713 000008CD 2C0A                            sub    al, 10
1714                                  iix:
1715 000008CF 0430                            add    al, '0'
1716 000008D1 66AB                            stosw
1717 000008D3 B020                            mov    al, ' '
1718 000008D5 66AB                            stosw
1719 000008D7 B021                            mov    al, '!'
1720 000008D9 66AB                            stosw
1721 000008DB B020                            mov    al, ' '
1722 000008DD 66AB                            stosw
1723                                          ; 23/02/2015
1724 000008DF 80FB07                          cmp    bl, 7 ; check for IRQ 8 to IRQ 15
1725 000008E2 0F868D010000                    jna    iiret
1726 000008E8 B020                            mov    al, 20h  ; END OF INTERRUPT COMMAND TO
1727 000008EA E6A0                            out    0A0h, al ; the 2nd 8259
1728 000008EC E984010000                      jmp     iiret
1729                                          ;
1730                                          ; 22/08/2014
1731                                          ;mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
1732                                          ;out   20h, al      ; 8259 PORT
1733                                          ;
1734                                          ;pop   es
1735                                          ;pop   ds
1736                                          ;pop   edi
1737                                          ;pop   esi
1738                                          ;pop   ebx
1739                                          ;pop   eax
1740                                          ;iret
1741
1742                                          ; 02/04/2015
1743                                          ; 25/08/2014
1744                                  exc0:
1745 000008F1 6A00                              push       dword 0
1746 000008F3 E990000000                        jmp     cpu_except
1747                                  exc1:
1748 000008F8 6A01                              push       dword 1
1749 000008FA E989000000                        jmp     cpu_except
1750                                  exc2:
1751 000008FF 6A02                              push       dword 2
1752 00000901 E982000000                        jmp     cpu_except
1753                                  exc3:
1754 00000906 6A03                              push       dword 3
1755 00000908 EB7E                              jmp     cpu_except
1756                                  exc4:
1757 0000090A 6A04                              push       dword 4
1758 0000090C EB7A                              jmp     cpu_except
1759                                  exc5:
1760 0000090E 6A05                              push       dword 5
1761 00000910 EB76                              jmp     cpu_except
1762                                  exc6:
1763 00000912 6A06                              push       dword 6
1764 00000914 EB72                              jmp     cpu_except
1765                                  exc7:
1766 00000916 6A07                              push       dword 7
1767 00000918 EB6E                              jmp     cpu_except
1768                                  exc8:
1769                                        ; [esp] = Error code
1770 0000091A 6A08                              push       dword 8
1771 0000091C EB5C                              jmp     cpu_except_en
1772                                  exc9:
1773 0000091E 6A09                              push       dword 9
1774 00000920 EB66                              jmp     cpu_except
1775                                  exc10:
1776                                        ; [esp] = Error code
1777 00000922 6A0A                              push       dword 10
1778 00000924 EB54                              jmp     cpu_except_en
1779                                  exc11:
1780                                        ; [esp] = Error code
1781 00000926 6A0B                              push       dword 11
1782 00000928 EB50                              jmp     cpu_except_en
```

```
1783                                  exc12:
1784                                        ; [esp] = Error code
1785 0000092A 6A0C                          push       dword 12
1786 0000092C EB4C                          jmp     cpu_except_en
1787                                  exc13:
1788                                        ; [esp] = Error code
1789 0000092E 6A0D                          push       dword 13
1790 00000930 EB48                          jmp     cpu_except_en
1791                                  exc14:
1792                                        ; [esp] = Error code
1793 00000932 6A0E                          push       dword 14
1794 00000934 EB44                        jmp    short cpu_except_en
1795                                  exc15:
1796 00000936 6A0F                          push       dword 15
1797 00000938 EB4E                          jmp     cpu_except
1798                                  exc16:
1799 0000093A 6A10                          push       dword 16
1800 0000093C EB4A                          jmp     cpu_except
1801                                  exc17:
1802                                        ; [esp] = Error code
1803 0000093E 6A11                          push       dword 17
1804 00000940 EB38                        jmp    short cpu_except_en
1805                                  exc18:
1806 00000942 6A12                          push       dword 18
1807 00000944 EB42                        jmp    short cpu_except
1808                                  exc19:
1809 00000946 6A13                          push       dword 19
1810 00000948 EB3E                        jmp    short cpu_except
1811                                  exc20:
1812 0000094A 6A14                          push       dword 20
1813 0000094C EB3A                        jmp    short cpu_except
1814                                  exc21:
1815 0000094E 6A15                          push       dword 21
1816 00000950 EB36                        jmp    short cpu_except
1817                                  exc22:
1818 00000952 6A16                          push       dword 22
1819 00000954 EB32                        jmp    short cpu_except
1820                                  exc23:
1821 00000956 6A17                          push       dword 23
1822 00000958 EB2E                        jmp    short cpu_except
1823                                  exc24:
1824 0000095A 6A18                          push       dword 24
1825 0000095C EB2A                        jmp    short cpu_except
1826                                  exc25:
1827 0000095E 6A19                          push       dword 25
1828 00000960 EB26                        jmp    short cpu_except
1829                                  exc26:
1830 00000962 6A1A                          push       dword 26
1831 00000964 EB22                        jmp    short cpu_except
1832                                  exc27:
1833 00000966 6A1B                          push       dword 27
1834 00000968 EB1E                        jmp    short cpu_except
1835                                  exc28:
1836 0000096A 6A1C                          push       dword 28
1837 0000096C EB1A                        jmp    short cpu_except
1838                                  exc29:
1839 0000096E 6A1D                          push       dword 29
1840 00000970 EB16                        jmp    short cpu_except
1841                                  exc30:
1842 00000972 6A1E                          push       dword 30
1843 00000974 EB04                        jmp    short cpu_except_en
1844                                  exc31:
1845 00000976 6A1F                          push       dword 31
1846 00000978 EB0E                          jmp     short cpu_except
1847
1848                                        ; 19/10/2015
1849                                        ; 19/09/2015
1850                                        ; 01/09/2015
1851                                        ; 28/08/2015
1852                                        ; 28/08/2014
1853                                  cpu_except_en:
1854 0000097A 87442404                      xchg   eax, [esp+4] ; Error code
1855 0000097E 36A3[28850000]                mov    [ss:error_code], eax
1856 00000984 58                            pop    eax  ; Exception number
1857 00000985 870424                        xchg   eax, [esp]
1858                                              ; eax = eax before exception
1859                                              ; [esp] -> exception number
1860                                              ; [esp+4] -> EIP to return
1861                                        ; 19/10/2015
1862                                        ; 19/09/2015
1863                                        ; 01/09/2015
1864                                        ; 28/08/2015
1865                                        ; 29/08/2014
1866                                        ; 28/08/2014
1867                                        ; 25/08/2014
1868                                        ; 21/08/2014
1869                                  cpu_except:  ; CPU Exceptions
1870 00000988 FC                            cld
1871 00000989 870424                        xchg   eax, [esp]
1872                                              ; eax = Exception number
1873                                              ; [esp] = eax (before exception)
1874 0000098C 53                            push   ebx
1875 0000098D 56                            push   esi
1876 0000098E 57                            push   edi
1877 0000098F 1E                            push   ds
1878 00000990 06                            push   es
1879                                        ; 28/08/2015
1880 00000991 66BB1000                      mov    bx, KDATA
1881 00000995 8EDB                          mov    ds, bx
1882 00000997 8EC3                          mov    es, bx
1883 00000999 0F20DB                        mov    ebx, cr3
1884 0000099C 53                            push   ebx ; (*) page directory
1885                                        ; 19/10/2015
1886 0000099D FC                            cld
1887                                        ; 25/03/2015
```

```
1888 0000099E 8B1D[68700000]              mov   ebx, [k_page_dir]
1889 000009A4 0F22DB                      mov   cr3, ebx
1890                                      ; 28/08/2015
1891 000009A7 83F80E                      cmp   eax, 0Eh ; 14, PAGE FAULT
1892 000009AA 7512                        jne   short cpu_except_nfp
1893 000009AC E816290000                  call  page_fault_handler
1894 000009B1 21C0                        and   eax, eax
1895 000009B3 0F84B8000000                  jz  iiretp ; 01/09/2015
1896 000009B9 B80E000000                  mov   eax, 0Eh ; 14
1897                                cpu_except_nfp:
1898                                      ; 02/04/2015
1899 000009BE BB[E0060000]                mov   ebx, hang
1900 000009C3 875C241C                    xchg  ebx, [esp+28]
1901                                            ; EIP (points to instruction which faults)
1902                                            ; New EIP (hang)
1903 000009C7 891D[2C850000]              mov   [FaultOffset], ebx
1904 000009CD C744242008000000            mov   dword [esp+32], KCODE ; kernel's code segment
1905 000009D5 814C242400020000            or    dword [esp+36], 200h ; enable interrupts (set IF)
1906                                      ;
1907 000009DD 88C4                        mov   ah, al
1908 000009DF 240F                        and   al, 0Fh
1909 000009E1 3C09                        cmp   al, 9
1910 000009E3 7602                        jna   short h1ok
1911 000009E5 0407                        add   al, 'A'-':'
1912                                h1ok:
1913 000009E7 D0EC                        shr   ah, 1
1914 000009E9 D0EC                        shr   ah, 1
1915 000009EB D0EC                        shr   ah, 1
1916 000009ED D0EC                        shr   ah, 1
1917 000009EF 80FC09                      cmp   ah, 9
1918 000009F2 7603                        jna   short h2ok
1919 000009F4 80C407                      add   ah, 'A'-':'
1920                                h2ok:
1921 000009F7 86E0                        xchg  ah, al
1922 000009F9 66053030                    add   ax, '00'
1923 000009FD 66A3[066C0000]              mov   [excnstr], ax
1924                                      ;
1925                                      ; 29/08/2014
1926 00000A03 A1[2C850000]                mov   eax, [FaultOffset]
1927 00000A08 51                          push  ecx
1928 00000A09 52                          push  edx
1929 00000A0A 89E3                        mov   ebx, esp
1930                                      ; 28/08/2015
1931 00000A0C B910000000                  mov   ecx, 16       ; divisor value to convert binary number
1932                                                  ; to hexadecimal string
1933                                      ;mov  ecx, 10         ; divisor to convert
1934                                                  ; binary number to decimal string
1935                                b2d1:
1936 00000A11 31D2                        xor   edx, edx
1937 00000A13 F7F1                        div   ecx
1938 00000A15 6652                        push  dx
1939 00000A17 39C8                        cmp   eax, ecx
1940 00000A19 73F6                        jnb   short b2d1
1941 00000A1B BF[116C0000]                mov   edi, EIPstr ; EIP value
1942                                                  ; points to instruction which faults
1943                                      ; 28/08/2015
1944 00000A20 89C2                        mov   edx, eax
1945                                b2d2:
1946                                      ;add  al, '0'
1947 00000A22 8A82[EF180000]              mov   al, [edx+hexchrs]
1948 00000A28 AA                          stosb           ; write hexadecimal digit to its place
1949 00000A29 39E3                        cmp   ebx, esp
1950 00000A2B 7606                        jna   short b2d3
1951 00000A2D 6658                        pop   ax
1952 00000A2F 88C2                        mov   dl, al
1953 00000A31 EBEF                        jmp   short b2d2
1954                                b2d3:
1955 00000A33 B068                        mov   al, 'h' ; 28/08/2015
1956 00000A35 AA                          stosb
1957 00000A36 B020                        mov   al, 20h         ; space
1958 00000A38 AA                          stosb
1959 00000A39 30C0                        xor   al, al    ; to do it an ASCIIZ string
1960 00000A3B AA                          stosb
1961                                      ;
1962 00000A3C 5A                          pop   edx
1963 00000A3D 59                          pop   ecx
1964                                      ;
1965 00000A3E B44F                        mov   ah, 4Fh      ; red (4) background,
1966                                                 ; white (F) forecolor
1967 00000A40 BE[F66B0000]                mov   esi, exc_msg ; message offset
1968                                      ;
1969 00000A45 EB11                        jmp   short piemsg
1970                                      ;
1971                                        ;add   dword [scr_row], 0A0h
1972                                        ;mov   edi, [scr_row]
1973                                        ;
1974                                      ;call printk
1975                                      ;
1976                                      ;mov  al, 20h ; END OF INTERRUPT COMMAND TO 8259
1977                                      ;out  20h, al      ; 8259 PORT
1978                                      ;
1979                                      ;pop  es
1980                                      ;pop  ds
1981                                      ;pop  edi
1982                                      ;pop  esi
1983                                      ;pop  eax
1984                                      ;iret
1985
1986                                      ; 28/08/2015
1987                                      ; 23/02/2015
1988                                      ; 20/08/2014
1989                                ignore_int:
1990 00000A47 50                          push  eax
1991 00000A48 53                          push  ebx ; 23/02/2015
1992 00000A49 56                          push  esi
```

```
1993 00000A4A 57                        push   edi
1994 00000A4B 1E                        push   ds
1995 00000A4C 06                        push   es
1996                                    ; 28/08/2015
1997 00000A4D 0F20D8                    mov    eax, cr3
1998 00000A50 50                        push   eax ; (*) page directory
1999                                    ;
2000 00000A51 B467                      mov    ah, 67h      ; brown (6) background,
2001                                                 ; light gray (7) forecolor
2002 00000A53 BE[B46B0000]              mov    esi, int_msg ; message offset
2003                                piemsg:
2004                                      ; 27/08/2014
2005 00000A58 8105[786B0000]A000-         add     dword [scr_row], 0A0h
2006 00000A60 0000
2007 00000A62 8B3D[786B0000]              mov     edi, [scr_row]
2008                                      ;
2009 00000A68 E8B0FCFFFF                  call   printk
2010                                      ;
2011                                      ; 23/02/2015
2012 00000A6D B020                        mov    al, 20h  ; END OF INTERRUPT COMMAND TO
2013 00000A6F E6A0                        out    0A0h, al ; the 2nd 8259
2014                                iiretp: ; 01/09/2015
2015                                      ; 28/08/2015
2016 00000A71 58                          pop    eax ; (*) page directory
2017 00000A72 0F22D8                      mov    cr3, eax
2018                                      ;
2019                                iiret:
2020                                      ; 22/08/2014
2021 00000A75 B020                        mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
2022 00000A77 E620                        out    20h, al      ; 8259 PORT
2023                                      ;
2024 00000A79 07                          pop    es
2025 00000A7A 1F                          pop    ds
2026 00000A7B 5F                          pop    edi
2027 00000A7C 5E                          pop    esi
2028 00000A7D 5B                          pop    ebx ; 29/08/2014
2029 00000A7E 58                          pop    eax
2030 00000A7F CF                          iretd
2031
2032                                      ; 26/02/2015
2033                                      ; 07/09/2014
2034                                      ; 25/08/2014
2035                                rtc_int:       ; Real Time Clock Interrupt (IRQ 8)
2036                                      ; 22/08/2014
2037 00000A80 50                          push   eax
2038 00000A81 53                          push   ebx ; 29/08/2014
2039 00000A82 56                          push   esi
2040 00000A83 57                          push   edi
2041 00000A84 1E                          push   ds
2042 00000A85 06                          push   es
2043                                      ;
2044 00000A86 B810000000                  mov    eax, KDATA
2045 00000A8B 8ED8                        mov    ds, ax
2046 00000A8D 8EC0                        mov    es, ax
2047                                      ;
2048                                      ; 25/08/2014
2049 00000A8F E884000000                  call   rtc_p
2050                                      ;
2051                                      ; 22/02/2015 - dsectpm.s
2052                                      ; [ source: http://wiki.osdev.org/RTC ]
2053                                      ; read status register C to complete procedure
2054                                      ;(it is needed to get a next IRQ 8)
2055 00000A94 B00C                        mov    al, 0Ch ;
2056 00000A96 E670                        out    70h, al ; select register C
2057 00000A98 90                          nop
2058 00000A99 E471                        in     al, 71h ; just throw away contents
2059                                      ; 22/02/2015
2060 00000A9B B020                        MOV    AL,EOI       ; END OF INTERRUPT
2061 00000A9D E6A0                        OUT    INTB00,AL    ; FOR CONTROLLER #2
2062                                      ;
2063 00000A9F EBD4                        jmp    short iiret
2064
2065                                      ; 22/08/2014
2066                                      ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
2067                                      ; (INT 1Ah)
2068                                      ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
2069                                time_of_day:
2070 00000AA1 E866010000                  call   UPD_IPR                   ; WAIT TILL UPDATE NOT IN PROGRESS
2071 00000AA6 726F                          jc       short rtc_retn
2072 00000AA8 B000                        mov    al, CMOS_SECONDS
2073 00000AAA E845010000                  call   CMOS_READ
2074 00000AAF A2[DC700000]                mov    [time_seconds], al
2075 00000AB4 B002                        mov    al, CMOS_MINUTES
2076 00000AB6 E839010000                  call   CMOS_READ
2077 00000ABB A2[DD700000]                mov    [time_minutes], al
2078 00000AC0 B004                        mov    al, CMOS_HOURS
2079 00000AC2 E82D010000                  call   CMOS_READ
2080 00000AC7 A2[DE700000]                  mov     [time_hours], al
2081 00000ACC B006                        mov    al, CMOS_DAY_WEEK
2082 00000ACE E821010000                  call   CMOS_READ
2083 00000AD3 A2[DF700000]                mov    [date_wday], al
2084 00000AD8 B007                        mov    al, CMOS_DAY_MONTH
2085 00000ADA E815010000                  call   CMOS_READ
2086 00000ADF A2[E0700000]                mov    [date_day], al
2087 00000AE4 B008                        mov    al, CMOS_MONTH
2088 00000AE6 E809010000                  call   CMOS_READ
2089 00000AEB A2[E1700000]                mov    [date_month], al
2090 00000AF0 B009                        mov    al, CMOS_YEAR
2091 00000AF2 E8FD000000                  call   CMOS_READ
2092 00000AF7 A2[E2700000]                mov    [date_year], al
2093 00000AFC B032                        mov    al, CMOS_CENTURY
2094 00000AFE E8F1000000                  call   CMOS_READ
2095 00000B03 A2[E3700000]                mov    [date_century], al
2096                                      ;
2097 00000B08 B000                        mov    al, CMOS_SECONDS
```

```
2098 00000B0A E8E5000000                call  CMOS_READ
2099 00000B0F 3A05[DC700000]            cmp   al, [time_seconds]
2100 00000B15 758A                      jne   short time_of_day
2101
2102                                rtc_retn:
2103 00000B17 C3                        retn
2104
2105                                rtc_p:
2106                                    ; 07/09/2014
2107                                    ; 29/08/2014
2108                                    ; 27/08/2014
2109                                    ; 25/08/2014
2110                                    ; Print Real Time Clock content
2111                                    ;
2112                                    ;
2113 00000B18 E884FFFFFF                call  time_of_day
2114 00000B1D 72F8                      jc    short rtc_retn
2115                                    ;
2116 00000B1F 3A05[686C0000]            cmp   al, [ptime_seconds]
2117 00000B25 74F0                        je    short rtc_retn ; 29/08/2014
2118                                    ;
2119 00000B27 A2[686C0000]              mov   [ptime_seconds], al
2120                                    ;
2121 00000B2C A0[E3700000]              mov   al, [date_century]
2122 00000B31 E8F1000000                call  bcd_to_ascii
2123 00000B36 66A3[356C0000]            mov   [datestr+6], ax
2124 00000B3C A0[E2700000]              mov   al, [date_year]
2125 00000B41 E8E1000000                call  bcd_to_ascii
2126 00000B46 66A3[376C0000]            mov   [datestr+8], ax
2127 00000B4C A0[E1700000]              mov   al, [date_month]
2128 00000B51 E8D1000000                call  bcd_to_ascii
2129 00000B56 66A3[326C0000]            mov   [datestr+3], ax
2130 00000B5C A0[E0700000]              mov   al, [date_day]
2131 00000B61 E8C1000000                call  bcd_to_ascii
2132 00000B66 66A3[2F6C0000]            mov   [datestr], ax
2133                                    ;
2134 00000B6C 0FB61D[DF700000]          movzx ebx, byte [date_wday]
2135 00000B73 C0E302                    shl   bl, 2
2136 00000B76 81C3[486C0000]            add   ebx, daytmp
2137 00000B7C 8B03                      mov   eax, [ebx]
2138 00000B7E A3[3A6C0000]              mov   [daystr], eax
2139                                    ;
2140 00000B83 A0[DE700000]              mov   al, [time_hours]
2141 00000B88 E89A000000                call  bcd_to_ascii
2142 00000B8D 66A3[3E6C0000]            mov   [timestr], ax
2143 00000B93 A0[DD700000]              mov   al, [time_minutes]
2144 00000B98 E88A000000                call  bcd_to_ascii
2145 00000B9D 66A3[416C0000]            mov   [timestr+3], ax
2146 00000BA3 A0[DC700000]              mov   al, [time_seconds]
2147 00000BA8 E87A000000                call  bcd_to_ascii
2148 00000BAD 66A3[446C0000]            mov   [timestr+6], ax
2149                                    ;
2150 00000BB3 BE[1D6C0000]              mov   esi, rtc_msg ; message offset
2151                                    ; 23/02/2015
2152 00000BB8 52                        push  edx
2153 00000BB9 51                        push  ecx
2154                                    ; 07/09/2014
2155 00000BBA 66BB0200                  mov   bx, 2        ; Video page 2
2156                                prtmsg:
2157 00000BBE AC                        lodsb
2158 00000BBF 08C0                      or    al, al
2159 00000BC1 740F                      jz    short prtmsg_ok
2160 00000BC3 56                        push  esi
2161 00000BC4 6653                      push  bx
2162 00000BC6 B43F                        mov ah, 3Fh      ; cyan (6) background,
2163                                                       ; white (F) forecolor
2164 00000BC8 E80D090000                call  write_tty
2165 00000BCD 665B                      pop   bx
2166 00000BCF 5E                        pop   esi
2167 00000BD0 EBEC                      jmp   short prtmsg
2168                                    ;
2169                                    ;mov   edi, 0B8000h+0A0h+0A0h ; Row 2
2170                                    ;call  printk
2171                                prtmsg_ok:
2172                                    ; 07/09/2014
2173 00000BD2 6631D2                    xor   dx, dx       ; column 0, row 0
2174 00000BD5 E8240A0000                call  set_cpos     ; set curspor position to 0,0
2175                                    ; 23/02/2015
2176 00000BDA 59                        pop   ecx
2177 00000BDB 5A                        pop   edx
2178 00000BDC C3                        retn
2179
2180                                ; Default IRQ 7 handler against spurious IRQs (from master PIC)
2181                                ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
2182                                default_irq7:
2183 00000BDD 6650                      push  ax
2184 00000BDF B00B                      mov   al, 0Bh  ; In-Service register
2185 00000BE1 E620                      out   20h, al
2186 00000BE3 EB00                        jmp short $+2
2187 00000BE5 EB00                      jmp short $+2
2188 00000BE7 E420                      in    al, 20h
2189 00000BE9 2480                      and   al, 80h ; bit 7 (is it real IRQ 7 or fake?)
2190 00000BEB 7404                        jz    short irq7_iret ; Fake (spurious) IRQ, do not send EOI
2191 00000BED B020                        mov   al, 20h ; EOI
2192 00000BEF E620                      out   20h, al
2193                                irq7_iret:
2194 00000BF1 6658                      pop   ax
2195 00000BF3 CF                        iretd
2196
2197                                    ; 22/08/2014
2198                                    ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
2199                                CMOS_READ:
2200 00000BF4 9C                        pushf          ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
2201 00000BF5 D0C0                      rol   al, 1  ; MOVE NMI BIT TO LOW POSITION
2202 00000BF7 F9                        stc          ; FORCE NMI BIT ON IN CARRY FLAG
```

```
2203 00000BF8 D0D8                      rcr    al, 1  ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
2204 00000BFA FA                        cli           ; DISABLE INTERRUPTS
2205 00000BFB E670                      out    CMOS_PORT, al; ADDRESS LOCATION AND DISABLE NMI
2206 00000BFD 90                        nop           ; I/O DELAY
2207 00000BFE E471                      in     al, CMOS_DATA; READ THE REQUESTED CMOS LOCATION
2208 00000C00 6650                      push   ax     ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
2209                                    ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
2210                                                 ; ----- 10/06/85 (test4.asm)
2211 00000C02 B01E                      mov    al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
2212                                    ;mov   al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
2213 00000C04 D0D8                      rcr    al, 1  ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
2214 00000C06 E670                      out    CMOS_PORT, al; SET DEFAULT TO READ ONLY REGISTER
2215 00000C08 6658                      pop    ax     ; RESTORE (AH) AND (AL), CMOS BYTE
2216 00000C0A 9D                        popf
2217 00000C0B C3                        retn          ; RETURN WITH FLAGS RESTORED
2218
2219                                    ; 22/08/2014
2220                                    ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
2221                                UPD_IPR:                        ; WAIT TILL UPDATE NOT IN PROGRESS
2222 00000C0C 51                        push   ecx
2223 00000C0D B9FFFF0000                mov    ecx, 65535         ; SET TIMEOUT LOOP COUNT (= 800)
2224                                            ; mov cx, 800
2225                                UPD_10:
2226 00000C12 B00A                      mov    al, CMOS_REG_A           ; ADDRESS STATUS REGISTER A
2227 00000C14 FA                        cli                     ; NO TIMER INTERRUPTS DURING UPDATES
2228 00000C15 E8DAFFFFFF                call   CMOS_READ        ; READ UPDATE IN PROCESS FLAG
2229 00000C1A A880                      test   al, 80h                 ; IF UIP BIT IS ON ( CANNOT READ TIME )
2230 00000C1C 7406                      jz     short UPD_90     ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
2231 00000C1E FB                        sti                     ; ALLOW INTERRUPTS WHILE WAITING
2232 00000C1F E2F1                      loop   UPD_10           ; LOOP TILL READY OR TIMEOUT
2233 00000C21 31C0                      xor    eax, eax         ; CLEAR RESULTS IF ERROR
2234                                            ; xor ax, ax
2235 00000C23 F9                        stc                     ; SET CARRY FOR ERROR
2236                                UPD_90:
2237 00000C24 59                        pop    ecx              ; RESTORE CALLERS REGISTER
2238 00000C25 FA                        cli                     ; INTERRUPTS OFF DURING SET
2239 00000C26 C3                        retn                    ; RETURN WITH CY FLAG SET
2240
2241                                bcd_to_ascii:
2242                                    ; 25/08/2014
2243                                    ; INPUT ->
2244                                    ;      al = Packed BCD number
2245                                    ; OUTPUT ->
2246                                    ;      ax  = ASCII word/number
2247                                    ;
2248                                    ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
2249                                    ;
2250 00000C27 D410                      db 0D4h,10h                    ; Undocumented inst. AAM
2251                                                            ; AH = AL / 10h
2252                                                            ; AL = AL MOD 10h
2253 00000C29 660D3030                  or ax,'00'                     ; Make it ASCII based
2254
2255 00000C2D 86E0                        xchg ah, al
2256
2257 00000C2F C3                        retn
2258
2259
2260                                %include 'keyboard.inc' ; 07/03/2015
2261                               <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC
2262                               <1> ; Last Modification: 17/10/2015
2263                               <1> ;               (Keyboard Data is in 'KYBDATA.INC')
2264                               <1> ;
2265                               <1> ; ///////// KEYBOARD FUNCTIONS (PROCEDURES) ///////////////
2266                               <1>
2267                               <1> ; 30/06/2015
2268                               <1> ; 11/03/2015
2269                               <1> ; 28/02/2015
2270                               <1> ; 25/02/2015
2271                               <1> ; 20/02/2015
2272                               <1> ; 18/02/2015
2273                               <1> ; 03/12/2014
2274                               <1> ; 07/09/2014
2275                               <1> ; KEYBOARD INTERRUPT HANDLER
2276                               <1> ; (kb_int - Retro UNIX 8086 v1 - U0.ASM, 30/06/2014)
2277                               <1>
2278                               <1> ;getch:
2279                               <1> ;     ; 18/02/2015
2280                               <1> ;     ; This routine will be replaced with Retro UNIX 386
2281                               <1> ;     ; version of Retro UNIX 8086 getch (tty input)
2282                               <1> ;     ; routine, later... (multi tasking ability)
2283                               <1> ;     ; 28/02/2015
2284                               <1> ;     sti    ; enable interrupts
2285                               <1> ;     ;
2286                               <1> ;     ;push  esi
2287                               <1> ;     ;push  ebx
2288                               <1> ;     ;xor   ebx, ebx
2289                               <1> ;     ;mov   bl, [ptty]  ; active_page
2290                               <1> ;     ;mov   esi, ebx
2291                               <1> ;     ;shl   si, 1
2292                               <1> ;     ;add   esi, ttychr
2293                               <1> ;getch_1:
2294                               <1> ;     ;mov   ax, [esi]
2295                               <1> ;     mov    ax, [ttychr] ; video page 0 (tty0)
2296                               <1> ;     and    ax, ax
2297                               <1> ;     jz     short getch_2
2298                               <1> ;     mov    word [ttychr], 0
2299                               <1> ;     ;mov   word [esi], 0
2300                               <1> ;     ;pop   ebx
2301                               <1> ;     ;pop   esi
2302                               <1> ;     retn
2303                               <1> ;getch_2:
2304                               <1> ;     hlt    ; not proper for multi tasking!
2305                               <1> ;            ; (temporary halt for now)
2306                               <1> ;            ; 'sleep' on tty
2307                               <1> ;            ; will (must) be located here
```

```
2308                                    <1> ;       nop
2309                                    <1> ;       jmp     short getch_1
2310                                    <1>
2311                                    <1> keyb_int:
2312                                    <1>         ; 30/06/2015
2313                                    <1>         ; 25/02/2015
2314                                    <1>         ; 20/02/2015
2315                                    <1>         ; 03/12/2014 (getc_int - INT 16h modifications)
2316                                    <1>         ; 07/09/2014 - Retro UNIX 386 v1
2317                                    <1>         ; 30/06/2014
2318                                    <1>         ; 10/05/2013
2319                                    <1>               ; Retro Unix 8086 v1 feature only!
2320                                    <1>         ; 03/03/2014
2321                                    <1>
2322 00000C30 1E                       <1>         push    ds
2323 00000C31 53                       <1>         push    ebx
2324 00000C32 50                       <1>         push    eax
2325                                    <1>         ;
2326 00000C33 66B81000                 <1>         mov     ax, KDATA
2327 00000C37 8ED8                     <1>         mov     ds, ax
2328                                    <1>         ;
2329 00000C39 9C                       <1>         pushfd
2330 00000C3A 0E                       <1>         push    cs
2331 00000C3B E823020000               <1>         call    kb_int  ; int_09h
2332                                    <1>         ;
2333 00000C40 B411                     <1>         mov     ah, 11h         ; 03/12/2014
2334                                    <1>         ;call   getc
2335 00000C42 E856000000               <1>         call    int_16h ; 30/06/2015
2336 00000C47 7450                     <1>         jz      short keyb_int4
2337                                    <1>         ;
2338 00000C49 B410                     <1>         mov     ah, 10h         ; 03/12/2014
2339                                    <1>         ;call   getc
2340 00000C4B E84D000000               <1>         call    int_16h ; 30/06/2015
2341                                    <1>         ;
2342                                    <1>         ; 20/02/2015
2343 00000C50 0FB61D[96700000]         <1>           movzx   ebx, byte [ptty]  ; active_page
2344                                    <1>         ;
2345 00000C57 20C0                     <1>         and     al, al
2346 00000C59 751E                     <1>         jnz     short keyb_int1
2347                                    <1>         ;
2348 00000C5B 80FC68                   <1>         cmp     ah, 68h       ; ALT + F1 key
2349 00000C5E 7219                     <1>         jb      short keyb_int1
2350 00000C60 80FC6F                   <1>         cmp     ah, 6Fh  ; ALT + F8 key
2351 00000C63 7714                     <1>         ja      short keyb_int1
2352                                    <1>         ;
2353 00000C65 88D8                     <1>         mov     al, bl
2354 00000C67 0468                     <1>         add     al, 68h
2355 00000C69 38E0                     <1>         cmp     al, ah
2356 00000C6B 7409                     <1>         je      short keyb_int0
2357 00000C6D 88E0                     <1>         mov     al, ah
2358 00000C6F 2C68                     <1>         sub     al, 68h
2359 00000C71 E8520B0000               <1>         call    tty_sw
2360                                    <1>         ;movzx ebx, [ptty]  ; active_page
2361                                    <1> keyb_int0: ; 30/06/2015
2362 00000C76 6631C0                   <1>         xor     ax, ax
2363                                    <1> keyb_int1:
2364 00000C79 D0E3                     <1>         shl     bl, 1
2365 00000C7B 81C3[98700000]           <1>         add     ebx, ttychr
2366                                    <1>         ;
2367 00000C81 6609C0                   <1>         or      ax, ax
2368 00000C84 7406                     <1>         jz      short keyb_int2
2369                                    <1>         ;
2370 00000C86 66833B00                 <1>         cmp     word [ebx], 0
2371 00000C8A 7703                     <1>           ja      short keyb_int3
2372                                    <1> keyb_int2:
2373 00000C8C 668903                   <1>           mov  [ebx], ax  ; Save ascii code
2374                                    <1>                           ; and scan code of the character
2375                                    <1>                           ; for current tty (or last tty
2376                                    <1>                           ; just before tty switch).
2377                                    <1> keyb_int3:
2378 00000C8F A0[96700000]             <1>           mov     al, [ptty]
2379 00000C94 E820480000               <1>         call    wakeup
2380                                    <1>         ;
2381                                    <1> keyb_int4:
2382 00000C99 58                       <1>         pop     eax
2383 00000C9A 5B                       <1>         pop     ebx
2384 00000C9B 1F                       <1>         pop     ds
2385 00000C9C CF                       <1>         iret
2386                                    <1>
2387                                    <1> ; 18/02/2015
2388                                    <1> ; REMINDER: Only 'keyb_int' (IRQ 9) must call getc.
2389                                    <1> ; 'keyb_int' always handles 'getc' at 1st and puts the
2390                                    <1> ; scancode and ascii code of the character
2391                                    <1> ; in the tty input (ttychr) buffer.
2392                                    <1> ; Test procedures must call 'getch' for tty input
2393                                    <1> ; otherwise, 'getc' will not be able to return to the caller
2394                                    <1> ; due to infinite (key press) waiting loop.
2395                                    <1> ;
2396                                    <1> ; 03/12/2014
2397                                    <1> ; 26/08/2014
2398                                    <1> ; KEYBOARD I/O
2399                                    <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
2400                                    <1>
2401                                    <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.
2402                                    <1> ;     (The reason of using '_k' labels!!!) (27/08/2014)
2403                                    <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.
2404                                    <1> ;     ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
2405                                    <1>
2406                                    <1> int_16h: ; 30/06/2015
2407                                    <1> ;getc:
2408 00000C9D 9C                       <1>         pushfd ; 28/08/2014
2409 00000C9E 0E                       <1>         push    cs
2410 00000C9F E801000000               <1>         call    getc_int
2411 00000CA4 C3                       <1>         retn
2412                                    <1>
```

```
2413                                    <1> getc_int:
2414                                    <1>         ; 28/02/2015
2415                                    <1>         ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
2416                                    <1>         ;             instead of pc-at bios - 1985-)
2417                                    <1>         ; 28/08/2014 (_k1d)
2418                                    <1>         ; 30/06/2014
2419                                    <1>         ; 03/03/2014
2420                                    <1>         ; 28/02/2014
2421                                    <1>         ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
2422                                    <1>         ; rombios source code (21/04/1986)
2423                                    <1>         ;       'keybd.asm', INT 16H, KEYBOARD_IO
2424                                    <1>         ;
2425                                    <1>         ; KYBD --- 03/06/86  KEYBOARD BIOS
2426                                    <1>         ;
2427                                    <1>         ;--- INT 16 H ----------------------------------------------------------------
-
2428                                    <1>         ; KEYBOARD I/O                                                          :
2429                                    <1>         ;     THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT                       :
2430                                    <1>         ; INPUT                                                                 :
2431                                    <1>         ;     (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,   :
2432                                    <1>         ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH).             :
2433                                    <1>         ;               THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE   :
2434                                    <1>         ;                  STANDARD PC OR PCAT KEYBOARD
:
2435                                    <1>         ;--------------------------------------------------------------------------
:
2436                                    <1>         ;     (AH)= 01H  SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS     :
2437                                    <1>         ;               AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.          :
2438                                    <1>         ;               (ZF)= 1 -- NO CODE AVAILABLE                             :
2439                                    <1>         ;               (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER             :
2440                                    <1>         ;               IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :
2441                                    <1>         ;               IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.             :
2442                                    <1>         ;               THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES    :
2443                                    <1>         ;--------------------------------------------------------------------------
:
2444                                    <1>         ;     (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER            :
2445                                    <1>         ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE        :
2446                                    <1>         ;               EQUATES FOR @KB_FLAG                                      :
2447                                    <1>         ;--------------------------------------------------------------------------
:
2448                                    <1>         ;     (AH)= 03H  SET TYPAMATIC RATE AND DELAY                              :
2449                                    <1>         ;           (AL) = 05H                                                     :
2450                                    <1>         ;           (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)          :
2451                                    <1>         ;                                                                        :
2452                                    <1>         ;                   REGISTER     RATE      REGISTER     RATE
:
2453                                    <1>         ;                    VALUE     SELECTED     VALUE     SELECTED
:
2454                                    <1>         ;                   ------------------------------------------
:
2455                                    <1>         ;                   00H        30.0        10H        7.5               :
2456                                    <1>         ;                   01H        26.7        11H        6.7               :
2457                                    <1>         ;                   02H        24.0        12H        6.0               :
2458                                    <1>         ;                   03H        21.8        13H        5.5               :
2459                                    <1>         ;                   04H        20.0        14H        5.0               :
2460                                    <1>         ;                   05H        18.5        15H        4.6               :
2461                                    <1>         ;                   06H        17.1        16H        4.3               :
2462                                    <1>         ;                   07H        16.0        17H        4.0               :
2463                                    <1>         ;                   08H        15.0        18H        3.7               :
2464                                    <1>         ;                   09H        13.3        19H        3.3               :
2465                                    <1>         ;                   0AH        12.0        1AH        3.0               :
2466                                    <1>         ;                   0BH        10.9        1BH        2.7               :
2467                                    <1>         ;                   0CH        10.0        1CH        2.5               :
2468                                    <1>         ;                   0DH         9.2        1DH        2.3               :
2469                                    <1>         ;                   0EH         8.6        1EH        2.1               :
2470                                    <1>         ;                   0FH         8.0        1FH        2.0               :
2471                                    <1>         ;                                                                       :
2472                                    <1>         ;           (BH) = TYPAMATIC DELAY  (BITS 2 - 7 MUST BE RESET TO 0)      :
2473                                    <1>         ;                                                                      :
2474                                    <1>         ;                   REGISTER     DELAY
:
2475                                    <1>         ;                    VALUE       VALUE
:
2476                                    <1>         ;                   ------------------
:
2477                                    <1>         ;                   00H        250 ms                                    :
2478                                    <1>         ;                   01H        500 ms                                    :
2479                                    <1>         ;                   02H        750 ms                                    :
2480                                    <1>         ;                   03H       1000 ms                                    :
2481                                    <1>         ;--------------------------------------------------------------------------
:
2482                                    <1>         ;     (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD    :
2483                                    <1>         ;               BUFFER AS IF STRUCK FROM KEYBOARD                         :
2484                                    <1>         ;               ENTRY:  (CL) = ASCII CHARACTER                           :
2485                                    <1>         ;                       (CH) = SCAN CODE                                  :
2486                                    <1>         ;               EXIT:   (AH) = 00H = SUCCESSFUL OPERATION                 :
2487                                    <1>         ;                       (AL) = 01H = UNSUCCESSFUL - BUFFER FULL           :
2488                                    <1>         ;               FLAGS:  CARRY IF ERROR                                    :
2489                                    <1>         ;--------------------------------------------------------------------------
:
2490                                    <1>         ;     (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,         :
2491                                    <1>         ;               OTHERWISE SAME AS FUNCTION AH=0                          :
2492                                    <1>         ;--------------------------------------------------------------------------
:
2493                                    <1>         ;     (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,           :
2494                                    <1>         ;               OTHERWISE SAME AS FUNCTION AH=1                          :
2495                                    <1>         ;--------------------------------------------------------------------------
:
2496                                    <1>         ;     (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER            :
2497                                    <1>         ;               AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT      :
2498                                    <1>         ;               CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3             :
2499                                    <1>         ; OUTPUT                                                                 :
2500                                    <1>         ;     AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED                        :
2501                                    <1>         ;     ALL REGISTERS RETAINED                                              :
```

```
2502                            <1>     ;--------------------------------------------------------------------------------
-
2503                            <1>
2504 00000CA5 FB               <1>     sti                     ; INTERRUPTS BACK ON
2505 00000CA6 1E               <1>     push  ds                ; SAVE CURRENT DS
2506 00000CA7 53               <1>     push  ebx               ; SAVE BX TEMPORARILY
2507                            <1>     ;push ecx               ; SAVE CX TEMPORARILY
2508 00000CA8 66BB1000         <1>       mov   bx, KDATA
2509 00000CAC 8EDB             <1>     mov   ds, bx            ; PUT SEGMENT VALUE OF DATA AREA INTO DS
2510 00000CAE 08E4             <1>     or    ah, ah            ; CHECK FOR (AH)= 00H
2511 00000CB0 7439             <1>     jz    short _K1         ; ASCII_READ
2512 00000CB2 FECC             <1>     dec   ah                    ; CHECK FOR (AH)= 01H
2513 00000CB4 7452             <1>       jz    short _K2             ; ASCII_STATUS
2514 00000CB6 FECC             <1>     dec   ah                ; CHECK FOR (AH)= 02H
2515 00000CB8 0F8485000000     <1>       jz    _K3                   ; SHIFT STATUS
2516 00000CBE FECC             <1>     dec   ah                ; CHECK FOR (AH)= 03H
2517 00000CC0 0F8484000000     <1>       jz    _K300                 ; SET TYPAMATIC RATE/DELAY
2518 00000CC6 80EC02           <1>     sub   ah, 2             ; CHECK FOR (AH)= 05H
2519 00000CC9 0F84A1000000     <1>       jz    _K500                 ; KEYBOARD WRITE
2520                            <1> _KIO1:
2521 00000CCF 80EC0B           <1>     sub   ah, 11            ; AH =  10H
2522 00000CD2 740B             <1>     jz    short _K1E        ; EXTENDED ASCII READ
2523 00000CD4 FECC             <1>     dec   ah                ; CHECK FOR (AH)= 11H
2524 00000CD6 7421             <1>     jz    short _K2E        ; EXTENDED_ASCII_STATUS
2525 00000CD8 FECC             <1>     dec   ah                ; CHECK FOR (AH)= 12H
2526 00000CDA 7449             <1>     jz    short _K3E        ; EXTENDED_SHIFT_STATUS
2527                            <1> _KIO_EXIT:
2528                            <1>     ;pop  ecx               ; RECOVER REGISTER
2529 00000CDC 5B               <1>     pop   ebx               ; RECOVER REGISTER
2530 00000CDD 1F               <1>     pop   ds                ; RECOVER SEGMENT
2531 00000CDE CF               <1>     iretd                   ; INVALID COMMAND, EXIT
2532                            <1>
2533                            <1>     ;----- ASCII CHARACTER
2534                            <1> _K1E:
2535 00000CDF E8B9000000       <1>     call  _K1S              ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
2536 00000CE4 E82E010000       <1>     call  _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
2537 00000CE9 EBF1             <1>     jmp   short _KIO_EXIT       ; GIVE IT TO THE CALLER
2538                            <1> _K1:
2539 00000CEB E8AD000000       <1>     call  _K1S              ; GET A CHARACTER FROM THE BUFFER
2540 00000CF0 E82D010000       <1>     call  _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
2541 00000CF5 72F4             <1>     jc    short _K1         ; CARRY SET MEANS TROW CODE AWAY
2542                            <1> _K1A:
2543 00000CF7 EBE3             <1>     jmp   short _KIO_EXIT       ; RETURN TO CALLER
2544                            <1>
2545                            <1>     ;----- ASCII STATUS
2546                            <1> _K2E:
2547 00000CF9 E8EA000000       <1>     call  _K2S              ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
2548 00000CFE 7420             <1>     jz    short _K2B        ; RETURN IF BUFFER EMPTY
2549 00000D00 9C               <1>     pushf                   ; SAVE ZF FROM TEST
2550 00000D01 E811010000       <1>     call  _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
2551 00000D06 EB17             <1>     jmp   short _K2A        ; GIVE IT TO THE CALLER
2552                            <1> _K2:
2553 00000D08 E8DB000000       <1>     call  _K2S              ; TEST FOR CHARACTER IN BUFFER
2554 00000D0D 7411             <1>     jz    short _K2B        ; RETURN IF BUFFER EMPTY
2555 00000D0F 9C               <1>     pushf                   ; SAVE ZF FROM TEST
2556 00000D10 E80D010000       <1>     call  _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
2557 00000D15 7308             <1>     jnc   short _K2A        ; CARRY CLEAR MEANS PASS VALID CODE
2558 00000D17 9D               <1>     popf                    ; INVALID CODE FOR THIS TYPE OF CALL
2559 00000D18 E880000000       <1>     call  _K1S              ; THROW THE CHARACTER AWAY
2560 00000D1D EBE9             <1>     jmp   short _K2         ; GO LOOK FOR NEXT CHAR, IF ANY
2561                            <1> _K2A:
2562 00000D1F 9D               <1>     popf                    ; RESTORE ZF FROM TEST
2563                            <1> _K2B:
2564                            <1>     ;pop  ecx               ; RECOVER REGISTER
2565 00000D20 5B               <1>     pop   ebx               ; RECOVER REGISTER
2566 00000D21 1F               <1>     pop   ds                ; RECOVER SEGMENT
2567 00000D22 CA0400           <1>     retf  4                 ; THROW AWAY (e)FLAGS
2568                            <1>
2569                            <1>     ;----- SHIFT STATUS
2570                            <1> _K3E:                           ; GET THE EXTENDED SHIFT STATUS FLAGS
2571 00000D25 8A25[546A0000]   <1>     mov   ah, [KB_FLAG_1]       ; GET SYSTEM SHIFT KEY STATUS
2572 00000D2B 80E404           <1>     and   ah, SYS_SHIFT     ; MASK ALL BUT SYS KEY BIT
2573                            <1>     ;mov  cl, 5             ; SHIFT THEW SYSTEMKEY BIT OVER TO
2574                            <1>     ;shl  ah, cl            ; BIT 7 POSITION
2575 00000D2E C0E405           <1>       shl ah, 5
2576 00000D31 A0[546A0000]     <1>     mov   al, [KB_FLAG_1]         ; GET SYSTEM SHIFT STATES BACK
2577 00000D36 2473             <1>     and   al, 01110011b     ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
2578 00000D38 08C4             <1>     or    ah, al                ; MERGE REMAINING BITS INTO AH
2579 00000D3A A0[566A0000]     <1>     mov   al, [KB_FLAG_3]         ; GET RIGHT CTL AND ALT
2580 00000D3F 240C             <1>     and   al, 00001100b     ; ELIMINATE LC_E0 AND LC_E1
2581 00000D41 08C4             <1>     or    ah, al            ; OR THE SHIFT FLAGS TOGETHER
2582                            <1> _K3:
2583 00000D43 A0[536A0000]     <1>     mov   al, [KB_FLAG]     ; GET THE SHIFT STATUS FLAGS
2584 00000D48 EB92             <1>     jmp   short _KIO_EXIT         ; RETURN TO CALLER
2585                            <1>
2586                            <1>     ;----- SET TYPAMATIC RATE AND DELAY
2587                            <1> _K300:
2588 00000D4A 3C05             <1>     cmp   al, 5             ; CORRECT FUNCTION CALL?
2589 00000D4C 758E             <1>     jne   short _KIO_EXIT         ; NO, RETURN
2590 00000D4E F6C3E0           <1>     test  bl, 0E0h          ; TEST FOR OUT-OF-RANGE RATE
2591 00000D51 7589             <1>     jnz   short _KIO_EXIT         ; RETURN IF SO
2592 00000D53 F6C7FC           <1>     test  BH, 0FCh          ; TEST FOR OUT-OF-RANGE DELAY
2593 00000D56 7584             <1>     jnz   short _KIO_EXIT         ; RETURN IF SO
2594 00000D58 B0F3             <1>     mov   al, KB_TYPA_RD          ; COMMAND FOR TYPAMATIC RATE/DELAY

2595 00000D5A E8B6060000       <1>     call  SND_DATA          ; SEND TO KEYBOARD
2596                            <1>     ;mov  cx, 5             ; SHIFT COUNT
2597                            <1>     ;shl  bh, cl            ; SHIFT DELAY OVER
2598 00000D5F C0E705           <1>     shl   bh, 5
2599 00000D62 88D8             <1>     mov   al, bl            ; PUT IN RATE
2600 00000D64 08F8             <1>     or    al, bh            ; AND DELAY
2601 00000D66 E8AA060000       <1>     call  SND_DATA          ; SEND TO KEYBOARD
2602 00000D6B E96CFFFFFF       <1>       jmp   _KIO_EXIT             ; RETURN TO CALLER
2603                            <1>
2604                            <1>     ;----- WRITE TO KEYBOARD BUFFER
```

```
2605                                    <1> _K500:
2606 00000D70 56                        <1>         push   esi                  ; SAVE SI (esi)
2607 00000D71 FA                        <1>         cli                         ;
2608 00000D72 8B1D[646A0000]            <1>         mov    ebx, [BUFFER_TAIL]   ; GET THE 'IN TO' POINTER TO THE BUFFER
2609 00000D78 89DE                      <1>         mov    esi, ebx             ; SAVE A COPY IN CASE BUFFER NOT FULL
2610 00000D7A E8D3000000                <1>         call   _K4                  ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
2611 00000D7F 3B1D[606A0000]            <1>         cmp    ebx, [BUFFER_HEAD]   ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
2612 00000D85 740D                      <1>         je     short _K502          ; YES - INFORM CALLER OF ERROR
2613 00000D87 66890E                    <1>         mov    [esi], cx            ; NO - PUT ASCII/SCAN CODE INTO BUFFER
2614 00000D8A 891D[646A0000]            <1>         mov    [BUFFER_TAIL], ebx   ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
2615 00000D90 28C0                      <1>         sub    al, al               ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
2616 00000D92 EB02                      <1>         jmp    short _K504          ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
2617                                    <1> _K502:
2618 00000D94 B001                      <1>         mov    al, 01h                    ; BUFFER FULL INDICATION
2619                                    <1> _K504:
2620 00000D96 FB                        <1>         sti
2621 00000D97 5E                        <1>         pop    esi                  ; RECOVER SI (esi)
2622 00000D98 E93FFFFFFF                <1>           jmp     _KIO_EXIT             ; RETURN TO CALLER WITH STATUS IN AL
2623                                    <1>
2624                                    <1>         ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
2625                                    <1> _K1S:
2626 00000D9D FA                        <1>         cli    ; 03/12/2014
2627 00000D9E 8B1D[606A0000]            <1>           mov     ebx, [BUFFER_HEAD]     ; GET POINTER TO HEAD OF BUFFER
2628 00000DA4 3B1D[646A0000]            <1>           cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
2629                                    <1>         ;jne   short _K1U          ; IF ANYTHING IN BUFFER SKIP INTERRUPT
2630 00000DAA 750F                      <1>         jne    short _k1x ; 03/12/2014
2631                                    <1>         ;
2632                                    <1>         ; 03/12/2014
2633                                    <1>         ; 28/08/2014
2634                                    <1>         ; PERFORM OTHER FUNCTION ?? here !
2635                                    <1>         ;; MOV AX, 9002h           ; MOVE IN WAIT CODE & TYPE
2636                                    <1>         ;; INT       15H                ; PERFORM OTHER FUNCTION
2637                                    <1> _K1T:                                   ; ASCII READ
2638 00000DAC FB                        <1>         sti                        ; INTERRUPTS BACK ON DURING LOOP
2639 00000DAD 90                        <1>         nop                        ; ALLOW AN INTERRUPT TO OCCUR
2640                                    <1> _K1U:
2641 00000DAE FA                        <1>         cli                        ; INTERRUPTS BACK OFF
2642 00000DAF 8B1D[606A0000]            <1>           mov        ebx, [BUFFER_HEAD]  ; GET POINTER TO HEAD OF BUFFER
2643 00000DB5 3B1D[646A0000]            <1>           cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
2644                                    <1> _k1x:
2645 00000DBB 53                        <1>         push   ebx                 ; SAVE ADDRESS
2646 00000DBC 9C                        <1>         pushf                      ; SAVE FLAGS
2647 00000DBD E80B070000                <1>         call   MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
2648 00000DC2 8A1D[556A0000]            <1>         mov    bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
2649 00000DC8 30C3                      <1>         xor    bl, al              ; SEE IF ANY DIFFERENT
2650 00000DCA 80E307                    <1>         and    bl, 07h      ; KB_LEDS    ; ISOLATE INDICATOR BITS
2651 00000DCD 7406                      <1>         jz     short _K1V          ; IF NO CHANGE BYPASS UPDATE
2652 00000DCF E8A5060000                <1>         call   SND_LED1
2653 00000DD4 FA                        <1>         cli                        ; DISABLE INTERRUPTS
2654                                    <1> _K1V:
2655 00000DD5 9D                        <1>         popf                       ; RESTORE FLAGS
2656 00000DD6 5B                        <1>         pop    ebx                 ; RESTORE ADDRESS
2657 00000DD7 74D3                      <1>           je      short _K1T             ; LOOP UNTIL SOMETHING IN BUFFER
2658                                    <1>         ;
2659 00000DD9 668B03                    <1>         mov    ax, [ebx]           ; GET SCAN CODE AND ASCII CODE
2660 00000DDC E871000000                <1>           call    _K4                    ; MOVE POINTER TO NEXT POSITION
2661 00000DE1 891D[606A0000]            <1>           mov     [BUFFER_HEAD], ebx     ; STORE VALUE IN VARIABLE
2662 00000DE7 C3                        <1>         retn                       ; RETURN
2663                                    <1>
2664                                    <1>         ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
2665                                    <1> _K2S:
2666 00000DE8 FA                        <1>         cli                        ; INTERRUPTS OFF
2667 00000DE9 8B1D[606A0000]            <1>           mov     ebx, [BUFFER_HEAD]     ; GET HEAD POINTER
2668 00000DEF 3B1D[646A0000]            <1>           cmp     ebx, [BUFFER_TAIL]     ; IF EQUAL (Z=1) THEN NOTHING THERE
2669 00000DF5 668B03                    <1>         mov    ax, [ebx]
2670 00000DF8 9C                        <1>         pushf                      ; SAVE FLAGS
2671 00000DF9 6650                      <1>         push   ax                  ; SAVE CODE
2672 00000DFB E8CD060000                <1>         call   MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
2673 00000E00 8A1D[556A0000]            <1>         mov    bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
2674 00000E06 30C3                      <1>         xor    bl, al              ; SEE IF ANY DIFFERENT
2675 00000E08 80E307                    <1>         and    bl, 07h ; KB_LEDS   ; ISOLATE INDICATOR BITS
2676 00000E0B 7405                      <1>         jz     short _K2T          ; IF NO CHANGE BYPASS UPDATE
2677 00000E0D E850060000                <1>         call   SND_LED                  ; GO TURN ON MODE INDICATORS
2678                                    <1> _K2T:
2679 00000E12 6658                      <1>         pop    ax                  ; RESTORE CODE
2680 00000E14 9D                        <1>         popf                       ; RESTORE FLAGS
2681 00000E15 FB                        <1>         sti                        ; INTERRUPTS BACK ON
2682 00000E16 C3                        <1>         retn                       ; RETURN
2683                                    <1>
2684                                    <1>         ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
2685                                    <1> _KIO_E_XLAT:
2686 00000E17 3CF0                      <1>         cmp    al, 0F0h            ; IS IT ONE OF THE FILL-INs?
2687 00000E19 7506                      <1>         jne    short _KIO_E_RET    ; NO, PASS IT ON
2688 00000E1B 08E4                      <1>           or   ah, ah              ; AH = 0 IS SPECIAL CASE
2689 00000E1D 7402                      <1>           jz   short _KIO_E_RET        ; PASS THIS ON UNCHANGED
2690 00000E1F 30C0                      <1>         xor    al, al              ; OTHERWISE SET AL = 0
2691                                    <1> _KIO_E_RET:
2692 00000E21 C3                        <1>         retn                       ; GO BACK
2693                                    <1>
2694                                    <1>         ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
2695                                    <1> _KIO_S_XLAT:
2696 00000E22 80FCE0                    <1>         cmp    ah, 0E0h            ; IS IT KEYPAD ENTER OR / ?
2697 00000E25 750F                      <1>         jne    short _KIO_S2       ; NO, CONTINUE
2698 00000E27 3C0D                      <1>         cmp    al, 0Dh                  ; KEYPAD ENTER CODE?
2699 00000E29 7408                      <1>           je   short _KIO_S1       ; YES, MASSAGE A BIT
2700 00000E2B 3C0A                      <1>         cmp    al, 0Ah                  ; CTRL KEYPAD ENTER CODE?
2701 00000E2D 7404                      <1>           je   short _KIO_S1       ; YES, MASSAGE THE SAME
2702 00000E2F B435                      <1>         mov    ah, 35h                  ; NO, MUST BE KEYPAD /
2703                                    <1> _kio_ret: ; 03/12/2014
2704 00000E31 F8                        <1>         clc
2705 00000E32 C3                        <1>         retn
2706                                    <1>         ;jmp   short _KIO_USE           ; GIVE TO CALLER
2707                                    <1> _KIO_S1:
2708 00000E33 B41C                      <1>         mov    ah, 1Ch                  ; CONVERT TO COMPATIBLE OUTPUT
2709                                    <1>         ;jmp   short _KIO_USE           ; GIVE TO CALLER
```

```
2710 00000E35 C3                <1>       retn
2711                            <1> _KIO_S2:
2712 00000E36 80FC84            <1>       cmp    ah, 84h                  ; IS IT ONE OF EXTENDED ONES?
2713 00000E39 7715              <1>       ja     short _KIO_DIS           ; YES, THROW AWAY AND GET ANOTHER CHAR
2714 00000E3B 3CF0              <1>       cmp    al, 0F0h           ; IS IT ONE OF THE FILL-INs?
2715 00000E3D 7506              <1>         jne  short _KIO_S3      ; NO, TRY LAST TEST
2716 00000E3F 08E4              <1>       or     ah, ah             ; AH = 0 IS SPECIAL CASE
2717 00000E41 740C              <1>         jz   short _KIO_USE           ; PASS THIS ON UNCHANGED
2718 00000E43 EB0B              <1>       jmp    short _KIO_DIS           ; THROW AWAY THE REST
2719                            <1> _KIO_S3:
2720 00000E45 3CE0              <1>       cmp    al, 0E0h           ; IS IT AN EXTENSION OF A PREVIOUS ONE?
2721                            <1>       ;jne   short _KIO_USE           ; NO, MUST BE A STANDARD CODE
2722 00000E47 75E8              <1>       jne    short _kio_ret
2723 00000E49 08E4              <1>       or     ah, ah             ; AH = 0 IS SPECIAL CASE
2724 00000E4B 7402              <1>         jz   short _KIO_USE           ; JUMP IF AH = 0
2725 00000E4D 30C0              <1>       xor    al, al             ; CONVERT TO COMPATIBLE OUTPUT
2726                            <1>       ;jmp   short _KIO_USE           ; PASS IT ON TO CALLER
2727                            <1> _KIO_USE:
2728                            <1>       ;clc                      ; CLEAR CARRY TO INDICATE GOOD CODE
2729 00000E4F C3                <1>       retn                      ; RETURN
2730                            <1> _KIO_DIS:
2731 00000E50 F9                <1>       stc                       ; SET CARRY TO INDICATE DISCARD CODE
2732 00000E51 C3                <1>       retn                      ; RETURN
2733                            <1>
2734                            <1>       ;----- INCREMENT BUFFER POINTER ROUTINE -----
2735                            <1> _K4:
2736 00000E52 43                <1>       inc     ebx
2737 00000E53 43                <1>       inc    ebx                ; MOVE TO NEXT WORD IN LIST
2738 00000E54 3B1D[5C6A0000]    <1>         cmp     ebx, [BUFFER_END]      ; AT END OF BUFFER?
2739                            <1>         ;jne    short _K5               ; NO, CONTINUE
2740 00000E5A 7206              <1>       jb     short _K5
2741 00000E5C 8B1D[586A0000]    <1>         mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
2742                            <1> _K5:
2743 00000E62 C3                <1>       retn
2744                            <1>
2745                            <1> ; 20/02/2015
2746                            <1> ; 05/12/2014
2747                            <1> ; 26/08/2014
2748                            <1> ; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
2749                            <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
2750                            <1> ;
2751                            <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
2752                            <1> ; rombios source code (06/10/1985)
2753                            <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
2754                            <1>
2755                            <1> ;--------- 8042 COMMANDS ----------------------------------------------------
2756                            <1> ENA_KBD            equ    0AEh   ; ENABLE KEYBOARD COMMAND
2757                            <1> DIS_KBD            equ    0ADh   ; DISABLE KEYBOARD COMMAND
2758                            <1> SHUT_CMD    equ    0FEh   ; CAUSE A SHUTDOWN COMMAND
2759                            <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
2760                            <1> STATUS_PORT  equ    064h   ; 8042 STATUS PORT
2761                            <1> INPT_BUF_FULL      equ    00000010b ; 1 = +INPUT BUFFER FULL
2762                            <1> PORT_A             equ    060h   ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
2763                            <1> ;---------- 8042 KEYBOARD RESPONSE -----------------------------------------
2764                            <1> KB_ACK             equ    0FAh   ; ACKNOWLEDGE PROM TRANSMISSION
2765                            <1> KB_RESEND   equ    0FEh   ; RESEND REQUEST
2766                            <1> KB_OVER_RUN  equ    0FFh   ; OVER RUN SCAN CODE
2767                            <1> ;---------- KEYBOARD/LED COMMANDS -------------------------------------------
2768                            <1> KB_ENABLE   equ    0F4h         ; KEYBOARD ENABLE
2769                            <1> LED_CMD            equ    0EDh         ; LED WRITE COMMAND
2770                            <1> KB_TYPA_RD  equ    0F3h         ; TYPAMATIC RATE/DELAY COMMAND
2771                            <1> ;---------- KEYBOARD SCAN CODES ---------------------------------------------
2772                            <1> NUM_KEY            equ    69           ; SCAN CODE FOR      NUMBER LOCK KEY
2773                            <1> SCROLL_KEY  equ    70           ; SCAN CODE FOR      SCROLL LOCK KEY
2774                            <1> ALT_KEY            equ    56           ; SCAN CODE FOR      ALTERNATE SHIFT KEY
2775                            <1> CTL_KEY            equ    29           ; SCAN CODE FOR      CONTROL KEY
2776                            <1> CAPS_KEY    equ    58           ; SCAN CODE FOR      SHIFT LOCK KEY
2777                            <1> DEL_KEY            equ    83           ; SCAN CODE FOR      DELETE KEY
2778                            <1> INS_KEY            equ    82           ; SCAN CODE FOR      INSERT KEY
2779                            <1> LEFT_KEY    equ    42           ; SCAN CODE FOR      LEFT SHIFT
2780                            <1> RIGHT_KEY   equ    54           ; SCAN CODE FOR      RIGHT SHIFT
2781                            <1> SYS_KEY            equ    84           ; SCAN CODE FOR      SYSTEM KEY
2782                            <1> ;---------- ENHANCED KEYBOARD SCAN CODES ------------------------------------
2783                            <1> ID_1        equ    0ABh         ; 1ST ID CHARACTER FOR KBX
2784                            <1> ID_2        equ    041h         ; 2ND ID CHARACTER FOR KBX
2785                            <1> ID_2A       equ    054h         ; ALTERNATE 2ND ID CHARACTER FOR KBX
2786                            <1> F11_M       equ    87           ; F11 KEY MAKE
2787                            <1> F12_M       equ    88           ; F12 KEY MAKE
2788                            <1> MC_E0       equ    224          ; GENERAL MARKER CODE
2789                            <1> MC_E1       equ    225          ; PAUSE KEY MARKER CODE
2790                            <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG-------------------------------------
2791                            <1> RIGHT_SHIFT  equ    00000001b    ; RIGHT SHIFT KEY DEPRESSED
2792                            <1> LEFT_SHIFT   equ    00000010b    ; LEFT SHIFT KEY DEPRESSED
2793                            <1> CTL_SHIFT    equ    00000100b    ; CONTROL SHIFT KEY DEPRESSED
2794                            <1> ALT_SHIFT    equ    00001000b    ; ALTERNATE SHIFT KEY DEPRESSED
2795                            <1> SCROLL_STATE equ    00010000b    ; SCROLL LOCK STATE IS ACTIVE
2796                            <1> NUM_STATE    equ    00100000b    ; NUM LOCK STATE IS ACTIVE
2797                            <1> CAPS_STATE   equ    01000000b    ; CAPS LOCK STATE IS ACTIVE
2798                            <1> INS_STATE    equ    10000000b    ; INSERT STATE IS ACTIVE
2799                            <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG_1 -----------------------------------
2800                            <1> L_CTL_SHIFT  equ    00000001b    ; LEFT CTL KEY DOWN
2801                            <1> L_ALT_SHIFT  equ    00000010b    ; LEFT ALT KEY DOWN
2802                            <1> SYS_SHIFT    equ    00000100b    ; SYSTEM KEY DEPRESSED AND HELD
2803                            <1> HOLD_STATE   equ    00001000b    ; SUSPEND KEY HAS BEEN TOGGLED
2804                            <1> SCROLL_SHIFT equ    00010000b    ; SCROLL LOCK KEY IS DEPRESSED
2805                            <1> NUM_SHIFT    equ    00100000b    ; NUM LOCK KEY IS DEPRESSED
2806                            <1> CAPS_SHIFT   equ    01000000b    ; CAPS LOCK KEY IS DEPRE55ED
2807                            <1> INS_SHIFT    equ    10000000b    ; INSERT KEY IS DEPRESSED
2808                            <1> ;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 ----------------------------------
2809                            <1> KB_LEDS            equ    00000111b    ; KEYBOARD LED STATE BITS
2810                            <1> ;           equ    00000001b    ; SCROLL LOCK INDICATOR
2811                            <1> ;           equ    00000010b    ; NUM LOCK INDICATOR
2812                            <1> ;           equ    00000100b    ; CAPS LOCK INDICATOR
2813                            <1> ;           equ    00001000b    ; RESERVED (MUST BE ZERO)
2814                            <1> KB_FA       equ    00010000b    ; ACKNOWLEDGMENT RECEIVED
```

```
2815                            <1> KB_FE       equ    00100000b    ; RESEND RECEIVED FLAG
2816                            <1> KB_PR_LED   equ    01000000b    ; MODE INDICATOR UPDATE
2817                            <1> KB_ERR              equ    10000000b    ; KEYBOARD TRANSMIT ERROR FLAG
2818                            <1> ;----------- FLAGS EQUATES WITHIN @KB_FLAG_3 ----------------------------------
2819                            <1> LC_E1       equ    00000001b    ; LAST CODE WAS THE E1 HIDDEN CODE
2820                            <1> LC_E0       equ    00000010b    ; LAST CODE WAS THE E0 HIDDEN CODE
2821                            <1> R_CTL_SHIFT equ    00000100b    ; RIGHT CTL KEY DOWN
2822                            <1> R_ALT_SHIFT equ    00001000b    ; RIGHT ALT KEY DOWN
2823                            <1> GRAPH_ON    equ    00001000b    ; ALT GRAPHICS KEY DOWN (WT ONLY)
2824                            <1> KBX         equ    00010000b    ; ENHANCED KEYBOARD INSTALLED
2825                            <1> SET_NUM_LK  equ    00100000b    ; FORCE NUM LOCK IF READ ID AND KBX
2826                            <1> LC_AB       equ    01000000b    ; LAST CHARACTER WAS FIRST ID CHARACTER
2827                            <1> RD_ID       equ    10000000b    ; DOING A READ ID (MUST BE BIT0)
2828                            <1> ;
2829                            <1> ;----------- INTERRUPT EQUATES -----------------------------------------------
2830                            <1> EOI         equ    020h         ; END OF INTERRUPT COMMAND TO 8259
2831                            <1> INTA00              equ    020h         ; 8259 PORT
2832                            <1>
2833                            <1>
2834                            <1> kb_int:
2835                            <1>
2836                            <1> ; 17/10/2015 ('ctrlbrk')
2837                            <1> ; 05/12/2014
2838                            <1> ; 04/12/2014 (derivation from pc-xt-286 bios source code -1986-,
2839                            <1>       ;           instead of pc-at bios - 1985-)
2840                            <1> ; 26/08/2014
2841                            <1> ;
2842                            <1> ; 03/06/86  KEYBOARD BIOS
2843                            <1> ;
2844                            <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) ----------------------------------------
2845                            <1> ;                                                                            ;
2846                            <1> ;     KEYBOARD INTERRUPT ROUTINE                                             ;
2847                            <1> ;                                                                            ;
2848                            <1> ;--------------------------------------------------------------------------------
2849                            <1>
2850                            <1> KB_INT_1:
2851 00000E63 FB                <1>       sti                       ; ENABLE INTERRUPTS
2852                            <1>       ;push  ebp
2853 00000E64 50                <1>       push   eax
2854 00000E65 53                <1>       push   ebx
2855 00000E66 51                <1>       push   ecx
2856 00000E67 52                <1>       push   edx
2857 00000E68 56                <1>       push   esi
2858 00000E69 57                <1>       push   edi
2859 00000E6A 1E                <1>       push   ds
2860 00000E6B 06                <1>       push   es
2861 00000E6C FC                <1>       cld                       ; FORWARD DIRECTION
2862 00000E6D 66B81000          <1>       mov    ax, KDATA
2863 00000E71 8ED8              <1>       mov    ds, ax
2864 00000E73 8EC0              <1>       mov    es, ax
2865                            <1>       ;
2866                            <1>       ;-----WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
2867 00000E75 B0AD              <1>       mov    al, DIS_KBD        ; DISABLE THE KEYBOARD COMMAND
2868 00000E77 E885050000        <1>       call   SHIP_IT                 ; EXECUTE DISABLE
2869 00000E7C FA                <1>       cli                       ; DISABLE INTERRUPTS
2870 00000E7D B900000100        <1>       mov    ecx, 10000h        ; SET MAXIMUM TIMEOUT
2871                            <1> KB_INT_01:
2872 00000E82 E464              <1>       in     al, STATUS_PORT           ; READ ADAPTER STATUS
2873 00000E84 A802              <1>       test   al, INPT_BUF_FULL  ; CHECK INPUT BUFFER FULL STATUS BIT
2874 00000E86 E0FA              <1>       loopnz KB_INT_01          ; WAIT FOR COMMAND TO BE ACCEPTED
2875                            <1>       ;
2876                            <1>       ;-----READ CHARACTER FROM KEYBOARD INTERFACE
2877 00000E88 E460              <1>       in     al, PORT_A         ; READ IN THE CHARACTER
2878                            <1>       ;
2879                            <1>       ;-----SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
2880                            <1>       ;MOV   AH, 04FH           ; SYSTEM INTERCEPT - KEY CODE FUNCTION
2881                            <1>       ;STC                      ; SET CY=1 (IN CASE OF IRET)
2882                            <1>       ;INT   15H                ; CASETTE CALL (AL)=KEY SCAN CODE
2883                            <1>       ;                         ; RETURNS CY=1 FOR INVALID FUNCTION
2884                            <1>       ;JC    KB_INT_02          ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
2885                            <1>       ;JMP   K26                ; EXIT IF SYSTEM HANDLES SCAN CODE
2886                            <1>       ;                         ; EXIT HANDLES HARDWARE EOI AND ENABLE
2887                            <1>       ;
2888                            <1>       ;-----CHECK FOR A RESEND COMMAND TO KEYBOARD
2889                            <1> KB_INT_02:                      ;       (AL)= SCAN CODE
2890 00000E8A FB                <1>       sti                       ; ENABLE INTERRUPTS AGAIN
2891 00000E8B 3CFE              <1>       cmp    al, KB_RESEND      ; IS THE INPUT A RESEND
2892 00000E8D 7411              <1>         je     short KB_INT_4          ; GO IF RESEND
2893                            <1>       ;
2894                            <1>       ;-----CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
2895 00000E8F 3CFA              <1>       cmp    al, KB_ACK         ; IS THE INPUT AN ACKNOWLEDGE
2896 00000E91 751A              <1>         jne    short KB_INT_2          ; GO IF NOT
2897                            <1>       ;
2898                            <1>       ;-----A COMMAND TO THE KEYBOARD WAS ISSUED
2899 00000E93 FA                <1>       cli                       ; DISABLE INTERRUPTS
2900 00000E94 800D[556A0000]10  <1>       or     byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
2901 00000E9B E97A020000        <1>         jmp    K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
2902                            <1>       ;
2903                            <1>       ;-----RESEND THE LAST BYTE
2904                            <1> KB_INT_4:
2905 00000EA0 FA                <1>       cli                       ; DISABLE INTERRUPTS
2906 00000EA1 800D[556A0000]20  <1>       or     byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
2907 00000EA8 E96D020000        <1>         jmp    K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
2908                            <1>       ;
2909                            <1> ;-----      UPDATE MODE INDICATORS IF CHANGE IN STATE
2910                            <1> KB_INT_2:
2911 00000EAD 6650              <1>       push   ax                 ; SAVE DATA IN
2912 00000EAF E819060000        <1>       call   MAKE_LED           ; GO GET MODE INDICATOR DATA BYTE
2913 00000EB4 8A1D[556A0000]    <1>       mov    bl, [KB_FLAG_2]    ; GET PREVIOUS BITS
2914 00000EBA 30C3              <1>       xor    bl, al             ; SEE IF ANY DIFFERENT
2915 00000EBC 80E307            <1>       and    bl, KB_LEDS        ; ISOLATE INDICATOR BITS
2916 00000EBF 7405              <1>       jz     short UP0          ; IF NO CHANGE BYPASS UPDATE
2917 00000EC1 E89C050000        <1>       call   SND_LED                ; GO TURN ON MODE INDICATORS
2918                            <1> UP0:
2919 00000EC6 6658              <1>       pop    ax                 ; RESTORE DATA IN
```

```
2920                                    <1> ;-----------------------------------------------------------------------
2921                                    <1> ;     START OF KEY PROCESSING                                         ;
2922                                    <1> ;-----------------------------------------------------------------------
2923 00000EC8 88C4                      <1>       mov    ah, al              ; SAVE SCAN CODE IN AH ALSO
2924                                    <1>       ;
2925                                    <1>       ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
2926 00000ECA 3CFF                      <1>       cmp    al, KB_OVER_RUN           ; IS THIS AN OVERRUN CHAR
2927 00000ECC 0F841B050000              <1>         je      K62                     ; BUFFER_FULL_BEEP
2928                                    <1>       ;
2929                                    <1> K16:
2930 00000ED2 8A3D[566A0000]            <1>       mov    bh, [KB_FLAG_3]           ; LOAD FLAGS FOR TESTING
2931                                    <1>       ;
2932                                    <1>       ;----- TEST TO SEE IF A READ_ID IS IN PROGRESS
2933 00000ED8 F6C7C0                    <1>       test   bh, RD_ID+LC_AB     ; ARE WE DOING A READ ID?
2934 00000EDB 7449                      <1>       jz     short NOT_ID        ; CONTINUE IF NOT
2935 00000EDD 7917                      <1>       jns    short TST_ID_2            ; IS THE RD_ID FLAG ON?
2936 00000EDF 3CAB                      <1>       cmp    al, ID_1            ; IS THIS THE 1ST ID CHARACTER?
2937 00000EE1 7507                      <1>       jne    short RST_RD_ID
2938 00000EE3 800D[566A0000]40          <1>       or     byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
2939                                    <1> RST_RD_ID:
2940 00000EEA 8025[566A0000]7F          <1>       and    byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
2941                                    <1>         ;jmp    short ID_EX            ; AND EXIT
2942 00000EF1 E924020000                <1>       jmp    K26
2943                                    <1>       ;
2944                                    <1> TST_ID_2:
2945 00000EF6 8025[566A0000]BF          <1>       and    byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
2946 00000EFD 3C54                      <1>       cmp    al, ID_2A           ; IS THIS THE 2ND ID CHARACTER?
2947 00000EFF 7419                      <1>         je   short KX_BIT        ; JUMP IF SO
2948 00000F01 3C41                      <1>       cmp    al, ID_2            ; IS THIS THE 2ND ID CHARACTER?
2949                                    <1>         ;jne short ID_EX         ; LEAVE IF NOT
2950 00000F03 0F8511020000              <1>       jne    K26
2951                                    <1>       ;
2952                                    <1>       ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
2953 00000F09 F6C720                    <1>       test   bh, SET_NUM_LK            ; SHOULD WE SET NUM LOCK?
2954 00000F0C 740C                      <1>         jz      short KX_BIT           ; EXIT IF NOT
2955 00000F0E 800D[536A0000]20          <1>       or     byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
2956 00000F15 E848050000                <1>       call   SND_LED                  ; GO SET THE NUM LOCK INDICATOR
2957                                    <1> KX_BIT:
2958 00000F1A 800D[566A0000]10          <1>       or     byte [KB_FLAG_3], KBX     ; INDICATE ENHANCED KEYBOARD WAS FOUND
2959 00000F21 E9F4010000                <1> ID_EX:       jmp     K26              ; EXIT
2960                                    <1>       ;
2961                                    <1> NOT_ID:
2962 00000F26 3CE0                      <1>       cmp    al, MC_E0           ; IS THIS THE GENERAL MARKER CODE?
2963 00000F28 750C                      <1>       jne    short TEST_E1
2964 00000F2A 800D[566A0000]12          <1>       or     byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
2965                                    <1>       ;jmp   short EXIT          ; THROW AWAY THIS CODE
2966 00000F31 E9EB010000                <1>       jmp    K26A
2967                                    <1> TEST_E1:
2968 00000F36 3CE1                      <1>       cmp    al, MC_E1           ; IS THIS THE PAUSE KEY?
2969 00000F38 750C                      <1>       jne    short NOT_HC
2970 00000F3A 800D[566A0000]11          <1>       or     byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
2971 00000F41 E9DB010000                <1> EXIT: jmp    K26A                ; THROW AWAY THIS CODE
2972                                    <1>       ;
2973                                    <1> NOT_HC:
2974 00000F46 247F                      <1>       and    al, 07Fh            ; TURN OFF THE BREAK BIT
2975 00000F48 F6C702                    <1>       test   bh, LC_E0           ; LAST CODE THE E0 MARKER CODE
2976 00000F4B 7414                      <1>       jz     short NOT_LC_E0           ; JUMP IF NOT
2977                                    <1>       ;
2978 00000F4D BF[3E690000]              <1>       mov    edi, _K6+6          ; IS THIS A SHIFT KEY?
2979 00000F52 AE                        <1>       scasb
2980 00000F53 0F84C1010000              <1>         je      K26 ; K16B              ; YES, THROW AWAY & RESET FLAG
2981 00000F59 AE                        <1>       scasb
2982 00000F5A 757C                      <1>       jne    short K16A          ; NO, CONTINUE KEY PROCESSING
2983                                    <1>       ;jmp   short K16B          ; YES, THROW AWAY & RESET FLAG
2984 00000F5C E9B9010000                <1>       jmp    K26
2985                                    <1>       ;
2986                                    <1> NOT_LC_E0:
2987 00000F61 F6C701                    <1>       test   bh, LC_E1           ; LAST CODE THE E1 MARKER CODE?
2988 00000F64 7435                      <1>       jz     short T_SYS_KEY           ; JUMP IF NOT
2989 00000F66 B904000000                <1>       mov    ecx, 4              ; LENGHT OF SEARCH
2990 00000F6B BF[3C690000]              <1>       mov    edi, _K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
2991 00000F70 F2AE                      <1>       repne  scasb               ; CHECK IT
2992                                    <1>       ;je    short EXIT          ; THROW AWAY IF SO
2993 00000F72 0F84A9010000              <1>       je     K26A
2994                                    <1>       ;
2995 00000F78 3C45                      <1>       cmp    al, NUM_KEY         ; IS IT THE PAUSE KEY?
2996                                    <1>       ;jne   short K16B          ; NO, THROW AWAY & RESET FLAG
2997 00000F7A 0F859A010000              <1>       jne    K26
2998 00000F80 F6C480                    <1>       test   ah, 80h                   ; YES, IS IT THE BREAK OF THE KEY?
2999                                    <1>       ;jnz   short K16B          ; YES, THROW THIS AWAY, TOO
3000 00000F83 0F8591010000              <1>       jnz    K26
3001                                    <1>         ; 20/02/2015
3002 00000F89 F605[546A0000]08          <1>       test   byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
3003                                    <1>       ;jnz   short K16B          ;  YES, THROW AWAY
3004 00000F90 0F8584010000              <1>       jnz    K26
3005 00000F96 E9E1020000                <1>       jmp     K39P                    ; NO, THIS IS THE REAL PAUSE STATE
3006                                    <1>       ;
3007                                    <1>       ;----- TEST FOR SYSTEM KEY
3008                                    <1> T_SYS_KEY:
3009 00000F9B 3C54                      <1>       cmp    al, SYS_KEY         ; IS IT THE SYSTEM KEY?
3010 00000F9D 7539                      <1>       jnz    short K16A          ; CONTINUE IF NOT
3011                                    <1>       ;
3012 00000F9F F6C480                    <1>       test   ah, 80h                   ; CHECK IF THIS A BREAK CODE
3013 00000FA2 7524                      <1>       jnz    short K16C          ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
3014                                    <1>       ;
3015 00000FA4 F605[546A0000]04          <1>       test   byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
3016                                    <1>         ;jnz short K16B          ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
3017 00000FAB 0F8569010000              <1>       jnz     K26
3018                                    <1>       ;
3019 00000FB1 800D[546A0000]04          <1>       or     byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
3020 00000FB8 B020                      <1>       mov    al, EOI                   ; END OF INTERRUPT COMMAND
3021 00000FBA E620                      <1>       out    20h, al ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
3022                                    <1>                                      ; INTERRUPT-RETURN-NO-EOI
3023 00000FBC B0AE                      <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
3024 00000FBE E83E040000                <1>       call   SHIP_IT                   ; EXECUTE ENABLE
```

```
3025                                    <1>       ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
3026                                    <1>       ;MOV   AL, 8500H           ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
3027                                    <1>       ;STI                       ; MAKE SURE INTERRUPTS ENABLED
3028                                    <1>       ;INT   15H                 ; USER INTERRUPT
3029 00000FC3 E965010000                <1>         jmp     K27A                    ; END PROCESSING
3030                                    <1>       ;
3031                                    <1> ;K16B:      jmp    K26                ; IGNORE SYSTEM KEY
3032                                    <1>       ;
3033                                    <1> K16C:
3034 00000FC8 8025[546A0000]FB          <1>       and    byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
3035 00000FCF B020                      <1>       mov    al, EOI                  ; END OF INTERRUPT COMMAND
3036 00000FD1 E620                      <1>       out    20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
3037                                    <1>                                     ; INTERRUPT-RETURN-NO-EOI
3038                                    <1>       ;MOV   AL, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
3039                                    <1>       ;CALL  SHIP_IT                  ; EXECUTE ENABLE
3040                                    <1>       ;
3041                                    <1>       ;MOV   AX, 8501H           ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
3042                                    <1>       ;STI                       ; MAKE SURE INTERRUPTS ENABLED
3043                                    <1>       ;INT   15H                 ; USER INTERRUPT
3044                                    <1>       ;JMP   K27A                ; INGONRE SYSTEM KEY
3045                                    <1>       ;
3046 00000FD3 E94E010000                <1>       jmp     K27                ; IGNORE SYSTEM KEY
3047                                    <1>       ;
3048                                    <1>       ;----- TEST FOR SHIFT KEYS
3049                                    <1> K16A:
3050 00000FD8 8A1D[536A0000]            <1>       mov    bl, [KB_FLAG]       ; PUT STATE FLAGS IN BL
3051 00000FDE BF[38690000]              <1>       mov    edi, _K6            ; SHIFT KEY TABLE offset
3052 00000FE3 B908000000                <1>       mov    ecx, _K6L           ; LENGTH
3053 00000FE8 F2AE                      <1>       repne  scasb               ; LOOK THROUGH THE TABLE FOR A MATCH
3054 00000FEA 88E0                      <1>       mov    al, ah              ; RECOVER SCAN CODE
3055 00000FEC 0F8510010000              <1>         jne     K25                    ; IF NO MATCH, THEN SHIFT NOT FOUND
3056                                    <1>       ;
3057                                    <1>       ;------      SHIFT KEY FOUND
3058                                    <1> K17:
3059 00000FF2 81EF[39690000]            <1>         sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
3060 00000FF8 8AA7[40690000]            <1>             mov     ah, [edi+_K7]           ; GET MASK INTO AH
3061 00000FFE B102                      <1>       mov    cl, 2               ; SETUP COUNT FOR FLAG SHIFTS
3062 00001000 A880                      <1>       test   al, 80h                   ; TEST FOR BREAK KEY
3063 00001002 0F8596000000              <1>         jnz     K23                    ; JUMP OF BREAK
3064                                    <1>       ;
3065                                    <1>       ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
3066                                    <1> K17C:
3067 00001008 80FC10                    <1>       cmp    ah, SCROLL_SHIFT
3068 0000100B 732B                      <1>       jae    short K18           ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
3069                                    <1>       ;
3070                                    <1>       ;----- PLAIN SHIFT KEY, SET SHIFT ON
3071 0000100D 0825[536A0000]            <1>       or     [KB_FLAG], ah       ; TURN ON SHIFT BIT
3072 00001013 A80C                      <1>         test al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
3073                                    <1>       ;jnz   short K17D          ; YES, MORE FLAGS TO SET
3074 00001015 0F84FF000000              <1>       jz     K26                 ; NO, INTERRUPT RETURN
3075                                    <1> K17D:
3076 0000101B F6C702                    <1>       test   bh, LC_E0           ; IS THIS ONE OF NEW KEYS?
3077 0000101E 740B                      <1>       jz     short K17E          ; NO, JUMP
3078 00001020 0825[566A0000]            <1>       or     [KB_FLAG_3], ah           ; SET BITS FOR RIGHT CTRL, ALT
3079 00001026 E9EF000000                <1>       jmp    K26                 ; INTERRUPT RETURN
3080                                    <1> K17E:
3081 0000102B D2EC                      <1>       shr    ah, cl              ; MOVE FLAG BITS TWO POSITIONS
3082 0000102D 0825[546A0000]            <1>       or     [KB_FLAG_1], ah           ; SET BITS FOR LEFT CTRL, ALT
3083 00001033 E9E2000000                <1>       jmp    K26
3084                                    <1>       ;
3085                                    <1>       ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
3086                                    <1> K18:                                ; SHIFT-TOGGLE
3087 00001038 F6C304                    <1>       test   bl, CTL_SHIFT             ; CHECK CTL SHIFT STATE
3088                                    <1>         ;jz        short K18A             ; JUMP IF NOT CTL STATE
3089 0000103B 0F85C1000000              <1>         jnz     K25                    ; JUMP IF CTL STATE
3090                                    <1> K18A:
3091 00001041 3C52                      <1>       cmp    al, INS_KEY         ; CHECK FOR INSERT KEY
3092 00001043 7524                      <1>       jne    short K22           ; JUMP IF NOT INSERT KEY
3093 00001045 F6C308                    <1>       test   bl, ALT_SHIFT             ; CHECK FOR ALTERNATE SHIFT
3094                                    <1>              ;jz    short K18B          ; JUMP IF NOT ALTERNATE SHIFT
3095 00001048 0F85B4000000              <1>         jnz     K25                    ; JUMP IF ALTERNATE SHIFT
3096                                    <1> K18B:
3097 0000104E F6C702                    <1>       test   bh, LC_E0 ;20/02/2015     ; IS THIS NEW INSERT KEY?
3098 00001051 7516                      <1>       jnz    short K22           ; YES, THIS ONE'S NEVER A '0'
3099                                    <1> K19:
3100 00001053 F6C320                    <1>       test   bl, NUM_STATE             ; CHECK FOR BASE STATE
3101 00001056 750C                      <1>       jnz    short K21           ; JUMP IF NUM LOCK IS ON
3102 00001058 F6C303                    <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
3103 0000105B 740C                      <1>       jz     short K22           ; JUMP IF BASE STATE
3104                                    <1> K20:                                ; NUMERIC ZERO, NOT INSERT KEY
3105 0000105D 88C4                      <1>       mov    ah, al              ; PUT SCAN CODE BACK IN AH
3106 0000105F E99E000000                <1>         jmp     K25                    ; NUMERAL '0', STNDRD. PROCESSING
3107                                    <1> K21:                                ; MIGHT BE NUMERIC
3108 00001064 F6C303                    <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT
3109 00001067 74F4                      <1>       jz     short K20           ; IS NUMERIC, STD. PROC.
3110                                    <1>       ;
3111                                    <1> K22:                                ; SHIFT TOGGLE KEY HIT; PROCESS IT
3112 00001069 8425[546A0000]            <1>       test   ah, [KB_FLAG_1]     ; IS KEY ALREADY DEPRESSED
3113 0000106F 0F85A5000000              <1>         jnz     K26                    ; JUMP IF KEY ALREADY DEPRESSED
3114                                    <1> K22A:
3115 00001075 0825[546A0000]            <1>         or      [KB_FLAG_1], ah ; INDICATE THAT THE KEY IS DEPRESSED
3116 0000107B 3025[536A0000]            <1>       xor    [KB_FLAG], ah       ; TOGGLE THE SHIFT STATE
3117                                    <1>       ;
3118                                    <1>       ;----- TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
3119 00001081 F6C470                    <1>       test   ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
3120 00001084 7409                      <1>       jz     short K22B          ; GO IF NOT
3121                                    <1>       ;
3122 00001086 6650                      <1>       push   ax                  ; SAVE SCAN CODE AND SHIFT MASK
3123 00001088 E8D5030000                <1>       call   SND_LED                  ; GO TURN MODE INDICATORS ON
3124 0000108D 6658                      <1>       pop    ax                  ; RESTORE SCAN CODE
3125                                    <1> K22B:
3126 0000108F 3C52                      <1>       cmp    al, INS_KEY         ; TEST FOR 1ST MAKE OF INSERT KEY
3127 00001091 0F8583000000              <1>         jne     K26                    ; JUMP IF NOT INSERT KEY
3128 00001097 88C4                      <1>       mov    ah, al               ; SCAN CODE IN BOTH HALVES OF AX
3129 00001099 E999000000                <1>         jmp     K28                    ; FLAGS UPDATED, PROC. FOR BUFFER
```

```
3130                                    <1>     ;
3131                                    <1>     ;----- BREAK SHIFT FOUND
3132                                    <1> K23:                            ; BREAK-SHIFT-FOUND
3133 0000109E 80FC10                    <1>     cmp   ah, SCROLL_SHIFT      ; IS THIS A TOGGLE KEY
3134 000010A1 F6D4                      <1>     not   ah                    ; INVERT MASK
3135 000010A3 7355                      <1>     jae   short K24             ; YES, HANDLE BREAK TOGGLE
3136 000010A5 2025[536A0000]            <1>     and   [KB_FLAG], ah         ; TURN OFF SHIFT BIT
3137 000010AB 80FCFB                    <1>     cmp   ah, ~CTL_SHIFT             ; IS THIS ALT OR CTL?
3138 000010AE 7730                      <1>     ja    short K23D            ; NO, ALL DONE
3139                                    <1>     ;
3140 000010B0 F6C702                    <1>     test  bh, LC_E0             ; 2ND ALT OR CTL?
3141 000010B3 7408                      <1>     jz    short K23A            ; NO, HANSLE NORMALLY
3142 000010B5 2025[566A0000]            <1>     and   [KB_FLAG_3], ah            ; RESET BIT FOR RIGHT ALT OR CTL
3143 000010BB EB08                      <1>     jmp   short K23B            ; CONTINUE
3144                                    <1> K23A:
3145 000010BD D2FC                      <1>     sar   ah, cl                ; MOVE THE MASK BIT TWO POSITIONS
3146 000010BF 2025[546A0000]            <1>     and   [KB_FLAG_1], ah            ; RESET BIT FOR LEFT ALT AND CTL
3147                                    <1> K23B:
3148 000010C5 88C4                      <1>     mov   ah, al                ; SAVE SCAN CODE
3149 000010C7 A0[566A0000]              <1>     mov   al, [KB_FLAG_3]            ; GET RIGHT ALT & CTRL FLAGS
3150 000010CC D2E8                      <1>     shr   al, cl                ; MOVE TO BITS 1 & 0
3151 000010CE 0A05[546A0000]            <1>     or    al, [KB_FLAG_1]           ; PUT IN LEFT ALŞT & CTL FLAGS
3152 000010D4 D2E0                      <1>     shl   al, cl                ; MOVE BACK TO BITS 3 & 2
3153 000010D6 240C                      <1>     and   al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
3154 000010D8 0805[536A0000]            <1>     or    [KB_FLAG], al         ; PUT RESULT IN THE REAL FLAGS
3155 000010DE 88E0                      <1>     mov   al, ah
3156                                    <1> K23D:
3157 000010E0 3CB8                      <1>     cmp   al, ALT_KEY+80h            ; IS THIS ALTERNATE SHIFT RELEASE
3158 000010E2 7536                      <1>     jne   short K26             ; INTERRUPT RETURN
3159                                    <1>     ;
3160                                    <1>     ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
3161 000010E4 A0[576A0000]              <1>     mov   al, [ALT_INPUT]
3162 000010E9 B400                      <1>     mov   ah, 0                 ; SCAN CODE OF 0
3163 000010EB 8825[576A0000]            <1>     mov   [ALT_INPUT], ah       ; ZERO OUT THE FIELD
3164 000010F1 3C00                      <1>     cmp   al, 0                 ; WAS THE INPUT = 0?
3165 000010F3 7425                      <1>     je    short K26             ; INTERRUPT_RETURN
3166 000010F5 E9D0020000                <1>       jmp     K61                     ; IT WASN'T, SO PUT IN BUFFER
3167                                    <1>     ;
3168                                    <1> K24:                            ; BREAK-TOGGLE
3169 000010FA 2025[546A0000]            <1>     and   [KB_FLAG_1], ah       ; INDICATE NO LONGER DEPRESSED
3170 00001100 EB18                      <1>     jmp   short K26             ; INTERRUPT_RETURN
3171                                    <1>     ;
3172                                    <1>     ;----- TEST FOR HOLD STATE
3173                                    <1>                                 ; AL, AH = SCAN CODE
3174                                    <1> K25:                            ; NO-SHIFT-FOUND
3175 00001102 3C80                      <1>     cmp   al, 80h                    ; TEST FOR BREAK KEY
3176 00001104 7314                      <1>     jae   short K26             ; NOTHING FOR BREAK CHARS FROM HERE ON
3177 00001106 F605[546A0000]08          <1>     test  byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
3178 0000110D 7428                      <1>     jz    short K28             ; BRANCH AROUND TEST IF NOT
3179 0000110F 3C45                      <1>     cmp   al, NUM_KEY
3180 00001111 7407                      <1>     je    short K26             ; CAN'T END HOLD ON NUM_LOCK
3181 00001113 8025[546A0000]F7          <1>     and   byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
3182                                    <1>     ;
3183                                    <1> K26:
3184 0000111A 8025[566A0000]FC          <1>     and   byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
3185                                    <1> K26A:                           ; INTERRUPT-RETURN
3186 00001121 FA                        <1>     cli                         ; TURN OFF INTERRUPTS
3187 00001122 B020                      <1>     mov   al, EOI                    ; END OF INTERRUPT COMMAND
3188 00001124 E620                      <1>     out   20h, al      ;out INTA00, al     ; SEND COMMAND TO INTERRUPT CONTROL
PORT
3189                                    <1> K27:                            ; INTERRUPT-RETURN-NO-EOI
3190 00001126 B0AE                      <1>     mov   al, ENA_KBD           ; INSURE KEYBOARD IS ENABLED
3191 00001128 E8D4020000                <1>     call  SHIP_IT                    ; EXECUTE ENABLE
3192                                    <1> K27A:
3193 0000112D FA                        <1>     cli                         ; DISABLE INTERRUPTS
3194 0000112E 07                        <1>     pop   es                    ; RESTORE REGISTERS
3195 0000112F 1F                        <1>     pop   ds
3196 00001130 5F                        <1>     pop   edi
3197 00001131 5E                        <1>     pop   esi
3198 00001132 5A                        <1>     pop   edx
3199 00001133 59                        <1>     pop   ecx
3200 00001134 5B                        <1>     pop   ebx
3201 00001135 58                        <1>     pop   eax
3202                                    <1>     ;pop  ebp
3203 00001136 CF                        <1>     iret                        ; RETURN
3204                                    <1>
3205                                    <1>     ;----- NOT IN HOLD STATE
3206                                    <1> K28:                            ; NO-HOLD-STATE
3207 00001137 3C58                      <1>     cmp   al, 88                ; TEST FOR OUT-OF-RANGE SCAN CODES
3208 00001139 77DF                      <1>     ja    short K26             ; IGNORE IF OUT-OF-RANGE
3209                                    <1>     ;
3210 0000113B F6C308                    <1>     test  bl, ALT_SHIFT              ; ARE WE IN ALTERNATE SHIFT
3211                                    <1>       ;jz  short K28A           ; IF NOT ALTERNATE
3212 0000113E 0F84F1000000              <1>       jz      K38
3213                                    <1>     ;
3214 00001144 F6C710                    <1>     test  bh, KBX                    ; IS THIS THE ENCHANCED KEYBOARD?
3215 00001147 740D                      <1>     jz    short K29             ; NO, ALT STATE IS REAL
3216                                    <1>      ;28/02/2015
3217 00001149 F605[546A0000]04          <1>     test  byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
3218                                    <1>     ;jz   short K29             ;  NO, ALT STATE IS REAL
3219 00001150 0F85DF000000              <1>     jnz   K38                   ; YES, THIS IS PHONY ALT STATE
3220                                    <1>       ;                         ; DUE TO PRESSING SYSREQ
3221                                    <1> ;K28A:      jmp   short K38
3222                                    <1>     ;
3223                                    <1>     ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
3224                                    <1> K29:                            ; TEST-RESET
3225 00001156 F6C304                    <1>     test  bl, CTL_SHIFT              ; ARE WE IN CONTROL SHIFT ALSO?
3226 00001159 740B                      <1>     jz    short K31             ; NO_RESET
3227 0000115B 3C53                      <1>     cmp   al, DEL_KEY           ; CTL-ALT STATE, TEST FOR DELETE KEY
3228 0000115D 7507                      <1>     jne   short K31             ; NO_RESET, IGNORE
3229                                    <1>     ;
3230                                    <1>     ;----- CTL-ALT-DEL HAS BEEN FOUND
3231                                    <1>     ; 26/08/2014
3232                                    <1> cpu_reset:
3233                                    <1>     ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
```

```
3234                                    <1>       ; Send FEh (system reset command) to the keyboard controller.
3235 0000115F B0FE                      <1>       mov    al, SHUT_CMD        ; SHUTDOWN COMMAND
3236 00001161 E664                      <1>       out    STATUS_PORT, al           ; SEND TO KEYBOARD CONTROL PORT
3237                                    <1> khere:
3238 00001163 F4                        <1>       hlt                        ; WAIT FOR 80286 RESET
3239 00001164 EBFD                      <1>       jmp    short khere         ; INSURE HALT
3240                                    <1>
3241                                    <1>       ;
3242                                    <1>       ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
3243                                    <1> K31:                             ; NO-RESET
3244 00001166 3C39                      <1>       cmp    al, 57              ; TEST FOR SPACE KEY
3245 00001168 7507                      <1>       jne    short K311          ; NOT THERE
3246 0000116A B020                      <1>       mov    al, ' '                   ; SET SPACE CHAR
3247 0000116C E948020000                <1>         jmp     K57                    ; BUFFER_FILL
3248                                    <1> K311:
3249 00001171 3C0F                      <1>       cmp    al, 15              ; TEST FOR TAB KEY
3250 00001173 7509                      <1>       jne    short K312          ; NOT THERE
3251 00001175 66B800A5                  <1>       mov    ax, 0A500h          ; SET SPECIAL CODE FOR ALT-TAB
3252 00001179 E93B020000                <1>         jmp     K57                    ; BUFFER_FILL
3253                                    <1> K312:
3254 0000117E 3C4A                      <1>       cmp    al, 74              ; TEST FOR KEY PAD -
3255 00001180 0F84A2000000              <1>         je      K37B                   ; GO PROCESS
3256 00001186 3C4E                      <1>       cmp    al, 78              ; TEST FOR KEY PAD +
3257 00001188 0F849A000000              <1>         je      K37B                   ; GO PROCESS
3258                                    <1>       ;
3259                                    <1>       ;----- LOOK FOR KEY PAD ENTRY
3260                                    <1> K32:                             ; ALT-KEY-PAD
3261 0000118E BF[14690000]              <1>       mov    edi, K30            ; ALT-INPUT-TABLE offset
3262 00001193 B90A000000                <1>       mov    ecx, 10                   ; LOOK FOR ENTRY USING KEYPAD
3263 00001198 F2AE                      <1>       repne  scasb               ; LOOK FOR MATCH
3264 0000119A 7525                      <1>       jne    short K33           ; NO_ALT_KEYPAD
3265 0000119C F6C702                    <1>       test   bh, LC_E0           ; IS THIS ONE OF THE NEW KEYS?
3266 0000119F 0F858A000000              <1>         jnz     K37C                   ; YES, JUMP, NOT NUMPAD KEY
3267 000011A5 81EF[15690000]            <1>       sub    edi, K30+1          ; DI NOW HAS ENTRY VALUE
3268 000011AB A0[576A0000]              <1>       mov    al, [ALT_INPUT]     ; GET THE CURRENT BYTE
3269 000011B0 B40A                      <1>       mov    ah, 10              ; MULTIPLY BY 10
3270 000011B2 F6E4                      <1>       mul    ah
3271 000011B4 6601F8                    <1>       add    ax, di              ; ADD IN THE LATEST ENTRY
3272 000011B7 A2[576A0000]              <1>       mov    [ALT_INPUT], al     ; STORE IT AWAY
3273                                    <1> ;K32A:
3274 000011BC E959FFFFFF                <1>         jmp     K26                    ; THROW AWAY THAT KEYSTROKE
3275                                    <1>       ;
3276                                    <1>       ;----- LOOK FOR SUPERSHIFT ENTRY
3277                                    <1> K33:                             ; NO-ALT-KEYPAD
3278 000011C1 C605[576A0000]00          <1>         mov     byte [ALT_INPUT], 0     ; ZERO ANY PREVIOUS ENTRY INTO INPUT
3279 000011C8 B91A000000                <1>       mov    ecx, 26                  ; (DI),(ES) ALREADY POINTING
3280 000011CD F2AE                      <1>       repne  scasb               ; LOOK FOR MATCH IN ALPHABET
3281 000011CF 7450                      <1>       je     short K37A          ; MATCH FOUND, GO FILLL THE BUFFER
3282                                    <1>       ;
3283                                    <1>       ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
3284                                    <1> K34:                             ; ALT-TOP-ROW
3285 000011D1 3C02                      <1>       cmp    al, 2               ; KEY WITH '1' ON IT
3286 000011D3 7253                      <1>       jb     short K37B          ; MUST BE ESCAPE
3287 000011D5 3C0D                      <1>       cmp    al, 13              ; IS IT IN THE REGION
3288 000011D7 7705                      <1>       ja     short K35           ; NO, ALT SOMETHING ELSE
3289 000011D9 80C476                    <1>       add    ah, 118                  ; CONVERT PSEUDO SCAN CODE TO RANGE
3290 000011DC EB43                      <1>       jmp    short K37A          ; GO FILL THE BUFFER
3291                                    <1>       ;
3292                                    <1>       ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
3293                                    <1> K35:                             ; ALT-FUNCTION
3294 000011DE 3C57                      <1>       cmp    al, F11_M           ; IS IT F11?
3295 000011E0 7209                      <1>       jb     short K35A ; 20/02/2015   ; NO, BRANCH
3296 000011E2 3C58                      <1>       cmp    al, F12_M           ; IS IT F12?
3297 000011E4 7705                      <1>       ja     short K35A ; 20/02/2015   ; NO, BRANCH
3298 000011E6 80C434                    <1>       add    ah, 52              ; CONVERT TO PSEUDO SCAN CODE
3299 000011E9 EB36                      <1>       jmp    short K37A          ; GO FILL THE BUFFER
3300                                    <1> K35A:
3301 000011EB F6C702                    <1>       test   bh, LC_E0           ; DO WE HAVE ONE OF THE NEW KEYS?
3302 000011EE 7422                      <1>       jz     short K37           ; NO, JUMP
3303 000011F0 3C1C                      <1>       cmp    al, 28              ; TEST FOR KEYPAD ENTER
3304 000011F2 7509                      <1>         jne     short K35B             ; NOT THERE
3305 000011F4 66B800A6                  <1>       mov    ax, 0A600h          ; SPECIAL CODE
3306 000011F8 E9BC010000                <1>       jmp    K57                 ; BUFFER FILL
3307                                    <1> K35B:
3308 000011FD 3C53                      <1>       cmp    al, 83              ; TEST FOR DELETE KEY
3309 000011FF 742E                      <1>       je     short K37C          ; HANDLE WITH OTHER EDIT KEYS
3310 00001201 3C35                      <1>       cmp    al, 53              ; TEST FOR KEYPAD /
3311                                    <1>       ;jne   short K32A          ; NOT THERE, NO OTHER E0 SPECIALS
3312 00001203 0F8511FFFFFF              <1>         jne     K26
3313 00001209 66B800A4                  <1>       mov    ax, 0A400h          ; SPECIAL CODE
3314 0000120D E9A7010000                <1>       jmp    K57                 ; BUFFER FILL
3315                                    <1> K37:
3316 00001212 3C3B                      <1>       cmp    al, 59              ; TEST FOR FUNCTION KEYS (F1)
3317 00001214 7212                      <1>         jb      short K37B             ; NO FN, HANDLE W/OTHER EXTENDED
3318 00001216 3C44                      <1>       cmp    al, 68              ; IN KEYPAD REGION?
3319                                    <1>         ;ja   short K32A          ; IF SO, IGNORE
3320 00001218 0F87FCFEFFFF              <1>         ja      K26
3321 0000121E 80C42D                    <1>       add    ah, 45              ; CONVERT TO PSEUDO SCAN CODE
3322                                    <1> K37A:
3323 00001221 B000                      <1>       mov    al, 0               ; ASCII CODE OF ZERO
3324 00001223 E991010000                <1>         jmp     K57                    ; PUT IT IN THE BUFFER
3325                                    <1> K37B:
3326 00001228 B0F0                      <1>       mov    al, 0F0h            ; USE SPECIAL ASCII CODE
3327 0000122A E98A010000                <1>       jmp     K57                   ; PUT IT IN THE BUFFER
3328                                    <1> K37C:
3329 0000122F 0450                      <1>       add    al, 80              ; CONVERT SCAN CODE (EDIT KEYS)
3330 00001231 88C4                      <1>       mov    ah, al              ; (SCAN CODE NOT IN AH FOR INSERT)
3331 00001233 EBEC                      <1>       jmp     short K37A             ; PUT IT IN THE BUFFER
3332                                    <1>       ;
3333                                    <1>       ;----- NOT IN ALTERNATE SHIFT
3334                                    <1> K38:                             ; NOT-ALT-SHIFT
3335                                    <1>                                  ; BL STILL HAS SHIFT FLAGS
3336 00001235 F6C304                    <1>       test   bl, CTL_SHIFT             ; ARE WE IN CONTROL SHIFT?
3337                                    <1>       ;jnz   short K38A          ; YES, START PROCESSING
3338 00001238 0F84B0000000              <1>         jz      K44                    ; NOT-CTL-SHIFT
```

```
3339                                    <1>       ;
3340                                    <1>       ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
3341                                    <1>       ;----- TEST FOR BREAK
3342                                    <1> K38A:
3343 0000123E 3C46                      <1>       cmp   al, SCROLL_KEY            ; TEST FOR BREAK
3344 00001240 7531                      <1>       jne   short K39           ; JUMP, NO-BREAK
3345 00001242 F6C710                    <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
3346 00001245 7405                      <1>       jz    short K38B          ; NO, BREAK IS VALID
3347 00001247 F6C702                    <1>       test  bh, LC_E0           ; YES, WAS LAST CODE AN E0?
3348 0000124A 7427                      <1>       jz    short K39           ; NO-BREAK, TEST FOR PAUSE
3349                                    <1> K38B:
3350 0000124C 8B1D[606A0000]            <1>       mov   ebx, [BUFFER_HEAD]  ; RESET BUFFER TO EMPTY
3351 00001252 891D[646A0000]            <1>       mov   [BUFFER_TAIL], ebx
3352 00001258 C605[526A0000]80          <1>       mov   byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
3353                                    <1>       ;
3354                                    <1>       ;----- ENABLE KEYBOARD
3355 0000125F B0AE                      <1>       mov   al, ENA_KBD         ; ENABLE KEYBOARD
3356 00001261 E89B010000                <1>       call  SHIP_IT                   ; EXECUTE ENABLE
3357                                    <1>       ;
3358                                    <1>       ; CTRL+BREAK code here !!!
3359                                    <1>       ;INT   1BH                ; BREAK INTERRUPT VECTOR
3360                                    <1>       ; 17/10/2015
3361 00001266 E89F260000                <1>       call  ctrlbrk ; control+break subroutine
3362                                    <1>       ;
3363 0000126B 6629C0                    <1>       sub   ax, ax              ; PUT OUT DUMMY CHARACTER
3364 0000126E E946010000                <1>         jmp     K57                   ; BUFFER_FILL
3365                                    <1>       ;
3366                                    <1>       ;----- TEST FOR PAUSE
3367                                    <1> K39:                            ; NO_BREAK
3368 00001273 F6C710                    <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
3369 00001276 7537                      <1>       jnz   short K41           ; YES, THEN THIS CAN'T BE PAUSE
3370 00001278 3C45                      <1>       cmp   al, NUM_KEY         ; LOOK FOR PAUSE KEY
3371 0000127A 7533                      <1>       jne   short K41           ; NO-PAUSE
3372                                    <1> K39P:
3373 0000127C 800D[546A0000]08          <1>       or    byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
3374                                    <1>       ;
3375                                    <1>       ;----- ENABLE KEYBOARD
3376 00001283 B0AE                      <1>       mov   al, ENA_KBD         ; ENABLE KEYBOARD
3377 00001285 E877010000                <1>       call  SHIP_IT                   ; EXECUTE ENABLE
3378                                    <1> K39A:
3379 0000128A B020                      <1>       mov   al, EOI                   ; END OF INTERRUPT TO CONTROL PORT
3380 0000128C E620                      <1>       out   20h, al ;out INTA00, al   ; ALLOW FURTHER KEYSTROKE INTERRUPTS
3381                                    <1>       ;
3382                                    <1>       ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
3383 0000128E 803D[506A0000]07          <1>         cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
3384 00001295 740A                      <1>         je      short K40                     ; YES, NOTHING TO DO
3385 00001297 66BAD803                  <1>       mov   dx, 03D8h           ; PORT FOR COLOR CARD
3386 0000129B A0[516A0000]              <1>         mov     al, [CRT_MODE_SET]     ; GET THE VALUE OF THE CURRENT MODE
3387 000012A0 EE                        <1>       out   dx, al              ; SET THE CRT MODE, SO THAT CRT IS ON
3388                                    <1>       ;
3389                                    <1> K40:                            ; PAUSE-LOOP
3390 000012A1 F605[546A0000]08          <1>         test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
3391 000012A8 75F7                      <1>       jnz   short K40           ; LOOP UNTIL FLAG TURNED OFF
3392                                    <1>       ;
3393 000012AA E977FEFFFF                <1>         jmp     K27                     ; INTERRUPT_RETURN_NO_EOI
3394                                    <1>         ;
3395                                    <1>       ;----- TEST SPECIAL CASE KEY 55
3396                                    <1> K41:                            ; NO-PAUSE
3397 000012AF 3C37                      <1>       cmp   al, 55              ; TEST FOR */PRTSC KEY
3398 000012B1 7513                      <1>       jne   short K42           ; NOT-KEY-55
3399 000012B3 F6C710                    <1>       test  bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
3400 000012B6 7405                      <1>       jz    short K41A          ; NO, CTL-PRTSC IS VALID
3401 000012B8 F6C702                    <1>       test  bh, LC_E0           ; YES, WAS LAST CODE AN E0?
3402 000012BB 7421                      <1>       jz    short K42B          ; NO, TRANSLATE TO A FUNCTION
3403                                    <1> K41A:
3404 000012BD 66B80072                  <1>       mov   ax, 114*256         ; START/STOP PRINTING SWITCH
3405 000012C1 E9F3000000                <1>         jmp     K57                   ; BUFFER_FILL
3406                                    <1>       ;
3407                                    <1>       ;----- SET UP TO TRANSLATE CONTROL SHIFT
3408                                    <1> K42:                            ; NOT-KEY-55
3409 000012C6 3C0F                      <1>       cmp   al, 15              ; IS IT THE TAB KEY?
3410 000012C8 7414                      <1>       je    short K42B          ; YES, XLATE TO FUNCTION CODE
3411 000012CA 3C35                      <1>       cmp   al, 53              ; IS IT THE / KEY?
3412 000012CC 750E                      <1>       jne   short K42A          ; NO, NO MORE SPECIAL CASES
3413 000012CE F6C702                    <1>       test  bh, LC_E0           ; YES, IS IT FROM THE KEY PAD?
3414 000012D1 7409                      <1>       jz    short K42A          ; NO, JUST TRANSLATE
3415 000012D3 66B80095                  <1>       mov   ax, 9500h           ; YES, SPECIAL CODE FOR THIS ONE
3416 000012D7 E9DD000000                <1>       jmp   K57                 ; BUFFER FILL
3417                                    <1> K42A:
3418                                    <1>       ;;mov  ebx, _K8           ; SET UP TO TRANSLATE CTL
3419 000012DC 3C3B                      <1>       cmp   al, 59              ; IS IT IN CHARACTER TABLE?
3420                                    <1>         ;jb  short K45F             ; YES, GO TRANSLATE CHAR
3421                                    <1>       ;;jb  K56 ; 20/02/2015
3422                                    <1>       ;;jmp  K64 ; 20/02/2015
3423                                    <1> K42B:
3424 000012DE BB[48690000]              <1>       mov   ebx, _K8            ; SET UP TO TRANSLATE CTL
3425 000012E3 0F82AE000000              <1>       jb    K56 ;; 20/02/2015
3426 000012E9 E9B9000000                <1>       jmp   K64
3427                                    <1>         ;
3428                                    <1>       ;----- NOT IN CONTROL SHIFT
3429                                    <1> K44:                            ; NOT-CTL-SHIFT
3430 000012EE 3C37                      <1>       cmp   al, 55              ; PRINT SCREEN KEY?
3431 000012F0 7528                      <1>       jne   short K45           ; NOT PRINT SCREEN
3432 000012F2 F6C710                    <1>       test  bh, KBX                   ; IS THIS ENHANCED KEYBOARD?
3433 000012F5 7407                      <1>       jz    short K44A          ; NO, TEST FOR SHIFT STATE
3434 000012F7 F6C702                    <1>       test  bh, LC_E0           ; YES, LAST CODE A MARKER?
3435 000012FA 7507                      <1>       jnz   short K44B          ; YES, IS PRINT SCREEN
3436 000012FC EB41                      <1>       jmp   short K45C          ; NO, TRANSLATE TO '*' CHARACTER
3437                                    <1> K44A:
3438 000012FE F6C303                    <1>       test  bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
3439 00001301 743C                      <1>       jz    short K45C          ; NO, TRANSLATE TO '*' CHARACTER
3440                                    <1>       ;
3441                                    <1>       ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
3442                                    <1> K44B:
3443 00001303 B0AE                      <1>       mov   al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
```

```
3444 00001305 E8F7000000          <1>       call   SHIP_IT                   ; EXECUTE ENABLE
3445 0000130A B020                <1>       mov    al, EOI                    ; END OF CURRENT INTERRUPT
3446 0000130C E620                <1>       out    20h, al ;out INTA00, al    ; SO FURTHER THINGS CAN HAPPEN
3447                              <1>       ; Print Screen !!!          ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
3448                              <1>       ;PUSH  BP                  ; SAVE POINTER
3449                              <1>       ;INT   5H                  ; ISSUE PRINT SCREEN INTERRUPT
3450                              <1>       ;POP   BP                  ; RESTORE POINTER
3451 0000130E 8025[566A0000]FC    <1>         and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
3452 00001315 E90CFEFFFF          <1>         jmp     K27                      ; GO BACK WITHOUT EOI OCCURRING
3453                              <1>       ;
3454                              <1>       ;----- HANDLE IN-CORE KEYS
3455                              <1> K45:                              ; NOT-PRINT-SCREEN
3456 0000131A 3C3A                <1>       cmp    al, 58               ; TEST FOR IN-CORE AREA
3457 0000131C 7734                <1>       ja     short K46            ; JUMP IF NOT
3458 0000131E 3C35                <1>       cmp    al, 53               ; IS THIS THE '/' KEY?
3459 00001320 7505                <1>       jne    short K45A           ; NO, JUMP
3460 00001322 F6C702              <1>       test   bh, LC_E0            ; WAS THE LAST CODE THE MARKER?
3461 00001325 7518                <1>       jnz    short K45C           ; YES, TRANSLATE TO CHARACTER
3462                              <1> K45A:
3463 00001327 B91A000000          <1>       mov    ecx, 26                    ; LENGHT OF SEARCH
3464 0000132C BF[1E690000]        <1>       mov    edi, K30+10          ; POINT TO TABLE OF A-Z CHARS
3465 00001331 F2AE                <1>       repne  scasb                ; IS THIS A LETTER KEY?
3466                              <1>              ; 20/02/2015
3467 00001333 7505                <1>       jne    short K45B              ; NO, SYMBOL KEY
3468                              <1>       ;
3469 00001335 F6C340              <1>       test   bl, CAPS_STATE              ; ARE WE IN CAPS_LOCK?
3470 00001338 750C                <1>       jnz    short K45D           ; TEST FOR SURE
3471                              <1> K45B:
3472 0000133A F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
3473 0000133D 750C                <1>       jnz    short K45E           ; YES, UPPERCASE
3474                              <1>                                   ; NO, LOWERCASE
3475                              <1> K45C:
3476 0000133F BB[A0690000]        <1>       mov    ebx, K10             ; TRANSLATE TO LOWERCASE LETTERS
3477 00001344 EB51                <1>       jmp    short K56
3478                              <1> K45D:                             ; ALMOST-CAPS-STATE
3479 00001346 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
3480 00001349 75F4                <1>       jnz    short K45C           ; SHIFTED TEMP OUT OF CAPS STATE
3481                              <1> K45E:
3482 0000134B BB[F8690000]        <1>       mov    ebx, K11             ; TRANSLATE TO UPPER CASE LETTERS
3483 00001350 EB45                <1> K45F: jmp    short K56
3484                              <1>       ;
3485                              <1>       ;----- TEST FOR KEYS F1 - F10
3486                              <1> K46:                              ; NOT IN-CORE AREA
3487 00001352 3C44                <1>       cmp    al, 68               ; TEST FOR F1 - F10
3488                              <1>       ;ja    short K47            ; JUMP IF NOT
3489                              <1>       ;jmp   short K53            ; YES, GO DO FN KEY PROCESS
3490 00001354 7635                <1>       jna    short K53
3491                              <1>       ;
3492                              <1>       ;----- HANDLE THE NUMERIC PAD KEYS
3493                              <1> K47:                              ; NOT F1 - F10
3494 00001356 3C53                <1>       cmp    al, 83               ; TEST NUMPAD KEYS
3495 00001358 772D                <1>       ja     short K52            ; JUMP IF NOT
3496                              <1>       ;
3497                              <1>       ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
3498                              <1> K48:
3499 0000135A 3C4A                <1>       cmp    al , 74                    ; SPECIAL CASE FOR MINUS
3500 0000135C 74ED                <1>       je     short K45E           ; GO TRANSLATE
3501 0000135E 3C4E                <1>       cmp    al , 78                    ; SPECIAL CASE FOR PLUS
3502 00001360 74E9                <1>       je     short K45E           ; GO TRANSLATE
3503 00001362 F6C702              <1>       test   bh, LC_E0            ; IS THIS ONE OFTHE NEW KEYS?
3504 00001365 750A                <1>       jnz    short K49            ; YES, TRANSLATE TO BASE STATE
3505                              <1>       ;
3506 00001367 F6C320              <1>       test   bl, NUM_STATE        ; ARE WE IN NUM LOCK
3507 0000136A 7514                <1>       jnz    short K50            ; TEST FOR SURE
3508 0000136C F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
3509                              <1>       ;jnz   short K51            ; IF SHIFTED, REALLY NUM STATE
3510 0000136F 75DA                <1>       jnz    short K45E
3511                              <1>       ;
3512                              <1>       ;----- BASE CASE FOR KEYPAD
3513                              <1> K49:
3514 00001371 3C4C                <1>       cmp    al, 76               ; SPECIAL CASE FOR BASE STATE 5
3515 00001373 7504                <1>       jne    short K49A           ; CONTINUE IF NOT KEYPAD 5
3516 00001375 B0F0                <1>       mov    al, 0F0h             ; SPECIAL ASCII CODE
3517 00001377 EB40                <1>       jmp    short K57            ; BUFFER FILL
3518                              <1> K49A:
3519 00001379 BB[A0690000]        <1>       mov    ebx, K10             ; BASE CASE TABLE
3520 0000137E EB27                <1>       jmp    short K64            ; CONVERT TO PSEUDO SCAN
3521                              <1>       ;
3522                              <1>       ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
3523                              <1> K50:                              ; ALMOST-NUM-STATE
3524 00001380 F6C303              <1>         test    bl, LEFT_SHIFT+RIGHT_SHIFT
3525 00001383 75EC                <1>       jnz    short K49            ; SHIFTED TEMP OUT OF NUM STATE
3526 00001385 EBC4                <1> K51:  jmp    short K45E           ; REALLY NUM STATE
3527                              <1>       ;
3528                              <1>       ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS
3529                              <1> K52:                              ; NOT A NUMPAD KEY
3530 00001387 3C56                <1>       cmp    al, 86               ; IS IT THE NEW WT KEY?
3531                              <1>       ;jne   short K53            ; JUMP IF NOT
3532                              <1>       ;jmp   short K45B           ; HANDLE WITH REST OF LETTER KEYS
3533 00001389 74AF                <1>       je     short K45B
3534                              <1>       ;
3535                              <1>       ;----- MUST BE F11 OR F12
3536                              <1> K53:                              ; F1 - F10 COME HERE, TOO
3537 0000138B F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
3538 0000138E 74E1                <1>       jz     short K49            ; JUMP, LOWER CASE PSEUDO SC'S
3539                              <1>              ; 20/02/2015
3540 00001390 BB[F8690000]        <1>       mov    ebx, K11             ; UPPER CASE PSEUDO SCAN CODES
3541 00001395 EB10                <1>       jmp    short K64            ; TRANSLATE SCAN
3542                              <1>       ;
3543                              <1>       ;----- TRANSLATE THE CHARACTER
3544                              <1> K56:                              ; TRANSLATE-CHAR
3545 00001397 FEC8                <1>       dec    al                   ; CONVERT ORIGIN
3546 00001399 D7                  <1>       xlat                        ; CONVERT THE SCAN CODE TO ASCII
3547 0000139A F605[566A0000]02    <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
3548 000013A1 7416                <1>       jz     short K57            ; NO, GO FILL BUFFER
```

```
3549 000013A3 B4E0                <1>       mov   ah, MC_E0           ; YES, PUT SPECIAL MARKER IN AH
3550 000013A5 EB12                <1>       jmp   short K57           ; PUT IT INTO THE BUFFER
3551                              <1>       ;
3552                              <1>       ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
3553                              <1> K64:                            ; TRANSLATE-SCAN-ORGD
3554 000013A7 FEC8                <1>       dec   al                  ; CONVERT ORIGIN
3555 000013A9 D7                  <1>             xlat                        ; CTL TABLE SCAN
3556 000013AA 88C4                <1>       mov   ah, al              ; PUT VALUE INTO AH
3557 000013AC B000                <1>       mov   al, 0               ; ZERO ASCII CODE
3558 000013AE F605[566A0000]02    <1>       test  byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
3559 000013B5 7402                <1>       jz    short K57           ; NO, GO FILL BUFFER
3560 000013B7 B0E0                <1>       mov   al, MC_E0           ; YES, PUT SPECIAL MARKER IN AL
3561                              <1>       ;
3562                              <1>       ;----- PUT CHARACTER INTO BUFFER
3563                              <1> K57:                            ; BUFFER_FILL
3564 000013B9 3CFF                <1>       cmp   al, -1              ; IS THIS AN IGNORE CHAR
3565                              <1>         ;je  short K59          ; YES, DO NOTHING WITH IT
3566 000013BB 0F8459FDFFFF        <1>       je      K26               ; YES, DO NOTHING WITH IT
3567 000013C1 80FCFF              <1>       cmp   ah, -1              ; LOOK FOR -1 PSEUDO SCAN
3568                              <1>         ;jne short K61          ; NEAR_INTERRUPT_RETURN
3569 000013C4 0F8450FDFFFF        <1>       je      K26               ; INTERRUPT_RETURN
3570                              <1> ;K59:                           ; NEAR_INTERRUPT_RETURN
3571                              <1> ;     jmp   K26                 ; INTERRUPT_RETURN
3572                              <1> K61:                            ; NOT-CAPS-STATE
3573 000013CA 8B1D[646A0000]      <1>       mov   ebx, [BUFFER_TAIL]  ; GET THE END POINTER TO THE BUFFER
3574 000013D0 89DE                <1>       mov   esi, ebx            ; SAVE THE VALUE
3575 000013D2 E87BFAFFFF          <1>       call  _K4                 ; ADVANCE THE TAIL
3576 000013D7 3B1D[606A0000]      <1>       cmp   ebx, [BUFFER_HEAD]  ; HAS THE BUFFER WRAPPED AROUND
3577 000013DD 740E                <1>       je    short K62           ; BUFFER_FULL_BEEP
3578 000013DF 668906              <1>       mov   [esi], ax           ; STORE THE VALUE
3579 000013E2 891D[646A0000]      <1>       mov   [BUFFER_TAIL], ebx  ; MOVE THE POINTER UP
3580 000013E8 E92DFDFFFF          <1>       jmp   K26
3581                              <1>       ;;cli                     ; TURN OFF INTERRUPTS
3582                              <1>       ;;mov  al, EOI                  ; END OF INTERRUPT COMMAND
3583                              <1>       ;;out  INTA00, al         ; SEND COMMAND TO INTERRUPT CONTROL PORT
3584                              <1>       ;MOV   AL, ENA_KBD        ; INSURE KEYBOARD IS ENABLED
3585                              <1>       ;CALL  SHIP_IT                  ; EXECUTE ENABLE
3586                              <1>       ;MOV   AX, 9102H          ; MOVE IN POST CODE & TYPE
3587                              <1>       ;INT   15H                ; PERFORM OTHER FUNCTION
3588                              <1>       ;;and  byte [KB_FLAG_3],~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
3589                              <1>       ;JMP   K27A               ; INTERRUPT_RETURN
3590                              <1>       ;;jmp   K27
3591                              <1>       ;
3592                              <1>       ;----- BUFFER IS FULL SOUND THE BEEPER
3593                              <1> K62:
3594 000013ED B020                <1>       mov   al, EOI                   ; ENABLE INTERRUPT CONTROLLER CHIP
3595 000013EF E620                <1>       out   INTA00, al
3596 000013F1 66B9A602            <1>       mov   cx, 678                   ; DIVISOR FOR 1760 HZ
3597 000013F5 B304                <1>       mov   bl, 4               ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
3598 000013F7 E8A1010000          <1>       call  beep                ; GO TO COMMON BEEP HANDLER
3599 000013FC E925FDFFFF          <1>       jmp    K27                ; EXIT
3600                              <1>
3601                              <1> SHIP_IT:
3602                              <1>       ;--------------------------------------------------------------------------------
----
3603                              <1>       ; SHIP_IT
3604                              <1>       ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
3605                              <1>       ;     TO THE KEYBOARD CONTROLLER.
3606                              <1>       ;--------------------------------------------------------------------------------
----
3607                              <1>       ;
3608 00001401 6650                <1>       push  ax                  ; SAVE DATA TO SEND
3609                              <1>
3610                              <1>       ;----- WAIT FOR COMMAND TO ACCEPTED
3611 00001403 FA                  <1>       cli                       ; DISABLE INTERRUPTS TILL DATA SENT
3612                              <1>       ; xor  ecx, ecx           ; CLEAR TIMEOUT COUNTER
3613 00001404 B900000100          <1>       mov   ecx, 10000h
3614                              <1> S10:
3615 00001409 E464                <1>       in    al, STATUS_PORT           ; READ KEYBOARD CONTROLLER STATUS
3616 0000140B A802                <1>       test  al, INPT_BUF_FULL   ; CHECK FOR ITS INPUT BUFFER BUSY
3617 0000140D E0FA                <1>       loopnz S10                ; WAIT FOR COMMAND TO BE ACCEPTED
3618                              <1>
3619 0000140F 6658                <1>       pop   ax                  ; GET DATA TO SEND
3620 00001411 E664                <1>       out   STATUS_PORT, al           ; SEND TO KEYBOARD CONTROLLER
3621 00001413 FB                  <1>       sti                       ; ENABLE INTERRUPTS AGAIN
3622 00001414 C3                  <1>       retn                      ; RETURN TO CALLER
3623                              <1>
3624                              <1> SND_DATA:
3625                              <1>       ; ------------------------------------------------------------------------------
-----
3626                              <1>       ; SND_DATA
3627                              <1>       ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
3628                              <1>       ;     TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
3629                              <1>       ;     HANDLES ANY RETRIES IF REQUIRED
3630                              <1>       ; ------------------------------------------------------------------------------
-----
3631                              <1>       ;
3632 00001415 6650                <1>       push  ax                  ; SAVE REGISTERS
3633 00001417 6653                <1>       push  bx
3634 00001419 51                  <1>       push  ecx
3635 0000141A 88C7                <1>       mov   bh, al              ; SAVE TRANSMITTED BYTE FOR RETRIES
3636 0000141C B303                <1>       mov   bl, 3               ; LOAD RETRY COUNT
3637                              <1> SD0:
3638 0000141E FA                  <1>       cli                       ; DISABLE INTERRUPTS
3639 0000141F 8025[556A0000]CF    <1>       and   byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
3640                              <1>       ;
3641                              <1>       ;----- WAIT FOR COMMAND TO BE ACCEPTED
3642 00001426 B900000100          <1>       mov   ecx, 10000h         ; MAXIMUM WAIT COUNT
3643                              <1> SD5:
3644 0000142B E464                <1>       in    al, STATUS_PORT           ; READ KEYBOARD PROCESSOR STATUS PORT
3645 0000142D A802                <1>       test  al, INPT_BUF_FULL   ; CHECK FOR ANY PENDING COMMAND
3646 0000142F E0FA                <1>       loopnz SD5                ; WAIT FOR COMMAND TO BE ACCEPTED
3647                              <1>       ;
3648 00001431 88F8                <1>       mov   al, bh              ; REESTABLISH BYTE TO TRANSMIT
3649 00001433 E660                <1>       out   PORT_A, al          ; SEND BYTE
```

```
  3650 00001435 FB                <1>       sti                     ; ENABLE INTERRUPTS
  3651                            <1>       ;mov   cx, 01A00h       ; LOAD COUNT FOR 10 ms+
  3652 00001436 B9FFFF0000        <1>       mov    ecx, 0FFFFh
  3653                            <1> SD1:
  3654 0000143B F605[556A0000]30  <1>       test   byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
  3655 00001442 750F              <1>       jnz    short SD3        ; IF SET, SOMETHING RECEIVED GO PROCESS
  3656 00001444 E2F5              <1>       loop   SD1              ; OTHERWISE WAIT
  3657                            <1> SD2:
  3658 00001446 FECB              <1>       dec    bl               ; DECREMENT RETRY COUNT
  3659 00001448 75D4              <1>       jnz    short SD0        ; RETRY TRANSMISSION
  3660 0000144A 800D[556A0000]80  <1>       or     byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
  3661 00001451 EB09              <1>       jmp    short SD4        ; RETRIES EXHAUSTED FORGET TRANSMISSION
  3662                            <1> SD3:
  3663 00001453 F605[556A0000]10  <1>       test   byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
  3664 0000145A 74EA              <1>       jz     short SD2        ; IF NOT, GO RESEND
  3665                            <1> SD4:
  3666 0000145C 59                <1>       pop    ecx              ; RESTORE REGISTERS
  3667 0000145D 665B              <1>       pop    bx
  3668 0000145F 6658              <1>       pop    ax
  3669 00001461 C3                <1>       retn                    ; RETURN, GOOD TRANSMISSION
  3670                            <1>
  3671                            <1> SND_LED:
  3672                            <1>       ; ------------------------------------------------------------------------
-----
  3673                            <1>       ; SND_LED
  3674                            <1>       ;     THIS ROUTINES TURNS ON THE MODE INDICATORS.
  3675                            <1>       ;
  3676                            <1>       ;-------------------------------------------------------------------------
-----
  3677                            <1>       ;
  3678 00001462 FA                <1>       cli                     ; TURN OFF INTERRUPTS
  3679 00001463 F605[556A0000]40  <1>       test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
  3680 0000146A 755F              <1>       jnz    short SL1        ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
  3681                            <1>       ;
  3682 0000146C 800D[556A0000]40  <1>       or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
  3683 00001473 B020              <1>       mov    al, EOI                 ; END OF INTERRUPT COMMAND
  3684 00001475 E620              <1>       out    20h, al ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
  3685 00001477 EB11              <1>       jmp    short SL0        ; GO SEND MODE INDICATOR COMMAND
  3686                            <1> SND_LED1:
  3687 00001479 FA                <1>       cli                     ; TURN OFF INTERRUPTS
  3688 0000147A F605[556A0000]40  <1>       test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
  3689 00001481 7548              <1>       jnz    short SL1        ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
  3690                            <1>       ;
  3691 00001483 800D[556A0000]40  <1>       or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
  3692                            <1> SL0:
  3693 0000148A B0ED              <1>       mov    al, LED_CMD      ; LED CMD BYTE
  3694 0000148C E884FFFFFF        <1>       call   SND_DATA         ; SEND DATA TO KEYBOARD
  3695 00001491 FA                <1>       cli
  3696 00001492 E836000000        <1>       call   MAKE_LED         ; GO FORM INDICATOR DATA BYTE
  3697 00001497 8025[556A0000]F8  <1>       and    byte [KB_FLAG_2], 0F8h    ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
  3698 0000149E 0805[556A0000]    <1>       or     [KB_FLAG_2], al    ; SAVE PRESENT INDICATORS FOR NEXT TIME
  3699 000014A4 F605[556A0000]80  <1>       test   byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
  3700 000014AB 750F              <1>       jnz    short SL2        ; IF SO, BYPASS SECOND BYTE TRANSMISSION
  3701                            <1>       ;
  3702 000014AD E863FFFFFF        <1>       call   SND_DATA         ; SEND DATA TO KEYBOARD
  3703 000014B2 FA                <1>       cli                     ; TURN OFF INTERRUPTS
  3704 000014B3 F605[556A0000]80  <1>       test   byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
  3705 000014BA 7408              <1>       jz     short SL3        ; IF NOT, DON'T SEND AN ENABLE COMMAND
  3706                            <1> SL2:
  3707 000014BC B0F4              <1>       mov    al, KB_ENABLE    ; GET KEYBOARD CSA ENABLE COMMAND
  3708 000014BE E852FFFFFF        <1>       call   SND_DATA         ; SEND DATA TO KEYBOARD
  3709 000014C3 FA                <1>       cli                     ; TURN OFF INTERRUPTS
  3710                            <1> SL3:
  3711 000014C4 8025[556A0000]3F  <1>       and    byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
  3712                            <1> SL1:                        ; UPDATE AND TRANSMIT ERROR FLAG
  3713 000014CB FB                <1>       sti                     ; ENABLE INTERRUPTS
  3714 000014CC C3                <1>       retn                    ; RETURN TO CALLER
  3715                            <1>
  3716                            <1> MAKE_LED:
  3717                            <1>       ;-------------------------------------------------------------------------
----
  3718                            <1>       ; MAKE_LED
  3719                            <1>       ;     THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
  3720                            <1>       ;     THE MODE INDICATORS.
  3721                            <1>       ;-------------------------------------------------------------------------
----
  3722                            <1>       ;
  3723                            <1>       ;push  cx               ; SAVE CX
  3724 000014CD A0[536A0000]      <1>       mov    al, [KB_FLAG]    ; GET CAPS & NUM LOCK INDICATORS
  3725 000014D2 2470              <1>       and    al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
  3726                            <1>       ;mov   cl, 4            ; SHIFT COUNT
  3727                            <1>       ;rol   al, cl           ; SHIFT BITS OVER TO TURN ON INDICATORS
  3728 000014D4 C0C004            <1>       rol    al, 4 ; 20/02/2015
  3729 000014D7 2407              <1>       and    al, 07h                ; MAKE SURE ONLY MODE BITS ON
  3730                            <1>       ;pop   cx
  3731 000014D9 C3                <1>       retn                    ; RETURN TO CALLER
  3732                            <1>
  3733                            <1> ; % include 'kybdata.inc'   ; KEYBOARD DATA ; 11/03/2015
  3734                            <1>
  3735                            <1>
  3736                            <1> ; /// End Of KEYBOARD FUNCTIONS ///
  3737
  3738                                %include 'video.inc' ; 07/03/2015
  3739                            <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC
  3740                            <1> ; Last Modification: 16/01/2016
  3741                            <1> ;             (Video Data is in 'VIDATA.INC')
  3742                            <1> ;
  3743                            <1> ; ///////// VIDEO (CGA) FUNCTIONS ///////////////
  3744                            <1>
  3745                            <1> ; 30/06/2015
  3746                            <1> ; 27/06/2015
  3747                            <1> ; 11/03/2015
  3748                            <1> ; 02/09/2014
  3749                            <1> ; 30/08/2014
  3750                            <1> ; VIDEO FUNCTIONS
```

```
3751                                <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
3752                                <1>
3753                                <1> write_tty:
3754                                <1>       ; 13/08/2015
3755                                <1>       ; 02/09/2014
3756                                <1>       ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
3757                                <1>       ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
3758                                <1>       ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
3759                                <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
3760                                <1>       ;
3761                                <1>       ; INPUT -> AH = Color (Forecolor, Backcolor)
3762                                <1>       ;        AL = Character to be written
3763                                <1>       ;        EBX = Video Page (0 to 7)
3764                                <1>       ;        (BH = 0 --> Video Mode 3)
3765                                <1>
3766                                <1> RVRT  equ    00001000b    ; VIDEO VERTICAL RETRACE BIT
3767                                <1> RHRZ  equ    00000001b    ; VIDEO HORIZONTAL RETRACE BIT
3768                                <1>
3769                                <1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
3770                                <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
3771                                <1> ;
3772                                <1> ; 06/10/85  VIDEO DISPLAY BIOS
3773                                <1> ;
3774                                <1> ;--- WRITE_TTY ----------------------------------------------------------------
3775                                <1> ;                                                       :
3776                                <1> ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE                 :
3777                                <1> ;   VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT                :
3778                                <1> ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.           :
3779                                <1> ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN            :
3780                                <1> ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW       :
3781                                <1> ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,        :
3782                                <1> ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.       :
3783                                <1> ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE            :
3784                                <1> ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS      :
3785                                <1> ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,       :
3786                                <1> ;   THE 0 COLOR IS USED.                                          :
3787                                <1> ;   ENTRY --                                                      :
3788                                <1> ;     (AH) = CURRENT CRT MODE                                          :
3789                                <1> ;     (AL) = CHARACTER TO BE WRITTEN                                   :
3790                                <1> ;         NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :
3791                                <1> ;         HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS      :
3792                                <1> ;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE  :
3793                                <1> ;   EXIT --                                                       :
3794                                <1> ;     ALL REGISTERS SAVED                                         :
3795                                <1> ;-------------------------------------------------------------------------------
3796                                <1>
3797 000014DA FA                   <1>       cli
3798                                <1>       ;
3799                                <1>       ; READ CURSOR (04/12/2013)
3800                                <1>       ; Retro UNIX 386 v1 Modifications: 30/08/2014
3801 000014DB 08FF                 <1>       or    bh, bh
3802 000014DD 0F85AB000000         <1>       jnz   beeper
3803                                <1>       ; 01/09/2014
3804 000014E3 803D[506A0000]03     <1>       cmp   byte [CRT_MODE], 3
3805 000014EA 7405                 <1>       je    short m3
3806                                <1>       ;
3807 000014EC E887020000           <1>       call  set_mode
3808                                <1> m3:
3809 000014F1 89DE                 <1>       mov   esi, ebx ; 13/08/2015 (0 to 7)
3810 000014F3 66D1E6               <1>       shl   si, 1
3811 000014F6 81C6[86700000]       <1>       add   esi, cursor_posn
3812 000014FC 668B16               <1>       mov   dx, [esi]
3813                                <1>       ;
3814                                <1>       ; dx now has the current cursor position
3815                                <1>       ;
3816 000014FF 3C0D                 <1>       cmp   al, 0Dh            ; is it carriage return or control character
3817 00001501 764D                 <1>       jbe   short u8
3818                                <1>       ;
3819                                <1>       ; write the char to the screen
3820                                <1> u0:
3821                                <1>       ; ah = attribute/color
3822                                <1>       ; al = character
3823                                <1>       ; bl = video page number (0 to 7)
3824                                <1>       ; bh = 0
3825                                <1>       ;
3826 00001503 E83F020000           <1>       call  write_c_current
3827                                <1>       ;
3828                                <1>       ; position the cursor for next char
3829 00001508 FEC2                 <1>       inc   dl           ; next column
3830                                <1>       ;cmp  dl, [CRT_COLS]
3831 0000150A 80FA50               <1>       cmp   dl, 80       ; test for column overflow
3832 0000150D 0F85EB000000         <1>         jne     set_cpos
3833 00001513 B200                 <1>       mov   dl, 0        ; column = 0
3834                                <1> u10:                     ; (line feed found)
3835 00001515 80FE18               <1>       cmp   dh, 25-1     ; check for last row
3836 00001518 722F                 <1>       jb    short u6
3837                                <1>       ;
3838                                <1>       ; scroll required
3839                                <1> u1:
3840                                <1>       ; SET CURSOR POSITION (04/12/2013)
3841 0000151A E8DF000000           <1>       call  set_cpos
3842                                <1>       ;
3843                                <1>       ; determine value to fill with during scroll
3844                                <1> u2:
3845                                <1>       ; READ_AC_CURRENT         :
3846                                <1>       ;   THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
3847                                <1>       ;    AT THE CURRENT CURSOR POSITION
3848                                <1>       ;
3849                                <1>       ; INPUT
3850                                <1>       ;     (AH) = CURRENT CRT MODE
3851                                <1>       ;     (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
3852                                <1>       ;     (DS) = DATA SEGMENT
3853                                <1>       ;     (ES) = REGEN SEGMENT
3854                                <1>       ; OUTPUT
3855                                <1>       ;     (AL) = CHARACTER READ
```

```
3856                                    <1>       ;     (AH) = ATTRIBUTE READ
3857                                    <1>       ;
3858                                    <1>       ; mov  ah, [CRT_MODE] ; move current mode into ah
3859                                    <1>       ;
3860                                    <1>       ; bl = video page number
3861                                    <1>       ;
3862 0000151F E837010000               <1>       call   find_position; get regen location and port address
3863                                    <1>       ; dx = status port
3864                                    <1>       ; esi = cursor location/address
3865                                    <1> p11:
3866 00001524 FB                       <1>       sti                ; enable interrupts
3867 00001525 90                       <1>       nop                ; allow for small interupts window
3868 00001526 FA                       <1>       cli                ; blocks interrupts for single loop
3869 00001527 EC                       <1>       in     al, dx      ; get status from adapter
3870 00001528 A801                     <1>       test   al, RHRZ    ; is horizontal retrace low
3871 0000152A 75F8                     <1>       jnz    short p11   ; wait until it is
3872                                    <1> p12:                     ; now wait for either retrace high
3873 0000152C EC                       <1>       in     al, dx      ; get status
3874 0000152D A809                     <1>       test   al, RVRT+RHRZ; is horizontal or vertical retrace high
3875 0000152F 74FB                     <1>       jz     short p12   ; wait until either is active
3876                                    <1> p13:
3877 00001531 81C600800B00             <1>       add    esi, 0B8000h ; 30/08/2014 (Retro UNIX 386 v1)
3878 00001537 668B06                   <1>       mov    ax, [esi]    ; get the character and attribute
3879                                    <1>       ;
3880                                    <1>       ; al = character, ah = attribute
3881                                    <1>       ;
3882 0000153A FB                       <1>       sti
3883                                    <1>       ; bl = video page number
3884                                    <1> u3:
3885                                    <1>       ;;mov  ax, 0601h    ; scroll one line
3886                                    <1>       ;;sub  cx, cx       ; upper left corner
3887                                    <1>       ;;mov  dh, 25-1     ; lower right row
3888                                    <1>       ;;;mov dl, [CRT_COLS]
3889                                    <1>       ;mov   dl, 80       ; lower right column
3890                                    <1>       ;;dec  dl
3891                                    <1>       ;;mov  dl, 79
3892                                    <1>
3893                                    <1>       ;;call scroll_up    ; 04/12/2013
3894                                    <1>       ;;; 11/03/2015
3895                                    <1>       ; 02/09/2014
3896                                    <1>       ;;;mov cx, [crt_ulc] ; Upper left corner  (0000h)
3897                                    <1>       ;;;mov dx, [crt_lrc] ; Lower right corner (184Fh)
3898                                    <1>       ; 11/03/2015
3899 0000153B 6629C9                   <1>       sub    cx, cx
3900 0000153E 66BA4F18                 <1>       mov    dx, 184Fh ; dl= 79 (column), dh = 24 (row)
3901                                    <1>       ;
3902 00001542 B001                     <1>       mov    al, 1        ; scroll 1 line up
3903                                    <1>              ; ah = attribute
3904 00001544 E93E010000               <1>       jmp    scroll_up
3905                                    <1> ;u4:
3906                                    <1>       ;;int  10h          ; video-call return
3907                                    <1>                           ; scroll up the screen
3908                                    <1>                           ; tty return
3909                                    <1> ;u5:
3910                                    <1>       ;retn               ; return to the caller
3911                                    <1>
3912                                    <1> u6:                       ; set-cursor-inc
3913 00001549 FEC6                     <1>       inc    dh           ; next row
3914                                    <1>                           ; set cursor
3915                                    <1> ;u7:
3916                                    <1>       ;;mov  ah, 02h
3917                                    <1>       ;;jmp  short u4     ; establish the new cursor
3918                                    <1>       ;call  set_cpos
3919                                    <1>       ;jmp   short u5
3920 0000154B E9AE000000               <1>       jmp     set_cpos
3921                                    <1>
3922                                    <1>       ; check for control characters
3923                                    <1> u8:
3924 00001550 7438                     <1>       je     short u9
3925 00001552 3C0A                     <1>       cmp    al, 0Ah             ; is it a line feed (0Ah)
3926 00001554 74BF                     <1>       je     short u10
3927 00001556 3C07                     <1>       cmp    al, 07h      ; is it a bell
3928 00001558 7434                     <1>       je     short u11
3929 0000155A 3C08                     <1>       cmp    al, 08h             ; is it a backspace
3930                                    <1>       ;jne   short u0
3931 0000155C 7424                     <1>       je     short bs     ; 12/12/2013
3932                                    <1>       ; 12/12/2013 (tab stop)
3933 0000155E 3C09                     <1>       cmp    al, 09h             ; is it a tab stop
3934 00001560 75A1                     <1>       jne    short u0
3935 00001562 88D0                     <1>       mov    al, dl
3936 00001564 6698                     <1>       cbw
3937 00001566 B108                     <1>       mov    cl, 8
3938 00001568 F6F1                     <1>       div    cl
3939 0000156A 28E1                     <1>       sub    cl, ah
3940                                    <1> ts:
3941                                    <1>       ; 02/09/2014
3942                                    <1>       ; 01/09/2014
3943 0000156C B020                     <1>       mov    al, 20h
3944                                    <1> tsloop:
3945 0000156E 6651                     <1>       push   cx
3946 00001570 6650                     <1>       push   ax
3947 00001572 30FF                     <1>       xor    bh, bh
3948                                    <1>       ;mov   bl, [active_page]
3949 00001574 E878FFFFFF               <1>       call   m3
3950 00001579 6658                     <1>       pop    ax  ; ah = attribute/color
3951 0000157B 6659                     <1>       pop    cx
3952 0000157D FEC9                     <1>       dec    cl
3953 0000157F 75ED                     <1>       jnz    short tsloop
3954 00001581 C3                       <1>       retn
3955                                    <1> bs:
3956                                    <1>       ; back space found
3957                                    <1>
3958 00001582 08D2                     <1>       or     dl, dl              ; is it already at start of line
3959                                    <1>       ;je    short u7     ; set_cursor
3960 00001584 7478                     <1>       jz     short set_cpos
```

```
3961 00001586 664A              <1>       dec    dx                  ; no -- just move it back
3962                            <1>       ;jmp   short u7
3963 00001588 EB74              <1>       jmp    short set_cpos
3964                            <1>
3965                            <1>       ; carriage return found
3966                            <1> u9:
3967 0000158A B200              <1>       mov    dl, 0        ; move to first column
3968                            <1>       ;jmp   short u7
3969 0000158C EB70              <1>       jmp    short set_cpos
3970                            <1>
3971                            <1>       ; line feed found
3972                            <1> ;u10:
3973                            <1> ;     cmp    dh, 25-1     ; bottom of screen
3974                            <1> ;     jne    short u6     ; no, just set the cursor
3975                            <1> ;       jmp     u1              ; yes, scroll the screen
3976                            <1>
3977                            <1> beeper:
3978                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
3979                            <1>       ; 18/01/2014
3980                            <1>       ; 03/12/2013
3981                            <1>       ; bell found
3982                            <1> u11:
3983 0000158E FB                <1>       sti
3984 0000158F 3A1D[96700000]    <1>       cmp    bl, [active_page]
3985 00001595 7551              <1>       jne    short u12    ; Do not sound the beep
3986                            <1>                           ; if it is not written on the active page
3987 00001597 66B93305          <1>       mov    cx, 1331     ; divisor for 896 hz tone
3988 0000159B B31F              <1>       mov    bl, 31       ; set count for 31/64 second for beep
3989                            <1>       ;call  beep         ; sound the pod bell
3990                            <1>       ;jmp   short u5     ; tty_return
3991                            <1>       ;retn
3992                            <1>
3993                            <1> TIMER equ    040h              ; 8254 TIMER - BASE ADDRESS
3994                            <1> PORT_B       equ    061h        ; PORT B READ/WRITE DIAGNOSTIC REGISTER
3995                            <1> GATE2 equ    00000001b    ; TIMER 2 INPUT CATE CLOCK BIT
3996                            <1> SPK2  equ    00000010b    ; SPEAKER OUTPUT DATA ENABLE BIT
3997                            <1>
3998                            <1> beep:
3999                            <1>       ; 07/02/2015
4000                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
4001                            <1>       ; 18/01/2014
4002                            <1>       ; 03/12/2013
4003                            <1>       ;
4004                            <1>       ; TEST4.ASM - 06/10/85  POST AND BIOS UTILITY ROUTINES
4005                            <1>       ;
4006                            <1>       ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
4007                            <1>       ;
4008                            <1>       ; ENTRY:
4009                            <1>       ;    (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
4010                            <1>       ;    (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
4011                            <1>       ; EXIT:                          :
4012                            <1>       ;    (AX),(BL),(CX) MODIFIED.
4013                            <1>
4014 0000159D 9C                <1>       pushf  ; 18/01/2014 ; save interrupt status
4015 0000159E FA                <1>       cli                 ; block interrupts during update
4016 0000159F B0B6              <1>       mov    al, 10110110b; select timer 2, lsb, msb binary
4017 000015A1 E643              <1>       out    TIMER+3, al  ; write timer mode register
4018 000015A3 EB00              <1>       jmp    $+2          ; I/O delay
4019 000015A5 88C8              <1>       mov    al, cl       ; divisor for hz (low)
4020 000015A7 E642              <1>       out    TIMER+2,AL   ; write timer 2 count - lsb
4021 000015A9 EB00              <1>       jmp    $+2          ; I/O delay
4022 000015AB 88E8              <1>       mov    al, ch       ; divisor for hz (high)
4023 000015AD E642              <1>       out    TIMER+2, al  ; write timer 2 count - msb
4024 000015AF E461              <1>       in     al, PORT_B   ; get current setting of port
4025 000015B1 88C4              <1>       mov    ah, al       ; save that setting
4026 000015B3 0C03              <1>       or     al, GATE2+SPK2      ; gate timer 2 and turn speaker on
4027 000015B5 E661              <1>       out    PORT_B, al   ; and restore interrupt status
4028                            <1>       ;popf  ; 18/01/2014
4029 000015B7 FB                <1>       sti
4030                            <1> g7:                        ; 1/64 second per count (bl)
4031 000015B8 B90B040000        <1>       mov    ecx, 1035    ; delay count for 1/64 of a second
4032 000015BD E827000000        <1>       call   waitf        ; go to beep delay 1/64 count
4033 000015C2 FECB              <1>       dec    bl           ; (bl) length count expired?
4034 000015C4 75F2              <1>       jnz    short g7     ; no - continue beeping speaker
4035                            <1>       ;
4036                            <1>       ;pushf              ; save interrupt status
4037 000015C6 FA                <1>       cli    ; 18/01/2014 ; block interrupts during update
4038 000015C7 E461              <1>       in     al, PORT_B   ; get current port value
4039                            <1>         ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
4040 000015C9 0CFC              <1>         or      al, ~(GATE2+SPK2)
4041 000015CB 20C4              <1>         and  ah, al         ; someone turned them off during beep
4042 000015CD 88E0              <1>       mov    al, ah       ; recover value of port
4043                            <1>         ;or      al, not (GATE2+SPK2) ; force speaker data off
4044 000015CF 0CFC              <1>       or     al, ~(GATE2+SPK2) ; isolate current speaker bits in case
4045 000015D1 E661              <1>       out    PORT_B, al   ; and stop speaker timer
4046                            <1>       ;popf               ; restore interrupt flag state
4047 000015D3 FB                <1>       sti
4048 000015D4 B90B040000        <1>       mov    ecx, 1035    ; force 1/64 second delay (short)
4049 000015D9 E80B000000        <1>       call   waitf        ; minimum delay between all beeps
4050                            <1>       ;pushf              ; save interrupt status
4051 000015DE FA                <1>       cli                 ; block interrupts during update
4052 000015DF E461              <1>       in     al, PORT_B   ; get current port value in case
4053 000015E1 2403              <1>       and    al, GATE2+SPK2      ; someone turned them on
4054 000015E3 08E0              <1>       or     al, ah       ; recover value of port_b
4055 000015E5 E661              <1>       out    PORT_B, al   ; restore speaker status
4056 000015E7 9D                <1>       popf                ; restore interrupt flag state
4057                            <1> u12:
4058 000015E8 C3                <1>       retn
4059                            <1>
4060                            <1> REFRESH_BIT equ    00010000b    ; REFRESH TEST BIT
4061                            <1>
4062                            <1> WAITF:
4063                            <1> waitf:
4064                            <1>       ; 30/08/2014 (Retro UNIX 386 v1)
4065                            <1>       ; 03/12/2013
```

```
4066                                    <1>       ;
4067                                    <1> ;     push ax                   ; save work register (ah)
4068                                    <1> ;waitf1:
4069                                    <1>                                 ; use timer 1 output bits
4070                                    <1> ;     in    al, PORT_B    ; read current counter output status
4071                                    <1> ;     and   al, REFRESH_BIT     ; mask for refresh determine bit
4072                                    <1> ;     cmp   al, ah        ; did it just change
4073                                    <1> ;     je    short waitf1 ; wait for a change in output line
4074                                    <1> ;     ;
4075                                    <1> ;     mov   ah, al        ; save new lflag state
4076                                    <1> ;     loop  waitf1        ; decrement half cycles till count end
4077                                    <1> ;     ;
4078                                    <1> ;     pop   ax            ; restore (ah)
4079                                    <1> ;     retn                ; return (cx)=0
4080                                    <1>
4081                                    <1> ; 06/02/2015 (unix386.s <-- dsectrm2.s)
4082                                    <1> ; 17/12/2014 (dsectrm2.s)
4083                                    <1> ; WAITF
4084                                    <1> ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
4085                                    <1> ;
4086                                    <1> ;---WAITF-----------------------------------------------------------------------
4087                                    <1> ;     FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
4088                                    <1> ; ENTRY:
4089                                    <1> ;     (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
4090                                    <1> ;            MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
4091                                    <1> ; EXIT:
4092                                    <1> ;                  AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
4093                                    <1> ;     (CX) = 0
4094                                    <1> ;-----------------------------------------------------------------------------
4095                                    <1>
4096                                    <1> ; Refresh period: 30 micro seconds (15-80 us)
4097                                    <1> ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)
4098                                    <1>
4099                                    <1> ;WAITF:                             ; DELAY FOR (CX)*15.085737 US
4100 000015E9 6650                     <1>       PUSH  AX                  ; SAVE WORK REGISTER (AH)
4101                                    <1>       ; 16/12/2014
4102                                    <1>       ;shr   cx, 1              ; convert to count of 30 micro seconds
4103 000015EB D1E9                     <1>       shr   ecx, 1 ; 21/02/2015
4104                                    <1> ;17/12/2014
4105                                    <1> ;WAITF1:
4106                                    <1> ;     IN    AL, PORT_B   ;061h  ; READ CURRENT COUNTER OUTPUT STATUS
4107                                    <1> ;     AND   AL, REFRESH_BIT     ;00010000b ; MASK FOR REFRESH DETERMINE BIT
4108                                    <1> ;     CMP   AL, AH              ; DID IT JUST CHANGE
4109                                    <1> ;     JE    short WAITF1        ; WAIT FOR A CHANGE IN OUTPUT LINE
4110                                    <1> ;     MOV   AH, AL              ; SAVE NEW FLAG STATE
4111                                    <1> ;     LOOP  WAITF1              ; DECREMENT HALF CYCLES TILL COUNT END
4112                                    <1>       ;
4113                                    <1>       ; 17/12/2014
4114                                    <1>       ;
4115                                    <1>       ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
4116                                    <1>       ;
4117                                    <1> ;WAIT_REFRESH:  Uses port 61, bit 4 to have CPU speed independent waiting.
4118                                    <1> ;     INPUT:  CX = number of refresh periods to wait
4119                                    <1> ;                  (refresh periods = 1 per 30 microseconds on most machines)
4120                                    <1> WR_STATE_0:
4121 000015ED E461                     <1>       IN    AL,PORT_B           ; IN AL,SYS1
4122 000015EF A810                     <1>       TEST  AL,010H
4123 000015F1 74FA                     <1>       JZ    SHORT WR_STATE_0
4124                                    <1> WR_STATE_1:
4125 000015F3 E461                     <1>       IN    AL,PORT_B           ; IN AL,SYS1
4126 000015F5 A810                     <1>       TEST  AL,010H
4127 000015F7 75FA                     <1>       JNZ   SHORT WR_STATE_1
4128 000015F9 E2F2                     <1>         LOOP    WR_STATE_0
4129                                    <1>       ;
4130 000015FB 6658                     <1>       POP   AX                  ; RESTORE (AH)
4131 000015FD C3                       <1>       RETn                      ; (CX) = 0
4132                                    <1>
4133                                    <1> set_cpos:
4134                                    <1>       ; 27/06/2015
4135                                    <1>       ; 01/09/2014
4136                                    <1>       ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
4137                                    <1>       ;
4138                                    <1>       ; 12/12/2013 (Retro UNIX 8086 v1 - last update)
4139                                    <1>       ; 04/12/2013 (Retro UNIX 8086 v1 - beginning)
4140                                    <1>       ;
4141                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4142                                    <1>       ;
4143                                    <1>       ; SET_CPOS
4144                                    <1>       ;     THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
4145                                    <1>       ;     NEW X-Y VALUES PASSED
4146                                    <1>       ; INPUT
4147                                    <1>       ;     DX - ROW,COLUMN OF NEW CURSOR
4148                                    <1>       ;     BH - DISPLAY PAGE OF CURSOR
4149                                    <1>       ; OUTPUT
4150                                    <1>       ;     CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
4151                                    <1>       ;
4152 000015FE 0FB6C3                   <1>         movzx   eax, bl  ; BL = video page number ; 27/06/2015 (movzx)
4153 00001601 D0E0                     <1>         shl     al, 1   ; word offset
4154 00001603 BE[86700000]             <1>       mov   esi, cursor_posn
4155 00001608 01C6                     <1>         add     esi, eax
4156 0000160A 668916                   <1>       mov   [esi], dx ; save the pointer
4157 0000160D 381D[96700000]           <1>       cmp   [active_page], bl
4158 00001613 7532                     <1>       jne   short m17
4159                                    <1>       ;call  m18    ; CURSOR SET
4160                                    <1> ;m17:             ; SET_CPOS_RETURN
4161                                    <1>       ; 01/09/2014
4162                                    <1> ;     retn
4163                                    <1>             ; DX  = row/column
4164                                    <1> m18:
4165 00001615 E832000000               <1>       call  position ; determine location in regen buffer
4166 0000161A 668B0D[84700000]         <1>       mov   cx, [CRT_START]
4167 00001621 6601C1                   <1>       add   cx, ax  ; add char position in regen buffer
4168                                    <1>                     ; to the start address (offset) for this page
4169 00001624 66D1E9                   <1>       shr   cx, 1  ; divide by 2 for char only count
4170 00001627 B40E                     <1>       mov   ah, 14 ; register number for cursor
```

```
4171                                    <1>       ;call  m16    ; output value to the 6845
4172                                    <1>       ;retn
4173                                    <1>
4174                                    <1>       ;----- THIS ROUTINE OUTPUTS THE CX REGISTER
4175                                    <1>       ;      TO THE 6845 REGISTERS NAMED IN (AH)
4176                                    <1> m16:
4177 00001629 FA                       <1>       cli
4178                                    <1>       ;mov   dx, [addr_6845] ; address register
4179 0000162A 66BAD403                 <1>       mov    dx, 03D4h ; I/O address of color card
4180 0000162E 88E0                     <1>       mov    al, ah ; get value
4181 00001630 EE                       <1>       out    dx, al ; register set
4182 00001631 6642                     <1>       inc    dx     ; data register
4183 00001633 EB00                     <1>       jmp    $+2    ; i/o delay
4184 00001635 88E8                     <1>       mov    al, ch ; data
4185 00001637 EE                       <1>       out    dx, al
4186 00001638 664A                     <1>       dec    dx
4187 0000163A 88E0                     <1>       mov    al, ah
4188 0000163C FEC0                     <1>       inc    al     ; point to other data register
4189 0000163E EE                       <1>       out    dx, al ; set for second register
4190 0000163F 6642                     <1>       inc    dx
4191 00001641 EB00                     <1>       jmp    $+2    ; i/o delay
4192 00001643 88C8                     <1>       mov    al, cl ; second data value
4193 00001645 EE                       <1>       out    dx, al
4194 00001646 FB                       <1>       sti
4195                                    <1> m17:
4196 00001647 C3                       <1>       retn
4197                                    <1>
4198                                    <1>
4199                                    <1> set_ctype:
4200                                    <1>       ; 02/09/2014 (Retro UNIX 386 v1)
4201                                    <1>       ;
4202                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4203                                    <1>
4204                                    <1> ;     CH) = BITS 4-0 = START LINE FOR CURSOR
4205                                    <1> ;       ** HARDWARE WILL ALWAYS CAUSE BLINK
4206                                    <1> ;       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
4207                                    <1> ;          OR NO CURSOR AT ALL
4208                                    <1> ;     (CL) = BITS 4-0 = END LINE FOR CURSOR
4209                                    <1>
4210                                    <1> ;-----------------------------------------------
4211                                    <1> ; SET_CTYPE
4212                                    <1> ;     THIS ROUTINE SETS THE CURSOR VALUE
4213                                    <1> ; INPUT
4214                                    <1> ;     (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
4215                                    <1> ; OUTPUT
4216                                    <1> ;     NONE
4217                                    <1> ;-----------------------------------------------
4218                                    <1>
4219 00001648 B40A                     <1>       mov    ah, 10 ; 6845 register for cursor set
4220                                    <1>       ;mov   [CURSOR_MODE], cx ; save in data area
4221                                    <1>       ;call  m16    ; output cx register
4222                                    <1>       ;retn
4223 0000164A EBDD                     <1>       jmp    m16
4224                                    <1>
4225                                    <1>
4226                                    <1> position:
4227                                    <1>       ; 27/06/2015
4228                                    <1>       ; 02/09/2014
4229                                    <1>       ; 30/08/2014 (Retro UNIX 386 v1)
4230                                    <1>       ; 04/12/2013 (Retro UNIX 8086 v1)
4231                                    <1>       ;
4232                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4233                                    <1>       ;
4234                                    <1>       ; POSITION
4235                                    <1>       ;     THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
4236                                    <1>       ;     OF A CHARACTER IN THE ALPHA MODE
4237                                    <1>       ; INPUT
4238                                    <1>       ;     AX = ROW, COLUMN POSITION
4239                                    <1>       ; OUTPUT
4240                                    <1>       ;     AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
4241                                    <1>
4242                                    <1>             ; DX = ROW, COLUMN POSITION
4243                                    <1>       ;movzx eax, byte [CRT_COLS] ; 27/06/2015
4244 0000164C 31C0                     <1>       xor    eax, eax ; 02/09/2014
4245 0000164E B050                     <1>       mov    al, 80   ; determine bytes to row
4246 00001650 F6E6                     <1>       mul    dh ;    row value
4247 00001652 30F6                     <1>       xor    dh, dh   ; 0
4248 00001654 6601D0                   <1>       add    ax, dx  ; add column value to the result
4249 00001657 66D1E0                   <1>       shl    ax, 1  ; * 2 for attribute bytes
4250                                    <1>             ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
4251 0000165A C3                       <1>       retn
4252                                    <1>
4253                                    <1> find_position:
4254                                    <1>       ; 27/06/2015
4255                                    <1>       ; 07/09/2014
4256                                    <1>       ; 02/09/2014
4257                                    <1>       ; 30/08/2014 (Retro UNIX 386 v1)
4258                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4259 0000165B 0FB6CB                   <1>       movzx  ecx, bl ; video page number ; 27/06/2015 (movzx)
4260 0000165E 89CE                     <1>       mov    esi, ecx
4261 00001660 66D1E6                   <1>       shl    si, 1
4262 00001663 668B96[86700000]         <1>       mov    dx, [esi + cursor_posn]
4263 0000166A 740A                     <1>       jz     short p21
4264 0000166C 6631F6                   <1>       xor    si, si
4265                                    <1> p20:
4266                                    <1>       ;add   si, [CRT_LEN]
4267 0000166F 6681C6A00F               <1>       add    si, 80*25*2 ; add length of buffer for one page
4268 00001674 E2F9                     <1>       loop   p20
4269                                    <1> p21:
4270 00001676 6621D2                   <1>       and    dx, dx
4271 00001679 7407                     <1>       jz     short p22
4272 0000167B E8CCFFFFFF               <1>       call   position ; determine location in regen in page
4273 00001680 01C6                     <1>       add    esi, eax ; add location to start of regen page
4274                                    <1> p22:
4275                                    <1>       ;mov   dx, [addr_6845] ; get base address of active display
```

```
4276                                    <1>       ;mov   dx, 03D4h ; I/O address of color card
4277                                    <1>       ;add   dx, 6  ; point at status port
4278 00001682 66BADA03                  <1>       mov    dx, 03DAh ; status port
4279                                    <1>       ; cx = 0
4280 00001686 C3                        <1>       retn
4281                                    <1>
4282                                    <1> scroll_up:
4283                                    <1>       ; 16/01/2016
4284                                    <1>       ; 07/09/2014
4285                                    <1>       ; 02/09/2014
4286                                    <1>       ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
4287                                    <1>       ; 04/04/2014
4288                                    <1>       ; 04/12/2013
4289                                    <1>       ;
4290                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4291                                    <1>       ;
4292                                    <1>       ; SCROLL UP
4293                                    <1>       ;     THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
4294                                    <1>       ;     ON THE SCREEN
4295                                    <1>       ; INPUT
4296                                    <1>       ;     (AH) = CURRENT CRT MODE
4297                                    <1>       ;     (AL) = NUMBER OF ROWS TO SCROLL
4298                                    <1>       ;     (CX) = ROW/COLUMN OF UPPER LEFT CORNER
4299                                    <1>       ;     (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
4300                                    <1>       ;     (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
4301                                    <1>       ;     (DS) = DATA SEGMENT
4302                                    <1>       ;     (ES) = REGEN BUFFER SEGMENT
4303                                    <1>       ; OUTPUT
4304                                    <1>       ;     NONE -- THE REGEN BUFFER IS MODIFIED
4305                                    <1>       ;
4306                                    <1>       ;     bh = 0  (02/09/2014)
4307                                    <1>       ;
4308                                    <1>       ; ((ah = 3))
4309                                    <1>       ; cl = left upper column
4310                                    <1>       ; ch = left upper row
4311                                    <1>       ; dl = right lower column
4312                                    <1>       ; dh = right lower row
4313                                    <1>       ;
4314                                    <1>       ; al = line count
4315                                    <1>       ; ah = attribute to be used on blanked line
4316                                    <1>       ; bl = video page number (0 to 7)
4317                                    <1>       ;
4318                                    <1>
4319                                    <1>       ; Test Line Count
4320 00001687 08C0                     <1>       or     al, al
4321 00001689 740C                     <1>       jz     short al_set
4322 0000168B 88F7                     <1>       mov    bh, dh ; subtract lower row from upper row
4323 0000168D 28EF                     <1>       sub    bh, ch
4324 0000168F FEC7                     <1>       inc    bh     ; adjust difference by 1
4325 00001691 38C7                     <1>       cmp    bh, al       ; line count = amount of rows in window?
4326 00001693 7502                     <1>       jne    short al_set ; if not the we're all set
4327 00001695 30C0                     <1>       xor    al, al ; otherwise set al to zero
4328                                    <1> al_set:
4329 00001697 30FF                     <1>       xor    bh, bh ; 0
4330 00001699 6650                     <1>       push   ax
4331                                    <1>       ;mov   esi, [crt_base]
4332 0000169B BE00800B00               <1>         mov     esi, 0B8000h
4333 000016A0 3A1D[96700000]           <1>         cmp     bl, [active_page]
4334 000016A6 750B                     <1>       jne    short n0
4335                                    <1>       ;
4336 000016A8 66A1[84700000]           <1>         mov     ax, [CRT_START]
4337 000016AE 6601C6                   <1>         add     si, ax
4338 000016B1 EB0F                     <1>         jmp     short n1
4339                                    <1> n0:
4340 000016B3 20DB                     <1>         and     bl, bl
4341 000016B5 740B                     <1>       jz     short n1
4342 000016B7 88D8                     <1>       mov    al, bl
4343                                    <1> n0x:
4344                                    <1>         ;add    si, [CRT_LEN]
4345                                    <1>         ;add    esi, 80*25*2
4346 000016B9 6681C6A00F               <1>         add     si, 80*25*2
4347 000016BE FEC8                     <1>         dec  al
4348 000016C0 75F7                     <1>       jnz    short n0x
4349                                    <1> n1:
4350                                    <1>         ;Scroll position
4351 000016C2 6652                     <1>       push   dx
4352 000016C4 6689CA                   <1>       mov    dx, cx ; now, upper left position in DX
4353 000016C7 E880FFFFFF               <1>       call   position
4354 000016CC 01C6                     <1>       add    esi, eax
4355 000016CE 89F7                     <1>       mov    edi, esi
4356 000016D0 665A                     <1>       pop    dx     ; lower right position in DX
4357 000016D2 6629CA                   <1>       sub    dx, cx
4358 000016D5 FEC6                     <1>       inc    dh     ; dh = #rows
4359 000016D7 FEC2                     <1>       inc    dl     ; dl = #cols in block
4360 000016D9 6658                     <1>       pop    ax     ; al = line count, ah = attribute
4361 000016DB 31C9                     <1>       xor    ecx, ecx
4362 000016DD 6689C1                   <1>       mov    cx, ax
4363                                    <1>       ;mov   ah, [CRT_COLS]
4364 000016E0 B450                     <1>       mov    ah, 80
4365 000016E2 F6E4                     <1>       mul    ah     ; determine offset to from address
4366 000016E4 6601C0                   <1>       add    ax, ax  ; *2 for attribute byte
4367                                    <1>       ;
4368 000016E7 6650                     <1>       push   ax     ; offset
4369 000016E9 6652                     <1>       push   dx
4370                                    <1>       ;
4371                                    <1>       ; 04/04/2014
4372 000016EB 66BADA03                 <1>       mov    dx, 3DAh ; guaranteed to be color card here
4373                                    <1> n8:                       ; wait_display_enable
4374 000016EF EC                       <1>         in      al, dx   ; get port
4375 000016F0 A808                     <1>       test   al, RVRT ; wait for vertical retrace
4376 000016F2 74FB                     <1>       jz     short n8 ; wait_display_enable
4377 000016F4 B025                     <1>       mov    al, 25h
4378 000016F6 B2D8                     <1>       mov    dl, 0D8h ; address control port
4379 000016F8 EE                       <1>       out    dx, al ; turn off video during vertical retrace
4380 000016F9 665A                     <1>       pop    dx     ; #rows, #cols
```

```
4381 000016FB 6658                <1>             pop    ax     ; offset
4382 000016FD 6691                <1>       xchg   ax, cx ;
4383                              <1>       ; ecx = offset, al = line count, ah = attribute
4384                              <1> ;n9:
4385 000016FF 08C0                <1>       or     al, al
4386 00001701 7420                <1>         jz      short n3
4387 00001703 01CE                <1>         add     esi, ecx ; from address for scroll
4388 00001705 88F7                <1>       mov    bh, dh  ; #rows in block
4389 00001707 28C7                <1>       sub    bh, al ; #rows to be moved
4390                              <1> n2:
4391                              <1>       ; Move rows
4392 00001709 88D1                <1>       mov    cl, dl ; get # of cols to move
4393 0000170B 56                  <1>       push   esi
4394 0000170C 57                  <1>       push   edi    ; save start address
4395                              <1> n10:
4396 0000170D 66A5                <1>       movsw         ; move that line on screen
4397 0000170F FEC9                <1>       dec    cl
4398 00001711 75FA                <1>         jnz     short n10
4399 00001713 5F                  <1>       pop    edi
4400 00001714 5E                  <1>       pop    esi    ; recover addresses
4401                              <1>         ;mov    cl, [CRT_COLS]
4402                              <1>       ;add   cl, cl
4403                              <1>         ;mov    ecx, 80*2
4404 00001715 66B9A000            <1>         mov     cx, 80*2
4405 00001719 01CE                <1>         add     esi, ecx  ; next line
4406 0000171B 01CF                <1>         add     edi, ecx
4407 0000171D FECF                <1>       dec    bh      ; count of lines to move
4408 0000171F 75E8                <1>       jnz    short n2 ; row loop
4409                              <1>       ; bh = 0
4410 00001721 88C6                <1>       mov    dh, al  ; #rows
4411                              <1> n3:
4412                              <1>       ; attribute in ah
4413 00001723 B020                <1>       mov    al, ' '       ; fill with blanks
4414                              <1> n3x:
4415                              <1>       ; Clear rows
4416                              <1>                ; dh =  #rows
4417 00001725 88D1                <1>         mov cl, dl ; get # of cols to clear
4418 00001727 57                  <1>         push    edi     ; save address
4419                              <1> n11:
4420 00001728 66AB                <1>         stosw           ; store fill character
4421 0000172A FEC9                <1>       dec    cl
4422 0000172C 75FA                <1>         jnz     short n11
4423 0000172E 5F                  <1>         pop     edi     ; recover address
4424                              <1>       ;mov   cl, [CRT_COLS]
4425                              <1>       ;add   cl, cl
4426                              <1>         ;mov    ecx, 80*2
4427 0000172F B1A0                <1>         mov cl, 80*2
4428 00001731 01CF                <1>         add     edi, ecx
4429 00001733 FECE                <1>       dec    dh
4430 00001735 75EE                <1>       jnz    short n3x ; 16/01/2016
4431                              <1>       ;
4432 00001737 3A1D[96700000]      <1>       cmp    bl, [active_page]
4433 0000173D 7507                <1>       jne    short n6
4434                              <1>       ;mov   al, [CRT_MODE_SET] ; get the value of mode set
4435 0000173F B029                <1>       mov    al, 29h ; (ORGS.ASM), M7 mode set table value for mode 3
4436 00001741 66BAD803            <1>       mov    dx, 03D8h ; always set color card port
4437 00001745 EE                  <1>       out    dx, al
4438                              <1> n6:
4439 00001746 C3                  <1>       retn
4440                              <1>
4441                              <1>
4442                              <1> write_c_current:
4443                              <1>       ; 30/08/2014 (Retro UNIX 386 v1)
4444                              <1>       ; 18/01/2014
4445                              <1>       ; 04/12/2013
4446                              <1>       ;
4447                              <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4448                              <1>       ;
4449                              <1>       ; WRITE_C_CURRENT
4450                              <1>       ;     THIS ROUTINE WRITES THE CHARACTER AT
4451                              <1>       ;     THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
4452                              <1>       ; INPUT
4453                              <1>       ;     (AH) = CURRENT CRT MODE
4454                              <1>       ;     (BH) = DISPLAY PAGE
4455                              <1>       ;     (CX) = COUNT OF CHARACTERS TO WRITE
4456                              <1>       ;     (AL) = CHAR TO WRITE
4457                              <1>       ;     (DS) = DATA SEGMENT
4458                              <1>       ;     (ES) = REGEN SEGMENT
4459                              <1>       ; OUTPUT
4460                              <1>       ;     DISPLAY REGEN BUFFER UPDATED
4461                              <1>
4462 00001747 FA                  <1>       cli
4463                              <1>       ; bl = video page
4464                              <1>       ; al = character
4465                              <1>       ; ah = color/attribute
4466 00001748 6652                <1>       push   dx
4467 0000174A 6650                <1>       push   ax     ; save character & attribute/color
4468 0000174C E80AFFFFFF          <1>       call   find_position  ; get regen location and port address
4469                              <1>       ; esi = regen location
4470                              <1>       ; dx = status port
4471                              <1>       ;
4472                              <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE
4473                              <1>       ;
4474                              <1> p41:              ; wait for horizontal retrace is low or vertical
4475 00001751 FB                  <1>       sti           ; enable interrupts first
4476 00001752 3A1D[96700000]      <1>         cmp     bl, [active_page]
4477 00001758 7510                <1>       jne    short p44
4478 0000175A FA                  <1>       cli           ; block interrupts for single loop
4479 0000175B EC                  <1>       in     al, dx ; get status from the adapter
4480 0000175C A808                <1>       test   al, RVRT ; check for vertical retrace first
4481 0000175E 7509                <1>       jnz    short p43 ; Do fast write now if vertical retrace
4482 00001760 A801                <1>       test   al, RHRZ ; is horizontal retrace low
4483 00001762 75ED                <1>       jnz    short p41 ; wait until it is
4484                              <1> p42:              ;  wait for either retrace high
4485 00001764 EC                  <1>       in     al, dx ; get status again
```

```
4486 00001765 A809              <1>       test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
4487 00001767 74FB              <1>       jz     short p42 ; wait until either retrace active
4488                            <1> p43:
4489 00001769 FB                <1>       sti
4490                            <1> p44:
4491 0000176A 6658              <1>       pop    ax     ; restore the character (al) & attribute (ah)
4492 0000176C 81C600800B00      <1>       add    esi, 0B8000h ; 30/08/2014 (crt_base)
4493                            <1>                           ; Retro UNIX 386 v1 feature only!
4494 00001772 668906            <1>       mov    [esi], ax
4495 00001775 665A              <1>       pop    dx
4496 00001777 C3                <1>       retn
4497                            <1>
4498                            <1> set_mode:
4499                            <1>       ; 16/01/2016
4500                            <1>       ; 02/09/2014 (Retro UNIX 386 v1)
4501                            <1>       ;
4502                            <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4503                            <1>
4504                            <1> ;--------------------------------------------------
4505                            <1> ; SET MODE                                      :
4506                            <1> ;     THIS ROUTINE INITIALIZES THE ATTACHMENT TO    :
4507                            <1> ;     THE SELECTED MODE, THE SCREEN IS BLANKED.     :
4508                            <1> ; INPUT                                          :
4509                            <1> ;     (AL) - MODE SELECTED (RANGE 0-7)       :
4510                            <1> ; OUTPUT                                        :
4511                            <1> ;     NONE                                      :
4512                            <1> ;--------------------------------------------------
4513                            <1>
4514 00001778 57                <1>       push   edi ; 16/01/2016
4515 00001779 53                <1>       push   ebx
4516 0000177A 52                <1>       push   edx
4517 0000177B 51                <1>       push   ecx ; 16/01/2016
4518 0000177C 50                <1>         push    eax
4519                            <1>
4520                            <1>       ;mov   dx, 03D4h    ; address or color card
4521 0000177D B003              <1>       mov    al, 3
4522                            <1> ;M8:
4523 0000177F A2[506A0000]      <1>       mov    [CRT_MODE], al  ; save mode in global variable
4524 00001784 B029              <1>       mov    al, 29h
4525                            <1>       ;mov   [CRT_MODE_SET], al ; save the mode set value
4526 00001786 2437              <1>       and    al, 037h     ; video off, save high resolution bit
4527                            <1>       ;push  dx           ; save port value
4528                            <1>       ;add   dx, 4        ; point to control register
4529 00001788 66BAD803          <1>       mov    dx, 3D8h
4530 0000178C EE                <1>       out    dx, al       ; reset video to off to suppress rolling
4531                            <1>       ;pop   dx
4532                            <1> ;M9:
4533 0000178D BB[886A0000]      <1>       mov    ebx, video_params ; initialization table
4534                            <1>       ;mov   ax, [ebx+10]      ; get the cursor mode from the table
4535                            <1>       ;xchg  ah, al
4536                            <1>       ;mov   [CURSOR_MODE], ax ; save cursor mode
4537 00001792 30E4              <1>       xor    ah, ah         ; ah is register number during loop
4538                            <1>
4539                            <1> ;-----      LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
4540 00001794 B910000000        <1>       mov    ecx, 16 ; 16/01/2016
4541                            <1> M10:                ;  initialization loop
4542 00001799 88E0              <1>       mov    al, ah       ; get 6845 register number
4543 0000179B EE                <1>       out    dx, al
4544 0000179C 6642              <1>       inc    dx      ; point to data port
4545 0000179E FEC4              <1>       inc    ah     ; next register value
4546 000017A0 8A03              <1>       mov    al, [ebx] ; get table value
4547 000017A2 EE                <1>       out    dx, al ; out to chip
4548 000017A3 43                <1>       inc    ebx    ; next in table
4549 000017A4 664A              <1>       dec    dx     ; back to pointer register
4550 000017A6 E2F1              <1>       loop   M10    ; do the whole table
4551                            <1>
4552                            <1> ;-----      FILL REGEN AREA WITH BLANK
4553                            <1>       ;xor   ax, ax
4554                            <1>       ;mov   [CRT_START], ax  ; start address saved in global
4555                            <1>       ;mov   [ACTIVE_PAGE], al ; 0 ; (re)set page value
4556                            <1>       ;mov   ecx, 8192 ; number of words in color card
4557                            <1>       ; black background, light gray characeter color, space character
4558                            <1>       ;mov   ax, 0720h ; fill char for alpha - attribute
4559                            <1> ;M13:               ; clear buffer
4560                            <1>       ;add   edi, 0B8000h ; [crt_base]
4561                            <1>       ;rep   stosw  ; FILL THE REGEN BUFFER WITH BLANKS
4562                            <1>
4563                            <1> ;-----      ENABLE VIDEO AND CORRECT PORT SETTING
4564                            <1>       ;mov   dx, 3D4h ; mov dx, word [ADDR_6845]
4565                            <1>                       ; prepare to output to video enable port
4566                            <1>       ;add   dx,4    ; point to the mode control gerister
4567 000017A8 66BAD803          <1>       mov    dx, 3D8h
4568                            <1>       ;mov   al, [CRT_MODE_SET] ; get the mode set value
4569 000017AC B029              <1>       mov    al, 29h
4570 000017AE EE                <1>       out    dx, al  ; set video enable port
4571                            <1>
4572                            <1> ;-----      DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
4573                            <1> ;-----      AND THE NUMBER TO BE USED FOR TTY INTERFACE
4574                            <1>       ;
4575                            <1>       ;mov byte [CRT_COLS], 80h ; initialize number of columns count
4576                            <1>       ;
4577                            <1> ;-----      SET CURSOR POSITIONS
4578                            <1>       ;push  edi
4579                            <1>       ;mov   word [CRT_LEN], 80*25*2
4580 000017AF BF[86700000]      <1>       mov    edi, cursor_posn
4581 000017B4 B904000000        <1>       mov    ecx, 4 ; clear all cursor positions (16 bytes)
4582 000017B9 31C0              <1>       xor    eax, eax
4583 000017BB F3AB              <1>       rep    stosd  ; fill with zeroes
4584                            <1>       ;pop   edi
4585                            <1>
4586                            <1> ;-----      SET UP OVERSCAN REGISTER
4587 000017BD 6642              <1>       inc    dx     ; set overscan port to a default
4588 000017BF B030              <1>       mov    al, 30h      ; 30H valuye for all modes except 640X200 bw
4589                            <1> ;M14:
4590 000017C1 EE                <1>       out    dx, al ; output the correct value to 3D9 port
```

```
4591                                    <1>       ;mov   [CRT_PALETTE], al ; save the value for future use
4592                                    <1>
4593                                    <1> ;-----       NORMAL RETURN FROM ALL VIDEO RETURNS
4594                                    <1>       ;
4595 000017C2 58                       <1>       pop    eax
4596 000017C3 59                       <1>       pop    ecx ; 16/01/2016
4597 000017C4 5A                       <1>       pop    edx
4598 000017C5 5B                       <1>       pop    ebx
4599 000017C6 5F                       <1>       pop    edi ; 16/01/2016
4600 000017C7 C3                       <1>       retn
4601                                    <1>
4602                                    <1> tty_sw:
4603                                    <1>       ; 30/06/2015
4604                                    <1>       ; 27/06/2015
4605                                    <1>       ; 07/09/2014
4606                                    <1>       ; 02/09/2014 (Retro UNIX 386 v1 - beginning)
4607                                    <1>       ;
4608                                    <1>       ; (Modified registers : EAX)
4609                                    <1>       ;
4610                                    <1>         ;mov     byte [u.quant], 0  ; 04/03/2014
4611                                    <1>       ;
4612                                    <1> ;act_disp_page:
4613                                    <1>       ; 30/06/2015
4614                                    <1>       ; 04/03/2014  (act_disp_page --> tty_sw)
4615                                    <1>       ; 10/12/2013
4616                                    <1>       ; 04/12/2013
4617                                    <1>       ;
4618                                    <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4619                                    <1>       ;
4620                                    <1>       ; ACT_DISP_PAGE
4621                                    <1>       ;     THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
4622                                    <1>       ;     THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
4623                                    <1>       ; INPUT
4624                                    <1>       ;     AL HAS THE NEW ACTIVE DISPLAY PAGE
4625                                    <1>       ; OUTPUT
4626                                    <1>       ;     THE 6845 IS RESET TO DISPLAY THAT PAGE
4627                                    <1>
4628                                    <1>       ;cli
4629                                    <1>
4630 000017C8 53                       <1>       push   ebx
4631 000017C9 6651                     <1>       push   cx
4632 000017CB 6652                     <1>       push   dx
4633                                    <1>       ;
4634 000017CD A2[96700000]             <1>       mov    [active_page], al ; save active page value ; [ptty]
4635                                    <1>       ;mov   cx, [CRT_LEN] ; get saved length of regen buffer
4636 000017D2 66B9A00F                 <1>       mov    cx, 25*80*2
4637                                    <1>       ; 27/06/2015
4638 000017D6 0FB6D8                   <1>       movzx  ebx, al
4639                                    <1>       ;
4640 000017D9 6698                     <1>       cbw    ; 07/09/2014 (ah=0)
4641 000017DB 66F7E1                   <1>       mul    cx     ; display page times regen length
4642                                    <1>       ; 10/12/2013
4643 000017DE 66A3[84700000]           <1>       mov    [CRT_START], ax ; save start address for later
4644 000017E4 6689C1                   <1>       mov    cx, ax ; start address to cx
4645                                    <1>       ;sar   cx, 1
4646 000017E7 66D1E9                   <1>       shr    cx, 1  ; divide by 2 for 6845 handling
4647 000017EA B40C                     <1>       mov    ah, 12 ; 6845 register for start address
4648 000017EC E838FEFFFF               <1>       call   m16
4649                                    <1>       ;sal   bx, 1
4650                                    <1>       ; 01/09/2014
4651 000017F1 D0E3                     <1>       shl    bl, 1  ; *2 for word offset
4652 000017F3 81C3[86700000]           <1>       add    ebx, cursor_posn
4653 000017F9 668B13                   <1>       mov    dx, [ebx] ; get cursor for this page
4654 000017FC E814FEFFFF               <1>       call   m18
4655                                    <1>       ;
4656 00001801 665A                     <1>       pop    dx
4657 00001803 6659                     <1>       pop    cx
4658 00001805 5B                       <1>       pop    ebx
4659                                    <1>       ;
4660                                    <1>       ;sti
4661                                    <1>       ;
4662 00001806 C3                       <1>       retn
4663                                    <1>
4664                                    <1> ; % include 'vidata.inc' ; VIDEO DATA ; 11/03/2015
4665                                    <1>
4666                                    <1>
4667                                    <1> ; /// End Of VIDEO FUNCTIONS ///
4668
4669                                        setup_rtc_int:
4670                                        ; source: http://wiki.osdev.org/RTC
4671 00001807 FA                             cli          ; disable interrupts
4672                                         ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
4673                                         ; in order to change this ...
4674                                         ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
4675                                         ; (rate must be above 2 and not over 15)
4676                                         ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
4677 00001808 B08A                         mov    al, 8Ah
4678 0000180A E670                         out    70h, al ; set index to register A, disable NMI
4679 0000180C 90                           nop
4680 0000180D E471                         in     al, 71h ; get initial value of register A
4681 0000180F 88C4                         mov    ah, al
4682 00001811 80E4F0                       and    ah, 0F0h
4683 00001814 B08A                         mov    al, 8Ah
4684 00001816 E670                         out    70h, al ; reset index to register A
4685 00001818 88E0                         mov    al, ah
4686 0000181A 0C0F                         or     al, 0Fh      ; new rate (0Fh -> 15)
4687 0000181C E671                         out    71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
4688                                       ; enable RTC interrupt
4689 0000181E B08B                         mov    al, 8Bh ;
4690 00001820 E670                         out    70h, al ; select register B and disable NMI
4691 00001822 90                           nop
4692 00001823 E471                         in     al, 71h ; read the current value of register B
4693 00001825 88C4                         mov    ah, al  ;
4694 00001827 B08B                         mov    al, 8Bh ;
```

```
  4695 00001829 E670                    out   70h, al ; set the index again (a read will reset the index to register
B)
  4696 0000182B 88E0                    mov   al, ah  ;
  4697 0000182D 0C40                    or    al, 40h ;
  4698 0000182F E671                    out   71h, al ; write the previous value ORed with 0x40. This turns on bit 6
of register B
  4699 00001831 FB                      sti
  4700 00001832 C3                      retn
  4701
  4702                                ; Write memory information
  4703                                ; Temporary Code
  4704                                ; 06/11/2014
  4705                                ; 14/08/2015
  4706                                memory_info:
  4707 00001833 A1[6C700000]            mov   eax, [memory_size] ; in pages
  4708 00001838 50                      push  eax
  4709 00001839 C1E00C                  shl   eax, 12               ; in bytes
  4710 0000183C BB0A000000              mov   ebx, 10
  4711 00001841 89D9                    mov   ecx, ebx        ; 10
  4712 00001843 BE[CD6C0000]            mov   esi, mem_total_b_str
  4713 00001848 E8B2000000              call  bintdstr
  4714 0000184D 58                      pop   eax
  4715 0000184E B107                    mov   cl, 7
  4716 00001850 BE[F16C0000]            mov   esi, mem_total_p_str
  4717 00001855 E8A5000000              call  bintdstr
  4718                                  ; 14/08/2015
  4719 0000185A E8BD000000              call  calc_free_mem
  4720                                  ; edx = calculated free pages
  4721                                  ; ecx = 0
  4722 0000185F A1[70700000]            mov   eax, [free_pages]
  4723 00001864 39D0                    cmp   eax, edx ; calculated free mem value
  4724                                        ; and initial free mem value are same or not?
  4725 00001866 751D                    jne   short pmim ; print mem info with '?' if not
  4726 00001868 52                      push  edx ; free memory in pages
  4727                                  ;mov  eax, edx
  4728 00001869 C1E00C                  shl   eax, 12 ; convert page count
  4729                                        ; to byte count
  4730 0000186C B10A                    mov   cl, 10
  4731 0000186E BE[116D0000]            mov   esi, free_mem_b_str
  4732 00001873 E887000000              call  bintdstr
  4733 00001878 58                      pop   eax
  4734 00001879 B107                    mov   cl, 7
  4735 0000187B BE[356D0000]            mov   esi, free_mem_p_str
  4736 00001880 E87A000000              call  bintdstr
  4737                                pmim:
  4738 00001885 BE[BB6C0000]            mov   esi, msg_memory_info
  4739                                pmim_nb:
  4740 0000188A AC                      lodsb
  4741 0000188B 08C0                    or    al, al
  4742 0000188D 740D                    jz    short pmim_ok
  4743 0000188F 56                      push  esi
  4744 00001890 31DB                    xor   ebx, ebx ; 0
  4745                                        ; Video page 0 (bl=0)
  4746 00001892 B407                    mov   ah, 07h ; Black background,
  4747                                        ; light gray forecolor
  4748 00001894 E841FCFFFF              call  write_tty
  4749 00001899 5E                      pop   esi
  4750 0000189A EBEE                    jmp   short pmim_nb
  4751                                pmim_ok:
  4752 0000189C C3                      retn
  4753
  4754                                ; Convert binary number to hexadecimal string
  4755                                ; 10/05/2015
  4756                                ; dsectpm.s (28/02/2015)
  4757                                ; Retro UNIX 386 v1 - Kernel v0.2.0.6
  4758                                ; 01/12/2014
  4759                                ; 25/11/2014
  4760                                ;
  4761                                bytetohex:
  4762                                      ; INPUT ->
  4763                                      ;     AL = byte (binary number)
  4764                                      ; OUTPUT ->
  4765                                      ;     AX = hexadecimal string
  4766                                      ;
  4767 0000189D 53                      push  ebx
  4768 0000189E 31DB                    xor   ebx, ebx
  4769 000018A0 88C3                    mov   bl, al
  4770 000018A2 C0EB04                  shr   bl, 4
  4771 000018A5 8A9B[EF180000]          mov   bl, [ebx+hexchrs]
  4772 000018AB 86D8                    xchg  bl, al
  4773 000018AD 80E30F                  and   bl, 0Fh
  4774 000018B0 8AA3[EF180000]          mov   ah, [ebx+hexchrs]
  4775 000018B6 5B                      pop   ebx
  4776 000018B7 C3                      retn
  4777
  4778                                wordtohex:
  4779                                      ; INPUT ->
  4780                                      ;     AX = word (binary number)
  4781                                      ; OUTPUT ->
  4782                                      ;     EAX = hexadecimal string
  4783                                      ;
  4784 000018B8 53                      push  ebx
  4785 000018B9 31DB                    xor   ebx, ebx
  4786 000018BB 86E0                    xchg  ah, al
  4787 000018BD 6650                    push  ax
  4788 000018BF 88E3                    mov   bl, ah
  4789 000018C1 C0EB04                  shr   bl, 4
  4790 000018C4 8A83[EF180000]          mov   al, [ebx+hexchrs]
  4791 000018CA 88E3                    mov   bl, ah
  4792 000018CC 80E30F                  and   bl, 0Fh
  4793 000018CF 8AA3[EF180000]          mov   ah, [ebx+hexchrs]
  4794 000018D5 C1E010                  shl   eax, 16
  4795 000018D8 6658                    pop   ax
  4796 000018DA 5B                      pop   ebx
  4797 000018DB EBC0                    jmp   short bytetohex
```

```
                                        ;mov   bl, al
                                        ;shr   bl, 4
                                        ;mov   bl, [ebx+hexchrs]
                                        ;xchg  bl, al
                                        ;and   bl, 0Fh
                                        ;mov   ah, [ebx+hexchrs]
                                        ;pop   ebx
                                        ;retn
                                
                                    dwordtohex:
                                        ; INPUT ->
                                        ;      EAX = dword (binary number)
                                        ; OUTPUT ->
                                        ;      EDX:EAX = hexadecimal string
                                        ;
000018DD 50                             push   eax
000018DE C1E810                         shr    eax, 16
000018E1 E8D2FFFFFF                     call   wordtohex
000018E6 89C2                           mov    edx, eax
000018E8 58                             pop    eax
000018E9 E8CAFFFFFF                     call   wordtohex
000018EE C3                             retn
                                
                                    ; 10/05/2015
                                    hex_digits:
                                    hexchrs:
000018EF 303132333435363738-            db '0123456789ABCDEF'
000018F8 39414243444546
                                
                                    ; Convert binary number to decimal/numeric string
                                    ; 06/11/2014
                                    ; Temporary Code
                                    ;
                                
                                    bintdstr:
                                        ; EAX = binary number
                                        ; ESI = decimal/numeric string address
                                        ; EBX = divisor (10)
                                        ; ECX = string length (<=10)
000018FF 01CE                           add    esi, ecx
                                    btdstr0:
00001901 4E                             dec    esi
00001902 31D2                           xor    edx, edx
00001904 F7F3                           div    ebx
00001906 80C230                         add    dl, 30h
00001909 8816                           mov    [esi], dl
0000190B FEC9                           dec    cl
0000190D 740C                           jz     btdstr2
0000190F 09C0                           or     eax, eax
00001911 75EE                           jnz    short btdstr0
                                    btdstr1:
00001913 4E                             dec    esi
00001914 C60620                           mov     byte [esi], 20h ; blank space
00001917 FEC9                           dec    cl
00001919 75F8                           jnz    short btdstr1
                                    btdstr2:
0000191B C3                             retn
                                
                                    ; Calculate free memory pages on M.A.T.
                                    ; 06/11/2014
                                    ; Temporary Code
                                    ;
                                
                                    calc_free_mem:
0000191C 31D2                           xor    edx, edx
                                        ;xor   ecx, ecx
0000191E 668B0D[80700000]               mov    cx, [mat_size] ; in pages
00001925 C1E10A                         shl    ecx, 10      ; 1024 dwords per page
00001928 BE00001000                     mov    esi, MEM_ALLOC_TBL
                                    cfm0:
0000192D AD                             lodsd
0000192E 51                             push   ecx
0000192F B920000000                     mov    ecx, 32
                                    cfm1:
00001934 D1E8                           shr    eax, 1
00001936 7301                           jnc    short cfm2
00001938 42                             inc    edx
                                    cfm2:
00001939 E2F9                           loop   cfm1
0000193B 59                             pop    ecx
0000193C E2EF                           loop   cfm0
0000193E C3                             retn
                                
                                    %include 'diskio.inc'  ; 07/03/2015
                                <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
                                <1> ; Last Modification: 04/02/2016
                                <1> ;     (Initialized Disk Parameters Data is in 'DISKDATA.INC')
                                <1> ;     (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
                                <1>
                                <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
                                <1>
                                <1> ; ///////// DISK I/O SYSTEM ///////////////
                                <1>
                                <1> ; 06/02/2015
                                <1> diskette_io:
0000193F 9C                     <1>     pushfd
00001940 0E                     <1>     push   cs
00001941 E809000000             <1>     call   DISKETTE_IO_1
00001946 C3                     <1>     retn
                                <1>
                                <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
                                <1> ;////////////////////////////////////////////////////////
                                <1>
                                <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
                                <1> ; 20/02/2015
```

```
4903                            <1> ; 06/02/2015 (unix386.s)
4904                            <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
4905                            <1> ;
4906                            <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
4907                            <1> ;
4908                            <1> ; ADISK.EQU
4909                            <1>
4910                            <1> ;----- Wait control constants
4911                            <1>
4912                            <1> ;amount of time to wait while RESET is active.
4913                            <1>
4914                            <1> WAITCPU_RESET_ON   EQU    21           ;Reset on must last at least 14us
4915                            <1>                                        ;at 250 KBS xfer rate.
4916                            <1>                                        ;see INTEL MCS, 1985, pg. 5-456
4917                            <1>
4918                            <1> WAITCPU_FOR_STATUS EQU    100          ;allow 30 microseconds for
4919                            <1>                                        ;status register to become valid
4920                            <1>                                        ;before re-reading.
4921                            <1>
4922                            <1> ;After sending a byte to NEC, status register may remain
4923                            <1> ;incorrectly set for 24 us.
4924                            <1>
4925                            <1> WAITCPU_RQM_LOW          EQU    24           ;number of loops to check for
4926                            <1>                                        ;RQM low.
4927                            <1>
4928                            <1> ; COMMON.MAC
4929                            <1> ;
4930                            <1> ;     Timing macros
4931                            <1> ;
4932                            <1>
4933                            <1> %macro             SIODELAY 0                ; SHORT IODELAY
4934                            <1>             jmp short $+2
4935                            <1> %endmacro
4936                            <1>
4937                            <1> %macro             IODELAY  0                ; NORMAL IODELAY
4938                            <1>             jmp short $+2
4939                            <1>             jmp short $+2
4940                            <1> %endmacro
4941                            <1>
4942                            <1> %macro             NEWIODELAY 0
4943                            <1>             out    0ebh,al
4944                            <1> %endmacro
4945                            <1>
4946                            <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
4947                            <1> ;;; WAIT_FOR_MEM
4948                            <1> ;WAIT_FDU_INT_LO   equ    017798       ; 2.5 secs in 30 micro units.
4949                            <1> ;WAIT_FDU_INT_HI   equ    1
4950                            <1> WAIT_FDU_INT_LH           equ    83334        ; 27/02/2015 (2.5 seconds waiting)
4951                            <1> ;;; WAIT_FOR_PORT
4952                            <1> ;WAIT_FDU_SEND_LO  equ    16667        ; .5 secons in 30 us units.
4953                            <1> ;WAIT_FDU_SEND_HI  equ    0
4954                            <1> WAIT_FDU_SEND_LH   equ    16667        ; 27/02/2015
4955                            <1> ;Time to wait while waiting for each byte of NEC results = .5
4956                            <1> ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
4957                            <1> ;WAIT_FDU_RESULTS_LO      equ    16667        ; .5 seconds in 30 micro units.
4958                            <1> ;WAIT_FDU_RESULTS_HI      equ    0
4959                            <1> WAIT_FDU_RESULTS_LH       equ    16667  ; 27/02/2015
4960                            <1> ;;; WAIT_REFRESH
4961                            <1> ;amount of time to wait for head settle, per unit in parameter
4962                            <1> ;table = 1 ms.
4963                            <1> WAIT_FDU_HEAD_SETTLE      equ    33           ; 1 ms in 30 micro units.
4964                            <1>
4965                            <1>
4966                            <1> ; ///////////////// DISKETTE I/O ///////////////
4967                            <1>
4968                            <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
4969                            <1>
4970                            <1> ;----------------------------------------
4971                            <1> ;     EQUATES USED BY POST AND BIOS    :
4972                            <1> ;----------------------------------------
4973                            <1>
4974                            <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
4975                            <1> ;PORT_A            EQU    060H         ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
4976                            <1> ;PORT_B            EQU    061H         ; PORT B READ/WRITE DIAGNOSTIC REGISTER
4977                            <1> ;REFRESH_BIT EQU    00010000B    ; REFRESH TEST BIT
4978                            <1>
4979                            <1> ;----------------------------------------
4980                            <1> ;     CMOS EQUATES FOR THIS SYSTEM     :
4981                            <1> ;--------------------------------------------------------------------------------
4982                            <1> ;CMOS_PORT   EQU    070H         ; I/O ADDRESS OF CMOS ADDRESS PORT
4983                            <1> ;CMOS_DATA   EQU    071H         ; I/O ADDRESS OF CMOS DATA PORT
4984                            <1> ;NMI         EQU    10000000B    ; DISABLE NMI INTERRUPTS MASK -
4985                            <1>                                 ;  HIGH BIT OF CMOS LOCATION ADDRESS
4986                            <1>
4987                            <1> ;---------- CMOS TABLE LOCATION ADDRESS'S ## ---------------------------------
4988                            <1> CMOS_DISKETTE      EQU    010H         ; DISKETTE DRIVE TYPE BYTE      ;
4989                            <1> ;            EQU    011H         ; - RESERVED                    ;C
4990                            <1> CMOS_DISK    EQU    012H         ; FIXED DISK TYPE BYTE                  ;H
4991                            <1> ;            EQU    013H         ; - RESERVED                    ;E
4992                            <1> CMOS_EQUIP   EQU    014H         ; EQUIPMENT WORD LOW BYTE       ;C
4993                            <1>
4994                            <1> ;---------- DISKETTE EQUATES -----------------------------------------------------
4995                            <1> INT_FLAG     EQU    10000000B    ; INTERRUPT OCCURRENCE FLAG
4996                            <1> DSK_CHG      EQU    10000000B    ; DISKETTE CHANGE FLAG MASK BIT
4997                            <1> DETERMINED   EQU    00010000B    ; SET STATE DETERMINED IN STATE BITS
4998                            <1> HOME         EQU    00010000B    ; TRACK 0 MASK
4999                            <1> SENSE_DRV_ST EQU    00000100B    ; SENSE DRIVE STATUS COMMAND
5000                            <1> TRK_SLAP     EQU    030H         ; CRASH STOP (48 TPI DRIVES)
5001                            <1> QUIET_SEEK   EQU    00AH         ; SEEK TO TRACK 10
5002                            <1> ;MAX_DRV     EQU    2            ; MAX NUMBER OF DRIVES
5003                            <1> HD12_SETTLE  EQU    15           ; 1.2 M HEAD SETTLE TIME
5004                            <1> HD320_SETTLE EQU    20           ; 320 K HEAD SETTLE TIME
5005                            <1> MOTOR_WAIT   EQU    37           ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
5006                            <1>
5007                            <1> ;---------- DISKETTE ERRORS ------------------------------------------------------
```

```
5008                                <1> ;TIME_OUT    EQU    080H         ; ATTACHMENT FAILED TO RESPOND
5009                                <1> ;BAD_SEEK    EQU    040H         ; SEEK OPERATION FAILED
5010                                <1> BAD_NEC      EQU    020H         ; DISKETTE CONTROLLER HAS FAILED
5011                                <1> BAD_CRC      EQU    010H         ; BAD CRC ON DISKETTE READ
5012                                <1> MED_NOT_FND  EQU    00CH         ; MEDIA TYPE NOT FOUND
5013                                <1> DMA_BOUNDARY EQU    009H         ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
5014                                <1> BAD_DMA      EQU    008H         ; DMA OVERRUN ON OPERATION
5015                                <1> MEDIA_CHANGE EQU    006H         ; MEDIA REMOVED ON DUAL ATTACH CARD
5016                                <1> RECORD_NOT_FND     EQU    004H         ; REQUESTED SECTOR NOT FOUND
5017                                <1> WRITE_PROTECT      EQU    003H         ; WRITE ATTEMPTED ON WRITE PROTECT DISK
5018                                <1> BAD_ADDR_MARK      EQU    002H         ; ADDRESS MARK NOT FOUND
5019                                <1> BAD_CMD      EQU    001H         ; BAD COMMAND PASSED TO DISKETTE I/O
5020                                <1>
5021                                <1> ;---------- DISK CHANGE LINE EQUATES ----------------------------------------
5022                                <1> NOCHGLN      EQU    001H         ; NO DISK CHANGE LINE AVAILABLE
5023                                <1> CHGLN        EQU    002H         ; DISK CHANGE LINE AVAILABLE
5024                                <1>
5025                                <1> ;---------- MEDIA/DRIVE STATE INDICATORS ------------------------------------
5026                                <1> TRK_CAPA     EQU    00000001B    ; 80 TRACK CAPABILITY
5027                                <1> FMT_CAPA     EQU    00000010B    ; MULTIPLE FORMAT CAPABILITY (1.2M)
5028                                <1> DRV_DET      EQU    00000100B    ; DRIVE DETERMINED
5029                                <1> MED_DET      EQU    00010000B    ; MEDIA DETERMINED BIT
5030                                <1> DBL_STEP     EQU    00100000B    ; DOUBLE STEP BIT
5031                                <1> RATE_MSK     EQU    11000000B    ; MASK FOR CLEARING ALL BUT RATE
5032                                <1> RATE_500     EQU    00000000B    ; 500 KBS DATA RATE
5033                                <1> RATE_300     EQU    01000000B    ; 300 KBS DATA RATE
5034                                <1> RATE_250     EQU    10000000B    ; 250 KBS DATA RATE
5035                                <1> STRT_MSK     EQU    00001100B    ; OPERATION START RATE MASK
5036                                <1> SEND_MSK     EQU    11000000B    ; MASK FOR SEND RATE BITS
5037                                <1>
5038                                <1> ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY ------------------------
5039                                <1> M3D3U        EQU    00000000B    ; 360 MEDIA/DRIVE NOT ESTABLISHED
5040                                <1> M3D1U        EQU    00000001B    ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
5041                                <1> M1D1U        EQU    00000010B    ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
5042                                <1> MED_UNK      EQU    00000111B    ; NONE OF THE ABOVE
5043                                <1>
5044                                <1> ;---------- INTERRUPT EQUATES -----------------------------------------------
5045                                <1> ;EOI         EQU    020H         ; END OF INTERRUPT COMMAND TO 8259
5046                                <1> ;INTA00            EQU    020H         ; 8259 PORT
5047                                <1> INTA01             EQU    021H         ; 8259 PORT
5048                                <1> INTB00             EQU    0A0H         ; 2ND 8259
5049                                <1> INTB01             EQU    0A1H         ;
5050                                <1>
5051                                <1> ;--------------------------------------------------------------------------------
5052                                <1> DMA08        EQU    008H         ; DMA STATUS REGISTER PORT ADDRESS
5053                                <1> DMA          EQU    000H         ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
5054                                <1> DMA18        EQU    0D0H         ; 2ND DMA STATUS PORT ADDRESS
5055                                <1> DMA1         EQU    0C0H         ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
5056                                <1> ;--------------------------------------------------------------------------------
5057                                <1> ;TIMER             EQU    040H         ; 8254 TIMER - BASE ADDRESS
5058                                <1>
5059                                <1> ;--------------------------------------------------------------------------------
5060                                <1> DMA_PAGE     EQU    081H         ; START OF DMA PAGE REGISTERS
5061                                <1>
5062                                <1> ; 06/02/2015 (unix386.s, protected mode modifications)
5063                                <1> ; (unix386.s <-- dsectrm2.s)
5064                                <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
5065                                <1>
5066                                <1> ; 10/12/2014
5067                                <1> ;
5068                                <1> ;int40h:
5069                                <1> ;     pushf
5070                                <1> ;     push   cs
5071                                <1> ;     ;cli
5072                                <1> ;     call   DISKETTE_IO_1
5073                                <1> ;     retn
5074                                <1>
5075                                <1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS
5076                                <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
5077                                <1> ;
5078                                <1>
5079                                <1> ;-- INT13H --------------------------------------------------------------------
5080                                <1> ; DISKETTE I/O
5081                                <1> ;     THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
5082                                <1> ;     1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
5083                                <1> ; INPUT
5084                                <1> ;     (AH) =  00H RESET DISKETTE SYSTEM
5085                                <1> ;             HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
5086                                <1> ;             ON ALL DRIVES
5087                                <1> ;--------------------------------------------------------------------------------
5088                                <1> ;     (AH)= 01H  READ THE STATUS OF THE SYSTEM INTO (AH)
5089                                <1> ;             @DISKETTE_STATUS FROM LAST OPERATION IS USED
5090                                <1> ;--------------------------------------------------------------------------------
5091                                <1> ;     REGISTERS FOR READ/WRITE/VERIFY/FORMAT
5092                                <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5093                                <1> ;     (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
5094                                <1> ;     (CH) - TRACK NUMBER (NOT VALUE CHECKED)
5095                                <1> ;             MEDIA  DRIVE  TRACK NUMBER
5096                                <1> ;             320/360      320/360          0-39
5097                                <1> ;             320/360      1.2M       0-39
5098                                <1> ;             1.2M   1.2M       0-79
5099                                <1> ;             720K   720K       0-79
5100                                <1> ;             1.44M  1.44M      0-79
5101                                <1> ;     (CL) -        SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
5102                                <1> ;             MEDIA  DRIVE  SECTOR NUMBER
5103                                <1> ;             320/360      320/360           1-8/9
5104                                <1> ;             320/360      1.2M        1-8/9
5105                                <1> ;             1.2M   1.2M        1-15
5106                                <1> ;             720K   720K        1-9
5107                                <1> ;             1.44M  1.44M       1-18
5108                                <1> ;     (AL)   NUMBER OF SECTORS (NOT VALUE CHECKED)
5109                                <1> ;             MEDIA  DRIVE  MAX NUMBER OF SECTORS
5110                                <1> ;             320/360      320/360              8/9
5111                                <1> ;             320/360      1.2M           8/9
5112                                <1> ;             1.2M   1.2M         15
```

```
5113                            <1> ;           720K   720K         9
5114                            <1> ;           1.44M  1.44M        18
5115                            <1> ;
5116                            <1> ;    (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
5117                            <1> ;
5118                            <1> ;--------------------------------------------------------------------------
5119                            <1> ;    (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY
5120                            <1> ;--------------------------------------------------------------------------
5121                            <1> ;    (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY
5122                            <1> ;--------------------------------------------------------------------------
5123                            <1> ;    (AH)= 04H  VERIFY THE DESIRED SECTORS
5124                            <1> ;--------------------------------------------------------------------------
5125                            <1> ;    (AH)= 05H  FORMAT THE DESIRED TRACK
5126                            <1> ;          (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
5127                            <1> ;          FOR THE      TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
5128                            <1> ;          WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
5129                            <1> ;          N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
5130                            <1> ;          THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
5131                            <1> ;          THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
5132                            <1> ;          READ/WRITE ACCESS.
5133                            <1> ;          PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
5134                            <1> ;          ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
5135                            <1> ;          THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
5136                            <1> ;          (INT 13H, AH =  18H) MUST BE CALLED TO SET THE DISKETTE TYPE
5137                            <1> ;          THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
5138                            <1> ;          IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
5139                            <1> ;          MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
5140                            <1> ;
5141                            <1> ;          THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
5142                            <1> ;          FORMAT THE FOLLOWING MEDIAS:
5143                            <1> ;          ------------------------------------------
5144                            <1> ;          : MEDIA  :     DRIVE      : PARM 1 : PARM 2 :
5145                            <1> ;          ------------------------------------------
5146                            <1> ;          : 320K  : 320K/360K/1.2M :  50H   :   8    :
5147                            <1> ;          : 360K  : 320K/360K/1.2M :  50H   :   9    :
5148                            <1> ;          : 1.2M  : 1.2M           :  54H   :  15    :
5149                            <1> ;          : 720K  : 720K/1.44M     :  50H   :   9    :
5150                            <1> ;          : 1.44M       : 1.44M          :  6CH   :  18    :
5151                            <1> ;          ------------------------------------------
5152                            <1> ;          NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
5153                            <1> ;                 - PARM 2 = EOT (LAST SECTOR ON TRACK)
5154                            <1> ;                 - DISK BASE IS POINTED BY DISK POINTER LOCATED
5155                            <1> ;                  AT ABSOLUTE ADDRESS 0:78.
5156                            <1> ;                 - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
5157                            <1> ;                  SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.

5158                            <1> ;--------------------------------------------------------------------------
5159                            <1> ;    (AH) = 08H READ DRIVE PARAMETERS
5160                            <1> ;    REGISTERS
5161                            <1> ;      INPUT
5162                            <1> ;        (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5163                            <1> ;      OUTPUT
5164                            <1> ;        (ES:DI) POINTS TO DRIVE PARAMETER TABLE
5165                            <1> ;        (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
5166                            <1> ;        (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
5167                            <1> ;               BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
5168                            <1> ;        (DH) - MAXIMUM HEAD NUMBER
5169                            <1> ;        (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
5170                            <1> ;        (BH) - 0
5171                            <1> ;        (BL) - BITS 7 THRU 4 - 0
5172                            <1> ;               BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
5173                            <1> ;        (AX) - 0
5174                            <1> ;     UNDER THE FOLLOWING CIRCUMSTANCES:
5175                            <1> ;        (1) THE DRIVE NUMBER IS INVALID,
5176                            <1> ;        (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
5177                            <1> ;        (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
5178                            <1> ;        (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
5179                            <1> ;        THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
5180                            <1> ;        IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
5181                            <1> ;        @DISKETTE_STATUS = 0 AND CY IS RESET.
5182                            <1> ;--------------------------------------------------------------------------
5183                            <1> ;    (AH)= 15H  READ DASD TYPE
5184                            <1> ;    OUTPUT REGISTERS
5185                            <1> ;    (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
5186                            <1> ;           00 - DRIVE NOT PRESENT
5187                            <1> ;           01 - DISKETTE, NO CHANGE LINE AVAILABLE
5188                            <1> ;           02 - DISKETTE, CHANGE LINE AVAILABLE
5189                            <1> ;           03 - RESERVED (FIXED DISK)
5190                            <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5191                            <1> ;--------------------------------------------------------------------------
5192                            <1> ;    (AH)= 16H  DISK CHANGE LINE STATUS
5193                            <1> ;    OUTPUT REGISTERS
5194                            <1> ;    (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
5195                            <1> ;           06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
5196                            <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5197                            <1> ;--------------------------------------------------------------------------
5198                            <1> ;    (AH)= 17H  SET DASD TYPE FOR FORMAT
5199                            <1> ;    INPUT REGISTERS
5200                            <1> ;    (AL) - 00 - NOT USED
5201                            <1> ;           01 - DISKETTE 320/360K IN 360K DRIVE
5202                            <1> ;           02 - DISKETTE 360K IN 1.2M DRIVE
5203                            <1> ;           03 - DISKETTE 1.2M IN 1.2M DRIVE
5204                            <1> ;           04 - DISKETTE 720K IN 720K DRIVE
5205                            <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
5206                            <1> ;           (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
5207                            <1> ;--------------------------------------------------------------------------
5208                            <1> ;    (AH)= 18H  SET MEDIA TYPE FOR FORMAT
5209                            <1> ;    INPUT REGISTERS
5210                            <1> ;    (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
5211                            <1> ;    (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
5212                            <1> ;           BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
5213                            <1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
5214                            <1> ;    OUTPUT REGISTERS:
5215                            <1> ;    (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
5216                            <1> ;            UNCHANGED IF (AH) IS NON-ZERO
```

```
                        <1> ;     (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
                        <1> ;          - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
                        <1> ;          - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
                        <1> ;          - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
                        <1> ;--------------------------------------------------------------------------------
                        <1> ;     DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
                        <1> ;     THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
                        <1> ;     ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
                        <1> ;           ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
                        <1> ;           IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
                        <1> ;           CHANGE ERROR CODE
                        <1> ;           IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
                        <1> ;           TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
                        <1> ;     IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
                        <1> ;
                        <1> ; DATA VARIABLE -- @DISK_POINTER
                        <1> ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                        <1> ;--------------------------------------------------------------------------------
                        <1> ; OUTPUT FOR ALL FUNCTIONS
                        <1> ;     AH = STATUS OF OPERATION
                        <1> ;           STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
                        <1> ;           VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                        <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
                        <1> ;           TYPE AH=(15)).
                        <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
                        <1> ;     FOR READ/WRITE/VERIFY
                        <1> ;           DS,BX,DX,CX PRESERVED
                        <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                        <1> ;           ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                        <1> ;           ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                        <1> ;           THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                        <1> ;           PROBLEM IS NOT DUE TO MOTOR START-UP.
                        <1> ;--------------------------------------------------------------------------------
                        <1> ;
                        <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
                        <1> ;
                        <1> ;   ---------------------------------------------------------------
                        <1> ;   |       |       |       |       |       |       |       |       |
                        <1> ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
                        <1> ;   |       |       |       |       |       |       |       |       |
                        <1> ;   ---------------------------------------------------------------
                        <1> ;     |       |       |       |     |     |     |     |
                        <1> ;     |       |       |       |     |     ----------------
                        <1> ;     |       |       |       |     |           |
                        <1> ;     |       |       |       |   RESERVED      |
                        <1> ;     |       |       |       |                 PRESENT STATE
                        <1> ;     |       |       |       |     000: 360K IN 360K DRIVE UNESTABLISHED
                        <1> ;     |       |       |       |     001: 360K IN 1.2M DRIVE UNESTABLISHED
                        <1> ;     |       |       |       |     010: 1.2M IN 1.2M DRIVE UNESTABLISHED
                        <1> ;     |       |       |       |     011: 360K IN 360K DRIVE ESTABLISHED
                        <1> ;     |       |       |       |     100: 360K IN 1.2M DRIVE ESTABLISHED
                        <1> ;     |       |       |       |     101: 1.2M IN 1.2M DRIVE ESTABLISHED
                        <1> ;     |       |       |       |     110: RESERVED
                        <1> ;     |       |       |       |     111: NONE OF THE ABOVE
                        <1> ;     |       |       |       |
                        <1> ;     |       |       |     ------>     MEDIA/DRIVE ESTABLISHED
                        <1> ;     |       |       |
                        <1> ;     |       |     -------------->    DOUBLE STEPPING REQUIRED (360K IN 1.2M
                        <1> ;     |       |                 DRIVE)
                        <1> ;     |       |
                        <1> ;     ----------------------------->  DATA TRANSFER RATE FOR THIS DRIVE:
                        <1> ;
                        <1> ;                                     00: 500 KBS
                        <1> ;                                     01: 300 KBS
                        <1> ;                                     10: 250 KBS
                        <1> ;                                     11: RESERVED
                        <1> ;
                        <1> ;
                        <1> ;--------------------------------------------------------------------------------
                        <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
                        <1> ;--------------------------------------------------------------------------------
                        <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
                        <1> ;--------------------------------------------------------------------------------
                        <1>
                        <1> struc MD
00000000 <res 00000001> <1>       .SPEC1       resb   1      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00000001 <res 00000001> <1>       .SPEC2       resb   1      ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00000002 <res 00000001> <1>       .OFF_TIM     resb   1      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00000003 <res 00000001> <1>       .BYT_SEC     resb   1      ; 512 BYTES/SECTOR
00000004 <res 00000001> <1>       .SEC_TRK     resb   1      ; EOT (LAST SECTOR ON TRACK)
00000005 <res 00000001> <1>       .GAP         resb   1      ; GAP LENGTH
00000006 <res 00000001> <1>       .DTL         resb   1      ; DTL
00000007 <res 00000001> <1>       .GAP3        resb   1      ; GAP LENGTH FOR FORMAT
00000008 <res 00000001> <1>       .FIL_BYT     resb   1      ; FILL BYTE FOR FORMAT
00000009 <res 00000001> <1>       .HD_TIM              resb   1      ; HEAD SETTLE TIME (MILLISECONDS)
0000000A <res 00000001> <1>       .STR_TIM     resb   1      ; MOTOR START TIME (1/8 SECONDS)
0000000B <res 00000001> <1>       .MAX_TRK     resb   1      ; MAX. TRACK NUMBER
0000000C <res 00000001> <1>       .RATE        resb   1      ; DATA TRANSFER RATE
                        <1> endstruc
                        <1>
                        <1> BIT7OFF     EQU    7FH
                        <1> BIT7ON      EQU    80H
                        <1>
                        <1> ;;int13h: ; 16/02/2015
                        <1> ;; 16/02/2015 - 21/02/2015
                        <1> int40h:
00001947 9C             <1>       pushfd
00001948 0E             <1>       push   cs
00001949 E801000000     <1>       call   DISKETTE_IO_1
0000194E C3             <1>       retn
                        <1>
                        <1> DISKETTE_IO_1:
                        <1>
0000194F FB             <1>       STI                     ; INTERRUPTS BACK ON
00001950 55             <1>       PUSH   eBP              ; USER REGISTER
```

```
5322 00001951 57                <1>       PUSH   eDI                ; USER REGISTER
5323 00001952 52                <1>       PUSH   eDX                ; HEAD #, DRIVE # OR USER REGISTER
5324 00001953 53                <1>       PUSH   eBX                ; BUFFER OFFSET PARAMETER OR REGISTER
5325 00001954 51                <1>       PUSH   eCX                ; TRACK #-SECTOR # OR USER REGISTER
5326 00001955 89E5              <1>       MOV    eBP,eSP                 ; BP     => PARAMETER LIST DEP. ON AH
5327                            <1>                                ; [BP]   = SECTOR #
5328                            <1>                                ; [BP+1] = TRACK #
5329                            <1>                                ; [BP+2] = BUFFER OFFSET
5330                            <1>                                ; FOR RETURN OF DRIVE PARAMETERS:
5331                            <1>                                ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
5332                            <1>                                ;          BITS 0-5 MAX SECTORS/TRACK
5333                            <1>                                ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
5334                            <1>                                ; BL/[BP+2] = BITS 7-4 = 0
5335                            <1>                                ;           BITS 3-0 = VALID CMOS TYPE
5336                            <1>                                ; BH/[BP+3] = 0
5337                            <1>                                ; DL/[BP+4] = # DRIVES INSTALLED
5338                            <1>                                ; DH/[BP+5] = MAX HEAD #
5339                            <1>                                ; DI/[BP+6] = OFFSET TO DISK BASE
5340 00001957 06                <1>       push   es ; 06/02/2015
5341 00001958 1E                <1>       PUSH   DS                 ; BUFFER SEGMENT PARM OR USER REGISTER
5342 00001959 56                <1>       PUSH   eSI                ; USER REGISTERS
5343                            <1>       ;CALL  DDS                ; SEGMENT OF BIOS DATA AREA TO DS
5344                            <1>       ;mov   cx, cs
5345                            <1>       ;mov   ds, cx
5346 0000195A 66B91000          <1>       mov    cx, KDATA
5347 0000195E 8ED9              <1>         mov     ds, cx
5348 00001960 8EC1              <1>         mov     es, cx
5349                            <1>
5350                            <1>       ;CMP   AH,(FNC_TAE-FNC_TAB)/2    ; CHECK FOR > LARGEST FUNCTION
5351 00001962 80FC19            <1>       cmp    ah,(FNC_TAE-FNC_TAB)/4 ; 18/02/2015
5352 00001965 7202              <1>       JB     short OK_FUNC      ; FUNCTION OK
5353 00001967 B414              <1>       MOV    AH,14H             ; REPLACE WITH KNOWN INVALID FUNCTION
5354                            <1> OK_FUNC:
5355 00001969 80FC01            <1>       CMP    AH,1               ; RESET OR STATUS ?
5356 0000196C 760C              <1>       JBE    short OK_DRV       ; IF RESET OR STATUS DRIVE ALWAYS OK
5357 0000196E 80FC08            <1>       CMP    AH,8               ; READ DRIVE PARMS ?
5358 00001971 7407              <1>       JZ     short OK_DRV       ; IF SO DRIVE CHECKED LATER
5359 00001973 80FA01            <1>       CMP    DL,1               ; DRIVES 0 AND 1 OK
5360 00001976 7602              <1>       JBE    short OK_DRV       ; IF 0 OR 1 THEN JUMP
5361 00001978 B414              <1>       MOV    AH,14H             ; REPLACE WITH KNOWN INVALID FUNCTION
5362                            <1> OK_DRV:
5363 0000197A 31C9              <1>       xor    ecx, ecx
5364                            <1>       ;mov   esi, ecx ; 08/02/2015
5365 0000197C 89CF              <1>       mov    edi, ecx ; 08/02/2015
5366 0000197E 88E1              <1>       MOV    CL,AH              ; CL = FUNCTION
5367                            <1>       ;XOR   CH,CH              ; CX = FUNCTION
5368                            <1>       ;SHL   CL, 1              ; FUNCTION TIMES 2
5369 00001980 C0E102            <1>       SHL    CL, 2 ; 20/02/2015  ; FUNCTION TIMES 4 (for 32 bit offset)
5370 00001983 BB[BB190000]      <1>       MOV    eBX,FNC_TAB        ; LOAD START OF FUNCTION TABLE
5371 00001988 01CB              <1>       ADD    eBX,eCX                 ; ADD OFFSET INTO TABLE => ROUTINE
5372 0000198A 88F4              <1>       MOV    AH,DH              ; AX = HEAD #,# OF SECTORS OR DASD TYPE
5373 0000198C 30F6              <1>       XOR    DH,DH              ; DX = DRIVE #
5374 0000198E 6689C6            <1>       MOV    SI,AX              ; SI = HEAD #,# OF SECTORS OR DASD TYPE
5375 00001991 6689D7            <1>       MOV     DI,DX                  ; DI = DRIVE #
5376                            <1>       ;
5377                            <1>       ; 11/12/2014
5378 00001994 8815[116B0000]    <1>         mov     [cfd], dl              ; current floppy drive (for 'GET_PARM')
5379                            <1>       ;
5380 0000199A 8A25[EC700000]    <1>       MOV    AH, [DSKETTE_STATUS]      ; LOAD STATUS TO AH FOR STATUS FUNCTION
5381 000019A0 C605[EC700000]00  <1>       MOV    byte [DSKETTE_STATUS],0   ; INITIALIZE FOR ALL OTHERS
5382                            <1>
5383                            <1> ;     THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
5384                            <1> ;     THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
5385                            <1> ;     FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
5386                            <1> ;
5387                            <1> ;           DI     : DRIVE #
5388                            <1> ;           SI-HI  : HEAD #
5389                            <1> ;           SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
5390                            <1> ;           ES     : BUFFER SEGMENT
5391                            <1> ;           [BP]   : SECTOR #
5392                            <1> ;           [BP+1] : TRACK #
5393                            <1> ;           [BP+2] : BUFFER OFFSET
5394                            <1> ;
5395                            <1> ;     ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
5396                            <1> ;     SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
5397                            <1> ;     CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
5398                            <1> ;     SPECIFIC ERROR CODE.
5399                            <1> ;
5400                            <1>                                ; (AH) = @DSKETTE_STATUS
5401 000019A7 FF13              <1>       CALL   dWORD [eBX]         ; CALL THE REQUESTED FUNCTION
5402 000019A9 5E                <1>       POP    eSI                 ; RESTORE ALL REGISTERS
5403 000019AA 1F                <1>       POP    DS
5404 000019AB 07                <1>       pop    es     ; 06/02/2015
5405 000019AC 59                <1>       POP    eCX
5406 000019AD 5B                <1>       POP    eBX
5407 000019AE 5A                <1>       POP    eDX
5408 000019AF 5F                <1>       POP    eDI
5409 000019B0 89E5              <1>       MOV    eBP, eSP
5410 000019B2 50                <1>       PUSH   eAX
5411 000019B3 9C                <1>       PUSHFd
5412 000019B4 58                <1>       POP    eAX
5413                            <1>       ;MOV   [BP+6], AX
5414 000019B5 89450C            <1>       mov    [ebp+12], eax  ; 18/02/2015, flags
5415 000019B8 58                <1>       POP    eAX
5416 000019B9 5D                <1>       POP    eBP
5417 000019BA CF                <1>       IRETd
5418                            <1>
5419                            <1> ;---------------------------------------------------------------------------
5420                            <1> ; DW --> dd (06/02/2015)
5421 000019BB [1F1A0000]        <1> FNC_TAB     dd     DSK_RESET           ; AH = 00H; RESET
5422 000019BF [981A0000]        <1>       dd     DSK_STATUS          ; AH = 01H; STATUS
5423 000019C3 [A91A0000]        <1>       dd     DSK_READ            ; AH = 02H; READ
5424 000019C7 [BA1A0000]        <1>       dd     DSK_WRITE           ; AH = 03H; WRITE
5425 000019CB [CB1A0000]        <1>       dd     DSK_VERF            ; AH = 04H; VERIFY
5426 000019CF [DC1A0000]        <1>       dd     DSK_FORMAT          ; AH = 05H; FORMAT
```

```
5427 000019D3 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 06H; INVALID
5428 000019D7 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 07H; INVALID
5429 000019DB [6E1B0000]          <1>       dd    DSK_PARMS           ; AH = 08H; READ DRIVE PARAMETERS
5430 000019DF [611B0000]          <1>       dd    FNC_ERR                   ; AH = 09H; INVALID
5431 000019E3 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0AH; INVALID
5432 000019E7 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0BH; INVALID
5433 000019EB [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0CH; INVALID
5434 000019EF [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0DH; INVALID
5435 000019F3 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0EH; INVALID
5436 000019F7 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 0FH; INVALID
5437 000019FB [611B0000]          <1>       dd    FNC_ERR                   ; AH = 10H; INVALID
5438 000019FF [611B0000]          <1>       dd    FNC_ERR                   ; AH = 11H; INVALID
5439 00001A03 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 12H; INVALID
5440 00001A07 [611B0000]          <1>       dd    FNC_ERR                   ; AH = 13H; INVALID
5441 00001A0B [611B0000]          <1>       dd    FNC_ERR                   ; AH = 14H; INVALID
5442 00001A0F [2F1C0000]          <1>       dd    DSK_TYPE            ; AH = 15H; READ DASD TYPE
5443 00001A13 [5A1C0000]          <1>       dd    DSK_CHANGE          ; AH = 16H; CHANGE STATUS
5444 00001A17 [941C0000]          <1>       dd    FORMAT_SET          ; AH = 17H; SET DASD TYPE
5445 00001A1B [171D0000]          <1>       dd    SET_MEDIA           ; AH = 18H; SET MEDIA TYPE
5446                              <1> FNC_TAE EQU     $                       ; END
5447                              <1>
5448                              <1> ;--------------------------------------------------------------------------
5449                              <1> ; DISK_RESET(AH = 00H)
5450                              <1> ;           RESET THE DISKETTE SYSTEM.
5451                              <1> ;
5452                              <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5453                              <1> ;--------------------------------------------------------------------------
5454                              <1> DSK_RESET:
5455 00001A1F 66BAF203            <1>       MOV   DX,03F2H            ; ADAPTER CONTROL PORT
5456 00001A23 FA                  <1>       CLI                       ; NO INTERRUPTS
5457 00001A24 A0[EA700000]        <1>       MOV   AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
5458 00001A29 243F                <1>       AND   AL,00111111B        ; KEEP SELECTED AND MOTOR ON BITS
5459 00001A2B C0C004              <1>       ROL   AL,4                ; MOTOR VALUE TO HIGH NIBBLE
5460                              <1>                                 ; DRIVE SELECT TO LOW NIBBLE
5461 00001A2E 0C08                <1>       OR    AL,00001000B        ; TURN ON INTERRUPT ENABLE
5462 00001A30 EE                  <1>       OUT   DX,AL               ; RESET THE ADAPTER
5463 00001A31 C605[E9700000]00    <1>       MOV   byte [SEEK_STATUS],0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
5464                              <1>       ;JMP  $+2                 ; WAIT FOR I/O
5465                              <1>       ;JMP  $+2                 ; WAIT FOR I/O (TO INSURE MINIMUM
5466                              <1>                                 ;      PULSE WIDTH)
5467                              <1>       ; 19/12/2014
5468                              <1>       NEWIODELAY
5469 00001A38 E6EB                <2>  out 0ebh,al
5470                              <1>
5471                              <1>       ; 17/12/2014
5472                              <1>       ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
5473 00001A3A B915000000          <1>       mov   ecx, WAITCPU_RESET_ON     ; cx = 21 -- Min. 14 micro seconds !?
5474                              <1> wdw1:
5475                              <1>       NEWIODELAY   ; 27/02/2015
5476 00001A3F E6EB                <2>  out 0ebh,al
5477 00001A41 E2FC                <1>       loop  wdw1
5478                              <1>       ;
5479 00001A43 0C04                <1>       OR    AL,00000100B        ; TURN OFF RESET BIT
5480 00001A45 EE                  <1>       OUT   DX,AL               ; RESET THE ADAPTER
5481                              <1>       ; 16/12/2014
5482                              <1>       IODELAY
5483 00001A46 EB00                <2>  jmp short $+2
5484 00001A48 EB00                <2>  jmp short $+2
5485                              <1>       ;
5486                              <1>       ;STI                      ; ENABLE THE INTERRUPTS
5487 00001A4A E8250C0000          <1>       CALL  WAIT_INT            ; WAIT FOR THE INTERRUPT
5488 00001A4F 723E                <1>       JC    short DR_ERR        ; IF ERROR, RETURN IT
5489 00001A51 66B9C000            <1>       MOV   CX,11000000B        ; CL = EXPECTED @NEC_STATUS
5490                              <1> NXT_DRV:
5491 00001A55 6651                <1>       PUSH  CX                  ; SAVE FOR CALL
5492 00001A57 B8[8D1A0000]        <1>       MOV   eAX, DR_POP_ERR     ; LOAD NEC_OUTPUT ERROR ADDRESS
5493 00001A5C 50                  <1>       PUSH  eAX                 ; "
5494 00001A5D B408                <1>       MOV   AH,08H              ; SENSE INTERRUPT STATUS COMMAND
5495 00001A5F E8030B0000          <1>       CALL  NEC_OUTPUT
5496 00001A64 58                  <1>       POP   eAX                 ; THROW AWAY ERROR RETURN
5497 00001A65 E83A0C0000          <1>       CALL  RESULTS                   ; READ IN THE RESULTS
5498 00001A6A 6659                <1>       POP   CX                  ; RESTORE AFTER CALL
5499 00001A6C 7221                <1>       JC    short DR_ERR        ; ERROR RETURN
5500 00001A6E 3A0D[ED700000]      <1>       CMP   CL, [NEC_STATUS]    ; TEST FOR DRIVE READY TRANSITION
5501 00001A74 7519                <1>       JNZ   short DR_ERR        ; EVERYTHING OK
5502 00001A76 FEC1                <1>       INC   CL                  ; NEXT EXPECTED @NEC_STATUS
5503 00001A78 80F9C3              <1>       CMP   CL,11000011B        ; ALL POSSIBLE DRIVES CLEARED
5504 00001A7B 76D8                <1>       JBE   short NXT_DRV       ; FALL THRU IF 11000100B OR >
5505                              <1>       ;
5506 00001A7D E852030000          <1>       CALL  SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
5507                              <1> RESBAC:
5508 00001A82 E806090000          <1>       CALL  SETUP_END           ; VARIOUS CLEANUPS
5509 00001A87 6689F3              <1>       MOV   BX,SI               ; GET SAVED AL TO BL
5510 00001A8A 88D8                <1>       MOV   AL,BL               ; PUT BACK FOR RETURN
5511 00001A8C C3                  <1>       RETn
5512                              <1> DR_POP_ERR:
5513 00001A8D 6659                <1>       POP   CX                  ; CLEAR STACK
5514                              <1> DR_ERR:
5515 00001A8F 800D[EC700000]20    <1>       OR    byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
5516 00001A96 EBEA                <1>       JMP   SHORT RESBAC        ; RETURN FROM RESET
5517                              <1>
5518                              <1> ;--------------------------------------------------------------------------
5519                              <1> ; DISK_STATUS      (AH = 01H)
5520                              <1> ;     DISKETTE STATUS.
5521                              <1> ;
5522                              <1> ; ON ENTRY:  AH : STATUS OF PREVIOUS OPERATION
5523                              <1> ;
5524                              <1> ; ON EXIT:  AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
5525                              <1> ;--------------------------------------------------------------------------
5526                              <1> DSK_STATUS:
5527 00001A98 8825[EC700000]      <1>       MOV   [DSKETTE_STATUS],AH ; PUT BACK FOR SETUP END
5528 00001A9E E8EA080000          <1>       CALL  SETUP_END           ; VARIOUS CLEANUPS
5529 00001AA3 6689F3              <1>       MOV   BX,SI               ; GET SAVED AL TO BL
5530 00001AA6 88D8                <1>       MOV   AL,BL               ; PUT BACK FOR RETURN
5531 00001AA8 C3                  <1>       RETn
```

```
5532                                <1>
5533                                <1> ;--------------------------------------------------------------------------
5534                                <1> ; DISK_READ  (AH = 02H)
5535                                <1> ;     DISKETTE READ.
5536                                <1> ;
5537                                <1> ; ON ENTRY:  DI     : DRIVE #
5538                                <1> ;            SI-HI  : HEAD #
5539                                <1> ;            SI-LOW : # OF SECTORS
5540                                <1> ;            ES     : BUFFER SEGMENT
5541                                <1> ;            [BP]   : SECTOR #
5542                                <1> ;            [BP+1] : TRACK #
5543                                <1> ;            [BP+2] : BUFFER OFFSET
5544                                <1> ;
5545                                <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5546                                <1> ;--------------------------------------------------------------------------
5547                                <1>
5548                                <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
5549                                <1>
5550                                <1> DSK_READ:
5551 00001AA9 8025[EA700000]7F     <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
5552 00001AB0 66B846E6             <1>       MOV    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
5553 00001AB4 E825040000           <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
5554 00001AB9 C3                   <1>       RETn
5555                                <1>
5556                                <1> ;--------------------------------------------------------------------------
5557                                <1> ; DISK_WRITE (AH = 03H)
5558                                <1> ;     DISKETTE WRITE.
5559                                <1> ;
5560                                <1> ; ON ENTRY:  DI     : DRIVE #
5561                                <1> ;            SI-HI  : HEAD #
5562                                <1> ;            SI-LOW : # OF SECTORS
5563                                <1> ;            ES     : BUFFER SEGMENT
5564                                <1> ;            [BP]   : SECTOR #
5565                                <1> ;            [BP+1] : TRACK #
5566                                <1> ;            [BP+2] : BUFFER OFFSET
5567                                <1> ;
5568                                <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5569                                <1> ;--------------------------------------------------------------------------
5570                                <1>
5571                                <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
5572                                <1>
5573                                <1> DSK_WRITE:
5574 00001ABA 66B84AC5             <1>       MOV    AX,0C54AH           ; AX = NEC COMMAND, DMA COMMAND
5575 00001ABE 800D[EA700000]80     <1>         OR      byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
5576 00001AC5 E814040000           <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
5577 00001ACA C3                   <1>       RETn
5578                                <1>
5579                                <1> ;--------------------------------------------------------------------------
5580                                <1> ; DISK_VERF  (AH = 04H)
5581                                <1> ;     DISKETTE VERIFY.
5582                                <1> ;
5583                                <1> ; ON ENTRY:  DI     : DRIVE #
5584                                <1> ;            SI-HI  : HEAD #
5585                                <1> ;            SI-LOW : # OF SECTORS
5586                                <1> ;            ES     : BUFFER SEGMENT
5587                                <1> ;            [BP]   : SECTOR #
5588                                <1> ;            [BP+1] : TRACK #
5589                                <1> ;            [BP+2] : BUFFER OFFSET
5590                                <1> ;
5591                                <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5592                                <1> ;--------------------------------------------------------------------------
5593                                <1> DSK_VERF:
5594 00001ACB 8025[EA700000]7F     <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
5595 00001AD2 66B842E6             <1>       MOV    AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
5596 00001AD6 E803040000           <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
5597 00001ADB C3                   <1>       RETn
5598                                <1>
5599                                <1> ;--------------------------------------------------------------------------
5600                                <1> ; DISK_FORMAT      (AH = 05H)
5601                                <1> ;     DISKETTE FORMAT.
5602                                <1> ;
5603                                <1> ; ON ENTRY:  DI     : DRIVE #
5604                                <1> ;            SI-HI  : HEAD #
5605                                <1> ;            SI-LOW : # OF SECTORS
5606                                <1> ;            ES     : BUFFER SEGMENT
5607                                <1> ;            [BP]   : SECTOR #
5608                                <1> ;            [BP+1] : TRACK #
5609                                <1> ;            [BP+2] : BUFFER OFFSET
5610                                <1> ;            @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
5611                                <1> ;
5612                                <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5613                                <1> ;--------------------------------------------------------------------------
5614                                <1> DSK_FORMAT:
5615 00001ADC E83C030000           <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
5616 00001AE1 E838050000           <1>       CALL   FMT_INIT            ; ESTABLISH STATE IF UNESTABLISHED
5617 00001AE6 800D[EA700000]80     <1>         OR      byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
5618 00001AED E880050000           <1>       CALL   MED_CHANGE          ; CHECK MEDIA CHANGE AND RESET IF SO
5619 00001AF2 725D                 <1>         JC      short FM_DON             ; MEDIA CHANGED, SKIP
5620 00001AF4 E8DB020000           <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
5621 00001AF9 E8E6050000           <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
5622 00001AFE 7405                 <1>         JZ      short FM_WR              ; YES, SKIP SPECIFY COMMAND
5623 00001B00 E8BD050000           <1>       CALL   SEND_RATE           ; SEND DATA RATE TO CONTROLLER
5624                                <1> FM_WR:
5625 00001B05 E873060000           <1>       CALL   FMTDMA_SET          ; SET UP THE DMA FOR FORMAT
5626 00001B0A 7245                 <1>         JC      short FM_DON             ; RETURN WITH ERROR
5627 00001B0C B44D                 <1>       MOV    AH,04DH                  ; ESTABLISH THE FORMAT COMMAND
5628 00001B0E E8D0060000           <1>       CALL   NEC_INIT            ; INITIALIZE THE NEC
5629 00001B13 723C                 <1>         JC      short FM_DON             ; ERROR - EXIT
5630 00001B15 B8[511B0000]         <1>         MOV     eAX, FM_DON              ; LOAD ERROR ADDRESS
5631 00001B1A 50                   <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
5632 00001B1B B203                 <1>       MOV    DL,3                ; BYTES/SECTOR VALUE TO NEC
5633 00001B1D E83F090000           <1>       CALL   GET_PARM
5634 00001B22 E8400A0000           <1>       CALL   NEC_OUTPUT
5635 00001B27 B204                 <1>       MOV    DL,4                ; SECTORS/TRACK VALUE TO NEC
5636 00001B29 E833090000           <1>       CALL   GET_PARM
```

```
5637 00001B2E E8340A0000          <1>       CALL   NEC_OUTPUT
5638 00001B33 B207                <1>       MOV    DL,7                ; GAP LENGTH VALUE TO NEC
5639 00001B35 E827090000          <1>       CALL   GET_PARM
5640 00001B3A E8280A0000          <1>       CALL   NEC_OUTPUT
5641 00001B3F B208                <1>       MOV    DL,8                ; FILLER BYTE TO NEC
5642 00001B41 E81B090000          <1>       CALL   GET_PARM
5643 00001B46 E81C0A0000          <1>       CALL   NEC_OUTPUT
5644 00001B4B 58                  <1>       POP    eAX                 ; THROW AWAY ERROR
5645 00001B4C E810070000          <1>       CALL   NEC_TERM            ; TERMINATE, RECEIVE STATUS, ETC,
5646                              <1> FM_DON:
5647 00001B51 E8F8020000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
5648 00001B56 E832080000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
5649 00001B5B 6689F3              <1>       MOV    BX,SI               ; GET SAVED AL TO BL
5650 00001B5E 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
5651 00001B60 C3                  <1>       RETn
5652                              <1>
5653                              <1> ;-----------------------------------------------------------------------------
5654                              <1> ; FNC_ERR
5655                              <1> ;     INVALID FUNCTION REQUESTED OR INVALID DRIVE:
5656                              <1> ;     SET BAD COMMAND IN STATUS.
5657                              <1> ;
5658                              <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5659                              <1> ;-----------------------------------------------------------------------------
5660                              <1> FNC_ERR:                           ; INVALID FUNCTION REQUEST
5661 00001B61 6689F0              <1>       MOV    AX,SI               ; RESTORE AL
5662 00001B64 B401                <1>       MOV    AH,BAD_CMD          ; SET BAD COMMAND ERROR
5663 00001B66 8825[EC700000]      <1>       MOV    [DSKETTE_STATUS],AH ; STORE IN DATA AREA
5664 00001B6C F9                  <1>       STC                        ; SET CARRY INDICATING ERROR
5665 00001B6D C3                  <1>       RETn
5666                              <1>
5667                              <1> ;-----------------------------------------------------------------------------
5668                              <1> ; DISK_PARMS(AH = 08H)
5669                              <1> ;     READ DRIVE PARAMETERS.
5670                              <1> ;
5671                              <1> ; ON ENTRY:  DI : DRIVE #
5672                              <1> ;
5673                              <1> ; ON EXIT:   CL/[BP]   = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
5674                              <1> ;                        BITS 0-5 MAX SECTORS/TRACK
5675                              <1> ;            CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
5676                              <1> ;            BL/[BP+2] = BITS 7-4 = 0
5677                              <1> ;                        BITS 3-0 = VALID CMOS DRIVE TYPE
5678                              <1> ;            BH/[BP+3] = 0
5679                              <1> ;            DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
5680                              <1> ;            DH/[BP+5] = MAX HEAD #
5681                              <1> ;            DI/[BP+6] = OFFSET TO DISK_BASE
5682                              <1> ;            ES        = SEGMENT OF DISK_BASE
5683                              <1> ;            AX        = 0
5684                              <1> ;
5685                              <1> ;            NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
5686                              <1> ;                   THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
5687                              <1> ;                   INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
5688                              <1> ;                   CALLER.
5689                              <1> ;-----------------------------------------------------------------------------
5690                              <1> DSK_PARMS:
5691 00001B6E E8AA020000          <1>        CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH,
5692                              <1>      ;        MOV    WORD [BP+2],0       ; DRIVE TYPE = 0
5693 00001B73 29D2                <1>        sub     edx, edx ; 20/02/2015
5694 00001B75 895504              <1>          mov  [ebp+4], edx ; 20/02/2015
5695                              <1>      ;  MOV     AX, [EQUIP_FLAG]        ; LOAD EQUIPMENT FLAG FOR # DISKETTES
5696                              <1>      ;  AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
5697                              <1>      ;  MOV     DL,2                    ; DISKETTE DRIVES = 2
5698                              <1>      ;  CMP     AL,01000001B            ; 2 DRIVES INSTALLED ?
5699                              <1>      ;  JZ      short STO_DL            ; IF YES JUMP
5700                              <1>      ;  DEC     DL                      ; DISKETTE DRIVES = 1
5701                              <1>      ;  CMP     AL,00000001B            ; 1 DRIVE INSTALLED ?
5702                              <1>      ;  JNZ     short NON_DRV           ; IF NO JUMP
5703                              <1>        ;sub   edx, edx
5704 00001B78 66A1[1E6B0000]      <1>        mov     ax, [fd0_type]
5705 00001B7E 6621C0              <1>        and     ax, ax
5706 00001B81 7474                <1>        jz      short NON_DRV
5707 00001B83 FEC2                <1>        inc     dl
5708 00001B85 20E4                <1>        and     ah, ah
5709 00001B87 7402                <1>        jz      short STO_DL
5710 00001B89 FEC2                <1>        inc     dl
5711                              <1> STO_DL:
5712                              <1>        ;MOV   [BP+4],DL           ; STORE NUMBER OF DRIVES
5713 00001B8B 895508              <1>        mov    [ebp+8], edx ; 20/02/2015
5714 00001B8E 6683FF01            <1>        CMP    DI,1                ; CHECK FOR VALID DRIVE
5715 00001B92 7766                <1>        JA     short NON_DRV1              ; DRIVE INVALID
5716                              <1>        ;MOV   BYTE [BP+5],1       ; MAXIMUM HEAD NUMBER =   1
5717 00001B94 C6450901            <1>        mov    byte [ebp+9], 1  ; 20/02/2015
5718 00001B98 E8BB080000          <1>        CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL
5719                              <1>        ;;20/02/2015
5720                              <1>        ;;JC   short CHK_EST       ; IF CMOS BAD CHECKSUM ESTABLISHED
5721                              <1>        ;;OR   AL,AL               ; TEST FOR NO DRIVE TYPE
5722 00001B9D 7412                <1>        JZ     short CHK_EST       ; JUMP IF SO
5723 00001B9F E805020000          <1>        CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5724 00001BA4 720B                <1>        JC     short CHK_EST       ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
5725                              <1>        ;MOV   [BP+2],AL           ; STORE VALID CMOS DRIVE TYPE
5726 00001BA6 884504              <1>          mov  [ebp+4], al ; 06/02/2015
5727 00001BA9 8A4B04              <1>        MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
5728 00001BAC 8A6B0B              <1>          MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
5729 00001BAF EB36                <1>        JMP    SHORT STO_CX        ; CMOS GOOD, USE CMOS
5730                              <1> CHK_EST:
5731 00001BB1 8AA7[F9700000]      <1>        MOV    AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
5732 00001BB7 F6C410              <1>        TEST   AH,MED_DET          ; CHECK FOR ESTABLISHED STATE
5733 00001BBA 743E                <1>        JZ     short NON_DRV1              ; CMOS BAD/INVALID OR UNESTABLISHED
5734                              <1> USE_EST:
5735 00001BBC 80E4C0              <1>        AND    AH,RATE_MSK         ; ISOLATE STATE
5736 00001BBF 80FC80              <1>        CMP    AH,RATE_250         ; RATE 250 ?
5737 00001BC2 7557                <1>        JNE    short USE_EST2              ; NO, GO CHECK OTHER RATE
5738                              <1>
5739                              <1> ;-----       DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
5740                              <1>
5741 00001BC4 B001                <1>        MOV    AL,01               ; DRIVE TYPE 1 (360KB)
```

```
5742 00001BC6 E8DE010000          <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5743 00001BCB 8A4B04              <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
5744 00001BCE 8A6B0B              <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
5745 00001BD1 F687[F9700000]01    <1>       TEST   byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
5746 00001BD8 740D                <1>       JZ     short STO_CX        ; MUST BE 360KB DRIVE
5747                              <1>
5748                              <1> ;-----      IT IS 1.44 MB DRIVE
5749                              <1>
5750                              <1> PARM144:
5751 00001BDA B004                <1>       MOV    AL,04               ; DRIVE TYPE 4 (1.44MB)
5752 00001BDC E8C8010000          <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5753 00001BE1 8A4B04              <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
5754 00001BE4 8A6B0B              <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
5755                              <1> STO_CX:
5756 00001BE7 894D00              <1>       MOV    [eBP],eCX           ; SAVE POINTER IN STACK FOR RETURN
5757                              <1> ES_DI:
5758                              <1>       ;MOV   [BP+6],BX           ; ADDRESS OF MEDIA/DRIVE PARM TABLE
5759 00001BEA 895D0C              <1>       mov    [ebp+12], ebx ; 06/02/2015
5760                              <1>       ;MOV   AX,CS               ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
5761                              <1>       ;MOV   ES,AX               ; ES IS SEGMENT OF TABLE
5762                              <1> DP_OUT:
5763 00001BED E85C020000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
5764 00001BF2 6631C0              <1>       XOR    AX,AX               ; CLEAR
5765 00001BF5 F8                  <1>       CLC
5766 00001BF6 C3                  <1>       RETn
5767                              <1>
5768                              <1> ;-----      NO DRIYE PRESENT HANDLER
5769                              <1>
5770                              <1> NON_DRV:
5771                              <1>       ;MOV   BYTE [BP+4],0       ; CLEAR NUMBER OF DRIVES
5772 00001BF7 895508              <1>       mov    [ebp+8], edx ; 0 ; 20/02/2015
5773                              <1> NON_DRV1:
5774 00001BFA 6681FF8000          <1>       CMP    DI,80H              ; CHECK FOR FIXED MEDIA TYPE REQUEST
5775 00001BFF 720C                <1>       JB     short NON_DRV2            ; CONTINUE IF NOT REQUEST FALL THROUGH
5776                              <1>
5777                              <1> ;-----      FIXED DISK REQUEST FALL THROUGH ERROR
5778                              <1>
5779 00001C01 E848020000          <1>       CALL   XLAT_OLD            ; ELSE TRANSLATE TO COMPATIBLE MODE
5780 00001C06 6689F0              <1>       MOV    AX,SI               ; RESTORE AL
5781 00001C09 B401                <1>       MOV    AH,BAD_CMD          ; SET BAD COMMAND ERROR
5782 00001C0B F9                  <1>       STC
5783 00001C0C C3                  <1>       RETn
5784                              <1>
5785                              <1> NON_DRV2:
5786                              <1>       ;XOR   AX,AX               ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
5787 00001C0D 31C0                <1>       xor    eax, eax
5788 00001C0F 66894500            <1>       MOV    [eBP],AX            ; TRACKS, SECTORS/TRACK = 0
5789                              <1>       ;MOV   [BP+5],AH           ; HEAD = 0
5790 00001C13 886509              <1>       mov    [ebp+9], ah ; 06/02/2015
5791                              <1>       ;MOV   [BP+6],AX           ; OFFSET TO DISK_BASE = 0
5792 00001C16 89450C              <1>       mov    [ebp+12], eax
5793                              <1>       ;MOV   ES,AX               ; ES IS SEGMENT OF TABLE
5794 00001C19 EBD2                <1>       JMP    SHORT DP_OUT
5795                              <1>
5796                              <1> ;-----      DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
5797                              <1>
5798                              <1> USE_EST2:
5799 00001C1B B002                <1>       MOV    AL,02               ; DRIVE TYPE 2 (1.2MB)
5800 00001C1D E887010000          <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5801 00001C22 8A4B04              <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
5802 00001C25 8A6B0B              <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
5803 00001C28 80FC40              <1>       CMP    AH,RATE_300         ; RATE 300 ?
5804 00001C2B 74BA                <1>       JZ     short STO_CX        ; MUST BE 1.2MB DRIVE
5805 00001C2D EBAB                <1>       JMP    SHORT PARM144       ; ELSE, IT IS 1.44MB DRIVE
5806                              <1>
5807                              <1> ;--------------------------------------------------------------------------------
5808                              <1> ; DISK_TYPE (AH = 15H)
5809                              <1> ;     THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
5810                              <1> ;
5811                              <1> ;  ON ENTRY: DI = DRIVE #
5812                              <1> ;
5813                              <1> ;  ON EXIT:  AH = DRIVE TYPE, CY=0
5814                              <1> ;--------------------------------------------------------------------------------
5815                              <1> DSK_TYPE:
5816 00001C2F E8E9010000          <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
5817 00001C34 8A87[F9700000]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
5818 00001C3A 08C0                <1>       OR     AL,AL               ; CHECK FOR NO DRIVE
5819 00001C3C 7418                <1>       JZ     short NO_DRV
5820 00001C3E B401                <1>       MOV    AH,NOCHGLN          ; NO CHANGE LINE FOR 40 TRACK DRIVE
5821 00001C40 A801                <1>       TEST   AL,TRK_CAPA         ; IS THIS DRIVE AN 80 TRACK DRIVE?
5822 00001C42 7402                <1>       JZ     short DT_BACK            ; IF NO JUMP
5823 00001C44 B402                <1>       MOV    AH,CHGLN            ; CHANGE LINE FOR 80 TRACK DRIVE
5824                              <1> DT_BACK:
5825 00001C46 6650                <1>       PUSH   AX                  ; SAVE RETURN VALUE
5826 00001C48 E801020000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
5827 00001C4D 6658                <1>       POP    AX                  ; RESTORE RETURN VALUE
5828 00001C4F F8                  <1>       CLC                        ; NO ERROR
5829 00001C50 6689F3              <1>       MOV    BX,SI               ; GET SAVED AL TO BL
5830 00001C53 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
5831 00001C55 C3                  <1>       RETn
5832                              <1> NO_DRV:
5833 00001C56 30E4                <1>       XOR    AH,AH               ; NO DRIVE PRESENT OR UNKNOWN
5834 00001C58 EBEC                <1>       JMP    SHORT DT_BACK
5835                              <1>
5836                              <1> ;--------------------------------------------------------------------------------
5837                              <1> ; DISK_CHANGE      (AH = 16H)
5838                              <1> ;     THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
5839                              <1> ;
5840                              <1> ; ON ENTRY:  DI = DRIVE #
5841                              <1> ;
5842                              <1> ; ON EXIT:   AH = @DSKETTE_STATUS
5843                              <1> ;                 00 - DISK CHANGE LINE INACTIVE, CY = 0
5844                              <1> ;                 06 - DISK CHANGE LINE ACTIVE, CY = 1
5845                              <1> ;--------------------------------------------------------------------------------
5846                              <1> DSK_CHANGE:
```

```
5847 00001C5A E8BE010000          <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
5848 00001C5F 8A87[F9700000]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
5849 00001C65 08C0                <1>       OR     AL,AL               ; DRIVE PRESENT ?
5850 00001C67 7422                <1>       JZ     short DC_NON        ; JUMP IF NO DRIVE
5851 00001C69 A801                <1>       TEST   AL,TRK_CAPA         ; 80 TRACK DRIVE ?
5852 00001C6B 7407                <1>       JZ     short SETIT         ; IF SO , CHECK CHANGE LINE
5853                              <1> DC0:
5854 00001C6D E88D0A0000          <1>         CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
5855 00001C72 7407                <1>       JZ     short FINIS         ; CHANGE LINE NOT ACTIVE
5856                              <1>
5857 00001C74 C605[EC700000]06    <1> SETIT:      MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
5858                              <1>
5859 00001C7B E8CE010000          <1> FINIS:      CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
5860 00001C80 E808070000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
5861 00001C85 6689F3              <1>       MOV    BX,SI               ; GET SAVED AL TO BL
5862 00001C88 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
5863 00001C8A C3                  <1>       RETn
5864                              <1> DC_NON:
5865 00001C8B 800D[EC700000]80    <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
5866 00001C92 EBE7                <1>       JMP    SHORT FINIS
5867                              <1>
5868                              <1> ;--------------------------------------------------------------------------
5869                              <1> ; FORMAT_SET (AH = 17H)
5870                              <1> ;     THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
5871                              <1> ;     FOR THE FOLLOWING FORMAT OPERATION.
5872                              <1> ;
5873                              <1> ; ON ENTRY:  SI LOW = DASD TYPE FOR FORMAT
5874                              <1> ;            DI     = DRIVE #
5875                              <1> ;
5876                              <1> ; ON EXIT:   @DSKETTE_STATUS REFLECTS STATUS
5877                              <1> ;            AH = @DSKETTE_STATUS
5878                              <1> ;            CY = 1 IF ERROR
5879                              <1> ;--------------------------------------------------------------------------
5880                              <1> FORMAT_SET:
5881 00001C94 E884010000          <1>       CALL   XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
5882 00001C99 6656                <1>       PUSH   SI                 ; SAVE DASD TYPE
5883 00001C9B 6689F0              <1>       MOV    AX,SI              ; AH = ? , AL , DASD TYPE
5884 00001C9E 30E4                <1>       XOR    AH,AH              ; AH , 0 , AL , DASD TYPE
5885 00001CA0 6689C6              <1>       MOV    SI,AX              ; SI = DASD TYPE
5886 00001CA3 80A7[F9700000]0F    <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
5887 00001CAA 664E                <1>       DEC    SI                 ; CHECK FOR 320/360K MEDIA & DRIVE
5888 00001CAC 7509                <1>       JNZ    short NOT_320      ; BYPASS IF NOT
5889 00001CAE 808F[F9700000]90    <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
5890 00001CB5 EB48                <1>       JMP    SHORT S0
5891                              <1>
5892                              <1> NOT_320:
5893 00001CB7 E8B6030000          <1>       CALL   MED_CHANGE         ; CHECK FOR TIME_OUT
5894 00001CBC 803D[EC700000]80    <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT
5895 00001CC3 743A                <1>       JZ     short S0           ; IF TIME OUT TELL CALLER
5896                              <1> S3:
5897 00001CC5 664E                <1>       DEC    SI                 ; CHECK FOR 320/360K IN 1.2M DRIVE
5898 00001CC7 7509                <1>       JNZ    short NOT_320_12   ; BYPASS IF NOT
5899 00001CC9 808F[F9700000]70    <1>       OR     byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
5900 00001CD0 EB2D                <1>       JMP    SHORT S0
5901                              <1>
5902                              <1> NOT_320_12:
5903 00001CD2 664E                <1>       DEC    SI                 ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
5904 00001CD4 7509                <1>       JNZ    short NOT_12       ; BYPASS IF NOT
5905 00001CD6 808F[F9700000]10    <1>       OR     byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
5906 00001CDD EB20                <1>       JMP    SHORT S0           ; RETURN TO CALLER
5907                              <1>
5908                              <1> NOT_12:
5909 00001CDF 664E                <1>       DEC    SI                 ; CHECK FOR SET DASD TYPE 04
5910 00001CE1 752B                <1>       JNZ    short FS_ERR       ; BAD COMMAND EXIT IF NOT VALID TYPE
5911                              <1>
5912 00001CE3 F687[F9700000]04    <1>       TEST   byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
5913 00001CEA 740B                <1>       JZ     short ASSUME       ; IF STILL NOT DETERMINED ASSUME
5914 00001CEC B050                <1>       MOV    AL,MED_DET+RATE_300
5915 00001CEE F687[F9700000]02    <1>         TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
5916 00001CF5 7502                <1>       JNZ    short OR_IT_IN            ; IF 1.2 M THEN DATA RATE 300
5917                              <1>
5918                              <1> ASSUME:
5919 00001CF7 B090                <1>       MOV    AL,MED_DET+RATE_250 ; SET UP
5920                              <1>
5921                              <1> OR_IT_IN:
5922 00001CF9 0887[F9700000]      <1>       OR     [DSK_STATE+eDI], AL ; OR IN THE CORRECT STATE
5923                              <1> S0:
5924 00001CFF E84A010000          <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
5925 00001D04 E884060000          <1>       CALL   SETUP_END          ; VARIOUS CLEANUPS
5926 00001D09 665B                <1>       POP    BX                 ; GET SAVED AL TO BL
5927 00001D0B 88D8                <1>       MOV    AL,BL              ; PUT BACK FOR RETURN
5928 00001D0D C3                  <1>       RETn
5929                              <1>
5930                              <1> FS_ERR:
5931 00001D0E C605[EC700000]01    <1>       MOV    byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
5932 00001D15 EBE8                <1>       JMP    SHORT S0
5933                              <1>
5934                              <1> ;--------------------------------------------------------------------------
5935                              <1> ; SET_MEDIA  (AH = 18H)
5936                              <1> ;     THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
5937                              <1> ;     TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
5938                              <1> ;
5939                              <1> ; ON ENTRY:
5940                              <1> ;     [BP]   = SECTOR PER TRACK
5941                              <1> ;     [BP+1] = TRACK #
5942                              <1> ;     DI     = DRIVE #
5943                              <1> ;
5944                              <1> ; ON EXIT:
5945                              <1> ;     @DSKETTE_STATUS REFLECTS STATUS
5946                              <1> ;     IF NO ERROR:
5947                              <1> ;           AH = 0
5948                              <1> ;           CY = 0
5949                              <1> ;           ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
5950                              <1> ;           DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
5951                              <1> ;     IF ERROR:
```

```
5952                                    <1> ;           AH = @DSKETTE_STATUS
5953                                    <1> ;           CY = 1
5954                                    <1> ;--------------------------------------------------------------------------
5955                                    <1> SET_MEDIA:
5956 00001D17 E801010000                <1>         CALL   XLAT_NEW             ; TRANSLATE STATE TO PRESENT ARCH.
5957 00001D1C F687[F9700000]01          <1>           TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
5958 00001D23 7415                      <1>         JZ     short SM_CMOS        ; JUMP IF 40 TRACK DRIVE
5959 00001D25 E848030000                <1>         CALL   MED_CHANGE           ; RESET CHANGE LINE
5960 00001D2A 803D[EC700000]80          <1>         CMP    byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
5961 00001D31 746B                      <1>         JE     short SM_RTN
5962 00001D33 C605[EC700000]00          <1>         MOV    byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
5963                                    <1> SM_CMOS:
5964 00001D3A E819070000                <1>         CALL   CMOS_TYPE            ; RETURN DRIVE TYPE IN (AL)
5965                                    <1>         ;;20/02/2015
5966                                    <1>         ;;JC   short MD_NOT_FND     ; ERROR IN CMOS
5967                                    <1>         ;;OR   AL,AL                ; TEST FOR NO DRIVE
5968 00001D3F 745D                      <1>         JZ     short SM_RTN         ; RETURN IF SO
5969 00001D41 E863000000                <1>         CALL   DR_TYPE_CHECK        ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5970 00001D46 7231                      <1>         JC     short MD_NOT_FND     ; TYPE NOT IN TABLE (BAD CMOS)
5971 00001D48 57                        <1>         PUSH   eDI                  ; SAVE REG.
5972 00001D49 31DB                      <1>         XOR    eBX,eBX                   ; BX = INDEX TO DR. TYPE TABLE
5973 00001D4B B906000000                <1>         MOV    eCX,DR_CNT           ; CX = LOOP COUNT
5974                                    <1> DR_SEARCH:
5975 00001D50 8AA3[9C6A0000]            <1>         MOV    AH, [DR_TYPE+eBX]    ; GET DRIVE TYPE
5976 00001D56 80E47F                    <1>         AND    AH,BIT7OFF           ; MASK OUT MSB
5977 00001D59 38E0                      <1>         CMP    AL,AH                ; DRIVE TYPE MATCH ?
5978 00001D5B 7516                      <1>         JNE    short NXT_MD         ; NO, CHECK NEXT DRIVE TYPE
5979                                    <1> DR_FND:
5980 00001D5D 8BBB[9D6A0000]            <1>         MOV    eDI, [DR_TYPE+eBX+1]       ; DI = MEDIA/DRIVE PARAM TABLE
5981                                    <1> MD_SEARCH:
5982 00001D63 8A6704                    <1>           MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
5983 00001D66 386500                    <1>         CMP    [eBP],AH             ; MATCH?
5984 00001D69 7508                      <1>         JNE    short NXT_MD         ; NO, CHECK NEXT MEDIA
5985 00001D6B 8A670B                    <1>           MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
5986 00001D6E 386501                    <1>         CMP    [eBP+1],AH           ; MATCH?
5987 00001D71 740F                      <1>         JE     short MD_FND         ; YES, GO GET RATE
5988                                    <1> NXT_MD:
5989                                    <1>         ;ADD   BX,3                 ; CHECK NEXT DRIVE TYPE
5990 00001D73 83C305                    <1>           add  ebx, 5 ; 18/02/2015
5991 00001D76 E2D8                      <1>         LOOP    DR_SEARCH
5992 00001D78 5F                        <1>         POP    eDI                  ; RESTORE REG.
5993                                    <1> MD_NOT_FND:
5994 00001D79 C605[EC700000]0C          <1>         MOV    byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
5995 00001D80 EB1C                      <1>         JMP    SHORT SM_RTN         ; RETURN
5996                                    <1> MD_FND:
5997 00001D82 8A470C                    <1>           MOV     AL, [eDI+MD.RATE]       ; GET RATE
5998 00001D85 3C40                      <1>         CMP    AL,RATE_300          ; DOUBLE STEP REQUIRED FOR RATE 300
5999 00001D87 7502                      <1>         JNE    short MD_SET
6000 00001D89 0C20                      <1>         OR     AL,DBL_STEP
6001                                    <1> MD_SET:
6002                                    <1>         ;MOV   [BP+6],DI            ; SAVE TABLE POINTER IN STACK
6003 00001D8B 897D0C                    <1>         mov    [ebp+12], edi ; 18/02/2015
6004 00001D8E 0C10                      <1>         OR     AL,MED_DET           ; SET MEDIA ESTABLISHED
6005 00001D90 5F                        <1>         POP    eDI
6006 00001D91 80A7[F9700000]0F          <1>         AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
6007 00001D98 0887[F9700000]            <1>         OR     [DSK_STATE+eDI], AL
6008                                    <1>         ;MOV   AX, CS               ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
6009                                    <1>         ;MOV   ES, AX               ; ES IS SEGMENT OF TABLE
6010                                    <1> SM_RTN:
6011 00001D9E E8AB000000                <1>         CALL   XLAT_OLD             ; TRANSLATE STATE TO COMPATIBLE MODE
6012 00001DA3 E8E5050000                <1>         CALL   SETUP_END            ; VARIOUS CLEANUPS
6013 00001DA8 C3                        <1>         RETn
6014                                    <1>
6015                                    <1> ;-----------------------------------------------------------
6016                                    <1> ; DR_TYPE_CHECK                                         :
6017                                    <1> ;     CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)          :
6018                                    <1> ;     IS SUPPORTED IN BIOS DRIVE TYPE TABLE                 :
6019                                    <1> ; ON ENTRY:                                          :
6020                                    <1> ;     AL = DRIVE TYPE                                       :
6021                                    <1> ; ON EXIT:                                           :
6022                                    <1> ;     CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)           :
6023                                    <1> ;     CY = 0       DRIVE TYPE SUPPORTED                         :
6024                                    <1> ;          BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE           :
6025                                    <1> ;     CY = 1 DRIVE TYPE NOT SUPPORTED               :
6026                                    <1> ; REGISTERS ALTERED: eBX                                :
6027                                    <1> ;-----------------------------------------------------------
6028                                    <1> DR_TYPE_CHECK:
6029 00001DA9 6650                      <1>         PUSH   AX
6030 00001DAB 51                        <1>         PUSH   eCX
6031 00001DAC 31DB                      <1>         XOR    eBX,eBX                   ; BX = INDEX TO DR_TYPE TABLE
6032 00001DAE B906000000                <1>         MOV    eCX,DR_CNT           ; CX = LOOP COUNT
6033                                    <1> TYPE_CHK:
6034 00001DB3 8AA3[9C6A0000]            <1>         MOV    AH,[DR_TYPE+eBX]     ; GET DRIVE TYPE
6035 00001DB9 38E0                      <1>         CMP    AL,AH                ; DRIVE TYPE MATCH?
6036 00001DBB 740D                      <1>         JE     short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
6037                                    <1>         ;ADD   BX,3                 ; CHECK NEXT DRIVE TYPE
6038 00001DBD 83C305                    <1>           add  ebx, 5 ; 16/02/2015 (32 bit address modification)
6039 00001DC0 E2F1                      <1>         LOOP    TYPE_CHK
6040                                    <1>         ;
6041 00001DC2 BB[FB6A0000]              <1>         mov    ebx, MD_TBL6         ; 1.44MB fd parameter table
6042                                    <1>                                     ; Default for GET_PARM (11/12/2014)
6043                                    <1>         ;
6044 00001DC7 F9                        <1>         STC                         ; DRIVE TYPE NOT FOUND IN TABLE
6045 00001DC8 EB06                      <1>         JMP    SHORT TYPE_RTN
6046                                    <1> DR_TYPE_VALID:
6047 00001DCA 8B9B[9D6A0000]            <1>         MOV    eBX,[DR_TYPE+eBX+1]       ; BX = MEDIA TABLE
6048                                    <1> TYPE_RTN:
6049 00001DD0 59                        <1>         POP    eCX
6050 00001DD1 6658                      <1>         POP    AX
6051 00001DD3 C3                        <1>         RETn
6052                                    <1>
6053                                    <1> ;-----------------------------------------------------------
6054                                    <1> ; SEND_SPEC                                             :
6055                                    <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
6056                                    <1> ;     THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER   :
```

```
6057                            <1> ; ON ENTRY:  @DISK_POINTER = DRIVE PARAMETER TABLE         :
6058                            <1> ; ON EXIT:   NONE                                          :
6059                            <1> ; REGISTERS ALTERED: CX, DX                                :
6060                            <1> ;-----------------------------------------------------------------
6061                            <1> SEND_SPEC:
6062 00001DD4 50                  <1>       PUSH   eAX                ; SAVE AX
6063 00001DD5 B8[FB1D0000]        <1>       MOV    eAX, SPECBAC       ; LOAD ERROR ADDRESS
6064 00001DDA 50                  <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
6065 00001DDB B403                <1>       MOV    AH,03H             ; SPECIFY COMMAND
6066 00001DDD E885070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6067 00001DE2 28D2                <1>       SUB    DL,DL              ; FIRST SPECIFY BYTE
6068 00001DE4 E878060000          <1>       CALL   GET_PARM           ; GET PARAMETER TO AH
6069 00001DE9 E879070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6070 00001DEE B201                <1>       MOV    DL,1               ; SECOND SPECIFY BYTE
6071 00001DF0 E86C060000          <1>       CALL   GET_PARM           ; GET PARAMETER TO AH
6072 00001DF5 E86D070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6073 00001DFA 58                  <1>       POP    eAX                ; POP ERROR RETURN
6074                            <1> SPECBAC:
6075 00001DFB 58                  <1>       POP    eAX                ; RESTORE ORIGINAL AX VALUE
6076 00001DFC C3                  <1>       RETn
6077                            <1>
6078                            <1> ;-----------------------------------------------------------------
6079                            <1> ; SEND_SPEC_MD                                             :
6080                            <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
6081                            <1> ;     THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)   :
6082                            <1> ; ON ENTRY:  CS:BX = MEDIA/DRIVE PARAMETER TABLE           :
6083                            <1> ; ON EXIT:   NONE                                          :
6084                            <1> ; REGISTERS ALTERED: AX                                    :
6085                            <1> ;-----------------------------------------------------------------
6086                            <1> SEND_SPEC_MD:
6087 00001DFD 50                  <1>       PUSH   eAX                ; SAVE RATE DATA
6088 00001DFE B8[1B1E0000]        <1>       MOV    eAX, SPEC_ESBAC          ; LOAD ERROR ADDRESS
6089 00001E03 50                  <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
6090 00001E04 B403                <1>       MOV    AH,03H             ; SPECIFY COMMAND
6091 00001E06 E85C070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6092 00001E0B 8A23                <1>         MOV     AH, [eBX+MD.SPEC1]      ; GET 1ST SPECIFY BYTE
6093 00001E0D E855070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6094 00001E12 8A6301              <1>         MOV     AH, [eBX+MD.SPEC2]      ; GET SECOND SPECIFY BYTE
6095 00001E15 E84D070000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE COMMAND
6096 00001E1A 58                  <1>       POP    eAX                ; POP ERROR RETURN
6097                            <1> SPEC_ESBAC:
6098 00001E1B 58                  <1>       POP    eAX                ; RESTORE ORIGINAL AX VALUE
6099 00001E1C C3                  <1>       RETn
6100                            <1>
6101                            <1> ;-----------------------------------------------------------------------------
6102                            <1> ; XLAT_NEW
6103                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
6104                            <1> ;     MODE TO NEW ARCHITECTURE.
6105                            <1> ;
6106                            <1> ; ON ENTRY:  DI = DRIVE #
6107                            <1> ;-----------------------------------------------------------------------------
6108                            <1> XLAT_NEW:
6109 00001E1D 83FF01              <1>       CMP    eDI,1                   ; VALID DRIVE
6110 00001E20 7725                <1>       JA     short XN_OUT            ; IF INVALID BACK
6111 00001E22 80BF[F9700000]00    <1>       CMP    byte [DSK_STATE+eDI], 0         ; NO DRIVE ?
6112 00001E29 741D                <1>       JZ     short DO_DET            ; IF NO DRIVE ATTEMPT DETERMINE
6113 00001E2B 6689F9              <1>       MOV    CX,DI                   ; CX = DRIVE NUMBER
6114 00001E2E C0E102              <1>       SHL    CL,2                    ; CL = SHIFT COUNT, A=0, B=4
6115 00001E31 A0[F8700000]        <1>       MOV    AL, [HF_CNTRL]                ; DRIVE INFORMATION
6116 00001E36 D2C8                <1>       ROR    AL,CL                   ; TO LOW NIBBLE
6117 00001E38 2407                <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA     ; KEEP DRIVE BITS
6118 00001E3A 80A7[F9700000]F8    <1>         AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
6119 00001E41 0887[F9700000]      <1>       OR     [DSK_STATE+eDI], AL     ; UPDATE DRIVE STATE
6120                            <1> XN_OUT:
6121 00001E47 C3                  <1>       RETn
6122                            <1> DO_DET:
6123 00001E48 E8BF080000          <1>       CALL   DRIVE_DET               ; TRY TO DETERMINE
6124 00001E4D C3                  <1>       RETn
6125                            <1>
6126                            <1> ;-----------------------------------------------------------------------------
6127                            <1> ; XLAT_OLD
6128                            <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
6129                            <1> ;     ARCHITECTURE TO COMPATIBLE MODE.
6130                            <1> ;
6131                            <1> ; ON ENTRY:  DI = DRIVE
6132                            <1> ;-----------------------------------------------------------------------------
6133                            <1> XLAT_OLD:
6134 00001E4E 83FF01              <1>       CMP    eDI,1              ; VALID DRIVE ?
6135                            <1>         ;JA     short XO_OUT            ; IF INVALID BACK
6136 00001E51 0F8786000000        <1>         ja      XO_OUT
6137 00001E57 80BF[F9700000]00    <1>         CMP  byte [DSK_STATE+eDI],0    ; NO DRIVE ?
6138 00001E5E 747D                <1>       JZ     short XO_OUT       ; IF NO DRIVE TRANSLATE DONE
6139                            <1>
6140                            <1> ;-----      TEST FOR SAVED DRIVE INFORMATION ALREADY SET
6141                            <1>
6142 00001E60 6689F9              <1>       MOV    CX,DI              ; CX = DRIVE NUMBER
6143 00001E63 C0E102              <1>       SHL    CL,2               ; CL = SHIFT COUNT, A=0, B=4
6144 00001E66 B402                <1>       MOV    AH,FMT_CAPA        ; LOAD MULTIPLE DATA RATE BIT MASK
6145 00001E68 D2CC                <1>       ROR    AH,CL              ; ROTATE BY MASK
6146 00001E6A 8425[F8700000]      <1>       TEST   [HF_CNTRL], AH           ; MULTIPLE-DATA RATE DETERMINED ?
6147 00001E70 751C                <1>       JNZ    short SAVE_SET           ; IF SO, NO NEED TO RE-SAVE
6148                            <1>
6149                            <1> ;-----      ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
6150                            <1>
6151 00001E72 B407                <1>       MOV    AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
6152 00001E74 D2CC                <1>       ROR    AH,CL              ; FIX MASK TO KEEP
6153 00001E76 F6D4                <1>       NOT    AH                 ; TRANSLATE MASK
6154 00001E78 2025[F8700000]      <1>       AND    [HF_CNTRL], AH           ; KEEP BITS FROM OTHER DRIVE INTACT
6155                            <1>
6156                            <1> ;-----      ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
6157                            <1>
6158 00001E7E 8A87[F9700000]      <1>       MOV    AL, [DSK_STATE+eDI] ; ACCESS STATE
6159 00001E84 2407                <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
6160 00001E86 D2C8                <1>       ROR    AL,CL              ; FIX FOR THIS DRIVE
6161 00001E88 0805[F8700000]      <1>       OR     [HF_CNTRL], AL           ; UPDATE SAVED DRIVE STATE
```

```
6162                            <1>
6163                            <1> ;-----      TRANSLATE TO COMPATIBILITY MODE
6164                            <1>
6165                            <1> SAVE_SET:
6166 00001E8E 8AA7[F9700000]    <1>       MOV   AH, [DSK_STATE+eDI] ; ACCESS STATE
6167 00001E94 88E7              <1>       MOV   BH,AH               ; TO BH FOR LATER
6168 00001E96 80E4C0            <1>       AND   AH,RATE_MSK         ; KEEP ONLY RATE
6169 00001E99 80FC00            <1>       CMP   AH,RATE_500         ; RATE 500 ?
6170 00001E9C 7410              <1>       JZ    short CHK_144       ; YES 1.2/1.2 OR 1.44/1.44
6171 00001E9E B001              <1>       MOV   AL,M3D1U            ; AL = 360 IN 1.2 UNESTABLISHED
6172 00001EA0 80FC40            <1>       CMP   AH,RATE_300         ; RATE 300 ?
6173 00001EA3 7518              <1>       JNZ   short CHK_250       ; NO, 360/360, 720/720 OR 720/1.44
6174 00001EA5 F6C720            <1>       TEST  BH,DBL_STEP         ; CHECK FOR DOUBLE STEP
6175 00001EA8 751F              <1>       JNZ   short TST_DET       ; MUST BE 360 IN 1.2
6176                            <1> UNKNO:
6177 00001EAA B007              <1>       MOV   AL,MED_UNK          ; NONE OF THE ABOVE
6178 00001EAC EB22              <1>       JMP   SHORT AL_SET        ; PROCESS COMPLETE
6179                            <1> CHK_144:
6180 00001EAE E8A5050000        <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
6181                            <1>       ;;20/02/2015
6182                            <1>       ;;JC  short UNKNO         ; ERROR, SET 'NONE OF ABOVE'
6183 00001EB3 74F5              <1>       jz    short UNKNO ;; 20/02/2015
6184 00001EB5 3C02              <1>       CMP   AL,2                ; 1.2MB DRIVE ?
6185 00001EB7 75F1              <1>       JNE   short UNKNO         ; NO, GO SET 'NONE OF ABOVE'
6186 00001EB9 B002              <1>       MOV   AL,M1D1U            ; AL = 1.2 IN 1.2 UNESTABLISHED
6187 00001EBB EB0C              <1>       JMP   SHORT TST_DET
6188                            <1> CHK_250:
6189 00001EBD B000              <1>       MOV   AL,M3D3U            ; AL = 360 IN 360 UNESTABLISHED
6190 00001EBF 80FC80            <1>       CMP   AH,RATE_250         ; RATE 250 ?
6191 00001EC2 75E6              <1>       JNZ   short UNKNO         ; IF SO FALL IHRU
6192 00001EC4 F6C701            <1>       TEST  BH,TRK_CAPA         ; 80 TRACK CAPABILITY ?
6193 00001EC7 75E1              <1>       JNZ   short UNKNO         ; IF SO JUMP, FALL THRU TEST DET
6194                            <1> TST_DET:
6195 00001EC9 F6C710            <1>       TEST  BH,MED_DET          ; DETERMINED ?
6196 00001ECC 7402              <1>       JZ    short AL_SET        ; IF NOT THEN SET
6197 00001ECE 0403              <1>       ADD   AL,3                ; MAKE DETERMINED/ESTABLISHED
6198                            <1> AL_SET:
6199 00001ED0 80A7[F9700000]F8  <1>       AND   byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
6200 00001ED7 0887[F9700000]    <1>       OR    [DSK_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE
6201                            <1> XO_OUT:
6202 00001EDD C3                <1>       RETn
6203                            <1>
6204                            <1> ;---------------------------------------------------------------------------
6205                            <1> ; RD_WR_VF
6206                            <1> ;     COMMON READ, WRITE AND VERIFY:
6207                            <1> ;     MAIN LOOP FOR STATE RETRIES.
6208                            <1> ;
6209                            <1> ; ON ENTRY:  AH = READ/WRITE/VERIFY NEC PARAMETER
6210                            <1> ;            AL = READ/WRITE/VERIFY DMA PARAMETER
6211                            <1> ;
6212                            <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6213                            <1> ;---------------------------------------------------------------------------
6214                            <1> RD_WR_VF:
6215 00001EDE 6650              <1>       PUSH  AX                  ; SAVE DMA, NEC PARAMETERS
6216 00001EE0 E838FFFFFF        <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
6217 00001EE5 E8F3000000        <1>       CALL  SETUP_STATE         ; INITIALIZE START AND END RATE
6218 00001EEA 6658              <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
6219                            <1> DO_AGAIN:
6220 00001EEC 6650              <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
6221 00001EEE E87F010000        <1>       CALL  MED_CHANGE          ; MEDIA CHANGE AND RESET IF CHANGED
6222 00001EF3 6658              <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
6223 00001EF5 0F82C9000000      <1>         JC    RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
6224                            <1> RWV:
6225 00001EFB 6650              <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
6226 00001EFD 8AB7[F9700000]    <1>       MOV   DH, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
6227 00001F03 80E6C0            <1>       AND   DH,RATE_MSK         ; KEEP ONLY RATE
6228 00001F06 E84D050000        <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN AL (AL)
6229                            <1>       ;;20/02/2015
6230                            <1>       ;;JC  short RWV_ASSUME    ; ERROR IN CMOS
6231 00001F0B 7451              <1>       jz    short RWV_ASSUME ; 20/02/2015
6232 00001F0D 3C01              <1>       CMP   AL,1                ; 40 TRACK DRIVE?
6233 00001F0F 750D              <1>       JNE   short RWV_1         ; NO, BYPASS CMOS VALIDITY CHECK
6234 00001F11 F687[F9700000]01  <1>       TEST  byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
6235 00001F18 7413              <1>       JZ    short RWV_2         ; YES, CMOS IS CORRECT
6236 00001F1A B002              <1>       MOV   AL,2                ; CHANGE TO 1.2M
6237 00001F1C EB0F              <1>       JMP   SHORT RWV_2
6238                            <1> RWV_1:
6239 00001F1E 720D              <1>       JB    short RWV_2         ; NO DRIVE SPECIFIED, CONTINUE
6240 00001F20 F687[F9700000]01  <1>       TEST    byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
6241 00001F27 7504              <1>       JNZ   short RWV_2         ; NO, 80 TRACK
6242 00001F29 B001              <1>       MOV   AL,1                ; IT IS 40 TRACK, FIX CMOS VALUE
6243 00001F2B EB04              <1>       jmp   short rwv_3
6244                            <1> RWV_2:
6245 00001F2D 08C0              <1>       OR    AL,AL               ; TEST FOR NO DRIVE
6246 00001F2F 742D              <1>       JZ    short RWV_ASSUME    ; ASSUME TYPE, USE MAX TRACK
6247                            <1> rwv_3:
6248 00001F31 E873FEFFFF        <1>       CALL  DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
6249 00001F36 7226              <1>       JC    short RWV_ASSUME    ; TYPE NOT IN TABLE (BAD CMOS)
6250                            <1>
6251                            <1> ;-----      SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
6252                            <1>
6253 00001F38 57                <1>       PUSH  eDI                 ; SAVE DRIVE #
6254 00001F39 31DB              <1>       XOR   eBX,eBX                   ; BX = INDEX TO DR_TYPE TABLE
6255 00001F3B B906000000        <1>       MOV   eCX,DR_CNT          ; CX = LOOP COUNT
6256                            <1> RWV_DR_SEARCH:
6257 00001F40 8AA3[9C6A0000]    <1>       MOV   AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
6258 00001F46 80E47F            <1>       AND   AH,BIT7OFF          ; MASK OUT MSB
6259 00001F49 38E0              <1>       CMP   AL,AH               ; DRIVE TYPE MATCH?
6260 00001F4B 750B              <1>       JNE   short RWV_NXT_MD    ; NO, CHECK NEXT DRIVE TYPE
6261                            <1> RWV_DR_FND:
6262 00001F4D 8BBB[9D6A0000]    <1>       MOV   eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAMETER TABLE
6263                            <1> RWV_MD_SEARH:
6264 00001F53 3A770C            <1>         CMP     DH, [eDI+MD.RATE]       ; MATCH?
6265 00001F56 741B              <1>       JE    short RWV_MD_FND    ; YES, GO GET 1ST SPECIFY BYTE
6266                            <1> RWV_NXT_MD:
```

```
6267                                <1>       ;ADD   BX,3               ; CHECK NEXT DRIVE TYPE
6268 00001F58 83C305               <1>       add    eBX, 5
6269 00001F5B E2E3                 <1>       LOOP   RWV_DR_SEARCH
6270 00001F5D 5F                   <1>       POP    eDI                ; RESTORE DRIVE #
6271                                <1>
6272                                <1> ;-----      ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
6273                                <1>
6274                                <1> RWV_ASSUME:
6275 00001F5E BB[BA6A0000]         <1>       MOV    eBX, MD_TBL1        ; POINT TO 40 TRACK 250 KBS
6276 00001F63 F687[F9700000]01     <1>       TEST   byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
6277 00001F6A 740A                 <1>       JZ     short RWV_MD_FND1   ; MUST BE 40 TRACK
6278 00001F6C BB[D46A0000]         <1>       MOV    eBX, MD_TBL3        ; POINT TO 80 TRACK 500 KBS
6279 00001F71 EB03                 <1>       JMP    short RWV_MD_FND1   ; GO SPECIFY PARAMTERS
6280                                <1>
6281                                <1> ;-----      CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
6282                                <1>
6283                                <1> RWV_MD_FND:
6284 00001F73 89FB                 <1>       MOV    eBX,eDI                  ; BX = MEDIA/DRIVE PARAMETER TABLE
6285 00001F75 5F                   <1>       POP    eDI                ; RESTORE DRIVE #
6286                                <1>
6287                                <1> ;-----      SEND THE SPECIFY COMMAND TO THE CONTROLLER
6288                                <1>
6289                                <1> RWV_MD_FND1:
6290 00001F76 E882FEFFFF           <1>       CALL   SEND_SPEC_MD
6291 00001F7B E864010000           <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
6292 00001F80 7405                 <1>       JZ     short RWV_DBL       ; YES,SKIP SEND RATE COMMAND
6293 00001F82 E83B010000           <1>       CALL   SEND_RATE           ; SEND DATA RATE TO NEC
6294                                <1> RWV_DBL:
6295 00001F87 53                   <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6296 00001F88 E822040000           <1>       CALL   SETUP_DBL           ; CHECK FOR DOUBLE STEP
6297 00001F8D 5B                   <1>       POP    eBX                 ; RESTORE ADDRESS
6298 00001F8E 7226                 <1>       JC     short CHK_RET       ; ERROR FROM READ ID, POSSIBLE RETRY
6299 00001F90 6658                 <1>       POP    AX                  ; RESTORE NEC, DMA COMMAND
6300 00001F92 6650                 <1>       PUSH   AX                  ; SAVE NEC COMMAND
6301 00001F94 53                   <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6302 00001F95 E861010000           <1>       CALL   DMA_SETUP           ; SET UP THE DMA
6303 00001F9A 5B                   <1>       POP    eBX
6304 00001F9B 6658                 <1>       POP    AX                  ; RESTORE NEC COMMAND
6305 00001F9D 722F                 <1>       JC     short RWV_BAC       ; CHECK FOR DMA BOUNDARY ERROR
6306 00001F9F 6650                 <1>       PUSH   AX                  ; SAVE NEC COMMAND
6307 00001FA1 53                   <1>       PUSH   eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6308 00001FA2 E83C020000           <1>       CALL   NEC_INIT            ; INITIALIZE NEC
6309 00001FA7 5B                   <1>       POP    eBX                 ; RESTORE ADDRESS
6310 00001FA8 720C                 <1>       JC     short CHK_RET       ; ERROR - EXIT
6311 00001FAA E866020000           <1>       CALL   RWV_COM                  ; OP CODE COMMON TO READ/WRITE/VERIFY
6312 00001FAF 7205                 <1>       JC     short CHK_RET       ; ERROR - EXIT
6313 00001FB1 E8AB020000           <1>       CALL   NEC_TERM            ; TERMINATE, GET STATUS, ETC.
6314                                <1> CHK_RET:
6315 00001FB6 E84A030000           <1>       CALL   RETRY               ; CHECK FOR, SETUP RETRY
6316 00001FBB 6658                 <1>       POP    AX                  ; RESTORE READ/WRITE/VERIFY PARAMETER
6317 00001FBD 7305                 <1>       JNC    short RWV_END       ; CY = 0 NO RETRY
6318 00001FBF E928FFFFFF           <1>         JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
6319                                <1> RWV_END:
6320 00001FC4 E8F4020000           <1>       CALL   DSTATE              ; ESTABLISH STATE IF SUCCESSFUL
6321 00001FC9 E887030000           <1>       CALL   NUM_TRANS           ; AL = NUMBER TRANSFERRED
6322                                <1> RWV_BAC:                       ; BAD DMA ERROR ENTRY
6323 00001FCE 6650                 <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
6324 00001FD0 E879FEFFFF           <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
6325 00001FD5 6658                 <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
6326 00001FD7 E8B1030000           <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
6327 00001FDC C3                   <1>       RETn
6328                                <1>
6329                                <1> ;--------------------------------------------------------------------------------
6330                                <1> ; SETUP_STATE:     INITIALIZES START AND END RATES.
6331                                <1> ;--------------------------------------------------------------------------------
6332                                <1> SETUP_STATE:
6333 00001FDD F687[F9700000]10     <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
6334 00001FE4 7537                 <1>       JNZ    short J1C           ; NO STATES IF DETERMINED
6335 00001FE6 66B84000             <1>         MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
6336 00001FEA F687[F9700000]04     <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
6337 00001FF1 740D                 <1>       JZ     short AX_SET        ; DO NOT KNOW DRIVE
6338 00001FF3 F687[F9700000]02     <1>       TEST   byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
6339 00001FFA 7504                 <1>       JNZ    short AX_SET        ; JUMP IF YES
6340 00001FFC 66B88080             <1>         MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
6341                                <1> AX_SET:
6342 00002000 80A7[F9700000]1F     <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
6343 00002007 08A7[F9700000]       <1>       OR     [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
6344 0000200D 8025[F4700000]F3     <1>       AND    byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
6345 00002014 C0C804               <1>       ROR    AL,4                ; TO OPERATION LAST RATE LOCATION
6346 00002017 0805[F4700000]       <1>       OR     [LASTRATE], AL             ; LAST RATE
6347                                <1> J1C:
6348 0000201D C3                   <1>       RETn
6349                                <1>
6350                                <1> ;--------------------------------------------------------------------------------
6351                                <1> ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
6352                                <1> ;--------------------------------------------------------------------------------
6353                                <1> FMT_INIT:
6354 0000201E F687[F9700000]10     <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
6355 00002025 7546                 <1>       JNZ    short F1_OUT        ; IF SO RETURN
6356 00002027 E82C040000           <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL
6357                                <1>       ;; 20/02/2015
6358                                <1>       ;;JC   short CL_DRV        ; ERROR IN CMOS ASSUME NO DRIVE
6359 0000202C 7440                 <1>       jz     short CL_DRV ;; 20/02/2015
6360 0000202E FEC8                 <1>       DEC    AL                  ; MAKE ZERO ORIGIN
6361                                <1>       ;;JS   short CL_DRV        ; NO DRIVE IF AL 0
6362 00002030 8AA7[F9700000]       <1>       MOV    AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
6363 00002036 80E40F               <1>       AND    AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
6364 00002039 08C0                 <1>       OR     AL,AL               ; CHECK FOR 360
6365 0000203B 7505                 <1>       JNZ    short N_360         ; IF 360 WILL BE 0
6366 0000203D 80CC90               <1>       OR     AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
6367 00002040 EB25                 <1>       JMP    SHORT SKP_STATE            ; SKIP OTHER STATE PROCESSING
6368                                <1> N_360:
6369 00002042 FEC8                 <1>       DEC    AL                  ; 1.2 M DRIVE
6370 00002044 7505                 <1>       JNZ    short N_12          ; JUMP IF NOT
6371                                <1> F1_RATE:
```

```
6372 00002046 80CC10              <1>       OR    AH,MED_DET+RATE_500 ; SET FORMAT RATE
6373 00002049 EB1C                <1>       JMP   SHORT SKP_STATE           ; SKIP OTHER STATE PROCESSING
6374                              <1> N_12:
6375 0000204B FEC8                <1>       DEC   AL                  ; CHECK FOR TYPE 3
6376 0000204D 750F                <1>       JNZ   short N_720         ; JUMP IF NOT
6377 0000204F F6C404              <1>       TEST  AH,DRV_DET          ; IS DRIVE DETERMINED
6378 00002052 7410                <1>       JZ    short ISNT_12       ; TREAT AS NON 1.2 DRIVE
6379 00002054 F6C402              <1>       TEST  AH,FMT_CAPA         ; IS 1.2M
6380 00002057 740B                <1>       JZ    short ISNT_12       ; JUMP IF NOT
6381 00002059 80CC50              <1>       OR    AH,MED_DET+RATE_300 ; RATE 300
6382 0000205C EB09                <1>       JMP   SHORT SKP_STATE           ; CONTINUE
6383                              <1> N_720:
6384 0000205E FEC8                <1>       DEC   AL                  ; CHECK FOR TYPE 4
6385 00002060 750C                <1>       JNZ   short CL_DRV        ; NO DRIVE, CMOS BAD
6386 00002062 EBE2                <1>       JMP   SHORT F1_RATE
6387                              <1> ISNT_12:
6388 00002064 80CC90              <1>       OR    AH,MED_DET+RATE_250 ; MUST BE RATE 250
6389                              <1>
6390                              <1> SKP_STATE:
6391 00002067 88A7[F9700000]      <1>       MOV   [DSK_STATE+eDI], AH ; STORE AWAY
6392                              <1> F1_OUT:
6393 0000206D C3                  <1>       RETn
6394                              <1> CL_DRV:
6395 0000206E 30E4                <1>       XOR   AH,AH               ; CLEAR STATE
6396 00002070 EBF5                <1>       JMP   SHORT SKP_STATE           ; SAVE IT
6397                              <1>
6398                              <1> ;--------------------------------------------------------------------------------
6399                              <1> ; MED_CHANGE
6400                              <1> ;     CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
6401                              <1> ;     CHECKS MEDIA CHANGE AGAIN.
6402                              <1> ;
6403                              <1> ; ON EXIT:   CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
6404                              <1> ;            @DSKETTE_STATUS = ERROR CODE
6405                              <1> ;--------------------------------------------------------------------------------
6406                              <1> MED_CHANGE:
6407 00002072 E888060000          <1>       CALL  READ_DSKCHNG        ; READ DISK CHANCE LINE STATE
6408 00002077 7447                <1>       JZ    short MC_OUT        ; BYPASS HANDLING DISK CHANGE LINE
6409 00002079 80A7[F9700000]EF    <1>       AND   byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
6410                              <1>
6411                              <1> ;     THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
6412                              <1> ;     ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
6413                              <1> ;     BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
6414                              <1>
6415 00002080 6689F9              <1>       MOV   CX,DI               ; CL = DRIVE 0
6416 00002083 B001                <1>       MOV   AL,1                ; MOTOR ON BIT MASK
6417 00002085 D2E0                <1>       SHL   AL,CL               ; TO APPROPRIATE POSITION
6418 00002087 F6D0                <1>       NOT   AL                  ; KEEP ALL BUT MOTOR ON
6419 00002089 FA                  <1>       CLI                       ; NO INTERRUPTS
6420 0000208A 2005[EA700000]      <1>       AND   [MOTOR_STATUS], AL  ; TURN MOTOR OFF INDICATOR
6421 00002090 FB                  <1>       STI                       ; INTERRUPTS ENABLED
6422 00002091 E810040000          <1>       CALL  MOTOR_ON            ; TURN MOTOR ON
6423                              <1>
6424                              <1> ;-----      THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
6425                              <1>
6426 00002096 E884F9FFFF          <1>       CALL  DSK_RESET           ; RESET NEC
6427 0000209B B501                <1>       MOV   CH,01H              ; MOVE TO CYLINDER 1
6428 0000209D E8FF040000          <1>       CALL  SEEK                ; ISSUE SEEK
6429 000020A2 30ED                <1>       XOR   CH,CH               ; MOVE TO CYLINDER 0
6430 000020A4 E8F8040000          <1>       CALL  SEEK                ; ISSUE SEEK
6431 000020A9 C605[EC700000]06    <1>       MOV   byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
6432                              <1> OK1:
6433 000020B0 E84A060000          <1>       CALL  READ_DSKCHNG        ; CHECK MEDIA CHANGED AGAIN
6434 000020B5 7407                <1>       JZ    short OK2           ; IF ACTIVE, NO DISKETTE, TIMEOUT
6435                              <1> OK4:
6436 000020B7 C605[EC700000]80    <1>       MOV   byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
6437                              <1> OK2:
6438 000020BE F9                  <1>       STC                       ; MEDIA CHANGED, SET CY
6439 000020BF C3                  <1>       RETn
6440                              <1> MC_OUT:
6441 000020C0 F8                  <1>       CLC                       ; NO MEDIA CHANGED, CLEAR CY
6442 000020C1 C3                  <1>       RETn
6443                              <1>
6444                              <1> ;--------------------------------------------------------------------------------
6445                              <1> ; SEND_RATE
6446                              <1> ;     SENDS DATA RATE COMMAND TO NEC
6447                              <1> ; ON ENTRY:  DI = DRIVE #
6448                              <1> ; ON EXIT:   NONE
6449                              <1> ; REGISTERS ALTERED: DX
6450                              <1> ;--------------------------------------------------------------------------------
6451                              <1> SEND_RATE:
6452 000020C2 6650                <1>       PUSH  AX                  ; SAVE REG.
6453 000020C4 8025[F4700000]3F    <1>       AND   byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
6454 000020CB 8A87[F9700000]      <1>       MOV   AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
6455 000020D1 24C0                <1>       AND   AL,SEND_MSK         ; KEEP ONLY RATE BITS
6456 000020D3 0805[F4700000]      <1>       OR    [LASTRATE], AL            ; SAVE NEW RATE FOR NEXT CHECK
6457 000020D9 C0C002              <1>       ROL   AL,2                ; MOVE TO BIT OUTPUT POSITIONS
6458 000020DC 66BAF703            <1>       MOV   DX,03F7H            ; OUTPUT NEW DATA RATE
6459 000020E0 EE                  <1>       OUT   DX,AL
6460 000020E1 6658                <1>       POP   AX                  ; RESTORE REG.
6461 000020E3 C3                  <1>       RETn
6462                              <1>
6463                              <1> ;--------------------------------------------------------------------------------
6464                              <1> ; CHK_LASTRATE
6465                              <1> ;     CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
6466                              <1> ; ON ENTRY:
6467                              <1> ;     DI = DRIVE #
6468                              <1> ; ON EXIT:
6469                              <1> ;     ZF =  1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
6470                              <1> ;     ZF =  0 DATA RATE IS DIFFERENT FROM LAST RATE
6471                              <1> ; REGISTERS ALTERED: DX
6472                              <1> ;--------------------------------------------------------------------------------
6473                              <1> CHK_LASTRATE:
6474 000020E4 6650                <1>       PUSH  AX                  ; SAVE REG
6475 000020E6 2225[F4700000]      <1>       AND   AH, [LASTRATE]            ; GET LAST DATA RATE SELECTED
6476 000020EC 8A87[F9700000]      <1>       MOV   AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
```

```
6477 000020F2 6625C0C0          <1>         AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
6478 000020F6 38E0              <1>       CMP    AL, AH              ; COMPARE TO PREVIOUSLY TRIED
6479                            <1>                                  ; ZF = 1 RATE IS THE SAME
6480 000020F8 6658              <1>       POP    AX                  ; RESTORE REG.
6481 000020FA C3                <1>       RETn
6482                            <1>
6483                            <1> ;--------------------------------------------------------------------------------
6484                            <1> ; DMA_SETUP
6485                            <1> ;     THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
6486                            <1> ;
6487                            <1> ; ON ENTRY:  AL = DMA COMMAND
6488                            <1> ;
6489                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6490                            <1> ;--------------------------------------------------------------------------------
6491                            <1>
6492                            <1> ; SI = Head #, # of Sectors or DASD Type
6493                            <1>
6494                            <1> ; 22/08/2015
6495                            <1> ; 08/02/2015 - Protected Mode Modification
6496                            <1> ; 06/02/2015 - 07/02/2015
6497                            <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
6498                            <1> ; (DMA Addres = Physical Address)
6499                            <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
6500                            <1> ;
6501                            <1> ; 04/02/2016 (clc)
6502                            <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
6503                            <1> ; 16/12/2014 (IODELAY)
6504                            <1>
6505                            <1> DMA_SETUP:
6506                            <1>
6507                            <1> ;; 20/02/2015
6508 000020FB 8B5504            <1>       mov    edx, [ebp+4]        ; Buffer address
6509 000020FE F7C2000000FF      <1>       test   edx, 0FF000000h           ; 16 MB limit (22/08/2015, bugfix)
6510 00002104 756E              <1>       jnz    short dma_bnd_err_stc
6511                            <1>       ;
6512 00002106 6650              <1>       push   ax                  ; DMA command
6513 00002108 52                <1>       push   edx                 ; *
6514 00002109 B203              <1>       mov    dl, 3               ; GET BYTES/SECTOR PARAMETER
6515 0000210B E851030000        <1>       call   GET_PARM            ;
6516 00002110 88E1              <1>       mov    cl, ah                   ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
6517 00002112 6689F0            <1>       mov    ax, si              ; Sector count
6518 00002115 88C4              <1>       mov    ah, al              ; AH =  # OF SECTORS
6519 00002117 28C0              <1>       sub    al, al              ; AL = 0, AX = # SECTORS * 256
6520 00002119 66D1E8            <1>       shr    ax, 1               ; AX = # SECTORS * 128
6521 0000211C 66D3E0            <1>       shl    ax, cl              ; SHIFT BY PARAMETER VALUE
6522 0000211F 6648              <1>       dec    ax                  ; -1 FOR DMA VALUE
6523 00002121 6689C1            <1>       mov    cx, ax
6524 00002124 5A                <1>       pop    edx                 ; *
6525 00002125 6658              <1>       pop    ax
6526 00002127 3C42              <1>       cmp    al, 42h
6527 00002129 7507              <1>         jne     short NOT_VERF
6528 0000212B BA0000FF00        <1>       mov    edx, 0FF0000h
6529 00002130 EB08              <1>       jmp    short J33
6530                            <1> NOT_VERF:
6531 00002132 6601CA            <1>       add    dx, cx              ; check for overflow
6532 00002135 723E              <1>       jc     short dma_bnd_err
6533                            <1>       ;
6534 00002137 6629CA            <1>       sub    dx, cx              ; Restore start address
6535                            <1> J33:
6536 0000213A FA                <1>       CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
6537 0000213B E60C              <1>       OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
6538                            <1>       IODELAY                          ; WAIT FOR I/O
6539 0000213D EB00              <2>  jmp short $+2
6540 0000213F EB00              <2>  jmp short $+2
6541 00002141 E60B              <1>       OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
6542 00002143 89D0              <1>       mov    eax, edx            ; Buffer address
6543 00002145 E604              <1>       OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
6544                            <1>       IODELAY                          ; WAIT FOR I/O
6545 00002147 EB00              <2>  jmp short $+2
6546 00002149 EB00              <2>  jmp short $+2
6547 0000214B 88E0              <1>       MOV    AL,AH
6548 0000214D E604              <1>       OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
6549 0000214F C1E810            <1>       shr    eax, 16
6550                            <1>       IODELAY                          ; I/O WAIT STATE
6551 00002152 EB00              <2>  jmp short $+2
6552 00002154 EB00              <2>  jmp short $+2
6553 00002156 E681              <1>       OUT    081H,AL                  ; OUTPUT highest BITS TO PAGE REGISTER
6554                            <1>       IODELAY
6555 00002158 EB00              <2>  jmp short $+2
6556 0000215A EB00              <2>  jmp short $+2
6557 0000215C 6689C8            <1>       mov    ax, cx              ; Byte count - 1
6558 0000215F E605              <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
6559                            <1>       IODELAY                          ; WAIT FOR I/O
6560 00002161 EB00              <2>  jmp short $+2
6561 00002163 EB00              <2>  jmp short $+2
6562 00002165 88E0              <1>       MOV    AL, AH
6563 00002167 E605              <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
6564                            <1>       IODELAY
6565 00002169 EB00              <2>  jmp short $+2
6566 0000216B EB00              <2>  jmp short $+2
6567 0000216D FB                <1>       STI                        ; RE-ENABLE INTERRUPTS
6568 0000216E B002              <1>       MOV    AL, 2               ; MODE FOR 8237
6569 00002170 E60A              <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
6570                            <1>
6571 00002172 F8                <1>       clc    ; 04/02/2016
6572 00002173 C3                <1>       retn
6573                            <1>
6574                            <1> dma_bnd_err_stc:
6575 00002174 F9                <1>       stc
6576                            <1> dma_bnd_err:
6577 00002175 C605[EC700000]09  <1>       MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6578 0000217C C3                <1>       RETn                       ; CY SET BY ABOVE IF ERROR
6579                            <1>
6580                            <1> ;; 16/12/2014
6581                            <1> ;;    CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
```

```
6582                            <1> ;;    OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
6583                            <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
6584                            <1> ;;    IODELAY
6585                            <1> ;;    OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
6586                            <1> ;;    ;SIODELAY
6587                            <1> ;;      ;CMP AL, 42H                  ; DMA VERIFY COMMAND
6588                            <1> ;;      ;JNE short NOT_VERF           ; NO
6589                            <1> ;;      ;XOR AX, AX             ; START ADDRESS
6590                            <1> ;;      ;JMP SHORT J33
6591                            <1> ;;;NOT_VERF:
6592                            <1> ;;    ;MOV   AX,ES              ; GET THE ES VALUE
6593                            <1> ;;    ;ROL   AX,4               ; ROTATE LEFT
6594                            <1> ;;    ;MOV   CH,AL              ; GET HIGHEST NIBBLE OF ES TO CH
6595                            <1> ;;    ;AND   AL,11110000B       ; ZERO THE LOW NIBBLE FROM SEGMENT
6596                            <1> ;;    ;ADD   AX,[BP+2]          ; TEST FOR CARRY FROM ADDITION
6597                            <1> ;;    mov    eax, [ebp+4] ; 06/02/2015
6598                            <1> ;;    ;JNC   short J33
6599                            <1> ;;    ;INC   CH                 ; CARRY MEANS HIGH 4 BITS MUST BE INC
6600                            <1> ;;;J33:
6601                            <1> ;;    PUSH   eAX                ; SAVE START ADDRESS
6602                            <1> ;;    OUT    DMA+4,AL           ; OUTPUT LOW ADDRESS
6603                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
6604                            <1> ;;    IODELAY
6605                            <1> ;;    MOV    AL,AH
6606                            <1> ;;    OUT    DMA+4,AL           ; OUTPUT HIGH ADDRESS
6607                            <1> ;;    shr    eax, 16           ; 07/02/2015
6608                            <1> ;;    ;MOV   AL,CH              ; GET HIGH 4 BITS
6609                            <1> ;;    ;JMP   $+2                ; I/O WAIT STATE
6610                            <1> ;;    IODELAY
6611                            <1> ;;    ;AND   AL,00001111B
6612                            <1> ;;    OUT    081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
6613                            <1> ;;    ;SIODELAY
6614                            <1> ;;
6615                            <1> ;;;----- DETERMINE COUNT
6616                            <1> ;;    sub    eax, eax ; 08/02/2015
6617                            <1> ;;    MOV    AX, SI             ; AL =  # OF SECTORS
6618                            <1> ;;    XCHG   AL, AH             ; AH =  # OF SECTORS
6619                            <1> ;;    SUB    AL, AL             ; AL = 0, AX = # SECTORS * 256
6620                            <1> ;;    SHR    AX, 1              ; AX = # SECTORS * 128
6621                            <1> ;;    PUSH   AX                 ; SAVE # OF SECTORS * 128
6622                            <1> ;;    MOV    DL, 3              ; GET BYTES/SECTOR PARAMETER
6623                            <1> ;;    CALL   GET_PARM           ; "
6624                            <1> ;;    MOV    CL,AH              ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
6625                            <1> ;;    POP    AX                 ; AX = # SECTORS * 128
6626                            <1> ;;    SHL    AX,CL              ; SHIFT BY PARAMETER VALUE
6627                            <1> ;;    DEC    AX                 ; -1 FOR DMA VALUE
6628                            <1> ;;    PUSH   eAX  ; 08/02/2015  ; SAVE COUNT VALUE
6629                            <1> ;;    OUT    DMA+5,AL           ; LOW BYTE OF COUNT
6630                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
6631                            <1> ;;    IODELAY
6632                            <1> ;;    MOV    AL, AH
6633                            <1> ;;    OUT    DMA+5,AL           ; HIGH BYTE OF COUNT
6634                            <1> ;;    ;IODELAY
6635                            <1> ;;    STI                       ; RE-ENABLE INTERRUPTS
6636                            <1> ;;    POP    eCX  ; 08/02/2015  ; RECOVER COUNT VALUE
6637                            <1> ;;    POP    eAX  ; 08/02/2015  ; RECOVER ADDRESS VALUE
6638                            <1> ;;    ;ADD   AX, CX             ; ADD, TEST FOR 64K OVERFLOW
6639                            <1> ;;    add    ecx, eax ; 08/02/2015
6640                            <1> ;;    MOV    AL, 2              ; MODE FOR 8237
6641                            <1> ;;    ;JMP   $+2                ; WAIT FOR I/O
6642                            <1> ;;    SIODELAY
6643                            <1> ;;    OUT    DMA+10, AL         ; INITIALIZE THE DISKETTE CHANNEL
6644                            <1> ;;    ;JNC   short NO_BAD       ; CHECK FOR ERROR
6645                            <1> ;;    jc     short dma_bnd_err ; 08/02/2015
6646                            <1> ;;    and    ecx, 0FFF00000h ; 16 MB limit
6647                            <1> ;;    jz     short NO_BAD
6648                            <1> ;;dma_bnd_err:
6649                            <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6650                            <1> ;;NO_BAD:
6651                            <1> ;;    RETn                      ; CY SET BY ABOVE IF ERROR
6652                            <1>
6653                            <1> ;--------------------------------------------------------------------------------
6654                            <1> ; FMTDMA_SET
6655                            <1> ;     THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
6656                            <1> ;
6657                            <1> ; ON ENTRY:  NOTHING REQUIRED
6658                            <1> ;
6659                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6660                            <1> ;--------------------------------------------------------------------------------
6661                            <1>
6662                            <1> FMTDMA_SET:
6663                            <1> ;; 20/02/2015 modification
6664 0000217D 8B5504           <1>       mov    edx, [ebp+4]       ; Buffer address
6665 00002180 F7C20000F0FF     <1>       test   edx, 0FFF00000h          ; 16 MB limit
6666 00002186 75EC             <1>       jnz    short dma_bnd_err_stc
6667                            <1>       ;
6668 00002188 6652             <1>       push   dx                 ; *
6669 0000218A B204             <1>       mov    DL, 4              ; SECTORS/TRACK VALUE IN PARM TABLE
6670 0000218C E8D0020000       <1>       call   GET_PARM           ; "
6671 00002191 88E0             <1>       mov    al, ah             ; AL = SECTORS/TRACK VALUE
6672 00002193 28E4             <1>       sub    ah, ah             ; AX = SECTORS/TRACK VALUE
6673 00002195 66C1E002         <1>       shl    ax, 2              ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
6674 00002199 6648             <1>       dec    ax                 ; -1 FOR DMA VALUE
6675 0000219B 6689C1           <1>       mov    cx, ax
6676 0000219E 665A             <1>       pop    dx                 ; *
6677 000021A0 6601CA           <1>       add    dx, cx             ; check for overflow
6678 000021A3 72D0             <1>       jc     short dma_bnd_err
6679                            <1>       ;
6680 000021A5 6629CA           <1>       sub    dx, cx             ; Restore start address
6681                            <1>       ;
6682 000021A8 B04A             <1>       MOV    AL, 04AH           ; WILL WRITE TO THE DISKETTE
6683 000021AA FA               <1>       CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
6684 000021AB E60C             <1>       OUT    DMA+12,AL          ; SET THE FIRST/LA5T F/F
6685                            <1>       IODELAY                       ; WAIT FOR I/O
6686 000021AD EB00             <2>  jmp short $+2
```

```
6687 000021AF EB00              <2>  jmp short $+2
6688 000021B1 E60B              <1>       OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
6689 000021B3 89D0              <1>       mov    eax, edx            ; Buffer address
6690 000021B5 E604              <1>       OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
6691                            <1>       IODELAY                          ; WAIT FOR I/O
6692 000021B7 EB00              <2>  jmp short $+2
6693 000021B9 EB00              <2>  jmp short $+2
6694 000021BB 88E0              <1>       MOV    AL,AH
6695 000021BD E604              <1>       OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
6696 000021BF C1E810            <1>       shr    eax, 16
6697                            <1>       IODELAY                          ; I/O WAIT STATE
6698 000021C2 EB00              <2>  jmp short $+2
6699 000021C4 EB00              <2>  jmp short $+2
6700 000021C6 E681              <1>       OUT    081H,AL                   ; OUTPUT highest BITS TO PAGE REGISTER
6701                            <1>       IODELAY
6702 000021C8 EB00              <2>  jmp short $+2
6703 000021CA EB00              <2>  jmp short $+2
6704 000021CC 6689C8            <1>       mov    ax, cx              ; Byte count - 1
6705 000021CF E605              <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
6706                            <1>       IODELAY                          ; WAIT FOR I/O
6707 000021D1 EB00              <2>  jmp short $+2
6708 000021D3 EB00              <2>  jmp short $+2
6709 000021D5 88E0              <1>       MOV    AL, AH
6710 000021D7 E605              <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
6711                            <1>       IODELAY
6712 000021D9 EB00              <2>  jmp short $+2
6713 000021DB EB00              <2>  jmp short $+2
6714 000021DD FB                <1>       STI                        ; RE-ENABLE INTERRUPTS
6715 000021DE B002              <1>       MOV    AL, 2               ; MODE FOR 8237
6716 000021E0 E60A              <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
6717 000021E2 C3                <1>       retn
6718                            <1>
6719                            <1> ;; 08/02/2015 - Protected Mode Modification
6720                            <1> ;;    MOV    AL, 04AH            ; WILL WRITE TO THE DISKETTE
6721                            <1> ;;    CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
6722                            <1> ;;    OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
6723                            <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
6724                            <1> ;;    IODELAY
6725                            <1> ;;    OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
6726                            <1> ;;    ;MOV   AX,ES               ; GET THE ES VALUE
6727                            <1> ;;    ;ROL   AX,4                ; ROTATE LEFT
6728                            <1> ;;    ;MOV   CH,AL               ; GET HIGHEST NIBBLE OF ES TO CH
6729                            <1> ;;    ;AND   AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
6730                            <1> ;;    ;ADD   AX,[BP+2]           ; TEST FOR CARRY FROM ADDITION
6731                            <1> ;;    ;JNC   short J33A
6732                            <1> ;;    ;INC   CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
6733                            <1> ;;    mov    eax, [ebp+4] ; 08/02/2015
6734                            <1> ;;;J33A:
6735                            <1> ;;    PUSH   eAX ; 08/02/2015    ; SAVE START ADDRESS
6736                            <1> ;;    OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
6737                            <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
6738                            <1> ;;    IODELAY
6739                            <1> ;;    MOV    AL,AH
6740                            <1> ;;    OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
6741                            <1> ;;    shr    eax, 16 ; 08/02/2015
6742                            <1> ;;    ;MOV   AL,CH               ; GET HIGH 4 BITS
6743                            <1> ;;    ;JMP   $+2                 ; I/O WAIT STATE
6744                            <1> ;;    IODELAY
6745                            <1> ;;    ;AND   AL,00001111B
6746                            <1> ;;    OUT    081H,AL                  ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
6747                            <1> ;;
6748                            <1> ;;;----- DETERMINE COUNT
6749                            <1> ;;    sub    eax, eax ; 08/02/2015
6750                            <1> ;;    MOV    DL, 4               ; SECTORS/TRACK VALUE IN PARM TABLE
6751                            <1> ;;    CALL   GET_PARM            ; "
6752                            <1> ;;    XCHG   AL, AH              ; AL = SECTORS/TRACK VALUE
6753                            <1> ;;    SUB    AH, AH              ; AX = SECTORS/TRACK VALUE
6754                            <1> ;;    SHL    AX, 2               ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
6755                            <1> ;;    DEC    AX                  ; -1 FOR DMA VALUE
6756                            <1> ;;    PUSH   eAX    ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED
6757                            <1> ;;    OUT    DMA+5,AL            ; LOW BYTE OF COUNT
6758                            <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
6759                            <1> ;;    IODELAY
6760                            <1> ;;    MOV    AL, AH
6761                            <1> ;;    OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
6762                            <1> ;;    STI                        ; RE-ENABLE INTERRUPTS
6763                            <1> ;;    POP    eCX    ; 08/02/2015 ; RECOVER COUNT VALUE
6764                            <1> ;;    POP    eAX    ; 08/02/2015 ; RECOVER ADDRESS VALUE
6765                            <1> ;;    ;ADD   AX, CX              ; ADD, TEST FOR 64K OVERFLOW
6766                            <1> ;;    add    ecx, eax ; 08/02/2015
6767                            <1> ;;    MOV    AL, 2               ; MODE FOR 8237
6768                            <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
6769                            <1> ;;    SIODELAY
6770                            <1> ;;    OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
6771                            <1> ;;    ;JNC   short FMTDMA_OK           ; CHECK FOR ERROR
6772                            <1> ;;    jc     short fmtdma_bnd_err ; 08/02/2015
6773                            <1> ;;    and    ecx, 0FFF00000h  ; 16 MB limit
6774                            <1> ;;    jz     short FMTDMA_OK
6775                            <1> ;;    stc    ; 20/02/2015
6776                            <1> ;;fmtdma_bnd_err:
6777                            <1> ;;    MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6778                            <1> ;;FMTDMA_OK:
6779                            <1> ;;    RETn                       ; CY SET BY ABOVE IF ERROR
6780                            <1>
6781                            <1> ;--------------------------------------------------------------------------------
6782                            <1> ; NEC_INIT
6783                            <1> ;     THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
6784                            <1> ;     THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
6785                            <1> ;
6786                            <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
6787                            <1> ;
6788                            <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6789                            <1> ;--------------------------------------------------------------------------------
6790                            <1> NEC_INIT:
6791 000021E3 6650              <1>       PUSH   AX                  ; SAVE NEC COMMAND
```

```
6792 000021E5 E8BC020000          <1>       CALL   MOTOR_ON           ; TURN MOTOR ON FOR SPECIFIC DRIVE
6793                              <1>
6794                              <1> ;-----      DO THE SEEK OPERATION
6795                              <1>
6796 000021EA 8A6D01              <1>       MOV    CH,[eBP+1]         ; CH = TRACK #
6797 000021ED E8AF030000          <1>       CALL   SEEK               ; MOVE TO CORRECT TRACK
6798 000021F2 6658                <1>       POP    AX                 ; RECOVER COMMAND
6799 000021F4 721E                <1>       JC     short ER_1         ; ERROR ON SEEK
6800 000021F6 BB[14220000]        <1>       MOV    eBX, ER_1          ; LOAD ERROR ADDRESS
6801 000021FB 53                  <1>       PUSH   eBX                ; PUSH NEC_OUT ERROR RETURN
6802                              <1>
6803                              <1> ;-----      SEND OUT THE PARAMETERS TO THE CONTROLLER
6804                              <1>
6805 000021FC E866030000          <1>       CALL   NEC_OUTPUT         ; OUTPUT THE OPERATION COMMAND
6806 00002201 6689F0              <1>       MOV    AX,SI              ; AH = HEAD #
6807 00002204 89FB                <1>       MOV    eBX,eDI                 ; BL = DRIVE #
6808 00002206 C0E402              <1>       SAL    AH,2               ; MOVE IT TO BIT 2
6809 00002209 80E404              <1>       AND    AH,00000100B       ; ISOLATE THAT BIT
6810 0000220C 08DC                <1>       OR     AH,BL              ; OR IN THE DRIVE NUMBER
6811 0000220E E854030000          <1>       CALL   NEC_OUTPUT         ; FALL THRU CY SET IF ERROR
6812 00002213 5B                  <1>       POP    eBX                ; THROW AWAY ERROR RETURN
6813                              <1> ER_1:
6814 00002214 C3                  <1>       RETn
6815                              <1>
6816                              <1> ;--------------------------------------------------------------------------------
6817                              <1> ; RWV_COM
6818                              <1> ;     THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
6819                              <1> ;     READ/WRITE/VERIFY OPERATIONS.
6820                              <1> ;
6821                              <1> ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
6822                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6823                              <1> ;--------------------------------------------------------------------------------
6824                              <1> RWV_COM:
6825 00002215 B8[60220000]        <1>       MOV    eAX, ER_2          ; LOAD ERROR ADDRESS
6826 0000221A 50                  <1>       PUSH   eAX                ; PUSH NEC_OUT ERROR RETURN
6827 0000221B 8A6501              <1>       MOV    AH,[eBP+1]         ; OUTPUT TRACK #
6828 0000221E E844030000          <1>       CALL   NEC_OUTPUT
6829 00002223 6689F0              <1>       MOV    AX,SI              ; OUTPUT HEAD #
6830 00002226 E83C030000          <1>       CALL   NEC_OUTPUT
6831 0000222B 8A6500              <1>         MOV     AH,[eBP]               ; OUTPUT SECTOR #
6832 0000222E E834030000          <1>       CALL   NEC_OUTPUT
6833 00002233 B203                <1>       MOV    DL,3               ; BYTES/SECTOR PARAMETER FROM BLOCK
6834 00002235 E827020000          <1>       CALL   GET_PARM           ; ... TO THE NEC
6835 0000223A E828030000          <1>       CALL   NEC_OUTPUT         ; OUTPUT TO CONTROLLER
6836 0000223F B204                <1>       MOV    DL,4               ; EOT PARAMETER FROM BLOCK
6837 00002241 E81B020000          <1>       CALL   GET_PARM           ; ... TO THE NEC
6838 00002246 E81C030000          <1>       CALL   NEC_OUTPUT         ; OUTPUT TO CONTROLLER
6839 0000224B 8A6305              <1>         MOV     AH, [eBX+MD.GAP]       ; GET GAP LENGTH
6840                              <1> _R15:
6841 0000224E E814030000          <1>       CALL   NEC_OUTPUT
6842 00002253 B206                <1>       MOV    DL,6               ; DTL PARAMETER PROM BLOCK
6843 00002255 E807020000          <1>       CALL   GET_PARM           ;  TO THE NEC
6844 0000225A E808030000          <1>       CALL   NEC_OUTPUT         ; OUTPUT TO CONTROLLER
6845 0000225F 58                  <1>       POP    eAX                ; THROW AWAY ERROR EXIT
6846                              <1> ER_2:
6847 00002260 C3                  <1>       RETn
6848                              <1>
6849                              <1> ;--------------------------------------------------------------------------------
6850                              <1> ; NEC_TERM
6851                              <1> ;     THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
6852                              <1> ;     FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
6853                              <1> ;
6854                              <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6855                              <1> ;--------------------------------------------------------------------------------
6856                              <1> NEC_TERM:
6857                              <1>
6858                              <1> ;-----      LET THE OPERATION HAPPEN
6859                              <1>
6860 00002261 56                  <1>       PUSH   eSI                ; SAVE HEAD #, # OF SECTORS
6861 00002262 E80D040000          <1>       CALL   WAIT_INT           ; WAIT FOR THE INTERRUPT
6862 00002267 9C                  <1>       PUSHF
6863 00002268 E837040000          <1>       CALL   RESULTS                  ; GET THE NEC STATUS
6864 0000226D 724B                <1>       JC     short SET_END_POP
6865 0000226F 9D                  <1>       POPF
6866 00002270 723E                <1>       JC     short SET_END      ; LOOK FOR ERROR
6867                              <1>
6868                              <1> ;-----      CHECK THE RESULTS RETURNED BY THE CONTROLLER
6869                              <1>
6870 00002272 FC                  <1>       CLD                       ; SET THE CORRECT DIRECTION
6871 00002273 BE[ED700000]        <1>       MOV    eSI, NEC_STATUS          ; POINT TO STATUS FIELD
6872 00002278 AC                  <1>       lodsb                     ; GET ST0
6873 00002279 24C0                <1>       AND    AL,11000000B       ; TEST FOR NORMAL TERMINATION
6874 0000227B 7433                <1>       JZ     short SET_END
6875 0000227D 3C40                <1>       CMP    AL,01000000B       ; TEST FOR ABNORMAL TERMINATION
6876 0000227F 7527                <1>       JNZ    short J18          ; NOT ABNORMAL, BAD NEC
6877                              <1>
6878                              <1> ;-----      ABNORMAL TERMINATION, FIND OUT WHY
6879                              <1>
6880 00002281 AC                  <1>       lodsb                     ; GET ST1
6881 00002282 D0E0                <1>       SAL    AL,1               ; TEST FOR EDT FOUND
6882 00002284 B404                <1>       MOV    AH,RECORD_NOT_FND
6883 00002286 7222                <1>       JC     short J19
6884 00002288 C0E002              <1>       SAL    AL,2
6885 0000228B B410                <1>       MOV    AH,BAD_CRC
6886 0000228D 721B                <1>       JC     short J19
6887 0000228F D0E0                <1>       SAL    AL,1               ; TEST FOR DMA OVERRUN
6888 00002291 B408                <1>       MOV    AH,BAD_DMA
6889 00002293 7215                <1>       JC     short J19
6890 00002295 C0E002              <1>       SAL    AL,2               ; TEST FOR RECORD NOT FOUND
6891 00002298 B404                <1>       MOV    AH,RECORD_NOT_FND
6892 0000229A 720E                <1>       JC     short J19
6893 0000229C D0E0                <1>       SAL    AL,1
6894 0000229E B403                <1>       MOV    AH,WRITE_PROTECT   ; TEST FOR WRITE_PROTECT
6895 000022A0 7208                <1>       JC     short J19
6896 000022A2 D0E0                <1>       SAL    AL,1               ; TEST MISSING ADDRESS MARK
```

```
6897 000022A4 B402                <1>       MOV    AH,BAD_ADDR_MARK
6898 000022A6 7202                <1>       JC     short J19
6899                              <1>
6900                              <1> ;-----       NEC MUST HAVE FAILED
6901                              <1> J18:
6902 000022A8 B420                <1>       MOV    AH,BAD_NEC
6903                              <1> J19:
6904 000022AA 0825[EC700000]      <1>       OR     [DSKETTE_STATUS], AH
6905                              <1> SET_END:
6906 000022B0 803D[EC700000]01    <1>       CMP    byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
6907 000022B7 F5                  <1>       CMC
6908 000022B8 5E                  <1>       POP    eSI
6909 000022B9 C3                  <1>       RETn                      ; RESTORE HEAD #, # OF SECTORS
6910                              <1>
6911                              <1> SET_END_POP:
6912 000022BA 9D                  <1>       POPF
6913 000022BB EBF3                <1>       JMP    SHORT SET_END
6914                              <1>
6915                              <1> ;--------------------------------------------------------------------------------
6916                              <1> ; DSTATE:    ESTABLISH STATE UPON SUCCESSFUL OPERATION.
6917                              <1> ;--------------------------------------------------------------------------------
6918                              <1> DSTATE:
6919 000022BD 803D[EC700000]00    <1>       CMP    byte [DSKETTE_STATUS],0   ; CHECK FOR ERROR
6920 000022C4 753E                <1>       JNZ    short SETBAC            ; IF ERROR JUMP
6921 000022C6 808F[F9700000]10    <1>       OR     byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
6922 000022CD F687[F9700000]04    <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
6923 000022D4 752E                <1>       JNZ    short SETBAC        ; IF DETERMINED NO TRY TO DETERMINE
6924 000022D6 8A87[F9700000]      <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
6925 000022DC 24C0                <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
6926 000022DE 3C80                <1>       CMP    AL,RATE_250         ; RATE 250 ?
6927 000022E0 751B                <1>       JNE    short M_12          ; NO, MUST BE 1.2M OR 1.44M DRIVE
6928                              <1>
6929                              <1> ;-----       CHECK IF IT IS 1.44M
6930                              <1>
6931 000022E2 E871010000          <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
6932                              <1>       ;;20/02/2015
6933                              <1>       ;;JC   short M_12          ; CMOS BAD
6934 000022E7 7414                <1>       jz     short M_12 ;; 20/02/2015
6935 000022E9 3C04                <1>       CMP    AL, 4               ; 1.44MB DRIVE ?
6936 000022EB 7410                <1>       JE     short M_12          ; YES
6937                              <1> M_720:
6938 000022ED 80A7[F9700000]FD    <1>       AND    byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
6939 000022F4 808F[F9700000]04    <1>       OR     byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
6940 000022FB EB07                <1>       JMP    SHORT SETBAC        ; BACK
6941                              <1> M_12:
6942 000022FD 808F[F9700000]06    <1>       OR     byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
6943                              <1>                                  ; TURN ON DETERMINED & FMT CAPA
6944                              <1> SETBAC:
6945 00002304 C3                  <1>       RETn
6946                              <1>
6947                              <1> ;--------------------------------------------------------------------------------
6948                              <1> ; RETRY
6949                              <1> ;     DETERMINES WHETHER A RETRY IS NECESSARY.
6950                              <1> ;     IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
6951                              <1> ;
6952                              <1> ; ON EXIT:  CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
6953                              <1> ;--------------------------------------------------------------------------------
6954                              <1> RETRY:
6955 00002305 803D[EC700000]00    <1>       CMP    byte [DSKETTE_STATUS],0   ; GET STATUS OF OPERATION
6956 0000230C 7445                <1>       JZ     short NO_RETRY            ; SUCCESSFUL OPERATION
6957 0000230E 803D[EC700000]80    <1>       CMP    byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
6958 00002315 743C                <1>       JZ     short NO_RETRY
6959 00002317 8AA7[F9700000]      <1>       MOV    AH,[DSK_STATE+eDI]  ; GET MEDIA STATE OF DRIVE
6960 0000231D F6C410              <1>       TEST   AH,MED_DET          ; ESTABLISHED/DETERMINED ?
6961 00002320 7531                <1>       JNZ    short NO_RETRY           ; IF ESTABLISHED STATE THEN TRUE ERROR
6962 00002322 80E4C0              <1>       AND    AH,RATE_MSK         ; ISOLATE RATE
6963 00002325 8A2D[F4700000]      <1>       MOV    CH,[LASTRATE]       ; GET START OPERATION STATE
6964 0000232B C0C504              <1>       ROL    CH,4                ; TO CORRESPONDING BITS
6965 0000232E 80E5C0              <1>       AND    CH,RATE_MSK         ; ISOLATE RATE BITS
6966 00002331 38E5                <1>       CMP    CH,AH               ; ALL RATES TRIED
6967 00002333 741E                <1>       JE     short NO_RETRY           ; IF YES, THEN TRUE ERROR
6968                              <1>
6969                              <1> ;     SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
6970                              <1> ;      00000000B (500) -> 10000000B    (250)
6971                              <1> ;      10000000B (250) -> 01000000B    (300)
6972                              <1> ;      01000000B (300) -> 00000000B    (500)
6973                              <1>
6974 00002335 80FC01              <1>       CMP    AH,RATE_500+1       ; SET CY FOR RATE 500
6975 00002338 D0DC                <1>       RCR    AH,1                ; TO NEXT STATE
6976 0000233A 80E4C0              <1>       AND    AH,RATE_MSK         ; KEEP ONLY RATE BITS
6977 0000233D 80A7[F9700000]1F    <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
6978                              <1>                                  ; RATE, DBL STEP OFF
6979 00002344 08A7[F9700000]      <1>       OR     [DSK_STATE+eDI],AH  ; TURN ON NEW RATE
6980 0000234A C605[EC700000]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; RESET STATUS FOR RETRY
6981 00002351 F9                  <1>       STC                        ; SET CARRY FOR RETRY
6982 00002352 C3                  <1>       RETn                       ; RETRY RETURN
6983                              <1>
6984                              <1> NO_RETRY:
6985 00002353 F8                  <1>       CLC                        ; CLEAR CARRY NO RETRY
6986 00002354 C3                  <1>       RETn                       ; NO RETRY RETURN
6987                              <1>
6988                              <1> ;--------------------------------------------------------------------------------
6989                              <1> ; NUM_TRANS
6990                              <1> ;     THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
6991                              <1> ;     ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
6992                              <1> ;
6993                              <1> ; ON ENTRY:  [BP+1] = TRACK
6994                              <1> ;            SI-HI  = HEAD
6995                              <1> ;            [BP]   = START SECTOR
6996                              <1> ;
6997                              <1> ; ON EXIT:   AL = NUMBER ACTUALLY TRANSFERRED
6998                              <1> ;--------------------------------------------------------------------------------
6999                              <1> NUM_TRANS:
7000 00002355 30C0                <1>       XOR    AL,AL               ; CLEAR FOR ERROR
7001 00002357 803D[EC700000]00    <1>       CMP    byte [DSKETTE_STATUS],0   ; CHECK FOR ERROR
```

```
7002 0000235E 752C              <1>       JNZ    NT_OUT              ; IF ERROR 0 TRANSFERRED
7003 00002360 B204              <1>       MOV    DL,4                ; SECTORS/TRACK OFFSET TO DL
7004 00002362 E8FA000000        <1>       CALL   GET_PARM            ; AH = SECTORS/TRACK
7005 00002367 8A1D[F2700000]    <1>       MOV    BL, [NEC_STATUS+5]  ; GET ENDING SECTOR
7006 0000236D 6689F1            <1>       MOV    CX,SI               ; CH = HEAD # STARTED
7007 00002370 3A2D[F1700000]    <1>       CMP    CH, [NEC_STATUS+4]  ; GET HEAD ENDED UP ON
7008 00002376 750D              <1>       JNZ    DIF_HD              ; IF ON SAME HEAD, THEN NO ADJUST
7009 00002378 8A2D[F0700000]    <1>       MOV    CH, [NEC_STATUS+3]  ; GET TRACK ENDED UP ON
7010 0000237E 3A6D01            <1>       CMP    CH,[eBP+1]          ; IS IT ASKED FOR TRACK
7011 00002381 7404              <1>       JZ     short SAME_TRK           ; IF SAME TRACK NO INCREASE
7012 00002383 00E3              <1>       ADD    BL,AH               ; ADD SECTORS/TRACK
7013                            <1> DIF_HD:
7014 00002385 00E3              <1>       ADD    BL,AH               ; ADD SECTORS/TRACK
7015                            <1> SAME_TRK:
7016 00002387 2A5D00            <1>       SUB    BL,[eBP]            ; SUBTRACT START FROM END
7017 0000238A 88D8              <1>       MOV    AL,BL               ; TO AL
7018                            <1> NT_OUT:
7019 0000238C C3                <1>       RETn
7020                            <1>
7021                            <1> ;---------------------------------------------------------------------------
7022                            <1> ; SETUP_END
7023                            <1> ;     RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
7024                            <1> ;     AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
7025                            <1> ;
7026                            <1> ; ON EXIT:
7027                            <1> ;     AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
7028                            <1> ;---------------------------------------------------------------------------
7029                            <1> SETUP_END:
7030 0000238D B202              <1>       MOV    DL,2                ; GET THE MOTOR WAIT PARAMETER
7031 0000238F 6650              <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
7032 00002391 E8CB000000        <1>       CALL   GET_PARM
7033 00002396 8825[EB700000]    <1>       MOV    [MOTOR_COUNT],AH    ; STORE UPON RETURN
7034 0000239C 6658              <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
7035 0000239E 8A25[EC700000]    <1>       MOV    AH, [DSKETTE_STATUS]      ; GET STATUS OF OPERATION
7036 000023A4 08E4              <1>       OR     AH,AH               ; CHECK FOR ERROR
7037 000023A6 7402              <1>       JZ     short NUN_ERR            ; NO ERROR
7038 000023A8 30C0              <1>       XOR    AL,AL               ; CLEAR NUMBER RETURNED
7039                            <1> NUN_ERR:
7040 000023AA 80FC01            <1>       CMP    AH,1                ; SET THE CARRY FLAG TO INDICATE
7041 000023AD F5                <1>       CMC                        ; SUCCESS OR FAILURE
7042 000023AE C3                <1>       RETn
7043                            <1>
7044                            <1> ;---------------------------------------------------------------------------
7045                            <1> ; SETUP_DBL
7046                            <1> ;     CHECK DOUBLE STEP.
7047                            <1> ;
7048                            <1> ; ON ENTRY : DI = DRIVE
7049                            <1> ;
7050                            <1> ; ON EXIT :  CY = 1 MEANS ERROR
7051                            <1> ;---------------------------------------------------------------------------
7052                            <1> SETUP_DBL:
7053 000023AF 8AA7[F9700000]    <1>       MOV    AH, [DSK_STATE+eDI] ; ACCESS STATE
7054 000023B5 F6C410            <1>       TEST   AH,MED_DET          ; ESTABLISHED STATE ?
7055 000023B8 757E              <1>       JNZ    short NO_DBL             ; IF ESTABLISHED THEN DOUBLE DONE
7056                            <1>
7057                            <1> ;-----      CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
7058                            <1>
7059 000023BA C605[E9700000]00  <1>       MOV    byte [SEEK_STATUS],0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
7060 000023C1 E8E0000000        <1>       CALL   MOTOR_ON            ; ENSURE MOTOR STAY ON
7061 000023C6 B500              <1>       MOV    CH,0                ; LOAD TRACK 0
7062 000023C8 E8D4010000        <1>       CALL   SEEK                ; SEEK TO TRACK 0
7063 000023CD E868000000        <1>       CALL   READ_ID                  ; READ ID FUNCTION
7064 000023D2 7249              <1>       JC     short SD_ERR        ; IF ERROR NO TRACK 0
7065                            <1>
7066                            <1> ;-----      INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
7067                            <1>
7068 000023D4 66B95004          <1>       MOV    CX,0450H            ; START, MAX TRACKS
7069 000023D8 F687[F9700000]01  <1>       TEST   byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
7070 000023DF 7402              <1>       JZ     short CNT_OK        ; IF NOT COUNT IS SETUP
7071 000023E1 B1A0              <1>       MOV    CL,0A0H                  ; MAXIMUM TRACK 1.2 MB
7072                            <1>
7073                            <1> ;     ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
7074                            <1> ;     MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
7075                            <1> ;     THEN SET DOUBLE STEP ON.
7076                            <1>
7077                            <1> CNT_OK:
7078 000023E3 C605[EB700000]FF  <1>         MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
7079 000023EA 6651              <1>       PUSH   CX                  ; SAVE TRACK, COUNT
7080 000023EC C605[EC700000]00  <1>       MOV    byte [DSKETTE_STATUS],0   ; CLEAR STATUS, EXPECT ERRORS
7081 000023F3 6631C0            <1>       XOR    AX,AX               ; CLEAR AX
7082 000023F6 D0ED              <1>       SHR    CH,1                ; HALVE TRACK, CY = HEAD
7083 000023F8 C0D003            <1>       RCL    AL,3                ; AX = HEAD IN CORRECT BIT
7084 000023FB 6650              <1>       PUSH   AX                  ; SAVE HEAD
7085 000023FD E89F010000        <1>       CALL   SEEK                ; SEEK TO TRACK
7086 00002402 6658              <1>       POP    AX                  ; RESTORE HEAD
7087 00002404 6609C7            <1>       OR     DI,AX               ; DI = HEAD OR'ED DRIVE
7088 00002407 E82E000000        <1>       CALL   READ_ID                  ; READ ID HEAD 0
7089 0000240C 9C                <1>       PUSHF                      ; SAVE RETURN FROM READ_ID
7090 0000240D 6681E7FB00        <1>       AND    DI,11111011B        ; TURN OFF HEAD 1 BIT
7091 00002412 9D                <1>       POPF                       ; RESTORE ERROR RETURN
7092 00002413 6659              <1>       POP    CX                  ; RESTORE COUNT
7093 00002415 7308              <1>       JNC    short DO_CHK        ; IF OK, ASKED = RETURNED TRACK ?
7094 00002417 FEC5              <1>       INC    CH                  ; INC FOR NEXT TRACK
7095 00002419 38CD              <1>       CMP    CH,CL               ; REACHED MAXIMUM YET
7096 0000241B 75C6              <1>       JNZ    short CNT_OK        ; CONTINUE TILL ALL TRIED
7097                            <1>
7098                            <1> ;-----      FALL THRU, READ ID FAILED FOR ALL TRACKS
7099                            <1>
7100                            <1> SD_ERR:
7101 0000241D F9                <1>       STC                        ; SET CARRY FOR ERROR
7102 0000241E C3                <1>       RETn                       ; SETUP_DBL ERROR EXIT
7103                            <1>
7104                            <1> DO_CHK:
7105 0000241F 8A0D[F0700000]    <1>       MOV    CL, [NEC_STATUS+3]  ; LOAD RETURNED TRACK
7106 00002425 888F[FD700000]    <1>       MOV    [DSK_TRK+eDI], CL   ; STORE TRACK NUMBER
```

```
7107 0000242B D0ED                <1>       SHR    CH,1                ; HALVE TRACK
7108 0000242D 38CD                <1>       CMP    CH,CL               ; IS IT THE SAME AS ASKED FOR TRACK
7109 0000242F 7407                <1>       JZ     short NO_DBL        ; IF SAME THEN NO DOUBLE STEP
7110 00002431 808F[F9700000]20    <1>       OR     byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
7111                              <1> NO_DBL:
7112 00002438 F8                  <1>       CLC                        ; CLEAR ERROR FLAG
7113 00002439 C3                  <1>       RETn
7114                              <1>
7115                              <1> ;--------------------------------------------------------------------------
7116                              <1> ; READ_ID
7117                              <1> ;     READ ID FUNCTION.
7118                              <1> ;
7119                              <1> ; ON ENTRY:  DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
7120                              <1> ;
7121                              <1> ; ON EXIT:   DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
7122                              <1> ;            @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
7123                              <1> ;--------------------------------------------------------------------------
7124                              <1> READ_ID:
7125 0000243A B8[57240000]        <1>       MOV    eAX, ER_3           ; MOVE NEC OUTPUT ERROR ADDRESS
7126 0000243F 50                  <1>       PUSH   eAX
7127 00002440 B44A                <1>       MOV    AH,4AH              ; READ ID COMMAND
7128 00002442 E820010000          <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
7129 00002447 6689F8              <1>       MOV    AX,DI               ; DRIVE # TO AH, HEAD 0
7130 0000244A 88C4                <1>       MOV    AH,AL
7131 0000244C E816010000          <1>       CALL   NEC_OUTPUT          ; TO CONTROLLER
7132 00002451 E80BFEFFFF          <1>       CALL   NEC_TERM            ; WAIT FOR OPERATION, GET STATUS
7133 00002456 58                  <1>       POP    eAX                 ; THROW AWAY ERROR ADDRESS
7134                              <1> ER_3:
7135 00002457 C3                  <1>       RETn
7136                              <1>
7137                              <1> ;--------------------------------------------------------------------------
7138                              <1> ; CMOS_TYPE
7139                              <1> ;     RETURNS DISKETTE TYPE FROM CMOS
7140                              <1> ;
7141                              <1> ; ON ENTRY:  DI = DRIVE #
7142                              <1> ;
7143                              <1> ; ON EXIT:   AL = TYPE; CY REFLECTS STATUS
7144                              <1> ;--------------------------------------------------------------------------
7145                              <1>
7146                              <1> CMOS_TYPE: ; 11/12/2014
7147 00002458 8A87[1E6B0000]      <1> mov   al, [eDI+fd0_type]
7148 0000245E 20C0                <1> and   al, al ; 18/12/2014
7149 00002460 C3                  <1> retn
7150                              <1>
7151                              <1> ;CMOS_TYPE:
7152                              <1> ;     MOV    AL, CMOS_DIAG       ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
7153                              <1> ;     CALL   CMOS_READ           ; GET CMOS STATUS
7154                              <1> ;     TEST   AL,BAD_BAT+BAD_CKSUM      ; BATTERY GOOD AND CHECKSUM VALID
7155                              <1> ;     STC                        ; SET CY = 1 INDICATING ERROR FOR RETURN
7156                              <1> ;     JNZ    short BAD_CM        ; ERROR IF EITHER BIT ON
7157                              <1> ;     MOV    AL,CMOS_DISKETTE    ; ADDRESS OF DISKETTE BYTE IN CMOS
7158                              <1> ;     CALL   CMOS_READ           ; GET DISKETTE BYTE
7159                              <1> ;     OR     DI,DI               ; SEE WHICH DRIVE IN QUESTION
7160                              <1> ;     JNZ    short TB            ; IF DRIVE 1, DATA IN LOW NIBBLE
7161                              <1> ;     ROR    AL,4                ; EXCHANGE NIBBLES IF SECOND DRIVE
7162                              <1> ;TB:
7163                              <1> ;     AND    AL,0FH              ; KEEP ONLY DRIVE DATA, RESET CY, 0
7164                              <1> ;BAD_CM:
7165                              <1> ;     RETn                       ; CY, STATUS OF READ
7166                              <1>
7167                              <1> ;--------------------------------------------------------------------------
7168                              <1> ; GET_PARM
7169                              <1> ;     THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
7170                              <1> ;     BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
7171                              <1> ;     THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
7172                              <1> ;     THE PARAMETER IN DL.
7173                              <1> ;
7174                              <1> ; ON ENTRY:  DL = INDEX OF BYTE TO BE FETCHED
7175                              <1> ;
7176                              <1> ; ON EXIT:   AH = THAT BYTE FROM BLOCK
7177                              <1> ;            AL,DH DESTROYED
7178                              <1> ;--------------------------------------------------------------------------
7179                              <1> GET_PARM:
7180                              <1>       ;PUSH  DS
7181 00002461 56                  <1>       PUSH   eSI
7182                              <1>       ;SUB   AX,AX               ; DS = 0, BIOS DATA AREA
7183                              <1>       ;MOV   DS,AX
7184                              <1>       ;;mov  ax, cs
7185                              <1>       ;;mov  ds, ax
7186                              <1>       ; 08/02/2015 (protected mode modifications, bx -> ebx)
7187 00002462 87D3                <1>       XCHG   eDX,eBX                   ; BL = INDEX
7188                              <1>       ;SUB   BH,BH               ; BX = INDEX
7189 00002464 81E3FF000000        <1>       and    ebx, 0FFh
7190                              <1>       ;LDS   SI, [DISK_POINTER]  ; POINT TO BLOCK
7191                              <1>       ;
7192                              <1>       ; 17/12/2014
7193 0000246A 66A1[116B0000]      <1>       mov    ax, [cfd] ; current (AL) and previous fd (AH)
7194 00002470 38E0                <1>       cmp    al, ah
7195 00002472 7425                <1>       je     short gpndc
7196 00002474 A2[126B0000]        <1>       mov    [pfd], al ; current drive -> previous drive
7197 00002479 53                  <1>       push   ebx ; 08/02/2015
7198 0000247A 88C3                <1>       mov    bl, al
7199                              <1>       ; 11/12/2014
7200 0000247C 8A83[1E6B0000]      <1>       mov    al, [eBX+fd0_type]  ; Drive type (0,1,2,3,4)
7201                              <1>       ; 18/12/2014
7202 00002482 20C0                <1>       and    al, al
7203 00002484 7507                <1>       jnz    short gpdtc
7204 00002486 BB[FB6A0000]        <1>       mov    ebx, MD_TBL6        ; 1.44 MB param. tbl. (default)
7205 0000248B EB05                <1>         jmp     short gpdpu
7206                              <1> gpdtc:
7207 0000248D E817F9FFFF          <1>       call   DR_TYPE_CHECK
7208                              <1>       ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
7209                              <1> gpdpu:
7210 00002492 891D[986A0000]      <1>       mov    [DISK_POINTER], ebx
7211 00002498 5B                  <1>       pop    ebx
```

```
7212                                <1> gpndc:
7213 00002499 8B35[986A0000]        <1>       mov    esi, [DISK_POINTER] ; 08/02/2015, si -> esi
7214 0000249F 8A241E                <1>       MOV    AH, [eSI+eBX]       ; GET THE WORD
7215 000024A2 87D3                  <1>       XCHG   eDX,eBX                   ; RESTORE BX
7216 000024A4 5E                    <1>       POP    eSI
7217                                <1>       ;POP   DS
7218 000024A5 C3                    <1>       RETn
7219                                <1>
7220                                <1> ;--------------------------------------------------------------------------
7221                                <1> ; MOTOR_ON
7222                                <1> ;     TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
7223                                <1> ;     IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
7224                                <1> ;     THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
7225                                <1> ;     MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
7226                                <1> ;     (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
7227                                <1> ;     THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
7228                                <1> ;     FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
7229                                <1> ;     HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
7230                                <1> ;     THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
7231                                <1> ;     NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
7232                                <1> ;     PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
7233                                <1> ;     IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
7234                                <1> ;     WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
7235                                <1> ;
7236                                <1> ; ON ENTRY:  DI = DRIVE #
7237                                <1> ; ON EXIT:   AX,CX,DX DESTROYED
7238                                <1> ;--------------------------------------------------------------------------
7239                                <1> MOTOR_ON:
7240 000024A6 53                    <1>       PUSH   eBX                 ; SAVE REG.
7241 000024A7 E82A000000            <1>       CALL   TURN_ON                   ; TURN ON MOTOR
7242 000024AC 7226                  <1>       JC     short MOT_IS_ON           ; IF CY=1 NO WAIT
7243 000024AE E89BF9FFFF            <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
7244 000024B3 E865F9FFFF            <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH,
7245                                <1>       ;CALL  TURN_ON             ; CHECK AGAIN IF MOTOR ON
7246                                <1>       ;JC    MOT_IS_ON           ; IF NO WAIT MEANS IT IS ON
7247                                <1> M_WAIT:
7248 000024B8 B20A                  <1>       MOV    DL,10               ; GET THE MOTOR WAIT PARAMETER
7249 000024BA E8A2FFFFFF            <1>       CALL   GET_PARM
7250                                <1>       ;MOV   AL,AH               ; AL = MOTOR WAIT PARAMETER
7251                                <1>       ;XOR   AH,AH               ; AX = MOTOR WAIT PARAMETER
7252                                <1>       ;CMP   AL,8                ; SEE IF AT LEAST A SECOND IS SPECIFIED
7253 000024BF 80FC08                <1>       cmp    ah, 8
7254                                <1>       ;JAE   short GP2           ; IF YES, CONTINUE
7255 000024C2 7702                  <1>       ja     short J13
7256                                <1>       ;MOV   AL,8                ; ONE SECOND WAIT FOR MOTOR START UP
7257 000024C4 B408                  <1>       mov    ah, 8
7258                                <1>
7259                                <1> ;-----       AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
7260                                <1> GP2:
7261                                <1> ;-----       FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
7262                                <1> J13:                           ; WAIT FOR 1/8 SECOND PER (AL)
7263 000024C6 B95E200000            <1>       MOV    eCX,8286            ; COUNT FOR 1/8 SECOND AT 15.085737 US
7264 000024CB E819F1FFFF            <1>       CALL   WAITF               ; GO TO FIXED WAIT ROUTINE
7265                                <1>       ;DEC   AL                  ; DECREMENT TIME VALUE
7266 000024D0 FECC                  <1>       dec    ah
7267 000024D2 75F2                  <1>       JNZ    short J13           ; ARE WE DONE YET
7268                                <1> MOT_IS_ON:
7269 000024D4 5B                    <1>       POP    eBX                 ; RESTORE REG.
7270 000024D5 C3                    <1>       RETn
7271                                <1>
7272                                <1> ;--------------------------------------------------------------------------
7273                                <1> ; TURN_ON
7274                                <1> ;     TURN MOTOR ON AND RETURN WAIT STATE.
7275                                <1> ;
7276                                <1> ; ON ENTRY:  DI = DRIVE #
7277                                <1> ;
7278                                <1> ; ON EXIT:   CY = 0 MEANS WAIT REQUIRED
7279                                <1> ;            CY = 1 MEANS NO WAIT REQUIRED
7280                                <1> ;            AX,BX,CX,DX DESTROYED
7281                                <1> ;--------------------------------------------------------------------------
7282                                <1> TURN_ON:
7283 000024D6 89FB                  <1>       MOV    eBX,eDI                  ; BX = DRIVE #
7284 000024D8 88D9                  <1>       MOV    CL,BL               ; CL = DRIVE #
7285 000024DA C0C304                <1>       ROL    BL,4                ; BL = DRIVE SELECT
7286 000024DD FA                    <1>       CLI                        ; NO INTERRUPTS WHILE DETERMINING STATUS
7287 000024DE C605[EB700000]FF      <1>       MOV    byte [MOTOR_COUNT],0FFH   ; ENSURE MOTOR STAYS ON FOR OPERATION
7288 000024E5 A0[EA700000]          <1>       MOV    AL, [MOTOR_STATUS]  ; GET DIGITAL OUTPUT REGISTER REFLECTION
7289 000024EA 2430                  <1>       AND    AL,00110000B        ; KEEP ONLY DRIVE SELECT BITS
7290 000024EC B401                  <1>       MOV    AH,1                ; MASK FOR DETERMINING MOTOR BIT
7291 000024EE D2E4                  <1>       SHL    AH,CL               ; AH = MOTOR ON, A=00000001, B=00000010
7292                                <1>
7293                                <1> ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
7294                                <1> ;  BL = DRIVE SELECT DESIRED
7295                                <1> ;  AH = MOTOR ON MASK DESIRED
7296                                <1>
7297 000024F0 38D8                  <1>       CMP    AL,BL               ; REQUESTED DRIVE ALREADY SELECTED ?
7298 000024F2 7508                  <1>       JNZ    short TURN_IT_ON    ; IF NOT SELECTED JUMP
7299 000024F4 8425[EA700000]        <1>       TEST   AH, [MOTOR_STATUS]  ; TEST MOTOR ON BIT
7300 000024FA 7535                  <1>       JNZ    short NO_MOT_WAIT   ; JUMP IF MOTOR ON AND SELECTED
7301                                <1>
7302                                <1> TURN_IT_ON:
7303 000024FC 08DC                  <1>       OR     AH,BL               ; AH = DRIVE SELECT AND MOTOR ON
7304 000024FE 8A3D[EA700000]        <1>       MOV    BH,[MOTOR_STATUS]   ; SAVE COPY OF @MOTOR_STATUS BEFORE
7305 00002504 80E70F                <1>       AND    BH,00001111B        ; KEEP ONLY MOTOR BITS
7306 00002507 8025[EA700000]CF      <1>       AND    byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
7307 0000250E 0825[EA700000]        <1>       OR     [MOTOR_STATUS],AH   ; OR IN DRIVE SELECTED AND MOTOR ON
7308 00002514 A0[EA700000]          <1>       MOV    AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
7309 00002519 88C3                  <1>       MOV    BL,AL               ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
7310 0000251B 80E30F                <1>       AND    BL,00001111B        ; KEEP ONLY MOTOR BITS
7311 0000251E FB                    <1>       STI                        ; ENABLE INTERRUPTS AGAIN
7312 0000251F 243F                  <1>       AND    AL,00111111B        ; STRIP AWAY UNWANTED BITS
7313 00002521 C0C004                <1>       ROL    AL,4                ; PUT BITS IN DESIRED POSITIONS
7314 00002524 0C0C                  <1>       OR     AL,00001100B        ; NO RESET, ENABLE DMA/INTERRUPT
7315 00002526 66BAF203              <1>       MOV    DX,03F2H            ; SELECT DRIVE AND TURN ON MOTOR
7316 0000252A EE                    <1>       OUT    DX,AL
```

```
7317 0000252B 38FB              <1>       CMP    BL,BH               ; NEW MOTOR TURNED ON ?
7318                            <1>       ;JZ    short NO_MOT_WAIT   ; NO WAIT REQUIRED IF JUST SELECT
7319 0000252D 7403              <1>       je     short no_mot_w1 ; 27/02/2015
7320 0000252F F8                <1>       CLC                        ; (re)SET CARRY MEANING WAIT
7321 00002530 C3                <1>       RETn
7322                            <1>
7323                            <1> NO_MOT_WAIT:
7324 00002531 FB                <1>       sti
7325                            <1> no_mot_w1: ; 27/02/2015
7326 00002532 F9                <1>       STC                        ; SET NO WAIT REQUIRED
7327                            <1>       ;STI                       ; INTERRUPTS BACK ON
7328 00002533 C3                <1>       RETn
7329                            <1>
7330                            <1> ;--------------------------------------------------------------------------------
7331                            <1> ; HD_WAIT
7332                            <1> ;     WAIT FOR HEAD SETTLE TIME.
7333                            <1> ;
7334                            <1> ; ON ENTRY:  DI = DRIVE #
7335                            <1> ;
7336                            <1> ; ON EXIT:   AX,BX,CX,DX DESTROYED
7337                            <1> ;--------------------------------------------------------------------------------
7338                            <1> HD_WAIT:
7339 00002534 B209              <1>       MOV    DL,9                ; GET HEAD SETTLE PARAMETER
7340 00002536 E826FFFFFF        <1>       CALL   GET_PARM
7341 0000253B 08E4              <1>       or     ah, ah ; 17/12/2014
7342 0000253D 7519              <1>       jnz    short DO_WAT
7343 0000253F F605[EA700000]80  <1>         TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
7344                            <1>       ;JZ    short ISNT_WRITE    ; IF NOT, DO NOT ENFORCE ANY VALUES
7345                            <1>       ;OR    AH,AH               ; CHECK FOR ANY WAIT?
7346                            <1>       ;JNZ   short DO_WAT        ; IF THERE DO NOT ENFORCE
7347 00002546 741E              <1>       jz     short HW_DONE
7348 00002548 B40F              <1>       MOV    AH,HD12_SETTLE            ; LOAD 1.2M HEAD SETTLE MINIMUM
7349 0000254A 8A87[F9700000]    <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
7350 00002550 24C0              <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
7351 00002552 3C80              <1>       CMP    AL,RATE_250         ; 1.2 M DRIVE ?
7352 00002554 7502              <1>       JNZ    short DO_WAT        ; DEFAULT HEAD SETTLE LOADED
7353                            <1> ;GP3:
7354 00002556 B414              <1>       MOV    AH,HD320_SETTLE           ; USE 320/360 HEAD SETTLE
7355                            <1> ;     JMP    SHORT DO_WAT
7356                            <1>
7357                            <1> ;ISNT_WRITE:
7358                            <1> ;     OR     AH,AH               ; CHECK FOR NO WAIT
7359                            <1> ;     JZ     short HW_DONE       ; IF NOT WRITE AND 0 ITS OK
7360                            <1>
7361                            <1> ;-----       AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
7362                            <1> DO_WAT:
7363                            <1> ;     MOV    AL,AH               ; AL = # MILLISECONDS
7364                            <1> ;     ;XOR   AH,AH               ; AX = # MILLISECONDS
7365                            <1> J29:                             ;     1 MILLISECOND LOOP
7366                            <1>       ;mov   cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
7367 00002558 B942000000        <1>       MOV    eCX,66              ; COUNT AT 15.085737 US PER COUNT
7368 0000255D E887F0FFFF        <1>       CALL   WAITF               ; DELAY FOR 1 MILLISECOND
7369                            <1>       ;DEC   AL                  ; DECREMENT THE COUNT
7370 00002562 FECC              <1>       dec    ah
7371 00002564 75F2              <1>       JNZ    short J29           ; DO AL MILLISECOND # OF TIMES
7372                            <1> HW_DONE:
7373 00002566 C3                <1>       RETn
7374                            <1>
7375                            <1> ;--------------------------------------------------------------------------------
7376                            <1> ; NEC_OUTPUT
7377                            <1> ;     THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
7378                            <1> ;     FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
7379                            <1> ;     TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
7380                            <1> ;     OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
7381                            <1> ;
7382                            <1> ; ON ENTRY:  AH = BYTE TO BE OUTPUT
7383                            <1> ;
7384                            <1> ; ON EXIT:   CY = 0  SUCCESS
7385                            <1> ;            CY = 1  FAILURE -- DISKETTE STATUS UPDATED
7386                            <1> ;                    IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
7387                            <1> ;                    HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
7388                            <1> ;                    REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
7389                            <1> ;           AX,CX,DX DESTROYED
7390                            <1> ;--------------------------------------------------------------------------------
7391                            <1>
7392                            <1> ; 09/12/2014 [Erdogan Tan]
7393                            <1> ;     (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
7394                            <1> ; Diskette Drive Controller Status Register (3F4h)
7395                            <1> ;     This read only register facilitates the transfer of data between
7396                            <1> ;     the system microprocessor and the controller.
7397                            <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
7398                            <1> ;       with the system micrprocessor.
7399                            <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
7400                            <1> ;       the transfer is to the controller.
7401                            <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
7402                            <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
7403                            <1> ; Bit 3 - Reserved.
7404                            <1> ; Bit 2 - Reserved.
7405                            <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
7406                            <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.
7407                            <1>
7408                            <1> ; Data Register (3F5h)
7409                            <1> ; This read/write register passes data, commands and parameters, and provides
7410                            <1> ; diskette status information.
7411                            <1>
7412                            <1> NEC_OUTPUT:
7413                            <1>       ;PUSH  BX                  ; SAVE REG.
7414 00002567 66BAF403          <1>       MOV    DX,03F4H            ; STATUS PORT
7415                            <1>       ;MOV   BL,2                ; HIGH ORDER COUNTER
7416                            <1>       ;XOR   CX,CX               ; COUNT FOR TIME OUT
7417                            <1>       ; 16/12/2014
7418                            <1>       ; waiting for (max.) 0.5 seconds
7419                            <1>         ;;mov     byte [wait_count], 0 ;; 27/02/2015
7420                            <1>       ;
7421                            <1>       ; 17/12/2014
```

```
7422                                    <1>       ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
7423                                    <1>       ;
7424                                    <1>       ;WAIT_FOR_PORT:    Waits for a bit at a port pointed to by DX to
7425                                    <1>       ;            go on.
7426                                    <1>       ;INPUT:
7427                                    <1>       ;      AH=Mask for isolation bits.
7428                                    <1>       ;      AL=pattern to look for.
7429                                    <1>       ;      DX=Port to test for
7430                                    <1>       ;      BH:CX=Number of memory refresh periods to delay.
7431                                    <1>       ;           (normally 30 microseconds per period.)
7432                                    <1>       ;
7433                                    <1>       ;WFP_SHORT:
7434                                    <1>       ;      Wait for port if refresh cycle is short (15-80 Us range).
7435                                    <1>       ;
7436                                    <1>
7437                                    <1> ;     mov    bl, WAIT_FDU_SEND_HI+1    ; 0+1
7438                                    <1> ;     mov    cx, WAIT_FDU_SEND_LO      ; 16667
7439 0000256B B91B410000               <1>       mov    ecx, WAIT_FDU_SEND_LH   ; 16667 (27/02/2015)
7440                                    <1> ;
7441                                    <1> ;WFPS_OUTER_LP:
7442                                    <1> ;     ;
7443                                    <1> ;WFPS_CHECK_PORT:
7444                                    <1> J23:
7445 00002570 EC                       <1>       IN     AL,DX              ; GET STATUS
7446 00002571 24C0                     <1>       AND    AL,11000000B       ; KEEP STATUS AND DIRECTION
7447 00002573 3C80                     <1>       CMP    AL,10000000B       ; STATUS 1 AND DIRECTION 0 ?
7448 00002575 7418                     <1>       JZ     short J27          ; STATUS AND DIRECTION OK
7449                                    <1> WFPS_HI:
7450 00002577 E461                     <1>       IN     AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
7451 00002579 A810                     <1>       TEST   AL,010H                  ; transition on memory
7452 0000257B 75FA                     <1>       JNZ    SHORT WFPS_HI       ; refresh.
7453                                    <1> WFPS_LO:
7454 0000257D E461                     <1>       IN     AL, PORT_B          ; SYS1
7455 0000257F A810                     <1>       TEST   AL,010H
7456 00002581 74FA                     <1>       JZ     SHORT WFPS_LO
7457                                    <1>       ;LOOP  SHORT WFPS_CHECK_PORT
7458 00002583 E2EB                     <1>       loop   J23    ; 27/02/2015
7459                                    <1> ;     ;
7460                                    <1> ;     dec    bl
7461                                    <1> ;     jnz    short WFPS_OUTER_LP
7462                                    <1> ;     jmp    short WFPS_TIMEOUT  ; fail
7463                                    <1> ;J23:
7464                                    <1> ;     IN     AL,DX              ; GET STATUS
7465                                    <1> ;     AND    AL,11000000B       ; KEEP STATUS AND DIRECTION
7466                                    <1> ;     CMP    AL,10000000B       ; STATUS 1 AND DIRECTION 0 ?
7467                                    <1> ;     JZ     short J27          ; STATUS AND DIRECTION OK
7468                                    <1>       ;LOOP  J23                ; CONTINUE TILL CX EXHAUSTED
7469                                    <1>       ;DEC   BL                 ; DECREMENT COUNTER
7470                                    <1>       ;JNZ   short J23          ; REPEAT TILL DELAY FINISHED, CX = 0
7471                                    <1>
7472                                    <1>       ;;27/02/2015
7473                                    <1>       ;16/12/2014
7474                                    <1>         ;;cmp     byte [wait_count], 10   ; (10/18.2 seconds)
7475                                    <1>       ;;jb   short J23
7476                                    <1>
7477                                    <1> ;WFPS_TIMEOUT:
7478                                    <1>
7479                                    <1> ;-----      FALL THRU TO ERROR RETURN
7480                                    <1>
7481 00002585 800D[EC700000]80         <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
7482                                    <1>       ;POP   BX                 ; RESTORE REG.
7483 0000258C 58                       <1>       POP    eAX ; 08/02/2015   ; DISCARD THE RETURN ADDRESS
7484 0000258D F9                       <1>       STC                       ; INDICATE ERROR TO CALLER
7485 0000258E C3                       <1>       RETn
7486                                    <1>
7487                                    <1> ;-----      DIRECTION AND STATUS OK; OUTPUT BYTE
7488                                    <1>
7489                                    <1> J27:
7490 0000258F 88E0                     <1>       MOV    AL,AH              ; GET BYTE TO OUTPUT
7491 00002591 6642                     <1>       INC    DX                 ; DATA PORT = STATUS PORT + 1
7492 00002593 EE                       <1>       OUT    DX,AL              ; OUTPUT THE BYTE
7493                                    <1>       ;;NEWIODELAY  ;; 27/02/2015
7494                                    <1>       ; 27/02/2015
7495 00002594 9C                       <1>       PUSHF                     ; SAVE FLAGS
7496 00002595 B903000000               <1>       MOV    eCX, 3             ; 30 TO 45 MICROSECONDS WAIT FOR
7497 0000259A E84AF0FFFF               <1>       CALL   WAITF              ; NEC FLAGS UPDATE CYCLE
7498 0000259F 9D                       <1>       POPF                      ; RESTORE FLAGS FOR EXIT
7499                                    <1>       ;POP   BX                 ; RESTORE REG
7500 000025A0 C3                       <1>       RETn                      ; CY = 0 FROM TEST INSTRUCTION
7501                                    <1>
7502                                    <1> ;--------------------------------------------------------------------------
7503                                    <1> ; SEEK
7504                                    <1> ;     THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
7505                                    <1> ;     TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
7506                                    <1> ;     RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
7507                                    <1> ;
7508                                    <1> ; ON ENTRY:  DI = DRIVE #
7509                                    <1> ;            CH = TRACK #
7510                                    <1> ;
7511                                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
7512                                    <1> ;            AX,BX,CX DX DESTROYED
7513                                    <1> ;--------------------------------------------------------------------------
7514                                    <1> SEEK:
7515 000025A1 89FB                     <1>       MOV    eBX,eDI                 ; BX = DRIVE #
7516 000025A3 B001                     <1>       MOV    AL,1               ; ESTABLISH MASK FOR RECALIBRATE TEST
7517 000025A5 86CB                     <1>       XCHG   CL,BL              ; SET DRIVE VALULE INTO CL
7518 000025A7 D2C0                     <1>       ROL    AL,CL              ; SHIFT MASK BY THE DRIVE VALUE
7519 000025A9 86CB                     <1>       XCHG   CL,BL              ; RECOVER TRACK VALUE
7520 000025AB 8405[E9700000]           <1>       TEST   AL,[SEEK_STATUS]   ; TEST FOR RECALIBRATE REQUIRED
7521 000025B1 7526                     <1>       JNZ    short J28A         ; JUMP IF RECALIBRATE NOT REQUIRED
7522                                    <1>
7523 000025B3 0805[E9700000]           <1>       OR     [SEEK_STATUS],AL   ; TURN ON THE NO RECALIBRATE BIT IN FLAG
7524 000025B9 E862000000               <1>       CALL   RECAL              ; RECALIBRATE DRIVE
7525 000025BE 730E                     <1>       JNC    short AFT_RECAL         ; RECALIBRATE DONE
7526                                    <1>
```

```
7527                                <1> ;-----      ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
7528                                <1>
7529 000025C0 C605[EC700000]00      <1>       MOV   byte [DSKETTE_STATUS],0   ; CLEAR OUT INVALID STATUS
7530 000025C7 E854000000            <1>       CALL  RECAL               ; RECALIBRATE DRIVE
7531 000025CC 7251                  <1>       JC    short RB            ; IF RECALIBRATE FAILS TWICE THEN ERROR
7532                                <1>
7533                                <1> AFT_RECAL:
7534 000025CE C687[FD700000]00      <1>         MOV     byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
7535 000025D5 08ED                  <1>       OR    CH,CH               ; CHECK FOR SEEK TO TRACK 0
7536 000025D7 743F                  <1>       JZ    short DO_WAIT       ; HEAD SETTLE, CY = 0 IF JUMP
7537                                <1>
7538                                <1> ;-----      DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
7539                                <1>
7540 000025D9 F687[F9700000]20      <1> J28A: TEST  byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
7541 000025E0 7402                  <1>       JZ    short _R7           ; SINGLE STEP REQUIRED BYPASS DOUBLE
7542 000025E2 D0E5                  <1>       SHL   CH,1                ; DOUBLE NUMBER OF STEP TO TAKE
7543                                <1>
7544 000025E4 3AAF[FD700000]        <1> _R7:  CMP   CH, [DSK_TRK+eDI]   ; SEE IF ALREADY AT THE DESIRED TRACK
7545 000025EA 7433                  <1>       JE    short RB            ; IF YES, DO NOT NEED TO SEEK
7546                                <1>
7547 000025EC BA[1F260000]          <1>       MOV   eDX, NEC_ERR        ; LOAD RETURN ADDRESS
7548 000025F1 52                    <1>       PUSH  eDX ; (*)           ; ON STACK FOR NEC OUTPUT ERROR
7549 000025F2 88AF[FD700000]        <1>       MOV   [DSK_TRK+eDI],CH    ; SAVE NEW CYLINDER AS PRESENT POSITION
7550 000025F8 B40F                  <1>       MOV   AH,0FH              ; SEEK COMMAND TO NEC
7551 000025FA E868FFFFFF            <1>       CALL  NEC_OUTPUT
7552 000025FF 89FB                  <1>       MOV   eBX,eDI                   ; BX = DRIVE #
7553 00002601 88DC                  <1>       MOV   AH,BL               ; OUTPUT DRIVE NUMBER
7554 00002603 E85FFFFFFF            <1>       CALL  NEC_OUTPUT
7555 00002608 8AA7[FD700000]        <1>       MOV   AH, [DSK_TRK+eDI]   ; GET CYLINDER NUMBER
7556 0000260E E854FFFFFF            <1>       CALL  NEC_OUTPUT
7557 00002613 E829000000            <1>       CALL  CHK_STAT_2          ; ENDING INTERRUPT AND SENSE STATUS
7558                                <1>
7559                                <1> ;-----      WAIT FOR HEAD SETTLE
7560                                <1>
7561                                <1> DO_WAIT:
7562 00002618 9C                    <1>       PUSHF                     ; SAVE STATUS
7563 00002619 E816FFFFFF            <1>       CALL  HD_WAIT                   ; WAIT FOR HEAD SETTLE TIME
7564 0000261E 9D                    <1>       POPF                      ; RESTORE STATUS
7565                                <1> RB:
7566                                <1> NEC_ERR:
7567                                <1>       ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
7568                                <1>       ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
7569 0000261F C3                    <1>       RETn                      ; RETURN TO CALLER
7570                                <1>
7571                                <1> ;--------------------------------------------------------------------------
7572                                <1> ; RECAL
7573                                <1> ;     RECALIBRATE DRIVE
7574                                <1> ;
7575                                <1> ; ON ENTRY:  DI = DRIVE #
7576                                <1> ;
7577                                <1> ; ON EXIT:   CY REFLECTS STATUS OF OPERATION.
7578                                <1> ;--------------------------------------------------------------------------
7579                                <1> RECAL:
7580 00002620 6651                  <1>       PUSH  CX
7581 00002622 B8[3E260000]          <1>       MOV   eAX, RC_BACK        ; LOAD NEC_OUTPUT ERROR
7582 00002627 50                    <1>       PUSH  eAX
7583 00002628 B407                  <1>       MOV   AH,07H              ; RECALIBRATE COMMAND
7584 0000262A E838FFFFFF            <1>       CALL  NEC_OUTPUT
7585 0000262F 89FB                  <1>       MOV   eBX,eDI                   ; BX = DRIVE #
7586 00002631 88DC                  <1>       MOV   AH,BL
7587 00002633 E82FFFFFFF            <1>       CALL  NEC_OUTPUT          ; OUTPUT THE DRIVE NUMBER
7588 00002638 E804000000            <1>       CALL  CHK_STAT_2          ; GET THE INTERRUPT AND SENSE INT STATUS
7589 0000263D 58                    <1>       POP   eAX                 ; THROW AWAY ERROR
7590                                <1> RC_BACK:
7591 0000263E 6659                  <1>       POP   CX
7592 00002640 C3                    <1>       RETn
7593                                <1>
7594                                <1> ;--------------------------------------------------------------------------
7595                                <1> ; CHK_STAT_2
7596                                <1> ;     THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
7597                                <1> ;     OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
7598                                <1> ;     INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
7599                                <1> ;
7600                                <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
7601                                <1> ;--------------------------------------------------------------------------
7602                                <1> CHK_STAT_2:
7603 00002641 B8[69260000]          <1>         MOV     eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
7604 00002646 50                    <1>       PUSH  eAX
7605 00002647 E828000000            <1>       CALL  WAIT_INT            ; WAIT FOR THE INTERRUPT
7606 0000264C 721A                  <1>       JC    short J34           ; IF ERROR, RETURN IT
7607 0000264E B408                  <1>       MOV   AH,08H              ; SENSE INTERRUPT STATUS COMMAND
7608 00002650 E812FFFFFF            <1>       CALL  NEC_OUTPUT
7609 00002655 E84A000000            <1>       CALL  RESULTS                   ; READ IN THE RESULTS
7610 0000265A 720C                  <1>       JC    short J34
7611 0000265C A0[ED700000]          <1>       MOV   AL,[NEC_STATUS]           ; GET THE FIRST STATUS BYTE
7612 00002661 2460                  <1>       AND   AL,01100000B        ; ISOLATE THE BITS
7613 00002663 3C60                  <1>       CMP   AL,01100000B        ; TEST FOR CORRECT VALUE
7614 00002665 7403                  <1>       JZ    short J35           ; IF ERROR, GO MARK IT
7615 00002667 F8                    <1>       CLC                       ; GOOD RETURN
7616                                <1> J34:
7617 00002668 58                    <1>       POP   eAX                 ; THROW AWAY ERROR RETURN
7618                                <1> CS_BACK:
7619 00002669 C3                    <1>       RETn
7620                                <1> J35:
7621 0000266A 800D[EC700000]40      <1>       OR    byte [DSKETTE_STATUS], BAD_SEEK
7622 00002671 F9                    <1>       STC                       ; ERROR RETURN CODE
7623 00002672 EBF4                  <1>       JMP   SHORT J34
7624                                <1>
7625                                <1> ;--------------------------------------------------------------------------
7626                                <1> ; WAIT_INT
7627                                <1> ;     THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
7628                                <1> ;     TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
7629                                <1> ;     IF THE DRIVE IS NOT READY.
7630                                <1> ;
7631                                <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
```

```
7632                            <1> ;---------------------------------------------------------------------------
7633                            <1>
7634                            <1> ; 17/12/2014
7635                            <1> ; 2.5 seconds waiting !
7636                            <1> ;(AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
7637                            <1> ; amount of time to wait for completion interrupt from NEC.
7638                            <1>
7639                            <1>
7640                            <1> WAIT_INT:
7641 00002674 FB               <1>       STI                     ; TURN ON INTERRUPTS, JUST IN CASE
7642 00002675 F8               <1>       CLC                     ; CLEAR TIMEOUT INDICATOR
7643                            <1>        ;MOV  BL,10            ; CLEAR THE COUNTERS
7644                            <1>        ;XOR  CX,CX            ; FOR 2 SECOND WAIT
7645                            <1>
7646                            <1>       ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
7647                            <1>       ;
7648                            <1>       ;WAIT_FOR_MEM:
7649                            <1>       ;     Waits for a bit at a specified memory location pointed
7650                            <1>       ;     to by ES:[DI] to become set.
7651                            <1>       ;INPUT:
7652                            <1>       ;     AH=Mask to test with.
7653                            <1>       ;     ES:[DI] = memory location to watch.
7654                            <1>       ;     BH:CX=Number of memory refresh periods to delay.
7655                            <1>       ;          (normally 30 microseconds per period.)
7656                            <1>
7657                            <1>       ; waiting for (max.) 2.5 secs in 30 micro units.
7658                            <1> ;     mov    cx, WAIT_FDU_INT_LO      ; 017798
7659                            <1> ;;    mov    bl, WAIT_FDU_INT_HI
7660                            <1> ;     mov    bl, WAIT_FDU_INT_HI + 1
7661                            <1>       ; 27/02/2015
7662 00002676 B986450100       <1>       mov    ecx, WAIT_FDU_INT_LH     ; 83334 (2.5 seconds)
7663                            <1> WFMS_CHECK_MEM:
7664 0000267B F605[E9700000]80  <1>       test   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
7665 00002682 7516              <1>         jnz     short J37
7666                            <1> WFMS_HI:
7667 00002684 E461              <1>       IN     AL,PORT_B  ; 061h   ; SYS1, wait for lo to hi
7668 00002686 A810              <1>       TEST   AL,010H                  ; transition on memory
7669 00002688 75FA              <1>       JNZ    SHORT WFMS_HI       ; refresh.
7670                            <1> WFMS_LO:
7671 0000268A E461              <1>       IN     AL,PORT_B           ;SYS1
7672 0000268C A810              <1>       TEST   AL,010H
7673 0000268E 74FA              <1>       JZ     SHORT WFMS_LO
7674 00002690 E2E9              <1>         LOOP    WFMS_CHECK_MEM
7675                            <1> ;WFMS_OUTER_LP:
7676                            <1> ;;    or     bl, bl              ; check outer counter
7677                            <1> ;;    jz     short J36A          ; WFMS_TIMEOUT
7678                            <1> ;     dec    bl
7679                            <1> ;     jz     short J36A
7680                            <1> ;     jmp    short WFMS_CHECK_MEM
7681                            <1>
7682                            <1>       ;17/12/2014
7683                            <1>       ;16/12/2014
7684                            <1> ;        mov     byte [wait_count], 0    ; Reset (INT 08H) counter
7685                            <1> ;J36:
7686                            <1> ;     TEST   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
7687                            <1> ;     JNZ    short J37
7688                            <1>       ;16/12/2014
7689                            <1>       ;LOOP  J36                 ; COUNT DOWN WHILE WAITING
7690                            <1>       ;DEC   BL                  ; SECOND LEVEL COUNTER
7691                            <1>       ;JNZ   short J36
7692                            <1> ;       cmp     byte [wait_count], 46   ; (46/18.2 seconds)
7693                            <1> ;     jb     short J36
7694                            <1>
7695                            <1> ;WFMS_TIMEOUT:
7696                            <1> ;J36A:
7697 00002692 800D[EC700000]80  <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
7698 00002699 F9                <1>       STC                        ; ERROR RETURN
7699                            <1> J37:
7700 0000269A 9C                <1>       PUSHF                      ; SAVE CURRENT CARRY
7701 0000269B 8025[E9700000]7F  <1>       AND    byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
7702 000026A2 9D                <1>       POPF                       ; RECOVER CARRY
7703 000026A3 C3                <1>       RETn                       ; GOOD RETURN CODE
7704                            <1>
7705                            <1> ;---------------------------------------------------------------------------
7706                            <1> ; RESULTS
7707                            <1> ;     THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
7708                            <1> ;     FOLLOWING AN INTERRUPT.
7709                            <1> ;
7710                            <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
7711                            <1> ;           AX,BX,CX,DX DESTROYED
7712                            <1> ;---------------------------------------------------------------------------
7713                            <1> RESULTS:
7714 000026A4 57                <1>       PUSH   eDI
7715 000026A5 BF[ED700000]      <1>       MOV    eDI, NEC_STATUS          ; POINTER TO DATA AREA
7716 000026AA B307              <1>       MOV    BL,7                ; MAX STATUS BYTES
7717 000026AC 66BAF403          <1>       MOV    DX,03F4H            ; STATUS PORT
7718                            <1>
7719                            <1> ;-----      WAIT FOR REQUEST FOR MASTER
7720                            <1>
7721                            <1> _R10:
7722                            <1>       ; 16/12/2014
7723                            <1>       ; wait for (max) 0.5 seconds
7724                            <1>       ;MOV   BH,2                ; HIGH ORDER COUNTER
7725                            <1>       ;XOR   CX,CX               ; COUNTER
7726                            <1>
7727                            <1>       ;Time to wait while waiting for each byte of NEC results = .5
7728                            <1>       ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
7729                            <1>       ; 27/02/2015
7730 000026B0 B91B410000        <1>       mov    ecx, WAIT_FDU_RESULTS_LH ; 16667
7731                            <1>       ;mov   cx, WAIT_FDU_RESULTS_LO  ; 16667
7732                            <1>       ;mov   bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
7733                            <1>
7734                            <1> WFPSR_OUTER_LP:
7735                            <1>       ;
7736                            <1> WFPSR_CHECK_PORT:
```

```
7737                                    <1> J39:                            ; WAIT FOR MASTER
7738 000026B5 EC                        <1>       IN    AL,DX               ; GET STATUS
7739 000026B6 24C0                      <1>       AND   AL,11000000B        ; KEEP ONLY STATUS AND DIRECTION
7740 000026B8 3CC0                      <1>       CMP   AL,11000000B        ; STATUS 1 AND DIRECTION 1 ?
7741 000026BA 7418                      <1>       JZ    short J42           ; STATUS AND DIRECTION OK
7742                                    <1> WFPSR_HI:
7743 000026BC E461                      <1>       IN    AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
7744 000026BE A810                      <1>       TEST  AL,010H                   ; transition on memory
7745 000026C0 75FA                      <1>       JNZ   SHORT WFPSR_HI            ; refresh.
7746                                    <1> WFPSR_LO:
7747 000026C2 E461                      <1>       IN    AL, PORT_B          ; SYS1
7748 000026C4 A810                      <1>       TEST  AL,010H
7749 000026C6 74FA                      <1>       JZ    SHORT WFPSR_LO
7750 000026C8 E2EB                      <1>         LOOP    WFPSR_CHECK_PORT
7751                                    <1>       ;; 27/02/2015
7752                                    <1>       ;;dec  bh
7753                                    <1>       ;;jnz  short WFPSR_OUTER_LP
7754                                    <1>       ;jmp   short WFPSR_TIMEOUT ; fail
7755                                    <1>
7756                                    <1>       ;;mov  byte [wait_count], 0
7757                                    <1> ;J39:                           ; WAIT FOR MASTER
7758                                    <1> ;     IN    AL,DX               ; GET STATUS
7759                                    <1> ;     AND   AL,11000000B        ; KEEP ONLY STATUS AND DIRECTION
7760                                    <1> ;     CMP   AL,11000000B        ; STATUS 1 AND DIRECTION 1 ?
7761                                    <1> ;     JZ    short J42           ; STATUS AND DIRECTION OK
7762                                    <1>       ;LOOP  J39                ; LOOP TILL TIMEOUT
7763                                    <1>       ;DEC   BH                 ; DECREMENT HIGH ORDER COUNTER
7764                                    <1>       ;JNZ   short J39          ; REPEAT TILL DELAY DONE
7765                                    <1>       ;
7766                                    <1>       ;;cmp  byte [wait_count], 10  ; (10/18.2 seconds)
7767                                    <1>       ;;jb   short J39
7768                                    <1>
7769                                    <1> ;WFPSR_TIMEOUT:
7770 000026CA 800D[EC700000]80          <1>       OR    byte [DSKETTE_STATUS],TIME_OUT
7771 000026D1 F9                        <1>       STC                       ; SET ERROR RETURN
7772 000026D2 EB29                      <1>       JMP   SHORT POPRES        ; POP REGISTERS AND RETURN
7773                                    <1>
7774                                    <1> ;-----      READ IN THE STATUS
7775                                    <1>
7776                                    <1> J42:
7777 000026D4 EB00                      <1>       JMP   $+2                 ; I/O DELAY
7778 000026D6 6642                      <1>       INC   DX                  ; POINT AT DATA PORT
7779 000026D8 EC                        <1>       IN    AL,DX               ; GET THE DATA
7780                                    <1>       ; 16/12/2014
7781                                    <1>       NEWIODELAY
7782 000026D9 E6EB                      <2>  out 0ebh,al
7783 000026DB 8807                      <1>         MOV     [eDI],AL              ; STORE THE BYTE
7784 000026DD 47                        <1>       INC   eDI                 ; INCREMENT THE POINTER
7785                                    <1>       ; 16/12/2014
7786                                    <1> ;     push   cx
7787                                    <1> ;     mov    cx, 30
7788                                    <1> ;wdw2:
7789                                    <1> ;     NEWIODELAY
7790                                    <1> ;     loop   wdw2
7791                                    <1> ;     pop    cx
7792                                    <1>
7793 000026DE B903000000                <1>       MOV   eCX,3               ; MINIMUM 24 MICROSECONDS FOR NEC
7794 000026E3 E801EFFFFF                <1>       CALL  WAITF               ; WAIT 30 TO 45 MICROSECONDS
7795 000026E8 664A                      <1>       DEC   DX                  ; POINT AT STATUS PORT
7796 000026EA EC                        <1>       IN    AL,DX               ; GET STATUS
7797                                    <1>       ; 16/12/2014
7798                                    <1>       NEWIODELAY
7799 000026EB E6EB                      <2>  out 0ebh,al
7800                                    <1>       ;
7801 000026ED A810                      <1>       TEST  AL,00010000B        ; TEST FOR NEC STILL BUSY
7802 000026EF 740C                      <1>       JZ    short POPRES        ; RESULTS DONE ?
7803                                    <1>
7804 000026F1 FECB                      <1>       DEC   BL                  ; DECREMENT THE STATUS COUNTER
7805 000026F3 75BB                      <1>         JNZ     short _R10              ; GO BACK FOR MORE
7806 000026F5 800D[EC700000]20          <1>       OR    byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
7807 000026FC F9                        <1>       STC                       ; SET ERROR FLAG
7808                                    <1>
7809                                    <1> ;-----      RESULT OPERATION IS DONE
7810                                    <1> POPRES:
7811 000026FD 5F                        <1>       POP   eDI
7812 000026FE C3                        <1>       RETn                      ; RETURN WITH CARRY SET
7813                                    <1>
7814                                    <1> ;--------------------------------------------------------------------------
7815                                    <1> ; READ_DSKCHNG
7816                                    <1> ;     READS THE STATE OF THE DISK CHANGE LINE.
7817                                    <1> ;
7818                                    <1> ; ON ENTRY:  DI = DRIVE #
7819                                    <1> ;
7820                                    <1> ; ON EXIT:   DI = DRIVE #
7821                                    <1> ;            ZF = 0 : DISK CHANGE LINE INACTIVE
7822                                    <1> ;            ZF = 1 : DISK CHANGE LINE ACTIVE
7823                                    <1> ;            AX,CX,DX DESTROYED
7824                                    <1> ;--------------------------------------------------------------------------
7825                                    <1> READ_DSKCHNG:
7826 000026FF E8A2FDFFFF                <1>       CALL  MOTOR_ON            ; TURN ON THE MOTOR IF OFF
7827 00002704 66BAF703                  <1>       MOV   DX,03F7H            ; ADDRESS DIGITAL INPUT REGISTER
7828 00002708 EC                        <1>       IN    AL,DX               ; INPUT DIGITAL INPUT REGISTER
7829 00002709 A880                      <1>       TEST  AL,DSK_CHG          ; CHECK FOR DISK CHANGE LINE ACTIVE
7830 0000270B C3                        <1>       RETn                      ; RETURN TO CALLER WITH ZERO FLAG SET
7831                                    <1>
7832                                    <1> ;--------------------------------------------------------------------------
7833                                    <1> ; DRIVE_DET
7834                                    <1> ;     DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
7835                                    <1> ;     UPDATES STATE INFORMATION ACCORDINGLY.
7836                                    <1> ; ON ENTRY:  DI = DRIVE #
7837                                    <1> ;--------------------------------------------------------------------------
7838                                    <1> DRIVE_DET:
7839 0000270C E895FDFFFF                <1>       CALL  MOTOR_ON            ; TURN ON MOTOR IF NOT ALREADY ON
7840 00002711 E80AFFFFFF                <1>       CALL  RECAL               ; RECALIBRATE DRIVE
7841 00002716 7251                      <1>       JC    short DD_BAC        ; ASSUME NO DRIVE PRESENT
```

```
7842 00002718 B530                <1>       MOV    CH,TRK_SLAP         ; SEEK TO TRACK 48
7843 0000271A E882FEFFFF          <1>       CALL   SEEK
7844 0000271F 7248                <1>       JC     short DD_BAC         ; ERROR NO DRIVE
7845 00002721 B50B                <1>       MOV    CH,QUIET_SEEK+1           ; SEEK TO TRACK 10
7846                              <1> SK_GIN:
7847 00002723 FECD                <1>       DEC    CH                   ; DECREMENT TO NEXT TRACK
7848 00002725 6651                <1>       PUSH   CX                   ; SAVE TRACK
7849 00002727 E875FEFFFF          <1>       CALL   SEEK
7850 0000272C 723C                <1>       JC     short POP_BAC        ; POP AND RETURN
7851 0000272E B8[6A270000]        <1>       MOV    eAX, POP_BAC         ; LOAD NEC OUTPUT ERROR ADDRESS
7852 00002733 50                  <1>       PUSH   eAX
7853 00002734 B404                <1>       MOV    AH,SENSE_DRV_ST           ; SENSE DRIVE STATUS COMMAND BYTE
7854 00002736 E82CFEFFFF          <1>       CALL   NEC_OUTPUT           ; OUTPUT TO NEC
7855 0000273B 6689F8              <1>       MOV    AX,DI                ; AL = DRIVE
7856 0000273E 88C4                <1>       MOV    AH,AL                ; AH = DRIVE
7857 00002740 E822FEFFFF          <1>       CALL   NEC_OUTPUT           ; OUTPUT TO NEC
7858 00002745 E85AFFFFFF          <1>       CALL   RESULTS                    ; GO GET STATUS
7859 0000274A 58                  <1>       POP    eAX                  ; THROW AWAY ERROR ADDRESS
7860 0000274B 6659                <1>       POP    CX                   ; RESTORE TRACK
7861 0000274D F605[ED700000]10    <1>       TEST   byte [NEC_STATUS], HOME    ; TRACK 0 ?
7862 00002754 74CD                <1>       JZ     short SK_GIN         ; GO TILL TRACK 0
7863 00002756 08ED                <1>       OR     CH,CH                ; IS HOME AT TRACK 0
7864 00002758 7408                <1>       JZ     short IS_80          ; MUST BE 80 TRACK DRIVE
7865                              <1>
7866                              <1> ;     DRIVE IS A 360; SET DRIVE TO DETERMINED;
7867                              <1> ;     SET MEDIA TO DETERMINED AT RATE 250.
7868                              <1>
7869 0000275A 808F[F9700000]94    <1>       OR     byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
7870 00002761 C3                  <1>       RETn                        ; ALL INFORMATION SET
7871                              <1> IS_80:
7872 00002762 808F[F9700000]01    <1>       OR     byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
7873                              <1> DD_BAC:
7874 00002769 C3                  <1>       RETn
7875                              <1> POP_BAC:
7876 0000276A 6659                <1>       POP    CX                   ; THROW AWAY
7877 0000276C C3                  <1>       RETn
7878                              <1>
7879                              <1> fdc_int:
7880                              <1>         ; 30/07/2015
7881                              <1>         ; 16/02/2015
7882                              <1> ;int_0Eh: ; 11/12/2014
7883                              <1>
7884                              <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) -------------------------------------
7885                              <1> ; DISK_INT
7886                              <1> ;     THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
7887                              <1> ;
7888                              <1> ; ON EXIT:  THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
7889                              <1> ;--------------------------------------------------------------------------------
7890                              <1> DISK_INT_1:
7891                              <1>
7892 0000276D 6650                <1>       PUSH   AX                   ; SAVE WORK REGISTER
7893 0000276F 1E                  <1>       push   ds
7894 00002770 66B81000            <1>       mov    ax, KDATA
7895 00002774 8ED8                <1>       mov    ds, ax
7896 00002776 800D[E9700000]80    <1>         OR      byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
7897 0000277D B020                <1>       MOV     AL,EOI                   ; END OF INTERRUPT MARKER
7898 0000277F E620                <1>       OUT    INTA00,AL            ; INTERRUPT CONTROL PORT
7899 00002781 1F                  <1>       pop    ds
7900 00002782 6658                <1>       POP    AX                   ; RECOVER REGISTER
7901 00002784 CF                  <1>       IRET                        ; RETURN FROM INTERRUPT
7902                              <1>
7903                              <1> ;--------------------------------------------------------------------------------
7904                              <1> ; DSKETTE_SETUP
7905                              <1> ;     THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
7906                              <1> ;     DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
7907                              <1> ;--------------------------------------------------------------------------------
7908                              <1> DSKETTE_SETUP:
7909                              <1>       ;PUSH  AX                   ; SAVE REGISTERS
7910                              <1>       ;PUSH  BX
7911                              <1>       ;PUSH  CX
7912 00002785 52                  <1>       PUSH   eDX
7913                              <1>       ;PUSH  DI
7914                              <1>       ;;PUSH DS
7915                              <1>       ; 14/12/2014
7916                              <1>       ;mov   word [DISK_POINTER], MD_TBL6
7917                              <1>       ;mov   [DISK_POINTER+2], cs
7918                              <1>       ;
7919                              <1>       ;OR    byte [RTC_WAIT_FLAG], 1   ; NO RTC WAIT, FORCE USE OF LOOP
7920 00002786 31FF                <1>       XOR    eDI,eDI                   ; INITIALIZE DRIVE POINTER
7921 00002788 66C705[F9700000]00- <1>       MOV    WORD [DSK_STATE],0  ; INITIALIZE STATES
7922 00002790 00                  <1>
7923 00002791 8025[F4700000]33    <1>       AND    byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
7924 00002798 800D[F4700000]C0    <1>       OR     byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
7925 0000279F C605[E9700000]00    <1>       MOV    byte [SEEK_STATUS],0      ; INDICATE RECALIBRATE NEEDED
7926 000027A6 C605[EB700000]00    <1>       MOV    byte [MOTOR_COUNT],0      ; INITIALIZE MOTOR COUNT
7927 000027AD C605[EA700000]00    <1>       MOV    byte [MOTOR_STATUS],0     ; INITIALIZE DRIVES TO OFF STATE
7928 000027B4 C605[EC700000]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; NO ERRORS
7929                              <1>       ;
7930                              <1>       ; 28/02/2015
7931                              <1>       ;mov   word [cfd], 100h
7932 000027BB E85FF2FFFF          <1>       call   DSK_RESET
7933 000027C0 5A                  <1>       pop    edx
7934 000027C1 C3                  <1>       retn
7935                              <1>
7936                              <1> ;SUP0:
7937                              <1> ;     CALL   DRIVE_DET           ; DETERMINE DRIVE
7938                              <1> ;     CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
7939                              <1> ;     ; 02/01/2015
7940                              <1> ;     ;INC   DI                  ; POINT TO NEXT DRIVE
7941                              <1> ;     ;CMP   DI,MAX_DRV          ; SEE IF DONE
7942                              <1> ;     ;JNZ   short SUP0          ; REPEAT FOR EACH ORIVE
7943                              <1> ;       cmp     byte [fd1_type], 0
7944                              <1> ;     jna    short sup1
7945                              <1> ;     or     di, di
7946                              <1> ;     jnz    short sup1
```

```
7947                                    <1> ;     inc    di
7948                                    <1> ;       jmp     short SUP0
7949                                    <1> ;sup1:
7950                                    <1> ;     MOV    byte [SEEK_STATUS],0      ; FORCE RECALIBRATE
7951                                    <1> ;     ;AND   byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
7952                                    <1> ;     CALL   SETUP_END          ; VARIOUS CLEANUPS
7953                                    <1> ;     ;;POP  DS                 ; RESTORE CALLERS REGISTERS
7954                                    <1> ;     ;POP   DI
7955                                    <1> ;     POP    eDX
7956                                    <1> ;     ;POP   CX
7957                                    <1> ;     ;POP   BX
7958                                    <1> ;     ;POP   AX
7959                                    <1> ;     RETn
7960                                    <1>
7961                                    <1> ;////////////////////////////////////////////////
7962                                    <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
7963                                    <1> ;
7964                                    <1>
7965                                    <1> int13h: ; 21/02/2015
7966 000027C2 9C                       <1>       pushfd
7967 000027C3 0E                       <1>       push   cs
7968 000027C4 E858000000               <1>       call   DISK_IO
7969 000027C9 C3                       <1>       retn
7970                                    <1>
7971                                    <1> ;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
7972                                    <1> ;////////////////////////////////////////////////////////////////////
7973                                    <1>
7974                                    <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
7975                                    <1> ; 23/02/2015
7976                                    <1> ; 21/02/2015 (unix386.s)
7977                                    <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
7978                                    <1> ;
7979                                    <1> ; Original Source Code:
7980                                    <1> ; DISK ----- 09/25/85 FIXED DISK BIOS
7981                                    <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
7982                                    <1> ;
7983                                    <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
7984                                    <1> ;            Source Code - ATORGS.ASM, AHDSK.ASM
7985                                    <1> ;
7986                                    <1>
7987                                    <1>
7988                                    <1> ;The wait for controller to be not busy is 10 seconds.
7989                                    <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
7990                                    <1> ;;WAIT_HDU_CTLR_BUSY_LO   equ    1615h
7991                                    <1> ;;WAIT_HDU_CTLR_BUSY_HI   equ      05h
7992                                    <1> WAIT_HDU_CTRL_BUSY_LH     equ    51615h  ;21/02/2015
7993                                    <1>
7994                                    <1> ;The wait for controller to issue completion interrupt is 10 seconds.
7995                                    <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
7996                                    <1> ;;WAIT_HDU_INT_LO  equ    1615h
7997                                    <1> ;;WAIT_HDU_INT_HI  equ      05h
7998                                    <1> WAIT_HDU_INT_LH           equ    51615h ; 21/02/2015
7999                                    <1>
8000                                    <1> ;The wait for Data request on read and write longs is
8001                                    <1> ;2000 us. (?)
8002                                    <1> ;;WAIT_HDU_DRQ_LO  equ    1000   ; 03E8h
8003                                    <1> ;;WAIT_HDU_DRQ_HI  equ    0
8004                                    <1> WAIT_HDU_DRQ_LH           equ    1000   ; 21/02/2015
8005                                    <1>
8006                                    <1> ; Port 61h (PORT_B)
8007                                    <1> SYS1        equ    61h    ; PORT_B  (diskette.inc)
8008                                    <1>
8009                                    <1> ; 23/12/2014
8010                                    <1> %define CMD_BLOCK       eBP-8  ; 21/02/2015
8011                                    <1>
8012                                    <1>
8013                                    <1> ;--- INT 13H -----------------------------------------------------------------------
8014                                    <1> ;                                                                              :
8015                                    <1> ; FIXED DISK I/O INTERFACE                                                       :
8016                                    <1> ;                                                                              :
8017                                    <1> ;     THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH              :
8018                                    <1> ;     THE IBM FIXED DISK CONTROLLER.                                           :
8019                                    <1> ;                                                                              :
8020                                    <1> ;     THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH              :
8021                                    <1> ;     SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT     IN             :
8022                                    <1> ;     THESE  LISTINGS  ARE  INCLUDED    ONLY FOR  COMPLETENESS,               :
8023                                    <1> ;     NOT  FOR  REFERENCE.  APPLICATIONS   WHICH  REFERENCE  ANY              :
8024                                    <1> ;     ABSOLUTE  ADDRESSES  WITHIN  THE  CODE SEGMENTS  OF  BIOS               :
8025                                    <1> ;     VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                          :
8026                                    <1> ;                                                                              :
8027                                    <1> ;------------------------------------------------------------------------------:
8028                                    <1> ;                                                                              :
8029                                    <1> ; INPUT  (AH)= HEX COMMAND VALUE                                               :
8030                                    <1> ;                                                                              :
8031                                    <1> ;     (AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                          :
8032                                    <1> ;     (AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)          :
8033                                    <1> ;                NOTE: DL < 80H - DISKETTE                                   :
8034                                    <1> ;                     DL > 80H - DISK                                        :
8035                                    <1> ;     (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                          :
8036                                    <1> ;     (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                         :
8037                                    <1> ;     (AH)= 04H  VERIFY THE DESIRED SECTORS                                    :
8038                                    <1> ;     (AH)= 05H  FORMAT THE DESIRED TRACK                                      :
8039                                    <1> ;     (AH)= 06H  UNUSED                                                        :
8040                                    <1> ;     (AH)= 07H  UNUSED                                                        :
8041                                    <1> ;     (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                             :
8042                                    <1> ;     (AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                         :
8043                                    <1> ;                INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0                :
8044                                    <1> ;                INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1                :
8045                                    <1> ;     (AH)= 0AH  READ LONG                                                     :
8046                                    <1> ;     (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC)   :
8047                                    <1> ;     (AH)= 0CH  SEEK                                                          :
8048                                    <1> ;     (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                                 :
8049                                    <1> ;     (AH)= 0EH  UNUSED                                                        :
8050                                    <1> ;     (AH)= 0FH  UNUSED                                                        :
8051                                    <1> ;     (AH)= 10H  TEST DRIVE READY                                              :
```

```
8052                                    <1> ;     (AH)= 11H  RECALIBRATE                                        :
8053                                    <1> ;     (AH)= 12H  UNUSED                                          :
8054                                    <1> ;     (AH)= 13H  UNUSED                                          :
8055                                    <1> ;     (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                     :
8056                                    <1> ;     (AH)= 15H  READ DASD TYPE                                  :
8057                                    <1> ;                                                                :
8058                                    <1> ;--------------------------------------------------------------------------
8059                                    <1> ;                                                                :
8060                                    <1> ;     REGISTERS USED FOR FIXED DISK OPERATIONS                      :
8061                                    <1> ;                                                                :
8062                                    <1> ;           (DL)   -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
8063                                    <1> ;           (DH)   -  HEAD NUMBER         (0-15 ALLOWED, NOT VALUE CHECKED)  :
8064                                    <1> ;           (CH)   -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
8065                                    <1> ;           (CL)   -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)       :
8066                                    <1> ;                                                                :
8067                                    <1> ;                     NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED    :
8068                                    <1> ;                          IN THE HIGH 2 BITS OF THE CL REGISTER       :
8069                                    <1> ;                          (10 BITS TOTAL)                           :
8070                                    <1> ;                                                                :
8071                                    <1> ;           (AL)   -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,   :
8072                                    <1> ;                                      FOR READ/WRITE LONG 1-79H)     :
8073                                    <1> ;                                                                :
8074                                    <1> ;           (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,           :
8075                                    <1> ;                      (NOT REQUIRED FOR VERIFY)                 :
8076                                    <1> ;                                                                :
8077                                    <1> ;           FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST    :
8078                                    <1> ;                      2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
8079                                    <1> ;                      F = 00H FOR A GOOD SECTOR                 :
8080                                    <1> ;                          80H FOR A BAD SECTOR                  :
8081                                    <1> ;                      N = SECTOR NUMBER                         :
8082                                    <1> ;                      FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK            :
8083                                    <1> ;                      THE TABLE SHOULD BE:                      :
8084                                    <1> ;                                                                :
8085                                    <1> ;                          DB     00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH    :
8086                                    <1> ;                          DB     00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH    :
8087                                    <1> ;                          DB     00H,07H,00H,10H,00H,08H,00H,11H,00H,09H          :
8088                                    <1> ;                                                                :
8089                                    <1> ;--------------------------------------------------------------------------
8090                                    <1>
8091                                    <1> ;--------------------------------------------------------------------------
8092                                    <1> ; OUTPUT                                                         :
8093                                    <1> ;     AH = STATUS OF CURRENT OPERATION                           :
8094                                    <1> ;          STATUS BITS ARE DEFINED IN THE EQUATES BELOW                :
8095                                    <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)                  :
8096                                    <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)                 :
8097                                    <1> ;                                                                :
8098                                    <1> ;     NOTE:  ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE     :
8099                                    <1> ;            ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA     :
8100                                    <1> ;            IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN     :
8101                                    <1> ;            ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE     :
8102                                    <1> ;            FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS     :
8103                                    <1> ;            REWRITTEN.                                          :
8104                                    <1> ;                                                                :
8105                                    <1> ;     IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),               :
8106                                    <1> ;        INPUT:                                                  :
8107                                    <1> ;          (DL) = DRIVE NUMBER                                   :
8108                                    <1> ;        OUTPUT:                                                 :
8109                                    <1> ;          (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
8110                                    <1> ;               (CONTROLLER CARD ZERO TALLY ONLY)                :
8111                                    <1> ;          (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER          :
8112                                    <1> ;          (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER            :
8113                                    <1> ;          (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER        :
8114                                    <1> ;               AND CYLINDER NUMBER HIGH BITS                    :
8115                                    <1> ;                                                                :
8116                                    <1> ;     IF READ DASD TYPE WAS REQUESTED,                           :
8117                                    <1> ;                                                                :
8118                                    <1> ;     AH = 0 - NOT PRESENT                                       :
8119                                    <1> ;          1 - DISKETTE - NO CHANGE LINE AVAILABLE               :
8120                                    <1> ;          2 - DISKETTE - CHANGE LINE AVAILABLE                  :
8121                                    <1> ;          3 - FIXED DISK                                        :
8122                                    <1> ;                                                                :
8123                                    <1> ;     CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                 :
8124                                    <1> ;                                                                :
8125                                    <1> ;     REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN     :
8126                                    <1> ;     INFORMATION.                                               :
8127                                    <1> ;                                                                :
8128                                    <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE     :
8129                                    <1> ;           ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.       :
8130                                    <1> ;                                                                :
8131                                    <1> ;--------------------------------------------------------------------------
8132                                    <1>
8133                                    <1> SENSE_FAIL   EQU    0FFH         ; NOT IMPLEMENTED
8134                                    <1> NO_ERR       EQU    0E0H         ; STATUS ERROR/ERROR REGISTER=0
8135                                    <1> WRITE_FAULT  EQU    0CCH         ; WRITE FAULT ON SELECTED DRIVE
8136                                    <1> UNDEF_ERR    EQU    0BBH         ; UNDEFINED ERROR OCCURRED
8137                                    <1> NOT_RDY      EQU    0AAH         ; DRIVE NOT READY
8138                                    <1> TIME_OUT     EQU    80H          ; ATTACHMENT FAILED TO RESPOND
8139                                    <1> BAD_SEEK     EQU    40H          ; SEEK OPERATION FAILED
8140                                    <1> BAD_CNTLR    EQU    20H          ; CONTROLLER HAS FAILED
8141                                    <1> DATA_CORRECTED     EQU    11H          ; ECC CORRECTED DATA ERROR
8142                                    <1> BAD_ECC      EQU    10H          ; BAD ECC ON DISK READ
8143                                    <1> BAD_TRACK    EQU    0BH          ; NOT IMPLEMENTED
8144                                    <1> BAD_SECTOR   EQU    0AH          ; BAD SECTOR FLAG DETECTED
8145                                    <1> ;DMA_BOUNDARY      EQU    09H          ; DATA EXTENDS TOO FAR
8146                                    <1> INIT_FAIL    EQU    07H          ; DRIVE PARAMETER ACTIVITY FAILED
8147                                    <1> BAD_RESET    EQU    05H          ; RESET FAILED
8148                                    <1> ;RECORD_NOT_FND    EQU    04H          ; REQUESTED SECTOR NOT FOUND
8149                                    <1> ;BAD_ADDR_MARK     EQU    02H          ; ADDRESS MARK NOT FOUND
8150                                    <1> ;BAD_CMD     EQU    01H          ; BAD COMMAND PASSED TO DISK I/O
8151                                    <1>
8152                                    <1> ;--------------------------------------------------
8153                                    <1> ;                                              :
8154                                    <1> ; FIXED DISK PARAMETER TABLE                        :
8155                                    <1> ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:  :
8156                                    <1> ;                                              :
```

```
8157                            <1> ;  +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS        :
8158                            <1> ;  +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS            :
8159                            <1> ;  +3 (1 WORD) - NOT USED/SEE PC-XT                 :
8160                            <1> ;  +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
8161                            <1> ;  +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH      :
8162                            <1> ;  +8 (1 BYTE) - CONTROL BYTE               :
8163                            <1> ;                BIT   7 DISABLE RETRIES -OR-   :
8164                            <1> ;                BIT   6 DISABLE RETRIES        :
8165                            <1> ;                BIT   3 MORE THAN 8 HEADS            :
8166                            <1> ;  +9 (3 BYTES)- NOT USED/SEE PC-XT                 :
8167                            <1> ; +12 (1 WORD) - LANDING ZONE                :
8168                            <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK            :
8169                            <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE            :
8170                            <1> ;                                           :
8171                            <1> ;      - TO DYNAMICALLY DEFINE A SET OF PARAMETERS  :
8172                            <1> ;        BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
8173                            <1> ;        THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
8174                            <1> ;        FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.  :
8175                            <1> ;                                           :
8176                            <1> ;-----------------------------------------------------
8177                            <1>
8178                            <1> ;-----------------------------------------------------
8179                            <1> ;                                           :
8180                            <1> ; HARDWARE SPECIFIC VALUES                          :
8181                            <1> ;                                           :
8182                            <1> ;  -  CONTROLLER I/O PORT                   :
8183                            <1> ;                                           :
8184                            <1> ;     > WHEN READ FROM:                     :
8185                            <1> ;     HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)      :
8186                            <1> ;     HF_PORT+1 - GET ERROR REGISTER                :
8187                            <1> ;     HF_PORT+2 - GET SECTOR COUNT                  :
8188                            <1> ;     HF_PORT+3 - GET SECTOR NUMBER                 :
8189                            <1> ;     HF_PORT+4 - GET CYLINDER LOW                  :
8190                            <1> ;     HF_PORT+5 - GET CYLINDER HIGH (2 BITS)        :
8191                            <1> ;     HF_PORT+6 - GET SIZE/DRIVE/HEAD         :
8192                            <1> ;     HF_PORT+7 - GET STATUS REGISTER         :
8193                            <1> ;                                           :
8194                            <1> ;     > WHEN WRITTEN TO:                    :
8195                            <1> ;     HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
8196                            <1> ;     HF_PORT+1 - SET PRECOMPENSATION CYLINDER      :
8197                            <1> ;     HF_PORT+2 - SET SECTOR COUNT                  :
8198                            <1> ;     HF_PORT+3 - SET SECTOR NUMBER                 :
8199                            <1> ;     HF_PORT+4 - SET CYLINDER LOW                  :
8200                            <1> ;     HF_PORT+5 - SET CYLINDER HIGH (2 BITS)        :
8201                            <1> ;     HF_PORT+6 - SET SIZE/DRIVE/HEAD         :
8202                            <1> ;     HF_PORT+7 - SET COMMAND REGISTER        :
8203                            <1> ;                                           :
8204                            <1> ;-----------------------------------------------------
8205                            <1>
8206                            <1> ;HF_PORT     EQU    01F0H  ; DISK PORT
8207                            <1> ;HF1_PORT    equ    0170h
8208                            <1> ;HF_REG_PORT EQU    03F6H
8209                            <1> ;HF1_REG_PORT       equ    0376h
8210                            <1>
8211                            <1> HDC1_BASEPORT       equ    1F0h
8212                            <1> HDC2_BASEPORT       equ    170h
8213                            <1>
8214                            <1> align 2
8215                            <1>
8216                            <1> ;-----              STATUS REGISTER
8217                            <1>
8218                            <1> ST_ERROR     EQU    00000001B    ;
8219                            <1> ST_INDEX     EQU    00000010B    ;
8220                            <1> ST_CORRCTD   EQU    00000100B    ; ECC CORRECTION SUCCESSFUL
8221                            <1> ST_DRQ              EQU    00001000B    ;
8222                            <1> ST_SEEK_COMPL       EQU    00010000B    ; SEEK COMPLETE
8223                            <1> ST_WRT_FLT   EQU    00100000B    ; WRITE FAULT
8224                            <1> ST_READY     EQU    01000000B    ;
8225                            <1> ST_BUSY      EQU    10000000B    ;
8226                            <1>
8227                            <1> ;-----              ERROR REGISTER
8228                            <1>
8229                            <1> ERR_DAM      EQU    00000001B    ; DATA ADDRESS MARK NOT FOUND
8230                            <1> ERR_TRK_0    EQU    00000010B    ; TRACK 0 NOT FOUND ON RECAL
8231                            <1> ERR_ABORT    EQU    00000100B    ; ABORTED COMMAND
8232                            <1> ;            EQU    00001000B    ; NOT USED
8233                            <1> ERR_ID               EQU    00010000B    ; ID NOT FOUND
8234                            <1> ;            EQU    00100000B    ; NOT USED
8235                            <1> ERR_DATA_ECC EQU    01000000B
8236                            <1> ERR_BAD_BLOCK       EQU    10000000B
8237                            <1>
8238                            <1>
8239                            <1> RECAL_CMD    EQU    00010000B    ; DRIVE RECAL(10H)
8240                            <1> READ_CMD     EQU    00100000B    ;      READ   (20H)
8241                            <1> WRITE_CMD    EQU    00110000B    ;      WRITE  (30H)
8242                            <1> VERIFY_CMD   EQU    01000000B    ;      VERIFY (40H)
8243                            <1> FMTTRK_CMD   EQU    01010000B    ; FORMAT TRACK      (50H)
8244                            <1> INIT_CMD     EQU    01100000B    ;   INITIALIZE      (60H)
8245                            <1> SEEK_CMD     EQU    01110000B    ;      SEEK   (70H)
8246                            <1> DIAG_CMD     EQU    10010000B    ; DIAGNOSTIC (90H)
8247                            <1> SET_PARM_CMD EQU    10010001B    ; DRIVE PARMS(91H)
8248                            <1> NO_RETRIES   EQU    00000001B    ; CHD MODIFIER      (01H)
8249                            <1> ECC_MODE     EQU    00000010B    ; CMD MODIFIER      (02H)
8250                            <1> BUFFER_MODE  EQU    00001000B    ; CMD MODIFIER      (08H)
8251                            <1>
8252                            <1> ;MAX_FILE    EQU    2
8253                            <1> ;S_MAX_FILE  EQU    2
8254                            <1> MAX_FILE     equ    4            ; 22/12/2014
8255                            <1> S_MAX_FILE   equ    4            ; 22/12/2014
8256                            <1>
8257                            <1> DELAY_1      EQU    25H          ; DELAY FOR OPERATION COMPLETE
8258                            <1> DELAY_2      EQU    0600H        ; DELAY FOR READY
8259                            <1> DELAY_3      EQU    0100H        ; DELAY FOR DATA REQUEST
8260                            <1>
8261                            <1> HF_FAIL      EQU    08H          ; CMOS FLAG IN BYTE 0EH
```

```
8262                                    <1>
8263                                    <1> ;-----              COMMAND BLOCK REFERENCE
8264                                    <1>
8265                                    <1> ;CMD_BLOCK      EQU     BP-8            ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
8266                                    <1>                                 ;  (BP) POINTS TO COMMAND BLOCK TAIL
8267                                    <1>                                 ;      AS DEFINED BY THE "ENTER" PARMS
8268                                    <1> ; 19/12/2014
8269                                    <1> ORG_VECTOR   equ    4*13h       ; INT 13h vector
8270                                    <1> DISK_VECTOR  equ    4*40h       ; INT 40h vector (for floppy disks)
8271                                    <1> ;HDISK_INT   equ    4*76h       ; Primary HDC - Hardware interrupt (IRQ14)
8272                                    <1> ;HDISK_INT1  equ    4*76h       ; Primary HDC - Hardware interrupt (IRQ14)
8273                                    <1> ;HDISK_INT2  equ    4*77h       ; Secondary HDC - Hardware interrupt (IRQ15)
8274                                    <1> ;HF_TBL_VEC  equ    4*41h       ; Pointer to 1st fixed disk parameter table
8275                                    <1> ;HF1_TBL_VEC equ    4*46h       ; Pointer to 2nd fixed disk parameter table
8276                                    <1>
8277                                    <1> align 2
8278                                    <1>
8279                                    <1> ;--------------------------------------------------------------
8280                                    <1> ; FIXED DISK I/O SETUP                                      :
8281                                    <1> ;                                                           :
8282                                    <1> ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK         :
8283                                    <1> ;  -  PERFORM POWER ON DIAGNOSTICS                          :
8284                                    <1> ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED    :
8285                                    <1> ;                                                           :
8286                                    <1> ;--------------------------------------------------------------
8287                                    <1>
8288                                    <1> DISK_SETUP:
8289                                    <1>       ;CLI
8290                                    <1>       ;;MOV  AX,ABS0                ; GET ABSOLUTE SEGMENT
8291                                    <1>       ;xor   ax,ax
8292                                    <1>       ;MOV   DS,AX                  ; SET SEGMENT REGISTER
8293                                    <1>       ;MOV   AX, [ORG_VECTOR]       ; GET DISKETTE VECTOR
8294                                    <1>       ;MOV   [DISK_VECTOR],AX       ;  INTO INT 40H
8295                                    <1>       ;MOV   AX, [ORG_VECTOR+2]
8296                                    <1>       ;MOV   [DISK_VECTOR+2],AX
8297                                    <1>       ;MOV   word [ORG_VECTOR],DISK_IO ; FIXED DISK HANDLER
8298                                    <1>       ;MOV   [ORG_VECTOR+2],CS
8299                                    <1>       ; 1st controller (primary master, slave)   - IRQ 14
8300                                    <1>       ;;MOV  word [HDISK_INT],HD_INT        ; FIXED DISK INTERRUPT
8301                                    <1>       ;mov   word [HDISK_INT1],HD_INT  ;
8302                                    <1>       ;;MOV  [HDISK_INT+2],CS
8303                                    <1>       ;mov   [HDISK_INT1+2],CS
8304                                    <1>       ; 2nd controller (secondary master, slave) - IRQ 15
8305                                    <1>       ;mov   word [HDISK_INT2],HD1_INT ;
8306                                    <1>       ;mov   [HDISK_INT2+2],CS
8307                                    <1>       ;
8308                                    <1>       ;;MOV  word [HF_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80
8309                                    <1>       ;;MOV  word [HF_TBL_VEC+2],DPT_SEGM
8310                                    <1>       ;;MOV  word [HF1_TBL_VEC],HD1_DPT; PARM TABLE DRIVE 81
8311                                    <1>       ;;MOV  word [HF1_TBL_VEC+2],DPT_SEGM
8312                                    <1>       ;push  cs
8313                                    <1>       ;pop   ds
8314                                    <1>       ;mov   word [HDPM_TBL_VEC],HD0_DPT     ; PARM TABLE DRIVE 80h
8315                                    <1>       ;mov   word [HDPM_TBL_VEC+2],DPT_SEGM
8316 000027CA C705[04710000]0000-      <1>       mov    dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
8317 000027D2 0900                     <1>
8318                                    <1>       ;mov   word [HDPS_TBL_VEC],HD1_DPT     ; PARM TABLE DRIVE 81h
8319                                    <1>       ;mov   word [HDPS_TBL_VEC+2],DPT_SEGM
8320 000027D4 C705[08710000]2000-      <1>       mov    dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
8321 000027DC 0900                     <1>
8322                                    <1>       ;mov   word [HDSM_TBL_VEC],HD2_DPT     ; PARM TABLE DRIVE 82h
8323                                    <1>       ;mov   word [HDSM_TBL_VEC+2],DPT_SEGM
8324 000027DE C705[0C710000]4000-      <1>       mov    dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
8325 000027E6 0900                     <1>
8326                                    <1>       ;mov   word [HDSS_TBL_VEC],HD3_DPT     ; PARM TABLE DRIVE 83h
8327                                    <1>       ;mov   word [HDSS_TBL_VEC+2],DPT_SEGM
8328 000027E8 C705[10710000]6000-      <1>       mov    dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
8329 000027F0 0900                     <1>
8330                                    <1>       ;
8331                                    <1>       ;;IN   AL,INTB01          ; TURN ON SECOND INTERRUPT CHIP
8332                                    <1>       ;;;AND AL,0BFH
8333                                    <1>       ;;and  al, 3Fh                  ; enable IRQ 14 and IRQ 15
8334                                    <1>       ;;;JMP $+2
8335                                    <1>       ;;IODELAY
8336                                    <1>       ;;OUT  INTB01,AL
8337                                    <1>       ;;IODELAY
8338                                    <1>       ;;IN   AL,INTA01          ; LET INTERRUPTS PASS THRU TO
8339                                    <1>       ;;AND  AL,0FBH            ;  SECOND CHIP
8340                                    <1>       ;;;JMP $+2
8341                                    <1>       ;;IODELAY
8342                                    <1>       ;;OUT  INTA01,AL
8343                                    <1>       ;
8344                                    <1>       ;STI
8345                                    <1>       ;;PUSH DS                 ; MOVE ABS0 POINTER TO
8346                                    <1>       ;;POP  ES                 ; EXTRA SEGMENT POINTER
8347                                    <1>       ;;;CALL      DDS                ; ESTABLISH DATA SEGMENT
8348                                    <1>       ;;MOV  byte [DISK_STATUS1],0     ; RESET THE STATUS INDICATOR
8349                                    <1>       ;;MOV  byte [HF_NUM],0           ; ZERO NUMBER OF FIXED DISKS
8350                                    <1>       ;;MOV  byte [CONTROL_BYTE],0
8351                                    <1>       ;;MOV  byte [PORT_OFF],0   ; ZERO CARD OFFSET
8352                                    <1>       ; 20/12/2014 - private code by Erdogan Tan
8353                                    <1>                    ; (out of original PC-AT, PC-XT BIOS code)
8354                                    <1>       ;mov   si, hd0_type
8355 000027F2 BE[206B0000]             <1>       mov    esi, hd0_type
8356                                    <1>       ;mov   cx, 4
8357 000027F7 B904000000               <1>       mov    ecx, 4
8358                                    <1> hde_l:
8359 000027FC AC                       <1>       lodsb
8360 000027FD 3C80                     <1>       cmp    al, 80h                  ; 8?h = existing
8361 000027FF 7206                     <1>       jb     short _L4
8362 00002801 FE05[00710000]           <1>       inc    byte [HF_NUM]       ; + 1 hard (fixed) disk drives
8363                                    <1> _L4: ; 26/02/2015
8364 00002807 E2F3                     <1>       loop   hde_l
8365                                    <1> ;_L4:                         ; 0 <= [HF_NUM] =< 4
8366                                    <1> ;L4:
```

```
8367                                    <1>       ;
8368                                    <1>       ;; 31/12/2014 - cancel controller diagnostics here
8369                                    <1>       ;;;mov       cx, 3  ; 26/12/2014 (Award BIOS 1999)
8370                                    <1>       ;;mov  cl, 3
8371                                    <1>       ;;
8372                                    <1>       ;;MOV  DL,80H              ; CHECK THE CONTROLLER
8373                                    <1> ;;hdc_dl:
8374                                    <1>       ;;MOV  AH,14H              ; USE CONTROLLER DIAGNOSTIC COMMAND
8375                                    <1>       ;;INT  13H                 ; CALL BIOS WITH DIAGNOSTIC COMMAND
8376                                    <1>       ;;;JC  short CTL_ERRX            ; DISPLAY ERROR MESSAGE IF BAD RETURN
8377                                    <1>       ;;;jc  short POD_DONE ;22/12/2014
8378                                    <1>       ;;jnc  short hdc_reset0
8379                                    <1>       ;;loop hdc_dl
8380                                    <1>       ;;; 27/12/2014
8381                                    <1>       ;;stc
8382                                    <1>       ;;retn
8383                                    <1>       ;
8384                                    <1> ;;hdc_reset0:
8385                                    <1>       ; 18/01/2015
8386 00002809 8A0D[00710000]           <1>       mov    cl, [HF_NUM]
8387 0000280F 20C9                     <1>       and    cl, cl
8388 00002811 740D                     <1>       jz     short POD_DONE
8389                                    <1>       ;
8390 00002813 B27F                     <1>       mov    dl, 7Fh
8391                                    <1> hdc_reset1:
8392 00002815 FEC2                     <1>       inc    dl
8393                                    <1>       ;; 31/12/2015
8394                                    <1>       ;;push dx
8395                                    <1>       ;;push cx
8396                                    <1>       ;;push ds
8397                                    <1>       ;;sub  ax, ax
8398                                    <1>       ;;mov  ds, ax
8399                                    <1>       ;;MOV  AX, [TIMER_LOW]           ; GET START TIMER COUNTS
8400                                    <1>       ;;pop  ds
8401                                    <1>       ;;MOV  BX,AX
8402                                    <1>       ;;ADD  AX,6*182            ; 60 SECONDS* 18.2
8403                                    <1>       ;;MOV  CX,AX
8404                                    <1>       ;;mov  word [wait_count], 0      ; 22/12/2014 (reset wait counter)
8405                                    <1>       ;;
8406                                    <1>       ;; 31/12/2014 - cancel HD_RESET_1
8407                                    <1>       ;;CALL HD_RESET_1          ; SET UP DRIVE 0, (1,2,3)
8408                                    <1>       ;;pop  cx
8409                                    <1>       ;;pop  dx
8410                                    <1>       ;;
8411                                    <1>       ; 18/01/2015
8412 00002817 B40D                     <1>       mov    ah, 0Dh ; ALTERNATE RESET
8413                                    <1>       ;int   13h
8414 00002819 E8A4FFFFFF               <1>       call   int13h
8415 0000281E E2F5                     <1>       loop   hdc_reset1
8416                                    <1> POD_DONE:
8417 00002820 C3                       <1>       RETn
8418                                    <1>
8419                                    <1> ;;-----     POD_ERROR
8420                                    <1>
8421                                    <1> ;;CTL_ERRX:
8422                                    <1> ;     ;MOV   SI,OFFSET F1782     ; CONTROLLER ERROR
8423                                    <1> ;     ;CALL  SET_FAIL            ; DO NOT IPL FROM DISK
8424                                    <1> ;     ;CALL  E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
8425                                    <1> ;     ;JMP   short POD_DONE
8426                                    <1>
8427                                    <1> ;;HD_RESET_1:
8428                                    <1> ;;    ;PUSH  BX                  ; SAVE TIMER LIMITS
8429                                    <1> ;;    ;PUSH  CX
8430                                    <1> ;;RES_1: MOV AH,09H              ; SET DRIVE PARAMETERS
8431                                    <1> ;;    INT    13H
8432                                    <1> ;;    JC     short RES_2
8433                                    <1> ;;    MOV    AH,11H              ; RECALIBRATE DRIVE
8434                                    <1> ;;    INT    13H
8435                                    <1> ;;    JNC    short RES_CK        ; DRIVE OK
8436                                    <1> ;;RES_2: ;CALL     POD_TCHK            ; CHECK TIME OUT
8437                                    <1> ;;    cmp    word [wait_count], 6*182 ; waiting time (in timer ticks)
8438                                    <1> ;;                                ; (30 seconds)
8439                                    <1> ;;    ;cmc
8440                                    <1> ;;    ;JNC   short RES_1
8441                                    <1> ;;    jb     short RES_1
8442                                    <1> ;;;RES_FL: ;MOV    SI,OFFSET F1781     ; INDICATE DISK 1 FAILURE;
8443                                    <1> ;;    ;TEST  DL,1
8444                                    <1> ;;    ;JNZ   RES_E1
8445                                    <1> ;;    ;MOV   SI,OFFSET F1780     ; INDICATE DISK 0 FAILURE
8446                                    <1> ;;    ;CALL  SET_FAIL            ; DO NOT TRY TO IPL DISK 0
8447                                    <1> ;;    ;JMP   SHORT RES_E1
8448                                    <1> ;;RES_ER: ; 22/12/2014
8449                                    <1> ;;RES_OK:
8450                                    <1> ;;    ;POP   CX                  ; RESTORE TIMER LIMITS
8451                                    <1> ;;    ;POP   BX
8452                                    <1> ;;    RETn
8453                                    <1> ;;
8454                                    <1> ;;RES_RS: MOV      AH,00H              ; RESET THE DRIVE
8455                                    <1> ;;    INT    13H
8456                                    <1> ;;RES_CK: MOV      AH,08H              ; GET MAX CYLINDER,HEAD,SECTOR
8457                                    <1> ;;    MOV    BL,DL               ; SAVE DRIVE CODE
8458                                    <1> ;;    INT    13H
8459                                    <1> ;;    JC     short RES_ER
8460                                    <1> ;;    MOV    [NEC_STATUS],CX     ; SAVE MAX CYLINDER, SECTOR
8461                                    <1> ;;    MOV    DL,BL               ; RESTORE DRIVE CODE
8462                                    <1> ;;RES_3: MOV AX,0401H            ; VERIFY THE LAST SECTOR
8463                                    <1> ;;    INT    13H
8464                                    <1> ;;    JNC    short RES_OK        ; VERIFY OK
8465                                    <1> ;;    CMP    AH,BAD_SECTOR       ; OK ALSO IF JUST ID READ
8466                                    <1> ;;    JE     short RES_OK
8467                                    <1> ;;    CMP    AH,DATA_CORRECTED
8468                                    <1> ;;    JE     short RES_OK
8469                                    <1> ;;    CMP    AH,BAD_ECC
8470                                    <1> ;;    JE     short RES_OK
8471                                    <1> ;;    ;CALL  POD_TCHK            ; CHECK FOR TIME OUT
```

```
8472                                <1> ;;    cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
8473                                <1> ;;                                   ; (60 seconds)
8474                                <1> ;;    cmc
8475                                <1> ;;    JC    short RES_ER       ; FAILED
8476                                <1> ;;    MOV   CX,[NEC_STATUS]    ; GET SECTOR ADDRESS, AND CYLINDER
8477                                <1> ;;    MOV   AL,CL              ; SEPARATE OUT SECTOR NUMBER
8478                                <1> ;;    AND   AL,3FH
8479                                <1> ;;    DEC   AL                 ; TRY PREVIOUS ONE
8480                                <1> ;;    JZ    short RES_RS       ; WE'VE TRIED ALL SECTORS ON TRACK
8481                                <1> ;;    AND   CL,0C0H            ; KEEP CYLINDER BITS
8482                                <1> ;;    OR    CL,AL              ; MERGE SECTOR WITH CYLINDER BITS
8483                                <1> ;;    MOV   [NEC_STATUS],CX    ; SAVE CYLINDER, NEW SECTOR NUMBER
8484                                <1> ;;    JMP   short RES_3        ; TRY AGAIN
8485                                <1> ;;;RES_ER: MOV    SI,OFFSET F1791    ; INDICATE DISK 1 ERROR
8486                                <1> ;;    ;TEST  DL,1
8487                                <1> ;;    ;JNZ   short RES_E1
8488                                <1> ;;    ;MOV   SI,OFFSET F1790    ; INDICATE DISK 0 ERROR
8489                                <1> ;;;RES_E1:
8490                                <1> ;;    ;CALL  E_MSG              ; DISPLAY ERROR AND SET (BP) ERROR FLAG
8491                                <1> ;;;RES_OK:
8492                                <1> ;;    ;POP   CX                 ; RESTORE TIMER LIMITS
8493                                <1> ;;    ;POP   BX
8494                                <1> ;;    ;RETn
8495                                <1> ;
8496                                <1> ;;SET_FAIL:
8497                                <1> ;     ;MOV   AX,X*(CMOS_DIAG+NMI)      ; GET CMOS ERROR BYTE
8498                                <1> ;     ;CALL  CMOS_READ
8499                                <1> ;     ;OR    AL,HF_FAIL         ; SET DO NOT IPL FROM DISK FLAG
8500                                <1> ;     ;XCHG  AH,AL              ; SAVE IT
8501                                <1> ;     ;CALL  CMOS_WRITE         ; PUT IT OUT
8502                                <1> ;     ;RETn
8503                                <1> ;
8504                                <1> ;;POD_TCHK:                     ; CHECK FOR 30 SECOND TIME OUT
8505                                <1> ;     ;POP   AX                 ; SAVE RETURN
8506                                <1> ;     ;POP   CX                 ; GET TIME OUT LIMITS
8507                                <1> ;     ;POP   BX
8508                                <1> ;     ;PUSH  BX                 ; AND SAVE THEM AGAIN
8509                                <1> ;     ;PUSH  CX
8510                                <1> ;     ;PUSH  AX
8511                                <1> ;     ;push  ds
8512                                <1> ;     ;xor   ax, ax
8513                                <1> ;     ;mov   ds, ax             ; RESTORE RETURN
8514                                <1> ;     ;MOV   AX, [TIMER_LOW]          ; AX = CURRENT TIME
8515                                <1> ;     ;                         ; BX = START TIME
8516                                <1> ;     ;                         ; CX = END TIME
8517                                <1> ;     ;pop   ds
8518                                <1> ;     ;CMP   BX,CX
8519                                <1> ;     ;JB    short TCHK1        ; START < END
8520                                <1> ;     ;CMP   BX,AX
8521                                <1> ;     ;JB    short TCHKG        ; END < START < CURRENT
8522                                <1> ;     ;JMP   SHORT TCHK2        ; END, CURRENT < START
8523                                <1> ;;TCHK1: CMP AX,BX
8524                                <1> ;;    JB     short TCHKNG       ; CURRENT < START < END
8525                                <1> ;;TCHK2: CMP AX,CX
8526                                <1> ;;    JB     short TCHKG        ; START < CURRENT < END
8527                                <1> ;;                               ; OR CURRENT < END < START
8528                                <1> ;;TCHKNG: STC                         ; CARRY SET INDICATES TIME OUT
8529                                <1> ;;    RETn
8530                                <1> ;;TCHKG: CLC                    ; INDICATE STILL TIME
8531                                <1> ;;    RETn
8532                                <1> ;;
8533                                <1> ;;int_13h:
8534                                <1>
8535                                <1> ;---------------------------------------
8536                                <1> ;     FIXED DISK BIOS ENTRY POINT     :
8537                                <1> ;---------------------------------------
8538                                <1>
8539                                <1> DISK_IO:
8540 00002821 80FA80               <1>       CMP    DL,80H             ; TEST FOR FIXED DISK DRIVE
8541                                <1>       ;JAE   short A1           ; YES, HANDLE HERE
8542                                <1>       ;;;INT 40H                ; DISKETTE HANDLER
8543                                <1>       ;;call int40h
8544 00002824 0F8225F1FFFF         <1>       jb     DISKETTE_IO_1
8545                                <1> ;RET_2:
8546                                <1>       ;RETf  2                  ; BACK TO CALLER
8547                                <1> ;     retf   4
8548                                <1> A1:
8549 0000282A FB                   <1>       STI                       ; ENABLE INTERRUPTS
8550                                <1>       ;; 04/01/2015
8551                                <1>       ;;OR   AH,AH
8552                                <1>       ;;JNZ  short A2
8553                                <1>       ;;INT  40H                ; RESET NEC WHEN AH=0
8554                                <1>       ;;SUB  AH,AH
8555 0000282B 80FA83               <1>       CMP    DL,(80H + S_MAX_FILE - 1)
8556 0000282E 772C                 <1>       JA     short RET_2
8557                                <1>       ; 18/01/2015
8558 00002830 08E4                 <1>       or     ah,ah
8559 00002832 742B                 <1>       jz     short A4
8560 00002834 80FC0D               <1>       cmp    ah, 0Dh      ; Alternate reset
8561 00002837 7504                 <1>       jne    short A2
8562 00002839 28E4                 <1>       sub    ah,ah  ; Reset
8563 0000283B EB22                 <1>       jmp    short A4
8564                                <1> A2:
8565 0000283D 80FC08               <1>       CMP    AH,08H             ; GET PARAMETERS IS A SPECIAL CASE
8566                                <1>       ;JNZ   short A3
8567                                <1>         ;JMP    GET_PARM_N
8568 00002840 0F841C030000         <1>       je     GET_PARM_N
8569 00002846 80FC15               <1> A3:   CMP    AH,15H             ; READ DASD TYPE IS ALSO
8570                                <1>       ;JNZ   short A4
8571                                <1>         ;JMP    READ_DASD_TYPE
8572 00002849 0F84C7020000         <1>         je      READ_DASD_TYPE
8573                                <1>       ; 02/02/2015
8574 0000284F 80FC1D               <1>       cmp    ah, 1Dh                  ;(Temporary for Retro UNIX 386 v1)
8575                                <1>       ; 12/01/2015
8576 00002852 F5                   <1>       cmc
```

```
8577 00002853 730A                <1>       jnc    short A4
8578                              <1>       ; 30/01/2015
8579                              <1>       ;mov     byte [CS:DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
8580 00002855 C605[FF700000]01    <1>         mov     byte [DISK_STATUS1], BAD_CMD
8581                              <1>       ;jmp   short RET_2
8582                              <1> RET_2:
8583 0000285C CA0400              <1>       retf   4
8584                              <1> A4:                                 ; SAVE REGISTERS DURING OPERATION
8585 0000285F C8080000            <1>       ENTER  8,0                    ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
8586 00002863 53                  <1>       PUSH   eBX                    ;  IN THE STACK, THE COMMAND BLOCK IS:
8587 00002864 51                  <1>       PUSH   eCX                    ;   @CMD_BLOCK == BYTE PTR [BP]-8
8588 00002865 52                  <1>       PUSH   eDX
8589 00002866 1E                  <1>       PUSH   DS
8590 00002867 06                  <1>       PUSH   ES
8591 00002868 56                  <1>       PUSH   eSI
8592 00002869 57                  <1>       PUSH   eDI
8593                              <1>       ;;04/01/2015
8594                              <1>       ;;OR   AH,AH                  ; CHECK FOR RESET
8595                              <1>       ;;JNZ  short A5
8596                              <1>       ;;MOV  DL,80H                 ; FORCE DRIVE 80 FOR RESET
8597                              <1> ;;A5:
8598                              <1>       ;push  cs
8599                              <1>       ;pop   ds
8600                              <1>       ; 21/02/2015
8601 0000286A 6650                <1>       push   ax
8602 0000286C 66B81000            <1>       mov    ax, KDATA
8603 00002870 8ED8                <1>       mov    ds, ax
8604 00002872 8EC0                <1>       mov    es, ax
8605 00002874 6658                <1>       pop    ax
8606 00002876 E889000000          <1>       CALL   DISK_IO_CONT           ; PERFORM THE OPERATION
8607                              <1>       ;;CALL DDS                    ; ESTABLISH SEGMENT
8608 0000287B 8A25[FF700000]      <1>       MOV    AH,[DISK_STATUS1]      ; GET STATUS FROM OPERATION
8609 00002881 80FC01              <1>       CMP    AH,1                   ; SET THE CARRY FLAG TO INDICATE
8610 00002884 F5                  <1>       CMC                           ; SUCCESS OR FAILURE
8611 00002885 5F                  <1>       POP    eDI                    ; RESTORE REGISTERS
8612 00002886 5E                  <1>       POP    eSI
8613 00002887 07                  <1>         POP     ES
8614 00002888 1F                  <1>         POP     DS
8615 00002889 5A                  <1>       POP    eDX
8616 0000288A 59                  <1>       POP    eCX
8617 0000288B 5B                  <1>       POP    eBX
8618 0000288C C9                  <1>       LEAVE                         ; ADJUST (SP) AND RESTORE (BP)
8619                              <1>       ;RETf  2                      ; THROW AWAY SAVED FLAGS
8620 0000288D CA0400              <1>       retf   4
8621                              <1> ; 21/02/2015
8622                              <1> ;       dw --> dd
8623                              <1> M1:                                 ; FUNCTION TRANSFER TABLE
8624 00002890 [522A0000]          <1>       dd     DISK_RESET             ; 000H
8625 00002894 [C92A0000]          <1>       dd     RETURN_STATUS          ; 001H
8626 00002898 [D62A0000]          <1>       dd     DISK_READ              ; 002H
8627 0000289C [DF2A0000]          <1>       dd     DISK_WRITE             ; 003H
8628 000028A0 [E82A0000]          <1>       dd     DISK_VERF              ; 004H
8629 000028A4 [002B0000]          <1>       dd     FMT_TRK                ; 005H
8630 000028A8 [482A0000]          <1>       dd     BAD_COMMAND            ; 006H FORMAT BAD SECTORS
8631 000028AC [482A0000]          <1>       dd     BAD_COMMAND            ; 007H FORMAT DRIVE
8632 000028B0 [482A0000]          <1>       dd     BAD_COMMAND            ; 008H RETURN PARAMETERS
8633 000028B4 [C72B0000]          <1>       dd     INIT_DRV               ; 009H
8634 000028B8 [262C0000]          <1>       dd     RD_LONG                ; 00AH
8635 000028BC [2F2C0000]          <1>       dd     WR_LONG                ; 00BH
8636 000028C0 [382C0000]          <1>       dd     DISK_SEEK              ; 00CH
8637 000028C4 [522A0000]          <1>       dd     DISK_RESET             ; 00DH
8638 000028C8 [482A0000]          <1>       dd     BAD_COMMAND            ; 00EH READ BUFFER
8639 000028CC [482A0000]          <1>       dd     BAD_COMMAND            ; 00FH WRITE BUFFER
8640 000028D0 [602C0000]          <1>       dd     TST_RDY                ; 010H
8641 000028D4 [842C0000]          <1>       dd     HDISK_RECAL            ; 011H
8642 000028D8 [482A0000]          <1>       dd     BAD_COMMAND            ; 012H MEMORY DIAGNOSTIC
8643 000028DC [482A0000]          <1>       dd     BAD_COMMAND            ; 013H DRIVE DIAGNOSTIC
8644 000028E0 [BA2C0000]          <1>       dd     CTLR_DIAGNOSTIC        ; 014H CONTROLLER DIAGNOSTIC
8645                              <1>       ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
8646 000028E4 [482A0000]          <1>       dd     BAD_COMMAND            ; 015h
8647 000028E8 [482A0000]          <1>       dd     BAD_COMMAND            ; 016h
8648 000028EC [482A0000]          <1>       dd     BAD_COMMAND            ; 017h
8649 000028F0 [482A0000]          <1>       dd     BAD_COMMAND            ; 018h
8650 000028F4 [482A0000]          <1>       dd     BAD_COMMAND            ; 019h
8651 000028F8 [482A0000]          <1>       dd     BAD_COMMAND            ; 01Ah
8652 000028FC [D62A0000]          <1>       dd     DISK_READ              ; 01Bh ; LBA read
8653 00002900 [DF2A0000]          <1>       dd     DISK_WRITE             ; 01Ch ; LBA write
8654                              <1> M1L     EQU    $-M1
8655                              <1>
8656                              <1> DISK_IO_CONT:
8657                              <1>       ;;CALL DDS                    ; ESTABLISH SEGMENT
8658 00002904 80FC01              <1>       CMP    AH,01H                 ; RETURN STATUS
8659                              <1>       ;;JNZ  short SU0
8660                              <1>         ;;JMP    RETURN_STATUS
8661 00002907 0F84BC010000        <1>       je     RETURN_STATUS
8662                              <1> SU0:
8663 0000290D C605[FF700000]00    <1>       MOV    byte [DISK_STATUS1],0  ; RESET THE STATUS INDICATOR
8664                              <1>       ;;PUSH BX                     ; SAVE DATA ADDRESS
8665                              <1>       ;mov   si, bx ;; 14/02/2015
8666 00002914 89DE                <1>       mov    esi, ebx ; 21/02/2015
8667 00002916 8A1D[00710000]      <1>       MOV    BL,[HF_NUM]            ; GET NUMBER OF DRIVES
8668                              <1>       ;; 04/01/2015
8669                              <1>       ;;PUSH AX
8670 0000291C 80E27F              <1>       AND    DL,7FH                 ; GET DRIVE AS 0 OR 1
8671                              <1>                                     ; (get drive number as 0 to 3)
8672 0000291F 38D3                <1>       CMP    BL,DL
8673                              <1>         ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
8674 00002921 0F8621010000        <1>         jbe     BAD_COMMAND ;; 14/02/2015
8675                              <1>         ;
8676                              <1>       ;;03/01/2015
8677 00002927 29DB                <1>       sub    ebx, ebx
8678 00002929 88D3                <1>       mov    bl, dl
8679                              <1>       ;sub   bh, bh
8680 0000292B 883D[14710000]      <1>       mov    [LBAMode], bh          ; 0
8681                              <1>       ;;test byte [bx+hd0_type], 1     ; LBA ready ?
```

```
8682                                    <1>       ;test  byte [ebx+hd0_type], 1
8683                                    <1>       ;jz    short su1           ; no
8684                                    <1>       ;inc   byte [LBAMode]
8685                                    <1> ;su1:
8686                                    <1>       ; 21/02/2015 (32 bit modification)
8687                                    <1>       ;04/01/2015
8688 00002931 6650                     <1>       push   ax ; ***
8689                                    <1>       ;PUSH  ES ; **
8690 00002933 6652                     <1>       PUSH   DX ; *
8691 00002935 6650                     <1>       push   ax
8692 00002937 E85C060000               <1>       CALL   GET_VEC             ; GET DISK PARAMETERS
8693                                    <1>       ; 02/02/2015
8694                                    <1>       ;mov   ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
8695 0000293C 668B4310                 <1>       mov    ax, [ebx+16]
8696 00002940 66A3[146B0000]           <1>       mov    [HF_PORT], ax
8697                                    <1>       ;mov   dx, [ES:BX+18] ; control port address (3F6h, 376h)
8698 00002946 668B5312                 <1>       mov    dx, [ebx+18]
8699 0000294A 668915[166B0000]         <1>       mov    [HF_REG_PORT], dx
8700                                    <1>       ;mov   al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
8701 00002951 8A4314                   <1>       mov    al, [ebx+20]
8702                                    <1>       ; 23/02/2015
8703 00002954 A840                     <1>       test   al, 40h       ; LBA bit (bit 6)
8704 00002956 7406                     <1>       jz     short su1
8705 00002958 FE05[14710000]           <1>       inc    byte [LBAMode] ; 1
8706                                    <1> su1:
8707 0000295E C0E804                   <1>       shr    al, 4
8708 00002961 2401                     <1>       and    al, 1
8709 00002963 A2[186B0000]             <1>       mov    [hf_m_s], al
8710                                    <1>       ;
8711                                    <1>       ; 03/01/2015
8712                                    <1>       ;MOV   AL,byte [ES:BX+8]   ; GET CONTROL BYTE MODIFIER
8713 00002968 8A4308                   <1>       mov    al, [ebx+8]
8714                                    <1>       ;MOV   DX,[HF_REG_PORT]    ; Device Control register
8715 0000296B EE                       <1>       OUT    DX,AL               ; SET EXTRA HEAD OPTION
8716                                    <1>                                  ; Control Byte:  (= 08h, here)
8717                                    <1>                                  ; bit 0 - 0
8718                                    <1>                                  ; bit 1 - nIEN (1 = disable irq)
8719                                    <1>                                  ; bit 2 - SRST (software RESET)
8720                                    <1>                                  ; bit 3 - use extra heads (8 to 15)
8721                                    <1>                                  ;         -always set to 1-
8722                                    <1>                                  ; (bits 3 to 7 are reserved
8723                                    <1>                                  ;          for ATA devices)
8724 0000296C 8A25[01710000]           <1>       MOV    AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
8725 00002972 80E4C0                   <1>       AND    AH,0C0H             ; CONTROL BYTE
8726 00002975 08C4                     <1>       OR     AH,AL
8727 00002977 8825[01710000]           <1>       MOV    [CONTROL_BYTE],AH
8728                                    <1>       ; 04/01/2015
8729 0000297D 6658                     <1>       pop    ax
8730 0000297F 665A                     <1>       pop    dx ; * ;; 14/02/2015
8731 00002981 20E4                     <1>       and    ah, ah ; Reset function ?
8732 00002983 7507                     <1>       jnz    short su2
8733                                    <1>       ;;pop  dx ; * ;; 14/02/2015
8734                                    <1>       ;pop   es ; **
8735 00002985 6658                     <1>       pop    ax ; ***
8736                                    <1>       ;;pop  bx
8737 00002987 E9C6000000               <1>         jmp     DISK_RESET
8738                                    <1> su2:
8739 0000298C 803D[14710000]00         <1>       cmp    byte [LBAMode], 0
8740 00002993 7661                     <1>       jna    short su3
8741                                    <1>       ;
8742                                    <1>       ; 02/02/2015 (LBA read/write function calls)
8743 00002995 80FC1B                   <1>       cmp    ah, 1Bh
8744 00002998 720B                     <1>       jb     short lbarw1
8745 0000299A 80FC1C                   <1>       cmp    ah, 1Ch
8746 0000299D 775C                     <1>       ja     short invldfnc
8747                                    <1>       ;;pop  dx ; * ; 14/02/2015
8748                                    <1>       ;mov   ax, cx ; Lower word of LBA address (bits 0-15)
8749 0000299F 89C8                     <1>       mov    eax, ecx ; LBA address (21/02/2015)
8750                                    <1>       ;; 14/02/2015
8751 000029A1 88D1                     <1>       mov    cl, dl ; 14/02/2015
8752                                    <1>       ;;mov  dx, bx
8753                                    <1>       ;mov   dx, si ; higher word of LBA address (bits 16-23)
8754                                    <1>       ;;mov  bx, di
8755                                    <1>       ;mov   si, di ; Buffer offset
8756 000029A3 EB31                     <1>       jmp    short lbarw2
8757                                    <1> lbarw1:
8758                                    <1>       ; convert CHS to LBA
8759                                    <1>       ;
8760                                    <1>       ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
8761                                    <1>       ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
8762                                    <1>       ;      + Sector - 1
8763 000029A5 6652                     <1>       push   dx ; * ;; 14/02/2015
8764                                    <1>       ;xor   dh, dh
8765 000029A7 31D2                     <1>       xor    edx, edx
8766                                    <1>       ;mov   dl, [ES:BX+14]      ; sectors per track (logical)
8767 000029A9 8A530E                   <1>       mov    dl, [ebx+14]
8768                                    <1>       ;xor   ah, ah
8769 000029AC 31C0                     <1>       xor    eax, eax
8770                                    <1>       ;mov   al, [ES:BX+2]; heads (logical)
8771 000029AE 8A4302                   <1>       mov    al, [ebx+2]
8772 000029B1 FEC8                     <1>       dec    al
8773 000029B3 6640                     <1>       inc    ax           ; 0 =  256
8774 000029B5 66F7E2                   <1>       mul    dx
8775                                    <1>              ; AX = # of Heads" * Sectors/Track
8776 000029B8 6689CA                   <1>       mov    dx, cx
8777                                    <1>       ;and   cx, 3Fh       ; sector  (1 to 63)
8778 000029BB 83E13F                   <1>       and    ecx, 3fh
8779 000029BE 86D6                     <1>       xchg   dl, dh
8780 000029C0 C0EE06                   <1>       shr    dh, 6
8781                                    <1>              ; DX = cylinder (0 to 1023)
8782                                    <1>       ;mul   dx
8783                                    <1>              ; DX:AX = # of Heads" * Sectors/Track * Cylinder
8784 000029C3 F7E2                     <1>       mul    edx
8785 000029C5 FEC9                     <1>       dec    cl  ; sector - 1
8786                                    <1>       ;add   ax, cx
```

```
8787                                    <1>       ;adc   dx, 0
8788                                    <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
8789 000029C7 01C8                      <1>       add    eax, ecx
8790 000029C9 6659                      <1>       pop    cx ; * ; ch = head, cl = drive number (zero based)
8791                                    <1>       ;push  dx
8792                                    <1>       ;push  ax
8793 000029CB 50                        <1>       push   eax
8794                                    <1>       ;mov   al, [ES:BX+14]      ; sectors per track (logical)
8795 000029CC 8A430E                    <1>       mov    al, [ebx+14]
8796 000029CF F6E5                      <1>       mul    ch
8797                                    <1>             ;  AX = Head * Sectors/Track
8798 000029D1 6699                      <1>         cwd
8799                                    <1>       ;pop   dx
8800 000029D3 5A                        <1>       pop    edx
8801                                    <1>       ;add   ax, dx
8802                                    <1>       ;pop   dx
8803                                    <1>       ;adc   dx, 0 ; add carry bit
8804 000029D4 01D0                      <1>       add    eax, edx
8805                                    <1> lbarw2:
8806 000029D6 29D2                      <1>       sub    edx, edx ; 21/02/2015
8807 000029D8 88CA                      <1>       mov    dl, cl ; 21/02/2015
8808 000029DA C645F800                  <1>         mov     byte [CMD_BLOCK], 0 ; Features Register
8809                                    <1>                         ; NOTE: Features register (1F1h, 171h)
8810                                    <1>                         ; is not used for ATA device R/W functions.
8811                                    <1>                         ; It is old/obsolete 'write precompensation'
8812                                    <1>                         ; register and error register
8813                                    <1>                         ; for old ATA/IDE devices.
8814                                    <1>       ; 18/01/2014
8815                                    <1>       ;mov   ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
8816 000029DE 8A0D[186B0000]           <1>       mov    cl, [hf_m_s]
8817                                    <1>       ;shl   ch, 4        ; bit 4 (drive bit)
8818                                    <1>       ;or    ch, 0E0h     ; bit 5 = 1
8819                                    <1>                         ; bit 6 = 1 = LBA mode
8820                                    <1>                         ; bit 7 = 1
8821 000029E4 80C90E                    <1>       or     cl, 0Eh ; 1110b
8822                                    <1>       ;and   dh, 0Fh             ; LBA byte 4 (bits 24 to 27)
8823 000029E7 25FFFFFF0F                <1>       and    eax, 0FFFFFFFh
8824 000029EC C1E11C                    <1>       shl    ecx, 28 ; 21/02/2015
8825                                    <1>       ;or    dh, ch
8826 000029EF 09C8                      <1>       or     eax, ecx
8827                                    <1>       ;;mov  [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
8828                                    <1>                              ; (Sector Number Register)
8829                                    <1>       ;;mov  [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
8830                                    <1>                              ; (Cylinder Low Register)
8831                                    <1>       ;mov   [CMD_BLOCK+2], ax ; LBA byte 1, 2
8832                                    <1>       ;mov   [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
8833                                    <1>                              ; (Cylinder High Register)
8834                                    <1>       ;;mov  [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
8835                                    <1>                              ; (Drive/Head Register)
8836                                    <1>
8837                                    <1>       ;mov   [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
8838 000029F1 8945FA                   <1>       mov    [CMD_BLOCK+2], eax ; 21/02/2015
8839                                    <1>       ;14/02/2015
8840                                    <1>       ;mov   dl, cl ; Drive number (INIT_DRV)
8841 000029F4 EB38                      <1>       jmp    short su4
8842                                    <1> su3:
8843                                    <1>       ; 02/02/2015
8844                                    <1>       ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
8845 000029F6 80FC14                    <1>       cmp    ah, 14h
8846 000029F9 7604                      <1>       jna    short chsfnc
8847                                    <1> invldfnc:
8848                                    <1>         ; 14/02/2015
8849                                    <1>       ;pop   es ; **
8850 000029FB 6658                      <1>         pop     ax ; ***
8851                                    <1>         ;jmp     short BAD_COMMAND_POP
8852 000029FD EB49                      <1>         jmp     short BAD_COMMAND
8853                                    <1> chsfnc:
8854                                    <1>       ;MOV   AX,[ES:BX+5]        ; GET WRITE PRE-COMPENSATION CYLINDER
8855 000029FF 668B4305                  <1>       mov    ax, [ebx+5]
8856 00002A03 66C1E802                  <1>       SHR    AX,2
8857 00002A07 8845F8                    <1>       MOV    [CMD_BLOCK],AL
8858                                    <1>       ;;MOV  AL,[ES:BX+8]        ; GET CONTROL BYTE MODIFIER
8859                                    <1>       ;;PUSH DX
8860                                    <1>       ;;MOV  DX,[HF_REG_PORT]
8861                                    <1>       ;;OUT  DX,AL               ; SET EXTRA HEAD OPTION
8862                                    <1>       ;;POP  DX ; *
8863                                    <1>       ;;POP  ES ; **
8864                                    <1>       ;;MOV  AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
8865                                    <1>       ;;AND  AH,0C0H             ; CONTROL BYTE
8866                                    <1>       ;;OR   AH,AL
8867                                    <1>       ;;MOV  [CONTROL_BYTE],AH
8868                                    <1>       ;
8869 00002A0A 88C8                      <1>       MOV    AL,CL               ; GET SECTOR NUMBER
8870 00002A0C 243F                      <1>       AND    AL,3FH
8871 00002A0E 8845FA                    <1>       MOV    [CMD_BLOCK+2],AL
8872 00002A11 886DFB                    <1>       MOV    [CMD_BLOCK+3],CH    ; GET CYLINDER NUMBER
8873 00002A14 88C8                      <1>       MOV    AL,CL
8874 00002A16 C0E806                    <1>       SHR    AL,6
8875 00002A19 8845FC                    <1>       MOV    [CMD_BLOCK+4],AL    ; CYLINDER HIGH ORDER 2 BITS
8876                                    <1>       ;;05/01/2015
8877                                    <1>       ;;MOV  AL,DL               ; DRIVE NUMBER
8878 00002A1C A0[186B0000]              <1>       mov    al, [hf_m_s]
8879 00002A21 C0E004                    <1>       SHL    AL,4
8880 00002A24 80E60F                    <1>       AND    DH,0FH              ; HEAD NUMBER
8881 00002A27 08F0                      <1>       OR     AL,DH
8882                                    <1>       ;OR    AL,80H or 20H
8883 00002A29 0CA0                      <1>       OR     AL,80h+20h          ; ECC AND 512 BYTE SECTORS
8884 00002A2B 8845FD                    <1>       MOV    [CMD_BLOCK+5],AL    ; ECC/SIZE/DRIVE/HEAD
8885                                    <1> su4:
8886                                    <1>       ;POP   ES ; **
8887                                    <1>         ;; 14/02/2015
8888                                    <1>         ;;POP   AX
8889                                    <1>         ;;MOV   [CMD_BLOCK+1],AL        ; SECTOR COUNT
8890                                    <1>         ;;PUSH  AX
8891                                    <1>         ;;MOV   AL,AH                   ; GET INTO LOW BYTE
```

```
8892                                    <1>         ;;XOR   AH,AH                   ; ZERO HIGH BYTE
8893                                    <1>         ;;SAL   AX,1                    ; *2 FOR TABLE LOOKUP
8894 00002A2E 6658                      <1>         pop     ax ; ***
8895 00002A30 8845F9                    <1>         mov     [CMD_BLOCK+1], al
8896 00002A33 29DB                      <1>         sub  ebx, ebx
8897 00002A35 88E3                      <1>       mov     bl, ah
8898                                    <1>         ;xor     bh, bh
8899                                    <1>         ;sal     bx, 1
8900 00002A37 66C1E302                  <1>         sal  bx, 2  ; 32 bit offset (21/02/2015)
8901                                    <1>       ;;MOV   SI,AX                   ; PUT INTO SI FOR BRANCH
8902                                    <1>         ;;CMP   AX,M1L                  ; TEST WITHIN RANGE
8903                                    <1>         ;;JNB   short BAD_COMMAND_POP
8904                                    <1>         ;cmp     bx, M1L
8905 00002A3B 83FB74                   <1>       cmp    ebx, M1L
8906 00002A3E 7308                      <1>       jnb    short BAD_COMMAND
8907                                    <1>         ;xchg    bx, si
8908 00002A40 87DE                      <1>         xchg ebx, esi
8909                                    <1>       ;;;POP AX                 ; RESTORE AX
8910                                    <1>       ;;;POP BX                 ; AND DATA ADDRESS
8911                                    <1>
8912                                    <1>       ;;PUSH CX
8913                                    <1>       ;;PUSH AX                 ; ADJUST ES:BX
8914                                    <1>       ;MOV   CX,BX              ; GET 3 HIGH ORDER NIBBLES OF BX
8915                                    <1>       ;SHR   CX,4
8916                                    <1>       ;MOV   AX,ES
8917                                    <1>       ;ADD   AX,CX
8918                                    <1>       ;MOV   ES,AX
8919                                    <1>       ;AND   BX,000FH           ; ES:BX CHANGED TO ES:000X
8920                                    <1>       ;;POP  AX
8921                                    <1>       ;;POP  CX
8922                                    <1>       ;;JMP  word [CS:SI+M1]
8923                                    <1>       ;jmp   word [SI+M1]
8924 00002A42 FFA6[90280000]            <1>       jmp    dword [esi+M1]
8925                                    <1> ;;BAD_COMMAND_POP:
8926                                    <1> ;;    POP    AX
8927                                    <1> ;;    POP    BX
8928                                    <1> BAD_COMMAND:
8929 00002A48 C605[FF700000]01          <1>         MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
8930 00002A4F B000                      <1>       MOV    AL,0
8931 00002A51 C3                        <1>       RETn
8932                                    <1>
8933                                    <1> ;----------------------------------------
8934                                    <1> ;     RESET THE DISK SYSTEM  (AH=00H) :
8935                                    <1> ;----------------------------------------
8936                                    <1>
8937                                    <1> ; 18-1-2015 : one controller reset (not other one)
8938                                    <1>
8939                                    <1> DISK_RESET:
8940 00002A52 FA                        <1>       CLI
8941 00002A53 E4A1                      <1>       IN     AL,INTB01          ; GET THE MASK REGISTER
8942                                    <1>       ;JMP   $+2
8943                                    <1>       IODELAY
8944 00002A55 EB00                      <2>  jmp short $+2
8945 00002A57 EB00                      <2>  jmp short $+2
8946                                    <1>       ;AND   AL,0BFH            ; ENABLE FIXED DISK INTERRUPT
8947 00002A59 243F                      <1>       and    al,3Fh             ; 22/12/2014 (IRQ 14 & IRQ 15)
8948 00002A5B E6A1                      <1>       OUT    INTB01,AL
8949 00002A5D FB                        <1>       STI                       ; START INTERRUPTS
8950                                    <1>       ; 14/02/2015
8951 00002A5E 6689D7                    <1>       mov    di, dx
8952                                    <1>       ; 04/01/2015
8953                                    <1>       ;xor   di,di
8954                                    <1> drst0:
8955 00002A61 B004                      <1>       MOV    AL,04H  ; bit 2 - SRST
8956                                    <1>       ;MOV   DX,HF_REG_PORT
8957 00002A63 668B15[166B0000]          <1>       MOV    DX,[HF_REG_PORT]
8958 00002A6A EE                        <1>       OUT    DX,AL              ; RESET
8959                                    <1> ;     MOV    CX,10              ; DELAY COUNT
8960                                    <1> ;DRD: DEC    CX
8961                                    <1> ;     JNZ    short DRD          ; WAIT 4.8 MICRO-SEC
8962                                    <1>       ;mov   cx,2               ; wait for 30 micro seconds
8963 00002A6B B902000000                <1>         mov  ecx, 2 ; 21/02/2015
8964 00002A70 E874EBFFFF                <1>       call    WAITF                   ; (Award Bios 1999 - WAIT_REFRESH,
8965                                    <1>                                       ; 40 micro seconds)
8966 00002A75 A0[01710000]              <1>       mov    al,[CONTROL_BYTE]
8967 00002A7A 240F                      <1>       AND    AL,0FH             ; SET HEAD OPTION
8968 00002A7C EE                        <1>       OUT    DX,AL              ; TURN RESET OFF
8969 00002A7D E80E040000                <1>       CALL   NOT_BUSY
8970 00002A82 7515                      <1>       JNZ    short DRERR        ; TIME OUT ON RESET
8971 00002A84 668B15[146B0000]          <1>       MOV    DX,[HF_PORT]
8972 00002A8B FEC2                      <1>       inc    dl  ; HF_PORT+1
8973                                    <1>       ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
8974                                    <1>         ;mov     cl, 10
8975 00002A8D B90A000000                <1>         mov     ecx, 10 ; 21/02/2015
8976                                    <1> drst1:
8977 00002A92 EC                        <1>       IN     AL,DX              ; GET RESET STATUS
8978 00002A93 3C01                      <1>       CMP    AL,1
8979                                    <1>       ; 04/01/2015
8980 00002A95 740A                      <1>       jz     short drst2
8981                                    <1>       ;JNZ   short DRERR        ; BAD RESET STATUS
8982                                    <1>              ; Drive/Head Register - bit 4
8983 00002A97 E2F9                      <1>       loop   drst1
8984                                    <1> DRERR:
8985 00002A99 C605[FF700000]05          <1>       MOV    byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
8986 00002AA0 C3                        <1>       RETn
8987                                    <1> drst2:
8988                                    <1>       ; 14/02/2015
8989 00002AA1 6689FA                    <1>       mov    dx,di
8990                                    <1> ;drst3:
8991                                    <1> ;     ; 05/01/2015
8992                                    <1> ;     shl    di,1
8993                                    <1> ;     ; 04/01/2015
8994                                    <1> ;     mov    ax,[di+hd_cports]
8995                                    <1> ;     cmp    ax,[HF_REG_PORT]
8996                                    <1> ;     je     short drst4
```

```
8997                                    <1> ;       mov     [HF_REG_PORT], ax
8998                                    <1> ;       ; 03/01/2015
8999                                    <1> ;       mov     ax,[di+hd_ports]
9000                                    <1> ;         mov     [HF_PORT], ax
9001                                    <1> ;       ; 05/01/2014
9002                                    <1> ;       shr     di,1
9003                                    <1> ;       ; 04/01/2015
9004                                    <1> ;       jmp     short drst0  ; reset other controller
9005                                    <1> ;drst4:
9006                                    <1> ;       ; 05/01/2015
9007                                    <1> ;       shr     di,1
9008                                    <1> ;       mov     al,[di+hd_dregs]
9009                                    <1> ;       and     al,10h ; bit 4 only
9010                                    <1> ;       shr     al,4 ; bit 4  -> bit 0
9011                                    <1> ;       mov     [hf_m_s], al ; (0 = master, 1 = slave)
9012                                    <1>         ;
9013 00002AA4 A0[186B0000]              <1>         mov     al, [hf_m_s] ; 18/01/2015
9014 00002AA9 A801                      <1>         test    al,1
9015                                    <1> ;       jnz     short drst6
9016 00002AAB 7516                      <1>           jnz     short drst4
9017 00002AAD 8065FDEF                  <1>         AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
9018                                    <1> ;drst5:
9019                                    <1> drst3:
9020 00002AB1 E811010000                <1>         CALL    INIT_DRV            ; SET MAX HEADS
9021                                    <1>         ;mov    dx,di
9022 00002AB6 E8C9010000                <1>         CALL    HDISK_RECAL         ; RECAL TO RESET SEEK SPEED
9023                                    <1>         ; 04/01/2014
9024                                    <1> ;       inc     di
9025                                    <1> ;       mov     dx,di
9026                                    <1> ;       cmp     dl,[HF_NUM]
9027                                    <1> ;       jb      short drst3
9028                                    <1> ;DRE:
9029 00002ABB C605[FF700000]00          <1>         MOV     byte [DISK_STATUS1],0     ; IGNORE ANY SET UP ERRORS
9030 00002AC2 C3                        <1>         RETn
9031                                    <1> ;drst6:
9032                                    <1> drst4:              ; Drive/Head Register - bit 4
9033 00002AC3 804DFD10                  <1>         OR       byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
9034                                    <1>           ;jmp    short drst5
9035 00002AC7 EBE8                      <1>           jmp     short drst3
9036                                    <1>
9037                                    <1> ;----------------------------------------
9038                                    <1> ;       DISK STATUS ROUTINE  (AH = 01H) :
9039                                    <1> ;----------------------------------------
9040                                    <1>
9041                                    <1> RETURN_STATUS:
9042 00002AC9 A0[FF700000]              <1>         MOV     AL,[DISK_STATUS1]   ; OBTAIN PREVIOUS STATUS
9043 00002ACE C605[FF700000]00          <1>           MOV     byte [DISK_STATUS1],0   ; RESET STATUS
9044 00002AD5 C3                        <1>         RETn
9045                                    <1>
9046                                    <1> ;----------------------------------------
9047                                    <1> ;       DISK READ ROUTINE    (AH = 02H) :
9048                                    <1> ;----------------------------------------
9049                                    <1>
9050                                    <1> DISK_READ:
9051 00002AD6 C645FE20                  <1>         MOV     byte [CMD_BLOCK+6],READ_CMD
9052 00002ADA E930020000                <1>           JMP     COMMANDI
9053                                    <1>
9054                                    <1> ;----------------------------------------
9055                                    <1> ;       DISK WRITE ROUTINE   (AH = 03H) :
9056                                    <1> ;----------------------------------------
9057                                    <1>
9058                                    <1> DISK_WRITE:
9059 00002ADF C645FE30                  <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD
9060 00002AE3 E97C020000                <1>           JMP     COMMANDO
9061                                    <1>
9062                                    <1> ;----------------------------------------
9063                                    <1> ;       DISK VERIFY       (AH = 04H) :
9064                                    <1> ;----------------------------------------
9065                                    <1>
9066                                    <1> DISK_VERF:
9067 00002AE8 C645FE40                  <1>         MOV     byte [CMD_BLOCK+6],VERIFY_CMD
9068 00002AEC E8EA020000                <1>         CALL    COMMAND
9069 00002AF1 750C                      <1>         JNZ     short VERF_EXIT           ; CONTROLLER STILL BUSY
9070 00002AF3 E85C030000                <1>         CALL    _WAIT               ; (Original: CALL WAIT)
9071 00002AF8 7505                      <1>         JNZ     short VERF_EXIT           ; TIME OUT
9072 00002AFA E8E9030000                <1>         CALL    CHECK_STATUS
9073                                    <1> VERF_EXIT:
9074 00002AFF C3                        <1>         RETn
9075                                    <1>
9076                                    <1> ;----------------------------------------
9077                                    <1> ;       FORMATTING        (AH = 05H) :
9078                                    <1> ;----------------------------------------
9079                                    <1>
9080                                    <1> FMT_TRK:                        ; FORMAT TRACK      (AH = 005H)
9081 00002B00 C645FE50                  <1>         MOV     byte [CMD_BLOCK+6],FMTTRK_CMD
9082                                    <1>         ;PUSH   ES
9083                                    <1>         ;PUSH   BX
9084 00002B04 53                        <1>         push    ebx
9085 00002B05 E88E040000                <1>         CALL    GET_VEC             ; GET DISK PARAMETERS ADDRESS
9086                                    <1>         ;MOV    AL,[ES:BX+14]       ; GET SECTORS/TRACK
9087 00002B0A 8A430E                    <1>         mov     al, [ebx+14]
9088 00002B0D 8845F9                    <1>         MOV     [CMD_BLOCK+1],AL    ; SET SECTOR COUNT IN COMMAND
9089 00002B10 5B                        <1>         pop     ebx
9090                                    <1>         ;POP    BX
9091                                    <1>         ;POP    ES
9092 00002B11 E955020000                <1>           JMP     CMD_OF                  ; GO EXECUTE THE COMMAND
9093                                    <1>
9094                                    <1> ;----------------------------------------
9095                                    <1> ;       READ DASD TYPE          (AH = 15H) :
9096                                    <1> ;----------------------------------------
9097                                    <1>
9098                                    <1> READ_DASD_TYPE:
9099                                    <1> READ_D_T:                       ; GET DRIVE PARAMETERS
9100 00002B16 1E                        <1>         PUSH    DS                  ; SAVE REGISTERS
9101                                    <1>         ;PUSH   ES
```

```
9102 00002B17 53                <1>       PUSH   eBX
9103                            <1>       ;CALL  DDS                ; ESTABLISH ADDRESSING
9104                            <1>       ;push  cs
9105                            <1>       ;pop   ds
9106 00002B18 66BB1000          <1>         mov  bx, KDATA
9107 00002B1C 8EDB              <1>       mov    ds, bx
9108                            <1>       ;mov   es, bx
9109 00002B1E C605[FF700000]00  <1>       MOV     byte [DISK_STATUS1],0
9110 00002B25 8A1D[00710000]    <1>       MOV    BL,[HF_NUM]        ; GET NUMBER OF DRIVES
9111 00002B2B 80E27F            <1>       AND    DL,7FH             ; GET DRIVE NUMBER
9112 00002B2E 38D3              <1>       CMP    BL,DL
9113 00002B30 7625              <1>       JBE    short RDT_NOT_PRESENT     ; RETURN DRIVE NOT PRESENT
9114 00002B32 E861040000        <1>       CALL   GET_VEC            ; GET DISK PARAMETER ADDRESS
9115                            <1>       ;MOV   AL,[ES:BX+2]       ; HEADS
9116 00002B37 8A4302            <1>       mov    al, [ebx+2]
9117                            <1>       ;MOV   CL,[ES:BX+14]
9118 00002B3A 8A4B0E            <1>       mov    cl, [ebx+14]
9119 00002B3D F6E9              <1>       IMUL   CL                 ; * NUMBER OF SECTORS
9120                            <1>       ;MOV   CX,[ES:BX]         ; MAX NUMBER OF CYLINDERS
9121 00002B3F 668B0B            <1>       mov    cx ,[ebx]
9122                            <1>       ;
9123                            <1>       ; 02/01/2015
9124                            <1>       ; ** leave the last cylinder as reserved for diagnostics **
9125                            <1>       ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
9126 00002B42 6649              <1>       DEC    CX                 ; LEAVE ONE FOR DIAGNOSTICS
9127                            <1>       ;
9128 00002B44 66F7E9            <1>       IMUL   CX                 ; NUMBER OF SECTORS
9129 00002B47 6689D1            <1>       MOV    CX,DX              ; HIGH ORDER HALF
9130 00002B4A 6689C2            <1>       MOV    DX,AX              ; LOW ORDER HALF
9131                            <1>       ;SUB   AX,AX
9132 00002B4D 28C0              <1>       sub    al, al
9133 00002B4F B403              <1>       MOV    AH,03H             ; INDICATE FIXED DISK
9134 00002B51 5B                <1> RDT2: POP    eBX                ; RESTORE REGISTERS
9135                            <1>       ;POP   ES
9136 00002B52 1F                <1>       POP    DS
9137 00002B53 F8                <1>       CLC                       ; CLEAR CARRY
9138                            <1>       ;RETf  2
9139 00002B54 CA0400            <1>       retf   4
9140                            <1> RDT_NOT_PRESENT:
9141 00002B57 6629C0            <1>       SUB    AX,AX              ; DRIVE NOT PRESENT RETURN
9142 00002B5A 6689C1            <1>       MOV    CX,AX              ; ZERO BLOCK COUNT
9143 00002B5D 6689C2            <1>       MOV    DX,AX
9144 00002B60 EBEF              <1>       JMP    short RDT2
9145                            <1>
9146                            <1> ;---------------------------------------
9147                            <1> ;     GET PARAMETERS          (AH = 08H) :
9148                            <1> ;---------------------------------------
9149                            <1>
9150                            <1> GET_PARM_N:
9151                            <1> ;GET_PARM:                       ; GET DRIVE PARAMETERS
9152 00002B62 1E                <1>       PUSH   DS                 ; SAVE REGISTERS
9153                            <1>       ;PUSH  ES
9154 00002B63 53                <1>       PUSH   eBX
9155                            <1>       ;MOV   AX,ABS0            ; ESTABLISH ADDRESSING
9156                            <1>       ;MOV   DS,AX
9157                            <1>       ;TEST  DL,1               ; CHECK FOR DRIVE 1
9158                            <1>       ;JZ    short G0
9159                            <1>       ;LES   BX,@HF1_TBL_VEC
9160                            <1>       ;JMP   SHORT G1
9161                            <1> ;G0:  LES    BX,@HF_TBL_VEC
9162                            <1> ;G1:
9163                            <1>       ;CALL  DDS                ; ESTABLISH SEGMENT
9164                            <1>       ; 22/12/2014
9165                            <1>       ;push  cs
9166                            <1>       ;pop   ds
9167 00002B64 66BB1000          <1>       mov    bx, KDATA
9168 00002B68 8EDB              <1>       mov    ds, bx
9169                            <1>       ;mov   es, bx
9170                            <1>       ;
9171 00002B6A 80EA80            <1>       SUB    DL,80H
9172 00002B6D 80FA04            <1>       CMP    DL,MAX_FILE        ; TEST WITHIN RANGE
9173 00002B70 7341              <1>       JAE    short G4
9174                            <1>       ;
9175 00002B72 31DB              <1>       xor    ebx, ebx ; 21/02/2015
9176                            <1>       ; 22/12/2014
9177 00002B74 88D3              <1>       mov    bl, dl
9178                            <1>       ;xor   bh, bh
9179 00002B76 C0E302            <1>       shl    bl, 2              ; convert index to offset
9180                            <1>       ;add   bx, HF_TBL_VEC
9181 00002B79 81C3[04710000]    <1>       add    ebx, HF_TBL_VEC
9182                            <1>       ;mov   ax, [bx+2]
9183                            <1>       ;mov   es, ax              ; dpt segment
9184                            <1>       ;mov   bx, [bx]            ; dpt offset
9185 00002B7F 8B1B              <1>       mov    ebx, [ebx] ; 32 bit offset
9186                            <1>
9187 00002B81 C605[FF700000]00  <1>       MOV    byte [DISK_STATUS1],0
9188                            <1>         ;MOV     AX,[ES:BX]              ; MAX NUMBER OF CYLINDERS
9189 00002B88 668B03            <1>       mov    ax, [ebx]
9190                            <1>       ;;SUB  AX,2               ; ADJUST FOR 0-N
9191 00002B8B 6648              <1>       dec    ax                 ; max. cylinder number
9192 00002B8D 88C5              <1>       MOV    CH,AL
9193 00002B8F 66250003          <1>       AND    AX,0300H           ; HIGH TWO BITS OF CYLINDER
9194 00002B93 66D1E8            <1>       SHR    AX,1
9195 00002B96 66D1E8            <1>       SHR    AX,1
9196                            <1>       ;OR    AL,[ES:BX+14]      ; SECTORS
9197 00002B99 0A430E            <1>       or     al, [ebx+14]
9198 00002B9C 88C1              <1>       MOV    CL,AL
9199                            <1>       ;MOV   DH,[ES:BX+2]       ; HEADS
9200 00002B9E 8A7302            <1>       mov    dh, [ebx+2]
9201 00002BA1 FECE              <1>       DEC    DH                 ; 0-N RANGE
9202 00002BA3 8A15[00710000]    <1>       MOV    DL,[HF_NUM]        ; DRIVE COUNT
9203 00002BA9 6629C0            <1>       SUB    AX,AX
9204                            <1>         ;27/12/2014
9205                            <1>       ; ES:DI = Address of disk parameter table from BIOS
9206                            <1>       ;(Programmer's Guide to the AMIBIOS - 1993)
```

```
9207                                    <1>       ;mov   di, bx             ; HDPT offset
9208 00002BAC 89DF                      <1>       mov    edi, ebx
9209                                    <1> G5:
9210 00002BAE 5B                        <1>       POP    eBX                ; RESTORE REGISTERS
9211                                    <1>       ;POP   ES
9212 00002BAF 1F                        <1>       POP    DS
9213                                    <1>       ;RETf  2
9214 00002BB0 CA0400                    <1>       retf   4
9215                                    <1> G4:
9216 00002BB3 C605[FF700000]07          <1>       MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
9217 00002BBA B407                      <1>       MOV    AH,INIT_FAIL
9218 00002BBC 28C0                      <1>       SUB    AL,AL
9219 00002BBE 6629D2                    <1>       SUB    DX,DX
9220 00002BC1 6629C9                    <1>       SUB    CX,CX
9221 00002BC4 F9                        <1>       STC                       ; SET ERROR FLAG
9222 00002BC5 EBE7                      <1>       JMP    short G5
9223                                    <1>
9224                                    <1> ;--------------------------------------
9225                                    <1> ;     INITIALIZE DRIVE     (AH = 09H) :
9226                                    <1> ;--------------------------------------
9227                                    <1>       ; 03/01/2015
9228                                    <1>       ; According to ATA-ATAPI specification v2.0 to v5.0
9229                                    <1>       ; logical sector per logical track
9230                                    <1>       ; and logical heads - 1 would be set but
9231                                    <1>       ; it is seen as it will be good
9232                                    <1>       ; if physical parameters will be set here
9233                                    <1>       ; because, number of heads <= 16.
9234                                    <1>       ; (logical heads usually more than 16)
9235                                    <1>       ; NOTE: ATA logical parameters (software C, H, S)
9236                                    <1>       ;     == INT 13h physical parameters
9237                                    <1>
9238                                    <1> ;INIT_DRV:
9239                                    <1> ;     MOV    byte [CMD_BLOCK+6],SET_PARM_CMD
9240                                    <1> ;     CALL   GET_VEC             ; ES:BX -> PARAMETER BLOCK
9241                                    <1> ;     MOV    AL,[ES:BX+2]        ; GET NUMBER OF HEADS
9242                                    <1> ;     DEC    AL                  ; CONVERT TO 0-INDEX
9243                                    <1> ;     MOV    AH,[CMD_BLOCK+5]    ; GET SDH REGISTER
9244                                    <1> ;     AND    AH,0F0H             ; CHANGE HEAD NUMBER
9245                                    <1> ;     OR     AH,AL               ; TO MAX HEAD
9246                                    <1> ;     MOV    [CMD_BLOCK+5],AH
9247                                    <1> ;     MOV    AL,[ES:BX+14]       ; MAX SECTOR NUMBER
9248                                    <1> ;     MOV    [CMD_BLOCK+1],AL
9249                                    <1> ;     SUB    AX,AX
9250                                    <1> ;     MOV    [CMD_BLOCK+3],AL    ; ZERO FLAGS
9251                                    <1> ;     CALL   COMMAND             ; TELL CONTROLLER
9252                                    <1> ;     JNZ    short INIT_EXIT           ; CONTROLLER BUSY ERROR
9253                                    <1> ;     CALL   NOT_BUSY            ; WAIT FOR IT TO BE DONE
9254                                    <1> ;     JNZ    short INIT_EXIT           ; TIME OUT
9255                                    <1> ;     CALL   CHECK_STATUS
9256                                    <1> ;INIT_EXIT:
9257                                    <1> ;     RETn
9258                                    <1>
9259                                    <1> ; 04/01/2015
9260                                    <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
9261                                    <1> ;                          AHDSK.ASM - INIT_DRIVE
9262                                    <1> INIT_DRV:
9263                                    <1>       ;xor   ah,ah
9264 00002BC7 31C0                      <1>       xor    eax, eax ; 21/02/2015
9265 00002BC9 B00B                      <1>       mov    al,11 ; Physical heads from translated HDPT
9266 00002BCB 3825[14710000]            <1>         cmp     [LBAMode], ah   ; 0
9267 00002BD1 7702                      <1>       ja     short idrv0
9268 00002BD3 B002                      <1>       mov    al,2  ; Physical heads from standard HDPT
9269                                    <1> idrv0:
9270                                    <1>       ; DL = drive number (0 based)
9271 00002BD5 E8BE030000                <1>       call   GET_VEC
9272                                    <1>       ;push  bx
9273 00002BDA 53                        <1>       push   ebx ; 21/02/2015
9274                                    <1>       ;add   bx,ax
9275 00002BDB 01C3                      <1>       add    ebx, eax
9276                                    <1>       ;; 05/01/2015
9277 00002BDD 8A25[186B0000]            <1>       mov    ah, [hf_m_s] ; drive number (0= master, 1= slave)
9278                                    <1>       ;;and  ah,1
9279 00002BE3 C0E404                    <1>       shl    ah,4
9280 00002BE6 80CCA0                    <1>       or     ah,0A0h  ; Drive/Head register - 10100000b (A0h)
9281                                    <1>       ;mov   al,[es:bx]
9282 00002BE9 8A03                      <1>       mov    al, [ebx] ; 21/02/2015
9283 00002BEB FEC8                      <1>       dec    al      ; last head number
9284                                    <1>       ;and   al,0Fh
9285 00002BED 08E0                      <1>       or     al,ah   ; lower 4 bits for head number
9286                                    <1>       ;
9287 00002BEF C645FE91                  <1>       mov    byte [CMD_BLOCK+6],SET_PARM_CMD
9288 00002BF3 8845FD                    <1>       mov    [CMD_BLOCK+5],al
9289                                    <1>       ;pop   bx
9290 00002BF6 5B                        <1>       pop    ebx
9291 00002BF7 29C0                      <1>       sub    eax, eax ; 21/02/2015
9292 00002BF9 B004                      <1>       mov    al,4 ; Physical sec per track from translated HDPT
9293 00002BFB 803D[14710000]00          <1>       cmp    byte [LBAMode], 0
9294 00002C02 7702                      <1>       ja     short idrv1
9295 00002C04 B00E                      <1>       mov    al,14 ; Physical sec per track from standard HDPT
9296                                    <1> idrv1:
9297                                    <1>       ;xor   ah,ah
9298                                    <1>       ;add   bx,ax
9299 00002C06 01C3                      <1>       add    ebx, eax ; 21/02/2015
9300                                    <1>       ;mov   al,[es:bx]
9301                                    <1>                    ; sector number
9302 00002C08 8A03                      <1>       mov    al, [ebx]
9303 00002C0A 8845F9                    <1>       mov    [CMD_BLOCK+1],al
9304 00002C0D 28C0                      <1>       sub    al,al
9305 00002C0F 8845FB                    <1>       mov    [CMD_BLOCK+3],al  ; ZERO FLAGS
9306 00002C12 E8C4010000                <1>       call   COMMAND         ; TELL CONTROLLER
9307 00002C17 750C                      <1>       jnz    short INIT_EXIT       ; CONTROLLER BUSY ERROR
9308 00002C19 E872020000                <1>       call   NOT_BUSY        ; WAIT FOR IT TO BE DONE
9309 00002C1E 7505                      <1>       jnz    short INIT_EXIT       ; TIME OUT
9310 00002C20 E8C3020000                <1>       call   CHECK_STATUS
9311                                    <1> INIT_EXIT:
```

```
9312 00002C25 C3                <1>       RETn
9313                            <1>
9314                            <1> ;----------------------------------------
9315                            <1> ;     READ LONG         (AH = 0AH) :
9316                            <1> ;----------------------------------------
9317                            <1>
9318                            <1> RD_LONG:
9319                            <1>       ;MOV   @CMD_BLOCK+6,READ_CMD OR ECC_MODE
9320 00002C26 C645FE22          <1>         mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
9321 00002C2A E9E0000000        <1>         JMP     COMMANDI
9322                            <1>
9323                            <1> ;----------------------------------------
9324                            <1> ;     WRITE LONG        (AH = 0BH) :
9325                            <1> ;----------------------------------------
9326                            <1>
9327                            <1> WR_LONG:
9328                            <1>       ;MOV   @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
9329 00002C2F C645FE32          <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
9330 00002C33 E92C010000        <1>         JMP     COMMANDO
9331                            <1>
9332                            <1> ;----------------------------------------
9333                            <1> ;     SEEK              (AH = 0CH) :
9334                            <1> ;----------------------------------------
9335                            <1>
9336                            <1> DISK_SEEK:
9337 00002C38 C645FE70          <1>         MOV     byte [CMD_BLOCK+6],SEEK_CMD
9338 00002C3C E89A010000        <1>       CALL   COMMAND
9339 00002C41 751C              <1>       JNZ    short DS_EXIT             ; CONTROLLER BUSY ERROR
9340 00002C43 E80C020000        <1>       CALL   _WAIT
9341 00002C48 7515              <1>         JNZ     DS_EXIT                 ; TIME OUT ON SEEK
9342 00002C4A E899020000        <1>       CALL   CHECK_STATUS
9343 00002C4F 803D[FF700000]40  <1>         CMP     byte [DISK_STATUS1],BAD_SEEK
9344 00002C56 7507              <1>       JNE    short DS_EXIT
9345 00002C58 C605[FF700000]00  <1>         MOV     byte [DISK_STATUS1],0
9346                            <1> DS_EXIT:
9347 00002C5F C3                <1>       RETn
9348                            <1>
9349                            <1> ;----------------------------------------
9350                            <1> ;     TEST DISK READY      (AH = 10H) :
9351                            <1> ;----------------------------------------
9352                            <1>
9353                            <1> TST_RDY:                          ; WAIT FOR CONTROLLER
9354 00002C60 E82B020000        <1>       CALL   NOT_BUSY
9355 00002C65 751C              <1>       JNZ    short TR_EX
9356 00002C67 8A45FD            <1>       MOV    AL,[CMD_BLOCK+5]    ; SELECT DRIVE
9357 00002C6A 668B15[146B0000]  <1>       MOV    DX,[HF_PORT]
9358 00002C71 80C206            <1>       add    dl,6
9359 00002C74 EE                <1>       OUT    DX,AL
9360 00002C75 E886020000        <1>       CALL   CHECK_ST            ; CHECK STATUS ONLY
9361 00002C7A 7507              <1>       JNZ    short TR_EX
9362 00002C7C C605[FF700000]00  <1>       MOV    byte [DISK_STATUS1],0     ; WIPE OUT DATA CORRECTED ERROR
9363                            <1> TR_EX:
9364 00002C83 C3                <1>       RETn
9365                            <1>
9366                            <1> ;----------------------------------------
9367                            <1> ;     RECALIBRATE       (AH = 11H) :
9368                            <1> ;----------------------------------------
9369                            <1>
9370                            <1> HDISK_RECAL:
9371 00002C84 C645FE10          <1>         MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
9372 00002C88 E84E010000        <1>       CALL   COMMAND             ; START THE OPERATION
9373 00002C8D 7523              <1>       JNZ    short RECAL_EXIT    ; ERROR
9374 00002C8F E8C0010000        <1>       CALL   _WAIT               ; WAIT FOR COMPLETION
9375 00002C94 7407              <1>       JZ     short RECAL_X             ; TIME OUT ONE OK ?
9376 00002C96 E8B9010000        <1>       CALL   _WAIT               ; WAIT FOR COMPLETION LONGER
9377 00002C9B 7515              <1>       JNZ    short RECAL_EXIT    ; TIME OUT TWO TIMES IS ERROR
9378                            <1> RECAL_X:
9379 00002C9D E846020000        <1>       CALL   CHECK_STATUS
9380 00002CA2 803D[FF700000]40  <1>       CMP    byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
9381 00002CA9 7507              <1>       JNE    short RECAL_EXIT    ; IS OK
9382 00002CAB C605[FF700000]00  <1>       MOV    byte [DISK_STATUS1],0
9383                            <1> RECAL_EXIT:
9384 00002CB2 803D[FF700000]00  <1>         CMP     byte [DISK_STATUS1],0
9385 00002CB9 C3                <1>       RETn
9386                            <1>
9387                            <1> ;----------------------------------------
9388                            <1> ;      CONTROLLER DIAGNOSTIC (AH = 14H) :
9389                            <1> ;----------------------------------------
9390                            <1>
9391                            <1> CTLR_DIAGNOSTIC:
9392 00002CBA FA                <1>         CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
9393 00002CBB E4A1              <1>       IN     AL,INTB01           ; TURN ON SECOND INTERRUPT CHIP
9394                            <1>       ;AND   AL,0BFH
9395 00002CBD 243F              <1>       and    al, 3Fh                   ; enable IRQ 14 & IRQ 15
9396                            <1>       ;JMP   $+2
9397                            <1>       IODELAY
9398 00002CBF EB00              <2>  jmp short $+2
9399 00002CC1 EB00              <2>  jmp short $+2
9400 00002CC3 E6A1              <1>       OUT    INTB01,AL
9401                            <1>       IODELAY
9402 00002CC5 EB00              <2>  jmp short $+2
9403 00002CC7 EB00              <2>  jmp short $+2
9404 00002CC9 E421              <1>       IN     AL,INTA01           ; LET INTERRUPTS PASS THRU TO
9405 00002CCB 24FB              <1>       AND    AL,0FBH             ;  SECOND CHIP
9406                            <1>       ;JMP   $+2
9407                            <1>       IODELAY
9408 00002CCD EB00              <2>  jmp short $+2
9409 00002CCF EB00              <2>  jmp short $+2
9410 00002CD1 E621              <1>       OUT    INTA01,AL
9411 00002CD3 FB                <1>       STI
9412 00002CD4 E8B7010000        <1>       CALL   NOT_BUSY           ; WAIT FOR CARD
9413 00002CD9 752B              <1>       JNZ    short CD_ERR       ; BAD CARD
9414                            <1>       ;MOV   DX, HF_PORT+7
9415 00002CDB 668B15[146B0000]  <1>       mov    dx, [HF_PORT]
9416 00002CE2 80C207            <1>       add    dl, 7
```

```
9417 00002CE5 B090                <1>       MOV   AL,DIAG_CMD         ; START DIAGNOSE
9418 00002CE7 EE                  <1>       OUT   DX,AL
9419 00002CE8 E8A3010000          <1>       CALL  NOT_BUSY            ; WAIT FOR IT TO COMPLETE
9420 00002CED B480                <1>       MOV   AH,TIME_OUT
9421 00002CEF 7517                <1>       JNZ   short CD_EXIT             ; TIME OUT ON DIAGNOSTIC
9422                              <1>       ;MOV  DX,HF_PORT+1        ; GET ERROR REGISTER
9423 00002CF1 668B15[146B0000]    <1>       mov   dx, [HF_PORT]
9424 00002CF8 FEC2                <1>       inc   dl
9425 00002CFA EC                  <1>       IN    AL,DX
9426 00002CFB A2[F6700000]        <1>       MOV   [HF_ERROR],AL       ; SAVE IT
9427 00002D00 B400                <1>       MOV   AH,0
9428 00002D02 3C01                <1>       CMP   AL,1                ; CHECK FOR ALL OK
9429 00002D04 7402                <1>       JE    SHORT CD_EXIT
9430 00002D06 B420                <1> CD_ERR: MOV AH,BAD_CNTLR
9431                              <1> CD_EXIT:
9432 00002D08 8825[FF700000]      <1>       MOV   [DISK_STATUS1],AH
9433 00002D0E C3                  <1>       RETn
9434                              <1>
9435                              <1> ;---------------------------------------
9436                              <1> ; COMMANDI                        :
9437                              <1> ;     REPEATEDLY INPUTS DATA TILL     :
9438                              <1> ;     NSECTOR RETURNS ZERO            :
9439                              <1> ;---------------------------------------
9440                              <1> COMMANDI:
9441 00002D0F E85A020000          <1>       CALL  CHECK_DMA           ; CHECK 64K BOUNDARY ERROR
9442 00002D14 724D                <1>       JC    short CMD_ABORT
9443                              <1>       ;MOV  DI,BX
9444 00002D16 89DF                <1>       mov   edi, ebx ; 21/02/2015
9445 00002D18 E8BE000000          <1>       CALL  COMMAND             ; OUTPUT COMMAND
9446 00002D1D 7544                <1>       JNZ   short CMD_ABORT
9447                              <1> CMD_I1:
9448 00002D1F E830010000          <1>       CALL  _WAIT               ; WAIT FOR DATA REQUEST INTERRUPT
9449 00002D24 753D                <1>       JNZ   short TM_OUT        ; TIME OUT
9450                              <1>       ;MOV  CX,256              ; SECTOR SIZE IN WORDS
9451 00002D26 B900010000          <1>       mov   ecx, 256 ; 21/02/2015
9452                              <1>       ;MOV  DX,HF_PORT
9453 00002D2B 668B15[146B0000]    <1>       mov   dx,[HF_PORT]
9454 00002D32 FA                  <1>       CLI
9455 00002D33 FC                  <1>       CLD
9456 00002D34 F3666D              <1>       REP   INSW                ; GET THE SECTOR
9457 00002D37 FB                  <1>       STI
9458 00002D38 F645FE02            <1>       TEST  byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
9459 00002D3C 7419                <1>       JZ    CMD_I3
9460 00002D3E E87A010000          <1>       CALL  WAIT_DRQ            ; WAIT FOR DATA REQUEST
9461 00002D43 721E                <1>       JC    short TM_OUT
9462                              <1>       ;MOV  DX,HF_PORT
9463 00002D45 668B15[146B0000]    <1>       mov   dx,[HF_PORT]
9464                              <1>       ;MOV  CX,4                ; GET ECC BYTES
9465 00002D4C B904000000          <1>       mov   ecx, 4 ; mov cx, 4
9466 00002D51 EC                  <1> CMD_I2: IN  AL,DX
9467                              <1>       ;MOV  [ES:DI],AL          ; GO SLOW FOR BOARD
9468 00002D52 8807                <1>       mov   [edi], al ; 21/02/2015
9469 00002D54 47                  <1>       INC   eDI
9470 00002D55 E2FA                <1>       LOOP  CMD_I2
9471 00002D57 E88C010000          <1> CMD_I3: CALL CHECK_STATUS
9472 00002D5C 7505                <1>       JNZ   short CMD_ABORT           ; ERROR RETURNED
9473 00002D5E FE4DF9              <1>       DEC   byte [CMD_BLOCK+1]  ; CHECK FOR MORE
9474 00002D61 75BC                <1>       JNZ   SHORT CMD_I1
9475                              <1> CMD_ABORT:
9476 00002D63 C3                  <1> TM_OUT: RETn
9477                              <1>
9478                              <1> ;---------------------------------------
9479                              <1> ; COMMANDO                        :
9480                              <1> ;     REPEATEDLY OUTPUTS DATA TILL    :
9481                              <1> ;     NSECTOR RETURNS ZERO            :
9482                              <1> ;---------------------------------------
9483                              <1> COMMANDO:
9484 00002D64 E805020000          <1>       CALL  CHECK_DMA           ; CHECK 64K BOUNDARY ERROR
9485 00002D69 72F8                <1>       JC    short CMD_ABORT
9486 00002D6B 89DE                <1> CMD_OF: MOV eSI,eBX ; 21/02/2015
9487 00002D6D E869000000          <1>       CALL  COMMAND             ; OUTPUT COMMAND
9488 00002D72 75EF                <1>       JNZ   short CMD_ABORT
9489 00002D74 E844010000          <1>       CALL  WAIT_DRQ            ; WAIT FOR DATA REQUEST
9490 00002D79 72E8                <1>       JC    short TM_OUT              ; TOO LONG
9491                              <1> CMD_O1: ;PUSH     DS
9492                              <1>       ;PUSH ES                  ; MOVE ES TO DS
9493                              <1>       ;POP  DS
9494                              <1>       ;MOV  CX,256              ; PUT THE DATA OUT TO THE CARD
9495                              <1>       ;MOV  DX,HF_PORT
9496                              <1>       ; 01/02/2015
9497 00002D7B 668B15[146B0000]    <1>       mov   dx, [HF_PORT]
9498                              <1>       ;push es
9499                              <1>       ;pop  ds
9500                              <1>       ;mov  cx, 256
9501 00002D82 B900010000          <1>       mov   ecx, 256 ; 21/02/2015
9502 00002D87 FA                  <1>       CLI
9503 00002D88 FC                  <1>       CLD
9504 00002D89 F3666F              <1>       REP   OUTSW
9505 00002D8C FB                  <1>       STI
9506                              <1>       ;POP  DS                  ; RESTORE DS
9507 00002D8D F645FE02            <1>       TEST  byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
9508 00002D91 7419                <1>       JZ    short CMD_O3
9509 00002D93 E825010000          <1>       CALL  WAIT_DRQ            ; WAIT FOR DATA REQUEST
9510 00002D98 72C9                <1>       JC    short TM_OUT
9511                              <1>       ;MOV  DX,HF_PORT
9512 00002D9A 668B15[146B0000]    <1>       mov   dx, [HF_PORT]
9513                              <1>       ;MOV  CX,4                ; OUTPUT THE ECC BYTES
9514 00002DA1 B904000000          <1>       mov   ecx, 4  ; mov cx, 4
9515                              <1> CMD_O2: ;MOV AL,[ES:SI]
9516 00002DA6 8A06                <1>       mov   al, [esi]
9517 00002DA8 EE                  <1>       OUT   DX,AL
9518 00002DA9 46                  <1>       INC   eSI
9519 00002DAA E2FA                <1>       LOOP  CMD_O2
9520                              <1> CMD_O3:
9521 00002DAC E8A3000000          <1>       CALL  _WAIT               ; WAIT FOR SECTOR COMPLETE INTERRUPT
```

```
9522 00002DB1 75B0                <1>       JNZ    short TM_OUT        ; ERROR RETURNED
9523 00002DB3 E830010000          <1>       CALL   CHECK_STATUS
9524 00002DB8 75A9                <1>       JNZ    short CMD_ABORT
9525 00002DBA F605[F5700000]08    <1>       TEST   byte [HF_STATUS],ST_DRQ   ; CHECK FOR MORE
9526 00002DC1 75B8                <1>       JNZ    SHORT CMD_O1
9527                              <1>       ;MOV   DX,HF_PORT+2        ; CHECK RESIDUAL SECTOR COUNT
9528 00002DC3 668B15[146B0000]    <1>       mov    dx, [HF_PORT]
9529                              <1>       ;add   dl, 2
9530 00002DCA FEC2                <1>       inc    dl
9531 00002DCC FEC2                <1>       inc    dl
9532 00002DCE EC                  <1>       IN     AL,DX               ;
9533 00002DCF A8FF                <1>       TEST   AL,0FFH             ;
9534 00002DD1 7407                <1>       JZ     short CMD_O4              ; COUNT = 0  OK
9535 00002DD3 C605[FF700000]BB    <1>       MOV    byte [DISK_STATUS1],UNDEF_ERR
9536                              <1>                                  ; OPERATION ABORTED - PARTIAL TRANSFER
9537                              <1> CMD_O4:
9538 00002DDA C3                  <1>       RETn
9539                              <1>
9540                              <1> ;--------------------------------------------------
9541                              <1> ; COMMAND                                       :
9542                              <1> ;     THIS ROUTINE OUTPUTS THE COMMAND BLOCK        :
9543                              <1> ; OUTPUT                                        :
9544                              <1> ;     BL = STATUS                               :
9545                              <1> ;     BH = ERROR REGISTER                       :
9546                              <1> ;--------------------------------------------------
9547                              <1>
9548                              <1> COMMAND:
9549 00002DDB 53                  <1>       PUSH   eBX                 ; WAIT FOR SEEK COMPLETE AND READY
9550                              <1>       ;;MOV  CX,DELAY_2          ; SET INITIAL DELAY BEFORE TEST
9551                              <1> COMMAND1:
9552                              <1>       ;;PUSH CX                  ; SAVE LOOP COUNT
9553 00002DDC E87FFEFFFF          <1>       CALL   TST_RDY             ; CHECK DRIVE READY
9554                              <1>       ;;POP  CX
9555 00002DE1 7419                <1>       JZ     short COMMAND2            ; DRIVE IS READY
9556 00002DE3 803D[FF700000]80    <1>         CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
9557                              <1>       ;JZ    short CMD_TIMEOUT
9558                              <1>       ;;LOOP COMMAND1           ; KEEP TRYING FOR A WHILE
9559                              <1>       ;JMP   SHORT COMMAND4            ; ITS NOT GOING TO GET READY
9560 00002DEA 7507                <1>       jne    short COMMAND4
9561                              <1> CMD_TIMEOUT:
9562 00002DEC C605[FF700000]20    <1>       MOV    byte [DISK_STATUS1],BAD_CNTLR
9563                              <1> COMMAND4:
9564 00002DF3 5B                  <1>       POP    eBX
9565 00002DF4 803D[FF700000]00    <1>         CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
9566 00002DFB C3                  <1>       RETn
9567                              <1> COMMAND2:
9568 00002DFC 5B                  <1>       POP    eBX
9569 00002DFD 57                  <1>       PUSH   eDI
9570 00002DFE C605[F7700000]00    <1>       MOV    byte [HF_INT_FLAG],0      ; RESET INTERRUPT FLAG
9571 00002E05 FA                  <1>       CLI                        ; INHIBIT INTERRUPTS WHILE CHANGING MASK
9572 00002E06 E4A1                <1>       IN     AL,INTB01           ; TURN ON SECOND INTERRUPT CHIP
9573                              <1>       ;AND   AL,0BFH
9574 00002E08 243F                <1>       and    al, 3Fh                   ; Enable IRQ 14 & 15
9575                              <1>       ;JMP   $+2
9576                              <1>       IODELAY
9577 00002E0A EB00                <2>  jmp short $+2
9578 00002E0C EB00                <2>  jmp short $+2
9579 00002E0E E6A1                <1>       OUT    INTB01,AL
9580 00002E10 E421                <1>       IN     AL,INTA01           ; LET INTERRUPTS PASS THRU TO
9581 00002E12 24FB                <1>       AND    AL,0FBH             ;  SECOND CHIP
9582                              <1>       ;JMP   $+2
9583                              <1>       IODELAY
9584 00002E14 EB00                <2>  jmp short $+2
9585 00002E16 EB00                <2>  jmp short $+2
9586 00002E18 E621                <1>       OUT    INTA01,AL
9587 00002E1A FB                  <1>       STI
9588 00002E1B 31FF                <1>       XOR    eDI,eDI                   ; INDEX THE COMMAND TABLE
9589                              <1>       ;MOV   DX,HF_PORT+1        ; DISK ADDRESS
9590 00002E1D 668B15[146B0000]    <1>       mov    dx, [HF_PORT]
9591 00002E24 FEC2                <1>       inc    dl
9592 00002E26 F605[01710000]C0    <1>       TEST   byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
9593 00002E2D 7411                <1>       JZ     short COMMAND3
9594 00002E2F 8A45FE              <1>       MOV    AL, [CMD_BLOCK+6]   ; YES-GET OPERATION CODE
9595 00002E32 24F0                <1>       AND    AL,0F0H             ; GET RID OF MODIFIERS
9596 00002E34 3C20                <1>       CMP    AL,20H              ; 20H-40H IS READ, WRITE, VERIFY
9597 00002E36 7208                <1>       JB     short COMMAND3
9598 00002E38 3C40                <1>       CMP    AL,40H
9599 00002E3A 7704                <1>       JA     short COMMAND3
9600 00002E3C 804DFE01            <1>       OR     byte [CMD_BLOCK+6],NO_RETRIES
9601                              <1>                                  ; VALID OPERATION FOR RETRY SUPPRESS
9602                              <1> COMMAND3:
9603 00002E40 8A443DF8            <1>       MOV    AL,[CMD_BLOCK+eDI]  ; GET THE COMMAND STRING BYTE
9604 00002E44 EE                  <1>       OUT    DX,AL               ; GIVE IT TO CONTROLLER
9605                              <1>       IODELAY
9606 00002E45 EB00                <2>  jmp short $+2
9607 00002E47 EB00                <2>  jmp short $+2
9608 00002E49 47                  <1>       INC    eDI                 ; NEXT BYTE IN COMMAND BLOCK
9609 00002E4A 6642                <1>       INC    DX                  ; NEXT DISK ADAPTER REGISTER
9610 00002E4C 6683FF07            <1>       cmp    di, 7  ; 1/1/2015   ; ALL DONE?
9611 00002E50 75EE                <1>       JNZ    short COMMAND3            ; NO--GO DO NEXT ONE
9612 00002E52 5F                  <1>       POP    eDI
9613 00002E53 C3                  <1>       RETn                       ; ZERO FLAG IS SET
9614                              <1>
9615                              <1> ;CMD_TIMEOUT:
9616                              <1> ;     MOV    byte [DISK_STATUS1],BAD_CNTLR
9617                              <1> ;COMMAND4:
9618                              <1> ;     POP    BX
9619                              <1> ;     CMP    [DISK_STATUS1],0    ; SET CONDITION CODE FOR CALLER
9620                              <1> ;     RETn
9621                              <1>
9622                              <1> ;----------------------------------------
9623                              <1> ;     WAIT FOR INTERRUPT        :
9624                              <1> ;----------------------------------------
9625                              <1> ;WAIT:
9626                              <1> _WAIT:
```

```
9627 00002E54 FB                <1>       STI                         ; MAKE SURE INTERRUPTS ARE ON
9628                            <1>       ;SUB   CX,CX                ; SET INITIAL DELAY BEFORE TEST
9629                            <1>       ;CLC
9630                            <1>       ;MOV   AX,9000H             ; DEVICE WAIT INTERRUPT
9631                            <1>       ;INT   15H
9632                            <1>       ;JC    WT2                  ; DEVICE TIMED OUT
9633                            <1>       ;MOV   BL,DELAY_1           ; SET DELAY COUNT
9634                            <1>
9635                            <1>       ;mov   bl, WAIT_HDU_INT_HI
9636                            <1>       ;; 21/02/2015
9637                            <1>       ;;mov  bl, WAIT_HDU_INT_HI + 1
9638                            <1>       ;;mov  cx, WAIT_HDU_INT_LO
9639 00002E55 B915160500       <1>       mov    ecx, WAIT_HDU_INT_LH
9640                            <1>                                   ; (AWARD BIOS -> WAIT_FOR_MEM)
9641                            <1> ;-----      WAIT LOOP
9642                            <1>
9643                            <1> WT1:
9644                            <1>       ;TEST  byte [HF_INT_FLAG],80H    ; TEST FOR INTERRUPT
9645 00002E5A F605[F7700000]C0  <1>       test   byte [HF_INT_FLAG],0C0h
9646                            <1>       ;LOOPZ WT1
9647 00002E61 7517              <1>       JNZ    short WT3            ; INTERRUPT--LETS GO
9648                            <1>       ;DEC   BL
9649                            <1>       ;JNZ   short WT1            ; KEEP TRYING FOR A WHILE
9650                            <1>
9651                            <1> WT1_hi:
9652 00002E63 E461              <1>       in     al, SYS1 ; 61h (PORT_B)   ; wait for lo to hi
9653 00002E65 A810              <1>       test   al, 10h                   ; transition on memory
9654 00002E67 75FA              <1>       jnz    short WT1_hi         ; refresh.
9655                            <1> WT1_lo:
9656 00002E69 E461              <1>       in     al, SYS1             ; 061h (PORT_B)
9657 00002E6B A810              <1>       test   al, 10h
9658 00002E6D 74FA              <1>       jz     short WT1_lo
9659 00002E6F E2E9              <1>       loop   WT1
9660                            <1>       ;;or   bl, bl
9661                            <1>       ;;jz   short WT2
9662                            <1>       ;;dec  bl
9663                            <1>       ;;jmp  short WT1
9664                            <1>       ;dec   bl
9665                            <1>       ;jnz   short WT1
9666                            <1>
9667 00002E71 C605[FF700000]80  <1> WT2:  MOV    byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
9668 00002E78 EB0E              <1>       JMP    SHORT WT4
9669 00002E7A C605[FF700000]00  <1> WT3:  MOV    byte [DISK_STATUS1],0
9670 00002E81 C605[F7700000]00  <1>       MOV    byte [HF_INT_FLAG],0
9671 00002E88 803D[FF700000]00  <1> WT4:  CMP    byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
9672 00002E8F C3                <1>       RETn
9673                            <1>
9674                            <1> ;----------------------------------------
9675                            <1> ;     WAIT FOR CONTROLLER NOT BUSY     :
9676                            <1> ;----------------------------------------
9677                            <1> NOT_BUSY:
9678 00002E90 FB                <1>       STI                         ; MAKE SURE INTERRUPTS ARE ON
9679                            <1>       ;PUSH  eBX
9680                            <1>       ;SUB   CX,CX                ; SET INITIAL DELAY BEFORE TEST
9681 00002E91 668B15[146B0000]  <1>       mov    DX, [HF_PORT]
9682 00002E98 80C207            <1>       add    dl, 7                ; Status port (HF_PORT+7)
9683                            <1>       ;MOV   BL,DELAY_1
9684                            <1>                                   ; wait for 10 seconds
9685                            <1>       ;mov   cx, WAIT_HDU_INT_LO ; 1615h
9686                            <1>       ;;mov  bl, WAIT_HDU_INT_HI ;   05h
9687                            <1>       ;mov   bl, WAIT_HDU_INT_HI + 1
9688 00002E9B B915160500        <1>       mov    ecx, WAIT_HDU_INT_LH  ; 21/02/2015
9689                            <1>       ;
9690                            <1> ;;      mov     byte [wait_count], 0    ; Reset wait counter
9691                            <1> NB1:
9692 00002EA0 EC                <1>       IN     AL,DX                ; CHECK STATUS
9693                            <1>       ;TEST  AL,ST_BUSY
9694 00002EA1 2480              <1>       and    al, ST_BUSY
9695                            <1>       ;LOOPNZ      NB1
9696 00002EA3 7410              <1>       JZ     short NB2            ; NOT BUSY--LETS GO
9697                            <1>       ;DEC   BL
9698                            <1>       ;JNZ   short NB1            ; KEEP TRYING FOR A WHILE
9699                            <1>
9700 00002EA5 E461              <1> NB1_hi: IN   AL,SYS1                   ; wait for hi to lo
9701 00002EA7 A810              <1>       TEST   AL,010H                   ; transition on memory
9702 00002EA9 75FA              <1>       JNZ    SHORT NB1_hi         ; refresh.
9703 00002EAB E461              <1> NB1_lo: IN   AL,SYS1
9704 00002EAD A810              <1>       TEST   AL,010H
9705 00002EAF 74FA              <1>       JZ     short NB1_lo
9706 00002EB1 E2ED              <1>       LOOP   NB1
9707                            <1>       ;dec   bl
9708                            <1>       ;jnz   short NB1
9709                            <1>       ;
9710                            <1> ;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
9711                            <1> ;;    jb     short NB1
9712                            <1>       ;
9713                            <1>       ;MOV   [DISK_STATUS1],TIME_OUT   ; REPORT TIME OUT ERROR
9714                            <1>       ;JMP   SHORT NB3
9715 00002EB3 B080              <1>       mov    al, TIME_OUT
9716                            <1> NB2:
9717                            <1>       ;MOV   byte [DISK_STATUS1],0
9718                            <1> ;NB3:
9719                            <1>       ;POP   eBX
9720 00002EB5 A2[FF700000]      <1>       mov    [DISK_STATUS1], al  ;;; will be set after return
9721                            <1>       ;CMP   byte [DISK_STATUS1],0     ; SET CONDITION CODE FOR CALLER
9722 00002EBA 08C0              <1>       or     al, al               ; (zf = 0 --> timeout)
9723 00002EBC C3                <1>       RETn
9724                            <1>
9725                            <1> ;----------------------------------------
9726                            <1> ;     WAIT FOR DATA REQUEST            :
9727                            <1> ;----------------------------------------
9728                            <1> WAIT_DRQ:
9729                            <1>       ;MOV   CX,DELAY_3
9730                            <1>       ;MOV   DX,HF_PORT+7
9731 00002EBD 668B15[146B0000]  <1>       mov    dx, [HF_PORT]
```

```
9732 00002EC4 80C207              <1>       add    dl, 7
9733                              <1>       ;;MOV  bl, WAIT_HDU_DRQ_HI ; 0
9734                              <1>       ;MOV   cx, WAIT_HDU_DRQ_LO ; 1000 (30 milli seconds)
9735                              <1>                                  ; (but it is written as 2000
9736                              <1>                                  ; micro seconds in ATORGS.ASM file
9737                              <1>                                  ; of Award Bios - 1999, D1A0622)
9738 00002EC7 B9E8030000          <1>       mov    ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
9739 00002ECC EC                  <1> WQ_1: IN     AL,DX               ; GET STATUS
9740 00002ECD A808                <1>       TEST   AL,ST_DRQ           ; WAIT FOR DRQ
9741 00002ECF 7516                <1>       JNZ    short WQ_OK
9742                              <1>       ;LOOP  WQ_1                ; KEEP TRYING FOR A SHORT WHILE
9743                              <1> WQ_hi:
9744 00002ED1 E461                <1>       IN     AL,SYS1                   ; wait for hi to lo
9745 00002ED3 A810                <1>       TEST   AL,010H                   ; transition on memory
9746 00002ED5 75FA                <1>       JNZ    SHORT WQ_hi         ; refresh.
9747 00002ED7 E461                <1> WQ_lo:  IN      AL,SYS1
9748 00002ED9 A810                <1>       TEST   AL,010H
9749 00002EDB 74FA                <1>       JZ     SHORT WQ_lo
9750 00002EDD E2ED                <1>       LOOP   WQ_1
9751                              <1>
9752 00002EDF C605[FF700000]80    <1>         MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
9753 00002EE6 F9                  <1>       STC
9754                              <1> WQ_OK:
9755 00002EE7 C3                  <1>       RETn
9756                              <1> ;WQ_OK:      ;CLC
9757                              <1> ;     RETn
9758                              <1>
9759                              <1> ;---------------------------------------
9760                              <1> ;     CHECK FIXED DISK STATUS   :
9761                              <1> ;---------------------------------------
9762                              <1> CHECK_STATUS:
9763 00002EE8 E813000000          <1>       CALL   CHECK_ST            ; CHECK THE STATUS BYTE
9764 00002EED 7509                <1>       JNZ    short CHECK_S1            ; AN ERROR WAS FOUND
9765 00002EEF A801                <1>       TEST   AL,ST_ERROR         ; WERE THERE ANY OTHER ERRORS
9766 00002EF1 7405                <1>       JZ     short CHECK_S1            ; NO ERROR REPORTED
9767 00002EF3 E847000000          <1>       CALL   CHECK_ER            ; ERROR REPORTED
9768                              <1> CHECK_S1:
9769 00002EF8 803D[FF700000]00    <1>       CMP    byte [DISK_STATUS1],0     ; SET STATUS FOR CALLER
9770 00002EFF C3                  <1>       RETn
9771                              <1>
9772                              <1> ;---------------------------------------
9773                              <1> ;     CHECK FIXED DISK STATUS BYTE     :
9774                              <1> ;---------------------------------------
9775                              <1> CHECK_ST:
9776                              <1>       ;MOV   DX,HF_PORT+7        ; GET THE STATUS
9777 00002F00 668B15[146B0000]    <1>       mov    dx, [HF_PORT]
9778 00002F07 80C207              <1>       add    dl, 7
9779 00002F0A EC                  <1>       IN     AL,DX
9780 00002F0B A2[F5700000]        <1>       MOV    [HF_STATUS],AL
9781 00002F10 B400                <1>       MOV    AH,0
9782 00002F12 A880                <1>       TEST   AL,ST_BUSY          ; IF STILL BUSY
9783 00002F14 751A                <1>       JNZ    short CKST_EXIT           ;  REPORT OK
9784 00002F16 B4CC                <1>       MOV    AH,WRITE_FAULT
9785 00002F18 A820                <1>       TEST   AL,ST_WRT_FLT       ; CHECK FOR WRITE FAULT
9786 00002F1A 7514                <1>       JNZ    short CKST_EXIT
9787 00002F1C B4AA                <1>       MOV    AH,NOT_RDY
9788 00002F1E A840                <1>       TEST   AL,ST_READY         ; CHECK FOR NOT READY
9789 00002F20 740E                <1>       JZ     short CKST_EXIT
9790 00002F22 B440                <1>       MOV    AH,BAD_SEEK
9791 00002F24 A810                <1>       TEST   AL,ST_SEEK_COMPL    ; CHECK FOR SEEK NOT COMPLETE
9792 00002F26 7408                <1>       JZ     short CKST_EXIT
9793 00002F28 B411                <1>       MOV    AH,DATA_CORRECTED
9794 00002F2A A804                <1>       TEST   AL,ST_CORRCTD       ; CHECK FOR CORRECTED ECC
9795 00002F2C 7502                <1>       JNZ    short CKST_EXIT
9796 00002F2E B400                <1>       MOV    AH,0
9797                              <1> CKST_EXIT:
9798 00002F30 8825[FF700000]      <1>       MOV    [DISK_STATUS1],AH   ; SET ERROR FLAG
9799 00002F36 80FC11              <1>       CMP    AH,DATA_CORRECTED   ; KEEP GOING WITH DATA CORRECTED
9800 00002F39 7403                <1>       JZ     short CKST_EX1
9801 00002F3B 80FC00              <1>       CMP    AH,0
9802                              <1> CKST_EX1:
9803 00002F3E C3                  <1>       RETn
9804                              <1>
9805                              <1> ;---------------------------------------
9806                              <1> ;     CHECK FIXED DISK ERROR REGISTER :
9807                              <1> ;---------------------------------------
9808                              <1> CHECK_ER:
9809                              <1>       ;MOV   DX, HF_PORT+1       ; GET THE ERROR REGISTER
9810 00002F3F 668B15[146B0000]    <1>       mov    dx, [HF_PORT]       ;
9811 00002F46 FEC2                <1>       inc    dl
9812 00002F48 EC                  <1>       IN     AL,DX
9813 00002F49 A2[F6700000]        <1>       MOV    [HF_ERROR],AL
9814 00002F4E 53                  <1>       PUSH   eBX ; 21/02/2015
9815 00002F4F B908000000          <1>       MOV    eCX,8               ; TEST ALL 8 BITS
9816 00002F54 D0E0                <1> CK1:  SHL    AL,1                ; MOVE NEXT ERROR BIT TO CARRY
9817 00002F56 7202                <1>       JC     short CK2           ; FOUND THE ERROR
9818 00002F58 E2FA                <1>       LOOP   CK1                 ; KEEP TRYING
9819 00002F5A BB[086B0000]        <1> CK2:  MOV    eBX, ERR_TBL        ; COMPUTE ADDRESS OF
9820 00002F5F 01CB                <1>       ADD    eBX,eCX                   ; ERROR CODE
9821                              <1>       ;;MOV  AH,BYTE [CS:BX]           ; GET ERROR CODE
9822                              <1>       ;mov   ah, [bx]
9823 00002F61 8A23                <1>       mov    ah, [ebx] ; 21/02/2015
9824 00002F63 8825[FF700000]      <1> CKEX: MOV    [DISK_STATUS1],AH   ; SAVE ERROR CODE
9825 00002F69 5B                  <1>       POP    eBX
9826 00002F6A 80FC00              <1>       CMP    AH,0
9827 00002F6D C3                  <1>       RETn
9828                              <1>
9829                              <1> ;--------------------------------------------------
9830                              <1> ; CHECK_DMA                                  :
9831                              <1> ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
9832                              <1> ;   FIT WITHOUT SEGMENT OVERFLOW.                    :
9833                              <1> ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X  :
9834                              <1> ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)     :
9835                              <1> ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
9836                              <1> ;  -ERROR OTHERWISE                                  :
```

```
9837                                    <1> ;--------------------------------------------------
9838                                    <1> CHECK_DMA:
9839 00002F6E 6650                     <1>       PUSH   AX                   ; SAVE REGISTERS
9840 00002F70 66B80080                 <1>       MOV    AX,8000H             ; AH = MAX # SECTORS AL = MAX OFFSET
9841 00002F74 F645FE02                 <1>         TEST    byte [CMD_BLOCK+6],ECC_MODE
9842 00002F78 7404                     <1>       JZ     short CKD1
9843 00002F7A 66B8047F                 <1>       MOV    AX,7F04H             ; ECC IS 4 MORE BYTES
9844 00002F7E 3A65F9                   <1> CKD1: CMP    AH, [CMD_BLOCK+1]    ; NUMBER OF SECTORS
9845 00002F81 7706                     <1>       JA     short CKDOK          ; IT WILL FIT
9846 00002F83 7208                     <1>       JB     short CKDERR         ; TOO MANY
9847 00002F85 38D8                     <1>       CMP    AL,BL                ; CHECK OFFSET ON MAX SECTORS
9848 00002F87 7204                     <1>       JB     short CKDERR         ; ERROR
9849 00002F89 F8                       <1> CKDOK:       CLC                        ; CLEAR CARRY
9850 00002F8A 6658                     <1>       POP    AX
9851 00002F8C C3                       <1>       RETn                        ; NORMAL RETURN
9852 00002F8D F9                       <1> CKDERR: STC                       ; INDICATE ERROR
9853 00002F8E C605[FF700000]09         <1>         MOV     byte [DISK_STATUS1],DMA_BOUNDARY
9854 00002F95 6658                     <1>       POP    AX
9855 00002F97 C3                       <1>       RETn
9856                                    <1>
9857                                    <1> ;---------------------------------------
9858                                    <1> ;     SET UP ES:BX-> DISK PARMS :
9859                                    <1> ;---------------------------------------
9860                                    <1>
9861                                    <1> ; INPUT -> DL = 0 based drive number
9862                                    <1> ; OUTPUT -> ES:BX = disk parameter table address
9863                                    <1>
9864                                    <1> GET_VEC:
9865                                    <1>       ;SUB   AX,AX               ; GET DISK PARAMETER ADDRESS
9866                                    <1>       ;MOV   ES,AX
9867                                    <1>       ;TEST  DL,1
9868                                    <1>       ;JZ    short GV_0
9869                                    <1> ;     LES    BX,[HF1_TBL_VEC]    ; ES:BX -> DRIVE PARAMETERS
9870                                    <1> ;     JMP    SHORT GV_EXIT
9871                                    <1> ;GV_0:
9872                                    <1> ;     LES    BX,[HF_TBL_VEC]           ; ES:BX -> DRIVE PARAMETERS
9873                                    <1> ;
9874                                    <1>       ;xor   bh, bh
9875 00002F98 31DB                     <1>       xor    ebx, ebx
9876 00002F9A 88D3                     <1>       mov    bl, dl
9877                                    <1>       ;;02/01/2015
9878                                    <1>       ;;shl  bl, 1               ; port address offset
9879                                    <1>       ;;mov  ax, [bx+hd_ports]   ; Base port address (1F0h, 170h)
9880                                    <1>       ;;shl  bl, 1               ; dpt pointer offset
9881 00002F9C C0E302                   <1>       shl    bl, 2  ;;
9882                                    <1>       ;add   bx, HF_TBL_VEC            ; Disk parameter table pointer
9883 00002F9F 81C3[04710000]           <1>       add    ebx, HF_TBL_VEC ; 21/02/2015
9884                                    <1>       ;push  word [bx+2]         ; dpt segment
9885                                    <1>       ;pop   es
9886                                    <1>       ;mov   bx, [bx]            ; dpt offset
9887 00002FA5 8B1B                     <1>       mov    ebx, [ebx]
9888                                    <1> ;GV_EXIT:
9889 00002FA7 C3                       <1>       RETn
9890                                    <1>
9891                                    <1> hdc1_int: ; 21/02/2015
9892                                    <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
9893                                    <1> ;                                                 :
9894                                    <1> ;     FIXED DISK INTERRUPT ROUTINE                      :
9895                                    <1> ;                                                 :
9896                                    <1> ;-----------------------------------------------------------------
9897                                    <1>
9898                                    <1> ; 22/12/2014
9899                                    <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
9900                                    <1> ;      '11/15/85'
9901                                    <1> ; AWARD BIOS 1999 (D1A0622)
9902                                    <1> ;     Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
9903                                    <1>
9904                                    <1> ;int_76h:
9905                                    <1> HD_INT:
9906 00002FA8 6650                     <1>       PUSH   AX
9907 00002FAA 1E                       <1>       PUSH   DS
9908                                    <1>       ;CALL  DDS
9909                                    <1>       ; 21/02/2015 (32 bit, 386 pm modification)
9910 00002FAB 66B81000                 <1>       mov    ax, KDATA
9911 00002FAF 8ED8                     <1>       mov    ds, ax
9912                                    <1>       ;
9913                                    <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
9914                                    <1>         ;mov     byte [CS:HF_INT_FLAG], 0FFh
9915 00002FB1 C605[F7700000]FF         <1>       mov    byte [HF_INT_FLAG], 0FFh
9916                                    <1>       ;
9917 00002FB8 6652                     <1>       push   dx
9918 00002FBA 66BAF701                 <1>       mov    dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
9919                                    <1>                                  ; Clear Controller
9920                                    <1> Clear_IRQ1415:                        ; (Award BIOS - 1999)
9921 00002FBE EC                       <1>       in     al, dx              ;
9922 00002FBF 665A                     <1>       pop    dx
9923                                    <1>       NEWIODELAY
9924 00002FC1 E6EB                     <2>  out 0ebh,al
9925                                    <1>       ;
9926 00002FC3 B020                     <1>       MOV    AL,EOI              ; NON-SPECIFIC END OF INTERRUPT
9927 00002FC5 E6A0                     <1>       OUT    INTB00,AL           ; FOR CONTROLLER #2
9928                                    <1>       ;JMP   $+2                 ; WAIT
9929                                    <1>       NEWIODELAY
9930 00002FC7 E6EB                     <2>  out 0ebh,al
9931 00002FC9 E620                     <1>       OUT    INTA00,AL           ; FOR CONTROLLER #1
9932 00002FCB 1F                       <1>       POP    DS
9933                                    <1>       ;STI                       ; RE-ENABLE INTERRUPTS
9934                                    <1>       ;MOV   AX,9100H            ; DEVICE POST
9935                                    <1>       ;INT   15H                 ;  INTERRUPT
9936                                    <1> irq15_iret: ; 25/02/2015
9937 00002FCC 6658                     <1>       POP    AX
9938 00002FCE CF                       <1>       IRETd                      ; RETURN FROM INTERRUPT
9939                                    <1>
9940                                    <1> hdc2_int: ; 21/02/2015
9941                                    <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++++++++++++++++
```

```
9942                            <1> ;                                                   :
9943                            <1> ;     FIXED DISK INTERRUPT ROUTINE                        :
9944                            <1> ;                                                   :
9945                            <1> ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
9946                            <1>
9947                            <1> ;int_77h:
9948                            <1> HD1_INT:
9949 00002FCF 6650             <1>       PUSH   AX
9950                            <1>       ; Check if that is a spurious IRQ (from slave PIC)
9951                            <1>       ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
9952 00002FD1 B00B             <1>       mov    al, 0Bh  ; In-Service Register
9953 00002FD3 E6A0             <1>       out    0A0h, al
9954 00002FD5 EB00             <1>         jmp short $+2
9955 00002FD7 EB00             <1>       jmp short $+2
9956 00002FD9 E4A0             <1>       in     al, 0A0h
9957 00002FDB 2480             <1>       and    al, 80h ; bit 7 (is it real IRQ 15 or fake?)
9958 00002FDD 74ED             <1>       jz     short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
9959                            <1>       ;
9960 00002FDF 1E               <1>       PUSH   DS
9961                            <1>       ;CALL  DDS
9962                            <1>       ; 21/02/2015 (32 bit, 386 pm modification)
9963 00002FE0 66B81000          <1>       mov    ax, KDATA
9964 00002FE4 8ED8              <1>       mov    ds, ax
9965                            <1>       ;
9966                            <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
9967                            <1>         ;or      byte [CS:HF_INT_FLAG],0C0h
9968 00002FE6 800D[F7700000]C0  <1>       or     byte [HF_INT_FLAG], 0C0h
9969                            <1>       ;
9970 00002FED 6652              <1>       push   dx
9971 00002FEF 66BA7701          <1>       mov    dx, HDC2_BASEPORT+7 ; Status Register (177h)
9972                            <1>                                  ; Clear Controller (Award BIOS 1999)
9973 00002FF3 EBC9              <1>       jmp    short Clear_IRQ1415
9974                            <1>
9975                            <1>
9976                            <1> ;%include 'diskdata.inc' ; 11/03/2015
9977                            <1> ;%include 'diskbss.inc' ; 11/03/2015
9978                            <1>
9979                            <1>
9980                            <1> ;////////////////////////////////////////////////////////////
9981                            <1> ;; END OF DISK I/O SYTEM ///
9982                                %include 'memory.inc'  ; 09/03/2015
9983                            <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
9984                            <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
9985                            <1> ; Last Modification: 18/10/2015 (!not completed!)
9986                            <1> ;
9987                            <1> ; Source code for NASM - Netwide Assembler (2.11)
9988                            <1>
9989                            <1> ; ///////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) ///////////////
9990                            <1>
9991                            <1> ;;04/11/2014 (unix386.s)
9992                            <1> ;PDE_A_PRESENT     equ    1            ; Present flag for PDE
9993                            <1> ;PDE_A_WRITE equ    2            ; Writable (write permission) flag
9994                            <1> ;PDE_A_USER  equ    4            ; User (non-system/kernel) page flag
9995                            <1> ;;
9996                            <1> ;PTE_A_PRESENT     equ    1            ; Present flag for PTE (bit 0)
9997                            <1> ;PTE_A_WRITE equ    2            ; Writable (write permission) flag (bit 1)
9998                            <1> ;PTE_A_USER  equ    4            ; User (non-system/kernel) page flag (bit 2)
9999                            <1> ;PTE_A_ACCESS   equ      32           ; Accessed flag (bit 5) ; 09/03/2015
10000                           <1>
10001                           <1> ; 27/04/2015
10002                           <1> ; 09/03/2015
10003                           <1> PAGE_SIZE    equ 4096            ; page size in bytes
10004                           <1> PAGE_SHIFT   equ 12              ; page table shift count
10005                           <1> PAGE_D_SHIFT       equ 22 ; 12 + 10    ; page directory shift count
10006                           <1> PAGE_OFF     equ 0FFFh           ; 12 bit byte offset in page frame
10007                           <1> PTE_MASK     equ 03FFh           ; page table entry mask
10008                           <1> PTE_DUPLICATED  equ 200h         ; duplicated page sign (AVL bit 0)
10009                           <1> PDE_A_CLEAR  equ 0F000h          ; to clear PDE attribute bits
10010                           <1> PTE_A_CLEAR  equ 0F000h          ; to clear PTE attribute bits
10011                           <1> LOGIC_SECT_SIZE equ 512                ; logical sector size
10012                           <1> ERR_MAJOR_PF equ 0E0h            ; major error: page fault
10013                           <1> ERR_MINOR_IM equ 1               ; insufficient (out of) memory
10014                           <1> ERR_MINOR_DSK      equ 2               ; disk read/write error
10015                           <1> ERR_MINOR_PV equ 3               ; protection violation
10016                           <1> SWP_DISK_READ_ERR equ 4
10017                           <1> SWP_DISK_NOT_PRESENT_ERR equ 5
10018                           <1> SWP_SECTOR_NOT_PRESENT_ERR equ 6
10019                           <1> SWP_NO_FREE_SPACE_ERR equ 7
10020                           <1> SWP_DISK_WRITE_ERR equ 8
10021                           <1> SWP_NO_PAGE_TO_SWAP_ERR equ 9
10022                           <1> PTE_A_ACCESS_BIT equ 5  ; Bit 5 (accessed flag)
10023                           <1> SECTOR_SHIFT    equ 3   ; sector shift (to convert page block number)
10024                           <1>
10025                           <1> ;
10026                           <1> ;; Retro Unix 386 v1 - paging method/principles
10027                           <1> ;;
10028                           <1> ;; 10/10/2014
10029                           <1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
10030                           <1> ;;
10031                           <1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map
10032                           <1> ;;    (virtual address = physical address)
10033                           <1> ;; KERNEL PAGE TABLES:
10034                           <1> ;;    Kernel page directory and all page tables are
10035                           <1> ;;    on memory as initialized, as equal to physical memory
10036                           <1> ;;    layout. Kernel pages can/must not be swapped out/in.
10037                           <1> ;;
10038                           <1> ;;    what for: User pages may be swapped out, when accessing
10039                           <1> ;;    a page in kernel/system mode, if it would be swapped out,
10040                           <1> ;;    kernel would have to swap it in! But it is also may be
10041                           <1> ;;    in use by a user process. (In system/kernel mode
10042                           <1> ;;    kernel can access all memory pages even if they are
10043                           <1> ;;    reserved/allocated for user processes. Swap out/in would
10044                           <1> ;;    cause conflicts.)
10045                           <1> ;;
10046                           <1> ;;    As result of these conditions,
```

```
10047                                  <1> ;;    all kernel pages must be initialized as equal to
10048                                  <1> ;;    physical layout for preventing page faults.
10049                                  <1> ;;    Also, calling "allocate page" procedure after
10050                                  <1> ;;    a page fault can cause another page fault (double fault)
10051                                  <1> ;;    if all kernel page tables would not be initialized.
10052                                  <1> ;;
10053                                  <1> ;;    [first_page] = Beginning of users space, as offset to
10054                                  <1> ;;    memory allocation table. (double word aligned)
10055                                  <1> ;;
10056                                  <1> ;;    [next_page] = first/next free space to be searched
10057                                  <1> ;;    as offset to memory allocation table. (dw aligned)
10058                                  <1> ;;
10059                                  <1> ;;    [last_page] = End of memory (users space), as offset
10060                                  <1> ;;    to memory allocation table. (double word aligned)
10061                                  <1> ;;
10062                                  <1> ;; USER PAGE TABLES:
10063                                  <1> ;;    Demand paging (& 'copy on write' allocation method) ...
10064                                  <1> ;;           'ready only' marked copies of the
10065                                  <1> ;;           parent process's page table entries (for
10066                                  <1> ;;           same physical memory).
10067                                  <1> ;;           (A page will be copied to a new page after
10068                                  <1> ;;            if it causes R/W page fault.)
10069                                  <1> ;;
10070                                  <1> ;;    Every user process has own (different)
10071                                  <1> ;;    page directory and page tables.
10072                                  <1> ;;
10073                                  <1> ;;    Code starts at virtual address 0, always.
10074                                  <1> ;;    (Initial value of EIP is 0 in user mode.)
10075                                  <1> ;;    (Programs can be written/developed as simple
10076                                  <1> ;;     flat memory programs.)
10077                                  <1> ;;
10078                                  <1> ;; MEMORY ALLOCATION STRATEGY:
10079                                  <1> ;;    Memory page will be allocated by kernel only
10080                                  <1> ;;          (in kernel/system mode only).
10081                                  <1> ;;    * After a
10082                                  <1> ;;      - 'not present' page fault
10083                                  <1> ;;      - 'writing attempt on read only page' page fault
10084                                  <1> ;;    * For loading (opening, reading) a file or disk/drive
10085                                  <1> ;;    * As responce to 'allocate additional memory blocks'
10086                                  <1> ;;      request by running process.
10087                                  <1> ;;    * While creating a process, allocating a new buffer,
10088                                  <1> ;;      new page tables etc.
10089                                  <1> ;;
10090                                  <1> ;;    At first,
10091                                  <1> ;;    - 'allocate page' procedure will be called;
10092                                  <1> ;,       if it will return with a valid (>0) physical address
10093                                  <1> ;;       (that means the relevant M.A.T. bit has been RESET)
10094                                  <1> ;;       relevant memory page/block will be cleared (zeroed).
10095                                  <1> ;;    - 'allocate page' will be called for allocating page
10096                                  <1> ;;       directory, page table and running space (data/code).
10097                                  <1> ;;    - every successful 'allocate page' call will decrease
10098                                  <1> ;;      'free_pages' count (pointer).
10099                                  <1> ;;    - 'out of (insufficient) memory error' will be returned
10100                                  <1> ;;      if 'free_pages' points to a ZERO.
10101                                  <1> ;;    - swapping out and swapping in (if it is not a new page)
10102                                  <1> ;;      procedures will be called as responce to 'out of memory'
10103                                  <1> ;;      error except errors caused by attribute conflicts.
10104                                  <1> ;;     (swapper functions)
10105                                  <1> ;;
10106                                  <1> ;;    At second,
10107                                  <1> ;;    - page directory entry will be updated then page table
10108                                  <1> ;;      entry will be updated.
10109                                  <1> ;;
10110                                  <1> ;; MEMORY ALLOCATION TABLE FORMAT:
10111                                  <1> ;;    - M.A.T. has a size according to available memory as
10112                                  <1> ;;      follows:
10113                                  <1> ;;            - 1 (allocation) bit per 1 page (4096 bytes)
10114                                  <1> ;;            - a bit with value of 0 means allocated page
10115                                  <1> ;;            - a bit with value of 1 means a free page
10116                                  <1> ;,    - 'free_pages' pointer holds count of free pages
10117                                  <1> ;;      depending on M.A.T.
10118                                  <1> ;;          (NOTE: Free page count will not be checked
10119                                  <1> ;;          again -on M.A.T.- after initialization.
10120                                  <1> ;;          Kernel will trust on initial count.)
10121                                  <1> ;,    - 'free_pages' count will be decreased by allocation
10122                                  <1> ;;      and it will be increased by deallocation procedures.
10123                                  <1> ;;
10124                                  <1> ;;    - Available memory will be calculated during
10125                                  <1> ;;      the kernel's initialization stage (in real mode).
10126                                  <1> ;;      Memory allocation table and kernel page tables
10127                                  <1> ;;      will be formatted/sized as result of available
10128                                  <1> ;;      memory calculation before paging is enabled.
10129                                  <1> ;;
10130                                  <1> ;; For 4GB Available/Present Memory: (max. possible memory size)
10131                                  <1> ;;    - Memory Allocation Table size will be 128 KB.
10132                                  <1> ;;    - Memory allocation for kernel page directory size
10133                                  <1> ;;      is always 4 KB. (in addition to total allocation size
10134                                  <1> ;;      for page tables)
10135                                  <1> ;;    - Memory allocation for kernel page tables (1024 tables)
10136                                  <1> ;;      is 4 MB (1024*4*1024 bytes).
10137                                  <1> ;;    - User (available) space will be started
10138                                  <1> ;;      at 6th MB of the memory (after 1MB+4MB).
10139                                  <1> ;;    - The first 640 KB is for kernel's itself plus
10140                                  <1> ;;      memory allocation table and kernel's page directory
10141                                  <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
10142                                  <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
10143                                  <1> ;;      for buffers.
10144                                  <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
10145                                  <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
10146                                  <1> ;;    - Kernel page tables start at 100000h (2nd MB)
10147                                  <1> ;;
10148                                  <1> ;; For 1GB Available Memory:
10149                                  <1> ;;    - Memory Allocation Table size will be 32 KB.
10150                                  <1> ;;    - Memory allocation for kernel page directory size
10151                                  <1> ;;      is always 4 KB. (in addition to total allocation size
```

```
10152                                  <1> ;;      for page tables)
10153                                  <1> ;;    - Memory allocation for kernel page tables (256 tables)
10154                                  <1> ;;      is 1 MB (256*4*1024 bytes).
10155                                  <1> ;;    - User (available) space will be started
10156                                  <1> ;;      at 3th MB of the memory (after 1MB+1MB).
10157                                  <1> ;;    - The first 640 KB is for kernel's itself plus
10158                                  <1> ;;      memory allocation table and kernel's page directory
10159                                  <1> ;;      (D0000h-EFFFFh may be used as kernel space...)
10160                                  <1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
10161                                  <1> ;;      for buffers.
10162                                  <1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
10163                                  <1> ;,      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
10164                                  <1> ;;    - Kernel page tables start at 100000h (2nd MB).
10165                                  <1> ;;
10166                                  <1> ;;
10167                                  <1>
10168                                  <1>
10169                                  <1>
;;*****************************************************************************
10170                                  <1> ;;
10171                                  <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
10172                                  <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
10173                                  <1>
10174                                  <1> ;; Main factor: "sys fork" system call
10175                                  <1> ;;
10176                                  <1> ;;          FORK
10177                                  <1> ;;                       |----> parent - duplicated PTEs, read only pages
10178                                  <1> ;;  writable pages ---->|
10179                                  <1> ;;                       |----> child - duplicated PTEs, read only pages
10180                                  <1> ;;
10181                                  <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
10182                                  <1> ;;
10183                                  <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is
set)
10184                                  <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
10185                                  <1> ;;       -while R/W bit is 0-.
10186                                  <1> ;;
10187                                  <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
10188                                  <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
10189                                  <1> ;;   as duplicated PTE bit  -AVL bit 0, PTE bit 9- are reset/clear.
10190                                  <1> ;; # Then Parent's Page Table is copied to Child's Page Table.
10191                                  <1> ;; # Child's Page Table Entries are updated as duplicated child bit
10192                                  <1> ;;   -AVL bit 0, PTE bit 9- is set.
10193                                  <1> ;;
10194                                  <1> ;; Duplicate page tables with read only pages (several sys fork system calls):
10195                                  <1> ;; # Parent's read only pages are copied to new child pages.
10196                                  <1> ;;   Parent's PTE attributes are not changed.
10197                                  <1> ;;   (Because, there is another parent-child fork before this fork! We must not
10198                                  <1> ;;    destroy/mix previous fork result).
10199                                  <1> ;; # Child's Page Table Entries (which are corresponding to Parent's
10200                                  <1> ;;   read only pages) are set as writable (while duplicated PTE bit is clear).
10201                                  <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
10202                                  <1> ;;   as read only, (while) duplicated PTE bit is reset (clear).
10203                                  <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
10204                                  <1> ;;   as Child's Page Table Entries without copying actual page.
10205                                  <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
10206                                  <1> ;;   pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
10207                                  <1> ;;
10208                                  <1> ;; !? WHAT FOR (duplication after duplication):
10209                                  <1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
10210                                  <1> ;; program/executable code continues from specified location as child process,
10211                                  <1> ;; returns back previous code location as parent process, every child after
10212                                  <1> ;; every sys fork uses last image of code and data just prior the fork.
10213                                  <1> ;; Even if the parent code changes data, the child will not see the changed data
10214                                  <1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
10215                                  <1> ;; was copied to child's process segment (all of code and data) according to
10216                                  <1> ;; original UNIX v1 which copies all of parent process code and data -core-
10217                                  <1> ;; to child space -core- but swaps that core image -of child- on to disk.
10218                                  <1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
10219                                  <1> ;; (complete running image of parent process) to the child process;
10220                                  <1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
10221                                  <1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
10222                                  <1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
10223                                  <1> ;; a new/fresh core -running space-, by clearing all code/data content).
10224                                  <1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
10225                                  <1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
10226                                  <1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
10227                                  <1> ;; new/fresh pages will be used to load and run new executable/program.
10228                                  <1> ;; That is what for i have preferred "copy on write", "duplication" method
10229                                  <1> ;; for sharing same read only pages between parent and child processes.
10230                                  <1> ;; That is a pitty i have to use new private flag (AVL bit 0, "duplicated PTE
10231                                  <1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
10232                                  <1> ;; and it's child processes; otherwise parent process would destroy data belongs
10233                                  <1> ;; to its child or vice versa; or some pages would remain unclaimed
10234                                  <1> ;; -deallocation problem-.
10235                                  <1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
10236                                  <1> ;;
10237                                  <1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
10238                                  <1> ;; # Page fault handler will do those:
10239                                  <1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
10240                                  <1> ;;   - If it is reset/clear, there is a child uses same page.
10241                                  <1> ;;   - Parent's read only page -previous page- is copied to a new writable page.
10242                                  <1> ;;   - Parent's PTE is updated as writable page, as unique page (AVL=0)
10243                                  <1> ;;   - (Page fault handler whill check this PTE later, if child process causes to
10244                                  <1> ;;     page fault due to write attempt on read only page. Of course, the previous
10245                                  <1> ;;     read only page will be converted to writable and unique page which belongs
10246                                  <1> ;;     to child process.)
10247                                  <1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
10248                                  <1> ;; # Page fault handler will do those:
10249                                  <1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
10250                                  <1> ;;   - If it is set, there is a parent uses -or was using- same page.
10251                                  <1> ;;   - Same PTE address within parent's page table is checked if it has same page
10252                                  <1> ;;     address or not.
10253                                  <1> ;;   - If parent's PTE has same address, child will continue with a new writable
page.
```

```
                                <1> ;;     Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
                                <1> ;;
                                <1> ;;   - If parent's PTE has different address, child will continue with it's
                                <1> ;;     own/same page but read only flag (0) will be changed to writable flag (1) and
                                <1> ;;     'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.
                                <1> ;;
                                <1> ;;
                                <1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables
                                <1> ;;       will be set as writable (and unique) in case of child process was using
                                <1> ;;       same pages with duplicated child PTE sign... Depending on sys fork and
                                <1> ;;       duplication method details, it is not possible multiple child processes
                                <1> ;;       were using same page with duplicated PTEs.
                                <1> ;;
                                <1>
;;******************************************************************************
                                <1>
                                <1> ;; 08/10/2014
                                <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
                                <1> ;;          by Erdogan Tan (Based on KolibriOS 'memory.inc')
                                <1>
                                <1> ;; 'allocate_page' code is derived and modified from KolibriOS
                                <1> ;; 'alloc_page' procedure in 'memory.inc'
                                <1> ;; (25/08/2014, Revision: 5057) file
                                <1> ;; by KolibriOS Team (2004-2012)
                                <1>
                                <1> allocate_page:
                                <1>       ; 01/07/2015
                                <1>       ; 05/05/2015
                                <1>       ; 30/04/2015
                                <1>       ; 16/10/2014
                                <1>       ; 08/10/2014
                                <1>       ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
                                <1>       ;
                                <1>       ; INPUT -> none
                                <1>       ;
                                <1>       ; OUTPUT ->
                                <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                                <1>       ;     (corresponding MEMORY ALLOCATION TABLE bit is RESET)
                                <1>       ;
                                <1>       ;     CF = 1 and EAX = 0
                                <1>       ;               if there is not a free page to be allocated
                                <1>       ;
                                <1>       ; Modified Registers -> none (except EAX)
                                <1>       ;
10295 00002FF5 A1[70700000]     <1>       mov    eax, [free_pages]
10296 00002FFA 21C0             <1>       and    eax, eax
10297 00002FFC 7438             <1>       jz     short out_of_memory
10298                           <1>       ;
10299 00002FFE 53               <1>       push   ebx
10300 00002FFF 51               <1>       push   ecx
10301                           <1>       ;
10302 00003000 BB00001000       <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table offset
10303 00003005 89D9             <1>       mov    ecx, ebx
10304                           <1>                                   ; NOTE: 32 (first_page) is initial
10305                           <1>                                   ; value of [next_page].
10306                           <1>                                   ; It points to the first available
10307                           <1>                                   ; page block for users (ring 3) ...
10308                           <1>                                   ; (MAT offset 32 = 1024/32)
10309                           <1>                                   ; (at the of the first 4 MB)
10310 00003007 031D[74700000]   <1>       add    ebx, [next_page] ; Free page searching starts from here
10311                           <1>                                 ; next_free_page >> 5
10312 0000300D 030D[78700000]   <1>       add    ecx, [last_page] ; Free page searching ends here
10313                           <1>                                 ; (total_pages - 1) >> 5
10314                           <1> al_p_scan:
10315 00003013 39CB             <1>       cmp    ebx, ecx
10316 00003015 770A             <1>       ja     short al_p_notfound
10317                           <1>       ;
10318                           <1>       ; 01/07/2015
10319                           <1>       ; AMD64 Architecture Programmer’s Manual
10320                           <1>       ; Volume 3:
10321                           <1>       ; General-Purpose and System Instructions
10322                           <1>       ;
10323                           <1>       ; BSF - Bit Scan Forward
10324                           <1>       ;
10325                           <1>       ;   Searches the value in a register or a memory location
10326                           <1>       ;   (second operand) for the least-significant set bit.
10327                           <1>       ;   If a set bit is found, the instruction clears the zero flag (ZF)
10328                           <1>       ;   and stores the index of the least-significant set bit in a destination
10329                           <1>       ;   register (first operand). If the second operand contains 0,
10330                           <1>       ;   the instruction sets ZF to 1 and does not change the contents of the
10331                           <1>       ;   destination register. The bit index is an unsigned offset from bit 0
10332                           <1>       ;   of the searched value
10333                           <1>       ;
10334 00003017 0FBC03           <1>       bsf    eax, [ebx] ; Scans source operand for first bit set (1).
10335                           <1>                         ; Clear ZF if a bit is found set (1) and
10336                           <1>                         ; loads the destination with an index to
10337                           <1>                         ; first set bit. (0 -> 31)
10338                           <1>                         ; Sets ZF to 1 if no bits are found set.
10339 0000301A 7525             <1>       jnz    short al_p_found ; ZF = 0 -> a free page has been found
10340                           <1>                       ;
10341                           <1>                       ; NOTE:  a Memory Allocation Table bit
10342                           <1>                       ;       with value of 1 means
10343                           <1>                       ;       the corresponding page is free
10344                           <1>                       ;       (Retro UNIX 386 v1 feaure only!)
10345 0000301C 83C304           <1>       add    ebx, 4
10346                           <1>                       ; We return back for searching next page block
10347                           <1>                       ; NOTE: [free_pages] is not ZERO; so,
10348                           <1>                       ;     we always will find at least 1 free page here.
10349 0000301F EBF2             <1>         jmp     short al_p_scan
10350                           <1>       ;
10351                           <1> al_p_notfound:
10352 00003021 81E900001000     <1>       sub    ecx, MEM_ALLOC_TBL
10353 00003027 890D[74700000]   <1>       mov    [next_page], ecx ; next/first free page = last page
10354                           <1>                               ; (deallocate_page procedure will change it)
10355 0000302D 31C0             <1>       xor    eax, eax
```

```
10356 0000302F A3[70700000]       <1>       mov   [free_pages], eax ; 0
10357 00003034 59                 <1>       pop   ecx
10358 00003035 5B                 <1>       pop   ebx
10359                             <1>       ;
10360                             <1> out_of_memory:
10361 00003036 E857040000         <1>       call  swap_out
10362 0000303B 7325               <1>       jnc   short al_p_ok  ; [free_pages] = 0, re-allocation by swap_out
10363                             <1>       ;
10364 0000303D 29C0               <1>       sub   eax, eax ; 0
10365 0000303F F9                 <1>       stc
10366 00003040 C3                 <1>       retn
10367                             <1>
10368                             <1> al_p_found:
10369 00003041 89D9               <1>       mov   ecx, ebx
10370 00003043 81E900001000       <1>       sub   ecx, MEM_ALLOC_TBL
10371 00003049 890D[74700000]     <1>       mov   [next_page], ecx ; Set first free page searching start
10372                             <1>                              ; address/offset (to the next)
10373 0000304F FF0D[70700000]     <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
10374                             <1>       ;
10375 00003055 0FB303             <1>       btr   [ebx], eax     ; The destination bit indexed by the source value
10376                             <1>                            ; is copied into the Carry Flag and then cleared
10377                             <1>                            ; in the destination.
10378                             <1>                            ;
10379                             <1>                            ; Reset the bit which is corresponding to the
10380                             <1>                            ; (just) allocated page.
10381                             <1>       ; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
10382 00003058 C1E103             <1>       shl   ecx, 3         ; (page block offset * 32) + page index
10383 0000305B 01C8               <1>       add   eax, ecx       ; = page number
10384 0000305D C1E00C             <1>       shl   eax, 12              ; physical address of the page (flat/real value)
10385                             <1>       ; EAX = physical address of memory page
10386                             <1>       ;
10387                             <1>       ; NOTE: The relevant page directory and page table entry will be updated
10388                             <1>       ;       according to this EAX value...
10389 00003060 59                 <1>       pop   ecx
10390 00003061 5B                 <1>       pop   ebx
10391                             <1> al_p_ok:
10392 00003062 C3                 <1>       retn
10393                             <1>
10394                             <1>
10395                             <1> make_page_dir:
10396                             <1>       ; 18/04/2015
10397                             <1>       ; 12/04/2015
10398                             <1>       ; 23/10/2014
10399                             <1>       ; 16/10/2014
10400                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10401                             <1>       ;
10402                             <1>       ; INPUT ->
10403                             <1>       ;     none
10404                             <1>       ; OUTPUT ->
10405                             <1>       ;     (EAX = 0)
10406                             <1>       ;     cf = 1 -> insufficient (out of) memory error
10407                             <1>       ;     cf = 0 ->
10408                             <1>       ;     u.pgdir = page directory (physical) address of the current
10409                             <1>       ;               process/user.
10410                             <1>       ;
10411                             <1>       ; Modified Registers -> EAX
10412                             <1>       ;
10413 00003063 E88DFFFFFF         <1>       call  allocate_page
10414 00003068 7216               <1>       jc    short mkpd_error
10415                             <1>       ;
10416 0000306A A3[A1740000]       <1>       mov   [u.pgdir], eax    ; Page dir address for current user/process
10417                             <1>                               ; (Physical address)
10418                             <1> clear_page:
10419                             <1>       ; 18/04/2015
10420                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10421                             <1>       ;
10422                             <1>       ; INPUT ->
10423                             <1>       ;     EAX = physical address of the page
10424                             <1>       ; OUTPUT ->
10425                             <1>       ;     all bytes of the page will be cleared
10426                             <1>       ;
10427                             <1>       ; Modified Registers -> none
10428                             <1>       ;
10429 0000306F 57                 <1>       push  edi
10430 00003070 51                 <1>       push  ecx
10431 00003071 50                 <1>       push  eax
10432 00003072 B900040000         <1>       mov   ecx, PAGE_SIZE / 4
10433 00003077 89C7               <1>       mov   edi, eax
10434 00003079 31C0               <1>       xor   eax, eax
10435 0000307B F3AB               <1>       rep   stosd
10436 0000307D 58                 <1>       pop   eax
10437 0000307E 59                 <1>       pop   ecx
10438 0000307F 5F                 <1>       pop   edi
10439                             <1> mkpd_error:
10440                             <1> mkpt_error:
10441 00003080 C3                 <1>       retn
10442                             <1>
10443                             <1> make_page_table:
10444                             <1>       ; 23/06/2015
10445                             <1>       ; 18/04/2015
10446                             <1>       ; 12/04/2015
10447                             <1>       ; 16/10/2014
10448                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10449                             <1>       ;
10450                             <1>       ; INPUT ->
10451                             <1>       ;     EBX = virtual (linear) address
10452                             <1>       ;     ECX = page table attributes (lower 12 bits)
10453                             <1>       ;           (higher 20 bits must be ZERO)
10454                             <1>       ;           (bit 0 must be 1)
10455                             <1>       ;     u.pgdir = page directory (physical) address
10456                             <1>       ; OUTPUT ->
10457                             <1>       ;     EDX = Page directory entry address
10458                             <1>       ;     EAX = Page table address
10459                             <1>       ;     cf = 1 -> insufficient (out of) memory error
10460                             <1>       ;     cf = 0 -> page table address in the PDE (EDX)
```

```
10461                                  <1>       ;
10462                                  <1>       ; Modified Registers -> EAX, EDX
10463                                  <1>       ;
10464 00003081 E86FFFFFFF               <1>       call   allocate_page
10465 00003086 72F8                     <1>       jc     short mkpt_error
10466 00003088 E811000000               <1>       call   set_pde
10467 0000308D EBE0                     <1>       jmp    short clear_page
10468                                  <1>
10469                                  <1> make_page:
10470                                  <1>       ; 24/07/2015
10471                                  <1>       ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
10472                                  <1>       ;
10473                                  <1>       ; INPUT ->
10474                                  <1>       ;      EBX = virtual (linear) address
10475                                  <1>       ;      ECX = page attributes (lower 12 bits)
10476                                  <1>       ;            (higher 20 bits must be ZERO)
10477                                  <1>       ;            (bit 0 must be 1)
10478                                  <1>       ;      u.pgdir = page directory (physical) address
10479                                  <1>       ; OUTPUT ->
10480                                  <1>       ;      EBX = Virtual address
10481                                  <1>       ;      (EDX = PTE value)
10482                                  <1>       ;      EAX = Physical address
10483                                  <1>       ;      cf = 1 -> insufficient (out of) memory error
10484                                  <1>       ;
10485                                  <1>       ; Modified Registers -> EAX, EDX
10486                                  <1>       ;
10487 0000308F E861FFFFFF               <1>       call   allocate_page
10488 00003094 7207                     <1>       jc     short mkp_err
10489 00003096 E821000000               <1>       call   set_pte
10490 0000309B 73D2                     <1>       jnc    short clear_page ; 18/04/2015
10491                                  <1> mkp_err:
10492 0000309D C3                       <1>       retn
10493                                  <1>
10494                                  <1>
10495                                  <1> set_pde:    ; Set page directory entry (PDE)
10496                                  <1>       ; 20/07/2015
10497                                  <1>       ; 18/04/2015
10498                                  <1>       ; 12/04/2015
10499                                  <1>       ; 23/10/2014
10500                                  <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10501                                  <1>       ;
10502                                  <1>       ; INPUT ->
10503                                  <1>       ;      EAX = physical address
10504                                  <1>       ;            (use present value if EAX = 0)
10505                                  <1>       ;      EBX = virtual (linear) address
10506                                  <1>       ;      ECX = page table attributes (lower 12 bits)
10507                                  <1>       ;            (higher 20 bits must be ZERO)
10508                                  <1>       ;            (bit 0 must be 1)
10509                                  <1>       ;      u.pgdir = page directory (physical) address
10510                                  <1>       ; OUTPUT ->
10511                                  <1>       ;      EDX = PDE address
10512                                  <1>       ;      EAX = page table address (physical)
10513                                  <1>       ;      ;(CF=1 -> Invalid page address)
10514                                  <1>       ;
10515                                  <1>       ; Modified Registers -> EDX
10516                                  <1>       ;
10517 0000309E 89DA                     <1>       mov    edx, ebx
10518 000030A0 C1EA16                   <1>       shr    edx, PAGE_D_SHIFT ; 22
10519 000030A3 C1E202                   <1>       shl    edx, 2 ; offset to page directory (1024*4)
10520 000030A6 0315[A1740000]           <1>       add    edx, [u.pgdir]
10521                                  <1>       ;
10522 000030AC 21C0                     <1>       and    eax, eax
10523 000030AE 7506                     <1>       jnz    short spde_1
10524                                  <1>       ;
10525 000030B0 8B02                     <1>       mov    eax, [edx]  ; old PDE value
10526                                  <1>       ;test  al, 1
10527                                  <1>       ;jz    short spde_2
10528 000030B2 662500F0                 <1>       and    ax, PDE_A_CLEAR ; 0F000h  ; clear lower 12 bits
10529                                  <1> spde_1:
10530                                  <1>       ;and   cx, 0FFFh
10531 000030B6 8902                     <1>       mov    [edx], eax
10532 000030B8 66090A                   <1>       or     [edx], cx
10533 000030BB C3                       <1>       retn
10534                                  <1> ;spde_2: ; error
10535                                  <1> ;     stc
10536                                  <1> ;     retn
10537                                  <1>
10538                                  <1> set_pte:    ; Set page table entry (PTE)
10539                                  <1>       ; 24/07/2015
10540                                  <1>       ; 20/07/2015
10541                                  <1>       ; 23/06/2015
10542                                  <1>       ; 18/04/2015
10543                                  <1>       ; 12/04/2015
10544                                  <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10545                                  <1>       ;
10546                                  <1>       ; INPUT ->
10547                                  <1>       ;      EAX = physical page address
10548                                  <1>       ;            (use present value if EAX = 0)
10549                                  <1>       ;      EBX = virtual (linear) address
10550                                  <1>       ;      ECX = page attributes (lower 12 bits)
10551                                  <1>       ;            (higher 20 bits must be ZERO)
10552                                  <1>       ;            (bit 0 must be 1)
10553                                  <1>       ;      u.pgdir = page directory (physical) address
10554                                  <1>       ; OUTPUT ->
10555                                  <1>       ;      EAX = physical page address
10556                                  <1>       ;      (EDX = PTE value)
10557                                  <1>       ;      EBX = virtual address
10558                                  <1>       ;
10559                                  <1>       ;      CF = 1 -> error
10560                                  <1>       ;
10561                                  <1>       ; Modified Registers -> EAX, EDX
10562                                  <1>       ;
10563 000030BC 50                       <1>       push   eax
10564 000030BD A1[A1740000]             <1>       mov    eax, [u.pgdir] ; 20/07/2015
10565 000030C2 E837000000               <1>       call   get_pde
```

```
10566                                   <1>             ; EDX = PDE address
10567                                   <1>             ; EAX = PDE value
10568 000030C7 5A                       <1>       pop   edx ; physical page address
10569 000030C8 722A                     <1>       jc    short spte_err ; PDE not present
10570                                   <1>       ;
10571 000030CA 53                       <1>       push  ebx ; 24/07/2015
10572 000030CB 662500F0                 <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
10573                                   <1>                             ; EDX = PT address (physical)
10574 000030CF C1EB0C                   <1>       shr   ebx, PAGE_SHIFT ; 12
10575 000030D2 81E3FF030000             <1>       and   ebx, PTE_MASK; 03FFh
10576                                   <1>                          ; clear higher 10 bits (PD bits)
10577 000030D8 C1E302                   <1>       shl   ebx, 2   ; offset to page table (1024*4)
10578 000030DB 01C3                     <1>       add   ebx, eax
10579                                   <1>       ;
10580 000030DD 8B03                     <1>       mov   eax, [ebx] ; Old PTE value
10581 000030DF A801                     <1>       test  al, 1
10582 000030E1 740C                     <1>       jz    short spte_0
10583 000030E3 09D2                     <1>       or    edx, edx
10584 000030E5 750F                     <1>       jnz   short spte_1
10585 000030E7 662500F0                 <1>       and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
10586 000030EB 89C2                     <1>       mov   edx, eax
10587 000030ED EB09                     <1>       jmp   short spte_2
10588                                   <1> spte_0:
10589                                   <1>       ; If this PTE contains a swap (disk) address,
10590                                   <1>       ; it can be updated by using 'swap_in' procedure
10591                                   <1>       ; only!
10592 000030EF 21C0                     <1>       and   eax, eax
10593 000030F1 7403                     <1>       jz    short spte_1
10594                                   <1>       ; 24/07/2015
10595                                   <1>       ; swapped page ! (on disk)
10596 000030F3 5B                       <1>       pop   ebx
10597                                   <1> spte_err:
10598 000030F4 F9                       <1>       stc
10599 000030F5 C3                       <1>       retn
10600                                   <1> spte_1:
10601 000030F6 89D0                     <1>       mov   eax, edx
10602                                   <1> spte_2:
10603 000030F8 09CA                     <1>       or    edx, ecx
10604                                   <1>       ; 23/06/2015
10605 000030FA 8913                     <1>       mov   [ebx], edx ; PTE value in EDX
10606                                   <1>       ; 24/07/2015
10607 000030FC 5B                       <1>       pop   ebx
10608 000030FD C3                       <1>       retn
10609                                   <1>
10610                                   <1> get_pde:    ; Get present value of the relevant PDE
10611                                   <1>       ; 20/07/2015
10612                                   <1>       ; 18/04/2015
10613                                   <1>       ; 12/04/2015
10614                                   <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10615                                   <1>       ;
10616                                   <1>       ; INPUT ->
10617                                   <1>       ;     EBX = virtual (linear) address
10618                                   <1>       ;     EAX = page directory (physical) address
10619                                   <1>       ; OUTPUT ->
10620                                   <1>       ;     EDX = Page directory entry address
10621                                   <1>       ;     EAX = Page directory entry value
10622                                   <1>       ;     CF = 1 -> PDE not present or invalid ?
10623                                   <1>       ; Modified Registers -> EDX, EAX
10624                                   <1>       ;
10625 000030FE 89DA                     <1>       mov   edx, ebx
10626 00003100 C1EA16                   <1>       shr   edx, PAGE_D_SHIFT ; 22  (12+10)
10627 00003103 C1E202                   <1>       shl   edx, 2 ; offset to page directory (1024*4)
10628 00003106 01C2                     <1>       add   edx, eax ; page directory address (physical)
10629 00003108 8B02                     <1>       mov   eax, [edx]
10630 0000310A A801                     <1>       test  al, PDE_A_PRESENT ; page table is present or not !
10631 0000310C 751F                     <1>       jnz   short gpte_retn
10632 0000310E F9                       <1>       stc
10633                                   <1> gpde_retn:
10634 0000310F C3                       <1>       retn
10635                                   <1>
10636                                   <1> get_pte:
10637                                   <1>             ; Get present value of the relevant PTE
10638                                   <1>       ; 29/07/2015
10639                                   <1>       ; 20/07/2015
10640                                   <1>       ; 18/04/2015
10641                                   <1>       ; 12/04/2015
10642                                   <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10643                                   <1>       ;
10644                                   <1>       ; INPUT ->
10645                                   <1>       ;     EBX = virtual (linear) address
10646                                   <1>       ;     EAX = page directory (physical) address
10647                                   <1>       ; OUTPUT ->
10648                                   <1>       ;     EDX = Page table entry address (if CF=0)
10649                                   <1>       ;           Page directory entry address (if CF=1)
10650                                   <1>       ;            (Bit 0 value is 0 if PT is not present)
10651                                   <1>       ;     EAX = Page table entry value (page address)
10652                                   <1>       ;     CF = 1 -> PDE not present or invalid ?
10653                                   <1>       ; Modified Registers -> EAX, EDX
10654                                   <1>       ;
10655 00003110 E8E9FFFFFF               <1>       call  get_pde
10656 00003115 72F8                     <1>       jc    short gpde_retn     ; page table is not present
10657                                   <1>       ;jnc  short gpte_1
10658                                   <1>       ;retn
10659                                   <1> ;gpte_1:
10660 00003117 662500F0                 <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
10661 0000311B 89DA                     <1>       mov   edx, ebx
10662 0000311D C1EA0C                   <1>       shr   edx, PAGE_SHIFT ; 12
10663 00003120 81E2FF030000             <1>       and   edx, PTE_MASK; 03FFh
10664                                   <1>                          ; clear higher 10 bits (PD bits)
10665 00003126 C1E202                   <1>       shl   edx, 2 ; offset from start of page table (1024*4)
10666 00003129 01C2                     <1>       add   edx, eax
10667 0000312B 8B02                     <1>       mov   eax, [edx]
10668                                   <1> gpte_retn:
10669 0000312D C3                       <1>       retn
10670                                   <1>
```

```
10671                                  <1> deallocate_page_dir:
10672                                  <1>       ; 15/09/2015
10673                                  <1>       ; 05/08/2015
10674                                  <1>       ; 30/04/2015
10675                                  <1>       ; 28/04/2015
10676                                  <1>       ; 17/10/2014
10677                                  <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
10678                                  <1>       ;
10679                                  <1>       ; INPUT ->
10680                                  <1>       ;     EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
10681                                  <1>       ;     EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
10682                                  <1>       ; OUTPUT ->
10683                                  <1>       ;     All of page tables in the page directory
10684                                  <1>       ;     and page dir's itself will be deallocated
10685                                  <1>       ;     except 'read only' duplicated pages (will be converted
10686                                  <1>       ;     to writable pages).
10687                                  <1>       ;
10688                                  <1>       ; Modified Registers -> EAX
10689                                  <1>       ;
10690                                  <1>       ;
10691 0000312E 56                     <1>       push   esi
10692 0000312F 51                     <1>       push   ecx
10693 00003130 50                     <1>       push   eax
10694 00003131 89C6                   <1>       mov    esi, eax
10695 00003133 31C9                   <1>       xor    ecx, ecx
10696                                  <1>       ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
10697                                  <1>       ; it must not be deallocated
10698 00003135 890E                   <1>       mov    [esi], ecx ; 0 ; clear PDE 0
10699                                  <1> dapd_0:
10700 00003137 AD                     <1>       lodsd
10701 00003138 A801                   <1>       test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
10702 0000313A 7409                   <1>       jz     short dapd_1
10703 0000313C 662500F0               <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
10704 00003140 E812000000             <1>       call   deallocate_page_table
10705                                  <1> dapd_1:
10706 00003145 41                     <1>       inc    ecx ; page directory entry index
10707 00003146 81F900040000           <1>       cmp    ecx, PAGE_SIZE / 4 ; 1024
10708 0000314C 72E9                   <1>       jb     short dapd_0
10709                                  <1> dapd_2:
10710 0000314E 58                     <1>       pop    eax
10711 0000314F E879000000             <1>       call   deallocate_page     ; deallocate the page dir's itself
10712 00003154 59                     <1>       pop    ecx
10713 00003155 5E                     <1>       pop    esi
10714 00003156 C3                     <1>       retn
10715                                  <1>
10716                                  <1> deallocate_page_table:
10717                                  <1>       ; 19/09/2015
10718                                  <1>       ; 15/09/2015
10719                                  <1>       ; 05/08/2015
10720                                  <1>       ; 30/04/2015
10721                                  <1>       ; 28/04/2015
10722                                  <1>       ; 24/10/2014
10723                                  <1>       ; 23/10/2014
10724                                  <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
10725                                  <1>       ;
10726                                  <1>       ; INPUT ->
10727                                  <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
10728                                  <1>       ;     EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
10729                                  <1>       ;     (ECX = page directory entry index)
10730                                  <1>       ; OUTPUT ->
10731                                  <1>       ;     All of pages in the page table and page table's itself
10732                                  <1>       ;     will be deallocated except 'read only' duplicated pages
10733                                  <1>       ;     (will be converted to writable pages).
10734                                  <1>       ;
10735                                  <1>       ; Modified Registers -> EAX
10736                                  <1>       ;
10737 00003157 56                     <1>       push   esi
10738 00003158 57                     <1>       push   edi
10739 00003159 52                     <1>       push   edx
10740 0000315A 50                     <1>       push   eax ; *
10741 0000315B 89C6                   <1>       mov    esi, eax
10742 0000315D 31FF                   <1>       xor    edi, edi ; 0
10743                                  <1> dapt_0:
10744 0000315F AD                     <1>       lodsd
10745 00003160 A801                   <1>       test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
10746 00003162 7441                   <1>       jz     short dapt_1
10747                                  <1>       ;
10748 00003164 A802                   <1>       test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
10749                                  <1>                                ; (must be 1)
10750 00003166 754C                   <1>       jnz    short dapt_3
10751                                  <1>       ; Read only -duplicated- page (belongs to a parent or a child)
10752 00003168 66A90002               <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
10753                                  <1>                                    ; as child's page ?
10754 0000316C 744B                   <1>       jz     short dapt_4 ; Clear PTE but don't deallocate the page!
10755                                  <1>       ; check the parent's PTE value is read only & same page or not..
10756                                  <1>       ; ECX = page directory entry index (0-1023)
10757 0000316E 53                     <1>       push   ebx
10758 0000316F 51                     <1>       push   ecx
10759 00003170 66C1E102               <1>       shl    cx, 2 ; *4
10760 00003174 01CB                   <1>       add    ebx, ecx ; PDE offset (for the parent)
10761 00003176 8B0B                   <1>       mov    ecx, [ebx]
10762 00003178 F6C101                 <1>       test   cl, PDE_A_PRESENT ; present (valid) or not ?
10763 0000317B 7435                   <1>       jz     short dapt_2 ; parent process does not use this page
10764 0000317D 6681E100F0             <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
10765                                  <1>       ; EDI = page table entry index (0-1023)
10766 00003182 89FA                   <1>       mov    edx, edi
10767 00003184 66C1E202               <1>       shl    dx, 2 ; *4
10768 00003188 01CA                   <1>       add    edx, ecx ; PTE offset (for the parent)
10769 0000318A 8B1A                   <1>       mov    ebx, [edx]
10770 0000318C F6C301                 <1>       test   bl, PTE_A_PRESENT ; present or not ?
10771 0000318F 7421                   <1>       jz     short dapt_2 ; parent process does not use this page
10772 00003191 662500F0               <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
10773 00003195 6681E300F0             <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
10774 0000319A 39D8                   <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
10775 0000319C 7514                   <1>       jne    short dapt_2 ; not same page
```

```
                                   <1>                            ; deallocate the child's page
0000319E 800A02                    <1>         or      byte [edx], PTE_A_WRITE ; convert to writable page (parent)
000031A1 59                        <1>       pop    ecx
000031A2 5B                        <1>       pop    ebx
000031A3 EB14                      <1>       jmp    short dapt_4
                                   <1> dapt_1:
000031A5 09C0                      <1>       or     eax, eax     ; swapped page ?
000031A7 7417                      <1>       jz     short dapt_5 ; no
                                   <1>                           ; yes
000031A9 D1E8                      <1>       shr    eax, 1
000031AB E848040000                <1>       call   unlink_swap_block ; Deallocate swapped page block
                                   <1>                             ; on the swap disk (or in file)
000031B0 EB0E                      <1>       jmp    short dapt_5
                                   <1> dapt_2:
000031B2 59                        <1>       pop    ecx
000031B3 5B                        <1>       pop    ebx
                                   <1> dapt_3:
                                   <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
000031B4 E814000000                <1>       call   deallocate_page
                                   <1> dapt_4:
000031B9 C746FC00000000            <1>       mov    dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                                   <1> dapt_5:
000031C0 47                        <1>       inc    edi ; page table entry index
000031C1 81FF00040000              <1>       cmp    edi, PAGE_SIZE / 4 ; 1024
000031C7 7296                      <1>       jb     short dapt_0
                                   <1>       ;
000031C9 58                        <1>       pop    eax ; *
000031CA 5A                        <1>       pop    edx
000031CB 5F                        <1>       pop    edi
000031CC 5E                        <1>       pop    esi
                                   <1>       ;
                                   <1>       ;call  deallocate_page     ; deallocate the page table's itself
                                   <1>       ;retn
                                   <1>
                                   <1> deallocate_page:
                                   <1>       ; 15/09/2015
                                   <1>       ; 28/04/2015
                                   <1>       ; 10/03/2015
                                   <1>       ; 17/10/2014
                                   <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                                   <1>       ;
                                   <1>       ; INPUT ->
                                   <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                                   <1>       ; OUTPUT ->
                                   <1>       ;      [free_pages] is increased
                                   <1>       ;      (corresponding MEMORY ALLOCATION TABLE bit is SET)
                                   <1>       ;      CF = 1 if the page is already deallocated
                                   <1>       ;             (or not allocated) before.
                                   <1>       ;
                                   <1>       ; Modified Registers -> EAX
                                   <1>       ;
000031CD 53                        <1>       push   ebx
000031CE 52                        <1>       push   edx
                                   <1>       ;
000031CF C1E80C                    <1>       shr    eax, PAGE_SHIFT      ; shift physical address to
                                   <1>                                 ; 12 bits right
                                   <1>                                 ; to get page number
000031D2 89C2                      <1>       mov    edx, eax
                                   <1>       ; 15/09/2015
000031D4 C1EA03                    <1>       shr    edx, 3             ; to get offset to M.A.T.
                                   <1>                                 ; (1 allocation bit = 1 page)
                                   <1>                                 ; (1 allocation bytes = 8 pages)
000031D7 80E2FC                    <1>       and    dl, 0FCh           ; clear lower 2 bits
                                   <1>                                 ; (to get 32 bit position)
                                   <1>       ;
000031DA BB00001000                <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address
000031DF 01D3                      <1>       add    ebx, edx
000031E1 83E01F                    <1>       and    eax, 1Fh           ; lower 5 bits only
                                   <1>                                 ; (allocation bit position)
000031E4 3B15[74700000]            <1>       cmp    edx, [next_page]     ; is the new free page address lower
                                   <1>                                 ; than the address in 'next_page' ?
                                   <1>                                 ; (next/first free page value)
000031EA 7306                      <1>       jnb    short dap_1        ; no
000031EC 8915[74700000]            <1>       mov    [next_page], edx     ; yes
                                   <1> dap_1:
000031F2 0FAB03                    <1>       bts    [ebx], eax         ; unlink/release/deallocate page
                                   <1>                                 ; set relevant bit to 1.
                                   <1>                                 ; set CF to the previous bit value
                                   <1>       ;cmc                      ; complement carry flag
                                   <1>       ;jc    short dap_2        ; do not increase free_pages count
                                   <1>                                 ; if the page is already deallocated
                                   <1>                                 ; before.
000031F5 FF05[70700000]            <1>         inc     dword [free_pages]
                                   <1> dap_2:
000031FB 5A                        <1>       pop    edx
000031FC 5B                        <1>       pop    ebx
000031FD C3                        <1>       retn
                                   <1>
                                   <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   <1> ;;                                                              ;;
                                   <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;;
                                   <1> ;; Distributed under terms of the GNU General Public License    ;;
                                   <1> ;;                                                              ;;
                                   <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   <1>
                                   <1> ;;$Revision: 5057 $
                                   <1>
                                   <1>
                                   <1> ;;align 4
                                   <1> ;;proc alloc_page
                                   <1>
                                   <1> ;;        pushfd
                                   <1> ;;        cli
                                   <1> ;;        push    ebx
                                   <1> ;;;//-
```

```
10881                                  <1> ;;        cmp     [pg_data.pages_free], 1
10882                                  <1> ;;        jle     .out_of_memory
10883                                  <1> ;;;//-
10884                                  <1> ;;
10885                                  <1> ;;        mov     ebx, [page_start]
10886                                  <1> ;;        mov     ecx, [page_end]
10887                                  <1> ;;.l1:
10888                                  <1> ;;        bsf     eax, [ebx];
10889                                  <1> ;;        jnz     .found
10890                                  <1> ;;        add     ebx, 4
10891                                  <1> ;;        cmp     ebx, ecx
10892                                  <1> ;;        jb      .l1
10893                                  <1> ;;        pop     ebx
10894                                  <1> ;;        popfd
10895                                  <1> ;;        xor     eax, eax
10896                                  <1> ;;        ret
10897                                  <1> ;;.found:
10898                                  <1> ;;;//-
10899                                  <1> ;;        dec     [pg_data.pages_free]
10900                                  <1> ;;        jz      .out_of_memory
10901                                  <1> ;;;//-
10902                                  <1> ;;        btr     [ebx], eax
10903                                  <1> ;;        mov     [page_start], ebx
10904                                  <1> ;;        sub     ebx, sys_pgmap
10905                                  <1> ;;        lea     eax, [eax+ebx*8]
10906                                  <1> ;;        shl     eax, 12
10907                                  <1> ;;;//-       dec [pg_data.pages_free]
10908                                  <1> ;;        pop     ebx
10909                                  <1> ;;        popfd
10910                                  <1> ;;        ret
10911                                  <1> ;;;//-
10912                                  <1> ;;.out_of_memory:
10913                                  <1> ;;        mov     [pg_data.pages_free], 1
10914                                  <1> ;;        xor     eax, eax
10915                                  <1> ;;        pop     ebx
10916                                  <1> ;;        popfd
10917                                  <1> ;;        ret
10918                                  <1> ;;;//-
10919                                  <1> ;;endp
10920                                  <1>
10921                                  <1> duplicate_page_dir:
10922                                  <1>       ; 21/09/2015
10923                                  <1>       ; 31/08/2015
10924                                  <1>       ; 20/07/2015
10925                                  <1>       ; 28/04/2015
10926                                  <1>       ; 27/04/2015
10927                                  <1>       ; 18/04/2015
10928                                  <1>       ; 12/04/2015
10929                                  <1>       ; 18/10/2014
10930                                  <1>       ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
10931                                  <1>       ;
10932                                  <1>       ; INPUT ->
10933                                  <1>       ;     [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
10934                                  <1>       ;                 page directory.
10935                                  <1>       ; OUTPUT ->
10936                                  <1>       ;     EAX =  PHYSICAL (real/flat) ADDRESS of the child's
10937                                  <1>       ;            page directory.
10938                                  <1>       ;     (New page directory with new page table entries.)
10939                                  <1>       ;     (New page tables with read only copies of the parent's
10940                                  <1>       ;     pages.)
10941                                  <1>       ;     EAX = 0 -> Error (CF = 1)
10942                                  <1>       ;
10943                                  <1>       ; Modified Registers -> none (except EAX)
10944                                  <1>       ;
10945 000031FE E8F2FDFFFF             <1>       call  allocate_page
10946 00003203 723E                   <1>       jc    short dpd_err
10947                                  <1>       ;
10948 00003205 55                     <1>       push  ebp ; 20/07/2015
10949 00003206 56                     <1>       push  esi
10950 00003207 57                     <1>       push  edi
10951 00003208 53                     <1>       push  ebx
10952 00003209 51                     <1>       push  ecx
10953 0000320A 8B35[A1740000]         <1>       mov   esi, [u.pgdir]
10954 00003210 89C7                   <1>       mov   edi, eax
10955 00003212 50                     <1>       push  eax ; save child's page directory address
10956                                  <1>       ; 31/08/2015
10957                                  <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
10958                                  <1>       ; (use same system space for all user page tables)
10959 00003213 A5                     <1>       movsd
10960 00003214 BD00004000             <1>       mov   ebp, 1024*4096 ; pass the 1st 4MB (system space)
10961 00003219 B9FF030000             <1>       mov   ecx, (PAGE_SIZE / 4) - 1 ; 1023
10962                                  <1> dpd_0:
10963 0000321E AD                     <1>       lodsd
10964                                  <1>       ;or   eax, eax
10965                                  <1>         ;jnz     short dpd_1
10966 0000321F A801                   <1>       test  al, PDE_A_PRESENT ;  bit 0 =  1
10967 00003221 7508                   <1>       jnz   short dpd_1
10968                                  <1>       ; 20/07/2015 (virtual address at the end of the page table)
10969 00003223 81C500004000           <1>       add   ebp, 1024*4096 ; page size * PTE count
10970 00003229 EB0F                   <1>       jmp   short dpd_2
10971                                  <1> dpd_1:
10972 0000322B 662500F0               <1>       and   ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
10973 0000322F 89C3                   <1>       mov   ebx, eax
10974                                  <1>       ; EBX = Parent's page table address
10975 00003231 E81F000000             <1>       call  duplicate_page_table
10976 00003236 720C                   <1>       jc    short dpd_p_err
10977                                  <1>       ; EAX = Child's page table address
10978 00003238 0C07                   <1>       or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
10979                                  <1>                     ; set bit 0, bit 1 and bit 2 to 1
10980                                  <1>                     ; (present, writable, user)
10981                                  <1> dpd_2:
10982 0000323A AB                     <1>       stosd
10983 0000323B E2E1                   <1>       loop  dpd_0
10984                                  <1>       ;
10985 0000323D 58                     <1>       pop   eax  ; restore child's page directory address
```

```
10986                                  <1> dpd_3:
10987 0000323E 59                      <1>         pop    ecx
10988 0000323F 5B                      <1>         pop    ebx
10989 00003240 5F                      <1>         pop    edi
10990 00003241 5E                      <1>         pop    esi
10991 00003242 5D                      <1>         pop    ebp ; 20/07/2015
10992                                  <1> dpd_err:
10993 00003243 C3                      <1>         retn
10994                                  <1> dpd_p_err:
10995                                  <1>         ; release the allocated pages missing (recover free space)
10996 00003244 58                      <1>         pop    eax  ; the new page directory address (physical)
10997 00003245 8B1D[A1740000]          <1>         mov    ebx, [u.pgdir] ; parent's page directory address
10998 0000324B E8DEFEFFFF              <1>         call   deallocate_page_dir
10999 00003250 29C0                    <1>         sub    eax, eax ; 0
11000 00003252 F9                      <1>         stc
11001 00003253 EBE9                    <1>         jmp    short dpd_3
11002                                  <1>
11003                                  <1> duplicate_page_table:
11004                                  <1>         ; 21/09/2015
11005                                  <1>         ; 20/07/2015
11006                                  <1>         ; 05/05/2015
11007                                  <1>         ; 28/04/2015
11008                                  <1>         ; 27/04/2015
11009                                  <1>         ; 18/04/2015
11010                                  <1>         ; 18/10/2014
11011                                  <1>         ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
11012                                  <1>         ;
11013                                  <1>         ; INPUT ->
11014                                  <1>         ;      EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
11015                                  <1>         ;      EBP = page table entry index (from 'duplicate_page_dir')
11016                                  <1>         ; OUTPUT ->
11017                                  <1>         ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
11018                                  <1>         ;            (with 'read only' attribute of page table entries)
11019                                  <1>         ;      EBP = (recent) page table index (for 'add_to_swap_queue')
11020                                  <1>         ;      CF = 1 -> error
11021                                  <1>         ;
11022                                  <1>         ; Modified Registers -> EBP (except EAX)
11023                                  <1>         ;
11024 00003255 E89BFDFFFF              <1>         call   allocate_page
11025 0000325A 726A                    <1>         jc     short dpt_err
11026                                  <1>         ;
11027 0000325C 50                      <1>         push   eax ; *
11028 0000325D 56                      <1>         push   esi
11029 0000325E 57                      <1>         push   edi
11030 0000325F 52                      <1>         push   edx
11031 00003260 51                      <1>         push   ecx
11032                                  <1>         ;
11033 00003261 89DE                    <1>         mov    esi, ebx
11034 00003263 89C7                    <1>         mov    edi, eax
11035 00003265 89C2                    <1>         mov    edx, eax
11036 00003267 81C200100000            <1>         add    edx, PAGE_SIZE
11037                                  <1> dpt_0:
11038 0000326D AD                      <1>         lodsd
11039 0000326E 21C0                    <1>         and    eax, eax
11040 00003270 7444                    <1>         jz     short dpt_3
11041 00003272 A801                    <1>         test   al, PTE_A_PRESENT ;  bit 0 =  1
11042 00003274 7507                    <1>         jnz    short dpt_1
11043                                  <1>         ; 20/07/2015
11044                                  <1>         ; ebp = virtual (linear) address of the memory page
11045 00003276 E887040000              <1>         call   reload_page ; 28/04/2015
11046 0000327B 7244                    <1>         jc     short dpt_p_err
11047                                  <1> dpt_1:
11048                                  <1>         ; 21/09/2015
11049 0000327D 89C1                    <1>         mov    ecx, eax
11050 0000327F 662500F0                <1>         and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11051 00003283 F6C102                  <1>         test   cl, PTE_A_WRITE ; writable page ?
11052 00003286 7525                    <1>         jnz    short dpt_2
11053                                  <1>         ; Read only (parent) page
11054                                  <1>         ;      - there is a third process which uses this page -
11055                                  <1>         ; Allocate a new page for the child process
11056 00003288 E868FDFFFF              <1>         call   allocate_page
11057 0000328D 7232                    <1>         jc     short dpt_p_err
11058 0000328F 57                      <1>         push   edi
11059 00003290 56                      <1>         push   esi
11060 00003291 89CE                    <1>         mov    esi, ecx
11061 00003293 89C7                    <1>         mov    edi, eax
11062 00003295 B900040000              <1>         mov    ecx, PAGE_SIZE/4
11063 0000329A F3A5                    <1>         rep    movsd  ; copy page (4096 bytes)
11064 0000329C 5E                      <1>         pop    esi
11065 0000329D 5F                      <1>         pop    edi
11066                                  <1>         ;
11067 0000329E 53                      <1>         push   ebx
11068 0000329F 50                      <1>         push   eax
11069                                  <1>         ; 20/07/2015
11070 000032A0 89EB                    <1>         mov    ebx, ebp
11071                                  <1>         ; ebx = virtual address of the memory page
11072 000032A2 E80B030000              <1>         call   add_to_swap_queue
11073 000032A7 58                      <1>         pop    eax
11074 000032A8 5B                      <1>         pop    ebx
11075                                  <1>         ; 21/09/2015
11076 000032A9 0C07                    <1>         or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
11077                                  <1>                ; user + writable + present page
11078 000032AB EB09                    <1>         jmp    short dpt_3
11079                                  <1> dpt_2:
11080                                  <1>         ;or    ax, PTE_A_USER+PTE_A_PRESENT
11081 000032AD 0C05                    <1>         or     al, PTE_A_USER+PTE_A_PRESENT
11082                                  <1>                    ; (read only page!)
11083 000032AF 8946FC                  <1>         mov    [esi-4], eax ; update parent's PTE
11084 000032B2 660D0002                <1>         or      ax, PTE_DUPLICATED  ; (read only page & duplicated PTE!)
11085                                  <1> dpt_3:
11086 000032B6 AB                      <1>         stosd  ; EDI points to child's PTE
11087                                  <1>         ;
11088 000032B7 81C500100000            <1>         add    ebp, 4096 ; 20/07/2015 (next page)
11089                                  <1>         ;
11090 000032BD 39D7                    <1>         cmp    edi, edx
```

```
11091 000032BF 72AC              <1>       jb    short dpt_0
11092                            <1> dpt_p_err:
11093 000032C1 59                <1>       pop   ecx
11094 000032C2 5A                <1>       pop   edx
11095 000032C3 5F                <1>       pop   edi
11096 000032C4 5E                <1>       pop   esi
11097 000032C5 58                <1>       pop   eax ; *
11098                            <1> dpt_err:
11099 000032C6 C3                <1>       retn
11100                            <1>
11101                            <1> page_fault_handler:      ; CPU EXCEPTION 0Eh (14) : Page Fault !
11102                            <1>       ; 21/09/2015
11103                            <1>       ; 19/09/2015
11104                            <1>       ; 17/09/2015
11105                            <1>       ; 28/08/2015
11106                            <1>       ; 20/07/2015
11107                            <1>       ; 28/06/2015
11108                            <1>       ; 03/05/2015
11109                            <1>       ; 30/04/2015
11110                            <1>       ; 18/04/2015
11111                            <1>       ; 12/04/2015
11112                            <1>       ; 30/10/2014
11113                            <1>       ; 11/09/2014
11114                            <1>       ; 10/09/2014 (Retro UNIX 386 v1 - beginning)
11115                            <1>       ;
11116                            <1>       ; Note: This is not an interrupt/exception handler.
11117                            <1>       ;     This is a 'page fault remedy' subroutine
11118                            <1>       ;     which will be called by standard/uniform
11119                            <1>       ;     exception handler.
11120                            <1>       ;
11121                            <1>       ; INPUT ->
11122                            <1>       ;     [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
11123                            <1>       ;
11124                            <1>       ;     cr2 = the virtual (linear) address
11125                            <1>       ;           which has caused to page fault (19/09/2015)
11126                            <1>       ;
11127                            <1>       ; OUTPUT ->
11128                            <1>       ;     (corresponding PAGE TABLE ENTRY is mapped/set)
11129                            <1>       ;     EAX = 0 -> no error
11130                            <1>       ;     EAX > 0 -> error code in EAX (also CF = 1)
11131                            <1>       ;
11132                            <1>       ; Modified Registers -> none (except EAX)
11133                            <1>       ;
11134                            <1>         ;
11135                            <1>         ; ERROR CODE:
11136                            <1>       ;      31  .....   4   3  2   1  0
11137                            <1>       ;     +---+-- --+---+---+---+---+---+---+
11138                            <1>       ;     |   Reserved  | I | R | U | W | P |
11139                            <1>       ;     +---+-- --+---+---+---+---+---+---+
11140                            <1>       ;
11141                            <1>       ; P : PRESENT -     When set, the page fault was caused by
11142                            <1>       ;             a page-protection violation. When not set,
11143                            <1>       ;             it was caused by a non-present page.
11144                            <1>       ; W : WRITE   -     When set, the page fault was caused by
11145                            <1>       ;             a page write. When not set, it was caused
11146                            <1>       ;             by a page read.
11147                            <1>       ; U : USER    -     When set, the page fault was caused
11148                            <1>       ;             while CPL = 3.
11149                            <1>       ;             This does not necessarily mean that
11150                            <1>       ;             the page fault was a privilege violation.
11151                            <1>       ; R : RESERVD -     When set, the page fault was caused by
11152                            <1>       ;     WRITE  reading a 1 in a reserved field.
11153                            <1>       ; I : INSTRUC -     When set, the page fault was caused by
11154                            <1>       ;     FETCH  an instruction fetch
11155                            <1>       ;
11156                            <1>       ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
11157                            <1>       ;  31               22                  12 11                      0
11158                            <1>       ; +------------------+------------------+-----------------------+
11159                            <1>             ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # |        OFFSET         |
11160                            <1>             ; +------------------+------------------+-----------------------+
11161                            <1>       ;
11162                            <1>
11163                            <1>       ;; CR3 REGISTER (Control Register 3)
11164                            <1>       ;  31                                    12             5 4 3 2   0
11165                            <1>       ; +----------------------------------------+------------+---+-----+
11166                            <1>             ; |                                         |            |P|P|     |
11167                            <1>             ; |   PAGE DIRECTORY TABLE BASE ADDRESS     |  reserved  |C|W|rsvrd|
11168                            <1>             ; |                                         |            |D|T|     |
11169                            <1>       ; +----------------------------------------+------------+---+-----+
11170                            <1>       ;
11171                            <1>       ;     PWT    - WRITE THROUGH
11172                            <1>       ;     PCD    - CACHE DISABLE
11173                            <1>       ;
11174                            <1>       ;
11175                            <1>       ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
11176                            <1>       ;  31                                    12 11  9 8 7 6 5 4 3 2 1 0
11177                            <1>       ; +----------------------------------------+-----+---+-+-+-+---+-+-+-+
11178                            <1>             ; |                                         |     | | | | |P|P|U|R| |
11179                            <1>             ; |     PAGE TABLE BASE ADDRESS 31..12      | AVL |G|0|D|A|C|W|/|/|P|
11180                            <1>             ; |                                         |     | | | | |D|T|S|W| |
11181                            <1>       ; +----------------------------------------+-----+---+-+-+-+---+-+-+-+
11182                            <1>       ;
11183                            <1>         ;       P      - PRESENT
11184                            <1>         ;       R/W    - READ/WRITE
11185                            <1>         ;       U/S    - USER/SUPERVISOR
11186                            <1>       ;     PWT    - WRITE THROUGH
11187                            <1>       ;     PCD    - CACHE DISABLE
11188                            <1>       ;     A      - ACCESSED
11189                            <1>         ;       D      - DIRTY (IGNORED)
11190                            <1>       ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
11191                            <1>       ;     G      - GLOBAL    (IGNORED)
11192                            <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
11193                            <1>       ;
11194                            <1>       ;
11195                            <1>       ;; x86 PAGE TABLE ENTRY (4 KByte Page)
```

```
11196                                  <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
11197                                  <1>       ; +-----------------------------------------+-----+---+-+-+-+---+-+-+-+
11198                                  <1>            ; |                                         |     | |P| | |P|P|U|R| |
11199                                  <1>            ; |     PAGE FRAME BASE ADDRESS 31..12      | AVL |G|A|D|A|C|W|/|/|P|
11200                                  <1>            ; |                                         |     | |T| | |D|T|S|W| |
11201                                  <1>       ; +-----------------------------------------+-----+---+-+-+-+---+-+-+-+
11202                                  <1>       ;
11203                                  <1>         ;       P      - PRESENT
11204                                  <1>         ;       R/W    - READ/WRITE
11205                                  <1>         ;       U/S    - USER/SUPERVISOR
11206                                  <1>       ;       PWT    - WRITE THROUGH
11207                                  <1>       ;       PCD    - CACHE DISABLE
11208                                  <1>       ;       A      - ACCESSED
11209                                  <1>         ;       D      - DIRTY
11210                                  <1>       ;       PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
11211                                  <1>       ;       G      - GLOBAL
11212                                  <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
11213                                  <1>       ;
11214                                  <1>       ;
11215                                  <1>       ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
11216                                  <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
11217                                  <1>       ; +-----------------------------------------+-----+-+-+-+-+---+-+-+-+
11218                                  <1>            ; |                                         |     | | | | | | |U|R| |
11219                                  <1>            ; |     PAGE FRAME BASE ADDRESS 31..12      | AVL |0|0|D|A|0|0|/|/|P|
11220                                  <1>            ; |                                         |     | | | | | | |S|W| |
11221                                  <1>            ; +-----------------------------------------+-----+-+-+-+-+---+-+-+-+
11222                                  <1>       ;
11223                                  <1>         ;       P      - PRESENT
11224                                  <1>         ;       R/W    - READ/WRITE
11225                                  <1>         ;       U/S    - USER/SUPERVISOR
11226                                  <1>         ;       D      - DIRTY
11227                                  <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
11228                                  <1>       ;
11229                                  <1>         ;       NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
11230                                  <1>       ;
11231                                  <1>       ;
11232                                  <1>       ;; Invalid Page Table Entry
11233                                  <1>       ; 31                                                            1 0
11234                                  <1>            ; +-------------------------------------------------------------+-+
11235                                  <1>            ; |                                                             | |
11236                                  <1>            ; |                            AVAILABLE                        |0|
11237                                  <1>            ; |                                                             | |
11238                                  <1>            ; +-------------------------------------------------------------+-+
11239                                  <1>       ;
11240                                  <1>
11241 000032C7 53                      <1>       push   ebx
11242 000032C8 52                      <1>       push   edx
11243 000032C9 51                      <1>       push   ecx
11244                                  <1>       ;
11245                                  <1>       ; 21/09/2015 (debugging)
11246 000032CA FF05[B1740000]          <1>       inc    dword [u.pfcount] ; page fault count for running process
11247 000032D0 FF05[30850000]          <1>       inc    dword [PF_Count] ; total page fault count
11248                                  <1>       ; 28/06/2015
11249                                  <1>       ;mov   edx, [error_code] ; Lower 5 bits are valid
11250 000032D6 8A15[28850000]          <1>       mov    dl, [error_code]
11251                                  <1>       ;
11252 000032DC F6C201                  <1>       test   dl, 1  ; page fault was caused by a non-present page
11253                                  <1>                     ; sign
11254 000032DF 7422                    <1>       jz     short pfh_alloc_np
11255                                  <1>       ;
11256                                  <1>       ; If it is not a 'write on read only page' type page fault
11257                                  <1>       ; major page fault error with minor reason must be returned without
11258                                  <1>       ; fixing the problem. 'sys_exit with error' will be needed
11259                                  <1>       ; after return here!
11260                                  <1>       ; Page fault will be remedied, by copying page contents
11261                                  <1>       ; to newly allocated page with write permission;
11262                                  <1>       ; sys_fork -> sys_exec -> copy on write, demand paging method is
11263                                  <1>       ; used for working with minimum possible memory usage.
11264                                  <1>       ; sys_fork will duplicate page directory and tables of parent
11265                                  <1>       ; process with 'read only' flag. If the child process attempts to
11266                                  <1>       ; write on these read only pages, page fault will be directed here
11267                                  <1>       ; for allocating a new page with same data/content.
11268                                  <1>       ;
11269                                  <1>       ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
11270                                  <1>       ; will not force to separate CODE and DATA space
11271                                  <1>       ; in a process/program...
11272                                  <1>       ; CODE segment/section may contain DATA!
11273                                  <1>       ; It is flat, smoth and simplest programming method already as in
11274                                  <1>       ; Retro UNIX 8086 v1 and MS-DOS programs.
11275                                  <1>       ;
11276 000032E1 F6C202                  <1>       test   dl, 2  ; page fault was caused by a page write
11277                                  <1>                     ; sign
11278 000032E4 0F84AB000000            <1>         jz     pfh_p_err
11279                                  <1>       ; 31/08/2015
11280 000032EA F6C204                  <1>       test   dl, 4  ; page fault was caused while CPL = 3 (user mode)
11281                                  <1>                     ; sign.  (U+W+P = 4+2+1 = 7)
11282 000032ED 0F84A2000000            <1>         jz   pfh_pv_err
11283                                  <1>       ;
11284                                  <1>       ; make a new page and copy the parent's page content
11285                                  <1>       ; as the child's new page content
11286                                  <1>       ;
11287 000032F3 0F20D3                  <1>       mov    ebx, cr2 ; CR2 contains the linear address
11288                                  <1>                       ; which has caused to page fault
11289 000032F6 E8A2000000              <1>       call   copy_page
11290 000032FB 0F828D000000            <1>         jc     pfh_im_err ; insufficient memory
11291                                  <1>       ;
11292 00003301 EB7D                    <1>         jmp    pfh_cpp_ok
11293                                  <1>       ;
11294                                  <1> pfh_alloc_np:
11295 00003303 E8EDFCFFFF              <1>       call   allocate_page; (allocate a new page)
11296 00003308 0F8280000000            <1>         jc     pfh_im_err      ; 'insufficient memory' error
11297                                  <1> pfh_chk_cpl:
11298                                  <1>       ; EAX = Physical (base) address of the allocated (new) page
11299                                  <1>            ; (Lower 12 bits are ZERO, because
11300                                  <1>            ;     the address is on a page boundary)
```

```
11301 0000330E 80E204              <1>       and    dl, 4  ; CPL = 3 ?
11302 00003311 7505                <1>       jnz    short pfh_um
11303                              <1>                     ; Page fault handler for kernel/system mode (CPL=0)
11304 00003313 0F20DB              <1>       mov    ebx, cr3 ; CR3 (Control Register 3) contains physical address
11305                              <1>                       ; of the current/active page directory
11306                              <1>                       ; (Always kernel/system mode page directory, here!)
11307                              <1>                       ; Note: Lower 12 bits are 0. (page boundary)
11308 00003316 EB06                <1>       jmp    short pfh_get_pde
11309                              <1>       ;
11310                              <1> pfh_um:                  ; Page fault handler for user/appl. mode (CPL=3)
11311 00003318 8B1D[A1740000]      <1>       mov    ebx, [u.pgdir] ; Page directory of current/active process
11312                              <1>                     ; Physical address of the USER's page directory
11313                              <1>                     ; Note: Lower 12 bits are 0. (page boundary)
11314                              <1> pfh_get_pde:
11315 0000331E 80CA03              <1>       or     dl, 3  ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
11316 00003321 0F20D1              <1>       mov    ecx, cr2 ; CR2 contains the virtual address
11317                              <1>                       ; which has been caused to page fault
11318                              <1>                       ;
11319 00003324 C1E914              <1>       shr    ecx, 20       ; shift 20 bits right
11320 00003327 80E1FC              <1>       and    cl, 0FCh ; mask lower 2 bits to get PDE offset
11321                              <1>       ;
11322 0000332A 01CB                <1>       add    ebx, ecx ; now, EBX points to the relevant page dir entry
11323 0000332C 8B0B                <1>       mov    ecx, [ebx] ; physical (base) address of the page table
11324 0000332E F6C101              <1>       test   cl, 1   ; check bit 0 is set (1) or not (0).
11325 00003331 740B                <1>       jz     short pfh_set_pde ; Page directory entry is not valid,
11326                              <1>                                ; set/validate page directory entry
11327 00003333 6681E100F0          <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
11328 00003338 89CB                <1>       mov    ebx, ecx ; Physical address of the page table
11329 0000333A 89C1                <1>       mov    ecx, eax ; new page address (physical)
11330 0000333C EB16                <1>       jmp    short pfh_get_pte
11331                              <1> pfh_set_pde:
11332                              <1>       ;; NOTE: Page directories and page tables never be swapped out!
11333                              <1>       ;;     (So, we know this PDE is empty or invalid)
11334                              <1>       ;
11335 0000333E 08D0                <1>       or     al, dl  ; lower 3 bits are used as U/S, R/W, P flags
11336 00003340 8903                <1>       mov    [ebx], eax ; Let's put the new page directory entry here !
11337 00003342 30C0                <1>       xor    al, al  ; clear lower (3..8) bits
11338 00003344 89C3                <1>       mov    ebx, eax
11339 00003346 E8AAFCFFFF          <1>       call   allocate_page ; (allocate a new page)
11340 0000334B 7241                <1>       jc     short pfh_im_err   ; 'insufficient memory' error
11341                              <1> pfh_spde_1:
11342                              <1>       ; EAX = Physical (base) address of the allocated (new) page
11343 0000334D 89C1                <1>       mov    ecx, eax
11344 0000334F E81BFDFFFF          <1>       call   clear_page ; Clear page content
11345                              <1> pfh_get_pte:
11346 00003354 0F20D0              <1>       mov    eax, cr2 ; virtual address
11347                              <1>                       ; which has been caused to page fault
11348 00003357 89C7                <1>       mov    edi, eax ; 20/07/2015
11349 00003359 C1E80C              <1>       shr    eax, 12       ; shift 12 bit right to get
11350                              <1>                       ; higher 20 bits of the page fault address
11351 0000335C 25FF030000          <1>       and    eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
11352 00003361 C1E002              <1>       shl    eax, 2 ; shift 2 bits left to get PTE offset
11353 00003364 01C3                <1>       add    ebx, eax ; now, EBX points to the relevant page table entry
11354 00003366 8B03                <1>       mov    eax, [ebx] ; get previous value of pte
11355                              <1>              ; bit 0 of EAX is always 0 (otherwise we would not be here)
11356 00003368 21C0                <1>       and    eax, eax
11357 0000336A 7410                <1>       jz     short pfh_gpte_1
11358                              <1>       ; 20/07/2015
11359 0000336C 87D9                <1>       xchg   ebx, ecx ; new page address (physical)
11360 0000336E 55                  <1>       push   ebp ; 20/07/2015
11361 0000336F 0F20D5              <1>       mov    ebp, cr2
11362                              <1>              ; ECX = physical address of the page table entry
11363                              <1>              ; EBX = Memory page address (physical!)
11364                              <1>              ; EAX = Swap disk (offset) address
11365                              <1>              ; EBP = virtual address (page fault address)
11366 00003372 E8B7000000          <1>       call   swap_in
11367 00003377 5D                  <1>       pop    ebp
11368 00003378 7210                <1>       jc      short pfh_err_retn
11369 0000337A 87CB                <1>       xchg   ecx, ebx
11370                              <1>              ; EBX = physical address of the page table entry
11371                              <1>              ; ECX = new page
11372                              <1> pfh_gpte_1:
11373 0000337C 08D1                <1>       or     cl, dl ; lower 3 bits are used as U/S, R/W, P flags
11374 0000337E 890B                <1>       mov    [ebx], ecx ; Let's put the new page table entry here !
11375                              <1> pfh_cpp_ok:
11376                              <1>       ; 20/07/2015
11377 00003380 0F20D3              <1>       mov    ebx, cr2
11378 00003383 E82A020000          <1>       call   add_to_swap_queue
11379                              <1>       ;
11380                              <1>       ; The new PTE (which contains the new page) will be added to
11381                              <1>       ; the swap queue, here.
11382                              <1>       ; (Later, if memory will become insufficient,
11383                              <1>       ; one page will be swapped out which is at the head of
11384                              <1>       ; the swap queue by using FIFO and access check methods.)
11385                              <1>       ;
11386 00003388 31C0                <1>       xor    eax, eax  ; 0
11387                              <1>       ;
11388                              <1> pfh_err_retn:
11389 0000338A 59                  <1>       pop    ecx
11390 0000338B 5A                  <1>       pop    edx
11391 0000338C 5B                  <1>       pop    ebx
11392 0000338D C3                  <1>       retn
11393                              <1>
11394                              <1> pfh_im_err:
11395 0000338E B8E1000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
11396                              <1>                     ; Major (Primary) Error: Page Fault
11397                              <1>                     ; Minor (Secondary) Error: Insufficient Memory !
11398 00003393 EBF5                <1>       jmp    short pfh_err_retn
11399                              <1>
11400                              <1>
11401                              <1> pfh_p_err: ; 09/03/2015
11402                              <1> pfh_pv_err:
11403                              <1>       ; Page fault was caused by a protection-violation
11404 00003395 B8E3000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
11405                              <1>                     ; Major (Primary) Error: Page Fault
```

```
11406                                  <1>                     ; Minor (Secondary) Error: Protection violation !
11407 0000339A F9                      <1>       stc
11408 0000339B EBED                    <1>       jmp    short pfh_err_retn
11409                                  <1>
11410                                  <1> copy_page:
11411                                  <1>       ; 22/09/2015
11412                                  <1>       ; 21/09/2015
11413                                  <1>       ; 19/09/2015
11414                                  <1>       ; 07/09/2015
11415                                  <1>       ; 31/08/2015
11416                                  <1>       ; 20/07/2015
11417                                  <1>       ; 05/05/2015
11418                                  <1>       ; 03/05/2015
11419                                  <1>       ; 18/04/2015
11420                                  <1>       ; 12/04/2015
11421                                  <1>       ; 30/10/2014
11422                                  <1>       ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
11423                                  <1>       ;
11424                                  <1>       ; INPUT ->
11425                                  <1>       ;     EBX = Virtual (linear) address of source page
11426                                  <1>       ;           (Page fault address)
11427                                  <1>       ; OUTPUT ->
11428                                  <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
11429                                  <1>       ;     (corresponding PAGE TABLE ENTRY is mapped/set)
11430                                  <1>       ;     EAX = 0 (CF = 1)
11431                                  <1>       ;            if there is not a free page to be allocated
11432                                  <1>       ;     (page content of the source page will be copied
11433                                  <1>       ;     onto the target/new page)
11434                                  <1>       ;
11435                                  <1>       ; Modified Registers -> ecx, ebx (except EAX)
11436                                  <1>       ;
11437 0000339D 56                      <1>       push   esi
11438 0000339E 57                      <1>       push   edi
11439                                  <1>       ;push  ebx
11440                                  <1>       ;push  ecx
11441 0000339F 31F6                    <1>       xor    esi, esi
11442 000033A1 C1EB0C                  <1>       shr    ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
11443 000033A4 89D9                    <1>       mov    ecx, ebx ; save page fault address (as 12 bit shifted)
11444 000033A6 C1EB08                  <1>       shr    ebx, 8  ; shift 8 bits right and then
11445 000033A9 80E3FC                  <1>       and    bl, 0FCh ; mask lower 2 bits to get PDE offset
11446 000033AC 89DF                    <1>       mov    edi, ebx ; save it for the parent of current process
11447 000033AE 031D[A1740000]          <1>       add    ebx, [u.pgdir] ; EBX points to the relevant page dir entry
11448 000033B4 8B03                    <1>       mov    eax, [ebx] ; physical (base) address of the page table
11449 000033B6 662500F0                <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11450 000033BA 89CB                    <1>       mov    ebx, ecx   ; (restore higher 20 bits of page fault address)
11451 000033BC 81E3FF030000            <1>       and    ebx, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
11452 000033C2 66C1E302                <1>       shl    bx, 2     ; shift 2 bits left to get PTE offset
11453 000033C6 01C3                    <1>       add    ebx, eax   ; EBX points to the relevant page table entry
11454                                  <1>       ; 07/09/2015
11455 000033C8 66F7030002              <1>         test    word [ebx], PTE_DUPLICATED ; (Does current process share this
11456                                  <1>                                ; read only page as a child process?)
11457 000033CD 7509                    <1>       jnz    short cpp_0 ; yes
11458 000033CF 8B0B                    <1>       mov    ecx, [ebx] ; PTE value
11459 000033D1 6681E100F0              <1>       and    cx, PTE_A_CLEAR ; 0F000h  ; clear page attributes
11460 000033D6 EB32                    <1>       jmp    short cpp_1
11461                                  <1> cpp_0:
11462 000033D8 89FE                    <1>       mov    esi, edi
11463 000033DA 0335[A5740000]          <1>       add    esi, [u.ppgdir] ; the parent's page directory entry
11464 000033E0 8B06                    <1>       mov    eax, [esi] ; physical (base) address of the page table
11465 000033E2 662500F0                <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11466 000033E6 89CE                    <1>       mov    esi, ecx   ; (restore higher 20 bits of page fault address)
11467 000033E8 81E6FF030000            <1>       and    esi, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
11468 000033EE 66C1E602                <1>       shl    si, 2     ; shift 2 bits left to get PTE offset
11469 000033F2 01C6                    <1>       add    esi, eax   ; EDX points to the relevant page table entry
11470 000033F4 8B0E                    <1>       mov    ecx, [esi] ; PTE value of the parent process
11471                                  <1>       ; 21/09/2015
11472 000033F6 8B03                    <1>       mov    eax, [ebx] ; PTE value of the child process
11473 000033F8 662500F0                <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
11474                                  <1>       ;
11475 000033FC F6C101                  <1>       test   cl, PTE_A_PRESENT ; is it a present/valid page ?
11476 000033FF 7424                    <1>       jz     short cpp_3 ; the parent's page is not same page
11477                                  <1>       ;
11478 00003401 6681E100F0              <1>       and    cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
11479 00003406 39C8                    <1>       cmp    eax, ecx   ; Same page?
11480 00003408 751B                    <1>       jne    short cpp_3 ; Parent page and child page are not same
11481                                  <1>                          ; Convert child's page to writable page
11482                                  <1> cpp_1:
11483 0000340A E8E6FBFFFF              <1>       call   allocate_page
11484 0000340F 721A                    <1>       jc     short cpp_4 ; 'insufficient memory' error
11485 00003411 21F6                    <1>       and    esi, esi    ; check ESI is valid or not
11486 00003413 7405                    <1>       jz     short cpp_2
11487                                  <1>              ; Convert read only page to writable page
11488                                  <1>              ;(for the parent of the current process)
11489                                  <1>       ;and   word [esi], PTE_A_CLEAR ; 0F000h
11490                                  <1>       ; 22/09/2015
11491 00003415 890E                    <1>       mov    [esi], ecx
11492 00003417 800E07                  <1>       or     byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
11493                                  <1>                                ; 1+2+4 = 7
11494                                  <1> cpp_2:
11495 0000341A 89C7                    <1>       mov    edi, eax ; new page address of the child process
11496                                  <1>       ; 07/09/2015
11497 0000341C 89CE                    <1>       mov    esi, ecx ; the page address of the parent process
11498 0000341E B900040000              <1>       mov    ecx, PAGE_SIZE / 4
11499 00003423 F3A5                    <1>       rep    movsd ; 31/08/2015
11500                                  <1> cpp_3:
11501 00003425 0C07                    <1>       or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
11502 00003427 8903                    <1>       mov    [ebx], eax ; Update PTE
11503 00003429 28C0                    <1>       sub    al, al ; clear attributes
11504                                  <1> cpp_4:
11505                                  <1>       ;pop   ecx
11506                                  <1>       ;pop   ebx
11507 0000342B 5F                      <1>       pop    edi
11508 0000342C 5E                      <1>       pop    esi
11509 0000342D C3                      <1>       retn
11510                                  <1>
```

```
11511                                  <1> ;; 28/04/2015
11512                                  <1> ;; 24/10/2014
11513                                  <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
11514                                  <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
11515                                  <1> ;;
11516                                  <1> ;;   0000   0001   0002   0003   ....   1020   1021   1022   1023
11517                                  <1> ;; +------+------+------+------+-    -+------+------+------+------+
11518                                  <1> ;; |  pg1 |  pg2 |  pg3 |  pg4 | .... |pg1021|pg1022|pg1023|pg1024|
11519                                  <1> ;; +------+------+------+------+-    -+------+------+------+------+
11520                                  <1> ;;
11521                                  <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
11522                                  <1> ;;
11523                                  <1> ;; Method:
11524                                  <1> ;;    Swap page queue is a list of allocated pages with physical
11525                                  <1> ;;    addresses (system mode virtual adresses = physical addresses).
11526                                  <1> ;;    It is used for 'swap_in' and 'swap_out' procedures.
11527                                  <1> ;;    When a new page is being allocated, swap queue is updated
11528                                  <1> ;;    by 'swap_queue_shift' procedure, header of the queue (offset 0)
11529                                  <1> ;;    is checked for 'accessed' flag. If the 1st page on the queue
11530                                  <1> ;;    is 'accessed' or 'read only', it is dropped from the list;
11531                                  <1> ;;    other pages from the 2nd to the last (in [swpq_last]) shifted
11532                                  <1> ;;    to head then the 2nd page becomes the 1st and '[swpq_last]'
11533                                  <1> ;;    offset value becomes it's previous offset value - 4.
11534                                  <1> ;;    If the 1st page of the swap page queue is not 'accessed'
11535                                  <1> ;;    the queue/list is not shifted.
11536                                  <1> ;;    After the queue/list shift, newly allocated page is added
11537                                  <1> ;;    to the tail of the queue at the [swpq_count*4] position.
11538                                  <1> ;;    But, if [swpq_count] > 1023, the newly allocated page
11539                                  <1> ;;    will not be added to the tail of swap page queue.
11540                                  <1> ;;
11541                                  <1> ;;    During 'swap_out' procedure, swap page queue is checked for
11542                                  <1> ;;    the first non-accessed, writable page in the list,
11543                                  <1> ;;    from the head to the tail. The list is shifted to left
11544                                  <1> ;;    (to the head) till a non-accessed page will be found in the list.
11545                                  <1> ;;    Then, this page     is swapped out (to disk) and then it is dropped
11546                                  <1> ;;    from the list by a final swap queue shift. [swpq_count] value
11547                                  <1> ;;    is changed. If all pages on the queue' are 'accessed',
11548                                  <1> ;;    'insufficient memory' error will be returned ('swap_out'
11549                                  <1> ;;    procedure will be failed)...
11550                                  <1> ;;
11551                                  <1> ;;    Note: If the 1st page of the queue is an 'accessed' page,
11552                                  <1> ;;    'accessed' flag of the page will be reset (0) and that page
11553                                  <1> ;;    (PTE) will be added to the tail of the queue after
11554                                  <1> ;;    the check, if [swpq_count] < 1023. If [swpq_count] = 1024
11555                                  <1> ;;    the queue will be rotated and the PTE in the head will be
11556                                  <1> ;;    added to the tail after resetting 'accessed' bit.
11557                                  <1> ;;
11558                                  <1> ;;
11559                                  <1> ;;
11560                                  <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
11561                                  <1> ;;
11562                                  <1> ;;  00000000  00000004  00000008  0000000C   ...   size-8    size-4
11563                                  <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
11564                                  <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
11565                                  <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
11566                                  <1> ;;
11567                                  <1> ;; [swpd_next] = the first free block address in swapped page records
11568                                  <1> ;;                  for next free block search by 'swap_out' procedure.
11569                                  <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
11570                                  <1> ;;           NOTE: max. possible swap disk size is 1024 GB
11571                                  <1> ;;           (entire swap space must be accessed by using
11572                                  <1> ;;           31 bit offset address)
11573                                  <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
11574                                  <1> ;; [swpd_start] = absolute/start address of the swap disk/file
11575                                  <1> ;;            0 for file, or beginning sector of the swap partition
11576                                  <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
11577                                  <1> ;;
11578                                  <1> ;;
11579                                  <1> ;; Method:
11580                                  <1> ;;    When the memory (ram) becomes insufficient, page allocation
11581                                  <1> ;;    procedure swaps out a page from memory to the swap disk
11582                                  <1> ;;    (partition) or swap file to get a new free page at the memory.
11583                                  <1> ;;    Swapping out is performed by using swap page queue.
11584                                  <1> ;;
11585                                  <1> ;;    Allocation block size of swap disk/file is equal to page size
11586                                  <1> ;;    (4096 bytes). Swapping address (in sectors) is recorded
11587                                  <1> ;;    into relevant page file entry as 31 bit physical (logical)
11588                                  <1> ;;    offset address as 1 bit shifted to left for present flag (0).
11589                                  <1> ;;    Swapped page address is between 1 and swap disk/file size - 4.
11590                                  <1> ;;    Absolute physical (logical) address of the swapped page is
11591                                  <1> ;;    calculated by adding offset value to the swap partition's
11592                                  <1> ;;    start address. If the swap device (disk) is a virtual disk
11593                                  <1> ;;    or it is a file, start address of the swap disk/volume is 0,
11594                                  <1> ;;    and offset value is equal to absolute (physical or logical)
11595                                  <1> ;;    address/position. (It has not to be ZERO if the swap partition
11596                                  <1> ;;    is in a partitioned virtual hard disk.)
11597                                  <1> ;;
11598                                  <1> ;;    Note: Swap addresses are always specified/declared in sectors,
11599                                  <1> ;;    not in bytes or     in blocks/zones/clusters (4096 bytes) as unit.
11600                                  <1> ;;
11601                                  <1> ;;    Swap disk/file allocation is mapped via 'Swap Allocation Table'
11602                                  <1> ;;    at memory as similar to 'Memory Allocation Table'.
11603                                  <1> ;;
11604                                  <1> ;;    Every bit of Swap Allocation Table repsesents one swap block
11605                                  <1> ;;    (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
11606                                  <1> ;;    is reserved for swap disk/file block 0 as descriptor block
11607                                  <1> ;;    (also for compatibility with PTE). If bit value is ZERO,
11608                                  <1> ;;    it means relevant (respective) block is in use, and,
11609                                  <1> ;;    of course, if bit value is 1, it means relevant (respective)
11610                                  <1> ;;      swap disk/file block is free.
11611                                  <1> ;;    For example: bit 1 of the byte 128 repsesents block 1025
11612                                  <1> ;;    (128*8+1) or sector (offset) 8200 on the swap disk or
11613                                  <1> ;;    byte (offset/position) 4198400 in the swap file.
11614                                  <1> ;;    4GB swap space is represented via 128KB Swap Allocation Table.
11615                                  <1> ;;    Initial layout of Swap Allocation Table is as follows:
```

```
11616                                <1> ;;   ------------------------------------------------------------
11617                                <1> ;;   01111111111111111111111111 .... 11111111111111111111111111111111
11618                                <1> ;;   ------------------------------------------------------------
11619                                <1> ;;   (0 is reserved block, 1s represent free blocks respectively.)
11620                                <1> ;;   (Note: Allocation cell/unit of the table is bit, not byte)
11621                                <1> ;;
11622                                <1> ;;   ............................................................
11623                                <1> ;;
11624                                <1> ;;   'swap_out' procedure checks 'free_swap_blocks' count at first,
11625                                <1> ;;   then it searches Swap Allocation Table if free count is not
11626                                <1> ;;   zero. From begining the [swpd_next] dword value, the first bit
11627                                <1> ;;   position with value of 1 on the table is converted to swap
11628                                <1> ;;   disk/file offset address, in sectors (not 4096 bytes block).
11629                                <1> ;;   'ldrv_write' procedure is called with ldrv (logical drive
11630                                <1> ;;   number of physical swap disk or virtual swap disk)
11631                                <1> ;;   number, sector offset (not absolute sector -LBA- number),
11632                                <1> ;;   and sector count (8, 512*8 = 4096) and buffer adress
11633                                <1> ;;   (memory page). That will be a direct disk write procedure.
11634                                <1> ;;   (for preventing late memory allocation, significant waiting).
11635                                <1> ;;   If disk write procedure returns with error or free count of
11636                                <1> ;;   swap blocks is ZERO, 'swap_out' procedure will return with
11637                                <1> ;;   'insufficient memory error' (cf=1).
11638                                <1> ;;
11639                                <1> ;;   (Note: Even if free swap disk/file blocks was not zero,
11640                                <1> ;;   any disk write error will not be fixed by 'swap_out' procedure,
11641                                <1> ;;   in other words, 'swap_out' will not check the table for other
11642                                <1> ;;   free blocks after a disk write error. It will return to
11643                                <1> ;;   the caller with error (CF=1) which means swapping is failed.
11644                                <1> ;;
11645                                <1> ;;   After writing the page on to swap disk/file address/sector,
11646                                <1> ;;   'swap_out' procesure returns with that swap (offset) sector
11647                                <1> ;;   address (cf=0).
11648                                <1> ;;
11649                                <1> ;;   ............................................................
11650                                <1> ;;
11651                                <1> ;;   'swap_in' procedure loads addressed (relevant) swap disk or
11652                                <1> ;;   file sectors at specified memory page. Then page allocation
11653                                <1> ;;   procedure updates relevant page table entry with 'present'
11654                                <1> ;;   attribute. If swap disk or file reading fails there is nothing
11655                                <1> ;;   to do, except to terminate the process which is the owner of
11656                                <1> ;;   the swapped page.
11657                                <1> ;;
11658                                <1> ;;   'swap_in' procedure sets the relevant/respective bit value
11659                                <1> ;;   in the Swap Allocation Table (as free block). 'swap_in' also
11660                                <1> ;;   updates [swpd_first] pointer if it is required.
11661                                <1> ;;
11662                                <1> ;;   ............................................................
11663                                <1> ;;
11664                                <1> ;;   Note: If [swap_enabled] value is ZERO, that means there is not
11665                                <1> ;;   a swap disk or swap file in use... 'swap_in' and 'swap_out'
11666                                <1> ;;   procedures ans 'swap page que' procedures will not be active...
11667                                <1> ;;   'Insufficient memory' error will be returned by 'swap_out'
11668                                <1> ;;   and 'general protection fault' will be returned by 'swap_in'
11669                                <1> ;;   procedure, if it is called mistakenly (a wrong value in a PTE).
11670                                <1> ;;
11671                                <1>
11672                                <1> swap_in:
11673                                <1>      ; 31/08/2015
11674                                <1>      ; 20/07/2015
11675                                <1>      ; 28/04/2015
11676                                <1>      ; 18/04/2015
11677                                <1>      ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
11678                                <1>      ;
11679                                <1>      ; INPUT ->
11680                                <1>      ;     EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
11681                                <1>      ;     EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
11682                                <1>      ;     EAX = Offset Address for the swapped page on the
11683                                <1>      ;           swap disk or in the swap file.
11684                                <1>      ;
11685                                <1>      ; OUTPUT ->
11686                                <1>      ;     EAX = 0 if loading at memory has been successful
11687                                <1>      ;
11688                                <1>      ;     CF = 1 -> swap disk reading error (disk/file not present
11689                                <1>      ;              or sector not present or drive not ready
11690                                <1>      ;          EAX = Error code
11691                                <1>      ;          [u.error] = EAX
11692                                <1>      ;                  = The last error code for the process
11693                                <1>      ;                    (will be reset after returning to user)
11694                                <1>      ;
11695                                <1>      ; Modified Registers -> EAX
11696                                <1>      ;
11697                                <1>
11698 0000342E 833D[12850000]00     <1>        cmp     dword [swp_drv], 0
11699 00003435 7648                 <1>      jna    short swpin_dnp_err
11700                                <1>
11701 00003437 3B05[16850000]       <1>      cmp    eax, [swpd_size]
11702 0000343D 734C                 <1>      jnb    short swpin_snp_err
11703                                <1>
11704 0000343F 56                   <1>      push   esi
11705 00003440 53                   <1>      push   ebx
11706 00003441 51                   <1>      push   ecx
11707 00003442 8B35[12850000]       <1>      mov    esi, [swp_drv]
11708 00003448 B908000000           <1>      mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
11709                                <1>             ; Note: Even if corresponding physical disk's sector
11710                                <1>             ; size different than 512 bytes, logical disk sector
11711                                <1>             ; size is 512 bytes and disk reading procedure
11712                                <1>             ; will be performed for reading 4096 bytes
11713                                <1>             ; (2*2048, 8*512).
11714                                <1>      ; ESI = Logical disk description table address
11715                                <1>      ; EBX = Memory page (buffer) address (physical!)
11716                                <1>      ; EAX = Sector adress (offset address, logical sector number)
11717                                <1>      ; ECX = Sector count ; 8 sectors
11718 0000344D 50                   <1>      push   eax
11719 0000344E E833020000           <1>      call   logical_disk_read
11720 00003453 58                   <1>      pop    eax
```

```
11721 00003454 730C                <1>       jnc    short swpin_read_ok
11722                              <1>       ;
11723 00003456 B804000000          <1>       mov    eax, SWP_DISK_READ_ERR ; drive not ready or read error
11724 0000345B A3[9D740000]        <1>       mov    [u.error], eax
11725 00003460 EB19                <1>       jmp    short swpin_retn
11726                              <1>       ;
11727                              <1> swpin_read_ok:
11728                              <1>       ; EAX = Offset address (logical sector number)
11729 00003462 E891010000          <1>       call   unlink_swap_block  ; Deallocate swap block
11730                              <1>       ;
11731                              <1>       ; EBX = Memory page (buffer) address (physical!)
11732                              <1>       ; 20/07/2015
11733 00003467 89EB                <1>       mov    ebx, ebp ; virtual address (page fault address)
11734 00003469 6681E300F0          <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
11735 0000346E 8A1D[97740000]      <1>       mov    bl, [u.uno] ; current process number
11736                              <1>       ; EBX = Virtual address & process number combination
11737 00003474 E89E000000          <1>       call   swap_queue_shift
11738 00003479 29C0                <1>       sub    eax, eax  ; 0 ; Error Code = 0  (no error)
11739                              <1>       ;
11740                              <1> swpin_retn:
11741 0000347B 59                  <1>       pop    ecx
11742 0000347C 5B                  <1>       pop    ebx
11743 0000347D 5E                  <1>       pop    esi
11744 0000347E C3                  <1>       retn
11745                              <1>
11746                              <1> swpin_dnp_err:
11747 0000347F B805000000          <1>       mov    eax, SWP_DISK_NOT_PRESENT_ERR
11748                              <1> swpin_err_retn:
11749 00003484 A3[9D740000]        <1>       mov    [u.error], eax
11750 00003489 F9                  <1>       stc
11751 0000348A C3                  <1>       retn
11752                              <1>
11753                              <1> swpin_snp_err:
11754 0000348B B806000000          <1>       mov    eax, SWP_SECTOR_NOT_PRESENT_ERR
11755 00003490 EBF2                <1>       jmp    short swpin_err_retn
11756                              <1>
11757                              <1> swap_out:
11758                              <1>       ; 31/08/2015
11759                              <1>       ; 05/05/2015
11760                              <1>       ; 30/04/2015
11761                              <1>       ; 28/04/2015
11762                              <1>       ; 18/04/2015
11763                              <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
11764                              <1>       ;
11765                              <1>       ; INPUT ->
11766                              <1>       ;     none
11767                              <1>       ;
11768                              <1>       ; OUTPUT ->
11769                              <1>       ;     EAX = Physical page address (which is swapped out
11770                              <1>       ;           for allocating a new page)
11771                              <1>       ;     CF = 1 -> swap disk writing error (disk/file not present
11772                              <1>       ;             or sector not present or drive not ready
11773                              <1>       ;          EAX = Error code
11774                              <1>       ;          [u.error] = EAX
11775                              <1>       ;                  = The last error code for the process
11776                              <1>       ;                    (will be reset after returning to user)
11777                              <1>       ;
11778                              <1>       ; Modified Registers -> non (except EAX)
11779                              <1>       ;
11780 00003492 66833D[10850000]01  <1>       cmp    word [swpq_count], 1
11781 0000349A 7274                <1>         jc      short swpout_im_err ; 'insufficient memory'
11782                              <1>
11783                              <1>         ;cmp     dword [swp_drv], 1
11784                              <1>       ;jc    short swpout_dnp_err ; 'swap disk/file not present'
11785                              <1>
11786 0000349C 833D[1A850000]01    <1>         cmp     dword [swpd_free], 1
11787 000034A3 7258                <1>       jc     short swpout_nfspc_err ; 'no free space on swap disk'
11788                              <1>
11789 000034A5 53                  <1>       push   ebx
11790                              <1> swpout_1:
11791 000034A6 31DB                <1>       xor    ebx, ebx
11792 000034A8 E86A000000          <1>       call   swap_queue_shift
11793 000034AD 21C0                <1>       and    eax, eax     ; entry count (before shifting)
11794 000034AF 7457                <1>       jz     short swpout_npts_err  ; There is no any PTE in
11795                              <1>                                     ; the swap queue
11796 000034B1 BB00E00800          <1>       mov    ebx, swap_queue           ; Addres of the head of
11797                              <1>                                     ; the swap queue
11798 000034B6 8B03                <1>       mov    eax, [ebx]                ; The PTE in the queue head
11799                              <1>
11800                              <1>       ;test  al, PTE_A_PRESENT      ; bit 0 = 1
11801                              <1>       ;jz    short swpout_1             ; non-present page already
11802                              <1>                                     ; must not be in the queue
11803                              <1>
11804                              <1>       ;test  al, PTE_A_WRITE            ; bit 1 = 0
11805                              <1>       ;jz    short swpout_1             ; read only page (must not be
11806                              <1>                                     ; swapped out)
11807                              <1>
11808 000034B8 A820                <1>       test   al, PTE_A_ACCESS       ; bit 5 = 1 (Accessed)
11809 000034BA 75EA                <1>       jnz    short swpout_1             ; accessed page (must not be
11810                              <1>                                     ; swapped out, at this stage)
11811                              <1>       ;
11812 000034BC 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11813                              <1>       ;
11814 000034C0 52                  <1>       push   edx
11815 000034C1 89DA                <1>       mov    edx, ebx                  ; Page table entry address
11816 000034C3 89C3                <1>       mov    ebx, eax                  ; Buffer (Page) Address
11817                              <1>       ;
11818 000034C5 E861010000          <1>       call   link_swap_block
11819 000034CA 7304                <1>       jnc    short swpout_2             ; It may not be needed here
11820 000034CC 5A                  <1>       pop    edx                 ; because [swpd_free] value
11821 000034CD 5B                  <1>       pop    ebx
11822 000034CE EB2D                <1>       jmp    short swpout_nfspc_err ; was checked at the beginging.
11823                              <1> swpout_2:
11824 000034D0 56                  <1>       push   esi
11825 000034D1 51                  <1>       push   ecx
```

```
11826 000034D2 50                  <1>       push   eax ; sector address
11827 000034D3 8B35[12850000]      <1>       mov    esi, [swp_drv]
11828 000034D9 B908000000          <1>       mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
11829                              <1>              ; Note: Even if corresponding physical disk's sector
11830                              <1>              ; size different than 512 bytes, logical disk sector
11831                              <1>              ; size is 512 bytes and disk writing procedure
11832                              <1>              ; will be performed for writing 4096 bytes
11833                              <1>              ; (2*2048, 8*512).
11834                              <1>       ; ESI = Logical disk description table address
11835                              <1>       ; EBX = Buffer address
11836                              <1>       ; EAX = Sector adress (offset address, logical sector number)
11837                              <1>       ; ECX = Sector count ; 8 sectors
11838 000034DE E8A4010000         <1>       call   logical_disk_write
11839 000034E3 59                  <1>       pop    ecx ; sector address
11840 000034E4 730C                <1>       jnc    short swpout_write_ok
11841                              <1>       ;
11842                              <1>       ;; call      unlink_swap_block ; this block must be left as 'in use'
11843                              <1> swpout_dw_err:
11844 000034E6 B808000000          <1>       mov    eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
11845 000034EB A3[9D740000]        <1>       mov    [u.error], eax
11846 000034F0 EB06                <1>       jmp    short swpout_retn
11847                              <1>       ;
11848                              <1> swpout_write_ok:
11849                              <1>       ; EBX = Buffer (page) address
11850                              <1>       ; EDX = Page Table entry address
11851                              <1>       ; ECX = Swap disk sector (file block) address (31 bit)
11852 000034F2 D1E1                <1>       shl    ecx, 1  ; 31 bit sector address from bit 1 to bit 31
11853 000034F4 890A                <1>       mov    [edx], ecx
11854                              <1>              ; bit 0 = 0 (swapped page)
11855 000034F6 89D8                <1>       mov    eax, ebx
11856                              <1> swpout_retn:
11857 000034F8 59                  <1>       pop    ecx
11858 000034F9 5E                  <1>       pop    esi
11859 000034FA 5A                  <1>       pop    edx
11860 000034FB 5B                  <1>       pop    ebx
11861 000034FC C3                  <1>       retn
11862                              <1>
11863                              <1> ; Note: Swap_queue will not be updated in 'swap_out' procedure
11864                              <1> ;     after the page is swapped out. (the PTE at the queue head
11865                              <1> ;     -with 'non-present' attribute- will be dropped from the
11866                              <1> ;     the queue in next 'swap_out' or in next 'swap_queue_shift'.
11867                              <1>
11868                              <1> ;swpout_dnp_err:
11869                              <1> ;     mov    eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
11870                              <1> ;     jmp    short swpout_err_retn
11871                              <1> swpout_nfspc_err:
11872 000034FD B807000000          <1>       mov    eax, SWP_NO_FREE_SPACE_ERR ; no free space
11873                              <1> swpout_err_retn:
11874 00003502 A3[9D740000]        <1>       mov    [u.error], eax
11875                              <1>       ;stc
11876 00003507 C3                  <1>       retn
11877                              <1> swpout_npts_err:
11878 00003508 B809000000          <1>       mov    eax, SWP_NO_PAGE_TO_SWAP_ERR
11879 0000350D 5B                  <1>       pop    ebx
11880 0000350E EBF2                <1>       jmp    short swpout_err_retn
11881                              <1> swpout_im_err:
11882 00003510 B801000000          <1>       mov    eax, ERR_MINOR_IM ; insufficient (out of) memory
11883 00003515 EBEB                <1>       jmp    short swpout_err_retn
11884                              <1>
11885                              <1> swap_queue_shift:
11886                              <1>       ; 20/07/2015
11887                              <1>       ; 28/04/2015
11888                              <1>       ; 18/04/2015
11889                              <1>       ; 23/10/2014 (Retro UNIX 386 v1 - beginning)
11890                              <1>       ;
11891                              <1>       ; INPUT ->
11892                              <1>       ;     EBX = Virtual (linear) address (bit 12 to 31)
11893                              <1>       ;           and process number combination (bit 0 to 11)
11894                              <1>       ;     EBX = 0 -> shift/drop from the head (offset 0)
11895                              <1>       ; OUTPUT ->
11896                              <1>       ;     If EBX input > 0
11897                              <1>       ;         the queue will be shifted 4 bytes (dword),
11898                              <1>       ;         from the tail to the head, up to entry offset
11899                              <1>       ;         which points to EBX input value or nothing
11900                              <1>       ;         to do if EBX value is not found in the queue.
11901                              <1>       ;         (The entry -with EBX value- will be removed
11902                              <1>       ;          from the queue if it is found.)
11903                              <1>       ;     If EBX input = 0
11904                              <1>       ;         the queue will be shifted 4 bytes (dword),
11905                              <1>       ;         from the tail to the head, if the PTE address
11906                              <1>       ;         in head of the queue is marked as "accessed"
11907                              <1>       ;         or it is marked as "non present".
11908                              <1>       ;         (If "accessed" flag of the PTE -in the head-
11909                              <1>       ;         is set -to 1-, it will be reset -to 0- and then,
11910                              <1>       ;         the queue will be rotated -without dropping
11911                              <1>       ;         the PTE from the queue-, for 4 bytes on head
11912                              <1>       ;         to tail direction. The PTE in the head will be
11913                              <1>       ;         moved in the tail, other PTEs will be shifted on
11914                              <1>       ;         head direction.)
11915                              <1>       ;
11916                              <1>       ;     EAX = [swpq_count] (before the shifting)
11917                              <1>       ;         (EAX = 0 -> next 'swap_out' stage
11918                              <1>       ;          is not applicable)
11919                              <1>       ;
11920                              <1>       ; Modified Registers -> EAX
11921                              <1>       ;
11922 00003517 0FB705[10850000]    <1>       movzx   eax, word [swpq_count]  ; Max. 1024
11923 0000351E 6621C0              <1>       and    ax, ax
11924 00003521 7433                <1>       jz     short swpqs_retn
11925 00003523 57                  <1>       push   edi
11926 00003524 56                  <1>       push   esi
11927 00003525 53                  <1>       push   ebx
11928 00003526 51                  <1>       push   ecx
11929 00003527 50                  <1>       push   eax
11930 00003528 BE00E00800          <1>       mov    esi, swap_queue
```

```
11931 0000352D 89C1                <1>       mov   ecx, eax
11932 0000352F 09DB                <1>       or    ebx, ebx
11933 00003531 7424                <1>       jz    short swpqs_7
11934                              <1> swpqs_1:
11935 00003533 AD                  <1>       lodsd
11936 00003534 39D8                <1>       cmp   eax, ebx
11937 00003536 7404                <1>       je    short swpqs_2
11938 00003538 E2F9                <1>       loop  swpqs_1
11939 0000353A EB15                <1>       jmp   short swpqs_6
11940                              <1> swpqs_2:
11941 0000353C 89F7                <1>       mov   edi, esi
11942 0000353E 83EF04              <1>       sub   edi, 4
11943                              <1> swpqs_3:
11944 00003541 66FF0D[10850000]    <1>       dec   word [swpq_count]
11945 00003548 7403                <1>       jz    short swpqs_5
11946                              <1> swpqs_4:
11947 0000354A 49                  <1>       dec   ecx
11948 0000354B F3A5                <1>       rep   movsd  ; shift up (to the head)
11949                              <1> swpqs_5:
11950 0000354D 31C0                <1>       xor   eax, eax
11951 0000354F 8907                <1>       mov   [edi], eax
11952                              <1> swpqs_6:
11953 00003551 58                  <1>       pop   eax
11954 00003552 59                  <1>       pop   ecx
11955 00003553 5B                  <1>       pop   ebx
11956 00003554 5E                  <1>       pop   esi
11957 00003555 5F                  <1>       pop   edi
11958                              <1> swpqs_retn:
11959 00003556 C3                  <1>       retn
11960                              <1> swpqs_7:
11961 00003557 89F7                <1>       mov   edi, esi ; head
11962 00003559 AD                  <1>       lodsd
11963                              <1>       ; 20/07/2015
11964 0000355A 89C3                <1>       mov   ebx, eax
11965 0000355C 81E300F0FFFF        <1>       and   ebx, ~PAGE_OFF ; ~0FFFh
11966                              <1>                ; ebx = virtual address (at page boundary)
11967 00003562 25FF0F0000          <1>       and   eax, PAGE_OFF ; 0FFFh
11968                              <1>                ; ax = process number (1 to 4095)
11969 00003567 3A05[97740000]      <1>       cmp   al, [u.uno]
11970                              <1>             ; Max. 16 (nproc) processes for Retro UNIX 386 v1
11971 0000356D 7507                <1>       jne   short swpqs_8
11972 0000356F A1[A1740000]        <1>       mov   eax, [u.pgdir]
11973 00003574 EB16                <1>       jmp   short swpqs_9
11974                              <1> swpqs_8:
11975                              <1>       ;shl  ax, 2
11976 00003576 C0E002              <1>       shl   al, 2
11977 00003579 8B80[D2710000]      <1>       mov   eax, [eax+p.upage-4]
11978 0000357F 09C0                <1>       or    eax, eax
11979 00003581 74BE                <1>       jz    short swpqs_3 ; invalid upage
11980 00003583 83C061              <1>       add   eax, u.pgdir - user
11981                              <1>                ; u.pgdir value for the process
11982                              <1>                ; is in [eax]
11983 00003586 8B00                <1>       mov   eax, [eax]
11984 00003588 21C0                <1>       and   eax, eax
11985 0000358A 74B5                <1>       jz    short swpqs_3 ; invalid page directory
11986                              <1> swpqs_9:
11987 0000358C 52                  <1>       push  edx
11988                              <1>       ; eax = page directory
11989                              <1>       ; ebx = virtual address
11990 0000358D E87EFBFFFF          <1>       call  get_pte
11991 00003592 89D3                <1>       mov   ebx, edx ; PTE address
11992 00003594 5A                  <1>       pop   edx
11993 00003595 72AA                <1>       jc    short swpqs_3 ; empty PDE
11994                              <1>       ; EAX = PTE value
11995 00003597 A801                <1>       test  al, PTE_A_PRESENT ; bit 0 = 1
11996 00003599 74A6                <1>       jz    short swpqs_3 ; Drop non-present page
11997                              <1>                           ; from the queue (head)
11998 0000359B A802                <1>       test  al, PTE_A_WRITE       ; bit 1 = 0
11999 0000359D 74A2                <1>       jz    short swpqs_3 ; Drop read only page
12000                              <1>                           ; from the queue (head)
12001                              <1>       ;test  al, PTE_A_ACCESS  ; bit 5 = 1 (Accessed)
12002                              <1>       ;jz    short swpqs_6 ; present
12003                              <1>                           ; non-accessed page
12004 0000359F 0FBAF005            <1>         btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
12005 000035A3 73AC                <1>       jnc   short swpqs_6  ; non-accessed page
12006 000035A5 8903                <1>       mov   [ebx], eax     ; save changed attribute
12007                              <1>       ;
12008                              <1>       ; Rotation (head -> tail)
12009 000035A7 49                  <1>       dec   ecx     ; entry count -> last entry number
12010 000035A8 74A7                <1>       jz    short swpqs_6
12011                              <1>             ; esi = head + 4
12012                              <1>             ; edi = head
12013 000035AA 8B07                <1>       mov   eax, [edi] ; 20/07/2015
12014 000035AC F3A5                <1>       rep   movsd  ; n = 1 to k-1, [n - 1] = [n]
12015 000035AE 8907                <1>       mov   [edi], eax ; head -> tail ; [k] = [1]
12016 000035B0 EB9F                <1>       jmp   short swpqs_6
12017                              <1>
12018                              <1> add_to_swap_queue:
12019                              <1> ; temporary - 16/09/2015
12020 000035B2 C3                  <1> retn
12021                              <1>       ; 20/07/2015
12022                              <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12023                              <1>       ;
12024                              <1>       ; Adds new page to swap queue
12025                              <1>       ; (page directories and page tables must not be added
12026                              <1>       ; to swap queue)
12027                              <1>       ;
12028                              <1>       ; INPUT ->
12029                              <1>       ;     EBX = Virtual address (for current process, [u.uno])
12030                              <1>       ;
12031                              <1>       ; OUTPUT ->
12032                              <1>       ;     EAX = [swpq_count]
12033                              <1>       ;           (after the PTE has been added)
12034                              <1>       ;     EAX = 0 -> Swap queue is full, (1024 entries)
12035                              <1>       ;           the pte could not be added.
```

```
12036                                  <1>       ;
12037                                  <1>       ; Modified Registers -> EAX
12038                                  <1>       ;
12039 000035B3 53                      <1>       push   ebx
12040 000035B4 6681E300F0              <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
12041 000035B9 8A1D[97740000]          <1>       mov    bl, [u.uno] ; current process number
12042 000035BF E853FFFFFF              <1>       call   swap_queue_shift ; drop from the queue if
12043                                  <1>                               ; it is already in the queue
12044                                  <1>              ; Then add it to the tail of the queue
12045 000035C4 0FB705[10850000]        <1>       movzx  eax, word [swpq_count]
12046 000035CB 663D0004                <1>       cmp    ax, 1024
12047 000035CF 7205                    <1>       jb     short atsq_1
12048 000035D1 6629C0                  <1>       sub    ax, ax
12049 000035D4 5B                      <1>       pop    ebx
12050 000035D5 C3                      <1>       retn
12051                                  <1> atsq_1:
12052 000035D6 56                      <1>       push   esi
12053 000035D7 BE00E00800              <1>       mov    esi, swap_queue
12054 000035DC 6621C0                  <1>       and    ax, ax
12055 000035DF 740A                    <1>       jz     short atsq_2
12056 000035E1 66C1E002                <1>       shl    ax, 2  ; convert to offset
12057 000035E5 01C6                    <1>       add    esi, eax
12058 000035E7 66C1E802                <1>       shr    ax, 2
12059                                  <1> atsq_2:
12060 000035EB 6640                    <1>       inc    ax
12061 000035ED 891E                    <1>       mov    [esi], ebx ; Virtual address + [u.uno] combination
12062 000035EF 66A3[10850000]          <1>       mov    [swpq_count], ax
12063 000035F5 5E                      <1>       pop    esi
12064 000035F6 5B                      <1>       pop    ebx
12065 000035F7 C3                      <1>       retn
12066                                  <1>
12067                                  <1> unlink_swap_block:
12068                                  <1>       ; 15/09/2015
12069                                  <1>       ; 30/04/2015
12070                                  <1>       ; 18/04/2015
12071                                  <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12072                                  <1>       ;
12073                                  <1>       ; INPUT ->
12074                                  <1>       ;     EAX = swap disk/file offset address
12075                                  <1>       ;           (bit 1 to bit 31)
12076                                  <1>       ; OUTPUT ->
12077                                  <1>       ;     [swpd_free] is increased
12078                                  <1>       ;     (corresponding SWAP DISK ALLOC. TABLE bit is SET)
12079                                  <1>       ;
12080                                  <1>       ; Modified Registers -> EAX
12081                                  <1>       ;
12082 000035F8 53                      <1>       push   ebx
12083 000035F9 52                      <1>       push   edx
12084                                  <1>       ;
12085 000035FA C1E804                  <1>       shr    eax, SECTOR_SHIFT+1  ;3+1 ; shift sector address to
12086                                  <1>                                   ; 3 bits right
12087                                  <1>                                   ; to get swap block/page number
12088 000035FD 89C2                    <1>       mov    edx, eax
12089                                  <1>       ; 15/09/2015
12090 000035FF C1EA03                  <1>       shr    edx, 3           ; to get offset to S.A.T.
12091                                  <1>                                   ; (1 allocation bit = 1 page)
12092                                  <1>                                   ; (1 allocation bytes = 8 pages)
12093 00003602 80E2FC                  <1>       and    dl, 0FCh         ; clear lower 2 bits
12094                                  <1>                                   ; (to get 32 bit position)
12095                                  <1>       ;
12096 00003605 BB00000D00              <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table address
12097 0000360A 01D3                    <1>       add    ebx, edx
12098 0000360C 83E01F                  <1>       and    eax, 1Fh         ; lower 5 bits only
12099                                  <1>                                   ; (allocation bit position)
12100 0000360F 3B05[1E850000]          <1>       cmp    eax, [swpd_next]     ; is the new free block addr. lower
12101                                  <1>                                   ; than the address in 'swpd_next' ?
12102                                  <1>                                   ; (next/first free block value)
12103 00003615 7305                    <1>       jnb    short uswpbl_1           ; no
12104 00003617 A3[1E850000]            <1>       mov    [swpd_next], eax     ; yes
12105                                  <1> uswpbl_1:
12106 0000361C 0FAB03                  <1>       bts    [ebx], eax       ; unlink/release/deallocate block
12107                                  <1>                                   ; set relevant bit to 1.
12108                                  <1>                                   ; set CF to the previous bit value
12109 0000361F F5                      <1>       cmc                      ; complement carry flag
12110 00003620 7206                    <1>       jc     short uswpbl_2          ; do not increase swfd_free count
12111                                  <1>                                   ; if the block is already deallocated
12112                                  <1>                                   ; before.
12113 00003622 FF05[1A850000]          <1>         inc     dword [swpd_free]
12114                                  <1> uswpbl_2:
12115 00003628 5A                      <1>       pop    edx
12116 00003629 5B                      <1>       pop    ebx
12117 0000362A C3                      <1>       retn
12118                                  <1>
12119                                  <1> link_swap_block:
12120                                  <1>       ; 01/07/2015
12121                                  <1>       ; 18/04/2015
12122                                  <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12123                                  <1>       ;
12124                                  <1>       ; INPUT -> none
12125                                  <1>       ;
12126                                  <1>       ; OUTPUT ->
12127                                  <1>       ;     EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
12128                                  <1>       ;           in sectors (corresponding
12129                                  <1>       ;           SWAP DISK ALLOCATION TABLE bit is RESET)
12130                                  <1>       ;
12131                                  <1>       ;     CF = 1 and EAX = 0
12132                                  <1>       ;               if there is not a free block to be allocated
12133                                  <1>       ;
12134                                  <1>       ; Modified Registers -> none (except EAX)
12135                                  <1>       ;
12136                                  <1>
12137                                  <1>       ;mov   eax, [swpd_free]
12138                                  <1>       ;and   eax, eax
12139                                  <1>       ;jz    short out_of_swpspc
12140                                  <1>       ;
```

```
12141 0000362B 53                <1>       push   ebx
12142 0000362C 51                <1>       push   ecx
12143                            <1>       ;
12144 0000362D BB00000D00        <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table offset
12145 00003632 89D9              <1>       mov    ecx, ebx
12146 00003634 031D[1E850000]    <1>       add    ebx, [swpd_next] ; Free block searching starts from here
12147                            <1>                               ; next_free_swap_block >> 5
12148 0000363A 030D[22850000]    <1>       add    ecx, [swpd_last] ; Free block searching ends here
12149                            <1>                               ; (total_swap_blocks - 1) >> 5
12150                            <1> lswbl_scan:
12151 00003640 39CB              <1>       cmp    ebx, ecx
12152 00003642 770A              <1>       ja     short lswbl_notfound
12153                            <1>       ;
12154 00003644 0FBC03            <1>       bsf    eax, [ebx] ; Scans source operand for first bit set (1).
12155                            <1>                         ; Clears ZF if a bit is found set (1) and
12156                            <1>                         ; loads the destination with an index to
12157                            <1>                         ; first set bit. (0 -> 31)
12158                            <1>                         ; Sets ZF to 1 if no bits are found set.
12159                            <1>       ; 01/07/2015
12160 00003647 751C              <1>       jnz    short lswbl_found ; ZF = 0 -> a free block has been found
12161                            <1>                          ;
12162                            <1>                          ; NOTE:  a Swap Disk Allocation Table bit
12163                            <1>                          ;       with value of 1 means
12164                            <1>                          ;       the corresponding page is free
12165                            <1>                          ;       (Retro UNIX 386 v1 feaure only!)
12166 00003649 83C304            <1>       add    ebx, 4
12167                            <1>                          ; We return back for searching next page block
12168                            <1>                          ; NOTE: [swpd_free] is not ZERO; so,
12169                            <1>                          ;     we always will find at least 1 free block here.
12170 0000364C EBF2              <1>       jmp          short lswbl_scan
12171                            <1>       ;
12172                            <1> lswbl_notfound:
12173 0000364E 81E900000D00      <1>       sub    ecx, swap_alloc_table
12174 00003654 890D[1E850000]    <1>       mov    [swpd_next], ecx ; next/first free page = last page
12175                            <1>                               ; (unlink_swap_block procedure will change it)
12176 0000365A 31C0              <1>       xor    eax, eax
12177 0000365C A3[1A850000]      <1>       mov    [swpd_free], eax
12178 00003661 F9                <1>       stc
12179                            <1> lswbl_ok:
12180 00003662 59                <1>       pop    ecx
12181 00003663 5B                <1>       pop    ebx
12182 00003664 C3                <1>       retn
12183                            <1>       ;
12184                            <1> ;out_of_swpspc:
12185                            <1> ;     stc
12186                            <1> ;     retn
12187                            <1>
12188                            <1> lswbl_found:
12189 00003665 89D9              <1>       mov    ecx, ebx
12190 00003667 81E900000D00      <1>       sub    ecx, swap_alloc_table
12191 0000366D 890D[1E850000]    <1>       mov    [swpd_next], ecx ; Set first free block searching start
12192                            <1>                               ; address/offset (to the next)
12193 00003673 FF0D[1A850000]    <1>         dec     dword [swpd_free] ; 1 block has been allocated (X = X-1)
12194                            <1>       ;
12195 00003679 0FB303            <1>       btr    [ebx], eax     ; The destination bit indexed by the source value
12196                            <1>                               ; is copied into the Carry Flag and then cleared
12197                            <1>                               ; in the destination.
12198                            <1>                               ;
12199                            <1>                               ; Reset the bit which is corresponding to the
12200                            <1>                               ; (just) allocated block.
12201 0000367C C1E105            <1>       shl    ecx, 5         ; (block offset * 32) + block index
12202 0000367F 01C8              <1>       add    eax, ecx       ; = block number
12203 00003681 C1E003            <1>       shl    eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
12204                            <1>                               ; 1 block =  8 sectors
12205                            <1>       ;
12206                            <1>       ; EAX = offset address of swap disk/file sector (beginning of the block)
12207                            <1>       ;
12208                            <1>       ; NOTE: The relevant page table entry will be updated
12209                            <1>       ;       according to this EAX value...
12210                            <1>       ;
12211 00003684 EBDC              <1>       jmp    short lswbl_ok
12212                            <1>
12213                            <1> logical_disk_read:
12214                            <1>       ; 20/07/2015
12215                            <1>       ; 09/03/2015 (temporary code here)
12216                            <1>       ;
12217                            <1>       ; INPUT ->
12218                            <1>       ;      ESI = Logical disk description table address
12219                            <1>       ;      EBX = Memory page (buffer) address (physical!)
12220                            <1>       ;      EAX = Sector adress (offset address, logical sector number)
12221                            <1>       ;      ECX = Sector count
12222                            <1>       ;
12223                            <1>       ;
12224 00003686 C3                <1>       retn
12225                            <1>
12226                            <1> logical_disk_write:
12227                            <1>       ; 20/07/2015
12228                            <1>       ; 09/03/2015 (temporary code here)
12229                            <1>       ;
12230                            <1>       ; INPUT ->
12231                            <1>       ;      ESI = Logical disk description table address
12232                            <1>       ;      EBX = Memory page (buffer) address (physical!)
12233                            <1>       ;      EAX = Sector adress (offset address, logical sector number)
12234                            <1>       ;      ECX = Sector count
12235                            <1>       ;
12236 00003687 C3                <1>       retn
12237                            <1>
12238                            <1> get_physical_addr:
12239                            <1>       ; 18/10/2015
12240                            <1>       ; 29/07/2015
12241                            <1>       ; 20/07/2015
12242                            <1>       ; 04/06/2015
12243                            <1>       ; 20/05/2015
12244                            <1>       ; 28/04/2015
12245                            <1>       ; 18/04/2015
```

```
12246                                  <1>       ; Get physical address
12247                                  <1>       ;     (allocates a new page for user if it is not present)
12248                                  <1>       ;
12249                                  <1>       ; (This subroutine is needed for mapping user's virtual
12250                                  <1>       ; (buffer) address to physical address (of the buffer).)
12251                                  <1>       ; ('sys write', 'sys read' system calls...)
12252                                  <1>       ;
12253                                  <1>       ; INPUT ->
12254                                  <1>       ;     EBX = virtual address
12255                                  <1>       ;     u.pgdir = page directory (physical) address
12256                                  <1>       ;
12257                                  <1>       ; OUTPUT ->
12258                                  <1>       ;     EAX = physical address
12259                                  <1>       ;     EBX = linear address
12260                                  <1>       ;     EDX = physical address of the page frame
12261                                  <1>       ;           (with attribute bits)
12262                                  <1>       ;     ECX = byte count within the page frame
12263                                  <1>       ;
12264                                  <1>       ; Modified Registers -> EAX, EBX, ECX, EDX
12265                                  <1>       ;
12266 00003688 81C300004000           <1>       add   ebx, CORE ; 18/10/2015
12267                                  <1>       ;
12268 0000368E A1[A1740000]           <1>       mov   eax, [u.pgdir]
12269 00003693 E878FAFFFF             <1>       call  get_pte
12270                                  <1>             ; EDX = Page table entry address (if CF=0)
12271                                  <1>               ;       Page directory entry address (if CF=1)
12272                                  <1>             ;       (Bit 0 value is 0 if PT is not present)
12273                                  <1>             ; EAX = Page table entry value (page address)
12274                                  <1>             ;     CF = 1 -> PDE not present or invalid ?
12275 00003698 731C                   <1>       jnc   short gpa_1
12276                                  <1>       ;
12277 0000369A E856F9FFFF             <1>       call  allocate_page
12278 0000369F 725B                   <1>       jc    short gpa_im_err  ; 'insufficient memory' error
12279                                  <1> gpa_0:
12280 000036A1 E8C9F9FFFF             <1>       call  clear_page
12281                                  <1>       ; EAX = Physical (base) address of the allocated (new) page
12282 000036A6 0C07                   <1>       or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
12283                                  <1>                       ; lower 3 bits are used as U/S, R/W, P flags
12284                                  <1>                       ; (user, writable, present page)
12285 000036A8 8902                   <1>       mov   [edx], eax ; Let's put the new page directory entry here !
12286 000036AA A1[A1740000]           <1>       mov   eax, [u.pgdir]
12287 000036AF E85CFAFFFF             <1>       call  get_pte
12288 000036B4 7246                   <1>       jc    short gpa_im_err ; 'insufficient memory' error
12289                                  <1> gpa_1:
12290                                  <1>       ; EAX = PTE value, EDX = PTE address
12291 000036B6 A801                   <1>       test  al, PTE_A_PRESENT
12292 000036B8 751A                   <1>       jnz   short gpa_3
12293 000036BA 09C0                   <1>       or    eax, eax
12294 000036BC 7430                   <1>       jz    short gpa_4  ; Allocate a new page
12295                                  <1>       ; 20/07/2015
12296 000036BE 55                     <1>       push  ebp
12297 000036BF 89DD                   <1>       mov   ebp, ebx ; virtual (linear) address
12298                                  <1>       ; reload swapped page
12299 000036C1 E83C000000             <1>       call  reload_page ; 28/04/2015
12300 000036C6 5D                     <1>       pop   ebp
12301 000036C7 7224                   <1>       jc    short gpa_retn
12302                                  <1> gpa_2:
12303                                  <1>       ; 20/07/2015
12304                                  <1>       ; 20/05/2015
12305                                  <1>       ; add this page to swap queue
12306 000036C9 50                     <1>       push  eax
12307                                  <1>       ; EBX = virtual address
12308 000036CA E8E3FEFFFF             <1>       call  add_to_swap_queue
12309 000036CF 58                     <1>       pop   eax
12310                                  <1>             ; PTE address in EDX
12311                                  <1>             ; virtual address in EBX
12312                                  <1>       ; EAX = memory page address
12313 000036D0 0C07                   <1>       or    al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
12314                                  <1>                               ; present flag, bit 0 = 1
12315                                  <1>                               ; user flag, bit 2 = 1
12316                                  <1>                               ; writable flag, bit 1 = 1
12317 000036D2 8902                   <1>       mov   [edx], eax  ; Update PTE value
12318                                  <1> gpa_3:
12319                                  <1>       ; 18/10/2015
12320 000036D4 89D9                   <1>       mov   ecx, ebx
12321 000036D6 81E1FF0F0000           <1>       and   ecx, PAGE_OFF
12322 000036DC 89C2                   <1>       mov   edx, eax
12323 000036DE 662500F0               <1>       and   ax, PTE_A_CLEAR
12324 000036E2 01C8                   <1>       add   eax, ecx
12325 000036E4 F7D9                   <1>       neg   ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095
12326 000036E6 81C100100000           <1>       add   ecx, PAGE_SIZE
12327 000036EC F8                     <1>       clc
12328                                  <1> gpa_retn:
12329 000036ED C3                     <1>       retn
12330                                  <1> gpa_4:
12331 000036EE E802F9FFFF             <1>       call  allocate_page
12332 000036F3 7207                   <1>       jc    short gpa_im_err ; 'insufficient memory' error
12333 000036F5 E875F9FFFF             <1>       call  clear_page
12334 000036FA EBCD                   <1>       jmp   short gpa_2
12335                                  <1>
12336                                  <1> gpa_im_err:
12337 000036FC B801000000             <1>       mov   eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
12338                                  <1>                               ; Major error = 0 (No protection fault)
12339 00003701 C3                     <1>       retn
12340                                  <1>
12341                                  <1> reload_page:
12342                                  <1>       ; 20/07/2015
12343                                  <1>       ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
12344                                  <1>       ;
12345                                  <1>       ; Reload (Restore) swapped page at memory
12346                                  <1>       ;
12347                                  <1>       ; INPUT ->
12348                                  <1>       ;     EBP = Virtual (linear) memory address
12349                                  <1>       ;     EAX = PTE value (swap disk sector address)
12350                                  <1>       ;     (Swap disk sector address = bit 1 to bit 31 of EAX)
```

```
12351                                   <1>       ; OUTPUT ->
12352                                   <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
12353                                   <1>       ;
12354                                   <1>       ;      CF = 1 and EAX = error code
12355                                   <1>       ;
12356                                   <1>       ; Modified Registers -> none (except EAX)
12357                                   <1>       ;
12358 00003702 D1E8                    <1>       shr    eax, 1   ; Convert PTE value to swap disk address
12359 00003704 53                      <1>       push   ebx      ;
12360 00003705 89C3                    <1>       mov    ebx, eax ; Swap disk (offset) address
12361 00003707 E8E9F8FFFF              <1>       call   allocate_page
12362 0000370C 720C                    <1>       jc     short rlp_im_err
12363 0000370E 93                      <1>       xchg   eax, ebx
12364                                   <1>       ; EBX = Physical memory (page) address
12365                                   <1>       ; EAX = Swap disk (offset) address
12366                                   <1>       ; EBP = Virtual (linear) memory address
12367 0000370F E81AFDFFFF              <1>       call   swap_in
12368 00003714 720B                    <1>       jc     short rlp_swp_err  ; (swap disk/file read error)
12369 00003716 89D8                    <1>       mov    eax, ebx
12370                                   <1> rlp_retn:
12371 00003718 5B                      <1>       pop    ebx
12372 00003719 C3                      <1>       retn
12373                                   <1>
12374                                   <1> rlp_im_err:
12375 0000371A B801000000              <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
12376                                   <1>                              ; Major error = 0 (No protection fault)
12377 0000371F EBF7                    <1>       jmp    short rlp_retn
12378                                   <1>
12379                                   <1> rlp_swp_err:
12380 00003721 B804000000              <1>       mov    eax, SWP_DISK_READ_ERR ; Swap disk read error !
12381 00003726 EBF0                    <1>       jmp    short rlp_retn
12382                                   <1>
12383                                   <1>
12384                                   <1> copy_page_dir:
12385                                   <1>       ; 19/09/2015
12386                                   <1>       ; temporary - 07/09/2015
12387                                   <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
12388                                   <1>       ;
12389                                   <1>       ; INPUT ->
12390                                   <1>       ;      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
12391                                   <1>       ;                  page directory.
12392                                   <1>       ; OUTPUT ->
12393                                   <1>       ;      EAX =  PHYSICAL (real/flat) ADDRESS of the child's
12394                                   <1>       ;             page directory.
12395                                   <1>       ;      (New page directory with new page table entries.)
12396                                   <1>       ;      (New page tables with read only copies of the parent's
12397                                   <1>       ;      pages.)
12398                                   <1>       ;      EAX = 0 -> Error (CF = 1)
12399                                   <1>       ;
12400                                   <1>       ; Modified Registers -> none (except EAX)
12401                                   <1>       ;
12402 00003728 E8C8F8FFFF              <1>       call   allocate_page
12403 0000372D 723E                    <1>       jc     short cpd_err
12404                                   <1>       ;
12405 0000372F 55                      <1>       push   ebp ; 20/07/2015
12406 00003730 56                      <1>       push   esi
12407 00003731 57                      <1>       push   edi
12408 00003732 53                      <1>       push   ebx
12409 00003733 51                      <1>       push   ecx
12410 00003734 8B35[A1740000]          <1>       mov    esi, [u.pgdir]
12411 0000373A 89C7                    <1>       mov    edi, eax
12412 0000373C 50                      <1>       push   eax ; save child's page directory address
12413                                   <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
12414                                   <1>       ; (use same system space for all user page tables)
12415 0000373D A5                      <1>       movsd
12416 0000373E BD00004000              <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
12417 00003743 B9FF030000              <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
12418                                   <1> cpd_0:
12419 00003748 AD                      <1>       lodsd
12420                                   <1>       ;or    eax, eax
12421                                   <1>         ;jnz     short cpd_1
12422 00003749 A801                    <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
12423 0000374B 7508                    <1>       jnz    short cpd_1
12424                                   <1>       ; (virtual address at the end of the page table)
12425 0000374D 81C500004000            <1>       add    ebp, 1024*4096 ; page size * PTE count
12426 00003753 EB0F                    <1>       jmp    short cpd_2
12427                                   <1> cpd_1:
12428 00003755 662500F0                <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
12429 00003759 89C3                    <1>       mov    ebx, eax
12430                                   <1>       ; EBX = Parent's page table address
12431 0000375B E81F000000              <1>       call   copy_page_table
12432 00003760 720C                    <1>       jc     short cpd_p_err
12433                                   <1>       ; EAX = Child's page table address
12434 00003762 0C07                    <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
12435                                   <1>                       ; set bit 0, bit 1 and bit 2 to 1
12436                                   <1>                       ; (present, writable, user)
12437                                   <1> cpd_2:
12438 00003764 AB                      <1>       stosd
12439 00003765 E2E1                    <1>       loop   cpd_0
12440                                   <1>       ;
12441 00003767 58                      <1>       pop    eax  ; restore child's page directory address
12442                                   <1> cpd_3:
12443 00003768 59                      <1>       pop    ecx
12444 00003769 5B                      <1>       pop    ebx
12445 0000376A 5F                      <1>       pop    edi
12446 0000376B 5E                      <1>       pop    esi
12447 0000376C 5D                      <1>       pop    ebp
12448                                   <1> cpd_err:
12449 0000376D C3                      <1>       retn
12450                                   <1> cpd_p_err:
12451                                   <1>       ; release the allocated pages missing (recover free space)
12452 0000376E 58                      <1>       pop    eax  ; the new page directory address (physical)
12453 0000376F 8B1D[A1740000]          <1>       mov    ebx, [u.pgdir] ; parent's page directory address
12454 00003775 E8B4F9FFFF              <1>       call   deallocate_page_dir
12455 0000377A 29C0                    <1>       sub    eax, eax ; 0
```

```
12456 0000377C F9                 <1>       stc
12457 0000377D EBE9               <1>       jmp    short cpd_3
12458                             <1>
12459                             <1> copy_page_table:
12460                             <1>       ; 19/09/2015
12461                             <1>       ; temporary - 07/09/2015
12462                             <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
12463                             <1>       ;
12464                             <1>       ; INPUT ->
12465                             <1>       ;     EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
12466                             <1>       ;     EBP = page table entry index (from 'copy_page_dir')
12467                             <1>       ; OUTPUT ->
12468                             <1>       ;     EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
12469                             <1>       ;     EBP = (recent) page table index (for 'add_to_swap_queue')
12470                             <1>       ;     CF = 1 -> error
12471                             <1>       ;
12472                             <1>       ; Modified Registers -> EBP (except EAX)
12473                             <1>       ;
12474 0000377F E871F8FFFF          <1>       call   allocate_page
12475 00003784 725A               <1>       jc     short cpt_err
12476                             <1>       ;
12477 00003786 50                 <1>       push   eax ; *
12478                             <1>       ;push  ebx
12479 00003787 56                 <1>       push   esi
12480 00003788 57                 <1>       push   edi
12481 00003789 52                 <1>       push   edx
12482 0000378A 51                 <1>       push   ecx
12483                             <1>       ;
12484 0000378B 89DE               <1>       mov    esi, ebx
12485 0000378D 89C7               <1>       mov    edi, eax
12486 0000378F 89C2               <1>       mov    edx, eax
12487 00003791 81C200100000       <1>       add    edx, PAGE_SIZE
12488                             <1> cpt_0:
12489 00003797 AD                 <1>       lodsd
12490 00003798 A801               <1>       test   al, PTE_A_PRESENT ;  bit 0 =  1
12491 0000379A 750B               <1>       jnz    short cpt_1
12492 0000379C 21C0               <1>       and    eax, eax
12493 0000379E 7430               <1>       jz     short cpt_2
12494                             <1>       ; ebp = virtual (linear) address of the memory page
12495 000037A0 E85DFFFFFF         <1>       call   reload_page ; 28/04/2015
12496 000037A5 7234               <1>       jc     short cpt_p_err
12497                             <1> cpt_1:
12498 000037A7 662500F0           <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
12499 000037AB 89C1               <1>       mov    ecx, eax
12500                             <1>       ; Allocate a new page for the child process
12501 000037AD E843F8FFFF         <1>       call   allocate_page
12502 000037B2 7227               <1>       jc     short cpt_p_err
12503 000037B4 57                 <1>       push   edi
12504 000037B5 56                 <1>       push   esi
12505 000037B6 89CE               <1>       mov    esi, ecx
12506 000037B8 89C7               <1>       mov    edi, eax
12507 000037BA B900040000         <1>       mov    ecx, PAGE_SIZE/4
12508 000037BF F3A5               <1>       rep    movsd  ; copy page (4096 bytes)
12509 000037C1 5E                 <1>       pop    esi
12510 000037C2 5F                 <1>       pop    edi
12511                             <1>       ;
12512 000037C3 53                 <1>       push   ebx
12513 000037C4 50                 <1>       push   eax
12514 000037C5 89EB               <1>       mov    ebx, ebp
12515                             <1>       ; ebx = virtual address of the memory page
12516 000037C7 E8E6FDFFFF         <1>       call   add_to_swap_queue
12517 000037CC 58                 <1>       pop    eax
12518 000037CD 5B                 <1>       pop    ebx
12519                             <1>       ;
12520                             <1>       ;or    ax, PTE_A_USER+PTE_A_PRESENT
12521 000037CE 0C07               <1>       or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
12522                             <1> cpt_2:
12523 000037D0 AB                 <1>       stosd  ; EDI points to child's PTE
12524                             <1>       ;
12525 000037D1 81C500100000       <1>       add    ebp, 4096 ; 20/07/2015 (next page)
12526                             <1>       ;
12527 000037D7 39D7               <1>       cmp    edi, edx
12528 000037D9 72BC               <1>       jb     short cpt_0
12529                             <1> cpt_p_err:
12530 000037DB 59                 <1>       pop    ecx
12531 000037DC 5A                 <1>       pop    edx
12532 000037DD 5F                 <1>       pop    edi
12533 000037DE 5E                 <1>       pop    esi
12534                             <1>       ;pop   ebx
12535 000037DF 58                 <1>       pop    eax ; *
12536                             <1> cpt_err:
12537 000037E0 C3                 <1>       retn
12538                             <1>
12539                             <1>
12540                             <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
12541                             <1>
12542                             <1> ;; Data:
12543                             <1>
12544                             <1> ; 09/03/2015
12545                             <1> ;swpq_count: dw 0 ; count of pages on the swap que
12546                             <1> ;swp_drv:    dd 0 ; logical drive description table address of the swap drive/disk
12547                             <1> ;swpd_size:  dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

12548                             <1> ;swpd_free:  dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
12549                             <1> ;swpd_next:  dd 0 ; next free page block
12550                             <1> ;swpd_last:  dd 0 ; last swap page block
12551                                 %include 'sysdefs.inc' ; 09/03/2015
12552                             <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
12553                             <1> ; Last Modification: 04/02/2016
12554                             <1> ;
12555                             <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////
12556                             <1> ; (Modified from
12557                             <1> ;     Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
12558                             <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
12559                             <1> ;     UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)
```

```
12560                                  <1> ; ----------------------------------------------------------------------------
12561                                  <1> ;
12562                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
12563                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
12564                                  <1> ; <Bell Laboratories (17/3/1972)>
12565                                  <1> ; <Preliminary Release of UNIX Implementation Document>
12566                                  <1> ;
12567                                  <1> ; ****************************************************************************
12568                                  <1>
12569                                  <1> nproc       equ    16  ; number of processes
12570                                  <1> nfiles      equ    50
12571                                  <1> ntty  equ     8   ; 8+1 -> 8 (10/05/2013)
12572                                  <1> nbuf  equ    6   ; number of buffers (04/02/2016)
12573                                  <1>
12574                                  <1> ;csgmnt     equ    2000h  ; 26/05/2013 (segment of process 1)
12575                                  <1> ;core equ    0          ; 19/04/2013
12576                                  <1> ;ecore      equ    32768 - 64  ; 04/06/2013 (24/05/2013)
12577                                  <1>       ; (if total size of argument list and arguments is 128 bytes)
12578                                  <1>       ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes
12579                                  <1>       ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)
12580                                  <1>       ; initial value of user's stack pointer = 32768-64-128-2 = 32574
12581                                  <1>       ;     (sp=32768-args_space-2 at the beginning of execution)
12582                                  <1>       ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)
12583                                  <1>       ; 'u' structure offset (for the '/core' dump file) = 32704
12584                                  <1>       ; '/core' dump file size = 32768 bytes
12585                                  <1>
12586                                  <1> ; 08/03/2014
12587                                  <1> ;sdsegmnt equ      6C0h  ; 256*16 bytes (swap data segment size for 16 processes)

12588                                  <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
12589                                  <1> ;;sdsegmnt equ     740h  ; swap data segment (for user structures and registers)
12590                                  <1>
12591                                  <1> ; 30/08/2013
12592                                  <1> time_count equ 4 ; 10 --> 4 01/02/2014
12593                                  <1>
12594                                  <1> ; 05/02/2014
12595                                  <1> ; process status
12596                                  <1> ;SFREE      equ 0
12597                                  <1> ;SRUN equ 1
12598                                  <1> ;SWAIT      equ 2
12599                                  <1> ;SZOMB      equ 3
12600                                  <1> ;SSLEEP     equ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)
12601                                  <1>
12602                                  <1> ; 09/03/2015
12603                                  <1> userdata equ 80000h ; user structure data address for current user ; temporary
12604                                  <1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
12605                                  <1> swap_alloc_table equ 0D0000h  ;  swap allocation table address ; temporary
12606                                  <1>
12607                                  <1> ; 17/09/2015
12608                                  <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
12609                                  <1>
12610                                  <1> ; 21/09/2015 (36)
12611                                  <1> ; 01/07/2015 (35)
12612                                  <1> ; 14/07/2013 (0-34)
12613                                  <1> ; UNIX v1 system calls
12614                                  <1> _rele       equ 0
12615                                  <1> _exit       equ 1
12616                                  <1> _fork       equ 2
12617                                  <1> _read       equ 3
12618                                  <1> _write      equ 4
12619                                  <1> _open equ 5
12620                                  <1> _close      equ 6
12621                                  <1> _wait       equ 7
12622                                  <1> _creat      equ 8
12623                                  <1> _link       equ 9
12624                                  <1> _unlink     equ 10
12625                                  <1> _exec equ 11
12626                                  <1> _chdir      equ 12
12627                                  <1> _time       equ 13
12628                                  <1> _mkdir      equ 14
12629                                  <1> _chmod      equ 15
12630                                  <1> _chown      equ 16
12631                                  <1> _break      equ 17
12632                                  <1> _stat equ 18
12633                                  <1> _seek equ 19
12634                                  <1> _tell       equ 20
12635                                  <1> _mount      equ 21
12636                                  <1> _umount     equ 22
12637                                  <1> _setuid     equ 23
12638                                  <1> _getuid     equ 24
12639                                  <1> _stime      equ 25
12640                                  <1> _quit equ 26
12641                                  <1> _intr equ 27
12642                                  <1> _fstat      equ 28
12643                                  <1> _emt  equ 29
12644                                  <1> _mdate      equ 30
12645                                  <1> _stty       equ 31
12646                                  <1> _gtty equ 32
12647                                  <1> _ilgins     equ 33
12648                                  <1> _sleep      equ 34 ; Retro UNIX 8086 v1 feature only !
12649                                  <1> _msg  equ 35 ; Retro UNIX 386 v1 feature only !
12650                                  <1> _geterr     equ 36 ; Retro UNIX 386 v1 feature only !
12651                                  <1>
12652                                  <1> %macro sys 1-4
12653                                  <1>     ; 13/04/2015
12654                                  <1>     ; Retro UNIX 386 v1 system call.
12655                                  <1>     mov eax, %1
12656                                  <1>     %if %0 >= 2
12657                                  <1>         mov ebx, %2
12658                                  <1>         %if %0 >= 3
12659                                  <1>             mov ecx, %3
12660                                  <1>             %if %0 = 4
12661                                  <1>                mov edx, %4
12662                                  <1>             %endif
12663                                  <1>         %endif
```

```
12664                                  <1>     %endif
12665                                  <1>     int 30h
12666                                  <1> %endmacro
12667                                  <1>
12668                                  <1> ; 13/05/2015 - ERROR CODES
12669                                  <1> ERR_FILE_NOT_OPEN  equ 10 ; 'file not open !' error
12670                                  <1> ERR_FILE_ACCESS    equ 11 ; 'permission denied !' error
12671                                  <1> ; 14/05/2015
12672                                  <1> ERR_DIR_ACCESS     equ 11 ; 'permission denied !' error
12673                                  <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
12674                                  <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
12675                                  <1> ERR_DIR_EXISTS     equ 14 ; 'directory already exists !' error
12676                                  <1> ; 16/05/2015
12677                                  <1> ERR_DRV_NOT_RDY    equ 15 ; 'drive not ready !' error
12678                                  <1> ; 18/05/2015
12679                                  <1> ERR_DEV_NOT_RDY    equ 15 ; 'device not ready !' error
12680                                  <1> ERR_DEV_ACCESS     equ 11 ; 'permission denied !' error
12681                                  <1> ERR_DEV_NOT_OPEN   equ 10 ; 'device not open !' error
12682                                  <1> ; 07/06/2015
12683                                  <1> ERR_FILE_EOF    equ 16 ; 'end of file !' error
12684                                  <1> ERR_DEV_VOL_SIZE   equ 16 ; 'out of volume' error
12685                                  <1> ; 09/06/2015
12686                                  <1> ERR_DRV_READ    equ 17 ; 'disk read error !'
12687                                  <1> ERR_DRV_WRITE         equ 18 ; 'disk write error !'
12688                                  <1> ; 16/06/2015
12689                                  <1> ERR_NOT_DIR     equ 19 ; 'not a (valid) directory !' error
12690                                  <1> ERR_FILE_SIZE         equ 20 ; 'file size error !'
12691                                  <1> ; 22/06/2015
12692                                  <1> ERR_NOT_SUPERUSER  equ 11 ; 'permission denied !' error
12693                                  <1> ERR_NOT_OWNER      equ 11 ; 'permission denied !' error
12694                                  <1> ERR_NOT_FILE       equ 11 ; 'permission denied !' error
12695                                  <1> ; 23/06/2015
12696                                  <1> ERR_FILE_EXISTS    equ 14 ; 'file already exists !' error
12697                                  <1> ERR_DRV_NOT_SAME   equ 21 ; 'not same drive !' error
12698                                  <1> ERR_DIR_NOT_FOUND  equ 12 ; 'directory not found !' error
12699                                  <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
12700                                  <1> ; 27/06/2015
12701                                  <1> ERR_INV_PARAMETER  equ 23 ; 'invalid parameter !' error
12702                                  <1> ERR_INV_DEV_NAME   equ 24 ; 'invalid device name !' error
12703                                  <1> ; 29/06/2015
12704                                  <1> ERR_TIME_OUT    equ 25 ; 'time out !' error
12705                                  <1> ERR_DEV_NOT_RESP   equ 25 ; 'device not responding !' error
12706                                  <1>
12707                                  <1> ; 26/08/2015
12708                                  <1> ; 24/07/2015
12709                                  <1> ; 24/06/2015
12710                                  <1> MAX_ARG_LEN     equ 256 ; max. length of sys exec arguments
12711                                  <1> ; 01/07/2015
12712                                  <1> MAX_MSG_LEN     equ 255 ; max. msg length for 'sysmsg'
12713                                  <1> ;
12714                                      %include 'u0.s'        ; 15/03/2015
12715                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC
12716                                  <1> ; Last Modification: 20/11/2015
12717                                  <1> ; ----------------------------------------------------------------------------
12718                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
12719                                  <1> ; (v0.1 - Beginning: 11/07/2012)
12720                                  <1> ;
12721                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
12722                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
12723                                  <1> ; <Bell Laboratories (17/3/1972)>
12724                                  <1> ; <Preliminary Release of UNIX Implementation Document>
12725                                  <1> ;
12726                                  <1> ; Retro UNIX 8086 v1 - U0.ASM (28/07/2014) //// UNIX v1 -> u0.s
12727                                  <1> ;
12728                                  <1> ; ****************************************************************************
12729                                  <1>
12730                                  <1> sys_init:
12731                                  <1>       ; 18/10/2015
12732                                  <1>       ; 28/08/2015
12733                                  <1>       ; 24/08/2015
12734                                  <1>       ; 14/08/2015
12735                                  <1>       ; 24/07/2015
12736                                  <1>       ; 02/07/2015
12737                                  <1>       ; 01/07/2015
12738                                  <1>       ; 23/06/2015
12739                                  <1>       ; 15/04/2015
12740                                  <1>       ; 13/04/2015
12741                                  <1>       ; 11/03/2015 (Retro UNIX 386 v1 - Beginning)
12742                                  <1>       ; 28/07/2014 (Retro UNIX 8086 v1)
12743                                  <1>       ;
12744                                  <1>       ;call  ldrv_init ; Logical drive description tables initialization
12745                                  <1>       ;
12746                                  <1>       ; 14/02/2014
12747                                  <1>       ; 14/07/2013
12748 000037E1 66B82900               <1>       mov    ax, 41
12749 000037E5 66A3[38740000]         <1>       mov    [rootdir], ax
12750 000037EB 66A3[4C740000]         <1>       mov    [u.cdir], ax
12751 000037F1 2401                   <1>       and    al, 1 ; 15/04/2015
12752 000037F3 A2[97740000]           <1>       mov    [u.uno], al
12753 000037F8 66A3[36740000]         <1>       mov    [mpid], ax
12754 000037FE 66A3[36710000]         <1>       mov    [p.pid], ax
12755 00003804 A2[C6710000]           <1>       mov    [p.stat], al ; SRUN, 05/02/2014
12756                                  <1>       ;
12757 00003809 B004                   <1>       mov    al, time_count ; 30/08/2013
12758 0000380B A2[8A740000]           <1>       mov    [u.quant], al ; 14/07/2013
12759                                  <1>       ; 02/07/2015
12760 00003810 A1[68700000]           <1>       mov    eax, [k_page_dir]
12761                                  <1>       ;sub   eax, eax
12762 00003815 A3[A1740000]           <1>       mov    [u.pgdir], eax ; reset
12763                                  <1>       ; 18/10/2015
12764                                  <1>       ;mov   [u.ppgdir], eax ; 0
12765                                  <1>         ;
12766 0000381A E856030000             <1>       call   epoch
12767 0000381F A3[94820000]           <1>       mov    [s.time], eax ; 13/03/2015
12768                                  <1>       ; 17/07/2013
```

```
12769 00003824 E8B0060000          <1>       call  bf_init ; buffer initialization
12770                              <1>       ; 23/06/2015
12771 00003829 E8C7F7FFFF          <1>       call  allocate_page
12772                              <1>       ;;jc  error
12773 0000382E 0F829C000000        <1>         jc      panic   ; jc short panic (01/07/2015)
12774 00003834 A3[98740000]        <1>       mov   [u.upage], eax ; user structure page
12775 00003839 A3[D6710000]        <1>       mov   [p.upage], eax
12776                              <1>       ;
12777 0000383E E82CF8FFFF          <1>       call  clear_page
12778                              <1>       ;
12779                              <1>       ; 14/08/2015
12780 00003843 FA                  <1>       cli
12781                              <1>       ; 14/03/2015
12782                              <1>       ; 17/01/2014
12783 00003844 E8DE010000          <1>       call  sp_init ; serial port initialization
12784                              <1>       ; 14/08/2015
12785 00003849 FB                  <1>       sti
12786                              <1>       ;
12787                              <1>       ; 30/06/2015
12788                              <1>       ;mov  esi, kernel_init_ok_msg
12789                              <1>       ;call print_msg
12790                              <1>       ;
12791 0000384A 30DB                <1>       xor   bl, bl ; video page 0
12792                              <1> vp_clr_nxt:  ; clear video pages (reset cursor positions)
12793 0000384C E8052E0000          <1>       call  vp_clr  ; 17/07/2013
12794 00003851 FEC3                <1>       inc   bl
12795 00003853 80FB08              <1>       cmp   bl, 8
12796 00003856 72F4                <1>       jb    short vp_clr_nxt
12797                              <1>       ;
12798                              <1>       ; 24/07/2015
12799                              <1>       ;push    KDATA
12800                              <1>         ;push    esp
12801                              <1>       ;mov  [tss.esp0], esp
12802                              <1>         ;mov     word [tss.ss0], KDATA
12803                              <1>       ;
12804                              <1>       ; 24/08/2015
12805                              <1>       ;; temporary (01/07/2015)
12806 00003858 C605[8A740000]04    <1>       mov   byte [u.quant], time_count ; 4
12807                              <1>                           ; it is not needed here !
12808                              <1>       ;;inc  byte [u.kcall] ; 'the caller is kernel' sign
12809 0000385F FE0D[3F740000]      <1>       dec   byte [sysflg] ; FFh = ready for system call
12810                              <1>                           ; 0 = executing a system call
12811                              <1>       ;;sys  _msg, kernel_init_ok_msg, 255, 0
12812                              <1>       ;
12813                              <1>       ;;; 06/08/2015
12814                              <1>       ;;;call      getch ; wait for a key stroke
12815                              <1>       ;;mov  ecx, 0FFFFFFFh
12816                              <1> ;;sys_init_msg_wait:
12817                              <1> ;;    push   ecx
12818                              <1> ;;    mov    al, 1
12819                              <1> ;;    mov    ah, [ptty] ; active (current) video page
12820                              <1> ;;    call   getc_n
12821                              <1> ;;    pop    ecx
12822                              <1> ;;    jnz    short sys_init_msg_ok
12823                              <1> ;;    loop   sys_init_msg_wait
12824                              <1>       ;
12825                              <1> ;;sys_init_msg_ok:
12826                              <1>       ; 28/08/2015 (initial settings for the 1st 'rswap')
12827 00003865 6A10                <1>       push  KDATA ; ss
12828 00003867 54                  <1>       push  esp
12829 00003868 9C                  <1>       pushfd
12830 00003869 6A08                <1>       push  KCODE ; cs
12831 0000386B 68[9F380000]        <1>       push  init_exec ; eip
12832 00003870 8925[40740000]      <1>       mov   [u.sp], esp
12833 00003876 1E                  <1>       push  ds
12834 00003877 06                  <1>       push  es
12835 00003878 0FA0                <1>       push  fs
12836 0000387A 0FA8                <1>       push  gs
12837 0000387C 60                  <1>       pushad
12838 0000387D 8925[44740000]      <1>       mov   [u.usp], esp
12839 00003883 E8561B0000          <1>       call  wswap ; save current user (u) structure, user registers
12840                              <1>                   ; and interrupt return components (for IRET)
12841 00003888 61                  <1>       popad
12842 00003889 6658                <1>       pop   ax ; gs
12843 0000388B 6658                <1>       pop   ax ; fs
12844 0000388D 6658                <1>       pop   ax ; es
12845 0000388F 6658                <1>       pop   ax ; ds
12846 00003891 58                  <1>       pop   eax ; eip (init_exec)
12847 00003892 6658                <1>       pop   ax ; cs (KCODE)
12848 00003894 58                  <1>       pop   eax ; E-FLAGS
12849 00003895 58                  <1>       pop   eax ; esp
12850 00003896 6658                <1>       pop   ax ; ss (KDATA)
12851                              <1>       ;
12852 00003898 31C0                <1>       xor   eax, eax ; 0
12853 0000389A A3[A5740000]        <1>       mov   [u.ppgdir], eax ; reset (to zero) for '/etc/init'
12854                              <1>       ;
12855                              <1>       ; 02/07/2015
12856                              <1>       ; [u.pgdir ] = [k_page_dir]
12857                              <1>       ; [u.ppgdir] = 0 (page dir of the parent process)
12858                              <1>       ;     (The caller is os kernel sign for 'sysexec')
12859                              <1> init_exec:
12860                              <1>       ; 13/03/2013
12861                              <1>       ; 24/07/2013
12862 0000389F BB[C6380000]        <1>       mov   ebx, init_file
12863 000038A4 B9[BE380000]        <1>       mov   ecx, init_argp
12864                              <1>       ; EBX contains 'etc/init' asciiz file name address
12865                              <1>       ; ECX contains address of argument list pointer
12866                              <1>       ;
12867                              <1>       ;dec   byte [sysflg] ; FFh = ready for system call
12868                              <1>                           ; 0 = executing a system call
12869                              <1>       sys   _exec  ; execute file
12870                              <2>
12871                              <2>
12872 000038A9 B80B000000         <2>  mov eax, %1
12873                              <2>  %if %0 >= 2
```

```
12874                                  <2>  mov ebx, %2
12875                                  <2>  %if %0 >= 3
12876                                  <2>  mov ecx, %3
12877                                  <2>  %if %0 = 4
12878                                  <2>  mov edx, %4
12879                                  <2>  %endif
12880                                  <2>  %endif
12881                                  <2>  %endif
12882 000038AE CD30                   <2>  int 30h
12883 000038B0 731E                   <1>       jnc    short panic
12884                                  <1>       ;
12885 000038B2 BE[786D0000]           <1>       mov    esi, etc_init_err_msg
12886 000038B7 E837000000             <1>       call   print_msg
12887 000038BC EB1C                   <1>       jmp    short key_to_reboot
12888                                  <1>
12889                                  <1> ;align 4
12890                                  <1> init_argp:
12891 000038BE [C6380000]00000000     <1>       dd     init_file, 0  ; 23/06/2015 (dw -> dd)
12892                                  <1> init_file:
12893                                  <1>       ; 24/08/2015
12894 000038C6 2F6574632F696E6974-    <1>       db     '/etc/init', 0
12895 000038CF 00                     <1>
12896                                  <1> panic:
12897                                  <1>       ; 13/03/2015 (Retro UNIX 386 v1)
12898                                  <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
12899 000038D0 BE[5D6D0000]           <1>       mov    esi, panic_msg
12900 000038D5 E819000000             <1>       call   print_msg
12901                                  <1> key_to_reboot:
12902                                  <1>       ; 15/11/2015
12903 000038DA E8B52B0000             <1>       call   getch
12904                                  <1>              ; wait for a character from the current tty
12905                                  <1>       ;
12906 000038DF B00A                   <1>       mov    al, 0Ah
12907 000038E1 8A1D[96700000]         <1>       mov    bl, [ptty] ; [active_page]
12908 000038E7 B407                   <1>       mov    ah, 07h ; Black background,
12909                                  <1>                      ; light gray forecolor
12910 000038E9 E8ECDBFFFF             <1>       call   write_tty
12911 000038EE E96CD8FFFF             <1>       jmp    cpu_reset
12912                                  <1>
12913                                  <1> print_msg:
12914                                  <1>       ; 01/07/2015
12915                                  <1>       ; 13/03/2015 (Retro UNIX 386 v1)
12916                                  <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
12917                                  <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
12918                                  <1>       ;
12919                                  <1>       ;
12920 000038F3 AC                     <1>       lodsb
12921                                  <1> pmsg1:
12922 000038F4 56                     <1>       push   esi
12923 000038F5 0FB61D[96700000]       <1>       movzx  ebx, byte [ptty]
12924 000038FC B407                   <1>       mov    ah, 07h ; Black background, light gray forecolor
12925 000038FE E8D7DBFFFF             <1>       call   write_tty
12926 00003903 5E                     <1>       pop    esi
12927 00003904 AC                     <1>       lodsb
12928 00003905 20C0                   <1>       and    al, al
12929 00003907 75EB                   <1>       jnz    short pmsg1
12930 00003909 C3                     <1>       retn
12931                                  <1>
12932                                  <1> ctrlbrk:
12933                                  <1>       ; 12/11/2015
12934                                  <1>       ; 13/03/2015 (Retro UNIX 386 v1)
12935                                  <1>       ; 06/12/2013 (Retro UNIX 8086 v1)
12936                                  <1>       ;
12937                                  <1>       ; INT 1Bh (control+break) handler
12938                                  <1>       ;
12939                                  <1>              ; Retro Unix 8086 v1 feature only!
12940                                  <1>              ;
12941 0000390A 66833D[8C740000]00     <1>       cmp    word [u.intr], 0
12942 00003912 7645                   <1>       jna    short cbrk4
12943                                  <1> cbrk0:
12944                                  <1>       ; 12/11/2015
12945                                  <1>       ; 06/12/2013
12946 00003914 66833D[8E740000]00     <1>       cmp    word [u.quit], 0
12947 0000391C 743B                   <1>       jz     short cbrk4
12948                                  <1>       ;
12949                                  <1>       ; 20/09/2013
12950 0000391E 6650                   <1>       push   ax
12951 00003920 A0[96700000]           <1>       mov    al, [ptty]
12952                                  <1>       ;
12953                                  <1>       ; 12/11/2015
12954                                  <1>       ;
12955                                  <1>       ; ctrl+break (EOT, CTRL+D) from serial port
12956                                  <1>       ; or ctrl+break from console (pseudo) tty
12957                                  <1>       ; (!redirection!)
12958                                  <1>       ;
12959 00003925 3C08                   <1>       cmp    al, 8 ; serial port tty nums > 7
12960 00003927 7211                   <1>         jb      short cbrk1 ; console (pseudo) tty
12961                                  <1>       ;
12962                                  <1>       ; Serial port interrupt handler sets [ptty]
12963                                  <1>       ; to the port's tty number (as temporary).
12964                                  <1>       ;
12965                                  <1>       ; If active process is using a stdin or
12966                                  <1>       ; stdout redirection (by the shell),
12967                                  <1>         ; console tty keyboard must be available
12968                                  <1>       ; to terminate running process,
12969                                  <1>       ; in order to prevent a deadlock.
12970                                  <1>       ;
12971 00003929 52                     <1>       push   edx
12972 0000392A 0FB615[97740000]       <1>       movzx  edx, byte [u.uno]
12973 00003931 3A82[95710000]         <1>       cmp     al, [edx+p.ttyc-1] ; console tty (rw)
12974 00003937 5A                     <1>       pop    edx
12975 00003938 7412                   <1>       je     short cbrk2
12976                                  <1> cbrk1:
12977 0000393A FEC0                   <1>       inc    al  ; [u.ttyp] : 1 based tty number
12978                                  <1>       ; 06/12/2013
```

```
12979 0000393C 3A05[78740000]      <1>       cmp    al, [u.ttyp] ; recent open tty (r)
12980 00003942 7408                <1>       je     short cbrk2
12981 00003944 3A05[79740000]      <1>         cmp     al, [u.ttyp+1] ; recent open tty (w)
12982 0000394A 750B                <1>       jne    short cbrk3
12983                              <1> cbrk2:
12984                              <1>       ;; 06/12/2013
12985                              <1>       ;mov   ax, [u.quit]
12986                              <1>       ;and   ax, ax
12987                              <1>       ;jz    short cbrk3
12988                              <1>       ;
12989 0000394C 6631C0              <1>       xor    ax, ax ; 0
12990 0000394F 6648                <1>       dec    ax
12991                              <1>       ; 0FFFFh = 'ctrl+brk' keystroke
12992 00003951 66A3[8E740000]      <1>       mov    [u.quit], ax
12993                              <1> cbrk3:
12994 00003957 6658                <1>       pop    ax
12995                              <1> cbrk4:
12996 00003959 C3                  <1>       retn
12997                              <1>
12998                              <1> com2_int:
12999                              <1>       ; 07/11/2015
13000                              <1>       ; 24/10/2015
13001                              <1>       ; 23/10/2015
13002                              <1>       ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
13003                              <1>       ; 28/07/2014 (Retro UNIX 8086 v1)
13004                              <1>       ; < serial port 2 interrupt handler >
13005                              <1>       ;
13006 0000395A 890424              <1>       mov    [esp], eax ; overwrite call return address
13007                              <1>       ;push  eax
13008 0000395D 66B80900            <1>       mov    ax, 9
13009 00003961 EB07                <1>       jmp    short comm_int
13010                              <1> com1_int:
13011                              <1>       ; 07/11/2015
13012                              <1>       ; 24/10/2015
13013 00003963 890424              <1>       mov    [esp], eax ; overwrite call return address
13014                              <1>       ; 23/10/2015
13015                              <1>       ;push  eax
13016 00003966 66B80800            <1>       mov    ax, 8
13017                              <1> comm_int:
13018                              <1>       ; 20/11/2015
13019                              <1>       ; 18/11/2015
13020                              <1>       ; 17/11/2015
13021                              <1>       ; 16/11/2015
13022                              <1>       ; 09/11/2015
13023                              <1>       ; 08/11/2015
13024                              <1>       ; 07/11/2015
13025                              <1>       ; 06/11/2015 (serial4.asm, 'serial')
13026                              <1>       ; 01/11/2015
13027                              <1>       ; 26/10/2015
13028                              <1>       ; 23/10/2015
13029 0000396A 53                  <1>       push   ebx
13030 0000396B 56                  <1>       push   esi
13031 0000396C 57                  <1>       push   edi
13032 0000396D 1E                  <1>       push   ds
13033 0000396E 06                  <1>       push   es
13034                              <1>       ; 18/11/2015
13035 0000396F 0F20DB              <1>       mov    ebx, cr3
13036 00003972 53                  <1>       push   ebx ; ****
13037                              <1>       ;
13038 00003973 51                  <1>       push   ecx ; ***
13039 00003974 52                  <1>       push   edx ; **
13040                              <1>       ;
13041 00003975 BB10000000          <1>       mov    ebx, KDATA
13042 0000397A 8EDB                <1>       mov    ds, bx
13043 0000397C 8EC3                <1>       mov    es, bx
13044                              <1>       ;
13045 0000397E 8B0D[68700000]      <1>       mov    ecx, [k_page_dir]
13046 00003984 0F22D9              <1>       mov    cr3, ecx
13047                              <1>       ; 20/11/2015
13048                              <1>       ; Interrupt identification register
13049 00003987 66BAFA02            <1>       mov    dx, 2FAh ; COM2
13050                              <1>       ;
13051 0000398B 3C08                <1>       cmp    al, 8
13052 0000398D 7702                <1>       ja     short com_i0
13053                              <1>       ;
13054                              <1>       ; 20/11/2015
13055                              <1>       ; 17/11/2015
13056                              <1>       ; 16/11/2015
13057                              <1>       ; 15/11/2015
13058                              <1>       ; 24/10/2015
13059                              <1>       ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
13060                              <1>       ; 28/07/2014 (Retro UNIX 8086 v1)
13061                              <1>       ; < serial port 1 interrupt handler >
13062                              <1>       ;
13063 0000398F FEC6                <1>       inc    dh ; 3FAh ; COM1 Interrupt id. register
13064                              <1> com_i0:
13065                              <1>       ;push  eax ; *
13066                              <1>       ; 07/11/2015
13067 00003991 A2[D6700000]        <1>       mov    byte [ccomport], al
13068                              <1>       ; 09/11/2015
13069 00003996 0FB7D8              <1>       movzx  ebx, ax ; 8 or 9
13070                              <1>       ; 17/11/2015
13071                              <1>       ; reset request for response status
13072 00003999 88A3[CC700000]      <1>       mov    [ebx+req_resp-8], ah ; 0
13073                              <1>       ;
13074                              <1>       ; 20/11/2015
13075 0000399F EC                  <1>       in     al, dx       ; read interrupt id. register
13076 000039A0 EB00                <1>       JMP    $+2          ; I/O DELAY
13077 000039A2 2404                <1>       and    al, 4        ; received data available?
13078 000039A4 7470                <1>       jz     short com_eoi; (transmit. holding reg. empty)
13079                              <1>       ;
13080                              <1>       ; 20/11/2015
13081 000039A6 80EA02              <1>       sub    dl, 3FAh-3F8h; data register (3F8h, 2F8h)
13082 000039A9 EC                  <1>       in     al, dx       ; read character
13083                              <1>       ;JMP   $+2          ; I/O DELAY
```

```
13084                                  <1>       ; 08/11/2015
13085                                  <1>       ; 07/11/2015
13086 000039AA 89DE                    <1>       mov    esi, ebx
13087 000039AC 89DF                    <1>       mov    edi, ebx
13088 000039AE 81C6[D0700000]          <1>       add    esi, rchar - 8 ; points to last received char
13089 000039B4 81C7[D2700000]          <1>       add    edi, schar - 8 ; points to last sent char
13090 000039BA 8806                    <1>       mov    [esi], al ; received char (current char)
13091                                  <1>       ; query
13092 000039BC 20C0                    <1>       and    al, al
13093 000039BE 7527                    <1>       jnz    short com_i2
13094                                  <1>       ; response
13095                                  <1>       ; 17/11/2015
13096                                  <1>       ; set request for response status
13097 000039C0 FE83[CC700000]          <1>         inc     byte [ebx+req_resp-8] ; 1
13098                                  <1>       ;
13099 000039C6 6683C205                <1>       add    dx, 3FDh-3F8h; (3FDh, 2FDh)
13100 000039CA EC                      <1>       in     al, dx       ; read line status register
13101 000039CB EB00                    <1>       JMP    $+2          ; I/O DELAY
13102 000039CD 2420                    <1>       and    al, 20h             ; transmitter holding reg. empty?
13103 000039CF 7445                    <1>       jz     short com_eoi       ; no
13104 000039D1 B0FF                    <1>       mov    al, 0FFh     ; response
13105 000039D3 6683EA05                <1>       sub    dx, 3FDh-3F8h       ; data port (3F8h, 2F8h)
13106 000039D7 EE                      <1>       out    dx, al       ; send on serial port
13107                                  <1>       ; 17/11/2015
13108 000039D8 803F00                  <1>       cmp    byte [edi], 0   ; query ? (schar)
13109 000039DB 7502                    <1>       jne    short com_i1    ; no
13110 000039DD 8807                    <1>       mov    [edi], al    ; 0FFh (responded)
13111                                  <1> com_i1:
13112                                  <1>       ; 17/11/2015
13113                                  <1>       ; reset request for response status (again)
13114 000039DF FE8B[CC700000]          <1>         dec     byte [ebx+req_resp-8] ; 0
13115 000039E5 EB2F                    <1>       jmp    short com_eoi
13116                                  <1> com_i2:
13117                                  <1>       ; 08/11/2015
13118 000039E7 3CFF                    <1>       cmp    al, 0FFh     ; (response ?)
13119 000039E9 7417                    <1>       je     short com_i3 ; (check for response signal)
13120                                  <1>       ; 07/11/2015
13121 000039EB 3C04                    <1>       cmp    al, 04h      ; EOT
13122 000039ED 751C                    <1>       jne    short com_i4
13123                                  <1>       ; EOT = 04h (End of Transmit) - 'CTRL + D'
13124                                  <1>       ;(an EOT char is supposed as a ctrl+brk from the terminal)
13125                                  <1>       ; 08/11/2015
13126                                  <1>             ; ptty -> tty 0 to 7 (pseudo screens)
13127 000039EF 861D[96700000]          <1>       xchg   bl, [ptty]  ; tty number (8 or 9)
13128 000039F5 E810FFFFFF              <1>       call   ctrlbrk
13129 000039FA 861D[96700000]          <1>       xchg   [ptty], bl ; (restore ptty value and BL value)
13130                                  <1>       ;mov   al, 04h ; EOT
13131                                  <1>       ; 08/11/2015
13132 00003A00 EB09                    <1>       jmp    short com_i4
13133                                  <1> com_i3:
13134                                  <1>       ; 08/11/2015
13135                                  <1>       ; If 0FFh has been received just after a query
13136                                  <1>       ; (schar, ZERO), it is a response signal.
13137                                  <1>       ; 17/11/2015
13138 00003A02 803F00                  <1>         cmp     byte [edi], 0 ; query ? (schar)
13139 00003A05 7704                    <1>       ja     short com_i4 ; no
13140                                  <1>       ; reset query status (schar)
13141 00003A07 8807                    <1>       mov    [edi], al ; 0FFh
13142 00003A09 FEC0                    <1>       inc    al ; 0
13143                                  <1> com_i4:
13144                                  <1>       ; 27/07/2014
13145                                  <1>       ; 09/07/2014
13146 00003A0B D0E3                    <1>       shl    bl, 1
13147 00003A0D 81C3[98700000]          <1>       add    ebx, ttychr
13148                                  <1>       ; 23/07/2014 (always overwrite)
13149                                  <1>       ;;cmp  word [ebx], 0
13150                                  <1>       ;;ja   short com_eoi
13151                                  <1>       ;
13152 00003A13 668903                  <1>       mov    [ebx], ax   ; Save ascii code
13153                                  <1>                          ; scan code = 0
13154                                  <1> com_eoi:
13155                                  <1>       ;mov   al, 20h
13156                                  <1>       ;out   20h, al         ; end of interrupt
13157                                  <1>       ;
13158                                  <1>       ; 07/11/2015
13159                                  <1>             ;pop   eax ; *
13160 00003A16 A0[D6700000]            <1>       mov    al, byte [ccomport] ; current COM port
13161                                  <1>        ; al = tty number (8 or 9)
13162 00003A1B E8991A0000              <1>         call wakeup
13163                                  <1> com_iret:
13164                                  <1>       ; 23/10/2015
13165 00003A20 5A                      <1>       pop    edx ; **
13166 00003A21 59                      <1>       pop    ecx ; ***
13167                                  <1>       ; 18/11/2015
13168                                  <1>       ;pop   eax ; ****
13169                                  <1>       ;mov   cr3, eax
13170                                  <1>       ;jmp   iiret
13171 00003A22 E94AD0FFFF              <1>       jmp    iiretp
13172                                  <1>
13173                                  <1> ;iiretp: ; 01/09/2015
13174                                  <1> ;     ; 28/08/2015
13175                                  <1> ;     pop    eax ; (*) page directory
13176                                  <1> ;     mov    cr3, eax
13177                                  <1> ;iiret:
13178                                  <1> ;     ; 22/08/2014
13179                                  <1> ;     mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
13180                                  <1> ;     out    20h, al      ; 8259 PORT
13181                                  <1> ;     ;
13182                                  <1> ;     pop    es
13183                                  <1> ;     pop    ds
13184                                  <1> ;     pop    edi
13185                                  <1> ;     pop    esi
13186                                  <1> ;     pop    ebx ; 29/08/2014
13187                                  <1> ;     pop    eax
13188                                  <1> ;     iretd
```

```
13189                                  <1>
13190                                  <1> sp_init:
13191                                  <1>         ; 07/11/2015
13192                                  <1>         ; 29/10/2015
13193                                  <1>         ; 26/10/2015
13194                                  <1>         ; 23/10/2015
13195                                  <1>         ; 29/06/2015
13196                                  <1>         ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)
13197                                  <1>         ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
13198                                  <1>         ; Initialization of Serial Port Communication Parameters
13199                                  <1>         ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
13200                                  <1>         ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
13201                                  <1>         ;
13202                                  <1>         ; ((Modified registers: EAX, ECX, EDX, EBX))
13203                                  <1>         ;
13204                                  <1>         ; INPUT:  (29/06/2015)
13205                                  <1>         ;     AL = 0 for COM1
13206                                  <1>         ;          1 for COM2
13207                                  <1>         ;     AH = Communication parameters
13208                                  <1>         ;
13209                                  <1>         ;  (*) Communication parameters (except BAUD RATE):
13210                                  <1>         ;     Bit    4      3      2      1      0
13211                                  <1>         ;            -PARITY--   STOP BIT  -WORD LENGTH-
13212                                  <1>         ;  this one -->     00 = none    0 = 1 bit  11 = 8 bits
13213                                  <1>         ;            01 = odd     1 = 2 bits   10 = 7 bits
13214                                  <1>         ;            11 = even
13215                                  <1>         ;  Baud rate setting bits: (29/06/2015)
13216                                  <1>         ;            Retro UNIX 386 v1 feature only !
13217                                  <1>         ;     Bit    7    6    5  | Baud rate
13218                                  <1>         ;            ------------------------
13219                                  <1>         ;     value  0    0    0  | Default (Divisor = 1)
13220                                  <1>         ;            0    0    1  | 9600 (12)
13221                                  <1>         ;            0    1    0  | 19200 (6)
13222                                  <1>         ;            0    1   1  | 38400 (3)
13223                                  <1>         ;            1    0   0  | 14400 (8)
13224                                  <1>         ;            1    0   1  | 28800 (4)
13225                                  <1>         ;            1    1    0  | 57600 (2)
13226                                  <1>         ;            1    1    1  | 115200 (1)
13227                                  <1>
13228                                  <1>         ; References:
13229                                  <1>         ; (1) IBM PC-XT Model 286 BIOS Source Code
13230                                  <1>         ;     RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
13231                                  <1>         ; (2) Award BIOS 1999 - ATORGS.ASM
13232                                  <1>         ; (3) http://wiki.osdev.org/Serial_Ports
13233                                  <1>         ;
13234                                  <1>         ; Set communication parameters for COM1 (= 03h)
13235                                  <1>         ;
13236 00003A27 BB[D2700000]           <1>         mov    ebx, com1p          ; COM1 parameters
13237 00003A2C 66BAF803               <1>         mov    dx, 3F8h            ; COM1
13238                                  <1>          ; 29/10/2015
13239 00003A30 66B90103               <1>         mov    cx, 301h  ; divisor = 1 (115200 baud)
13240 00003A34 E86F000000             <1>         call   sp_i3  ; call A4
13241 00003A39 A880                   <1>         test   al, 80h
13242 00003A3B 7410                   <1>         jz     short sp_i0 ; OK..
13243                                  <1>                ; Error !
13244                                  <1>         ;mov   dx, 3F8h
13245 00003A3D 80EA05                 <1>         sub    dl, 5 ; 3FDh -> 3F8h
13246 00003A40 66B90E03               <1>         mov    cx, 30Eh  ; divisor = 12 (9600 baud)
13247 00003A44 E85F000000             <1>         call   sp_i3  ; call A4
13248 00003A49 A880                   <1>         test   al, 80h
13249 00003A4B 7508                   <1>         jnz    short sp_i1
13250                                  <1> sp_i0:
13251                                  <1>           ; (Note: Serial port interrupts will be disabled here...)
13252                                  <1>           ; (INT 14h initialization code disables interrupts.)
13253                                  <1>         ;
13254 00003A4D C603E3                 <1>         mov    byte [ebx], 0E3h ; 11100011b
13255 00003A50 E8DC000000             <1>         call   sp_i5 ; 29/06/2015
13256                                  <1> sp_i1:
13257 00003A55 43                     <1>         inc    ebx
13258 00003A56 66BAF802               <1>         mov    dx, 2F8h            ; COM2
13259                                  <1>          ; 29/10/2015
13260 00003A5A 66B90103               <1>         mov    cx, 301h  ; divisor = 1 (115200 baud)
13261 00003A5E E845000000             <1>         call   sp_i3  ; call A4
13262 00003A63 A880                   <1>         test   al, 80h
13263 00003A65 7410                   <1>         jz     short sp_i2 ; OK..
13264                                  <1>                ; Error !
13265                                  <1>         ;mov   dx, 2F8h
13266 00003A67 80EA05                 <1>         sub    dl, 5 ; 2FDh -> 2F8h
13267 00003A6A 66B90E03               <1>         mov    cx, 30Eh  ; divisor = 12 (9600 baud)
13268 00003A6E E835000000             <1>         call   sp_i3  ; call A4
13269 00003A73 A880                   <1>         test   al, 80h
13270 00003A75 7530                   <1>         jnz    short sp_i7
13271                                  <1> sp_i2:
13272 00003A77 C603E3                 <1>         mov    byte [ebx], 0E3h ; 11100011b
13273                                  <1> sp_i6:
13274                                  <1>         ;; COM2 - enabling IRQ 3
13275                                  <1>         ; 07/11/2015
13276                                  <1>         ; 26/10/2015
13277 00003A7A 9C                     <1>         pushf
13278 00003A7B FA                     <1>         cli
13279                                  <1>         ;
13280 00003A7C 66BAFC02               <1>         mov    dx, 2FCh            ; modem control register
13281 00003A80 EC                     <1>         in     al, dx                     ; read register
13282 00003A81 EB00                   <1>         JMP    $+2                 ; I/O DELAY
13283 00003A83 0C08                   <1>         or     al, 8               ; enable bit 3 (OUT2)
13284 00003A85 EE                     <1>         out    dx, al              ; write back to register
13285 00003A86 EB00                   <1>         JMP    $+2                 ; I/O DELAY
13286 00003A88 66BAF902               <1>         mov    dx, 2F9h            ; interrupt enable register
13287 00003A8C EC                     <1>         in     al, dx              ; read register
13288 00003A8D EB00                   <1>         JMP    $+2                 ; I/O DELAY
13289                                  <1>         ;or    al, 1               ; receiver data interrupt enable and
13290 00003A8F 0C03                   <1>         or     al, 3               ; transmitter empty interrupt enable
13291 00003A91 EE                     <1>         out    dx, al                     ; write back to register
13292 00003A92 EB00                   <1>         JMP    $+2                 ; I/O DELAY
13293 00003A94 E421                   <1>         in     al, 21h             ; read interrupt mask register
```

```
13294 00003A96 EB00              <1>       JMP    $+2                ; I/O DELAY
13295 00003A98 24F7              <1>       and    al, 0F7h           ; enable IRQ 3 (COM2)
13296 00003A9A E621              <1>       out    21h, al            ; write back to register
13297                            <1>       ;
13298                            <1>       ; 23/10/2015
13299 00003A9C B8[5A390000]      <1>       mov    eax, com2_int
13300 00003AA1 A3[393F0000]      <1>       mov    [com2_irq3], eax
13301                            <1>       ; 26/10/2015
13302 00003AA6 9D                <1>       popf
13303                            <1> sp_i7:
13304 00003AA7 C3                <1>       retn
13305                            <1>
13306                            <1> sp_i3:
13307                            <1> ;A4:         ;----- INITIALIZE THE COMMUNICATIONS PORT
13308                            <1>       ; 28/10/2015
13309 00003AA8 FEC2              <1>       inc    dl     ; 3F9h (2F9h); 3F9h, COM1 Interrupt enable register
13310 00003AAA B000              <1>       mov    al, 0
13311 00003AAC EE                <1>       out    dx, al             ; disable serial port interrupt
13312 00003AAD EB00              <1>       JMP    $+2                ; I/O DELAY
13313 00003AAF 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); COM1 Line control register (3FBh)
13314 00003AB2 B080              <1>       mov    al, 80h
13315 00003AB4 EE                <1>       out    dx, al             ; SET DLAB=1 ; divisor latch access bit
13316                            <1>       ;----- SET BAUD RATE DIVISOR
13317                            <1>       ; 26/10/2015
13318 00003AB5 80EA03            <1>       sub    dl, 3   ; 3F8h (2F8h)     ; register for least significant byte
13319                            <1>                                  ; of the divisor value
13320 00003AB8 88C8              <1>       mov    al, cl ; 1
13321 00003ABA EE                <1>       out    dx, al             ; 1 = 115200 baud (Retro UNIX 386 v1)
13322                            <1>                                  ; 2 = 57600 baud
13323                            <1>                                  ; 3 = 38400 baud
13324                            <1>                                  ; 6 = 19200 baud
13325                            <1>                                  ; 12 = 9600 baud (Retro UNIX 8086 v1)
13326 00003ABB EB00              <1>       JMP    $+2                ; I/O DELAY
13327 00003ABD 28C0              <1>       sub    al, al
13328 00003ABF FEC2              <1>       inc    dl      ; 3F9h (2F9h)     ; register for most significant byte
13329                            <1>                                  ; of the divisor value
13330 00003AC1 EE                <1>       out    dx, al ; 0
13331 00003AC2 EB00              <1>       JMP    $+2                ; I/O DELAY
13332                            <1>       ;
13333 00003AC4 88E8              <1>       mov    al, ch ; 3         ; 8 data bits, 1 stop bit, no parity
13334                            <1>       ;and   al, 1Fh ; Bits 0,1,2,3,4
13335 00003AC6 80C202            <1>       add    dl, 2  ; 3FBh (2FBh); Line control register
13336 00003AC9 EE                <1>       out    dx, al
13337 00003ACA EB00              <1>       JMP    $+2                ; I/O DELAY
13338                            <1>       ; 29/10/2015
13339 00003ACC FECA              <1>       dec    dl     ; 3FAh (2FAh); FIFO Control register (16550/16750)
13340 00003ACE 30C0              <1>       xor    al, al             ; 0
13341 00003AD0 EE                <1>       out    dx, al             ; Disable FIFOs (reset to 8250 mode)
13342 00003AD1 EB00              <1>       JMP    $+2
13343                            <1> sp_i4:
13344                            <1> ;A18: ;----- COMM PORT STATUS ROUTINE
13345                            <1>       ; 29/06/2015 (line status after modem status)
13346 00003AD3 80C204            <1>       add    dl, 4  ; 3FEh (2FEh); Modem status register
13347                            <1> sp_i4s:
13348 00003AD6 EC                <1>       in     al, dx             ; GET MODEM CONTROL STATUS
13349 00003AD7 EB00              <1>       JMP    $+2                ; I/O DELAY
13350 00003AD9 88C4              <1>       mov    ah, al             ; PUT IN (AH) FOR RETURN
13351 00003ADB FECA              <1>       dec    dl     ; 3FDh (2FDh); POINT TO LINE STATUS REGISTER
13352                            <1>                                  ; dx = 3FDh for COM1, 2FDh for COM2
13353 00003ADD EC                <1>       in     al, dx             ; GET LINE CONTROL STATUS
13354                            <1>       ; AL = Line status, AH = Modem status
13355 00003ADE C3                <1>       retn
13356                            <1>
13357                            <1> sp_status:
13358                            <1>       ; 29/06/2015
13359                            <1>       ; 27/06/2015 (Retro UNIX 386 v1)
13360                            <1>       ; Get serial port status
13361 00003ADF 66BAFE03          <1>       mov    dx, 3FEh           ; Modem status register (COM1)
13362 00003AE3 28C6              <1>       sub    dh, al             ; dh = 2 for COM2 (al = 1)
13363                            <1>                                  ; dx = 2FEh for COM2
13364 00003AE5 EBEF              <1>       jmp    short sp_i4s
13365                            <1>
13366                            <1> sp_setp: ; Set serial port communication parameters
13367                            <1>       ; 07/11/2015
13368                            <1>       ; 29/10/2015
13369                            <1>       ; 29/06/2015
13370                            <1>       ; Retro UNIX 386 v1 feature only !
13371                            <1>       ;
13372                            <1>       ; INPUT:
13373                            <1>       ;     AL = 0 for COM1
13374                            <1>       ;          1 for COM2
13375                            <1>       ;     AH = Communication parameters (*)
13376                            <1>       ; OUTPUT:
13377                            <1>       ;     CL = Line status
13378                            <1>       ;     CH = Modem status
13379                            <1>       ;   If cf = 1 -> Error code in [u.error]
13380                            <1>       ;             'invalid parameter !'
13381                            <1>       ;                   or
13382                            <1>       ;             'device not ready !' error
13383                            <1>       ;
13384                            <1>       ;  (*) Communication parameters (except BAUD RATE):
13385                            <1>       ;     Bit    4      3      2      1      0
13386                            <1>       ;            -PARITY--   STOP BIT  -WORD LENGTH-
13387                            <1>       ;  this one -->     00 = none    0 = 1 bit  11 = 8 bits
13388                            <1>       ;            01 = odd     1 = 2 bits   10 = 7 bits
13389                            <1>       ;            11 = even
13390                            <1>       ;  Baud rate setting bits: (29/06/2015)
13391                            <1>       ;            Retro UNIX 386 v1 feature only !
13392                            <1>       ;     Bit    7    6    5  | Baud rate
13393                            <1>       ;            ------------------------
13394                            <1>       ;     value  0    0    0  | Default (Divisor = 1)
13395                            <1>       ;            0    0    1  | 9600 (12)
13396                            <1>       ;            0    1    0  | 19200 (6)
13397                            <1>       ;            0    1   1  | 38400 (3)
13398                            <1>       ;            1    0   0  | 14400 (8)
```

```
13399                                  <1>       ;            1    0   1  | 28800 (4)
13400                                  <1>       ;            1    1    0  | 57600 (2)
13401                                  <1>       ;            1    1    1  | 115200 (1)
13402                                  <1>       ;
13403                                  <1>       ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
13404                                  <1>       ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
13405                                  <1>       ;
13406                                  <1>       ; ((Modified registers: EAX, ECX, EDX, EBX))
13407                                  <1>       ;
13408 00003AE7 66BAF803                <1>       mov    dx, 3F8h
13409 00003AEB BB[D2700000]            <1>       mov    ebx, com1p ; COM1 control byte offset
13410 00003AF0 3C01                    <1>       cmp    al, 1
13411 00003AF2 776B                    <1>       ja     short sp_invp_err
13412 00003AF4 7203                    <1>       jb     short sp_setp1 ;  COM1 (AL = 0)
13413 00003AF6 FECE                    <1>       dec    dh ; 2F8h
13414 00003AF8 43                      <1>       inc    ebx ; COM2 control byte offset
13415                                  <1> sp_setp1:
13416                                  <1>       ; 29/10/2015
13417 00003AF9 8823                    <1>       mov    [ebx], ah
13418 00003AFB 0FB6CC                  <1>       movzx  ecx, ah
13419 00003AFE C0E905                  <1>       shr    cl, 5 ; -> baud rate index
13420 00003B01 80E41F                  <1>       and    ah, 1Fh ; communication parameters except baud rate
13421 00003B04 8A81[6E3B0000]          <1>       mov    al, [ecx+b_div_tbl]
13422 00003B0A 6689C1                  <1>       mov    cx, ax
13423 00003B0D E896FFFFFF              <1>       call   sp_i3
13424 00003B12 6689C1                  <1>       mov    cx, ax ; CL = Line status, CH = Modem status
13425 00003B15 A880                    <1>       test   al, 80h
13426 00003B17 740F                    <1>       jz     short sp_setp2
13427 00003B19 C603E3                  <1>         mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
13428                                  <1> stp_dnr_err:
13429 00003B1C C705[9D740000]0F00-     <1>       mov    dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
13430 00003B24 0000                    <1>
13431                                  <1>       ; CL = Line status, CH = Modem status
13432 00003B26 F9                      <1>       stc
13433 00003B27 C3                      <1>       retn
13434                                  <1> sp_setp2:
13435 00003B28 80FE02                  <1>       cmp    dh, 2 ; COM2 (2F?h)
13436 00003B2B 0F8649FFFFFF            <1>         jna     sp_i6
13437                                  <1>                   ; COM1 (3F?h)
13438                                  <1> sp_i5:
13439                                  <1>       ; 07/11/2015
13440                                  <1>       ; 26/10/2015
13441                                  <1>       ; 29/06/2015
13442                                  <1>       ;
13443                                  <1>       ;; COM1 - enabling IRQ 4
13444 00003B31 9C                      <1>       pushf
13445 00003B32 FA                      <1>       cli
13446 00003B33 66BAFC03                <1>       mov    dx, 3FCh           ; modem control register
13447 00003B37 EC                      <1>       in     al, dx                   ; read register
13448 00003B38 EB00                    <1>       JMP    $+2                ; I/O DELAY
13449 00003B3A 0C08                    <1>       or     al, 8              ; enable bit 3 (OUT2)
13450 00003B3C EE                      <1>       out    dx, al             ; write back to register
13451 00003B3D EB00                    <1>       JMP    $+2                ; I/O DELAY
13452 00003B3F 66BAF903                <1>       mov    dx, 3F9h           ; interrupt enable register
13453 00003B43 EC                      <1>       in     al, dx             ; read register
13454 00003B44 EB00                    <1>       JMP    $+2                ; I/O DELAY
13455                                  <1>       ;or    al, 1              ; receiver data interrupt enable and
13456 00003B46 0C03                    <1>       or     al, 3              ; transmitter empty interrupt enable
13457 00003B48 EE                      <1>       out    dx, al                   ; write back to register
13458 00003B49 EB00                    <1>       JMP    $+2                ; I/O DELAY
13459 00003B4B E421                    <1>       in     al, 21h            ; read interrupt mask register
13460 00003B4D EB00                    <1>       JMP    $+2                ; I/O DELAY
13461 00003B4F 24EF                    <1>       and    al, 0EFh           ; enable IRQ 4 (COM1)
13462 00003B51 E621                    <1>       out    21h, al            ; write back to register
13463                                  <1>       ;
13464                                  <1>       ; 23/10/2015
13465 00003B53 B8[63390000]            <1>       mov    eax, com1_int
13466 00003B58 A3[353F0000]            <1>       mov    [com1_irq4], eax
13467                                  <1>       ; 26/10/2015
13468 00003B5D 9D                      <1>       popf
13469 00003B5E C3                      <1>       retn
13470                                  <1>
13471                                  <1> sp_invp_err:
13472 00003B5F C705[9D740000]1700-     <1>       mov    dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
13473 00003B67 0000                    <1>
13474 00003B69 31C9                    <1>       xor    ecx, ecx
13475 00003B6B 49                      <1>       dec    ecx ; 0FFFFh
13476 00003B6C F9                      <1>       stc
13477 00003B6D C3                      <1>       retn
13478                                  <1>
13479                                  <1> ; 29/10/2015
13480                                  <1> b_div_tbl: ; Baud rate divisor table (115200/divisor)
13481 00003B6E 010C0603080401          <1>       db 1, 12, 6, 3, 8, 4, 1
13482                                  <1>
13483                                  <1> ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)
13484                                  <1> epoch:
13485                                  <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
13486                                  <1>       ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)
13487                                  <1>       ; 'epoch' procedure prototype:
13488                                  <1>       ;                 UNIXCOPY.ASM, 10/03/2013
13489                                  <1>       ; 14/11/2012
13490                                  <1>       ; unixboot.asm (boot file configuration)
13491                                  <1>       ; version of "epoch" procedure in "unixproc.asm"
13492                                  <1>       ; 21/7/2012
13493                                  <1>       ; 15/7/2012
13494                                  <1>       ; 14/7/2012
13495                                  <1>       ; Erdogan Tan - RETRO UNIX v0.1
13496                                  <1>       ; compute current date and time as UNIX Epoch/Time
13497                                  <1>       ; UNIX Epoch: seconds since 1/1/1970 00:00:00
13498                                  <1>       ;
13499                                  <1>         ;  ((Modified registers: EAX, EDX, ECX, EBX))
13500                                  <1>       ;
13501 00003B75 E81D010000              <1>       call   get_rtc_time       ; Return Current Time
13502 00003B7A 86E9                    <1>         xchg       ch,cl
13503 00003B7C 66890D[D26D0000]        <1>         mov  [hour], cx
```

```
13504 00003B83 86F2              <1>         xchg       dh,dl
13505 00003B85 668915[D66D0000]  <1>         mov  [second], dx
13506                            <1>       ;
13507 00003B8C E837010000        <1>         call       get_rtc_date       ; Return Current Date
13508 00003B91 86E9              <1>         xchg       ch,cl
13509 00003B93 66890D[CC6D0000]  <1>         mov  [year], cx
13510 00003B9A 86F2              <1>         xchg       dh,dl
13511 00003B9C 668915[CE6D0000]  <1>         mov  [month], dx
13512                            <1>       ;
13513 00003BA3 66B93030          <1>       mov    cx, 3030h
13514                            <1>       ;
13515 00003BA7 A0[D26D0000]      <1>       mov    al, [hour] ; Hour
13516                            <1>              ; AL <= BCD number)
13517 00003BAC D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13518                            <1>                                 ; AH = AL / 10h
13519                            <1>                                 ; AL = AL MOD 10h
13520 00003BAE D50A              <1>         aad  ; AX= AH*10+AL
13521 00003BB0 A2[D26D0000]      <1>       mov    [hour], al
13522 00003BB5 A0[D36D0000]      <1>       mov    al, [hour+1] ; Minute
13523                            <1>              ; AL <= BCD number)
13524 00003BBA D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13525                            <1>                                 ; AH = AL / 10h
13526                            <1>                                 ; AL = AL MOD 10h
13527 00003BBC D50A              <1>         aad  ; AX= AH*10+AL
13528 00003BBE A2[D46D0000]      <1>       mov    [minute], al
13529 00003BC3 A0[D66D0000]      <1>       mov    al, [second] ; Second
13530                            <1>              ; AL <= BCD number)
13531 00003BC8 D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13532                            <1>                                 ; AH = AL / 10h
13533                            <1>                                 ; AL = AL MOD 10h
13534 00003BCA D50A              <1>         aad  ; AX= AH*10+AL
13535 00003BCC A2[D66D0000]      <1>       mov    [second], al
13536 00003BD1 66A1[CC6D0000]    <1>       mov    ax, [year] ; Year (century)
13537 00003BD7 6650              <1>         push       ax
13538                            <1>              ; AL <= BCD number)
13539 00003BD9 D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13540                            <1>                                 ; AH = AL / 10h
13541                            <1>                                 ; AL = AL MOD 10h
13542 00003BDB D50A              <1>         aad  ; AX= AH*10+AL
13543 00003BDD B464              <1>       mov    ah, 100
13544 00003BDF F6E4              <1>       mul    ah
13545 00003BE1 66A3[CC6D0000]    <1>       mov    [year], ax
13546 00003BE7 6658              <1>       pop    ax
13547 00003BE9 88E0              <1>       mov    al, ah
13548                            <1>              ; AL <= BCD number)
13549 00003BEB D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13550                            <1>                                 ; AH = AL / 10h
13551                            <1>                                 ; AL = AL MOD 10h
13552 00003BED D50A              <1>         aad  ; AX= AH*10+AL
13553 00003BEF 660105[CC6D0000]  <1>       add    [year], ax
13554 00003BF6 A0[CE6D0000]      <1>       mov    al, [month] ; Month
13555                            <1>              ; AL <= BCD number)
13556 00003BFB D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13557                            <1>                                 ; AH = AL / 10h
13558                            <1>                                 ; AL = AL MOD 10h
13559 00003BFD D50A              <1>         aad  ; AX= AH*10+AL
13560 00003BFF A2[CE6D0000]      <1>       mov    [month], al
13561 00003C04 A0[CF6D0000]      <1>         mov     al, [month+1]          ; Day
13562                            <1>              ; AL <= BCD number)
13563 00003C09 D410              <1>         db   0D4h,10h           ; Undocumented inst. AAM
13564                            <1>                                 ; AH = AL / 10h
13565                            <1>                                 ; AL = AL MOD 10h
13566 00003C0B D50A              <1>         aad  ; AX= AH*10+AL
13567 00003C0D A2[D06D0000]      <1>         mov     [day], al
13568                            <1>
13569                            <1> convert_to_epoch:
13570                            <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)
13571                            <1>       ; 09/04/2013 (retro UNIX 8086 v1)
13572                            <1>       ;
13573                            <1>       ; ((Modified registers: EAX, EDX, EBX))
13574                            <1>       ;
13575                            <1>       ; Derived from DALLAS Semiconductor
13576                            <1>       ; Application Note 31 (DS1602/DS1603)
13577                            <1>       ; 6 May 1998
13578 00003C12 29C0              <1>       sub    eax, eax
13579 00003C14 66A1[CC6D0000]    <1>       mov    ax, [year]
13580 00003C1A 662DB207          <1>       sub    ax, 1970
13581 00003C1E BA6D010000        <1>       mov    edx, 365
13582 00003C23 F7E2              <1>       mul    edx
13583 00003C25 31DB              <1>       xor    ebx, ebx
13584 00003C27 8A1D[CE6D0000]    <1>       mov    bl, [month]
13585 00003C2D FECB              <1>       dec    bl
13586 00003C2F D0E3              <1>       shl    bl, 1
13587                            <1>       ;sub   edx, edx
13588 00003C31 668B93[D86D0000]  <1>       mov    dx, [EBX+DMonth]
13589 00003C38 8A1D[D06D0000]    <1>         mov     bl, [day]
13590 00003C3E FECB              <1>       dec    bl
13591 00003C40 01D0              <1>       add    eax, edx
13592 00003C42 01D8              <1>       add    eax, ebx
13593                            <1>                     ; EAX = days since 1/1/1970
13594 00003C44 668B15[CC6D0000]  <1>       mov    dx, [year]
13595 00003C4B 6681EAB107        <1>       sub    dx, 1969
13596 00003C50 66D1EA            <1>       shr    dx, 1
13597 00003C53 66D1EA            <1>       shr    dx, 1
13598                            <1>              ; (year-1969)/4
13599 00003C56 01D0              <1>       add    eax, edx
13600                            <1>                     ; + leap days since 1/1/1970
13601 00003C58 803D[CE6D0000]02  <1>       cmp    byte [month], 2     ; if past february
13602 00003C5F 7610              <1>       jna    short cte1
13603 00003C61 668B15[CC6D0000]  <1>       mov    dx, [year]
13604 00003C68 6683E203          <1>       and    dx, 3 ; year mod 4
13605 00003C6C 7503              <1>       jnz    short cte1
13606                            <1>                     ; and if leap year
13607 00003C6E 83C001            <1>       add    eax, 1       ; add this year's leap day (february 29)
13608                            <1> cte1:                    ; compute seconds since 1/1/1970
```

```
13609 00003C71 BA18000000          <1>       mov   edx, 24
13610 00003C76 F7E2                <1>       mul   edx
13611 00003C78 8A15[D26D0000]      <1>       mov   dl, [hour]
13612 00003C7E 01D0                <1>       add   eax, edx
13613                              <1>             ; EAX = hours since 1/1/1970 00:00:00
13614                              <1>       ;mov  ebx, 60
13615 00003C80 B33C                <1>       mov   bl, 60
13616 00003C82 F7E3                <1>       mul   ebx
13617 00003C84 8A15[D46D0000]      <1>       mov   dl, [minute]
13618 00003C8A 01D0                <1>       add   eax, edx
13619                              <1>             ; EAX = minutes since 1/1/1970 00:00:00
13620                              <1>       ;mov  ebx, 60
13621 00003C8C F7E3                <1>       mul   ebx
13622 00003C8E 8A15[D66D0000]      <1>       mov   dl, [second]
13623 00003C94 01D0                <1>       add   eax, edx
13624                              <1>             ; EAX -> seconds since 1/1/1970 00:00:00
13625 00003C96 C3                  <1>       retn
13626                              <1>
13627                              <1> get_rtc_time:
13628                              <1>       ; 15/03/2015
13629                              <1>       ; Derived from IBM PC-XT Model 286 BIOS Source Code
13630                              <1>       ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13631                              <1>       ; INT 1Ah
13632                              <1>       ; (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,    :
13633                              <1>       ;       (CH) = HOURS IN BCD (00-23)                 :
13634                              <1>       ;       (CL) = MINUTES IN BCD (00-59)               :
13635                              <1>       ;       (DH) = SECONDS IN BCD (00-59)               :
13636                              <1>       ;       (DL) = DAYLIGHT SAVINGS ENABLE (00-01).           :
13637                              <1>       ;
13638                              <1> RTC_20:                         ; GET RTC TIME
13639 00003C97 FA                  <1>       cli
13640 00003C98 E86FCFFFFF          <1>       CALL  UPD_IPR             ; CHECK FOR UPDATE IN PROCESS
13641 00003C9D 7227                <1>       JC    short RTC_29        ; EXIT IF ERROR (CY= 1)
13642                              <1>
13643 00003C9F B000                <1>       MOV   AL,CMOS_SECONDS     ; SET ADDRESS OF SECONDS
13644 00003CA1 E84ECFFFFF          <1>       CALL  CMOS_READ           ; GET SECONDS
13645 00003CA6 88C6                <1>       MOV   DH,AL               ; SAVE
13646 00003CA8 B00B                <1>       MOV   AL,CMOS_REG_B       ; ADDRESS ALARM REGISTER
13647 00003CAA E845CFFFFF          <1>       CALL  CMOS_READ           ; READ CURRENT VALUE OF DSE BIT
13648 00003CAF 2401                <1>       AND   AL,00000001B        ; MASK FOR VALID DSE BIT
13649 00003CB1 88C2                <1>       MOV   DL,AL               ; SET [DL] TO ZERO FOR NO DSE BIT
13650 00003CB3 B002                <1>       MOV   AL,CMOS_MINUTES     ; SET ADDRESS OF MINUTES
13651 00003CB5 E83ACFFFFF          <1>       CALL  CMOS_READ           ; GET MINUTES
13652 00003CBA 88C1                <1>       MOV   CL,AL               ; SAVE
13653 00003CBC B004                <1>       MOV   AL,CMOS_HOURS       ; SET ADDRESS OF HOURS
13654 00003CBE E831CFFFFF          <1>       CALL  CMOS_READ           ; GET HOURS
13655 00003CC3 88C5                <1>       MOV   CH,AL               ; SAVE
13656 00003CC5 F8                  <1>       CLC                       ; SET CY= 0
13657                              <1> RTC_29:
13658 00003CC6 FB                  <1>       sti
13659 00003CC7 C3                  <1>       RETn                      ; RETURN WITH RESULT IN CARRY FLAG
13660                              <1>
13661                              <1> get_rtc_date:
13662                              <1>       ; 15/03/2015
13663                              <1>       ; Derived from IBM PC-XT Model 286 BIOS Source Code
13664                              <1>       ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13665                              <1>       ; INT 1Ah
13666                              <1>       ; (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,:
13667                              <1>       ;      (CH) = CENTURY IN BCD (19 OR 20)                    :
13668                              <1>       ;      (CL) = YEAR IN BCD (00-99)                          :
13669                              <1>       ;      (DH) = MONTH IN BCD (01-12)                         :
13670                              <1>       ;      (DL) = DAY IN BCD (01-31).
13671                              <1>       ;
13672                              <1> RTC_40:                         ; GET RTC DATE
13673 00003CC8 FA                  <1>       cli
13674 00003CC9 E83ECFFFFF          <1>       CALL  UPD_IPR             ; CHECK FOR UPDATE IN PROCESS
13675 00003CCE 7225                <1>       JC    short RTC_49        ; EXIT IF ERROR (CY= 1)
13676                              <1>
13677 00003CD0 B007                <1>       MOV   AL,CMOS_DAY_MONTH   ; ADDRESS DAY OF MONTH
13678 00003CD2 E81DCFFFFF          <1>       CALL  CMOS_READ           ; READ DAY OF MONTH
13679 00003CD7 88C2                <1>       MOV   DL,AL               ; SAVE
13680 00003CD9 B008                <1>       MOV   AL,CMOS_MONTH       ; ADDRESS MONTH
13681 00003CDB E814CFFFFF          <1>       CALL  CMOS_READ           ; READ MONTH
13682 00003CE0 88C6                <1>       MOV   DH,AL               ; SAVE
13683 00003CE2 B009                <1>       MOV   AL,CMOS_YEAR        ; ADDRESS YEAR
13684 00003CE4 E80BCFFFFF          <1>       CALL  CMOS_READ           ; READ YEAR
13685 00003CE9 88C1                <1>       MOV   CL,AL               ; SAVE
13686 00003CEB B032                <1>       MOV   AL,CMOS_CENTURY     ; ADDRESS CENTURY LOCATION
13687 00003CED E802CFFFFF          <1>       CALL  CMOS_READ           ; GET CENTURY BYTE
13688 00003CF2 88C5                <1>       MOV   CH,AL               ; SAVE
13689 00003CF4 F8                  <1>       CLC                       ; SET CY=0
13690                              <1> RTC_49:
13691 00003CF5 FB                  <1>       sti
13692 00003CF6 C3                  <1>       RETn                      ; RETURN WITH RESULTS IN CARRY FLAG
13693                              <1>
13694                              <1> set_date_time:
13695                              <1> convert_from_epoch:
13696                              <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
13697                              <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
13698                              <1>       ; 'convert_from_epoch' procedure prototype:
13699                              <1>       ;                 UNIXCOPY.ASM, 10/03/2013
13700                              <1>       ;
13701                              <1>       ; ((Modified registers: EAX, EDX, ECX, EBX))
13702                              <1>       ;
13703                              <1>       ; Derived from DALLAS Semiconductor
13704                              <1>       ; Application Note 31 (DS1602/DS1603)
13705                              <1>       ; 6 May 1998
13706                              <1>       ;
13707                              <1>       ; INPUT:
13708                              <1>       ; EAX = Unix (Epoch) Time
13709                              <1>       ;
13710 00003CF7 31D2                <1>       xor   edx, edx
13711 00003CF9 B93C000000          <1>       mov   ecx, 60
13712 00003CFE F7F1                <1>       div   ecx
13713                              <1>       ;mov  [imin], eax   ; whole minutes
```

```
13714                                   <1>                     ; since 1/1/1970
13715 00003D00 668915[D66D0000]         <1>       mov   [second], dx  ; leftover seconds
13716 00003D07 29D2                     <1>       sub   edx, edx
13717 00003D09 F7F1                     <1>       div   ecx
13718                                   <1>       ;mov  [ihrs], eax   ; whole hours
13719                                   <1>       ;                   ; since 1/1/1970
13720 00003D0B 668915[D46D0000]         <1>       mov   [minute], dx  ; leftover minutes
13721 00003D12 31D2                     <1>       xor   edx, edx
13722                                   <1>       ;mov  cx, 24
13723 00003D14 B118                     <1>       mov   cl, 24
13724 00003D16 F7F1                     <1>       div   ecx
13725                                   <1>       ;mov  [iday], ax    ; whole days
13726                                   <1>                           ; since 1/1/1970
13727 00003D18 668915[D26D0000]         <1>       mov   [hour], dx    ; leftover hours
13728 00003D1F 05DB020000               <1>       add   eax, 365+366 ; whole day since
13729                                   <1>                           ; 1/1/1968
13730                                   <1>       ;mov  [iday], ax
13731 00003D24 50                       <1>       push  eax
13732 00003D25 29D2                     <1>       sub   edx, edx
13733 00003D27 B9B5050000               <1>       mov   ecx, (4*365)+1 ; 4 years = 1461 days
13734 00003D2C F7F1                     <1>       div   ecx
13735 00003D2E 59                       <1>       pop   ecx
13736                                   <1>       ;mov  [lday], ax    ; count of quadyrs (4 years)
13737 00003D2F 6652                     <1>       push  dx
13738                                   <1>       ;mov  [qday], dx    ; days since quadyr began
13739 00003D31 6683FA3C                 <1>       cmp   dx, 31 + 29   ; if past feb 29 then
13740 00003D35 F5                       <1>       cmc                 ; add this quadyr's leap day
13741 00003D36 83D000                   <1>       adc   eax, 0        ; to # of qadyrs (leap days)
13742                                   <1>       ;mov  [lday], ax    ; since 1968
13743                                   <1>       ;mov  cx, [iday]
13744 00003D39 91                       <1>       xchg  ecx, eax      ; ECX = lday, EAX = iday
13745 00003D3A 29C8                     <1>       sub   eax, ecx      ; iday - lday
13746 00003D3C B96D010000               <1>       mov   ecx, 365
13747 00003D41 31D2                     <1>       xor   edx, edx
13748                                   <1>       ; EAX = iday-lday, EDX = 0
13749 00003D43 F7F1                     <1>       div   ecx
13750                                   <1>       ;mov  [iyrs], ax    ; whole years since 1968
13751                                   <1>       ;jday = iday - (iyrs*365) - lday
13752                                   <1>       ;mov [jday], dx       ; days since 1/1 of current year
13753                                   <1>       ;add  eax, 1968
13754 00003D45 6605B007                 <1>       add   ax, 1968      ; compute year
13755 00003D49 66A3[CC6D0000]           <1>       mov   [year], ax
13756 00003D4F 6689D1                   <1>       mov   cx, dx
13757                                   <1>       ;mov  dx, [qday]
13758 00003D52 665A                     <1>       pop   dx
13759 00003D54 6681FA6D01               <1>       cmp   dx, 365           ; if qday <= 365 and qday >= 60
13760 00003D59 7709                     <1>       ja    short cfe1    ; jday = jday +1
13761 00003D5B 6683FA3C                 <1>       cmp   dx, 60        ; if past 2/29 and leap year then
13762 00003D5F F5                       <1>         cmc               ; add a leap day to the # of whole
13763 00003D60 6683D100                 <1>       adc   cx, 0         ; days since 1/1 of current year
13764                                   <1> cfe1:
13765                                   <1>       ;mov  [jday], cx
13766 00003D64 66BB0C00                 <1>       mov   bx, 12        ; estimate month
13767 00003D68 66BA6E01                 <1>       mov   dx, 366       ; mday, max. days since 1/1 is 365
13768 00003D6C 6683E003                 <1>       and   ax, 11b       ; year mod 4 (and dx, 3)
13769                                   <1> cfe2: ; Month calculation  ; 0 to 11  (11 to 0)
13770 00003D70 6639D1                   <1>       cmp   cx, dx        ; mday = # of days passed from 1/1
13771 00003D73 731D                     <1>       jnb   short cfe3
13772 00003D75 664B                     <1>       dec   bx            ; month = month - 1
13773 00003D77 66D1E3                   <1>       shl   bx, 1
13774 00003D7A 668B93[D86D0000]         <1>       mov   dx, [EBX+DMonth] ; # elapsed days at 1st of month
13775 00003D81 66D1EB                   <1>       shr   bx, 1         ; bx = month - 1 (0 to 11)
13776 00003D84 6683FB01                 <1>       cmp   bx, 1         ; if month > 2 and year mod 4  = 0
13777 00003D88 76E6                     <1>       jna   short cfe2    ; then mday = mday + 1
13778 00003D8A 08C0                     <1>       or    al, al        ; if past 2/29 and leap year then
13779 00003D8C 75E2                     <1>       jnz   short cfe2    ; add leap day (to mday)
13780 00003D8E 6642                     <1>       inc   dx            ; mday = mday + 1
13781 00003D90 EBDE                     <1>       jmp   short cfe2
13782                                   <1> cfe3:
13783 00003D92 6643                     <1>       inc   bx          ; -> bx = month, 1 to 12
13784 00003D94 66891D[CE6D0000]         <1>       mov   [month], bx
13785 00003D9B 6629D1                   <1>       sub   cx, dx      ; day = jday - mday + 1
13786 00003D9E 6641                     <1>       inc   cx
13787 00003DA0 66890D[D06D0000]         <1>       mov   [day], cx
13788                                   <1>
13789                                   <1>       ; eax, ebx, ecx, edx is changed at return
13790                                   <1>       ; output ->
13791                                   <1>       ; [year], [month], [day], [hour], [minute], [second]
13792                                   <1>
13793                                   <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
13794                                   <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
13795                                   <1> set_date:
13796 00003DA7 A0[CD6D0000]             <1>         mov     al, [year+1]
13797 00003DAC D40A                     <1>       aam   ; ah = al / 10, al = al mod 10
13798 00003DAE D510                     <1>       db    0D5h,10h     ; Undocumented inst. AAD
13799                                   <1>                          ; AL = AH * 10h + AL
13800 00003DB0 88C5                     <1>       mov   ch, al ; century (BCD)
13801 00003DB2 A0[CC6D0000]             <1>       mov   al, [year]
13802 00003DB7 D40A                     <1>       aam   ; ah = al / 10, al = al mod 10
13803 00003DB9 D510                     <1>       db    0D5h,10h     ; Undocumented inst. AAD
13804                                   <1>                          ; AL = AH * 10h + AL
13805 00003DBB 88C1                     <1>       mov   cl, al ; year (BCD)
13806 00003DBD A0[CE6D0000]             <1>         mov al, [month]
13807 00003DC2 D40A                     <1>       aam   ; ah = al / 10, al = al mod 10
13808 00003DC4 D510                     <1>       db    0D5h,10h     ; Undocumented inst. AAD
13809                                   <1>                          ; AL = AH * 10h + AL
13810 00003DC6 88C6                     <1>       mov   dh, al ; month (BCD)
13811 00003DC8 A0[D06D0000]             <1>       mov   al, [day]
13812 00003DCD D40A                     <1>       aam   ; ah = al / 10, al = al mod 10
13813 00003DCF D510                     <1>       db    0D5h,10h     ; Undocumented inst. AAD
13814                                   <1>                          ; AL = AH * 10h + AL
13815 00003DD1 88C6                     <1>       mov   dh, al ; day (BCD)
13816                                   <1>       ; Set real-time clock date
13817 00003DD3 E879000000               <1>       call  set_rtc_date
13818                                   <1> set_time:
```

```
13819                                  <1>         ; Read real-time clock time
13820                                  <1>       ; (get day light saving time bit status)
13821 00003DD8 FA                      <1>       cli
13822 00003DD9 E82ECEFFFF              <1>       CALL   UPD_IPR            ; CHECK FOR UPDATE IN PROCESS
13823                                  <1>       ; cf = 1 -> al = 0
13824 00003DDE 7207                    <1>         jc      short stime1
13825 00003DE0 B00B                    <1>       MOV    AL,CMOS_REG_B      ; ADDRESS ALARM REGISTER
13826 00003DE2 E80DCEFFFF              <1>       CALL   CMOS_READ          ; READ CURRENT VALUE OF DSE BIT
13827                                  <1> stime1:
13828 00003DE7 FB                      <1>       sti
13829 00003DE8 2401                    <1>       AND    AL,00000001B       ; MASK FOR VALID DSE BIT
13830 00003DEA 88C2                    <1>       MOV    DL,AL              ; SET [DL] TO ZERO FOR NO DSE BIT
13831                                  <1>       ; DL = 1 or 0 (day light saving time)
13832                                  <1>       ;
13833 00003DEC A0[D26D0000]            <1>       mov    al, [hour]
13834 00003DF1 D40A                    <1>       aam    ; ah = al / 10, al = al mod 10
13835 00003DF3 D510                    <1>       db     0D5h,10h     ; Undocumented inst. AAD
13836                                  <1>                           ; AL = AH * 10h + AL
13837 00003DF5 88C5                    <1>       mov    ch, al ; hour (BCD)
13838 00003DF7 A0[D46D0000]            <1>         mov     al, [minute]
13839 00003DFC D40A                    <1>       aam    ; ah = al / 10, al = al mod 10
13840 00003DFE D510                    <1>       db     0D5h,10h     ; Undocumented inst. AAD
13841                                  <1>                           ; AL = AH * 10h + AL
13842 00003E00 88C1                    <1>       mov    cl, al       ; minute (BCD)
13843 00003E02 A0[D66D0000]            <1>         mov     al, [second]
13844 00003E07 D40A                    <1>       aam    ; ah = al / 10, al = al mod 10
13845 00003E09 D510                    <1>       db     0D5h,10h     ; Undocumented inst. AAD
13846                                  <1>                           ; AL = AH * 10h + AL
13847 00003E0B 88C6                    <1>       mov    dh, al       ; second (BCD)
13848                                  <1>       ; Set real-time clock time
13849                                  <1>       ; call set_rtc_time
13850                                  <1> set_rtc_time:
13851                                  <1>       ; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)
13852                                  <1>       ; 15/03/2015
13853                                  <1>       ; Derived from IBM PC-XT Model 286 BIOS Source Code
13854                                  <1>       ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13855                                  <1>       ; INT 1Ah
13856                                  <1>       ; (AH) = 03H  SET THE REAL TIME CLOCK USING,               :
13857                                  <1>       ;      (CH) = HOURS IN BCD (00-23)                          :
13858                                  <1>       ;      (CL) = MINUTES IN BCD (00-59)                        :
13859                                  <1>       ;      (DH) = SECONDS IN BCD (00-59)                        :
13860                                  <1>       ;      (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.    :
13861                                  <1>       ;                                                           :
13862                                  <1>       ;  NOTE: (DL)= 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED. :
13863                                  <1>       ;        (DL)= 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN  :
13864                                  <1>       ;         APRIL   (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN :
13865                                  <1>       ;         OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.   :
13866                                  <1>       ;
13867                                  <1> RTC_30:                       ; SET RTC TIME
13868 00003E0D FA                      <1>       cli
13869 00003E0E E8F9CDFFFF              <1>       CALL   UPD_IPR            ; CHECK FOR UPDATE IN PROCESS
13870 00003E13 7305                    <1>       JNC    short RTC_35       ; GO AROUND IF CLOCK OPERATING
13871 00003E15 E886000000              <1>       CALL   RTC_STA            ; ELSE TRY INITIALIZING CLOCK
13872                                  <1> RTC_35:
13873 00003E1A 88F4                    <1>       MOV    AH,DH              ; GET TIME BYTE - SECONDS
13874 00003E1C B000                    <1>       MOV    AL,CMOS_SECONDS    ; ADDRESS SECONDS
13875 00003E1E E89E000000              <1>       CALL   CMOS_WRITE         ; UPDATE SECONDS
13876 00003E23 88CC                    <1>       MOV    AH,CL              ; GET TIME BYTE - MINUTES
13877 00003E25 B002                    <1>       MOV    AL,CMOS_MINUTES    ; ADDRESS MINUTES
13878 00003E27 E895000000              <1>       CALL   CMOS_WRITE         ; UPDATE MINUTES
13879 00003E2C 88EC                    <1>       MOV    AH,CH              ; GET TIME BYTE - HOURS
13880 00003E2E B004                    <1>       MOV    AL,CMOS_HOURS      ; ADDRESS HOURS
13881 00003E30 E88C000000              <1>       CALL   CMOS_WRITE         ; UPDATE ADDRESS
13882                                  <1>       ;MOV   AX,X*CMOS_REG_B    ; ADDRESS ALARM REGISTER
13883 00003E35 66B80B0B                <1>       MOV    AX,257*CMOS_REG_B  ;
13884 00003E39 E8B6CDFFFF              <1>       CALL   CMOS_READ          ; READ CURRENT TIME
13885 00003E3E 2462                    <1>       AND    AL,01100010B       ; MASK FOR VALID BIT POSITIONS
13886 00003E40 0C02                    <1>       OR     AL,00000010B       ; TURN ON 24 HOUR MODE
13887 00003E42 80E201                  <1>       AND    DL,00000001B       ; USE ONLY THE DSE BIT
13888 00003E45 08D0                    <1>       OR     AL,DL              ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
13889 00003E47 86E0                    <1>       XCHG   AH,AL              ; PLACE IN WORK REGISTER AND GET ADDRESS
13890 00003E49 E873000000              <1>       CALL   CMOS_WRITE         ; SET NEW ALARM BITS
13891 00003E4E F8                      <1>       CLC                       ; SET CY= 0
13892 00003E4F FB                      <1>       sti
13893 00003E50 C3                      <1>       RETn                      ; RETURN WITH CY= 0
13894                                  <1>
13895                                  <1> set_rtc_date:
13896                                  <1>       ; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)
13897                                  <1>       ; 15/03/2015
13898                                  <1>       ; Derived from IBM PC-XT Model 286 BIOS Source Code
13899                                  <1>       ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13900                                  <1>       ; INT 1Ah
13901                                  <1>       ; (AH) = 05H  SET THE DATE INTO THE REAL TIME CLOCK USING, :
13902                                  <1>       ;     (CH) = CENTURY IN BCD (19 OR 20)                   :
13903                                  <1>       ;     (CL) = YEAR IN BCD (00-99)                 :
13904                                  <1>       ;     (DH) = MONTH IN BCD (01-12)                        :
13905                                  <1>       ;     (DL) = DAY IN BCD (01-31).
13906                                  <1>       ;
13907                                  <1> RTC_50:                       ; SET RTC DATE
13908 00003E51 FA                      <1>       cli
13909 00003E52 E8B5CDFFFF              <1>       CALL   UPD_IPR            ; CHECK FOR UPDATE IN PROCESS
13910 00003E57 7305                    <1>       JNC    short RTC_55       ; GO AROUND IF NO ERROR
13911 00003E59 E842000000              <1>       CALL   RTC_STA            ; ELSE INITIALIZE CLOCK
13912                                  <1> RTC_55:
13913 00003E5E 66B80600                <1>       MOV    AX,CMOS_DAY_WEEK   ; ADDRESS OF DAY OF WEEK BYTE
13914 00003E62 E85A000000              <1>       CALL   CMOS_WRITE         ; LOAD ZEROS TO DAY OF WEEK
13915 00003E67 88D4                    <1>       MOV    AH,DL              ; GET DAY OF MONTH BYTE
13916 00003E69 B007                    <1>       MOV    AL,CMOS_DAY_MONTH  ; ADDRESS DAY OF MONTH BYTE
13917 00003E6B E851000000              <1>       CALL   CMOS_WRITE         ; WRITE OF DAY OF MONTH REGISTER
13918 00003E70 88F4                    <1>       MOV    AH,DH              ; GET MONTH
13919 00003E72 B008                    <1>       MOV    AL,CMOS_MONTH      ; ADDRESS MONTH BYTE
13920 00003E74 E848000000              <1>       CALL   CMOS_WRITE         ; WRITE MONTH REGISTER
13921 00003E79 88CC                    <1>       MOV    AH,CL              ; GET YEAR BYTE
13922 00003E7B B009                    <1>       MOV    AL,CMOS_YEAR       ; ADDRESS YEAR REGISTER
13923 00003E7D E83F000000              <1>       CALL   CMOS_WRITE         ; WRITE YEAR REGISTER
```

```
13924 00003E82 88EC              <1>       MOV    AH,CH               ; GET CENTURY BYTE
13925 00003E84 B032              <1>       MOV    AL,CMOS_CENTURY     ; ADDRESS CENTURY BYTE
13926 00003E86 E836000000        <1>       CALL   CMOS_WRITE          ; WRITE CENTURY LOCATION
13927                            <1>       ;MOV   AX,X*CMOS_REG_B     ; ADDRESS ALARM REGISTER
13928 00003E8B 66B80B0B          <1>       MOV    AX,257*CMOS_REG_B   ;
13929 00003E8F E860CDFFFF        <1>       CALL   CMOS_READ           ; READ CURRENT SETTINGS
13930 00003E94 247F              <1>       AND    AL,07FH             ; CLEAR 'SET BIT'
13931 00003E96 86E0              <1>       XCHG   AH,AL               ; MOVE TO WORK REGISTER
13932 00003E98 E824000000        <1>       CALL   CMOS_WRITE          ; AND START CLOCK UPDATING
13933 00003E9D F8                <1>       CLC                        ; SET CY= 0
13934 00003E9E FB                <1>       sti
13935 00003E9F C3                <1>       RETn                       ; RETURN CY=0
13936                            <1>
13937                            <1>       ; 15/03/2015
13938                            <1> RTC_STA:                         ; INITIALIZE REAL TIME CLOCK
13939 00003EA0 B426              <1>       mov    ah, 26h
13940 00003EA2 B00A              <1>       mov    al, CMOS_REG_A            ; ADDRESS REGISTER A AND LOAD DATA MASK
13941 00003EA4 E818000000        <1>       CALL   CMOS_WRITE          ; INITIALIZE STATUS REGISTER A
13942 00003EA9 B482              <1>       mov    ah, 82h
13943 00003EAB B00B              <1>       mov    al, CMOS_REG_B            ; SET "SET BIT" FOR CLOCK INITIALIZATION
13944 00003EAD E80F000000        <1>       CALL   CMOS_WRITE          ; AND 24 HOUR MODE TO REGISTER B
13945 00003EB2 B00C              <1>       MOV    AL,CMOS_REG_C       ; ADDRESS REGISTER C
13946 00003EB4 E83BCDFFFF        <1>       CALL   CMOS_READ           ; READ REGISTER C TO INITIALIZE
13947 00003EB9 B00D              <1>       MOV    AL,CMOS_REG_D       ; ADDRESS REGISTER D
13948 00003EBB E834CDFFFF        <1>       CALL   CMOS_READ           ; READ REGISTER D TO INITIALIZE
13949 00003EC0 C3                <1>       RETn
13950                            <1>
13951                            <1>       ; 15/03/2015
13952                            <1>       ; IBM PC/XT Model 286 BIOS source code ----- 10/06/85 (test4.asm)
13953                            <1> CMOS_WRITE:              ; WRITE (AH) TO LOCATION (AL)
13954 00003EC1 9C                <1>       pushf              ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
13955                            <1>       ;push  ax          ; SAVE WORK REGISTER VALUES
13956 00003EC2 D0C0              <1>       rol    al, 1       ; MOVE NMI BIT TO LOW POSITION
13957 00003EC4 F9                <1>       stc                ; FORCE NMI BIT ON IN CARRY FLAG
13958 00003EC5 D0D8              <1>       rcr    al, 1       ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
13959 00003EC7 FA                <1>       cli                ; DISABLE INTERRUPTS
13960 00003EC8 E670              <1>       out    CMOS_PORT, al; ADDRESS LOCATION AND DISABLE NMI
13961 00003ECA 88E0              <1>       mov    al, ah      ; GET THE DATA BYTE TO WRITE
13962 00003ECC E671              <1>       out    CMOS_DATA, al; PLACE IN REQUESTED CMOS LOCATION
13963 00003ECE B01E              <1>       mov    al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
13964 00003ED0 D0D8              <1>       rcr    al, 1       ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
13965 00003ED2 E670              <1>       out    CMOS_PORT, al; SET DEFAULT TO READ ONLY REGISTER
13966 00003ED4 90                <1>       nop                ; I/O DELAY
13967 00003ED5 E471              <1>       in     al, CMOS_DATA; OPEN STANDBY LATCH
13968                            <1>       ;pop   ax          ; RESTORE WORK REGISTERS
13969 00003ED7 9D                <1>       popf
13970 00003ED8 C3                <1>       RETn
13971                            <1>
13972                            <1> bf_init:
13973                            <1>       ; 14/08/2015
13974                            <1>       ; 02/07/2015
13975                            <1>       ; 01/07/2015
13976                            <1>       ; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
13977                            <1>       ; Buffer (pointer) initialization !
13978                            <1>       ;
13979                            <1>       ; 17/07/2013 - 24/07/2013
13980                            <1>       ; Retro UNIX 8086 v1 (U9.ASM)
13981                            <1>       ; (Retro UNIX 8086 v1 feature only !)
13982                            <1>       ;
13983 00003ED9 BF[0A740000]      <1>       mov    edi, bufp
13984 00003EDE B8[00810000]      <1>       mov    eax, buffer + (nbuf*520)
13985 00003EE3 29D2              <1>       sub    edx, edx
13986 00003EE5 FECA              <1>       dec    dl
13987 00003EE7 31C9              <1>       xor    ecx, ecx
13988 00003EE9 49                <1>       dec    ecx
13989                            <1> bi0:
13990 00003EEA 2D08020000        <1>       sub    eax, 520 ; 8 header + 512 data
13991 00003EEF AB                <1>       stosd
13992 00003EF0 89C6              <1>       mov    esi, eax
13993 00003EF2 8916              <1>       mov    [esi], edx ; 000000FFh
13994                            <1>                         ; Not a valid device sign
13995 00003EF4 894E04            <1>       mov    [esi+4], ecx ; 0FFFFFFFFh
13996                            <1>                           ; Not a valid block number sign
13997 00003EF7 3D[D0740000]      <1>       cmp    eax, buffer
13998 00003EFC 77EC              <1>       ja     short bi0
13999 00003EFE B8[00810000]      <1>       mov    eax, sb0
14000 00003F03 AB                <1>       stosd
14001 00003F04 B8[08830000]      <1>       mov    eax, sb1
14002 00003F09 AB                <1>       stosd
14003 00003F0A 89C6              <1>       mov    esi, eax ; offset sb1
14004 00003F0C 8916              <1>       mov    [esi], edx ; 000000FFh
14005                            <1>                           ; Not a valid device sign
14006 00003F0E 894E04            <1>       mov    [esi+4], ecx ; 0FFFFFFFFh
14007                            <1>                           ; Not a valid block number sign
14008                            <1>       ; 14/08/2015
14009                            <1>       ;call  rdev_init
14010                            <1>       ;retn
14011                            <1>
14012                            <1> rdev_init: ; root device, super block buffer initialization
14013                            <1>       ; 14/08/2015
14014                            <1>       ; Retro UNIX 386 v1 feature only !
14015                            <1>       ;
14016                            <1>       ; NOTE: Disk partitions (file systems), logical
14017                            <1>       ; drive initialization, partition's start sector etc.
14018                            <1>       ; will be coded here, later in 'ldrv_init'
14019                            <1>
14020 00003F11 0FB605[1A6B0000]  <1>       movzx  eax, byte [boot_drv]
14021                            <1> rdi_0:
14022 00003F18 3C80              <1>       cmp    al, 80h
14023 00003F1A 7202              <1>       jb     short rdi_1
14024 00003F1C 2C7E              <1>       sub    al, 7Eh ; 80h = 2 (hd0), 81h = 3 (hd1)
14025                            <1> rdi_1:
14026 00003F1E A2[30740000]      <1>       mov    [rdev], al
14027 00003F23 BB[00810000]      <1>         mov  ebx, sb0 ; super block buffer
14028 00003F28 8903              <1>       mov    [ebx], eax
```

```
14029 00003F2A B001                <1>       mov    al, 1 ; eax = 1
14030 00003F2C 894304              <1>       mov    [ebx+4], eax ; super block address on disk
14031 00003F2F E838240000          <1>       call   diskio
14032 00003F34 C3                  <1>       retn
14033                              <1>
14034                              <1> ; 23/10/2015
14035                              <1> com1_irq4:
14036 00003F35 [3D3F0000]          <1>       dd dummy_retn
14037                              <1> com2_irq3:
14038 00003F39 [3D3F0000]          <1>       dd dummy_retn
14039                              <1>
14040                              <1> dummy_retn:
14041 00003F3D C3                  <1>       retn
14042                                  %include 'u1.s'        ; 10/05/2015
14043                              <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS1.INC
14044                              <1> ; Last Modification: 27/12/2015
14045                              <1> ; ----------------------------------------------------------------------------
14046                              <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
14047                              <1> ; (v0.1 - Beginning: 11/07/2012)
14048                              <1> ;
14049                              <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
14050                              <1> ; (Original) Source Code by Ken Thompson (1971-1972)
14051                              <1> ; <Bell Laboratories (17/3/1972)>
14052                              <1> ; <Preliminary Release of UNIX Implementation Document>
14053                              <1> ;
14054                              <1> ; Retro UNIX 8086 v1 - U1.ASM (12/07/2014) //// UNIX v1 -> u1.s
14055                              <1> ;
14056                              <1> ; ***************************************************************************
14057                              <1>
14058                              <1> unkni: ; / used for all system calls
14059                              <1> sysent: ; < enter to system call >
14060                              <1>        ;19/10/2015
14061                              <1>       ; 21/09/2015
14062                              <1>       ; 01/07/2015
14063                              <1>       ; 19/05/2015
14064                              <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14065                              <1>       ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
14066                              <1>       ;
14067                              <1>       ; 'unkni' or 'sysent' is sytem entry from various traps.
14068                              <1>       ; The trap type is determined and an indirect jump is made to
14069                              <1>       ; the appropriate system call handler. If there is a trap inside
14070                              <1>       ; the system a jump to panic is made. All user registers are saved
14071                              <1>       ; and u.sp points to the end of the users stack. The sys (trap)
14072                              <1>       ; instructor is decoded to get the the system code part (see
14073                              <1>       ; trap instruction in the PDP-11 handbook) and from this
14074                              <1>       ; the indirect jump address is calculated. If a bad system call is
14075                              <1>       ; made, i.e., the limits of the jump table are exceeded, 'badsys'
14076                              <1>       ; is called. If the call is legitimate control passes to the
14077                              <1>       ; appropriate system routine.
14078                              <1>       ;
14079                              <1>       ; Calling sequence:
14080                              <1>       ;     Through a trap caused by any sys call outside the system.
14081                              <1>       ; Arguments:
14082                              <1>       ;     Arguments of particular system call.
14083                              <1>       ; ....................................................
14084                              <1>       ;
14085                              <1>       ; Retro UNIX 8086 v1 modification:
14086                              <1>       ;      System call number is in EAX register.
14087                              <1>       ;
14088                              <1>       ;      Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
14089                              <1>       ;     registers depending of function details.
14090                              <1>       ;
14091                              <1>       ; 16/04/2015
14092 00003F3E 368925[40740000]    <1>         mov     [ss:u.sp], esp ; Kernel stack points to return address
14093                              <1>       ; save user registers
14094 00003F45 1E                  <1>       push   ds
14095 00003F46 06                  <1>       push   es
14096 00003F47 0FA0                <1>       push   fs
14097 00003F49 0FA8                <1>       push   gs
14098 00003F4B 60                  <1>       pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
14099                              <1>       ;
14100                              <1>       ; ESPACE = esp - [ss:u.sp] ; 4*12 = 48 ; 17/09/2015
14101                              <1>       ;     (ESPACE is size of space in kernel stack
14102                              <1>       ;     for saving/restoring user registers.)
14103                              <1>       ;
14104 00003F4C 50                  <1>       push   eax ; 01/07/2015
14105 00003F4D 66B81000            <1>       mov     ax, KDATA
14106 00003F51 8ED8                <1>         mov     ds, ax
14107 00003F53 8EC0                <1>         mov     es, ax
14108 00003F55 8EE0                <1>         mov     fs, ax
14109 00003F57 8EE8                <1>         mov     gs, ax
14110 00003F59 A1[68700000]        <1>       mov    eax, [k_page_dir]
14111 00003F5E 0F22D8              <1>       mov    cr3, eax
14112 00003F61 58                  <1>       pop    eax ; 01/07/2015
14113                              <1>       ; 19/10/2015
14114 00003F62 FC                  <1>       cld
14115                              <1>       ;
14116 00003F63 FE05[3F740000]      <1>       inc    byte [sysflg]
14117                              <1>              ; incb sysflg / indicate a system routine is in progress
14118 00003F69 FB                  <1>         sti  ; 18/01/2014
14119 00003F6A 0F8560F9FFFF        <1>       jnz     panic ; 24/05/2013
14120                              <1>              ; beq 1f
14121                              <1>              ; jmp panic ; / called if trap inside system
14122                              <1> ;1:
14123                              <1>       ; 16/04/2015
14124 00003F70 A3[48740000]        <1>       mov    [u.r0], eax
14125 00003F75 8925[44740000]      <1>       mov    [u.usp], esp ; kernel stack points to user's registers
14126                              <1>       ;
14127                              <1>              ; mov $s.syst+2,clockp
14128                              <1>              ; mov r0,-(sp) / save user registers
14129                              <1>              ; mov sp,u.r0 / pointer to bottom of users stack
14130                              <1>                       ; / in u.r0
14131                              <1>              ; mov r1,-(sp)
14132                              <1>              ; mov r2,-(sp)
14133                              <1>              ; mov r3,-(sp)
```

```
14134                                  <1>             ; mov r4,-(sp)
14135                                  <1>             ; mov r5,-(sp)
14136                                  <1>             ; mov ac,-(sp) / "accumulator" register for extended
14137                                  <1>                             ; / arithmetic unit
14138                                  <1>             ; mov mq,-(sp) / "multiplier quotient" register for the
14139                                  <1>                             ; / extended arithmetic unit
14140                                  <1>             ; mov sc,-(sp) / "step count" register for the extended
14141                                  <1>                             ; / arithmetic unit
14142                                  <1>             ; mov sp,u.sp / u.sp points to top of users stack
14143                                  <1>             ; mov 18.(sp),r0 / store pc in r0
14144                                  <1>             ; mov -(r0),r0 / sys inst in r0      10400xxx
14145                                  <1>             ; sub $sys,r0 / get xxx code
14146 00003F7B C1E002                 <1>       shl   eax, 2
14147                                  <1>             ; asl r0 / multiply by 2 to jump indirect in bytes
14148 00003F7E 3D94000000             <1>       cmp   eax, end_of_syscalls - syscalls
14149                                  <1>             ; cmp r0,$2f-1f / limit of table (35) exceeded
14150                                  <1>       ;jnb  short badsys
14151                                  <1>             ; bhis badsys / yes, bad system call
14152 00003F83 F5                     <1>       cmc
14153 00003F84 9C                     <1>       pushf
14154 00003F85 50                     <1>       push  eax
14155 00003F86 8B2D[40740000]         <1>       mov   ebp, [u.sp] ; Kernel stack at the beginning of sys call
14156 00003F8C B0FE                   <1>       mov   al, 0FEh ; 11111110b
14157 00003F8E 1400                   <1>       adc   al, 0 ; al = al + cf
14158 00003F90 204508                 <1>       and   [ebp+8], al ; flags (reset carry flag)
14159                                  <1>             ; bic $341,20.(sp) / set users processor priority to 0
14160                                  <1>                               ; / and clear carry bit
14161 00003F93 5D                     <1>       pop   ebp ; eax
14162 00003F94 9D                     <1>       popf
14163 00003F95 0F8248010000           <1>         jc      badsys
14164 00003F9B A1[48740000]           <1>       mov   eax, [u.r0]
14165                                  <1>       ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
14166 00003FA0 FFA5[A63F0000]         <1>       jmp   dword [ebp+syscalls]
14167                                  <1>             ; jmp *1f(r0) / jump indirect thru table of addresses
14168                                  <1>                             ; / to proper system routine.
14169                                  <1> syscalls: ; 1:
14170                                  <1>       ; 21/09/2015
14171                                  <1>       ; 01/07/2015
14172                                  <1>       ; 16/04/2015 (32 bit address modification)
14173 00003FA6 [AD400000]             <1>       dd sysrele   ; / 0
14174 00003FAA [53410000]             <1>       dd sysexit   ; / 1
14175 00003FAE [78420000]             <1>       dd sysfork   ; / 2
14176 00003FB2 [8B430000]             <1>       dd sysread   ; / 3
14177 00003FB6 [A6430000]             <1>       dd syswrite  ; / 4
14178 00003FBA [10440000]             <1>       dd sysopen   ; / 5
14179 00003FBE [4A450000]             <1>       dd sysclose  ; / 6
14180 00003FC2 [FA410000]             <1>       dd syswait   ; / 7
14181 00003FC6 [C0440000]             <1>       dd syscreat  ; / 8
14182 00003FCA [71480000]             <1>       dd syslink   ; / 9
14183 00003FCE [33490000]             <1>       dd sysunlink ; / 10
14184 00003FD2 [064A0000]             <1>       dd sysexec   ; / 11
14185 00003FD6 [6D500000]             <1>       dd syschdir  ; / 12
14186 00003FDA [51510000]             <1>       dd systime   ; / 13
14187 00003FDE [01450000]             <1>       dd sysmkdir  ; / 14
14188 00003FE2 [BF500000]             <1>       dd syschmod  ; / 15
14189 00003FE6 [21510000]             <1>       dd syschown  ; / 16
14190 00003FEA [84510000]             <1>       dd sysbreak  ; / 17
14191 00003FEE [DE4D0000]             <1>       dd sysstat   ; / 18
14192 00003FF2 [49520000]             <1>       dd sysseek   ; / 19
14193 00003FF6 [5B520000]             <1>       dd systell   ; / 20
14194 00003FFA [585D0000]             <1>       dd sysmount  ; / 21
14195 00003FFE [0A5E0000]             <1>       dd sysumount ; / 22
14196 00004002 [D9520000]             <1>       dd syssetuid ; / 23
14197 00004006 [0A530000]             <1>       dd sysgetuid ; / 24
14198 0000400A [60510000]             <1>       dd sysstime  ; / 25
14199 0000400E [CD520000]             <1>       dd sysquit   ; / 26
14200 00004012 [C1520000]             <1>       dd sysintr   ; / 27
14201 00004016 [BA4D0000]             <1>       dd sysfstat  ; / 28
14202 0000401A [66450000]             <1>       dd sysemt    ; / 29
14203 0000401E [94450000]             <1>       dd sysmdate  ; / 30
14204 00004022 [DF450000]             <1>       dd sysstty   ; / 31
14205 00004026 [5E470000]             <1>       dd sysgtty   ; / 32
14206 0000402A [8F450000]             <1>       dd sysilgins ; / 33
14207 0000402E [3F660000]             <1>       dd syssleep  ; 34 ; Retro UNIX 8086 v1 feature only !
14208                                  <1>                            ; 11/06/2014
14209 00004032 [6E660000]             <1>       dd sysmsg    ; 35 ; Retro UNIX 386 v1 feature only !
14210                                  <1>                            ; 01/07/2015
14211 00004036 [45670000]             <1>       dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
14212                                  <1>                            ; 21/09/2015 - get last error number
14213                                  <1> end_of_syscalls:
14214                                  <1>
14215                                  <1> error:
14216                                  <1>       ; 17/09/2015
14217                                  <1>       ; 03/09/2015
14218                                  <1>       ; 01/09/2015
14219                                  <1>       ; 09/06/2015
14220                                  <1>       ; 13/05/2015
14221                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14222                                  <1>       ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
14223                                  <1>       ;
14224                                  <1>       ; 'error' merely sets the error bit off the processor status (c-bit)
14225                                  <1>       ; then falls right into the 'sysret', 'sysrele' return sequence.
14226                                  <1>       ;
14227                                  <1>       ; INPUTS -> none
14228                                  <1>       ; OUTPUTS ->
14229                                  <1>       ;     processor status - carry (c) bit is set (means error)
14230                                  <1>       ;
14231                                  <1>       ; 26/05/2013 (Stack pointer must be reset here!
14232                                  <1>       ;           Because, jumps to error procedure
14233                                  <1>       ;           disrupts push-pop nesting balance)
14234                                  <1>       ;
14235 0000403A 8B2D[40740000]         <1>       mov   ebp, [u.sp] ; interrupt (system call) return (iretd) address
14236 00004040 804D0801               <1>       or    byte [ebp+8], 1  ; set carry bit of flags register
14237                                  <1>                              ; (system call will return with cf = 1)
14238                                  <1>             ; bis $1,20.(r1) / set c bit in processor status word below
```

```
14239                                  <1>                             ; / users stack
14240                                  <1>       ; 17/09/2015
14241 00004044 83ED30                  <1>       sub    ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
14242                                  <1>                               ; for saving/restoring user registers
14243                                  <1>       ;cmp   ebp, [u.usp]
14244                                  <1>       ;je    short err0
14245 00004047 892D[44740000]          <1>       mov    [u.usp], ebp
14246                                  <1> ;err0:
14247                                  <1>       ; 01/09/2015
14248 0000404D 8B25[44740000]          <1>       mov    esp, [u.usp]     ; Retro Unix 8086 v1 modification!
14249                                  <1>                               ; 10/04/2013
14250                                  <1>                               ; (If an I/O error occurs during disk I/O,
14251                                  <1>                               ; related procedures will jump to 'error'
14252                                  <1>                               ; procedure directly without returning to
14253                                  <1>                               ; the caller procedure. So, stack pointer
14254                                  <1>                                    ; must be restored here.)
14255                                  <1>       ; 13/05/2015
14256                                  <1>       ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
14257                                  <1>       ;      'get last error' system call later.
14258                                  <1>
14259                                  <1>       ; 03/09/2015 - 09/06/2015 - 07/08/2013
14260 00004053 C605[AF740000]00        <1>       mov    byte [u.kcall], 0 ; namei_r, mkdir_w reset
14261                                  <1>
14262                                  <1> sysret: ; < return from system call>
14263                                  <1>       ; 10/09/2015
14264                                  <1>       ; 29/07/2015
14265                                  <1>       ; 25/06/2015
14266                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14267                                  <1>       ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
14268                                  <1>       ;
14269                                  <1>       ; 'sysret' first checks to see if process is about to be
14270                                  <1>       ; terminated (u.bsys). If it is, 'sysexit' is called.
14271                                  <1>       ; If not, following happens:
14272                                  <1>       ;     1) The user's stack pointer is restored.
14273                                  <1>       ;     2) r1=0 and 'iget' is called to see if last mentioned
14274                                  <1>       ;        i-node has been modified. If it has, it is written out
14275                                  <1>       ;        via 'ppoke'.
14276                                  <1>       ;     3) If the super block has been modified, it is written out
14277                                  <1>       ;        via 'ppoke'.
14278                                  <1>       ;     4) If the dismountable file system's super block has been
14279                                  <1>       ;        modified, it is written out to the specified device
14280                                  <1>       ;        via 'ppoke'.
14281                                  <1>       ;     5) A check is made if user's time quantum (uquant) ran out
14282                                  <1>       ;        during his execution. If so, 'tswap' is called to give
14283                                  <1>       ;        another user a chance to run.
14284                                  <1>       ;     6) 'sysret' now goes into 'sysrele'.
14285                                  <1>       ;         (See 'sysrele' for conclusion.)
14286                                  <1>       ;
14287                                  <1>       ; Calling sequence:
14288                                  <1>       ;     jump table or 'br sysret'
14289                                  <1>       ; Arguments:
14290                                  <1>       ;     -
14291                                  <1>       ; ..........................................................
14292                                  <1>       ;
14293                                  <1>       ; ((AX=r1 for 'iget' input))
14294                                  <1>       ;
14295 0000405A 6631C0                  <1>       xor    ax, ax ; 04/05/2013
14296                                  <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
14297 0000405D FEC0                    <1>       inc    al ; 04/05/2013
14298 0000405F 3805[96740000]          <1>       cmp    [u.bsys], al ; 1
14299                                  <1>              ; tstb u.bsys / is a process about to be terminated because
14300 00004065 0F83E8000000            <1>         jnb     sysexit ; 04/05/2013
14301                                  <1>              ; bne sysexit / of an error? yes, go to sysexit
14302                                  <1>       ;mov   esp, [u.usp] ; 24/05/2013 (that is not needed here)
14303                                  <1>              ; mov u.sp,sp / no point stack to users stack
14304 0000406B FEC8                    <1>       dec    al ; mov ax, 0
14305                                  <1>              ; clr r1 / zero r1 to check last mentioned i-node
14306 0000406D E874160000              <1>       call   iget
14307                                  <1>              ; jsr r0,iget / if last mentioned i-node has been modified
14308                                  <1>                              ; / it is written out
14309 00004072 6631C0                  <1>       xor    ax, ax ; 0
14310 00004075 3805[3D740000]          <1>       cmp    [smod], al ; 0
14311                                  <1>              ; tstb smod / has the super block been modified
14312 0000407B 7614                    <1>       jna    short sysret1
14313                                  <1>              ; beq  1f / no, 1f
14314 0000407D A2[3D740000]            <1>       mov    [smod], al ; 0
14315                                  <1>              ; clrb smod / yes, clear smod
14316 00004082 BB[00810000]            <1>       mov    ebx, sb0 ;; 07/08//2013
14317 00004087 66810B0002              <1>       or     word [ebx], 200h ;;
14318                                  <1>       ;or    word [sb0], 200h ; write bit, bit 9
14319                                  <1>              ; bis $1000,sb0 / set write bit in I/O queue for super block
14320                                  <1>                              ; / output
14321                                  <1>       ; AX = 0
14322 0000408C E8C0210000             <1>       call   poke ; 07/08/2013
14323                                  <1>       ; call ppoke
14324                                  <1>       ; AX = 0
14325                                  <1>              ; jsr r0,ppoke / write out modified super block to disk
14326                                  <1> sysret1: ;1:
14327 00004091 3805[3E740000]          <1>       cmp    [mmod], al ; 0
14328                                  <1>              ; tstb mmod / has the super block for the dismountable file
14329                                  <1>                           ; / system
14330 00004097 7614                    <1>       jna    short sysrel0
14331                                  <1>              ; beq 1f / been modified?  no, 1f
14332 00004099 A2[3E740000]            <1>       mov    [mmod], al ; 0
14333                                  <1>              ; clrb mmod / yes, clear mmod
14334                                  <1>         ;mov    ax, [mntd]
14335                                  <1>         ;;mov   al, [mdev] ; 26/04/2013
14336 0000409E BB[08830000]            <1>       mov    ebx, sb1 ;; 07/08//2013
14337                                  <1>         ;;mov      [ebx], al
14338                                  <1>       ;mov    [sb1], al
14339                                  <1>              ; movb mntd,sb1 / set the I/O queue
14340 000040A3 66810B0002             <1>       or     word [ebx], 200h
14341                                  <1>       ;or    word [sb1], 200h ; write bit, bit 9
14342                                  <1>              ; bis $1000,sb1 / set write bit in I/O queue for detached sb
14343 000040A8 E8A4210000             <1>       call   poke ; 07/08/2013
```

```
14344                                  <1>       ;call  ppoke
14345                                  <1>              ; jsr r0,ppoke / write it out to its device
14346                                  <1>         ;xor    al, al ; 26/04/2013
14347                                  <1> ;1:
14348                                  <1>              ; tstb uquant / is the time quantum 0?
14349                                  <1>              ; bne 1f / no, don't swap it out
14350                                  <1>
14351                                  <1> sysrele: ; < release >
14352                                  <1>       ; 14/10/2015
14353                                  <1>       ; 01/09/2015
14354                                  <1>       ; 24/07/2015
14355                                  <1>       ; 14/05/2015
14356                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14357                                  <1>       ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
14358                                  <1>       ;
14359                                  <1>       ; 'sysrele' first calls 'tswap' if the time quantum for a user is
14360                                  <1>       ;  zero (see 'sysret'). It then restores the user's registers and
14361                                  <1>       ; turns off the system flag. It then checked to see if there is
14362                                  <1>       ; an interrupt from the user by calling 'isintr'. If there is,
14363                                  <1>       ; the output gets flashed (see isintr) and interrupt action is
14364                                  <1>       ; taken by a branch to 'intract'. If there is no interrupt from
14365                                  <1>       ; the user, a rti is made.
14366                                  <1>       ;
14367                                  <1>       ; Calling sequence:
14368                                  <1>       ;     Fall through a 'bne' in 'sysret' & ?
14369                                  <1>       ; Arguments:
14370                                  <1>       ;     -
14371                                  <1>       ; ..........................................................
14372                                  <1>       ;
14373                                  <1>       ; 23/02/2014 (swapret)
14374                                  <1>       ; 22/09/2013
14375                                  <1> sysrel0: ;1:
14376 000040AD 803D[8A740000]00       <1>       cmp    byte [u.quant], 0 ; 16/05/2013
14377                                  <1>              ; tstb uquant / is the time quantum 0?
14378 000040B4 7705                   <1>         ja      short swapret
14379                                  <1>              ; bne 1f / no, don't swap it out
14380                                  <1> sysrelease: ; 07/12/2013 (jump from 'clock')
14381 000040B6 E8A1120000             <1>       call   tswap
14382                                  <1>              ; jsr r0,tswap / yes, swap it out
14383                                  <1> ;
14384                                  <1> ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
14385                                  <1> swapret: ;1:
14386                                  <1>       ; 10/09/2015
14387                                  <1>       ; 01/09/2015
14388                                  <1>       ; 14/05/2015
14389                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
14390                                  <1>       ; 26/05/2013 (Retro UNIX 8086 v1)
14391                                  <1>       ; cli
14392                                  <1>       ; 24/07/2015
14393                                  <1>       ;
14394                                  <1>       ;; 'esp' must be already equal to '[u.usp]' here !
14395                                  <1>       ;; mov esp, [u.usp]
14396                                  <1>
14397                                  <1>       ; 22/09/2013
14398 000040BB E886140000             <1>       call   isintr
14399                                  <1>       ; 20/10/2013
14400 000040C0 7405                   <1>       jz     short sysrel1
14401 000040C2 E875000000             <1>       call   intract
14402                                  <1>              ; jsr r0,isintr / is there an interrupt from the user
14403                                  <1>              ;     br intract / yes, output gets flushed, take interrupt
14404                                  <1>                                 ; / action
14405                                  <1> sysrel1:
14406 000040C7 FA                     <1>       cli ; 14/10/2015
14407 000040C8 FE0D[3F740000]         <1>       dec    byte [sysflg]
14408                                  <1>              ; decb sysflg / turn system flag off
14409 000040CE A1[A1740000]           <1>       mov     eax, [u.pgdir]
14410 000040D3 0F22D8                 <1>       mov    cr3, eax  ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
14411                                  <1>                        ; (others are different than kernel page tables)
14412                                  <1>       ; 10/09/2015
14413 000040D6 61                     <1>       popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
14414                                  <1>              ; mov (sp)+,sc / restore user registers
14415                                  <1>              ; mov (sp)+,mq
14416                                  <1>              ; mov (sp)+,ac
14417                                  <1>              ; mov (sp)+,r5
14418                                  <1>              ; mov (sp)+,r4
14419                                  <1>              ; mov (sp)+,r3
14420                                  <1>              ; mov (sp)+,r2
14421                                  <1>       ;
14422 000040D7 A1[48740000]           <1>       mov    eax, [u.r0]  ; ((return value in EAX))
14423 000040DC 0FA9                   <1>       pop    gs
14424 000040DE 0FA1                   <1>       pop    fs
14425 000040E0 07                     <1>       pop    es
14426 000040E1 1F                     <1>       pop    ds
14427 000040E2 CF                     <1>       iretd
14428                                  <1>              ; rti / no, return from interrupt
14429                                  <1>
14430                                  <1> badsys:
14431                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14432                                  <1>       ; (Major Modification: 'core' dumping procedure in
14433                                  <1>         ;       original UNIX v1 and Retro UNIX 8086 v1
14434                                  <1>       ;     has been changed to print 'Invalid System Call !'
14435                                  <1>       ;     message on the user's console tty.)
14436                                  <1>       ; (EIP, EAX values will be shown on screen with error message)
14437                                  <1>       ; (EIP = Return address just after the system call -INT 30h-)
14438                                  <1>       ; (EAX = Function number)
14439                                  <1>       ;
14440 000040E3 FE05[96740000]         <1>       inc    byte [u.bsys]
14441                                  <1>       ;
14442 000040E9 8B1D[40740000]         <1>       mov    ebx, [u.sp] ; esp at the beginning of 'sysent'
14443 000040EF 8B03                   <1>       mov    eax, [ebx] ; EIP (return address, not 'INT 30h' address)
14444 000040F1 E8E7D7FFFF             <1>       call   dwordtohex
14445 000040F6 8915[C06D0000]         <1>       mov    [bsys_msg_eip], edx
14446 000040FC A3[C46D0000]           <1>       mov    [bsys_msg_eip+4], eax
14447 00004101 A1[48740000]           <1>       mov    eax, [u.r0]
14448 00004106 E8D2D7FFFF             <1>       call   dwordtohex
```

```
14449 0000410B 8915[B06D0000]      <1>       mov   [bsys_msg_eax], edx
14450 00004111 A3[B46D0000]        <1>       mov   [bsys_msg_eax+4], eax
14451 00004116 31C0                <1>       xor   eax, eax
14452 00004118 C705[68740000]-     <1>         mov     dword [u.base], badsys_msg ; "Invalid System call !"
14453 0000411E [916D0000]          <1>
14454 00004122 8B1D[58740000]      <1>       mov   ebx, [u.fofp]
14455 00004128 8903                <1>       mov   [ebx], eax
14456                              <1>       ;mov  eax, 1 ; inode number of console tty (for user)
14457 0000412A 40                  <1>       inc   eax
14458 0000412B C705[6C740000]3B00- <1>       mov   dword [u.count], BSYS_M_SIZE
14459 00004133 0000                <1>
14460                              <1>             ; writei
14461                              <1>             ; INPUTS ->
14462                              <1>             ;    r1 - inode number
14463                              <1>             ;    u.count - byte count to be written
14464                              <1>             ;    u.base - points to user buffer
14465                              <1>             ;    u.fofp - points to word with current file offset
14466                              <1>             ; OUTPUTS ->
14467                              <1>             ;    u.count - cleared
14468                              <1>             ;    u.nread - accumulates total bytes passed back
14469                              <1>             ;
14470                              <1>             ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
14471 00004135 E882190000          <1>       call  writei
14472                              <1>       ;mov  eax, 1
14473 0000413A EB17                <1>       jmp   sysexit
14474                              <1>
14475                              <1>             ; incb u.bsys / turn on the user's bad-system flag
14476                              <1>             ; mov $3f,u.namep / point u.namep to "core\0\0"
14477                              <1>             ; jsr r0,namei / get the i-number for the core image file
14478                              <1>             ; br 1f / error
14479                              <1>             ; neg r1 / negate the i-number to open the core image file
14480                              <1>                   ; / for writing
14481                              <1>             ; jsr r0,iopen / open the core image file
14482                              <1>             ; jsr r0,itrunc / free all associated blocks
14483                              <1>             ; br 2f
14484                              <1> ;1:
14485                              <1>             ; mov $17,r1 / put i-node mode (17) in r1
14486                              <1>             ; jsr r0,maknod / make an i-node
14487                              <1>             ; mov u.dirbuf,r1 / put i-node number in r1
14488                              <1> ;2:
14489                              <1>             ; mov $core,u.base / move address core to u.base
14490                              <1>             ; mov $ecore-core,u.count / put the byte count in u.count
14491                              <1>             ; mov $u.off,u.fofp / more user offset to u.fofp
14492                              <1>             ; clr u.off / clear user offset
14493                              <1>             ; jsr r0,writei / write out the core image to the user
14494                              <1>             ; mov $user,u.base / pt. u.base to user
14495                              <1>             ; mov $64.,u.count / u.count = 64
14496                              <1>             ; jsr r0,writei / write out all the user parameters
14497                              <1>             ; neg r1 / make i-number positive
14498                              <1>             ; jsr r0,iclose / close the core image file
14499                              <1>             ; br sysexit /
14500                              <1> ;3:
14501                              <1>             ; <core\0\0>
14502                              <1>
14503                              <1> intract: ; / interrupt action
14504                              <1>       ; 14/10/2015
14505                              <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14506                              <1>       ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
14507                              <1>       ;
14508                              <1>       ; Retro UNIX 8086 v1 modification !
14509                              <1>       ; (Process/task switching and quit routine by using
14510                              <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.))
14511                              <1>       ;
14512                              <1>       ; input -> 'u.quit'  (also value of 'u.intr' > 0)
14513                              <1>       ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
14514                              <1>       ;            'intract' will jump to 'sysexit'.
14515                              <1>       ;          Intract will return to the caller
14516                              <1>       ;            if value of 'u.quit' <> FFFFh.
14517                              <1>       ; 14/10/2015
14518 0000413C FB                  <1>       sti
14519                              <1>       ; 07/12/2013
14520 0000413D 66FF05[8E740000]    <1>       inc   word [u.quit]
14521 00004144 7408                <1>       jz    short intrct0 ; FFFFh -> 0
14522 00004146 66FF0D[8E740000]    <1>       dec   word [u.quit]
14523                              <1>       ; 16/04/2015
14524 0000414D C3                  <1>       retn
14525                              <1> intrct0:
14526 0000414E 58                  <1>       pop   eax ; call intract -> retn
14527                              <1>       ;
14528 0000414F 31C0                <1>       xor   eax, eax
14529 00004151 FEC0                <1>       inc   al  ; mov ax, 1
14530                              <1> ;;;
14531                              <1>       ; UNIX v1 original 'intract' routine...
14532                              <1>       ; / interrupt action
14533                              <1>             ;cmp *(sp),$rti / are you in a clock interrupt?
14534                              <1>             ; bne 1f / no, 1f
14535                              <1>             ; cmp (sp)+,(sp)+ / pop clock pointer
14536                              <1>       ; 1: / now in user area
14537                              <1>             ; mov r1,-(sp) / save r1
14538                              <1>             ; mov u.ttyp,r1
14539                              <1>                   ; / pointer to tty buffer in control-to r1
14540                              <1>             ; cmpb 6(r1),$177
14541                              <1>                   ; / is the interrupt char equal to "del"
14542                              <1>             ; beq 1f / yes, 1f
14543                              <1>             ; clrb 6(r1)
14544                              <1>                    ; / no, clear the byte
14545                              <1>                   ; / (must be a quit character)
14546                              <1>             ; mov (sp)+,r1 / restore r1
14547                              <1>             ; clr u.quit / clear quit flag
14548                              <1>             ; bis $20,2(sp)
14549                              <1>                   ; / set trace for quit (sets t bit of
14550                              <1>                   ; / ps-trace trap)
14551                              <1>             ; rti   ;  / return from interrupt
14552                              <1>       ; 1: / interrupt char = del
14553                              <1>             ; clrb 6(r1) / clear the interrupt byte
```

```
14554                                  <1>                     ; / in the buffer
14555                                  <1>             ; mov (sp)+,r1 / restore r1
14556                                  <1>             ; cmp u.intr,$core / should control be
14557                                  <1>                     ; / transferred to loc core?
14558                                  <1>             ; blo 1f
14559                                  <1>             ; jmp *u.intr / user to do rti yes,
14560                                  <1>                     ; / transfer to loc core
14561                                  <1>       ; 1:
14562                                  <1>             ; sys 1 / exit
14563                                  <1>
14564                                  <1> sysexit: ; <terminate process>
14565                                  <1>       ; 01/09/2015
14566                                  <1>       ; 31/08/2015
14567                                  <1>       ; 14/05/2015
14568                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14569                                  <1>       ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
14570                                  <1>       ;
14571                                  <1>       ; 'sysexit' terminates a process. First each file that
14572                                  <1>       ; the process has opened is closed by 'flose'. The process
14573                                  <1>       ; status is then set to unused. The 'p.pid' table is then
14574                                  <1>       ; searched to find children of the dying process. If any of
14575                                  <1>       ; children are zombies (died by not waited for), they are
14576                                  <1>       ; set free. The 'p.pid' table is then searched to find the
14577                                  <1>       ; dying process's parent. When the parent is found, it is
14578                                  <1>       ; checked to see if it is free or it is a zombie. If it is
14579                                  <1>       ; one of these, the dying process just dies. If it is waiting
14580                                  <1>       ; for a child process to die, it notified that it doesn't
14581                                  <1>       ; have to wait anymore by setting it's status from 2 to 1
14582                                  <1>       ; (waiting to active). It is awakened and put on runq by
14583                                  <1>       ; 'putlu'. The dying process enters a zombie state in which
14584                                  <1>       ; it will never be run again but stays around until a 'wait'
14585                                  <1>       ; is completed by it's parent process. If the parent is not
14586                                  <1>       ; found, process just dies. This means 'swap' is called with
14587                                  <1>       ; 'u.uno=0'. What this does is the 'wswap' is not called
14588                                  <1>       ; to write out the process and 'rswap' reads the new process
14589                                  <1>       ; over the one that dies..i.e., the dying process is
14590                                  <1>       ; overwritten and destroyed.
14591                                  <1>       ;
14592                                  <1>       ; Calling sequence:
14593                                  <1>       ;     sysexit or conditional branch.
14594                                  <1>       ; Arguments:
14595                                  <1>       ;     -
14596                                  <1>       ; ..........................................................
14597                                  <1>       ;
14598                                  <1>       ; Retro UNIX 8086 v1 modification:
14599                                  <1>       ;      System call number (=1) is in EAX register.
14600                                  <1>       ;
14601                                  <1>       ;      Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
14602                                  <1>       ;      registers depending of function details.
14603                                  <1>       ;
14604                                  <1>       ; ('swap' procedure is mostly different than original UNIX v1.)
14605                                  <1>       ;
14606                                  <1> ; / terminate process
14607                                  <1>       ; AX = 1
14608 00004153 6648                   <1>       dec   ax ; 0
14609 00004155 66A3[8C740000]         <1>       mov   [u.intr], ax ; 0
14610                                  <1>             ; clr u.intr / clear interrupt control word
14611                                  <1>             ; clr r1 / clear r1
14612                                  <1>       ; AX = 0
14613                                  <1> sysexit_1: ; 1:
14614                                  <1>       ; AX = File descriptor
14615                                  <1>             ; / r1 has file descriptor (index to u.fp list)
14616                                  <1>             ; / Search the whole list
14617 0000415B E8140D0000             <1>       call  fclose
14618                                  <1>             ; jsr r0,fclose / close all files the process opened
14619                                  <1>       ;; ignore error return
14620                                  <1>             ; br .+2 / ignore error return
14621                                  <1>       ;inc   ax
14622 00004160 FEC0                   <1>       inc   al
14623                                  <1>             ; inc r1 / increment file descriptor
14624                                  <1>       ;cmp   ax, 10
14625 00004162 3C0A                   <1>       cmp   al, 10
14626                                  <1>             ; cmp r1,$10. / end of u.fp list?
14627 00004164 72F5                   <1>       jb    short sysexit_1
14628                                  <1>             ; blt 1b / no, go back
14629 00004166 0FB61D[97740000]       <1>       movzx ebx, byte [u.uno] ; 01/09/2015
14630                                  <1>             ; movb u.uno,r1 / yes, move dying process's number to r1
14631 0000416D 88A3[C5710000]         <1>       mov   [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
14632                                  <1>             ; clrb p.stat-1(r1) / free the process
14633                                  <1>       ;shl   bx, 1
14634 00004173 D0E3                   <1>       shl   bl, 1
14635                                  <1>             ; asl r1 / use r1 for index into the below tables
14636 00004175 668B8B[34710000]       <1>       mov   cx, [ebx+p.pid-2]
14637                                  <1>             ; mov p.pid-2(r1),r3 / move dying process's name to r3
14638 0000417C 668B93[54710000]       <1>       mov   dx, [ebx+p.ppid-2]
14639                                  <1>             ; mov p.ppid-2(r1),r4 / move its parents name to r4
14640                                  <1>       ; xor  bx, bx ; 0
14641 00004183 30DB                   <1>       xor   bl, bl ; 0
14642                                  <1>             ; clr r2
14643 00004185 31F6                   <1>       xor   esi, esi ; 0
14644                                  <1>             ; clr r5 / initialize reg
14645                                  <1> sysexit_2: ; 1:
14646                                  <1>              ; / find children of this dying process,
14647                                  <1>             ; / if they are zombies, free them
14648                                  <1>       ;add   bx, 2
14649 00004187 80C302                 <1>       add   bl, 2
14650                                  <1>             ; add $2,r2 / search parent process table
14651                                  <1>                       ; / for dying process's name
14652 0000418A 66398B[54710000]       <1>       cmp   [ebx+p.ppid-2], cx
14653                                  <1>             ; cmp p.ppid-2(r2),r3 / found it?
14654 00004191 7513                   <1>       jne   short sysexit_4
14655                                  <1>             ; bne 3f / no
14656                                  <1>       ;shr   bx, 1
14657 00004193 D0EB                   <1>       shr   bl, 1
14658                                  <1>             ; asr r2 / yes, it is a parent
```

```
14659 00004195 80BB[C5710000]03    <1>       cmp    byte [ebx+p.stat-1], 3 ; SZOMB, 05/02/2014
14660                              <1>              ; cmpb p.stat-1(r2),$3 / is the child of this
14661                              <1>                                     ; / dying process a zombie
14662 0000419C 7506                <1>       jne    short sysexit_3
14663                              <1>              ; bne 2f / no
14664 0000419E 88A3[C5710000]      <1>       mov    [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
14665                              <1>              ; clrb p.stat-1(r2) / yes, free the child process
14666                              <1> sysexit_3: ; 2:
14667                              <1>       ;shr   bx, 1
14668 000041A4 D0E3                <1>       shl    bl, 1
14669                              <1>              ; asl r2
14670                              <1> sysexit_4: ; 3:
14671                              <1>              ; / search the process name table
14672                              <1>              ; / for the dying process's parent
14673 000041A6 663993[34710000]    <1>       cmp    [ebx+p.pid-2], dx ; 17/09/2013
14674                              <1>              ; cmp p.pid-2(r2),r4 / found it?
14675 000041AD 7502                <1>       jne    short sysexit_5
14676                              <1>              ; bne 3f / no
14677 000041AF 89DE                <1>       mov    esi, ebx
14678                              <1>              ; mov r2,r5 / yes, put index to p.pid table (parents
14679                              <1>                          ; / process # x2) in r5
14680                              <1> sysexit_5: ; 3:
14681                              <1>       ;cmp   bx, nproc + nproc
14682 000041B1 80FB20              <1>       cmp    bl, nproc + nproc
14683                              <1>              ; cmp r2,$nproc+nproc / has whole table been searched?
14684 000041B4 72D1                <1>       jb     short sysexit_2
14685                              <1>              ; blt 1b / no, go back
14686                              <1>              ; mov r5,r1 / yes, r1 now has parents process # x2
14687 000041B6 21F6                <1>       and    esi, esi ; r5=r1
14688 000041B8 7431                <1>       jz     short sysexit_6
14689                              <1>              ; beq 2f / no parent has been found.
14690                              <1>                     ; / The process just dies
14691 000041BA 66D1EE              <1>       shr    si, 1
14692                              <1>              ; asr r1 / set up index to p.stat
14693 000041BD 8A86[C5710000]      <1>       mov    al, [esi+p.stat-1]
14694                              <1>              ; movb p.stat-1(r1),r2 / move status of parent to r2
14695 000041C3 20C0                <1>       and    al, al
14696 000041C5 7424                <1>       jz     short sysexit_6
14697                              <1>              ; beq 2f / if its been freed, 2f
14698 000041C7 3C03                <1>       cmp    al, 3
14699                              <1>              ; cmp r2,$3 / is parent a zombie?
14700 000041C9 7420                <1>       je     short sysexit_6
14701                              <1>              ; beq 2f / yes, 2f
14702                              <1>       ; BH = 0
14703 000041CB 8A1D[97740000]      <1>       mov    bl, [u.uno]
14704                              <1>              ; movb u.uno,r3 / move dying process's number to r3
14705 000041D1 C683[C5710000]03    <1>       mov    byte [ebx+p.stat-1], 3  ; SZOMB, 05/02/2014
14706                              <1>              ; movb $3,p.stat-1(r3) / make the process a zombie
14707                              <1>       ; 05/02/2014
14708 000041D8 3C01                <1>       cmp    al, 1 ; SRUN
14709 000041DA 740F                <1>       je     short sysexit_6
14710                              <1>       ;cmp   al, 2
14711                              <1>              ; cmp r2,$2 / is the parent waiting for
14712                              <1>                      ; / this child to die
14713                              <1>       ;jne   short sysexit_6
14714                              <1>              ; bne 2f / yes, notify parent not to wait any more
14715                              <1>       ; 05/02/2014
14716                              <1>       ; p.stat = 2 --> waiting
14717                              <1>       ; p.stat = 4 --> sleeping
14718 000041DC C686[C5710000]01    <1>       mov    byte [esi+p.stat-1], 1 ; SRUN ; 05/02/2014
14719                              <1>       ;dec   byte [esi+p.stat-1]
14720                              <1>              ; decb p.stat-1(r1) / awaken it by putting it (parent)
14721 000041E3 6689F0              <1>       mov    ax, si ; r1  (process number in AL)
14722                              <1>       ;
14723                              <1>       ;mov   ebx, runq + 4
14724                              <1>              ; mov $runq+4,r2 / on the runq
14725 000041E6 E849120000          <1>       call   putlu
14726                              <1>              ; jsr r0, putlu
14727                              <1> sysexit_6: ; 2:
14728                              <1>       ; 31/08/2015
14729                              <1>              ; / the process dies
14730 000041EB C605[97740000]00    <1>       mov    byte [u.uno], 0
14731                              <1>              ; clrb u.uno / put zero as the process number,
14732                              <1>                  ; / so "swap" will
14733 000041F2 E86F110000          <1>       call   swap
14734                              <1>              ; jsr r0,swap / overwrite process with another process
14735                              <1> hlt_sys:
14736                              <1>       ;sti ; 18/01/2014
14737                              <1> hlts0:
14738 000041F7 F4                  <1>       hlt
14739 000041F8 EBFD                <1>       jmp    short hlts0
14740                              <1>              ; 0 / and thereby kill it; halt?
14741                              <1>
14742                              <1>
14743                              <1> syswait: ; < wait for a processs to die >
14744                              <1>       ; 17/09/2015
14745                              <1>       ; 02/09/2015
14746                              <1>       ; 01/09/2015
14747                              <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14748                              <1>       ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
14749                              <1>       ;
14750                              <1>       ; 'syswait' waits for a process die.
14751                              <1>       ; It works in following way:
14752                              <1>       ;    1) From the parent process number, the parent's
14753                              <1>       ;      process name is found. The p.ppid table of parent
14754                              <1>       ;      names is then searched for this process name.
14755                              <1>       ;      If a match occurs, r2 contains child's process
14756                              <1>       ;      number. The child status is checked to see if it is
14757                              <1>       ;      a zombie, i.e; dead but not waited for (p.stat=3)
14758                              <1>       ;      If it is, the child process is freed and it's name
14759                              <1>       ;      is put in (u.r0). A return is then made via 'sysret'.
14760                              <1>       ;      If the child is not a zombie, nothing happens and
14761                              <1>       ;      the search goes on through the p.ppid table until
14762                              <1>       ;      all processes are checked or a zombie is found.
14763                              <1>       ;    2) If no zombies are found, a check is made to see if
```

```
14764                                   <1>       ;     there are any children at all. If there are none,
14765                                   <1>       ;     an error return is made. If there are, the parent's
14766                                   <1>       ;     status is set to 2 (waiting for child to die),
14767                                   <1>       ;     the parent is swapped out, and a branch to 'syswait'
14768                                   <1>       ;     is made to wait on the next process.
14769                                   <1>       ;
14770                                   <1>       ; Calling sequence:
14771                                   <1>       ;     ?
14772                                   <1>       ; Arguments:
14773                                   <1>       ;     -
14774                                   <1>       ; Inputs: -
14775                                   <1>       ; Outputs: if zombie found, it's name put in u.r0.
14776                                   <1>       ; ..........................................................
14777                                   <1>       ;
14778                                   <1>
14779                                   <1> ; / wait for a process to die
14780                                   <1>
14781                                   <1> syswait_0:
14782 000041FA 0FB61D[97740000]         <1>       movzx  ebx, byte [u.uno] ; 01/09/2015
14783                                   <1>              ; movb u.uno,r1 / put parents process number in r1
14784 00004201 D0E3                     <1>       shl    bl, 1
14785                                   <1>       ;shl   bx, 1
14786                                   <1>              ; asl r1 / x2 to get index into p.pid table
14787 00004203 668B83[34710000]         <1>       mov    ax, [ebx+p.pid-2]
14788                                   <1>              ; mov p.pid-2(r1),r1 / get the name of this process
14789 0000420A 31F6                     <1>       xor    esi, esi
14790                                   <1>              ; clr r2
14791 0000420C 31C9                     <1>       xor    ecx, ecx ; 30/10/2013
14792                                   <1>       ;xor   cl, cl
14793                                   <1>              ; clr r3 / initialize reg 3
14794                                   <1> syswait_1: ; 1:
14795 0000420E 6683C602                 <1>       add    si, 2
14796                                   <1>              ; add $2,r2 / use r2 for index into p.ppid table
14797                                   <1>                      ; / search table of parent processes
14798                                   <1>                      ; / for this process name
14799 00004212 663B86[54710000]         <1>       cmp    ax, [esi+p.ppid-2]
14800                                   <1>              ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
14801                                   <1>                                   ; / process number
14802 00004219 7535                     <1>       jne    short syswait_3
14803                                   <1>              ;bne 3f / branch if no match of parent process name
14804                                   <1>       ;inc   cx
14805 0000421B FEC1                     <1>       inc    cl
14806                                   <1>              ;inc r3 / yes, a match, r3 indicates number of children
14807 0000421D 66D1EE                   <1>       shr    si, 1
14808                                   <1>              ; asr r2 / r2/2 to get index to p.stat table
14809                                   <1>       ; The possible states ('p.stat' values) of a process are:
14810                                   <1>       ;     0 = free or unused
14811                                   <1>       ;     1 = active
14812                                   <1>       ;     2 = waiting for a child process to die
14813                                   <1>       ;     3 = terminated, but not yet waited for (zombie).
14814 00004220 80BE[C5710000]03         <1>       cmp    byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
14815                                   <1>              ; cmpb p.stat-1(r2),$3 / is the child process a zombie?
14816 00004227 7524                     <1>       jne    short syswait_2
14817                                   <1>              ; bne 2f / no, skip it
14818 00004229 88BE[C5710000]           <1>       mov    [esi+p.stat-1], bh ; 0
14819                                   <1>              ; clrb p.stat-1(r2) / yes, free it
14820 0000422F 66D1E6                   <1>       shl    si, 1
14821                                   <1>              ; asl r2 / r2x2 to get index into p.pid table
14822 00004232 0FB786[34710000]         <1>       movzx  eax, word [esi+p.pid-2]
14823 00004239 A3[48740000]             <1>       mov    [u.r0], eax
14824                                   <1>              ; mov p.pid-2(r2),*u.r0
14825                                   <1>                           ; / put childs process name in (u.r0)
14826                                   <1>       ;
14827                                   <1>       ; Retro UNIX 386 v1 modification ! (17/09/2015)
14828                                   <1>       ;
14829                                   <1>       ; Parent process ID -p.ppid- field (of the child process)
14830                                   <1>       ; must be cleared in order to prevent infinitive 'syswait'
14831                                   <1>       ; system call loop from the application/program if it calls
14832                                   <1>       ; 'syswait' again (mistakenly) while there is not a zombie
14833                                   <1>       ; or running child process to wait. ('forktest.s', 17/09/2015)
14834                                   <1>       ;
14835                                   <1>       ; Note: syswait will return with error if there is not a
14836                                   <1>       ;       zombie or running process to wait.
14837                                   <1>       ;
14838 0000423E 6629C0                   <1>       sub    ax, ax
14839 00004241 668986[54710000]         <1>       mov    [esi+p.ppid-2], ax ; 0 ; 17/09/2015
14840 00004248 E910FEFFFF               <1>       jmp    sysret0 ; ax = 0
14841                                   <1>       ;
14842                                   <1>       ;jmp   sysret
14843                                   <1>              ; br sysret1 / return cause child is dead
14844                                   <1> syswait_2: ; 2:
14845 0000424D 66D1E6                   <1>       shl    si, 1
14846                                   <1>              ; asl r2 / r2x2 to get index into p.ppid table
14847                                   <1> syswait_3: ; 3:
14848 00004250 6683FE20                 <1>       cmp    si, nproc+nproc
14849                                   <1>              ; cmp r2,$nproc+nproc / have all processes been checked?
14850 00004254 72B8                     <1>       jb     short syswait_1
14851                                   <1>              ; blt 1b / no, continue search
14852                                   <1>       ;and   cx, cx
14853 00004256 20C9                     <1>       and    cl, cl
14854                                   <1>              ; tst r3 / one gets here if there are no children
14855                                   <1>                     ; / or children that are still active
14856                                   <1>       ; 30/10/2013
14857 00004258 750B                     <1>       jnz    short syswait_4
14858                                   <1>       ;jz    error
14859                                   <1>              ; beq error1 / there are no children, error
14860 0000425A 890D[48740000]           <1>       mov    [u.r0], ecx ; 0
14861 00004260 E9D5FDFFFF               <1>       jmp    error
14862                                   <1> syswait_4:
14863 00004265 8A1D[97740000]           <1>       mov    bl, [u.uno]
14864                                   <1>              ; movb u.uno,r1 / there are children so put
14865                                   <1>                           ; / parent process number in r1
14866 0000426B FE83[C5710000]           <1>       inc    byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014
14867                                   <1>              ; incb p.stat-1(r1) / it is waiting for
14868                                   <1>                                 ; / other children to die
```

```
14869                                  <1>       ; 04/11/2013
14870 00004271 E8F0100000              <1>       call   swap
14871                                  <1>              ; jsr r0,swap / swap it out, because it's waiting
14872 00004276 EB82                    <1>       jmp    syswait_0
14873                                  <1>              ; br syswait / wait on next process
14874                                  <1>
14875                                  <1> sysfork: ; < create a new process >
14876                                  <1>       ; 18/09/2015
14877                                  <1>       ; 04/09/2015
14878                                  <1>       ; 02/09/2015
14879                                  <1>       ; 01/09/2015
14880                                  <1>       ; 28/08/2015
14881                                  <1>       ; 14/05/2015
14882                                  <1>       ; 10/05/2015
14883                                  <1>       ; 09/05/2015
14884                                  <1>       ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
14885                                  <1>       ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
14886                                  <1>       ;
14887                                  <1>       ; 'sysfork' creates a new process. This process is referred
14888                                  <1>       ; to as the child process. This new process core image is
14889                                  <1>       ; a copy of that of the caller of 'sysfork'. The only
14890                                  <1>       ; distinction is the return location and the fact that (u.r0)
14891                                  <1>       ; in the old process (parent) contains the process id (p.pid)
14892                                  <1>       ; of the new process (child). This id is used by 'syswait'.
14893                                  <1>       ; 'sysfork' works in the following manner:
14894                                  <1>       ;    1) The process status table (p.stat) is searched to find
14895                                  <1>       ;      a process number that is unused. If none are found
14896                                  <1>       ;      an error occurs.
14897                                  <1>       ;    2) when one is found, it becomes the child process number
14898                                  <1>       ;      and it's status (p.stat) is set to active.
14899                                  <1>       ;    3) If the parent had a control tty, the interrupt
14900                                  <1>       ;      character in that tty buffer is cleared.
14901                                  <1>       ;    4) The child process is put on the lowest priority run
14902                                  <1>       ;      queue via 'putlu'.
14903                                  <1>       ;    5) A new process name is gotten from 'mpid' (actually
14904                                  <1>       ;      it is a unique number) and is put in the child's unique
14905                                  <1>       ;      identifier; process id (p.pid).
14906                                  <1>       ;    6) The process name of the parent is then obtained and
14907                                  <1>       ;      placed in the unique identifier of the parent process
14908                                  <1>       ;      name is then put in 'u.r0'.
14909                                  <1>       ;    7) The child process is then written out on disk by
14910                                  <1>       ;      'wswap',i.e., the parent process is copied onto disk
14911                                  <1>       ;      and the child is born. (The child process is written
14912                                  <1>       ;      out on disk/drum with 'u.uno' being the child process
14913                                  <1>       ;      number.)
14914                                  <1>       ;    8) The parent process number is then restored to 'u.uno'.
14915                                  <1>       ;    9) The child process name is put in 'u.r0'.
14916                                  <1>       ;   10) The pc on the stack sp + 18 is incremented by 2 to
14917                                  <1>       ;      create the return address for the parent process.
14918                                  <1>       ;   11) The 'u.fp' list as then searched to see what files
14919                                  <1>       ;      the parent has opened. For each file the parent has
14920                                  <1>       ;      opened, the corresponding 'fsp' entry must be updated
14921                                  <1>       ;      to indicate that the child process also has opened
14922                                  <1>       ;      the file. A branch to 'sysret' is then made.
14923                                  <1>       ;
14924                                  <1>       ; Calling sequence:
14925                                  <1>       ;      from shell ?
14926                                  <1>       ; Arguments:
14927                                  <1>       ;      -
14928                                  <1>       ; Inputs: -
14929                                  <1>       ; Outputs: *u.r0 - child process name
14930                                  <1>       ; ............................................................
14931                                  <1>       ;
14932                                  <1>       ; Retro UNIX 8086 v1 modification:
14933                                  <1>       ;      AX = r0 = PID (>0) (at the return of 'sysfork')
14934                                  <1>       ;      = process id of child a parent process returns
14935                                  <1>       ;      = process id of parent when a child process returns
14936                                  <1>       ;
14937                                  <1>       ;        In original UNIX v1, sysfork is called and returns as
14938                                  <1>       ;      in following manner: (with an example: c library, fork)
14939                                  <1>       ;
14940                                  <1>       ;      1:
14941                                  <1>       ;            sys    fork
14942                                  <1>       ;                   br 1f  / child process returns here
14943                                  <1>       ;            bes    2f     / parent process returns here
14944                                  <1>       ;            / pid of new process in r0
14945                                  <1>       ;            rts    pc
14946                                  <1>       ;      2: / parent process condionally branches here
14947                                  <1>       ;            mov    $-1,r0 / pid = -1 means error return
14948                                  <1>       ;            rts    pc
14949                                  <1>       ;
14950                                  <1>       ;      1: / child process brances here
14951                                  <1>       ;            clr    r0   / pid = 0 in child process
14952                                  <1>       ;            rts    pc
14953                                  <1>       ;
14954                                  <1>       ;      In UNIX v7x86 (386) by Robert Nordier (1999)
14955                                  <1>       ;            // pid = fork();
14956                                  <1>       ;            //
14957                                  <1>       ;            // pid == 0 in child process;
14958                                  <1>       ;            // pid == -1 means error return
14959                                  <1>       ;            // in child,
14960                                  <1>       ;            //     parents id is in par_uid if needed
14961                                  <1>       ;
14962                                  <1>       ;            _fork:
14963                                  <1>       ;                   mov    $.fork,eax
14964                                  <1>       ;                   int    $0x30
14965                                  <1>       ;                   jmp    1f
14966                                  <1>       ;                   jnc    2f
14967                                  <1>       ;                   jmp    cerror
14968                                  <1>       ;            1:
14969                                  <1>       ;                   mov    eax,_par_uid
14970                                  <1>       ;                   xor    eax,eax
14971                                  <1>       ;            2:
14972                                  <1>       ;                   ret
```

```
14973                           <1>       ;
14974                           <1>       ;     In Retro UNIX 8086 v1,
14975                           <1>       ;     'sysfork' returns in following manner:
14976                           <1>       ;
14977                           <1>       ;           mov   ax, sys_fork
14978                           <1>       ;           mov   bx, offset @f ; routine for child
14979                           <1>       ;           int   20h
14980                           <1>       ;           jc    error
14981                           <1>       ;
14982                           <1>       ;     ; Routine for parent process here (just after 'jc')
14983                           <1>       ;           mov   word ptr [pid_of_child], ax
14984                           <1>       ;           jmp   next_routine_for_parent
14985                           <1>       ;
14986                           <1>       ;     @@: ; routine for child process here
14987                           <1>       ;           ....
14988                           <1>       ;     NOTE: 'sysfork' returns to specified offset
14989                           <1>       ;           for child process by using BX input.
14990                           <1>       ;           (at first, parent process will return then
14991                           <1>       ;           child process will return -after swapped in-
14992                           <1>       ;           'syswait' is needed in parent process
14993                           <1>       ;           if return from child process will be waited for.)
14994                           <1>       ;
14995                           <1>
14996                           <1> ; / create a new process
14997                           <1>       ; EBX = return address for child process
14998                           <1>           ; (Retro UNIX 8086 v1 modification !)
14999 00004278 31F6              <1>       xor   esi, esi
15000                           <1>             ; clr r1
15001                           <1> sysfork_1: ; 1: / search p.stat table for unused process number
15002 0000427A 46                <1>       inc   esi
15003                           <1>             ; inc r1
15004 0000427B 80BE[C5710000]00  <1>       cmp   byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
15005                           <1>             ; tstb p.stat-1(r1) / is process active, unused, dead
15006 00004282 760B              <1>       jna   short sysfork_2
15007                           <1>             ; beq 1f / it's unused so branch
15008 00004284 6683FE10          <1>       cmp   si, nproc
15009                           <1>             ; cmp r1,$nproc / all processes checked
15010 00004288 72F0              <1>       jb    short sysfork_1
15011                           <1>             ; blt 1b / no, branch back
15012                           <1>       ;
15013                           <1>       ; Retro UNIX 8086 v1. modification:
15014                           <1>       ;     Parent process returns from 'sysfork' to address
15015                           <1>       ;     which is just after 'sysfork' system call in parent
15016                           <1>       ;     process. Child process returns to address which is put
15017                           <1>       ;     in BX register by parent process for 'sysfork'.
15018                           <1>       ;
15019                           <1>             ;add $2,18.(sp) / add 2 to pc when trap occured, points
15020                           <1>                           ; / to old process return
15021                           <1>             ; br error1 / no room for a new process
15022 0000428A E9ABFDFFFF        <1>       jmp   error
15023                           <1> sysfork_2: ; 1:
15024 0000428F E861EDFFFF        <1>       call  allocate_page
15025 00004294 0F82A0FDFFFF      <1>       jc    error
15026 0000429A 50                <1>       push  eax   ; UPAGE (user structure page) address
15027                           <1>       ; Retro UNIX 386 v1 modification!
15028 0000429B E85EEFFFFF        <1>       call  duplicate_page_dir
15029                           <1>             ; EAX = New page directory
15030 000042A0 730B              <1>       jnc   short sysfork_3
15031 000042A2 58                <1>       pop   eax   ; UPAGE (user structure page) address
15032 000042A3 E825EFFFFF        <1>       call  deallocate_page
15033 000042A8 E98DFDFFFF        <1>       jmp   error
15034                           <1> sysfork_3:
15035                           <1>       ; Retro UNIX 386 v1 modification !
15036 000042AD 56                <1>       push  esi
15037 000042AE E82B110000        <1>       call  wswap ; save current user (u) structure, user registers
15038                           <1>                   ; and interrupt return components (for IRET)
15039 000042B3 8705[A1740000]    <1>       xchg  eax, [u.pgdir] ; page directory of the child process
15040 000042B9 A3[A5740000]      <1>       mov   [u.ppgdir], eax ; page directory of the parent process
15041 000042BE 5E                <1>       pop   esi
15042 000042BF 58                <1>       pop   eax   ; UPAGE (user structure page) address
15043                           <1>             ; [u.usp] = esp
15044 000042C0 89F7              <1>       mov   edi, esi
15045 000042C2 66C1E702          <1>       shl   di, 2
15046 000042C6 8987[D2710000]    <1>       mov   [edi+p.upage-4], eax ; memory page for 'user' struct
15047 000042CC A3[98740000]      <1>       mov   [u.upage], eax ; memory page for 'user' struct (child)
15048                           <1>       ; 28/08/2015
15049 000042D1 0FB605[97740000]  <1>       movzx eax, byte [u.uno] ; parent process number
15050                           <1>             ; movb u.uno,-(sp) / save parent process number
15051 000042D8 89C7              <1>       mov   edi, eax
15052 000042DA 50                <1>         push eax ; **
15053 000042DB 8A87[95710000]    <1>       mov    al, [edi+p.ttyc-1] ; console tty (parent)
15054                           <1>       ; 18/09/2015
15055                           <1>       ;mov    [esi+p.ttyc-1], al ; set child's console tty
15056                           <1>       ;mov    [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
15057 000042E1 668986[95710000]  <1>       mov    [esi+p.ttyc-1], ax ; al - set child's console tty
15058                           <1>                                 ; ah - reset child's wait channel
15059 000042E8 89F0              <1>       mov   eax, esi
15060 000042EA A2[97740000]      <1>       mov   [u.uno], al ; child process number
15061                           <1>             ;movb r1,u.uno / set child process number to r1
15062 000042EF FE86[C5710000]    <1>         inc    byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
15063                           <1>             ; incb p.stat-1(r1) / set p.stat entry for child
15064                           <1>                           ; / process to active status
15065                           <1>             ; mov u.ttyp,r2 / put pointer to parent process'
15066                           <1>                           ; / control tty buffer in r2
15067                           <1>                ; beq 2f / branch, if no such tty assigned
15068                           <1>             ; clrb 6(r2) / clear interrupt character in tty buffer
15069                           <1>       ; 2:
15070 000042F5 53                <1>       push  ebx  ; * return address for the child process
15071                           <1>                  ; * Retro UNIX 8086 v1 feature only !
15072                           <1>       ; (Retro UNIX 8086 v1 modification!)
15073                           <1>             ; mov $runq+4,r2
15074 000042F6 E839110000        <1>       call  putlu
15075                           <1>             ; jsr r0,putlu / put child process on lowest priority
15076                           <1>                          ; / run queue
15077 000042FB 66D1E6            <1>       shl   si, 1
```

```
15078                                  <1>             ; asl r1 / multiply r1 by 2 to get index
15079                                  <1>                     ; / into p.pid table
15080 000042FE 66FF05[36740000]        <1>       inc   word [mpid]
15081                                  <1>             ; inc mpid / increment m.pid; get a new process name
15082 00004305 66A1[36740000]          <1>       mov   ax, [mpid]
15083 0000430B 668986[34710000]        <1>       mov   [esi+p.pid-2], ax
15084                                  <1>             ;mov mpid,p.pid-2(r1) / put new process name
15085                                  <1>                                   ; / in child process' name slot
15086 00004312 5A                      <1>       pop   edx  ; * return address for the child process
15087                                  <1>                  ; * Retro UNIX 8086 v1 feature only !
15088 00004313 5B                      <1>       pop   ebx  ; **
15089                                  <1>       ;mov  ebx, [esp] ; ** parent process number
15090                                  <1>             ; movb (sp),r2 / put parent process number in r2
15091 00004314 66D1E3                  <1>       shl   bx, 1
15092                                  <1>             ;asl r2 / multiply by 2 to get index into below tables
15093                                  <1>       ;movzx eax, word [ebx+p.pid-2]
15094 00004317 668B83[34710000]        <1>       mov   ax, [ebx+p.pid-2]
15095                                  <1>             ; mov p.pid-2(r2),r2 / get process name of parent
15096                                  <1>                                  ; / process
15097 0000431E 668986[54710000]        <1>       mov   [esi+p.ppid-2], ax
15098                                  <1>             ; mov r2,p.ppid-2(r1) / put parent process name
15099                                  <1>                      ; / in parent process slot for child
15100 00004325 A3[48740000]            <1>       mov   [u.r0], eax
15101                                  <1>             ; mov r2,*u.r0 / put parent process name on stack
15102                                  <1>                            ; / at location where r0 was saved
15103 0000432A 8B2D[40740000]          <1>       mov   ebp, [u.sp] ; points to return address (EIP for IRET)
15104 00004330 895500                  <1>       mov   [ebp], edx ; *, CS:EIP -> EIP
15105                                  <1>                        ; * return address for the child process
15106                                  <1>             ; mov $sysret1,-(sp) /
15107                                  <1>             ; mov sp,u.usp / contents of sp at the time when
15108                                  <1>                            ; / user is swapped out
15109                                  <1>             ; mov $sstack,sp / point sp to swapping stack space
15110                                  <1>       ; 04/09/2015 - 01/09/2015
15111                                  <1>       ; [u.usp] = esp
15112 00004333 68[5A400000]            <1>       push  sysret ; ***
15113 00004338 8925[44740000]          <1>       mov   [u.usp], esp ; points to 'sysret' address (***)
15114                                  <1>                          ; (for child process)
15115 0000433E 31C0                    <1>       xor   eax, eax
15116 00004340 66A3[78740000]          <1>       mov   [u.ttyp], ax ; 0
15117                                  <1>       ;
15118 00004346 E893100000              <1>       call  wswap ; Retro UNIX 8086 v1 modification !
15119                                  <1>             ;jsr r0,wswap / put child process out on drum
15120                                  <1>             ;jsr r0,unpack / unpack user stack
15121                                  <1>             ;mov u.usp,sp / restore user stack pointer
15122                                  <1>             ; tst (sp)+ / bump stack pointer
15123                                  <1>       ; Retro UNIX 386 v1 modification !
15124 0000434B 58                      <1>       pop   eax ; ***
15125 0000434C 66D1E3                  <1>       shl   bx, 1
15126 0000434F 8B83[D2710000]          <1>       mov    eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
15127 00004355 E8AD100000              <1>       call  rswap ; restore parent process 'u' structure,
15128                                  <1>                   ; registers and return address (for IRET)
15129                                  <1>             ;movb (sp)+,u.uno / put parent process number in u.uno
15130 0000435A 0FB705[36740000]        <1>         movzx   eax, word [mpid]
15131 00004361 A3[48740000]            <1>       mov   [u.r0], eax
15132                                  <1>             ; mov mpid,*u.r0 / put child process name on stack
15133                                  <1>                              ; / where r0 was saved
15134                                  <1>             ; add $2,18.(sp) / add 2 to pc on stack; gives parent
15135                                  <1>                                ; / process return
15136                                  <1>       ;xor  ebx, ebx
15137 00004366 31F6                    <1>       xor    esi, esi
15138                                  <1>             ;clr r1
15139                                  <1> sysfork_4: ; 1: / search u.fp list to find the files
15140                                  <1>            ; / opened by the parent process
15141                                  <1>       ; 01/09/2015
15142                                  <1>       ;xor  bh, bh
15143                                  <1>       ;mov  bl, [esi+u.fp]
15144 00004368 8A86[4E740000]          <1>       mov   al, [esi+u.fp]
15145                                  <1>             ; movb u.fp(r1),r2 / get an open file for this process
15146                                  <1>         ;or      bl, bl
15147 0000436E 08C0                    <1>       or    al, al
15148 00004370 740D                    <1>       jz    short sysfork_5
15149                                  <1>             ; beq 2f / file has not been opened by parent,
15150                                  <1>                    ; / so branch
15151 00004372 B40A                    <1>       mov   ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
15152 00004374 F6E4                    <1>       mul   ah
15153                                  <1>       ;movzx ebx, ax
15154 00004376 6689C3                  <1>       mov   bx, ax
15155                                  <1>       ;shl     bx, 3
15156                                  <1>             ; asl r2 / multiply by 8
15157                                  <1>                   ; asl r2 / to get index into fsp table
15158                                  <1>                   ; asl r2
15159 00004379 FE83[14720000]          <1>       inc    byte [ebx+fsp-2]
15160                                  <1>             ; incb fsp-2(r2) / increment number of processes
15161                                  <1>                              ; / using file, because child will now be
15162                                  <1>                              ; / using this file
15163                                  <1> sysfork_5: ; 2:
15164 0000437F 46                      <1>         inc     esi
15165                                  <1>             ; inc r1 / get next open file
15166 00004380 6683FE0A                <1>         cmp     si, 10
15167                                  <1>             ; cmp r1,$10. / 10. files is the maximum number which
15168                                  <1>                      ; / can be opened
15169 00004384 72E2                    <1>       jb    short sysfork_4
15170                                  <1>             ; blt 1b / check next entry
15171 00004386 E9CFFCFFFF              <1>       jmp   sysret
15172                                  <1>             ; br sysret1
15173                                  <1>
15174                                  <1> sysread: ; < read from file >
15175                                  <1>       ; 13/05/2015
15176                                  <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15177                                  <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
15178                                  <1>       ;
15179                                  <1>       ; 'sysread' is given a buffer to read into and the number of
15180                                  <1>       ; characters to be read. If finds the file from the file
15181                                  <1>       ; descriptor located in *u.r0 (r0). This file descriptor
15182                                  <1>       ; is returned from a successful open call (sysopen).
```

```
15183                                  <1>       ; The i-number of file is obtained via 'rw1' and the data
15184                                  <1>       ; is read into core via 'readi'.
15185                                  <1>       ;
15186                                  <1>       ; Calling sequence:
15187                                  <1>       ;     sysread; buffer; nchars
15188                                  <1>       ; Arguments:
15189                                  <1>       ;     buffer - location of contiguous bytes where
15190                                  <1>       ;              input will be placed.
15191                                  <1>       ;     nchars - number of bytes or characters to be read.
15192                                  <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
15193                                  <1>       ; Outputs: *u.r0 - number of bytes read.
15194                                  <1>       ; ..........................................................
15195                                  <1>       ;
15196                                  <1>       ; Retro UNIX 8086 v1 modification:
15197                                  <1>       ;      'sysread' system call has three arguments; so,
15198                                  <1>       ;     * 1st argument, file descriptor is in BX register
15199                                  <1>       ;     * 2nd argument, buffer address/offset in CX register
15200                                  <1>       ;     * 3rd argument, number of bytes is in DX register
15201                                  <1>       ;
15202                                  <1>       ;     AX register (will be restored via 'u.r0') will return
15203                                  <1>       ;     to the user with number of bytes read.
15204                                  <1>       ;
15205 0000438B E83D000000              <1>       call   rw1
15206 00004390 0F82A4FCFFFF            <1>       jc     error ; 13/05/2015, ax < 1
15207                                  <1>              ; jsr r0,rw1 / get i-number of file to be read into r1
15208 00004396 F6C480                  <1>       test   ah, 80h
15209                                  <1>              ; tst r1 / negative i-number?
15210 00004399 0F859BFCFFFF            <1>       jnz    error
15211                                  <1>              ; ble error1 / yes, error 1 to read
15212                                  <1>                       ; / it should be positive
15213 0000439F E822150000              <1>       call   readi
15214                                  <1>              ; jsr r0,readi / read data into core
15215 000043A4 EB18                    <1>       jmp    short rw0
15216                                  <1>              ; br 1f
15217                                  <1> syswrite: ; < write to file >
15218                                  <1>       ; 13/05/2015
15219                                  <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15220                                  <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
15221                                  <1>       ;
15222                                  <1>       ; 'syswrite' is given a buffer to write onto an output file
15223                                  <1>       ; and the number of characters to write. If finds the file
15224                                  <1>       ; from the file descriptor located in *u.r0 (r0). This file
15225                                  <1>       ; descriptor is returned from a successful open or create call
15226                                  <1>       ; (sysopen or syscreat). The i-number of file is obtained via
15227                                  <1>       ; 'rw1' and buffer is written on the output file via 'write'.
15228                                  <1>       ;
15229                                  <1>       ; Calling sequence:
15230                                  <1>       ;     syswrite; buffer; nchars
15231                                  <1>       ; Arguments:
15232                                  <1>       ;     buffer - location of contiguous bytes to be writtten.
15233                                  <1>       ;     nchars - number of characters to be written.
15234                                  <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
15235                                  <1>       ; Outputs: *u.r0 - number of bytes written.
15236                                  <1>       ; ..........................................................
15237                                  <1>       ;
15238                                  <1>       ; Retro UNIX 8086 v1 modification:
15239                                  <1>       ;      'syswrite' system call has three arguments; so,
15240                                  <1>       ;     * 1st argument, file descriptor is in BX register
15241                                  <1>       ;     * 2nd argument, buffer address/offset in CX register
15242                                  <1>       ;     * 3rd argument, number of bytes is in DX register
15243                                  <1>       ;
15244                                  <1>       ;     AX register (will be restored via 'u.r0') will return
15245                                  <1>       ;     to the user with number of bytes written.
15246                                  <1>       ;
15247 000043A6 E822000000              <1>       call   rw1
15248 000043AB 0F8289FCFFFF            <1>       jc     error ; 13/05/2015, ax < 1
15249                                  <1>              ; jsr r0,rw1 / get i-number in r1 of file to write
15250 000043B1 F6C480                  <1>         test ah, 80h
15251                                  <1>              ; tst r1 / positive i-number ?
15252 000043B4 744E                    <1>         jz   short rw3 ; 13/05/2015
15253                                  <1>       ;jz    error
15254                                  <1>              ; bge error1 / yes, error 1
15255                                  <1>                       ; / negative i-number means write
15256 000043B6 66F7D8                  <1>         neg  ax
15257                                  <1>              ; neg r1 / make it positive
15258 000043B9 E8FE160000              <1>       call   writei
15259                                  <1>              ; jsr r0,writei / write data
15260                                  <1> rw0: ; 1:
15261 000043BE A1[70740000]            <1>         mov  eax, [u.nread]
15262 000043C3 A3[48740000]            <1>       mov    [u.r0], eax
15263                                  <1>              ; mov u.nread,*u.r0 / put no. of bytes transferred
15264                                  <1>                                  ; / into (u.r0)
15265 000043C8 E98DFCFFFF              <1>       jmp    sysret
15266                                  <1>              ; br sysret1
15267                                  <1> rw1:
15268                                  <1>       ; 14/05/2015
15269                                  <1>       ; 13/05/2015
15270                                  <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15271                                  <1>       ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
15272                                  <1>       ; System call registers: bx, cx, dx (through 'sysenter')
15273                                  <1>       ;
15274                                  <1>       ;mov   [u.base], ecx       ; buffer address/offset
15275                                  <1>                                  ;(in the user's virtual memory space)
15276                                  <1>       ;mov   [u.count], edx
15277                                  <1>              ; jsr r0,arg; u.base / get buffer pointer
15278                                  <1>              ; jsr r0,arg; u.count / get no. of characters
15279                                  <1>       ;;mov  eax, ebx ; file descriptor
15280                                  <1>              ; mov *u.r0,r1 / put file descriptor
15281                                  <1>                                  ; / (index to u.fp table) in r1
15282                                  <1>       ; 13/05/2015
15283 000043CD C705[48740000]0000-     <1>       mov    dword [u.r0], 0 ; r/w transfer count = 0 (reset)
15284 000043D5 0000                    <1>
15285                                  <1>       ;
15286                                  <1>       ;; call      getf
15287                                  <1>         ; eBX = File descriptor
```

```
15288 000043D7 E8E30A0000          <1>       call   getf1 ; calling point in 'getf' from 'rw1'
15289                              <1>              ; jsr r0,getf / get i-number of the file in r1
15290                              <1>       ; AX = I-number of the file ; negative i-number means write
15291                              <1>       ; 13/05/2015
15292 000043DC 6683F801           <1>       cmp    ax, 1
15293 000043E0 7217                <1>       jb     short rw2
15294                              <1>       ;
15295 000043E2 890D[68740000]      <1>       mov    [u.base], ecx       ; buffer address/offset
15296                              <1>                                  ;(in the user's virtual memory space)
15297 000043E8 8915[6C740000]      <1>       mov    [u.count], edx
15298                              <1>       ; 14/05/2015
15299 000043EE C705[9D740000]0000- <1>         mov     dword [u.error], 0 ; reset the last error code
15300 000043F6 0000                <1>
15301 000043F8 C3                  <1>       retn
15302                              <1>              ; rts r0
15303                              <1> rw2:
15304                              <1>       ; 13/05/2015
15305 000043F9 C705[9D740000]0A00- <1>       mov    dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
15306 00004401 0000                <1>
15307 00004403 C3                  <1>       retn
15308                              <1> rw3:
15309                              <1>       ; 13/05/2015
15310 00004404 C705[9D740000]0B00- <1>       mov    dword [u.error], ERR_FILE_ACCESS ; permission denied !
15311 0000440C 0000                <1>
15312 0000440E F9                  <1>       stc
15313 0000440F C3                  <1>       retn
15314                              <1>
15315                              <1> sysopen: ;<open file>
15316                              <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15317                              <1>       ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
15318                              <1>       ;
15319                              <1>       ; 'sysopen' opens a file in following manner:
15320                              <1>       ;    1) The second argument in a sysopen says whether to
15321                              <1>       ;      open the file ro read (0) or write (>0).
15322                              <1>       ;    2) I-node of the particular file is obtained via 'namei'.
15323                              <1>       ;    3) The file is opened by 'iopen'.
15324                              <1>       ;    4) Next housekeeping is performed on the fsp table
15325                              <1>       ;      and the user's open file list - u.fp.
15326                              <1>       ;      a) u.fp and fsp are scanned for the next available slot.
15327                              <1>       ;      b) An entry for the file is created in the fsp table.
15328                              <1>       ;      c) The number of this entry is put on u.fp list.
15329                              <1>       ;      d) The file descriptor index to u.fp list is pointed
15330                              <1>       ;         to by u.r0.
15331                              <1>       ;
15332                              <1>       ; Calling sequence:
15333                              <1>       ;      sysopen; name; mode
15334                              <1>       ; Arguments:
15335                              <1>       ;      name - file name or path name
15336                              <1>       ;      mode - 0 to open for reading
15337                              <1>       ;             1 to open for writing
15338                              <1>       ; Inputs: (arguments)
15339                              <1>       ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
15340                              <1>       ;              is put into r0's location on the stack.
15341                              <1>       ; ..........................................................
15342                              <1>       ;
15343                              <1>       ; Retro UNIX 8086 v1 modification:
15344                              <1>       ;       'sysopen' system call has two arguments; so,
15345                              <1>       ;      * 1st argument, name is pointed to by BX register
15346                              <1>       ;      * 2nd argument, mode is in CX register
15347                              <1>       ;
15348                              <1>       ;      AX register (will be restored via 'u.r0') will return
15349                              <1>       ;      to the user with the file descriptor/number
15350                              <1>       ;      (index to u.fp list).
15351                              <1>       ;
15352                              <1>       ;call  arg2
15353                              <1>       ; * name - 'u.namep' points to address of file/path name
15354                              <1>       ;          in the user's program segment ('u.segmnt')
15355                              <1>       ;          with offset in BX register (as sysopen argument 1).
15356                              <1>       ; * mode - sysopen argument 2 is in CX register
15357                              <1>       ;          which is on top of stack.
15358                              <1>       ;
15359                              <1>       ; jsr r0,arg2 / get sys args into u.namep and on stack
15360                              <1>       ;
15361                              <1>              ; system call registers: ebx, ecx (through 'sysenter')
15362                              <1>
15363 00004410 891D[60740000]      <1>       mov    [u.namep], ebx
15364 00004416 6651                <1>       push   cx
15365 00004418 E8D90A0000          <1>       call   namei
15366                              <1>              ; jsr r0,namei / i-number of file in r1
15367                              <1>       ;and   ax, ax
15368                              <1>       ;jz    error ; File not found
15369 0000441D 723B                <1>       jc     short fnotfound ; 14/05/2015
15370                              <1>       ;jc    error ; 27/05/2013
15371                              <1>              ; br  error2 / file not found
15372 0000441F 665A                <1>       pop    dx ; mode
15373 00004421 6652                <1>       push   dx
15374                              <1>       ;or    dx, dx
15375 00004423 08D2                <1>       or     dl, dl
15376                              <1>              ; tst (sp) / is mode = 0 (2nd arg of call;
15377                              <1>                       ; / 0 means, open for read)
15378 00004425 7403                <1>       jz     short sysopen_0
15379                              <1>              ; beq 1f / yes, leave i-number positive
15380                              <1> syscreat_0: ; 27/12/2015
15381 00004427 66F7D8              <1>       neg    ax
15382                              <1>              ; neg r1 / open for writing so make i-number negative
15383                              <1> sysopen_0: ;1:
15384 0000442A E8361A0000          <1>       call   iopen
15385                              <1>              ;jsr r0,iopen / open file whose i-number is in r1
15386 0000442F 665A                <1>       pop    dx
15387                              <1>       ;and   dx, dx
15388 00004431 20D2                <1>       and    dl, dl
15389                              <1>              ; tst (sp)+ / pop the stack and test the mode
15390 00004433 7403                <1>       jz     short sysopen_2
15391                              <1>              ; beq op1 / is open for read op1
15392                              <1> sysopen_1: ;op0:
```

```
15393 00004435 66F7D8              <1>       neg    ax
15394                              <1>             ; neg r1
15395                              <1>                 ;/ make i-number positive if open for writing [???]
15396                              <1>       ;; NOTE: iopen always make i-number positive.
15397                              <1>       ;; Here i-number becomes negative again. [22/05/2013]
15398                              <1> sysopen_2: ;op1:
15399 00004438 31F6                <1>         xor     esi, esi
15400                              <1>             ; clr r2 / clear registers
15401 0000443A 31DB                <1>         xor     ebx, ebx
15402                              <1>             ; clr r3
15403                              <1> sysopen_3: ;1: / scan the list of entries in fsp table
15404 0000443C 389E[4E740000]      <1>         cmp     [esi+u.fp], bl ; 0
15405                              <1>             ; tstb u.fp(r2) / test the entry in the u.fp list
15406 00004442 7625                <1>         jna      short sysopen_4
15407                              <1>             ; beq 1f / if byte in list is 0 branch
15408 00004444 46                  <1>         inc     esi
15409                              <1>             ; inc r2 / bump r2 so next byte can be checked
15410 00004445 6683FE0A            <1>         cmp     si, 10
15411                              <1>             ; cmp r2,$10. / reached end of list?
15412 00004449 72F1                <1>       jb     short sysopen_3
15413                              <1>             ; blt 1b / no, go back
15414                              <1> toomanyf:
15415                              <1>       ; 14/05/2015
15416 0000444B C705[9D740000]0D00- <1>       mov    dword [u.error], ERR_TOO_MANY_FILES ; too many open files !
15417 00004453 0000                <1>
15418 00004455 E9E0FBFFFF          <1>       jmp    error
15419                              <1>             ; br error2 / yes, error (no files open)
15420                              <1> fnotfound:
15421                              <1>       ; 14/05/2015
15422 0000445A C705[9D740000]0C00- <1>       mov    dword [u.error], ERR_FILE_NOT_FOUND ; file not found !
15423 00004462 0000                <1>
15424 00004464 E9D1FBFFFF          <1>       jmp    error
15425                              <1>
15426                              <1> sysopen_4: ; 1:
15427 00004469 6683BB[16720000]00  <1>         cmp     word [ebx+fsp], 0
15428                              <1>             ; tst fsp(r3) / scan fsp entries
15429 00004471 7610                <1>         jna     short sysopen_5
15430                              <1>             ; beq 1f / if 0 branch
15431                              <1>       ; 14/05/2015 - Retro UNIX 386 v1 modification !
15432 00004473 6683C30A            <1>         add     bx, 10 ; fsp structure size = 10 bytes/entry
15433                              <1>             ; add $8.,r3 / add 8 to r3
15434                              <1>                 ; / to bump it to next entry mfsp table
15435 00004477 6681FBF401          <1>         cmp     bx, nfiles*10
15436                              <1>             ; cmp r3,$[nfiles*8.] / done scanning
15437 0000447C 72EB                <1>       jb     short sysopen_4
15438                              <1>                 ; blt 1b / no, back
15439 0000447E E9B7FBFFFF          <1>       jmp    error
15440                              <1>             ; br error2 / yes, error
15441                              <1> sysopen_5: ; 1: / r2 has index to u.fp list; r3, has index to fsp table
15442 00004483 668983[16720000]    <1>         mov     [ebx+fsp], ax
15443                              <1>             ; mov r1,fsp(r3) / put i-number of open file
15444                              <1>                 ; / into next available entry in fsp table,
15445 0000448A 668B3D[2E740000]    <1>       mov    di, [cdev] ; word ? byte ?
15446 00004491 6689BB[18720000]    <1>         mov     [ebx+fsp+2], di ; device number
15447                              <1>             ; mov cdev,fsp+2(r3) / put # of device in next word
15448 00004498 31FF                <1>         xor  edi, edi
15449 0000449A 89BB[1A720000]      <1>         mov     [ebx+fsp+4], edi ; offset pointer (0)
15450                              <1>             ; clr fsp+4(r3)
15451 000044A0 6689BB[1E720000]    <1>         mov     [ebx+fsp+8], di ; open count (0), deleted flag (0)
15452                              <1>                 ; clr fsp+6(r3) / clear the next two words
15453 000044A7 89D8                <1>       mov    eax, ebx
15454 000044A9 B30A                <1>       mov    bl, 10
15455 000044AB F6F3                <1>       div    bl
15456                              <1>             ; asr r3
15457                              <1>             ; asr r3 / divide by 8
15458                              <1>             ; asr r3 ; / to get number of the fsp entry-1
15459 000044AD FEC0                <1>       inc    al
15460                              <1>             ; inc r3 / add 1 to get fsp entry number
15461 000044AF 8886[4E740000]      <1>         mov     [esi+u.fp], al
15462                              <1>             ; movb r3,u.fp(r2) / move entry number into
15463                              <1>                 ; / next available slot in u.fp list
15464 000044B5 8935[48740000]      <1>         mov     [u.r0], esi
15465                              <1>             ; mov r2,*u.r0 / move index to u.fp list
15466                              <1>                         ; / into r0 loc on stack
15467 000044BB E99AFBFFFF          <1>         jmp  sysret
15468                              <1>             ; br sysret2
15469                              <1>
15470                              <1>       ;
15471                              <1>       ; 'fsp' table (10 bytes/entry)
15472                              <1>       ; bit 15                          bit 0
15473                              <1>       ; ---|-----------------------------------------
15474                              <1>       ; r/w|       i-number of open file
15475                              <1>       ; ---|-----------------------------------------
15476                              <1>       ;              device number
15477                              <1>       ; -----------------------------------------------
15478                              <1>       ; offset pointer, r/w pointer to file (bit 0-15)
15479                              <1>       ; -----------------------------------------------
15480                              <1>       ; offset pointer, r/w pointer to file (bit 16-31)
15481                              <1>       ; ----------------------|------------------------
15482                              <1>       ;  flag that says file    | number of processes
15483                              <1>       ;   has been deleted      | that have file open
15484                              <1>       ; ----------------------|------------------------
15485                              <1>       ;
15486                              <1>
15487                              <1> syscreat: ; < create file >
15488                              <1>       ; 27/12/2015 (Retro UNIX 386 v1.1)
15489                              <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15490                              <1>       ; 27/05/2013 (Retro UNIX 8086 v1)
15491                              <1>       ;
15492                              <1>       ; 'syscreat' called with two arguments; name and mode.
15493                              <1>       ; u.namep points to name of the file and mode is put
15494                              <1>       ; on the stack. 'namei' is called to get i-number of the file.
15495                              <1>       ; If the file aready exists, it's mode and owner remain
15496                              <1>       ; unchanged, but it is truncated to zero length. If the file
15497                              <1>       ; did not exist, an i-node is created with the new mode via
```

```
15498                                  <1>       ; 'maknod' whether or not the file already existed, it is
15499                                  <1>       ; open for writing. The fsp table is then searched for a free
15500                                  <1>       ; entry. When a free entry is found, proper data is placed
15501                                  <1>       ; in it and the number of this entry is put in the u.fp list.
15502                                  <1>       ; The index to the u.fp (also know as the file descriptor)
15503                                  <1>       ; is put in the user's r0.
15504                                  <1>       ;
15505                                  <1>       ; Calling sequence:
15506                                  <1>       ;     syscreate; name; mode
15507                                  <1>       ; Arguments:
15508                                  <1>       ;     name - name of the file to be created
15509                                  <1>       ;     mode - mode of the file to be created
15510                                  <1>       ; Inputs: (arguments)
15511                                  <1>       ; Outputs: *u.r0 - index to u.fp list
15512                                  <1>       ;               (the file descriptor of new file)
15513                                  <1>       ; ..........................................................
15514                                  <1>       ;
15515                                  <1>       ; Retro UNIX 8086 v1 modification:
15516                                  <1>       ;      'syscreate' system call has two arguments; so,
15517                                  <1>       ;     * 1st argument, name is pointed to by BX register
15518                                  <1>       ;     * 2nd argument, mode is in CX register
15519                                  <1>       ;
15520                                  <1>       ;     AX register (will be restored via 'u.r0') will return
15521                                  <1>       ;     to the user with the file descriptor/number
15522                                  <1>       ;     (index to u.fp list).
15523                                  <1>       ;
15524                                  <1>       ;call  arg2
15525                                  <1>       ; * name - 'u.namep' points to address of file/path name
15526                                  <1>       ;          in the user's program segment ('u.segmnt')
15527                                  <1>       ;          with offset in BX register (as sysopen argument 1).
15528                                  <1>       ; * mode - sysopen argument 2 is in CX register
15529                                  <1>       ;          which is on top of stack.
15530                                  <1>       ;
15531                                  <1>             ; jsr r0,arg2 / put file name in u.namep put mode
15532                                  <1>                       ; / on stack
15533 000044C0 891D[60740000]         <1>       mov   [u.namep], ebx ; file name address
15534 000044C6 6651                   <1>       push  cx ; mode
15535 000044C8 E8290A0000             <1>       call  namei
15536                                  <1>             ; jsr r0,namei / get the i-number
15537                                  <1>         ;and ax, ax
15538                                  <1>       ;jz   short syscreat_1
15539 000044CD 721E                   <1>       jc    short syscreat_1
15540                                  <1>             ; br  2f / if file doesn't exist 2f
15541                                  <1>       ; 27/12/2015
15542 000044CF 6683F829               <1>       cmp   ax, 41 ; device inode ?
15543 000044D3 0F824EFFFFFF           <1>         jb      syscreat_0 ; yes
15544                                  <1>       ;
15545 000044D9 66F7D8                 <1>       neg   ax
15546                                  <1>             ; neg r1 / if file already exists make i-number
15547                                  <1>                    ; / negative (open for writing)
15548 000044DC E884190000             <1>       call  iopen
15549                                  <1>             ; jsr r0,iopen /
15550 000044E1 E835130000             <1>       call  itrunc
15551                                  <1>             ; jsr r0,itrunc / truncate to 0 length
15552 000044E6 6659                   <1>       pop   cx ; pop mode (did not exist in original Unix v1 !?)
15553 000044E8 E948FFFFFF             <1>         jmp     sysopen_1
15554                                  <1>             ; br op0
15555                                  <1> syscreat_1: ; 2: / file doesn't exist
15556 000044ED 6658                   <1>       pop   ax
15557                                  <1>             ; mov (sp)+,r1 / put the mode in r1
15558 000044EF 30E4                   <1>       xor   ah, ah
15559                                  <1>             ; bic $!377,r1 / clear upper byte
15560 000044F1 E8D30C0000             <1>       call  maknod
15561                                  <1>             ; jsr r0,maknod / make an i-node for this file
15562 000044F6 66A1[7A740000]         <1>       mov   ax, [u.dirbuf]
15563                                  <1>             ; mov u.dirbuf,r1 / put i-number
15564                                  <1>                               ; / for this new file in r1
15565 000044FC E934FFFFFF             <1>         jmp     sysopen_1
15566                                  <1>             ; br op0 / open the file
15567                                  <1>
15568                                  <1> sysmkdir: ; < make directory >
15569                                  <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15570                                  <1>       ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
15571                                  <1>       ;
15572                                  <1>       ; 'sysmkdir' creates an empty directory whose name is
15573                                  <1>       ; pointed to by arg 1. The mode of the directory is arg 2.
15574                                  <1>       ; The special entries '.' and '..' are not present.
15575                                  <1>       ; Errors are indicated if the directory already exists or
15576                                  <1>       ; user is not the super user.
15577                                  <1>       ;
15578                                  <1>       ; Calling sequence:
15579                                  <1>       ;     sysmkdir; name; mode
15580                                  <1>       ; Arguments:
15581                                  <1>       ;     name - points to the name of the directory
15582                                  <1>       ;     mode - mode of the directory
15583                                  <1>       ; Inputs: (arguments)
15584                                  <1>       ; Outputs: -
15585                                  <1>       ;    (sets 'directory' flag to 1;
15586                                  <1>       ;    'set user id on execution' and 'executable' flags to 0)
15587                                  <1>       ; ..........................................................
15588                                  <1>       ;
15589                                  <1>       ; Retro UNIX 8086 v1 modification:
15590                                  <1>       ;      'sysmkdir' system call has two arguments; so,
15591                                  <1>       ;     * 1st argument, name is pointed to by BX register
15592                                  <1>       ;     * 2nd argument, mode is in CX register
15593                                  <1>       ;
15594                                  <1>
15595                                  <1> ; / make a directory
15596                                  <1>
15597                                  <1>       ;call  arg2
15598                                  <1>       ; * name - 'u.namep' points to address of file/path name
15599                                  <1>       ;          in the user's program segment ('u.segmnt')
15600                                  <1>       ;          with offset in BX register (as sysopen argument 1).
15601                                  <1>       ; * mode - sysopen argument 2 is in CX register
15602                                  <1>       ;          which is on top of stack.
```

```
15603                                  <1>
15604                                  <1>             ; jsr r0,arg2 / put file name in u.namep put mode
15605                                  <1>                       ; / on stack
15606 00004501 891D[60740000]          <1>       mov   [u.namep], ebx
15607 00004507 6651                    <1>       push  cx ; mode
15608 00004509 E8E8090000              <1>       call  namei
15609                                  <1>             ; jsr r0,namei / get the i-number
15610                                  <1>             ;     br .+4 / if file not found branch around error
15611                                  <1>         ;xor       ax, ax
15612                                  <1>       ;jnz  error
15613 0000450E 731C                    <1>       jnc   short dir_exists ; 14/05/2015
15614                                  <1>       ;jnc  error
15615                                  <1>             ; br  error2 / directory already exists (error)
15616 00004510 803D[94740000]00        <1>       cmp   byte [u.uid], 0 ; 02/08/2013
15617                                  <1>             ;tstb u.uid / is user the super user
15618 00004517 7622                    <1>       jna   short dir_access_err ; 14/05/2015
15619                                  <1>       ;jna  error
15620                                  <1>             ;bne error2 / no, not allowed
15621 00004519 6658                    <1>       pop   ax
15622                                  <1>             ;mov (sp)+,r1 / put the mode in r1
15623 0000451B 6683E0CF                <1>       and   ax, 0FFCFh ; 1111111111001111b
15624                                  <1>             ;bic $!317,r1 / all but su and ex
15625                                  <1>       ;or   ax , 4000h ; 1011111111111111b
15626 0000451F 80CC40                  <1>       or    ah, 40h ; Set bit 14 to 1
15627                                  <1>             ;bis $40000,r1 / directory flag
15628 00004522 E8A20C0000              <1>       call  maknod
15629                                  <1>             ;jsr r0,maknod / make the i-node for the directory
15630 00004527 E92EFBFFFF              <1>       jmp   sysret
15631                                  <1>             ;br sysret2 /
15632                                  <1> dir_exists:
15633                                  <1>       ; 14/05/2015
15634 0000452C C705[9D740000]0E00-     <1>       mov   dword [u.error], ERR_DIR_EXISTS ; dir. already exists !
15635 00004534 0000                    <1>
15636 00004536 E9FFFAFFFF              <1>       jmp   error
15637                                  <1> dir_access_err:
15638                                  <1>       ; 14/05/2015
15639 0000453B C705[9D740000]0B00-     <1>       mov   dword [u.error], ERR_DIR_ACCESS ; permission denied !
15640 00004543 0000                    <1>
15641 00004545 E9F0FAFFFF              <1>       jmp   error
15642                                  <1>
15643                                  <1> sysclose: ;<close file>
15644                                  <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15645                                  <1>       ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
15646                                  <1>       ;
15647                                  <1>       ; 'sysclose', given a file descriptor in 'u.r0', closes the
15648                                  <1>       ; associated file. The file descriptor (index to 'u.fp' list)
15649                                  <1>       ; is put in r1 and 'fclose' is called.
15650                                  <1>       ;
15651                                  <1>       ; Calling sequence:
15652                                  <1>       ;     sysclose
15653                                  <1>       ; Arguments:
15654                                  <1>       ;     -
15655                                  <1>       ; Inputs: *u.r0 - file descriptor
15656                                  <1>       ; Outputs: -
15657                                  <1>       ; .........................................................
15658                                  <1>       ;
15659                                  <1>       ; Retro UNIX 8086 v1 modification:
15660                                  <1>       ;      The user/application program puts file descriptor
15661                                  <1>       ;        in BX register as 'sysclose' system call argument.
15662                                  <1>       ;      (argument transfer method 1)
15663                                  <1>
15664                                  <1>       ; / close the file
15665                                  <1>
15666 0000454A 89D8                    <1>       mov   eax, ebx
15667 0000454C E823090000              <1>       call  fclose
15668                                  <1>             ; mov *u.r0,r1 / move index to u.fp list into r1
15669                                  <1>             ; jsr r0,fclose / close the file
15670                                  <1>                   ; br error2 / unknown file descriptor
15671                                  <1>             ; br sysret2
15672                                  <1>       ; 14/05/2015
15673 00004551 0F8303FBFFFF            <1>       jnc   sysret
15674 00004557 C705[9D740000]0A00-     <1>       mov   dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
15675 0000455F 0000                    <1>
15676 00004561 E9D4FAFFFF              <1>       jmp   error
15677                                  <1>
15678                                  <1> sysemt:
15679                                  <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15680                                  <1>       ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
15681                                  <1>       ;
15682                                  <1>       ; Retro UNIX 8086 v1 modification:
15683                                  <1>       ;     'Enable Multi Tasking'  system call instead
15684                                  <1>       ;     of 'Emulator Trap' in original UNIX v1 for PDP-11.
15685                                  <1>       ;
15686                                  <1>       ; Retro UNIX 8086 v1 feature only!
15687                                  <1>       ;     Using purpose: Kernel will start without time-out
15688                                  <1>       ;     (internal clock/timer) functionality.
15689                                  <1>       ;     Then etc/init will enable clock/timer for
15690                                  <1>       ;     multi tasking. (Then it will not be disabled again
15691                                  <1>       ;     except hardware reset/restart.)
15692                                  <1>       ;
15693                                  <1>
15694 00004566 803D[94740000]00        <1>       cmp   byte [u.uid], 0 ; root ?
15695                                  <1>       ;ja   error
15696 0000456D 0F8770FBFFFF            <1>       ja    badsys ; 14/05/2015
15697                                  <1> emt_0:
15698 00004573 FA                      <1>       cli
15699 00004574 21DB                    <1>       and   ebx, ebx
15700 00004576 7410                    <1>       jz    short emt_2
15701                                  <1>       ; Enable multi tasking -time sharing-
15702 00004578 B8[7D540000]            <1>       mov   eax, clock
15703                                  <1> emt_1:
15704 0000457D A3[D8070000]            <1>       mov   [x_timer], eax
15705 00004582 FB                      <1>       sti
15706 00004583 E9D2FAFFFF              <1>       jmp   sysret
15707                                  <1> emt_2:
```

```
15708                                  <1>       ; Disable multi tasking -time sharing-
15709 00004588 B8[E0070000]            <1>       mov    eax, u_timer
15710 0000458D EBEE                    <1>       jmp    short emt_1
15711                                  <1>
15712                                  <1>       ; Original UNIX v1 'sysemt' routine
15713                                  <1> ;sysemt:
15714                                  <1>         ;
15715                                  <1>       ;jsr    r0,arg; 30 / put the argument of the sysemt call
15716                                  <1>                        ; / in loc 30
15717                                  <1>         ;cmp    30,$core / was the argument a lower address
15718                                  <1>                        ; / than core
15719                                  <1>         ;blo    1f / yes, rtssym
15720                                  <1>         ;cmp    30,$ecore / no, was it higher than "core"
15721                                  <1>                        ; / and less than "ecore"
15722                                  <1>         ;blo    2f / yes, sysret2
15723                                  <1> ;1:
15724                                  <1>         ;mov    $rtssym,30
15725                                  <1> ;2:
15726                                  <1>         ;br     sysret2
15727                                  <1>
15728                                  <1> sysilgins:
15729                                  <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15730                                  <1>       ; 03/06/2013
15731                                  <1>       ; Retro UNIX 8086 v1 modification:
15732                                  <1>       ;      not a valid system call ! (not in use)
15733                                  <1>       ;
15734 0000458F E94FFBFFFF              <1>       jmp    badsys
15735                                  <1>       ;jmp   error
15736                                  <1>       ;;jmp  sysret
15737                                  <1>
15738                                  <1>       ; Original UNIX v1 'sysemt' routine
15739                                  <1> ;sysilgins: / calculate proper illegal instruction trap address
15740                                  <1>         ;jsr    r0,arg; 10 / take address from sysilgins call
15741                                  <1>                      ;/ put it in loc 8.,
15742                                  <1>         ;cmp    10,$core / making it the illegal instruction
15743                                  <1>                     ; / trap address
15744                                  <1>         ;blo    1f / is the address a user core address?
15745                                  <1>             ; / yes, go to 2f
15746                                  <1>         ;cmp    10,$ecore
15747                                  <1>         ;blo    2f
15748                                  <1> ;1:
15749                                  <1>         ;mov    $fpsym,10 / no, make 'fpsum' the illegal
15750                                  <1>                  ; / instruction trap address for the system
15751                                  <1> ;2:
15752                                  <1>         ;br     sysret2 / return to the caller via 'sysret'
15753                                  <1>
15754                                  <1> sysmdate: ; < change the modification time of a file >
15755                                  <1>       ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
15756                                  <1>       ; 03/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
15757                                  <1>       ;
15758                                  <1>       ; 'sysmdate' is given a file name. It gets inode of this
15759                                  <1>       ; file into core. The user is checked if he is the owner
15760                                  <1>       ; or super user. If he is neither an error occurs.
15761                                  <1>       ; 'setimod' is then called to set the i-node modification
15762                                  <1>       ; byte and the modification time, but the modification time
15763                                  <1>       ; is overwritten by whatever get put on the stack during
15764                                  <1>       ; a 'systime' system call. This calls are restricted to
15765                                  <1>       ; the super user.
15766                                  <1>       ;
15767                                  <1>       ; Calling sequence:
15768                                  <1>       ;     sysmdate; name
15769                                  <1>       ; Arguments:
15770                                  <1>       ;     name - points to the name of file
15771                                  <1>       ; Inputs: (arguments)
15772                                  <1>       ; Outputs: -
15773                                  <1>       ; ..........................................................
15774                                  <1>       ;
15775                                  <1>       ; Retro UNIX 8086 v1 modification:
15776                                  <1>       ;      The user/application program puts address
15777                                  <1>       ;      of the file name in BX register
15778                                  <1>       ;      as 'sysmdate' system call argument.
15779                                  <1>       ;
15780                                  <1> ; / change the modification time of a file
15781                                  <1>                ; jsr r0,arg; u.namep / point u.namep to the file name
15782 00004594 891D[60740000]          <1>         mov  [u.namep], ebx
15783 0000459A E857090000              <1>       call   namei
15784                                  <1>                ; jsr r0,namei / get its i-number
15785 0000459F 0F82B5FEFFFF            <1>         jc   fnotfound ; file not found !
15786                                  <1>       ;jc    error
15787                                  <1>                ; br error2 / no, such file
15788 000045A5 E83C110000              <1>       call   iget
15789                                  <1>                ; jsr r0,iget / get i-node into core
15790 000045AA A0[94740000]            <1>       mov    al, [u.uid]
15791 000045AF 3A05[19710000]          <1>       cmp    al, [i.uid]
15792                                  <1>                ; cmpb u.uid,i.uid / is user same as owner
15793 000045B5 7413                    <1>       je     short mdate_1
15794                                  <1>                ; beq 1f / yes
15795 000045B7 20C0                    <1>       and    al, al
15796                                  <1>                ; tstb u.uid / no, is user the super user
15797                                  <1>       ;jnz   error
15798                                  <1>                ; bne error2 / no, error
15799 000045B9 740F                    <1>       jz     short mdate_1
15800 000045BB C705[9D740000]0B00-     <1>       mov    dword [u.error], ERR_FILE_ACCESS ; permission denied !
15801 000045C3 0000                    <1>
15802 000045C5 E970FAFFFF              <1>       jmp    error
15803                                  <1> mdate_1: ;1:
15804 000045CA E82A120000              <1>       call   setimod
15805                                  <1>                ; jsr r0,setimod / fill in modification data,
15806                                  <1>                                 ; / time etc.
15807 000045CF BE[B0700000]            <1>       mov    esi, p_time
15808 000045D4 BF[30710000]            <1>       mov    edi, i.mtim
15809 000045D9 A5                      <1>       movsd
15810                                  <1>                ; mov 4(sp),i.mtim / move present time to
15811                                  <1>                ; mov 2(sp),i.mtim+2 / modification time
15812 000045DA E97BFAFFFF              <1>         jmp  sysret
```

```
15813                                  <1>             ; br sysret2
15814                                  <1>
15815                                  <1> sysstty: ; < set tty status and mode >
15816                                  <1>       ; 17/11/2015
15817                                  <1>       ; 12/11/2015
15818                                  <1>       ; 29/10/2015
15819                                  <1>       ; 17/10/2015
15820                                  <1>       ; 13/10/2015
15821                                  <1>       ; 29/06/2015
15822                                  <1>       ; 27/06/2015 (Retro UNIX 386 v1 - Beginning)
15823                                  <1>       ; 02/06/2013 - 12/07/2014 (Retro UNIX 8086 v1)
15824                                  <1>       ;
15825                                  <1>       ; 'sysstty' sets the status and mode of the typewriter
15826                                  <1>       ; whose file descriptor is in (u.r0).
15827                                  <1>       ;
15828                                  <1>       ; Calling sequence:
15829                                  <1>       ;     sysstty; arg
15830                                  <1>       ; Arguments:
15831                                  <1>       ;     arg - address of 3 consequitive words that contain
15832                                  <1>       ;           the source of status data
15833                                  <1>       ; Inputs: ((*u.r0 - file descriptor & argument))
15834                                  <1>       ; Outputs: ((status in address which is pointed to by arg))
15835                                  <1>       ; ..........................................................
15836                                  <1>       ;
15837                                  <1>       ; Retro UNIX 8086 v1 modification:
15838                                  <1>       ;     'sysstty' system call will set the tty
15839                                  <1>       ;     (clear keyboard buffer and set cursor position)
15840                                  <1>       ;      in following manner:
15841                                  <1>       ;   NOTE: All of tty setting functions are here (16/01/2014)
15842                                  <1>       ;
15843                                  <1>       ; Inputs:
15844                                  <1>       ;     BX = 0 --> means
15845                                  <1>       ;        If CL = FFh
15846                                  <1>       ;           set cursor position for console tty, only
15847                                  <1>       ;           CH will be ignored (char. will not be written)
15848                                  <1>       ;        If CH = 0 (CL < FFh)
15849                                  <1>       ;           set console tty for (current) process
15850                                  <1>       ;           CL = tty number (0 to 9)
15851                                  <1>       ;           (If CH = 0, character will not be written)
15852                                  <1>       ;         If CH > 0 (CL < FFh)
15853                                  <1>       ;            CL = tty number (0 to 9)
15854                                  <1>       ;           CH = character will be written
15855                                  <1>       ;             at requested cursor position (in DX)
15856                                  <1>       ;        DX = cursor position for tty number 0 to 7.
15857                                  <1>       ;           (only tty number 0 to 7)
15858                                  <1>       ;         DL = communication parameters (for serial ports)
15859                                  <1>       ;            (only for COM1 and COM2 serial ports)
15860                                  <1>       ;        DH < 0FFh -> DL is valid, initialize serial port
15861                                  <1>       ;                  or set cursor position
15862                                  <1>       ;        DH = 0FFh -> DL is not valid
15863                                  <1>       ;           do not set serial port parameters
15864                                  <1>       ;           or do not set cursor position
15865                                  <1>       ;
15866                                  <1>       ;     BX > 0 --> points to name of tty
15867                                  <1>       ;        CH > 0 -->
15868                                  <1>       ;           CH = character will be written in current
15869                                  <1>       ;           cursor position (for tty number from 0 to 7)
15870                                  <1>       ;           or character will be sent to serial port
15871                                  <1>       ;           (for tty number 8 or 9)
15872                                  <1>       ;           CL = color of the character if tty number < 8.
15873                                  <1>       ;        CH = 0 --> Do not write a character,
15874                                  <1>       ;           set mode (tty 8 to 9) or
15875                                  <1>       ;           set current cursor positions (tty 0 to 7) only.
15876                                  <1>       ;        DX = cursor position for tty number 0 to 7.
15877                                  <1>       ;        DH = FFh --> Do not set cursor pos (or comm. params.)
15878                                  <1>       ;           (DL is not valid)
15879                                  <1>       ;        DL = communication parameters
15880                                  <1>       ;           for tty number 8 or 9 (COM1 or COM2).
15881                                  <1>       ; Outputs:
15882                                  <1>       ;     cf = 0 -> OK
15883                                  <1>       ;          AL = tty number (0 to 9)
15884                                  <1>       ;          AH = line status if tty number is 8 or 9
15885                                  <1>       ;          AH = process number (of the caller)
15886                                  <1>       ;     cf = 1 means error (requested tty is not ready)
15887                                  <1>       ;          AH = FFh if the tty is locked
15888                                  <1>       ;              (owned by another process)
15889                                  <1>       ;             = process number (of the caller)
15890                                  <1>       ;              (if < FFh and tty number < 8)
15891                                  <1>       ;          AL = tty number (0FFh if it does not exist)
15892                                  <1>       ;          AH = line status if tty number is 8 or 9
15893                                  <1>       ;     NOTE: Video page will be cleared if cf = 0.
15894                                  <1>       ;
15895                                  <1>       ; 27/06/2015 (32 bit modifications)
15896                                  <1>       ; 14/01/2014
15897 000045DF 31C0                   <1>       xor    eax, eax
15898 000045E1 6648                   <1>       dec    ax ; 17/10/2015
15899 000045E3 A3[48740000]           <1>       mov    [u.r0], eax ; 0FFFFh
15900 000045E8 21DB                   <1>       and    ebx, ebx
15901 000045EA 0F85CB000000           <1>         jnz     sysstty_6
15902                                  <1> ; set console tty
15903                                  <1>       ; 29/10/2015
15904                                  <1>       ; 17/01/2014
15905 000045F0 80F909                 <1>       cmp    cl, 9
15906 000045F3 7613                   <1>       jna    short sysstty_0
15907                                  <1>       ; 17/11/2015
15908 000045F5 80F9FF                 <1>       cmp    cl, 0FFh
15909 000045F8 7202                   <1>       jb     short sysstty_13
15910 000045FA 88CD                   <1>       mov    ch, cl ; force CH value to FFh
15911                                  <1> sysstty_13:
15912 000045FC 8A1D[97740000]         <1>       mov    bl, [u.uno] ; process number
15913 00004602 8A8B[95710000]         <1>       mov    cl, [ebx+p.ttyc-1] ; current/console tty
15914                                  <1> sysstty_0:
15915                                  <1>       ; 29/06/2015
15916 00004608 6652                   <1>       push   dx
15917 0000460A 6651                   <1>       push   cx
```

```
15918 0000460C 30D2               <1>       xor    dl, dl ; sysstty call sign
15919 0000460E 88C8               <1>       mov    al, cl
15920 00004610 A2[48740000]       <1>       mov    [u.r0], al ; tyy number (0 to 9)
15921 00004615 E8E5180000         <1>       call   ottyp
15922 0000461A 6659               <1>       pop    cx
15923 0000461C 665A               <1>       pop    dx
15924                             <1>       ;
15925 0000461E 7257               <1>       jc     short sysstty_pd_err
15926                             <1>       ;
15927 00004620 80F908             <1>       cmp    cl, 8
15928 00004623 7222               <1>       jb     short sysstty_2
15929                             <1>       ;
15930 00004625 80FEFF             <1>       cmp    dh, 0FFh
15931 00004628 741D               <1>       je     short sysstty_2
15932                             <1>              ; set communication parameters for serial ports
15933                             <1>       ; 29/10/2015
15934 0000462A 88D4               <1>       mov    ah, dl ; communication parameters
15935                             <1>              ; ah = 0E3h = 11100011b = 115200 baud,
15936                             <1>              ;                    THRE int + RDA int
15937                             <1>              ; ah = 23h = 00100011b = 9600 baud,
15938                             <1>              ;                    THRE int + RDA int
15939 0000462C 28C0               <1>       sub    al, al ; 0
15940                             <1>       ; 12/07/2014
15941 0000462E 80F909             <1>       cmp    cl, 9
15942 00004631 7202               <1>       jb     short sysstty_1
15943 00004633 FEC0               <1>       inc    al
15944                             <1> sysstty_1:
15945 00004635 6651               <1>       push   cx
15946                             <1>       ; 29/06/2015
15947 00004637 E8ABF4FFFF         <1>       call   sp_setp ; Set serial port communication parameters
15948 0000463C 66890D[49740000]   <1>       mov    [u.r0+1], cx ; Line status (ah)
15949                             <1>                           ; Modem status (EAX bits 16 to 23)
15950 00004643 6659               <1>       pop    cx
15951 00004645 7265               <1>         jc      short sysstty_tmout_err ; 29/10/2015
15952                             <1> sysstty_2:
15953                             <1>       ; 17/01/2014
15954 00004647 20ED               <1>       and    ch, ch       ; set cursor position
15955                             <1>                       ; or comm. parameters ONLY
15956 00004649 750D               <1>       jnz    short sysstty_3
15957 0000464B 0FB61D[97740000]   <1>       movzx  ebx, byte [u.uno] ; process number
15958 00004652 888B[95710000]     <1>       mov    [ebx+p.ttyc-1], cl ; console tty
15959                             <1> sysstty_3:
15960                             <1>       ; 16/01/2014
15961 00004658 88E8               <1>       mov    al, ch ; character  ; 0 to FFh
15962                             <1>       ; 17/11/2015
15963 0000465A B507               <1>       mov    ch, 7  ; Default color (light gray)
15964 0000465C 38E9               <1>       cmp    cl, ch ; 7 (tty number)
15965 0000465E 0F86C5000000       <1>         jna     sysstty_9
15966                             <1> sysstty_12:
15967                             <1>       ;; BX = 0, CL = 8 or CL = 9
15968                             <1>       ; (Set specified serial port as console tty port)
15969                             <1>       ; CH = character to be written
15970                             <1>       ; 15/04/2014
15971                             <1>       ; CH = 0 --> initialization only
15972                             <1>       ; AL = character
15973                             <1>       ; 26/06/2014
15974 00004664 880D[9C740000]     <1>       mov    [u.ttyn], cl
15975                             <1>       ; 12/07/2014
15976 0000466A 88CC               <1>       mov    ah, cl ; tty number (8 or 9)
15977 0000466C 20C0               <1>       and    al, al
15978 0000466E 7416               <1>       jz     short sysstty_4 ; al = ch = 0
15979                             <1>       ; 04/07/2014
15980 00004670 E8981E0000         <1>       call   sndc
15981                             <1>       ; 12/07/2014
15982 00004675 EB1B               <1>       jmp    short sysstty_5
15983                             <1> sysstty_pd_err: ; 29/06/2015
15984                             <1>       ; 'permission denied !' error
15985 00004677 C705[9D740000]0B00- <1>      mov    dword [u.error], ERR_NOT_OWNER
15986 0000467F 0000               <1>
15987 00004681 E9B4F9FFFF         <1>       jmp    error
15988                             <1> sysstty_4:
15989                             <1>       ; 12/07/2014
15990                             <1>       ;xchg  ah, al ; al = 0 -> al = ah, ah = 0
15991 00004686 88E0               <1>       mov    al, ah ; 29/06/2015
15992 00004688 2C08               <1>       sub    al, 8
15993                             <1>       ; 27/06/2015
15994 0000468A E850F4FFFF         <1>       call   sp_status ; get serial port status
15995                             <1>       ; AL = Line status, AH = Modem status
15996                             <1>       ; 12/11/2015
15997 0000468F 3C80               <1>       cmp    al, 80h
15998 00004691 F5                 <1>       cmc
15999                             <1> sysstty_5:
16000 00004692 66A3[49740000]     <1>       mov    [u.r0+1], ax ; ah = line status
16001                             <1>                           ; EAX bits 16-23 = modem status
16002 00004698 9C                 <1>       pushf
16003 00004699 30D2               <1>       xor    dl, dl ; sysstty call sign
16004 0000469B A0[9C740000]       <1>       mov    al, [u.ttyn] ; 26/06/2014
16005 000046A0 E86D190000         <1>       call   cttyp
16006 000046A5 9D                 <1>       popf
16007 000046A6 0F83AEF9FFFF       <1>       jnc    sysret ; time out error
16008                             <1>
16009                             <1> sysstty_tmout_err:
16010 000046AC C705[9D740000]1900- <1>      mov    dword [u.error], ERR_TIME_OUT
16011 000046B4 0000               <1>
16012 000046B6 E97FF9FFFF         <1>       jmp    error
16013                             <1> sysstty_6:
16014 000046BB 6652               <1>       push   dx
16015 000046BD 6651               <1>       push   cx
16016 000046BF 891D[60740000]     <1>       mov    [u.namep], ebx
16017 000046C5 E82C080000         <1>       call   namei
16018 000046CA 6659               <1>       pop    cx
16019 000046CC 665A               <1>       pop    dx
16020 000046CE 720E               <1>       jc     short sysstty_inv_dn
16021                             <1>       ;
16022 000046D0 6683F813           <1>       cmp    ax, 19  ; inode number of /dev/COM2
```

```
16023 000046D4 7708                <1>       ja    short sysstty_inv_dn ; 27/06/2015
16024                              <1>       ;
16025 000046D6 3C0A                <1>       cmp   al, 10 ; /dev/tty0 .. /dev/tty7
16026                              <1>                    ; /dev/COM1, /dev/COM2
16027 000046D8 7213                <1>       jb    short sysstty_7
16028 000046DA 2C0A                <1>       sub   al, 10
16029 000046DC EB20                <1>       jmp   short sysstty_8
16030                              <1> sysstty_inv_dn:
16031                              <1>       ; 27/06/2015
16032                              <1>       ; Invalid device name (not a tty) ! error
16033                              <1>       ; (Device is not a tty or device name not found)
16034 000046DE C705[9D740000]1800- <1>       mov   dword [u.error], ERR_INV_DEV_NAME
16035 000046E6 0000                <1>
16036 000046E8 E94DF9FFFF          <1>       jmp   error
16037                              <1> sysstty_7:
16038 000046ED 3C01                <1>       cmp   al, 1 ; /dev/tty
16039 000046EF 75ED                <1>       jne   short sysstty_inv_dn ; 27/06/2015
16040 000046F1 0FB61D[97740000]    <1>       movzx ebx, byte [u.uno] ; process number
16041 000046F8 8A83[95710000]      <1>       mov   al, [ebx+p.ttyc-1] ; console tty
16042                              <1> sysstty_8:
16043 000046FE A2[48740000]        <1>       mov   [u.r0], al
16044 00004703 6652                <1>       push  dx
16045 00004705 6650                <1>       push  ax
16046 00004707 6651                <1>       push  cx
16047 00004709 E8F1170000          <1>       call  ottyp
16048 0000470E 6659                <1>       pop   cx
16049 00004710 6658                <1>       pop   ax
16050 00004712 665A                <1>       pop   dx
16051 00004714 0F825DFFFFFF        <1>         jc      sysstty_pd_err ; 'permission denied !'
16052                              <1>       ; 29/10/2015
16053 0000471A 86E9                <1>       xchg  ch, cl
16054                              <1>             ; cl = character, ch = color code
16055 0000471C 86C1                <1>       xchg  al, cl
16056                              <1>             ; al = character, cl = tty number
16057 0000471E 80F907              <1>       cmp   cl, 7
16058 00004721 0F873DFFFFFF        <1>         ja      sysstty_12
16059                              <1>       ;
16060                              <1>       ; 16/01/2014
16061 00004727 30FF                <1>       xor   bh, bh
16062                              <1>       ;
16063                              <1> sysstty_9:  ; tty 0 to tty 7
16064                              <1>       ; al = character
16065 00004729 80FEFF              <1>       cmp   dh, 0FFh ; Do not set cursor position
16066 0000472C 740F                <1>       je    short sysstty_10
16067 0000472E 6651                <1>       push  cx
16068 00004730 6650                <1>       push  ax
16069                              <1>       ; movzx, ebx, cl
16070 00004732 88CB                <1>       mov   bl, cl ; (tty number = video page number)
16071 00004734 E8C5CEFFFF          <1>       call  set_cpos
16072 00004739 6658                <1>       pop   ax
16073 0000473B 6659                <1>       pop   cx
16074                              <1> sysstty_10:
16075                              <1>       ; 29/10/2015
16076 0000473D 08C0                <1>       or    al, al ; character
16077 0000473F 740F                <1>       jz      short sysstty_11 ; al = 0
16078                              <1>       ; 17/11/2015
16079 00004741 3CFF                <1>       cmp   al, 0FFh
16080 00004743 730B                <1>       jnb   short sysstty_11
16081                              <1>             ; ch > 0 and ch < FFh
16082                              <1>       ; write a character at current cursor position
16083 00004745 88EC                <1>       mov   ah, ch ; color/attribute
16084                              <1>       ; 12/07/2014
16085 00004747 6651                <1>       push  cx
16086 00004749 E8F9CFFFFF          <1>       call  write_c_current
16087 0000474E 6659                <1>       pop   cx
16088                              <1> sysstty_11:
16089                              <1>       ; 14/01/2014
16090 00004750 30D2                <1>       xor   dl, dl ; sysstty call sign
16091                              <1>       ; 18/01/2014
16092                              <1>       ;movzx eax, cl ; 27/06/2015
16093 00004752 88C8                <1>       mov   al, cl
16094 00004754 E8B9180000          <1>       call  cttyp
16095 00004759 E9FCF8FFFF          <1>       jmp   sysret
16096                              <1>
16097                              <1> ; Original UNIX v1 'sysstty' routine:
16098                              <1> ; gtty:
16099                              <1> ;sysstty: / set mode of typewriter; 3 consequtive word arguments
16100                              <1>         ;jsr    r0,gtty / r1 will have offset to tty block,
16101                              <1>       ;           / r2 has source
16102                              <1>         ;mov    r2,-(sp)
16103                              <1>         ;mov    r1,-(sp) / put r1 and r2 on the stack
16104                              <1> ;1: / flush the clist wait till typewriter is quiescent
16105                              <1>         ;mov    (sp),r1 / restore r1 to tty block offset
16106                              <1>         ;movb   tty+3(r1),0f / put cc offset into getc argument
16107                              <1>         ;mov    $240,*$ps / set processor priority to 5
16108                              <1>         ;jsr    r0,getc; 0:../ put character from clist in r1
16109                              <1>         ;       br .+4 / list empty, skip branch
16110                              <1>         ;br     1b / get another character until list is empty
16111                              <1>         ;mov    0b,r1 / move cc offset to r1
16112                              <1>         ;inc    r1 / bump it for output clist
16113                              <1>         ;tstb   cc(r1) / is it 0
16114                              <1>         ;beq    1f / yes, no characters to output
16115                              <1>       ;mov    r1,0f / no, put offset in sleep arg
16116                              <1>         ;jsr    r0,sleep; 0:.. / put tty output process to sleep
16117                              <1>         ;br     1b / try to calm it down again
16118                              <1> ;1:
16119                              <1>         ;mov    (sp)+,r1
16120                              <1>         ;mov    (sp)+,r2 / restore registers
16121                              <1>       ;mov    (r2)+,r3 / put reader control status in r3
16122                              <1>         ;beq    1f / if 0, 1f
16123                              <1>         ;mov    r3,rcsr(r1) / move r.c. status to reader
16124                              <1>         ;                  / control status register
16125                              <1> ;1:
16126                              <1>         ;mov    (r2)+,r3 / move pointer control status to r3
16127                              <1>         ;beq    1f / if 0 1f
```

```
16128                                 <1>         ;mov    r3,tcsr(r1) / move p.c. status to printer
16129                                 <1>       ;               / control status reg
16130                                 <1> ;1:
16131                                 <1>         ;mov    (r2)+,tty+4(r1) / move to flag byte of tty block
16132                                 <1>         ;jmp     sysret2 / return to user
16133                                 <1>
16134                                 <1> sysgtty: ; < get tty status >
16135                                 <1>       ; 23/11/2015
16136                                 <1>       ; 29/10/2015
16137                                 <1>       ; 17/10/2015
16138                                 <1>       ; 28/06/2015 (Retro UNIX 386 v1 - Beginning)
16139                                 <1>       ; 30/05/2013 - 12/07/2014 (Retro UNIX 8086 v1)
16140                                 <1>       ;
16141                                 <1>       ; 'sysgtty' gets the status of tty in question.
16142                                 <1>       ; It stores in the three words addressed by it's argument
16143                                 <1>       ; the status of the typewriter whose file descriptor
16144                                 <1>       ; in (u.r0).
16145                                 <1>       ;
16146                                 <1>       ; Calling sequence:
16147                                 <1>       ;     sysgtty; arg
16148                                 <1>       ; Arguments:
16149                                 <1>       ;     arg - address of 3 words destination of the status
16150                                 <1>       ; Inputs: ((*u.r0 - file descriptor))
16151                                 <1>       ; Outputs: ((status in address which is pointed to by arg))
16152                                 <1>       ; ..........................................................
16153                                 <1>       ;
16154                                 <1>       ; Retro UNIX 8086 v1 modification:
16155                                 <1>       ;     'sysgtty' system call will return status of tty
16156                                 <1>       ;     (keyboard, serial port and video page status)
16157                                 <1>       ;      in following manner:
16158                                 <1>       ;
16159                                 <1>       ; Inputs:
16160                                 <1>       ;     BX = 0 --> means
16161                                 <1>       ;          CH = 0 -->     'return status of the console tty'
16162                                 <1>       ;                      for (current) process
16163                                 <1>       ;          CL = 0 --> return keyboard status (tty 0 to 9)
16164                                 <1>       ;          CL = 1 --> return video page status (tty 0 to 7)
16165                                 <1>       ;          CL = 1 --> return serial port status (tty 8 & 9)
16166                                 <1>       ;          CH > 0 -->     tty number + 1
16167                                 <1>       ;
16168                                 <1>       ;     BX > 0 --> points to name of tty
16169                                 <1>       ;          CL = 0 --> return keyboard status
16170                                 <1>       ;          CL = 1 --> return video page status
16171                                 <1>       ;          CH = undefined
16172                                 <1>       ;
16173                                 <1>       ; Outputs:
16174                                 <1>       ;     cf = 0 ->
16175                                 <1>       ;
16176                                 <1>       ;          AL = tty number from 0 to 9
16177                                 <1>       ;             (0 to 7 is also the video page of the tty)
16178                                 <1>       ;          AH = 0 if the tty is free/unused
16179                                 <1>       ;          AH = the process number of the caller
16180                                 <1>       ;          AH = FFh if the tty is locked by another process
16181                                 <1>       ;
16182                                 <1>       ;       (if calling is for serial port status)
16183                                 <1>       ;          BX = serial port status if tty number is 8 or 9
16184                                 <1>       ;             (BH = modem status, BL = Line status)
16185                                 <1>       ;          CX = 0FFFFh (if data is ready)
16186                                 <1>       ;          CX = 0 (if data is not ready or undefined)
16187                                 <1>       ;
16188                                 <1>       ;       (if calling is for keyboard status)
16189                                 <1>       ;          BX = current character in tty/keyboard buffer
16190                                 <1>       ;             (BH = scan code, BL = ascii code)
16191                                 <1>       ;             (BX=0 if there is not a waiting character)
16192                                 <1>       ;          CX  is undefined
16193                                 <1>       ;
16194                                 <1>       ;       (if calling is for video page status)
16195                                 <1>       ;          BX = cursor position on the video page
16196                                 <1>       ;             if tty number < 8
16197                                 <1>       ;             (BH = row, BL = column)
16198                                 <1>       ;          CX = current character (in cursor position)
16199                                 <1>       ;             on the video page of the tty
16200                                 <1>       ;             if tty number < 8
16201                                 <1>       ;             (CH = color, CL = character)
16202                                 <1>       ;
16203                                 <1>       ;     cf = 1 means error (requested tty is not ready)
16204                                 <1>       ;
16205                                 <1>       ;          AH = FFh if the caller is not owner of
16206                                 <1>       ;              specified tty or console tty
16207                                 <1>       ;          AL = tty number (0FFh if it does not exist)
16208                                 <1>       ;          BX, CX are undefined if cf = 1
16209                                 <1>       ;
16210                                 <1>       ;       (If tty number is 8 or 9)
16211                                 <1>       ;          AL = tty number
16212                                 <1>       ;          AH = the process number of the caller
16213                                 <1>       ;          BX = serial port status
16214                                 <1>       ;             (BH = modem status, BL = Line status)
16215                                 <1>       ;          CX = 0
16216                                 <1>       ;
16217                                 <1>
16218                                 <1> gtty:   ; get (requested) tty number
16219                                 <1>       ; 17/10/2015
16220                                 <1>       ; 28/06/2015 (Retro UNIX 386 v1 - 32 bit modifications)
16221                                 <1>       ; 30/05/2013 - 12/07/2014
16222                                 <1>       ; Retro UNIX 8086 v1 modification !
16223                                 <1>       ;
16224                                 <1>       ; ((Modified regs: eAX, eBX, eCX, eDX, eSI, eDI, eBP))
16225                                 <1>       ;
16226                                 <1>       ; 28/06/2015 (32 bit modifications)
16227                                 <1>       ; 16/01/2014
16228 0000475E 31C0                  <1>       xor    eax, eax
16229 00004760 6648                  <1>       dec    ax ; 17/10/2015
16230 00004762 A3[48740000]          <1>       mov    [u.r0], eax ; 0FFFFh
16231 00004767 80F901                <1>       cmp    cl, 1
16232 0000476A 760F                  <1>       jna    short sysgtty_0
```

```
16233                                  <1> sysgtty_invp:
16234                                  <1>       ; 28/06/2015
16235 0000476C C705[9D740000]1700-     <1>         mov     dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
16236 00004774 0000                    <1>
16237 00004776 E9BFF8FFFF              <1>       jmp     error
16238                                  <1> sysgtty_0:
16239 0000477B 21DB                    <1>       and     ebx, ebx
16240 0000477D 7430                    <1>       jz      short sysgtty_1
16241                                  <1>       ;
16242 0000477F 891D[60740000]          <1>       mov     [u.namep], ebx
16243 00004785 6651                    <1>       push    cx ; 23/11/2015
16244 00004787 E86A070000              <1>       call    namei
16245 0000478C 6659                    <1>       pop     cx ; 23/11/2015
16246 0000478E 7210                    <1>       jc      short sysgtty_inv_dn ; 28/06/2015
16247                                  <1>       ;
16248 00004790 6683F801                <1>       cmp     ax, 1
16249 00004794 7622                    <1>       jna     short sysgtty_2
16250 00004796 6683E80A                <1>       sub     ax, 10
16251 0000479A 6683F809                <1>       cmp     ax, 9
16252                                  <1>       ;ja     short sysgtty_inv_dn
16253                                  <1>       ;mov    ch, al
16254                                  <1>       ;jmp    short sysgtty_4
16255                                  <1>       ; 23/11/2015
16256 0000479E 7629                    <1>       jna     short sysgtty_4
16257                                  <1> sysgtty_inv_dn:
16258                                  <1>       ; 28/06/2015
16259                                  <1>       ; Invalid device name (not a tty) ! error
16260                                  <1>       ; (Device is not a tty or device name not found)
16261 000047A0 C705[9D740000]1800-     <1>       mov     dword [u.error], ERR_INV_DEV_NAME
16262 000047A8 0000                    <1>
16263 000047AA E98BF8FFFF              <1>       jmp     error
16264                                  <1> sysgtty_1:
16265                                  <1>       ; 16/01/2014
16266 000047AF 80FD0A                  <1>       cmp     ch, 10
16267 000047B2 77B8                    <1>       ja      short sysgtty_invp ; 28/06/2015
16268 000047B4 FECD                    <1>       dec     ch ; 0 -> FFh (negative)
16269 000047B6 790F                    <1>       jns     short sysgtty_3 ; not negative
16270                                  <1>       ;
16271                                  <1> sysgtty_2:
16272                                  <1>       ; get tty number of console tty
16273 000047B8 8A25[97740000]          <1>       mov     ah, [u.uno]
16274                                  <1>       ; 28/06/2015
16275 000047BE 0FB6DC                  <1>       movzx   ebx, ah
16276 000047C1 8AAB[95710000]          <1>       mov     ch, [ebx+p.ttyc-1]
16277                                  <1> sysgtty_3:
16278 000047C7 88E8                    <1>       mov     al, ch
16279                                  <1> sysgtty_4:
16280 000047C9 A2[48740000]            <1>       mov     [u.r0], al
16281                                  <1>       ; 28/06/2015
16282                                  <1>       ;cmp    al, 9
16283                                  <1>       ;ja     short sysgtty_invp
16284 000047CE 8B2D[44740000]          <1>       mov     ebp, [u.usp]
16285                                  <1>       ; 23/11/2015
16286 000047D4 20C9                    <1>       and     cl, cl
16287 000047D6 7436                    <1>       jz      short sysgtty_6 ; keyboard status
16288 000047D8 3C08                    <1>       cmp     al, 8 ; cmp ch, 8
16289 000047DA 7232                    <1>       jb      short sysgtty_6 ; video page status
16290                                  <1>       ; serial port status
16291                                  <1>       ; 12/07/2014
16292                                  <1>       ;mov    dx, 0
16293                                  <1>       ;je     short sysgtty_5
16294                                  <1>       ;inc    dl
16295                                  <1> ;sysgtty_5:
16296                                  <1>       ; 28/06/2015
16297 000047DC 2C08                    <1>       sub     al, 8
16298 000047DE E8FCF2FFFF              <1>       call    sp_status ; serial (COM) port (line) status
16299                                  <1>       ; AL = Line status, AH = Modem status
16300 000047E3 66894510                <1>       mov     [ebp+16], ax ; serial port status (in EBX)
16301 000047E7 8A25[97740000]          <1>       mov     ah, [u.uno]
16302 000047ED 8825[49740000]          <1>         mov     [u.r0+1], ah
16303 000047F3 66C745180000            <1>       mov     word [ebp+24], 0 ; data status (0 = not ready)
16304                                  <1>                               ; (in ECX)
16305 000047F9 A880                    <1>       test    al, 80h
16306 000047FB 7565                    <1>       jnz     short sysgtty_dnr_err ; 29/06/2015
16307 000047FD A801                    <1>       test    al, 1
16308 000047FF 0F8455F8FFFF            <1>       jz      sysret
16309 00004805 66FF4D18                <1>       dec     word [ebp+24] ; data status (FFFFh = ready)
16310 00004809 E94CF8FFFF              <1>       jmp     sysret
16311                                  <1> sysgtty_6:
16312 0000480E A2[9C740000]            <1>       mov     [u.ttyn], al ; tty number
16313                                  <1>       ;movzx ebx, al
16314 00004813 88C3                    <1>       mov     bl, al ; tty number (0 to 9)
16315 00004815 D0E3                    <1>       shl     bl, 1  ; aligned to word
16316                                  <1>       ; 22/04/2014 - 29/06/2015
16317 00004817 81C3[B4700000]          <1>         add     ebx, ttyl
16318 0000481D 8A23                    <1>       mov     ah, [ebx]
16319 0000481F 3A25[97740000]          <1>       cmp     ah, [u.uno]
16320 00004825 7404                    <1>       je      short sysgtty_7
16321 00004827 20E4                    <1>       and     ah, ah
16322                                  <1>       ;jz     short sysgtty_7
16323 00004829 7506                    <1>       jnz     short sysgtty_8
16324                                  <1>       ;mov    ah, 0FFh
16325                                  <1> sysgtty_7:
16326 0000482B 8825[49740000]          <1>         mov     [u.r0+1], ah
16327                                  <1> sysgtty_8:
16328 00004831 08C9                    <1>       or      cl, cl
16329 00004833 7510                    <1>       jnz     short sysgtty_9
16330 00004835 B001                    <1>       mov     al, 1  ; test a key is available
16331 00004837 E8621C0000              <1>       call    getc
16332 0000483C 66894510                <1>       mov     [ebp+16], ax ; bx, character
16333 00004840 E915F8FFFF              <1>       jmp     sysret
16334                                  <1> sysgtty_9:
16335 00004845 8A1D[9C740000]          <1>       mov     bl, [u.ttyn]
16336                                  <1>       ; bl = video page number
16337 0000484B E8CF1D0000              <1>       call    get_cpos
```

```
16338                                  <1>       ; dx = cursor position
16339 00004850 66895510                <1>       mov   [ebp+16], dx ; bx
16340                                  <1>       ;mov  bl, [u.ttyn]
16341                                  <1>       ; bl = video page number
16342 00004854 E8D71D0000              <1>       call  read_ac_current
16343                                  <1>       ; ax = character and attribute/color
16344 00004859 66894518                <1>       mov   [ebp+24], ax ; cx
16345 0000485D E9F8F7FFFF              <1>       jmp   sysret
16346                                  <1> sysgtty_dnr_err:
16347                                  <1>       ; 'device not responding !' error
16348                                  <1>       ;mov  dword [u.error], ERR_TIME_OUT ; 25
16349 00004862 C705[9D740000]1900-     <1>       mov   dword [u.error], ERR_DEV_NOT_RESP ;  25
16350 0000486A 0000                    <1>
16351 0000486C E9C9F7FFFF              <1>       jmp   error
16352                                  <1>
16353                                  <1> ; Original UNIX v1 'sysgtty' routine:
16354                                  <1> ; sysgtty:
16355                                  <1>         ;jsr    r0,gtty / r1 will have offset to tty block,
16356                                  <1>       ;            / r2 has destination
16357                                  <1>         ;mov    rcsr(r1),(r2)+ / put reader control status
16358                                  <1>       ;                      / in 1st word of dest
16359                                  <1>         ;mov    tcsr(r1),(r2)+ / put printer control status
16360                                  <1>       ;                      / in 2nd word of dest
16361                                  <1>         ;mov    tty+4(r1),(r2)+ / put mode in 3rd word
16362                                  <1>         ;jmp    sysret2 / return to user
16363                                  <1>
16364                                  <1> ; Original UNIX v1 'gtty' routine:
16365                                  <1> ; gtty:
16366                                  <1>         ;jsr    r0,arg; u.off / put first arg in u.off
16367                                  <1>         ;mov    *u.r0,r1 / put file descriptor in r1
16368                                  <1>         ;jsr    r0,getf / get the i-number of the file
16369                                  <1>         ;tst    r1 / is it open for reading
16370                                  <1>         ;bgt    1f / yes
16371                                  <1>         ;neg    r1 / no, i-number is negative,
16372                                  <1>       ;          / so make it positive
16373                                  <1> ;1:
16374                                  <1>         ;sub    $14.,r1 / get i-number of tty0
16375                                  <1>         ;cmp    r1,$ntty-1 / is there such a typewriter
16376                                  <1>         ;bhis   error9 / no, error
16377                                  <1>         ;asl    r1 / 0%2
16378                                  <1>         ;asl    r1 / 0%4 / yes
16379                                  <1>         ;asl    r1 / 0%8 / multiply by 8 so r1 points to
16380                                  <1>       ;            ; / tty block
16381                                  <1>         ;mov    u.off,r2 / put argument in r2
16382                                  <1>         ;rts    r0 / return
16383                                      %include 'u2.s'        ; 11/05/2015
16384                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS2.INC
16385                                  <1> ; Last Modification: 03/01/2016
16386                                  <1> ; ----------------------------------------------------------------------------
16387                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
16388                                  <1> ; (v0.1 - Beginning: 11/07/2012)
16389                                  <1> ;
16390                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
16391                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
16392                                  <1> ; <Bell Laboratories (17/3/1972)>
16393                                  <1> ; <Preliminary Release of UNIX Implementation Document>
16394                                  <1> ;
16395                                  <1> ; Retro UNIX 8086 v1 - U2.ASM (24/03/2014) //// UNIX v1 -> u2.s
16396                                  <1> ;
16397                                  <1> ; ****************************************************************************
16398                                  <1>
16399                                  <1> syslink:
16400                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16401                                  <1>       ; 19/06/2013 (Retro UNIX 8086 v1)
16402                                  <1>       ;
16403                                  <1>       ; 'syslink' is given two arguments, name 1 and name 2.
16404                                  <1>       ; name 1 is a file that already exists. name 2 is the name
16405                                  <1>       ; given to the entry that will go in the current directory.
16406                                  <1>       ; name2 will then be a link to the name 1 file. The i-number
16407                                  <1>       ; in the name 2 entry of current directory is the same
16408                                  <1>       ; i-number for the name 1 file.
16409                                  <1>       ;
16410                                  <1>       ; Calling sequence:
16411                                  <1>       ;     syslink; name 1; name 2
16412                                  <1>       ; Arguments:
16413                                  <1>       ;     name 1 - file name to which link will be created.
16414                                  <1>       ;     name 2 - name of entry in current directory that
16415                                  <1>       ;              links to name 1.
16416                                  <1>       ; Inputs: -
16417                                  <1>       ; Outputs: -
16418                                  <1>       ; ..........................................................
16419                                  <1>       ;
16420                                  <1>       ; Retro UNIX 8086 v1 modification:
16421                                  <1>       ;      'syslink' system call has two arguments; so,
16422                                  <1>       ;     * 1st argument, name 1 is pointed to by BX register
16423                                  <1>       ;     * 2nd argument, name 2 is pointed to by CX register
16424                                  <1>       ;
16425                                  <1>             ; / name1, name2
16426                                  <1>             ;jsr r0,arg2 / u.namep has 1st arg u.off has 2nd
16427 00004871 891D[60740000]          <1>       mov   [u.namep], ebx
16428 00004877 51                      <1>       push  ecx
16429 00004878 E879060000              <1>       call  namei
16430                                  <1>             ; jsr r0,namei / find the i-number associated with
16431                                  <1>                        ; / the 1st path name
16432                                  <1>       ;;and  ax, ax
16433                                  <1>       ;;jz   error ; File not found
16434                                  <1>       ;jc    error
16435                                  <1>             ; br error9 / cannot be found
16436 0000487D 730F                    <1>       jnc   short syslink0
16437                                  <1>       ;pop   ecx
16438                                  <1>       ; 'file not found !' error
16439 0000487F C705[9D740000]0C00-      <1>       mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
16440 00004887 0000                    <1>
16441 00004889 E9ACF7FFFF              <1>       jmp   error
16442                                  <1> syslink0:
```

```
16443 0000488E E8530E0000          <1>       call   iget
16444                              <1>              ; jsr r0,iget / get the i-node into core
16445 00004893 8F05[60740000]      <1>       pop    dword [u.namep] ; ecx
16446                              <1>              ; mov (sp)+,u.namep / u.namep points to 2nd name
16447 00004899 6650                <1>       push   ax
16448                              <1>              ; mov r1,-(sp) / put i-number of name1 on the stack
16449                              <1>                             ; / (a link to this file is to be created)
16450 0000489B 66FF35[2E740000]    <1>       push   word [cdev]
16451                              <1>              ; mov cdev,-(sp) / put i-nodes device on the stack
16452 000048A2 E855000000          <1>       call   isdir
16453                              <1>              ; jsr r0,isdir / is it a directory
16454 000048A7 E84A060000          <1>       call   namei
16455                              <1>              ; jsr r0,namei / no, get i-number of name2
16456                              <1>       ;jnc   error
16457                              <1>              ; br .+4   / not found
16458                              <1>                     ; / so r1 = i-number of current directory
16459                              <1>                     ; / ii = i-number of current directory
16460                              <1>              ; br error9 / file already exists., error
16461 000048AC 720F                <1>       jc     short syslink1
16462                              <1>       ; pop ax
16463                              <1>       ; pop ax
16464                              <1>       ; 'file exists !' error
16465 000048AE C705[9D740000]0E00- <1>       mov    dword [u.error], ERR_FILE_EXISTS ; 14
16466 000048B6 0000                <1>
16467 000048B8 E97DF7FFFF          <1>       jmp    error
16468                              <1> syslink1:
16469 000048BD 6659                <1>       pop    cx
16470                              <1>       ;cmp   cx, [cdev]
16471 000048BF 3A0D[2E740000]      <1>       cmp    cl, [cdev]
16472                              <1>       ;jne   error
16473                              <1>              ; cmp (sp)+,cdev / u.dirp now points to
16474                              <1>                              ; / end of current directory
16475                              <1>                ; bne error9
16476 000048C5 740F                <1>       je     short syslink2
16477                              <1>       ; 'not same drive !' error
16478 000048C7 C705[9D740000]1500- <1>       mov    dword [u.error],  ERR_DRV_NOT_SAME ; 21
16479 000048CF 0000                <1>
16480 000048D1 E964F7FFFF          <1>       jmp    error
16481                              <1> syslink2:
16482 000048D6 6658                <1>       pop    ax
16483 000048D8 6650                <1>       push   ax
16484 000048DA 66A3[7A740000]      <1>       mov    [u.dirbuf], ax
16485                              <1>              ; mov (sp),u.dirbuf / i-number of name1 into u.dirbuf
16486 000048E0 E8A8000000          <1>       call   mkdir
16487                              <1>              ; jsr r0,mkdir / make directory entry for name2
16488                              <1>                             ; / in current directory
16489 000048E5 6658                <1>       pop    ax
16490                              <1>              ; mov (sp)+,r1 / r1 has i-number of name1
16491 000048E7 E8FA0D0000          <1>       call   iget
16492                              <1>              ; jsr r0,iget / get i-node into core
16493 000048EC FE05[18710000]      <1>       inc    byte [i.nlks]
16494                              <1>              ; incb i.nlks / add 1 to its number of links
16495 000048F2 E8020F0000          <1>       call   setimod
16496                              <1>              ; jsr r0,setimod / set the i-node modified flag
16497 000048F7 E95EF7FFFF          <1>       jmp    sysret
16498                              <1>
16499                              <1> isdir:
16500                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
16501                              <1>       ; 04/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
16502                              <1>       ;
16503                              <1>       ; 'isdir' check to see if the i-node whose i-number is in r1
16504                              <1>       ;  is a directory. If it is, an error occurs, because 'isdir'
16505                              <1>       ;  called by syslink and sysunlink to make sure directories
16506                              <1>       ;  are not linked. If the user is the super user (u.uid=0),
16507                              <1>       ; 'isdir' does not bother checking. The current i-node
16508                              <1>       ;  is not disturbed.
16509                              <1>       ;
16510                              <1>       ; INPUTS ->
16511                              <1>       ;    r1 - contains the i-number whose i-node is being checked.
16512                              <1>       ;    u.uid - user id
16513                              <1>       ; OUTPUTS ->
16514                              <1>       ;    r1 - contains current i-number upon exit
16515                              <1>       ;       (current i-node back in core)
16516                              <1>       ;
16517                              <1>       ; ((AX = R1))
16518                              <1>       ;
16519                              <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
16520                              <1>       ;
16521                              <1>
16522                              <1>       ; / if the i-node whose i-number is in r1 is a directory
16523                              <1>       ; / there is an error unless super user made the call
16524                              <1>
16525 000048FC 803D[94740000]00    <1>       cmp    byte [u.uid], 0
16526                              <1>              ; tstb u.uid / super user
16527 00004903 762D                <1>       jna    short isdir1
16528                              <1>              ; beq 1f / yes, don't care
16529 00004905 66FF35[2A740000]    <1>       push   word [ii]
16530                              <1>              ; mov ii,-(sp) / put current i-number on stack
16531 0000490C E8D50D0000          <1>       call   iget
16532                              <1>              ; jsr r0,iget / get i-node into core (i-number in r1)
16533 00004911 66F705[16710000]00- <1>       test   word [i.flgs], 4000h ; Bit 14 : Directory flag
16534 00004919 40                  <1>
16535                              <1>              ; bit $40000,i.flgs / is it a directory
16536                              <1>       ;jnz   error
16537                              <1>              ; bne error9 / yes, error
16538 0000491A 740F                <1>       jz     short isdir0
16539 0000491C C705[9D740000]0B00- <1>       mov    dword [u.error], ERR_NOT_FILE  ; 11 ; ERR_DIR_ACCESS
16540 00004924 0000                <1>
16541                              <1>                             ; 'permission denied !' error
16542                              <1>       ; pop  ax
16543 00004926 E90FF7FFFF          <1>       jmp    error
16544                              <1> isdir0:
16545 0000492B 6658                <1>       pop    ax
16546                              <1>              ; mov (sp)+,r1 / no, put current i-number in r1 (ii)
16547 0000492D E8B40D0000          <1>       call   iget
```

```
16548                                  <1>             ; jsr r0,iget / get it back in
16549                                  <1> isdir1: ; 1:
16550 00004932 C3                      <1>       retn
16551                                  <1>             ; rts r0
16552                                  <1>
16553                                  <1> sysunlink:
16554                                  <1>       ; 04/12/2015 (14 byte file names)
16555                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16556                                  <1>       ; 19/06/2013 (Retro UNIX 8086 v1)
16557                                  <1>       ;
16558                                  <1>       ; 'sysunlink' removes the entry for the file pointed to by
16559                                  <1>       ; name from its directory. If this entry was the last link
16560                                  <1>       ; to the file, the contents of the file are freed and the
16561                                  <1>       ; file is destroyed. If, however, the file was open in any
16562                                  <1>       ; process, the actual destruction is delayed until it is
16563                                  <1>       ; closed, even though the directory entry has disappeared.
16564                                  <1>       ;
16565                                  <1>       ; The error bit (e-bit) is set to indicate that the file
16566                                  <1>       ; does not exist or that its directory can not be written.
16567                                  <1>       ; Write permission is not required on the file itself.
16568                                  <1>       ; It is also illegal to unlink a directory (except for
16569                                  <1>       ; the superuser).
16570                                  <1>       ;
16571                                  <1>       ; Calling sequence:
16572                                  <1>       ;     sysunlink; name
16573                                  <1>       ; Arguments:
16574                                  <1>       ;     name - name of directory entry to be removed
16575                                  <1>       ; Inputs: -
16576                                  <1>       ; Outputs: -
16577                                  <1>       ; ..........................................................
16578                                  <1>       ;
16579                                  <1>       ; Retro UNIX 8086 v1 modification:
16580                                  <1>       ;      The user/application program puts address of the name
16581                                  <1>       ;        in BX register as 'sysunlink' system call argument.
16582                                  <1>
16583                                  <1>       ; / name - remove link name
16584 00004933 891D[60740000]          <1>       mov   [u.namep], ebx
16585                                  <1>             ;jsr r0,arg; u.namep / u.namep points to name
16586 00004939 E8B8050000              <1>       call  namei
16587                                  <1>             ; jsr r0,namei / find the i-number associated
16588                                  <1>                           ; / with the path name
16589                                  <1>       ;jc   error
16590                                  <1>             ; br error9 / not found
16591 0000493E 730F                    <1>       jnc   short sysunlink1
16592                                  <1>       ; 'file not found !' error
16593 00004940 C705[9D740000]0C00-     <1>       mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
16594 00004948 0000                    <1>
16595 0000494A E9EBF6FFFF              <1>       jmp   error
16596                                  <1> sysunlink1:
16597 0000494F 6650                    <1>       push  ax
16598                                  <1>             ; mov r1,-(sp) / put its i-number on the stack
16599 00004951 E8A6FFFFFF              <1>       call  isdir
16600                                  <1>             ; jsr r0,isdir / is it a directory
16601 00004956 6631C0                  <1>       xor   ax, ax
16602 00004959 66A3[7A740000]          <1>       mov   [u.dirbuf], ax ; 0
16603                                  <1>             ; clr u.dirbuf / no, clear the location that will
16604                                  <1>                      ; / get written into the i-number portion
16605                                  <1>                    ; / of the entry
16606 0000495F 832D[64740000]10        <1>       sub   dword [u.off], 16 ; 04/12/2015 (10 -> 16)
16607                                  <1>             ; sub $10.,u.off / move u.off back 1 directory entry
16608 00004966 E86E000000              <1>       call  wdir
16609                                  <1>             ; jsr r0,wdir / free the directory entry
16610 0000496B 6658                    <1>       pop   ax
16611                                  <1>             ; mov (sp)+,r1 / get i-number back
16612 0000496D E8740D0000              <1>       call  iget
16613                                  <1>             ; jsr r0,iget / get i-node
16614 00004972 E8820E0000              <1>       call  setimod
16615                                  <1>             ; jsr r0,setimod / set modified flag
16616 00004977 FE0D[18710000]          <1>       dec   byte [i.nlks]
16617                                  <1>             ; decb i.nlks / decrement the number of links
16618 0000497D 0F85D7F6FFFF            <1>       jnz   sysret
16619                                  <1>             ; bgt sysret9 / if this was not the last link
16620                                  <1>                           ; / to file return
16621                                  <1>       ; AX = r1 = i-number
16622 00004983 E893090000              <1>       call  anyi
16623                                  <1>             ; jsr r0,anyi / if it was, see if anyone has it open.
16624                                  <1>                     ; / Then free contents of file and destroy it.
16625 00004988 E9CDF6FFFF              <1>       jmp   sysret
16626                                  <1>             ; br sysret9
16627                                  <1>
16628                                  <1> mkdir:
16629                                  <1>       ; 04/12/2015 (14 byte directory names)
16630                                  <1>       ; 12/10/2015
16631                                  <1>       ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
16632                                  <1>       ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
16633                                  <1>       ;
16634                                  <1>       ; 'mkdir' makes a directory entry from the name pointed to
16635                                  <1>       ; by u.namep into the current directory.
16636                                  <1>       ;
16637                                  <1>       ; INPUTS ->
16638                                  <1>       ;    u.namep - points to a file name
16639                                  <1>       ;                  that is about to be a directory entry.
16640                                  <1>       ;    ii - current directory's i-number.
16641                                  <1>       ; OUTPUTS ->
16642                                  <1>       ;    u.dirbuf+2 - u.dirbuf+10 - contains file name.
16643                                  <1>       ;    u.off - points to entry to be filled
16644                                  <1>       ;           in the current directory
16645                                  <1>       ;    u.base - points to start of u.dirbuf.
16646                                  <1>       ;    r1 - contains i-number of current directory
16647                                  <1>       ;
16648                                  <1>       ; ((AX = R1)) output
16649                                  <1>       ;
16650                                  <1>       ;    (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
16651                                  <1>         ;    ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
16652                                  <1>       ;
```

```
16653                                  <1>
16654                                  <1>       ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
16655 0000498D 31C0                    <1>       xor    eax, eax
16656 0000498F BF[7C740000]            <1>       mov     edi, u.dirbuf+2
16657 00004994 89FE                    <1>       mov    esi, edi
16658 00004996 AB                      <1>       stosd
16659 00004997 AB                      <1>       stosd
16660                                  <1>       ; 04/12/2015 (14 byte directory names)
16661 00004998 AB                      <1>       stosd
16662 00004999 66AB                    <1>       stosw
16663                                  <1>             ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
16664 0000499B 89F7                    <1>       mov    edi, esi ; offset to u.dirbuf
16665                                  <1>       ; 12/10/2015 ([u.namep] -> ebp)
16666                                  <1>       ;mov   ebp, [u.namep]
16667 0000499D E899060000              <1>       call   trans_addr_nmbp ; convert virtual address to physical
16668                                  <1>             ; esi = physical address (page start + offset)
16669                                  <1>             ; ecx = byte count in the page (1 - 4096)
16670                                  <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
16671                                  <1>             ; mov u.namep,r2 / r2 points to name of directory entry
16672                                  <1>             ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
16673                                  <1> mkdir_1: ; 1:
16674 000049A2 45                      <1>       inc    ebp ; 12/10/2015
16675                                  <1>       ;
16676                                  <1>       ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
16677                                  <1>        ; 01/08/2013
16678 000049A3 AC                      <1>       lodsb
16679                                  <1>             ; movb (r2)+,r1 / move character in name to r1
16680 000049A4 20C0                    <1>       and    al, al
16681 000049A6 7427                    <1>       jz     short mkdir_3
16682                                  <1>             ; beq 1f / if null, done
16683 000049A8 3C2F                    <1>       cmp    al, '/'
16684                                  <1>             ; cmp r1,$'/ / is it a "/"?
16685 000049AA 7414                    <1>       je     short mkdir_err
16686                                  <1>       ;je    error
16687                                  <1>             ; beq error9 / yes, error
16688                                  <1>       ; 12/10/2015
16689 000049AC 6649                    <1>       dec    cx
16690 000049AE 7505                    <1>       jnz    short mkdir_2
16691                                  <1>       ; 12/10/2015 ([u.namep] -> ebp)
16692 000049B0 E88C060000              <1>       call   trans_addr_nm ; convert virtual address to physical
16693                                  <1>             ; esi = physical address (page start + offset)
16694                                  <1>             ; ecx = byte count in the page
16695                                  <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
16696                                  <1> mkdir_2:
16697 000049B5 81FF[8A740000]          <1>       cmp     edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)
16698                                  <1>             ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
16699                                  <1>                                    ; / a char?
16700 000049BB 74E5                    <1>       je     short mkdir_1
16701                                  <1>             ; beq 1b / yes, go back
16702 000049BD AA                      <1>       stosb
16703                                  <1>             ; movb r1,(r3)+ / no, put the char in the u.dirbuf
16704 000049BE EBE2                    <1>       jmp    short mkdir_1
16705                                  <1>             ; br 1b / get next char
16706                                  <1> mkdir_err:
16707                                  <1>       ; 17/06/2015
16708 000049C0 C705[9D740000]1300-     <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
16709 000049C8 0000                    <1>
16710 000049CA E96BF6FFFF              <1>       jmp    error
16711                                  <1>
16712                                  <1> mkdir_3: ; 1:
16713 000049CF A1[5C740000]            <1>       mov    eax, [u.dirp]
16714 000049D4 A3[64740000]            <1>       mov    [u.off], eax
16715                                  <1>             ; mov u.dirp,u.off / pointer to empty current directory
16716                                  <1>                                ; / slot to u.off
16717                                  <1> wdir: ; 29/04/2013
16718 000049D9 C705[68740000]-         <1>         mov     dword [u.base], u.dirbuf
16719 000049DF [7A740000]              <1>
16720                                  <1>             ; mov $u.dirbuf,u.base / u.base points to created file name
16721 000049E3 C705[6C740000]1000-     <1>         mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
16722 000049EB 0000                    <1>
16723                                  <1>             ; mov $10.,u.count / u.count = 10
16724 000049ED 66A1[2A740000]          <1>       mov    ax, [ii]
16725                                  <1>             ; mov ii,r1 / r1 has i-number of current directory
16726 000049F3 B201                    <1>       mov    dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
16727 000049F5 E8C70D0000              <1>       call   access
16728                                  <1>             ; jsr r0,access; 1 / get i-node and set its file up
16729                                  <1>                               ; / for writing
16730                                  <1>       ; AX = i-number of current directory
16731                                  <1>       ; 01/08/2013
16732 000049FA FE05[AF740000]          <1>       inc     byte [u.kcall] ; the caller is 'mkdir' sign
16733 00004A00 E8B7100000              <1>       call   writei
16734                                  <1>             ; jsr r0,writei / write into directory
16735 00004A05 C3                      <1>       retn
16736                                  <1>             ; rts r0
16737                                  <1>
16738                                  <1> sysexec:
16739                                  <1>       ; 23/10/2015
16740                                  <1>       ; 19/10/2015
16741                                  <1>       ; 18/10/2015
16742                                  <1>       ; 10/10/2015
16743                                  <1>       ; 26/08/2015
16744                                  <1>       ; 05/08/2015
16745                                  <1>       ; 29/07/2015
16746                                  <1>       ; 25/07/2015
16747                                  <1>       ; 24/07/2015
16748                                  <1>       ; 21/07/2015
16749                                  <1>       ; 20/07/2015
16750                                  <1>       ; 02/07/2015
16751                                  <1>       ; 01/07/2015
16752                                  <1>       ; 25/06/2015
16753                                  <1>       ; 24/06/2015
16754                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16755                                  <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
16756                                  <1>       ;
16757                                  <1>       ; 'sysexec' initiates execution of a file whose path name if
```

```
16758                                  <1>       ; pointed to by 'name' in the sysexec call.
16759                                  <1>       ; 'sysexec' performs the following operations:
16760                                  <1>       ;    1. obtains i-number of file to be executed via 'namei'.
16761                                  <1>       ;    2. obtains i-node of file to be exceuted via 'iget'.
16762                                  <1>       ;    3. sets trap vectors to system routines.
16763                                  <1>       ;    4. loads arguments to be passed to executing file into
16764                                  <1>       ;      highest locations of user's core
16765                                  <1>       ;    5. puts pointers to arguments in locations immediately
16766                                  <1>       ;      following arguments.
16767                                  <1>       ;    6.      saves number of arguments in next location.
16768                                  <1>       ;    7. intializes user's stack area so that all registers
16769                                  <1>       ;      will be zeroed and the PS is cleared and the PC set
16770                                  <1>       ;      to core when 'sysret' restores registers
16771                                  <1>       ;      and does an rti.
16772                                  <1>       ;    8. inializes u.r0 and u.sp
16773                                  <1>       ;    9. zeros user's core down to u.r0
16774                                  <1>       ;   10.      reads executable file from storage device into core
16775                                  <1>       ;      starting at location 'core'.
16776                                  <1>       ;   11.      sets u.break to point to end of user's code with
16777                                  <1>       ;      data area appended.
16778                                  <1>       ;   12.      calls 'sysret' which returns control at location
16779                                  <1>       ;      'core' via 'rti' instruction.
16780                                  <1>       ;
16781                                  <1>       ; Calling sequence:
16782                                  <1>       ;      sysexec; namep; argp
16783                                  <1>       ; Arguments:
16784                                  <1>       ;      namep - points to pathname of file to be executed
16785                                  <1>       ;      argp  - address of table of argument pointers
16786                                  <1>       ;      argp1... argpn - table of argument pointers
16787                                  <1>       ;      argp1:<...0> ... argpn:<...0> - argument strings
16788                                  <1>       ; Inputs: (arguments)
16789                                  <1>       ; Outputs: -
16790                                  <1>       ; ..........................................................
16791                                  <1>       ;
16792                                  <1>       ; Retro UNIX 386 v1 modification:
16793                                  <1>       ;      User application runs in it's own virtual space
16794                                  <1>       ;      which is izolated from kernel memory (and other
16795                                  <1>       ;      memory pages) via 80386   paging in ring 3
16796                                  <1>       ;      privilige mode. Virtual start address is always 0.
16797                                  <1>       ;      User's core memory starts at linear address 400000h
16798                                  <1>       ;      (the end of the 1st 4MB).
16799                                  <1>       ;
16800                                  <1>       ; Retro UNIX 8086 v1 modification:
16801                                  <1>       ;      user/application segment and system/kernel segment
16802                                  <1>       ;      are different and sysenter/sysret/sysrele routines
16803                                  <1>       ;      are different (user's registers are saved to
16804                                  <1>       ;      and then restored from system's stack.)
16805                                  <1>       ;
16806                                  <1>       ;      NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
16807                                  <1>       ;            arguments which were in these registers;
16808                                  <1>       ;            but, it returns by putting the 1st argument
16809                                  <1>       ;            in 'u.namep' and the 2nd argument
16810                                  <1>       ;            on top of stack. (1st argument is offset of the
16811                                  <1>       ;            file/path name in the user's program segment.)
16812                                  <1>
16813                                  <1>       ;call  arg2
16814                                  <1>       ; * name - 'u.namep' points to address of file/path name
16815                                  <1>       ;          in the user's program segment ('u.segmnt')
16816                                  <1>       ;          with offset in BX register (as sysopen argument 1).
16817                                  <1>       ; * argp - sysexec argument 2 is in CX register
16818                                  <1>       ;          which is on top of stack.
16819                                  <1>       ;
16820                                  <1>             ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
16821                                  <1>
16822                                  <1>       ; 23/06/2015 (32 bit modifications)
16823                                  <1>
16824 00004A06 891D[60740000]         <1>       mov    [u.namep], ebx ; argument 1
16825                                  <1>         ; 18/10/2015
16826 00004A0C 890D[C8740000]         <1>       mov     [argv], ecx  ; * ; argument 2
16827 00004A12 E8DF040000             <1>       call   namei
16828                                  <1>             ; jsr r0,namei / namei returns i-number of file
16829                                  <1>                          ; / named in sysexec call in r1
16830                                  <1>       ;jc    error
16831                                  <1>             ; br error9
16832 00004A17 731E                   <1>       jnc    short sysexec_0
16833                                  <1>       ;
16834                                  <1>       ; 'file not found !' error
16835 00004A19 C705[9D740000]0C00-    <1>       mov    dword [u.error], ERR_FILE_NOT_FOUND
16836 00004A21 0000                   <1>
16837 00004A23 E912F6FFFF             <1>       jmp    error
16838                                  <1> sysexec_not_exf:
16839                                  <1>       ; 'not executable file !' error
16840 00004A28 C705[9D740000]1600-    <1>       mov    dword [u.error], ERR_NOT_EXECUTABLE
16841 00004A30 0000                   <1>
16842 00004A32 E903F6FFFF             <1>       jmp    error
16843                                  <1> sysexec_0:
16844 00004A37 E8AA0C0000             <1>       call   iget
16845                                  <1>             ; jsr r0,iget / get i-node for file to be executed
16846 00004A3C 66F705[16710000]10-    <1>         test    word [i.flgs], 10h
16847 00004A44 00                     <1>
16848                                  <1>             ; bit $20,i.flgs / is file executable
16849 00004A45 74E1                   <1>       jz     short sysexec_not_exf
16850                                  <1>       ;jz    error
16851                                  <1>             ; beq error9
16852                                  <1>       ;;
16853 00004A47 E819140000             <1>       call   iopen
16854                                  <1>             ; jsr r0,iopen / gets i-node for file with i-number
16855                                  <1>                           ; / given in r1 (opens file)
16856                                  <1>       ; AX = i-number of the file
16857 00004A4C 66F705[16710000]20-    <1>       test   word [i.flgs], 20h
16858 00004A54 00                     <1>
16859                                  <1>             ; bit $40,i.flgs / test user id on execution bit
16860 00004A55 7415                   <1>       jz     short sysexec_1
16861                                  <1>             ; beq 1f
16862 00004A57 803D[94740000]00       <1>       cmp    byte [u.uid], 0 ; 02/08/2013
```

```
16863                                 <1>             ; tstb u.uid / test user id
16864 00004A5E 760C                   <1>       jna   short sysexec_1
16865                                 <1>             ; beq 1f / super user
16866 00004A60 8A0D[19710000]         <1>       mov   cl, [i.uid]
16867 00004A66 880D[94740000]         <1>       mov   [u.uid], cl ; 02/08/2013
16868                                 <1>             ; movb i.uid,u.uid / put user id of owner of file
16869                                 <1>                           ; / as process user id
16870                                 <1> sysexec_1:
16871                                 <1>       ; 18/10/2215
16872                                 <1>       ; 10/10/2015
16873                                 <1>       ; 24/07/2015
16874                                 <1>       ; 21/07/2015
16875                                 <1>       ; 25/06/2015
16876                                 <1>       ; 24/06/2015
16877                                 <1>         ; Moving arguments to the end of [u.upage]
16878                                 <1>       ; (by regarding page borders in user's memory space)
16879                                 <1>       ;
16880                                 <1>       ; 10/10/2015
16881                                 <1>       ; 21/07/2015
16882 00004A6C 89E5                   <1>       mov   ebp, esp ; (**)
16883                                 <1>       ; 18/10/2015
16884 00004A6E 89EF                   <1>       mov   edi, ebp
16885 00004A70 B900010000             <1>       mov   ecx, MAX_ARG_LEN ; 256
16886                                 <1>       ;sub  edi, MAX_ARG_LEN ; 256
16887 00004A75 29CF                   <1>       sub   edi, ecx
16888 00004A77 89FC                   <1>       mov   esp, edi
16889 00004A79 31C0                   <1>       xor   eax, eax
16890 00004A7B A3[70740000]           <1>       mov   [u.nread], eax ; 0
16891 00004A80 49                     <1>       dec   ecx ; 256 - 1
16892 00004A81 890D[6C740000]         <1>       mov   [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
16893                                 <1>       ;mov  dword [u.count], MAX_ARG_LEN - 1 ; 255
16894                                 <1> sysexec_2:
16895 00004A87 8B35[C8740000]         <1>       mov   esi, [argv] ; 18/10/2015
16896 00004A8D E873020000             <1>       call  get_argp
16897 00004A92 B904000000             <1>       mov   ecx, 4 ; mov ecx, 4
16898                                 <1> sysexec_3:
16899 00004A97 21C0                   <1>       and   eax, eax
16900 00004A99 7456                   <1>       jz    short sysexec_6
16901                                 <1>       ; 18/10/2015
16902 00004A9B 010D[C8740000]         <1>       add   [argv], ecx ; 4
16903 00004AA1 66FF05[C6740000]       <1>       inc   word [argc]
16904                                 <1>       ;
16905 00004AA8 A3[68740000]           <1>       mov   [u.base], eax
16906                                 <1>       ; 23/10/2015
16907 00004AAD 66C705[AD740000]00-    <1>       mov   word [u.pcount], 0
16908 00004AB5 00                     <1>
16909                                 <1> sysexec_4:
16910 00004AB6 E8BC110000             <1>       call  cpass ; get a character from user's core memory
16911 00004ABB 750B                   <1>         jnz     short sysexec_5
16912                                 <1>             ; (max. 255 chars + null)
16913                                 <1>       ; 18/10/2015
16914 00004ABD 28C0                   <1>       sub   al, al
16915 00004ABF AA                     <1>       stosb
16916 00004AC0 FF05[70740000]         <1>       inc   dword [u.nread]
16917 00004AC6 EB29                   <1>       jmp   short sysexec_6
16918                                 <1> sysexec_5:
16919 00004AC8 AA                     <1>       stosb
16920 00004AC9 20C0                   <1>       and   al, al
16921 00004ACB 75E9                   <1>       jnz   short sysexec_4
16922 00004ACD B904000000             <1>       mov   ecx, 4
16923 00004AD2 390D[C4740000]         <1>       cmp   [ncount], ecx ; 4
16924 00004AD8 72AD                   <1>       jb    short sysexec_2
16925 00004ADA 8B35[C0740000]         <1>       mov   esi, [nbase]
16926 00004AE0 010D[C0740000]         <1>       add   [nbase], ecx ; 4
16927 00004AE6 66290D[C4740000]       <1>       sub   [ncount], cx
16928 00004AED 8B06                   <1>       mov   eax, [esi]
16929 00004AEF EBA6                   <1>       jmp   short sysexec_3
16930                                 <1> sysexec_6:
16931                                 <1>       ; 18/10/2015
16932                                 <1>       ; argument list transfer from user's core memory to
16933                                 <1>       ; kernel stack frame is OK here.
16934                                 <1>       ; [u.nread] = ; argument list length
16935                                 <1>       ;mov  [argv], esp ; start address of argument list
16936                                 <1>       ;
16937                                 <1>       ; 18/10/2015
16938                                 <1>       ; 24/07/2015
16939                                 <1>         ; 21/07/2015
16940                                 <1>       ; 02/07/2015
16941                                 <1>       ; 25/06/2015
16942                                 <1>       ; 24/06/2015
16943                                 <1>       ; 23/06/2015
16944                                 <1>       ;
16945 00004AF1 8B1D[A5740000]         <1>       mov   ebx, [u.ppgdir] ; parent's page directory
16946 00004AF7 21DB                   <1>       and   ebx, ebx  ; /etc/init ? (u.ppgdir = 0)
16947 00004AF9 740A                   <1>       jz    short sysexec_7
16948 00004AFB A1[A1740000]           <1>       mov   eax, [u.pgdir] ; physical address of page directory
16949 00004B00 E829E6FFFF             <1>       call  deallocate_page_dir
16950                                 <1> sysexec_7:
16951 00004B05 E859E5FFFF             <1>       call  make_page_dir
16952                                 <1>       ;jc   short sysexec_14
16953 00004B0A 0F82C0EDFFFF           <1>       jc    panic  ; allocation error
16954                                 <1>                    ; after a deallocation would be nonsence !?
16955                                 <1>       ; 24/07/2015
16956                                 <1>       ; map kernel pages (1st 4MB) to PDE 0
16957                                 <1>       ;     of the user's page directory
16958                                 <1>       ;     (It is needed for interrupts!)
16959                                 <1>       ; 18/10/2015
16960 00004B10 8B15[68700000]         <1>       mov   edx, [k_page_dir] ; Kernel's page directory
16961 00004B16 8B02                   <1>       mov   eax, [edx] ; physical address of
16962                                 <1>                        ; kernel's first page table (1st 4 MB)
16963                                 <1>                        ; (PDE 0 of kernel's page directory)
16964 00004B18 8B15[A1740000]         <1>       mov   edx, [u.pgdir]
16965 00004B1E 8902                   <1>       mov   [edx], eax ; PDE 0 (1st 4MB)
16966                                 <1>       ;
16967                                 <1>       ; 20/07/2015
```

```
16968 00004B20 BB00004000          <1>       mov    ebx, CORE ; start address = 0 (virtual) + CORE
16969                              <1>       ; 18/10/2015
16970 00004B25 BE[B8740000]        <1>       mov    esi, pcore ; physical start address
16971                              <1> sysexec_8:
16972 00004B2A B907000000          <1>       mov    ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
16973 00004B2F E84DE5FFFF          <1>       call   make_page_table
16974 00004B34 0F8296EDFFFF        <1>       jc     panic
16975                              <1>       ;mov   ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
16976 00004B3A E850E5FFFF          <1>       call   make_page ; make new page, clear and set the pte
16977 00004B3F 0F828BEDFFFF        <1>       jc     panic
16978                              <1>       ;
16979 00004B45 8906                <1>       mov    [esi], eax ; 24/06/2015
16980                              <1>       ; ebx = virtual address (24/07/2015)
16981 00004B47 E866EAFFFF          <1>       call   add_to_swap_queue
16982                              <1>       ; 18/10/2015
16983 00004B4C 81FE[BC740000]      <1>       cmp    esi, ecore ; user's stack (last) page ?
16984 00004B52 740C                <1>       je     short sysexec_9 ; yes
16985 00004B54 BE[BC740000]        <1>       mov    esi, ecore  ; physical address of the last page
16986                              <1>       ; 20/07/2015
16987 00004B59 BB00F0FFFF          <1>       mov    ebx, (ECORE - PAGE_SIZE) + CORE
16988                              <1>       ; ebx = virtual end address + segment base address - 4K
16989 00004B5E EBCA                <1>         jmp     short sysexec_8
16990                              <1>
16991                              <1> sysexec_9:
16992                              <1>       ; 18/10/2015
16993                              <1>       ; 26/08/2015
16994                              <1>       ; 25/06/2015
16995                              <1>       ; move arguments from kernel stack to [ecore]
16996                              <1>       ; (argument list/line will be copied from kernel stack
16997                              <1>       ; frame to the last (stack) page of user's core memory)
16998                              <1>       ; 18/10/2015
16999 00004B60 8B3D[BC740000]      <1>       mov    edi, [ecore]
17000 00004B66 81C700100000        <1>       add    edi, PAGE_SIZE
17001 00004B6C 0FB705[C6740000]    <1>       movzx  eax, word [argc]
17002 00004B73 09C0                <1>       or     eax, eax
17003 00004B75 7509                <1>       jnz    short sysexec_10
17004 00004B77 89FB                <1>       mov    ebx, edi
17005 00004B79 83EB04              <1>       sub    ebx, 4
17006 00004B7C 8903                <1>       mov    [ebx], eax ; 0
17007 00004B7E EB40                <1>       jmp    short sysexec_13
17008                              <1> sysexec_10:
17009 00004B80 8B0D[70740000]      <1>       mov    ecx, [u.nread]
17010                              <1>       ;mov   esi, [argv}
17011 00004B86 89E6                <1>       mov    esi, esp ; start address of argument list
17012 00004B88 29CF                <1>       sub    edi, ecx ; page end address - argument list length
17013 00004B8A 89C2                <1>       mov    edx, eax
17014 00004B8C FEC2                <1>       inc    dl ; argument count + 1 for argc value
17015 00004B8E C0E202              <1>       shl    dl, 2  ; 4 * (argument count + 1)
17016 00004B91 89FB                <1>       mov    ebx, edi
17017 00004B93 80E3FC              <1>       and    bl, 0FCh ; 32 bit (dword) alignment
17018 00004B96 29D3                <1>       sub    ebx, edx
17019 00004B98 89FA                <1>       mov    edx, edi
17020 00004B9A F3A4                <1>       rep    movsb
17021 00004B9C 89D6                <1>       mov    esi, edx
17022 00004B9E 89DF                <1>       mov    edi, ebx
17023 00004BA0 BA00F0BFFF          <1>       mov    edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
17024 00004BA5 2B15[BC740000]      <1>       sub    edx, [ecore] ; difference (virtual - physical)
17025 00004BAB AB                  <1>       stosd  ; eax = argument count
17026                              <1> sysexec_11:
17027 00004BAC 89F0                <1>       mov    eax, esi
17028 00004BAE 01D0                <1>       add    eax, edx
17029 00004BB0 AB                  <1>       stosd  ; eax = virtual address
17030 00004BB1 FE0D[C6740000]      <1>       dec    byte [argc]
17031 00004BB7 7407                <1>       jz     short sysexec_13
17032                              <1> sysexec_12:
17033 00004BB9 AC                  <1>       lodsb
17034 00004BBA 20C0                <1>       and    al, al
17035 00004BBC 75FB                <1>       jnz    short sysexec_12
17036 00004BBE EBEC                <1>       jmp    short sysexec_11
17037                              <1>       ;
17038                              <1>       ; 1:
17039                              <1>             ; mov (sp)+,r5 / r5 now contains address of list of
17040                              <1>                      ; / pointers to arguments to be passed
17041                              <1>             ; mov $1,u.quit / u.quit determines handling of quits;
17042                              <1>                       ; / u.quit = 1 take quit
17043                              <1>             ; mov $1,u.intr / u.intr determines handling of
17044                              <1>                      ; / interrupts; u.intr = 1 take interrupt
17045                              <1>             ; mov $rtssym,30 / emt trap vector set to take
17046                              <1>                       ; / system routine
17047                              <1>             ; mov $fpsym,*10 / reserved instruction trap vector
17048                              <1>                       ; / set to take system routine
17049                              <1>             ; mov $sstack,sp / stack space used during swapping
17050                              <1>             ; mov r5,-(sp) / save arguments pointer on stack
17051                              <1>             ; mov $ecore,r5 / r5 has end of core
17052                              <1>             ; mov $core,r4 / r4 has start of users core
17053                              <1>             ; mov r4,u.base / u.base has start of users core
17054                              <1>             ; mov (sp),r2 / move arguments list pointer into r2
17055                              <1>       ; 1:
17056                              <1>             ; tst (r2)+ / argument char = "nul"
17057                              <1>             ; bne 1b
17058                              <1>             ; tst -(r2) / decrement r2 by 2; r2 has addr of
17059                              <1>                    ; / end of argument pointer list
17060                              <1>       ; 1:
17061                              <1>           ; / move arguments to bottom of users core
17062                              <1>             ; mov -(r2),r3 / (r3) last non zero argument ptr
17063                              <1>             ; cmp r2,(sp) / is r2 = beginning of argument
17064                              <1>                      ; / ptr list
17065                              <1>             ; blo 1f / branch to 1f when all arguments
17066                              <1>                   ; / are moved
17067                              <1>             ; mov -(r2),r3 / (r3) last non zero argument ptr
17068                              <1>       ; 2:
17069                              <1>             ; tstb (r3)+
17070                              <1>             ; bne 2b / scan argument for \0 (nul)
17071                              <1>
17072                              <1>       ; 2:
```

```
17073                                  <1>             ; movb -(r3),-(r5) / move argument char
17074                                  <1>                               ; / by char starting at "ecore"
17075                                  <1>             ; cmp r3,(r2) / moved all characters in
17076                                  <1>                       ; / this argument
17077                                  <1>             ; bhi 2b / branch 2b if not
17078                                  <1>             ; mov r5,(r4)+ / move r5 into top of users core;
17079                                  <1>                         ; / r5 has pointer to nth arg
17080                                  <1>             ; br 1b / string
17081                                  <1>     ; 1:
17082                                  <1>             ; clrb -(r5)
17083                                  <1>             ; bic $1,r5 / make r5 even, r5 points to
17084                                  <1>                  ; / last word of argument strings
17085                                  <1>             ; mov $core,r2
17086                                  <1>
17087                                  <1>     ; 1: / move argument pointers into core following
17088                                  <1>            ; / argument strings
17089                                  <1>             ; cmp r2,r4
17090                                  <1>             ; bhis 1f / branch to 1f when all pointers
17091                                  <1>                  ; / are moved
17092                                  <1>             ; mov (r2)+,-(r5)
17093                                  <1>             ; br 1b
17094                                  <1>     ; 1:
17095                                  <1>             ; sub $core,r4 / gives number of arguments *2
17096                                  <1>             ; asr r4 / divide r4 by 2 to calculate
17097                                  <1>                   ; / the number of args stored
17098                                  <1>             ; mov r4,-(r5) / save number of arguments ahead
17099                                  <1>                         ; / of the argument pointers
17100                                  <1> sysexec_13:
17101                                  <1>     ; 19/10/2015
17102                                  <1>     ; 18/10/2015
17103                                  <1>     ; 29/07/2015
17104                                  <1>     ; 25/07/2015
17105                                  <1>     ; 24/07/2015
17106                                  <1>     ; 20/07/2015
17107                                  <1>     ; 25/06/2015
17108                                  <1>     ; 24/06/2015
17109                                  <1>     ; 23/06/2015
17110                                  <1>     ;
17111                                  <1>     ; moving arguments to [ecore] is OK here..
17112                                  <1>     ; 18/10/2015
17113 00004BC0 89EC                    <1>     mov   esp, ebp ; (**) restore kernel stack pointer
17114                                  <1>     ; ebx = beginning addres of argument list pointers
17115                                  <1>     ;     in user's stack
17116                                  <1>     ; 19/10/2015
17117 00004BC2 2B1D[BC740000]          <1>     sub   ebx, [ecore]
17118 00004BC8 81C300F0BFFF            <1>     add     ebx, (ECORE - PAGE_SIZE)
17119                                  <1>                ; end of core - 4096 (last page)
17120                                  <1>                ; (virtual address)
17121 00004BCE 891D[C8740000]          <1>     mov   [argv], ebx
17122 00004BD4 891D[74740000]          <1>     mov   [u.break], ebx ; available user memory
17123                                  <1>     ;
17124 00004BDA 29C0                    <1>     sub   eax, eax
17125 00004BDC C705[6C740000]2000-     <1>     mov   dword [u.count], 32 ; Executable file header size
17126 00004BE4 0000                    <1>
17127                                  <1>           ; mov $14,u.count
17128 00004BE6 C705[58740000]-         <1>     mov   dword [u.fofp], u.off
17129 00004BEC [64740000]              <1>
17130                                  <1>           ; mov $u.off,u.fofp
17131 00004BF0 A3[64740000]            <1>     mov   [u.off], eax ; 0
17132                                  <1>           ; clr u.off / set offset in file to be read to zero
17133                                  <1>     ; 25/07/2015
17134 00004BF5 A3[68740000]            <1>     mov   [u.base], eax ; 0, start of user's core (virtual)
17135                                  <1>     ; 25/06/2015
17136 00004BFA 66A1[2A740000]          <1>     mov   ax, [ii]
17137                                  <1>     ; AX = i-number of the executable file
17138 00004C00 E8C10C0000              <1>     call  readi
17139                                  <1>           ; jsr r0,readi / read in first six words of
17140                                  <1>                 ; / user's file, starting at $core
17141                                  <1>           ; mov sp,r5 / put users stack address in r5
17142                                  <1>           ; sub $core+40.,r5 / subtract $core +40,
17143                                  <1>                       ; / from r5 (leaves number of words
17144                                  <1>                       ; / less 26 available for
17145                                  <1>                       ; / program in user core
17146                                  <1>           ; mov r5,u.count /
17147                                  <1>     ; 25/06/2015
17148 00004C05 8B0D[74740000]          <1>     mov   ecx, [u.break] ; top of user's stack (physical addr.)
17149 00004C0B 890D[6C740000]          <1>     mov   [u.count], ecx ; save for overrun check
17150                                  <1>     ;
17151 00004C11 8B0D[70740000]          <1>     mov   ecx, [u.nread]
17152 00004C17 890D[74740000]          <1>     mov   [u.break], ecx ; virtual address (offset from start)
17153 00004C1D 80F920                  <1>     cmp   cl, 32
17154 00004C20 7540                    <1>       jne     short sysexec_15
17155                                  <1>     ;:
17156                                  <1>     ; 25/06/2015
17157                                  <1>     ; Retro UNIX 386 v1 (32 bit) executable file header format
17158                                  <1>     ; 18/10/2015
17159 00004C22 8B35[B8740000]          <1>     mov   esi, [pcore] ; start address of user's core memory
17160                                  <1>                        ; (phys. start addr. of the exec. file)
17161 00004C28 AD                      <1>     lodsd
17162 00004C29 663DEB1E                <1>     cmp   ax, 1EEBh ; EBH, 1Eh -> jump to +32
17163 00004C2D 7533                    <1>     jne   short sysexec_15
17164                                  <1>           ; cmp core,$405 / br .+14 is first instruction
17165                                  <1>                          ; / if file is standard a.out format
17166                                  <1>           ; bne 1f / branch, if not standard format
17167 00004C2F AD                      <1>     lodsd
17168 00004C30 89C1                    <1>     mov   ecx, eax ; text (code) section size
17169 00004C32 AD                      <1>     lodsd
17170 00004C33 01C1                    <1>     add   ecx, eax ; + data section size (initialized data)
17171                                  <1>           ; mov core+2,r5 / put 2nd word of users program in r5;
17172                                  <1>                            ; / number of bytes in program text
17173                                  <1>           ; sub $14,r5 / subtract 12
17174 00004C35 89CB                    <1>     mov   ebx, ecx
17175                                  <1>     ;
17176                                  <1>     ; 25/06/2015
17177                                  <1>     ; NOTE: These are for next versions of Retro UNIX 386
```

```
17178                                  <1>       ;     and SINGLIX operating systems (as code template).
17179                                  <1>       ;     Current Retro UNIX 386 v1 files can be max. 64KB
17180                                  <1>       ;     due to RUFS (floppy disk file system) restriction...
17181                                  <1>       ;     Overrun is not possible for current version.
17182                                  <1>       ;
17183 00004C37 AD                      <1>       lodsd
17184 00004C38 01C3                    <1>       add   ebx, eax ; + bss section size (for overrun checking)
17185 00004C3A 3B1D[6C740000]          <1>       cmp   ebx, [u.count]
17186 00004C40 7711                    <1>       ja    short sysexec_14  ; program overruns stack !
17187                                  <1>       ;
17188                                  <1>       ; 24/07/2015
17189                                  <1>       ; add bss section size to [u.break]
17190 00004C42 0105[74740000]          <1>       add   [u.break], eax
17191                                  <1>       ;
17192 00004C48 83E920                  <1>       sub   ecx, 32  ; header size (already loaded)
17193                                  <1>       ;cmp  ecx, [u.count]
17194                                  <1>       ;jnb  short sysexec_16
17195                                  <1>             ; cmp r5,u.count /
17196                                  <1>             ; bgt 1f / branch if r5 greater than u.count
17197 00004C4B 890D[6C740000]          <1>       mov   [u.count], ecx ; required read count
17198                                  <1>             ; mov r5,u.count
17199                                  <1>       ;
17200 00004C51 EB2A                    <1>       jmp   short sysexec_16
17201                                  <1>       ;
17202                                  <1> sysexec_14:
17203                                  <1>       ; 23/06/2015
17204                                  <1>       ; insufficient (out of) memory
17205 00004C53 C705[9D740000]0100-     <1>       mov   dword [u.error], ERR_MINOR_IM ; 1
17206 00004C5B 0000                    <1>
17207 00004C5D E9D8F3FFFF              <1>       jmp   error
17208                                  <1>       ;
17209                                  <1> sysexec_15:
17210                                  <1>       ; 25/06/2015
17211 00004C62 0FB715[1A710000]        <1>         movzx   edx, word [i.size] ; file size
17212 00004C69 29CA                    <1>       sub   edx, ecx ; file size - loaded bytes
17213 00004C6B 7627                    <1>       jna   short sysexec_17 ; no need to next read
17214 00004C6D 01D1                    <1>       add   ecx, edx ; [i.size]
17215 00004C6F 3B0D[6C740000]          <1>       cmp   ecx, [u.count] ; overrun check (!)
17216 00004C75 77DC                    <1>       ja    short sysexec_14
17217 00004C77 8915[6C740000]          <1>       mov   [u.count], edx
17218                                  <1> sysexec_16:
17219 00004C7D 66A1[2A740000]          <1>       mov   ax, [ii] ; i-number
17220 00004C83 E83E0C0000              <1>       call  readi
17221                                  <1>             ; add core+10,u.nread / add size of user data area
17222                                  <1>                                  ; / to u.nread
17223                                  <1>             ; br 2f
17224                                  <1>       ; 1:
17225                                  <1>             ; jsr r0,readi / read in rest of file
17226                                  <1>       ; 2:
17227 00004C88 8B0D[70740000]          <1>       mov   ecx, [u.nread]
17228 00004C8E 010D[74740000]          <1>       add   [u.break], ecx
17229                                  <1>             ; mov u.nread,u.break / set users program break to end of
17230                                  <1>                                  ; / user code
17231                                  <1>             ; add $core+14,u.break / plus data area
17232                                  <1> sysexec_17: ; 20/07/2015
17233                                  <1>       ;mov  ax, [ii] ;rgc i-number
17234 00004C94 E8F9120000              <1>       call  iclose
17235                                  <1>             ; jsr r0,iclose / does nothing
17236 00004C99 31C0                    <1>         xor     eax, eax
17237 00004C9B FEC0                    <1>       inc   al
17238 00004C9D 66A3[8C740000]          <1>       mov   [u.intr], ax ; 1 (interrupt/time-out is enabled)
17239 00004CA3 66A3[8E740000]          <1>       mov   [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
17240                                  <1>       ; 02/07/2015
17241 00004CA9 833D[A5740000]00        <1>         cmp  dword [u.ppgdir], 0  ; is the caller sys_init (kernel) ?
17242 00004CB0 770C                    <1>       ja    short sysexec_18 ; no, the caller is user process
17243                                  <1>       ; If the caller is kernel (sys_init), 'sysexec' will come here
17244 00004CB2 8B15[68700000]          <1>       mov   edx, [k_page_dir] ; kernel's page directory
17245 00004CB8 8915[A5740000]          <1>       mov   [u.ppgdir], edx ; next time 'sysexec' must not come here
17246                                  <1> sysexec_18:
17247                                  <1>       ; 18/10/2015
17248                                  <1>       ; 05/08/2015
17249                                  <1>       ; 29/07/2015
17250 00004CBE 8B2D[C8740000]          <1>       mov   ebp, [argv] ; user's stack pointer must point to argument
17251                                  <1>                         ; list pointers (argument count)
17252 00004CC4 FA                      <1>       cli
17253 00004CC5 8B25[04700000]          <1>         mov     esp, [tss.esp0]  ; ring 0 (kernel) stack pointer
17254                                  <1>       ;mov         esp, [u.sp] ; Restore Kernel stack
17255                                  <1>                         ; for this process
17256                                  <1>       ;add  esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
17257                                  <1>       ;xor  eax, eax ; 0
17258 00004CCB FEC8                    <1>       dec   al ; eax = 0
17259 00004CCD 66BA2300                <1>       mov   dx, UDATA
17260 00004CD1 6652                    <1>       push  dx  ; user's stack segment
17261 00004CD3 55                      <1>       push  ebp ; user's stack pointer
17262                                  <1>                 ; (points to number of arguments)
17263 00004CD4 FB                      <1>       sti
17264 00004CD5 9C                      <1>       pushfd ; EFLAGS
17265                                  <1>              ; Set IF for enabling interrupts in user mode
17266                                  <1>       ;or   dword [esp], 200h
17267                                  <1>       ;
17268                                  <1>       ;mov  bx, UCODE
17269                                  <1>       ;push bx ; user's code segment
17270 00004CD6 6A1B                    <1>       push  UCODE
17271                                  <1>       ;push 0
17272 00004CD8 50                      <1>       push  eax ; EIP (=0) - start address -
17273                                  <1>              ; clr -(r5) / popped into ps when rti in
17274                                  <1>                       ; / sysrele is executed
17275                                  <1>              ; mov $core,-(r5) / popped into pc when rti
17276                                  <1>                                ; / in sysrele is executed
17277                                  <1>              ;mov r5,0f / load second copyz argument
17278                                  <1>              ;tst -(r5) / decrement r5
17279 00004CD9 8925[40740000]          <1>       mov   [u.sp], esp ; 29/07/2015
17280                                  <1>       ; 05/08/2015
17281                                  <1>       ; Remedy of a General Protection Fault during 'iretd' is here !
17282                                  <1>       ; ('push dx' would cause to general protection fault,
```

```
17283                                  <1>       ; after 'pop ds' etc.)
17284                                  <1>       ;
17285                                  <1>       ;; push dx ; ds (UDATA)
17286                                  <1>       ;; push dx ; es (UDATA)
17287                                  <1>       ;; push dx ; fs (UDATA)
17288                                  <1>       ;; push dx ; gs (UDATA)
17289                                  <1>       ;
17290                                  <1>       ; This is a trick to prevent general protection fault
17291                                  <1>       ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):
17292 00004CDF 8EC2                   <1>       mov    es, dx ; UDATA
17293 00004CE1 06                     <1>       push   es ; ds (UDATA)
17294 00004CE2 06                     <1>       push   es ; es (UDATA)
17295 00004CE3 06                     <1>       push   es ; fs (UDATA)
17296 00004CE4 06                     <1>       push   es ; gs (UDATA)
17297 00004CE5 66BA1000               <1>       mov    dx, KDATA
17298 00004CE9 8EC2                   <1>       mov    es, dx
17299                                  <1>       ;
17300                                  <1>       ;; pushad simulation
17301 00004CEB 89E5                   <1>       mov    ebp, esp ; esp before pushad
17302 00004CED 50                     <1>       push   eax ; eax (0)
17303 00004CEE 50                     <1>       push   eax ; ecx (0)
17304 00004CEF 50                     <1>       push   eax ; edx (0)
17305 00004CF0 50                     <1>       push   eax ; ebx (0)
17306 00004CF1 55                     <1>       push   ebp ; esp before pushad
17307 00004CF2 50                     <1>       push   eax ; ebp (0)
17308 00004CF3 50                     <1>       push   eax ; esi (0)
17309 00004CF4 50                     <1>       push   eax ; edi (0)
17310                                  <1>       ;
17311 00004CF5 A3[48740000]           <1>       mov    [u.r0], eax ; eax = 0
17312 00004CFA 8925[44740000]         <1>       mov    [u.usp], esp
17313                                  <1>              ; mov r5,u.r0 /
17314                                  <1>              ; sub $16.,r5 / skip 8 words
17315                                  <1>              ; mov r5,u.sp / assign user stack pointer value,
17316                                  <1>              ;             / effectively zeroes all regs
17317                                  <1>                      ; / when sysrele is executed
17318                                  <1>              ; jsr r0,copyz; core; 0:0 / zero user's core
17319                                  <1>              ; clr u.break
17320                                  <1>              ; mov r5,sp / point sp to user's stack
17321                                  <1>       ;
17322 00004D00 E958F3FFFF             <1>       jmp    sysret0
17323                                  <1>       ;jmp   sysret
17324                                  <1>              ; br sysret3 / return to core image at $core
17325                                  <1>
17326                                  <1> get_argp:
17327                                  <1>       ; 18/10/2015 (nbase, ncount)
17328                                  <1>       ; 21/07/2015
17329                                  <1>       ; 24/06/2015 (Retro UNIX 386 v1)
17330                                  <1>       ; Get (virtual) address of argument from user's core memory
17331                                  <1>       ;
17332                                  <1>       ; INPUT:
17333                                  <1>       ;     esi = virtual address of argument pointer
17334                                  <1>       ; OUTPUT:
17335                                  <1>       ;     eax = virtual address of argument
17336                                  <1>       ;
17337                                  <1>       ; Modified registers: EAX, EBX, ECX, EDX, ESI
17338                                  <1>       ;
17339 00004D05 833D[A5740000]00       <1>       cmp     dword [u.ppgdir], 0 ; /etc/init ?
17340                                  <1>                                   ; (the caller is kernel)
17341 00004D0C 7667                   <1>         jna     short get_argpk
17342                                  <1>       ;
17343 00004D0E 89F3                   <1>       mov    ebx, esi
17344 00004D10 E873E9FFFF             <1>       call   get_physical_addr ; get physical address
17345 00004D15 0F8289000000           <1>         jc      get_argp_err
17346 00004D1B A3[C0740000]           <1>       mov    [nbase], eax ; physical address
17347 00004D20 66890D[C4740000]       <1>       mov    [ncount], cx ; remain byte count in page (1-4096)
17348 00004D27 B804000000             <1>       mov    eax, 4 ; 21/07/2015
17349 00004D2C 6639C1                 <1>       cmp    cx, ax ; 4
17350 00004D2F 735D                   <1>       jnb    short get_argp2
17351 00004D31 89F3                   <1>       mov    ebx, esi
17352 00004D33 01CB                   <1>       add    ebx, ecx
17353 00004D35 E84EE9FFFF             <1>       call   get_physical_addr ; get physical address
17354 00004D3A 7268                   <1>       jc     short get_argp_err
17355                                  <1>       ;push  esi
17356 00004D3C 89C6                   <1>       mov    esi, eax
17357 00004D3E 66870D[C4740000]       <1>       xchg   cx, [ncount]
17358 00004D45 8735[C0740000]         <1>       xchg   esi, [nbase]
17359 00004D4B B504                   <1>       mov    ch, 4
17360 00004D4D 28CD                   <1>       sub    ch, cl
17361                                  <1> get_argp0:
17362 00004D4F AC                     <1>       lodsb
17363 00004D50 6650                   <1>       push   ax
17364 00004D52 FEC9                   <1>       dec    cl
17365 00004D54 75F9                   <1>         jnz     short get_argp0
17366 00004D56 8B35[C0740000]         <1>       mov    esi, [nbase]
17367                                  <1>       ; 21/07/2015
17368 00004D5C 0FB6C5                 <1>       movzx  eax, ch
17369 00004D5F 0105[C0740000]         <1>       add    [nbase], eax
17370 00004D65 662905[C4740000]       <1>       sub    [ncount], ax
17371                                  <1> get_argp1:
17372 00004D6C AC                     <1>       lodsb
17373 00004D6D FECD                   <1>       dec    ch
17374 00004D6F 743D                   <1>         jz      short get_argp3
17375 00004D71 6650                   <1>         push ax
17376 00004D73 EBF7                   <1>       jmp     short get_argp1
17377                                  <1> get_argpk:
17378                                  <1>       ; Argument is in kernel's memory space
17379 00004D75 66C705[C4740000]00-    <1>       mov    word [ncount], PAGE_SIZE ; 4096
17380 00004D7D 10                     <1>
17381 00004D7E 8935[C0740000]         <1>       mov    [nbase], esi
17382 00004D84 8305[C0740000]04       <1>       add    dword [nbase], 4
17383 00004D8B 8B06                   <1>       mov    eax, [esi] ; virtual addr. = physcal addr.
17384 00004D8D C3                     <1>       retn
17385                                  <1> get_argp2:
17386                                  <1>       ; 21/07/2015
17387                                  <1>       ;mov   eax, 4
```

```
17388 00004D8E 8B15[C0740000]      <1>       mov    edx, [nbase] ; 18/10/2015
17389 00004D94 0105[C0740000]      <1>       add    [nbase], eax
17390 00004D9A 662905[C4740000]    <1>       sub    [ncount], ax
17391                              <1>       ;
17392 00004DA1 8B02                <1>       mov    eax, [edx]
17393 00004DA3 C3                  <1>       retn
17394                              <1> get_argp_err:
17395 00004DA4 A3[9D740000]        <1>       mov    [u.error], eax
17396 00004DA9 E98CF2FFFF          <1>       jmp    error
17397                              <1> get_argp3:
17398 00004DAE B103                <1>       mov    cl, 3
17399                              <1> get_argp4:
17400 00004DB0 C1E008              <1>       shl    eax, 8
17401 00004DB3 665A                <1>       pop    dx
17402 00004DB5 88D0                <1>       mov    al, dl
17403 00004DB7 E2F7                <1>         loop    get_argp4
17404                              <1>       ;pop   esi
17405 00004DB9 C3                  <1>       retn
17406                              <1>
17407                              <1> sysfstat:
17408                              <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
17409                              <1>       ; 19/06/2013 (Retro UNIX 8086 v1)
17410                              <1>       ;
17411                              <1>       ; 'sysfstat' is identical to 'sysstat' except that it operates
17412                              <1>       ; on open files instead of files given by name. It puts the
17413                              <1>       ; buffer address on the stack, gets the i-number and
17414                              <1>       ; checks to see if the file is open for reading or writing.
17415                              <1>       ; If the file is open for writing (i-number is negative)
17416                              <1>       ; the i-number is set positive and a branch into 'sysstat'
17417                              <1>       ; is made.
17418                              <1>       ;
17419                              <1>       ; Calling sequence:
17420                              <1>       ;     sysfstat; buf
17421                              <1>       ; Arguments:
17422                              <1>       ;     buf - buffer address
17423                              <1>       ;
17424                              <1>       ; Inputs: *u.r0 - file descriptor
17425                              <1>       ; Outputs: buffer is loaded with file information
17426                              <1>       ; ..........................................................
17427                              <1>       ;
17428                              <1>       ; Retro UNIX 8086 v1 modification:
17429                              <1>       ;       'sysfstat' system call has two arguments; so,
17430                              <1>       ;     * 1st argument, file descriptor is in BX register
17431                              <1>       ;     * 2nd argument, buf is pointed to by CX register
17432                              <1>
17433                              <1>       ; / set status of open file
17434                              <1>             ; jsr r0,arg; u.off / put buffer address in u.off
17435 00004DBA 51                  <1>       push   ecx
17436                              <1>             ; mov u.off,-(sp) / put buffer address on the stack
17437                              <1>             ; mov *u.r0,r1 / put file descriptor in r1
17438                              <1>             ; jsr r0,getf / get the files i-number
17439                              <1>       ; BX = file descriptor (file number)
17440 00004DBB E8FF000000          <1>       call   getf1
17441 00004DC0 6621C0              <1>       and    ax, ax ; i-number of the file
17442                              <1>             ; tst  r1 / is it 0?
17443                              <1>       ;jz    error
17444                              <1>             ; beq error3 / yes, error
17445 00004DC3 750F                <1>       jnz    short sysfstat1
17446 00004DC5 C705[9D740000]0A00- <1>       mov    dword [u.error], ERR_FILE_NOT_OPEN  ; 'file not open !'
17447 00004DCD 0000                <1>
17448 00004DCF E966F2FFFF          <1>       jmp    error
17449                              <1> sysfstat1:
17450 00004DD4 80FC80              <1>       cmp    ah, 80h
17451 00004DD7 7223                <1>         jb      short sysstat1
17452                              <1>             ; bgt 1f / if i-number is negative (open for writing)
17453 00004DD9 66F7D8              <1>       neg    ax
17454                              <1>             ; neg r1 / make it positive, then branch
17455 00004DDC EB1E                <1>       jmp    short sysstat1
17456                              <1>             ; br 1f / to 1f
17457                              <1> sysstat:
17458                              <1>       ; 18/10/2015
17459                              <1>       ; 07/10/2015
17460                              <1>       ; 02/09/2015
17461                              <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
17462                              <1>       ; 19/06/2013 (Retro UNIX 8086 v1)
17463                              <1>       ;
17464                              <1>       ; 'sysstat' gets the status of a file. Its arguments are the
17465                              <1>       ; name of the file and buffer address. The buffer is 34 bytes
17466                              <1>       ; long and information about the file placed in it.
17467                              <1>       ; sysstat calls 'namei' to get the i-number of the file.
17468                              <1>       ; Then 'iget' is called to get i-node in core. The buffer
17469                              <1>       ; is then loaded and the results are given in the UNIX
17470                              <1>       ; Programmers Manual sysstat (II).
17471                              <1>       ;
17472                              <1>       ; Calling sequence:
17473                              <1>       ;     sysstat; name; buf
17474                              <1>       ; Arguments:
17475                              <1>       ;     name - points to the name of the file
17476                              <1>       ;     buf - address of a 34 bytes buffer
17477                              <1>       ; Inputs: -
17478                              <1>       ; Outputs: buffer is loaded with file information
17479                              <1>       ; ..........................................................
17480                              <1>       ;
17481                              <1>       ; Retro UNIX 8086 v1 modification:
17482                              <1>       ;       'sysstat' system call has two arguments; so,
17483                              <1>       ;     Retro UNIX 8086 v1 argument transfer method 2 is used
17484                              <1>       ;     to get sysstat system call arguments from the user;
17485                              <1>       ;     * 1st argument, name is pointed to by BX register
17486                              <1>       ;     * 2nd argument, buf is pointed to by CX register
17487                              <1>       ;
17488                              <1>       ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
17489                              <1>       ;           arguments which were in these registers;
17490                              <1>       ;           but, it returns by putting the 1st argument
17491                              <1>       ;           in 'u.namep' and the 2nd argument
17492                              <1>       ;           on top of stack. (1st argument is offset of the
```

```
17493                                  <1>       ;           file/path name in the user's program segment.)
17494                                  <1>
17495                                  <1>       ; / ; name of file; buffer - get files status
17496                                  <1>             ; jsr r0,arg2 / get the 2 arguments
17497 00004DDE 891D[60740000]          <1>       mov   [u.namep], ebx
17498 00004DE4 51                      <1>       push  ecx
17499 00004DE5 E80C010000              <1>       call  namei
17500                                  <1>             ; jsr r0,namei / get the i-number for the file
17501                                  <1>       ;jc   error
17502                                  <1>             ; br error3 / no such file, error
17503 00004DEA 7310                    <1>       jnc   short sysstat1
17504                                  <1>       ; pop  ecx
17505                                  <1> sysstat_err0:
17506                                  <1>       ; 'file not found !' error
17507 00004DEC C705[9D740000]0C00-     <1>       mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
17508 00004DF4 0000                    <1>
17509 00004DF6 E93FF2FFFF              <1>       jmp   error
17510                                  <1>
17511 00004DFB 00                      <1> statx: db 0
17512                                  <1>
17513                                  <1> sysstat1: ; 1:
17514 00004DFC E8E5080000              <1>       call  iget
17515                                  <1>             ; jsr r0,iget / get the i-node into core
17516                                  <1>       ; 07/10/2015 (ax = [ii], inode number)
17517                                  <1>       ; 02/09/2015
17518 00004E01 8F05[68740000]          <1>       pop   dword [u.base]
17519                                  <1>             ; mov (sp)+,r3 / move u.off to r3 (points to buffer)
17520 00004E07 E858000000              <1>       call  sysstat_gpa ; get physical address
17521 00004E0C 730A                    <1>       jnc   short sysstat2
17522                                  <1> sysstat_err1:
17523 00004E0E A3[9D740000]            <1>       mov   dword [u.error], eax ; error code
17524 00004E13 E922F2FFFF              <1>       jmp   error
17525                                  <1> sysstat2:
17526 00004E18 A0[2A740000]            <1>       mov   al, [ii] ; 07/10/2015 (result of 'iget' call, above)
17527 00004E1D AA                      <1>       stosb
17528 00004E1E FF05[68740000]          <1>       inc   dword [u.base]
17529 00004E24 6649                    <1>       dec   cx
17530 00004E26 7505                    <1>       jnz   short sysstat3
17531 00004E28 E837000000              <1>       call  sysstat_gpa
17532                                  <1>       ;jc   short sysstat_err1
17533                                  <1> sysstat3:
17534 00004E2D A0[2B740000]            <1>       mov   al, [ii+1] ; 07/10/2015 (result of 'iget' call, above)
17535 00004E32 AA                      <1>       stosb
17536                                  <1>             ; mov r1,(r3)+ / put i-number in 1st word of buffer
17537 00004E33 FF05[68740000]          <1>       inc   dword [u.base]
17538                                  <1>       ;dec  word [u.pcount]
17539 00004E39 6649                    <1>       dec   cx
17540 00004E3B 7505                    <1>       jnz   short sysstat4
17541 00004E3D E822000000              <1>       call  sysstat_gpa
17542                                  <1>       ;jc   short sysstat_err1
17543                                  <1> sysstat4:
17544 00004E42 BE[16710000]            <1>       mov   esi, inode
17545                                  <1>             ; mov $inode,r2 / r2 points to i-node
17546                                  <1> sysstat5: ; 1:
17547 00004E47 A4                      <1>       movsb
17548                                  <1>             ; mov (r2)+,(r3)+ / move rest of i-node to buffer
17549 00004E48 FF05[68740000]          <1>       inc   dword [u.base]
17550                                  <1>       ;dec  word [u.pcount]
17551 00004E4E 6649                    <1>       dec   cx
17552 00004E50 7505                    <1>       jnz   short sysstat6
17553 00004E52 E80D000000              <1>       call  sysstat_gpa
17554                                  <1>       ;jc   short sysstat_err1
17555                                  <1> sysstat6:
17556 00004E57 81FE[36710000]          <1>       cmp   esi, inode + 32
17557                                  <1>             ; cmp r2,$inode+32 / done?
17558 00004E5D 75E8                    <1>       jne   short sysstat5
17559                                  <1>             ; bne 1b / no, go back
17560 00004E5F E9F6F1FFFF              <1>       jmp   sysret
17561                                  <1>             ; br sysret3 / return through sysret
17562                                  <1>       ;
17563                                  <1> sysstat_gpa: ; get physical address of file status buffer
17564                                  <1>       ; 02/09/2015
17565 00004E64 8B1D[68740000]          <1>       mov   ebx, [u.base]
17566                                  <1>       ; 07/10/2015
17567 00004E6A E819E8FFFF              <1>       call  get_physical_addr ; get physical address
17568                                  <1>       ;jc   short sysstat_gpa1
17569 00004E6F 729D                    <1>       jc    short sysstat_err1
17570                                  <1>       ; 18/10/2015
17571 00004E71 89C7                    <1>       mov   edi, eax ; physical address
17572                                  <1>       ;mov  [u.pcount], cx ; remain bytes in page
17573                                  <1> ;sysstat_gpa1:
17574 00004E73 C3                      <1>       retn
17575                                  <1>
17576                                  <1> fclose:
17577                                  <1>       ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
17578                                  <1>       ;            (32 bit offset pointer modification)
17579                                  <1>       ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
17580                                  <1>       ;
17581                                  <1>       ; Given the file descriptor (index to the u.fp list)
17582                                  <1>       ; 'fclose' first gets the i-number of the file via 'getf'.
17583                                  <1>       ; If i-node is active (i-number > 0) the entry in
17584                                  <1>       ; u.fp list is cleared. If all the processes that opened
17585                                  <1>       ; that file close it, then fsp etry is freed and the file
17586                                  <1>       ; is closed. If not a return is taken.
17587                                  <1>       ; If the file has been deleted while open, 'anyi' is called
17588                                  <1>       ; to see anyone else has it open, i.e., see if it is appears
17589                                  <1>       ; in another entry in the fsp table. Upon return from 'anyi'
17590                                  <1>       ; a check is made to see if the file is special.
17591                                  <1>       ;
17592                                  <1>       ; INPUTS ->
17593                                  <1>       ;    r1 - contains the file descriptor (value=0,1,2...)
17594                                  <1>       ;    u.fp - list of entries in the fsp table
17595                                  <1>       ;    fsp - table of entries (4 words/entry) of open files.
17596                                  <1>       ; OUTPUTS ->
17597                                  <1>       ;    r1 - contains the same file descriptor
```

```
17598                                  <1>       ;    r2 - contains i-number
17599                                  <1>       ;
17600                                  <1>       ; ((AX = R1))
17601                                  <1>       ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
17602                                  <1>       ;
17603                                  <1>       ; Retro UNIX 8086 v1 modification : CF = 1
17604                                  <1>       ;              if i-number of the file is 0. (error)
17605                                  <1>       ;
17606 00004E74 0FB7D0                  <1>       movzx  edx, ax ; **
17607 00004E77 6650                    <1>       push   ax ; ***
17608                                  <1>              ; mov r1,-(sp) / put r1 on the stack (it contains
17609                                  <1>                             ; / the index to u.fp list)
17610 00004E79 E83F000000              <1>       call   getf
17611                                  <1>              ; jsr r0,getf / r1 contains i-number,
17612                                  <1>                            ; / cdev has device =, u.fofp
17613                                  <1>                            ; / points to 3rd word of fsp entry
17614 00004E7E 6683F801                <1>       cmp    ax, 1 ; r1
17615                                  <1>              ; tst r1 / is i-number 0?
17616 00004E82 7236                    <1>       jb     short fclose_2
17617                                  <1>              ; beq 1f / yes, i-node not active so return
17618                                  <1>              ; tst (r0)+ / no, jump over error return
17619 00004E84 89D3                    <1>       mov    ebx, edx ; **
17620 00004E86 6689C2                  <1>       mov    dx, ax ; *
17621                                  <1>              ; mov r1,r2 / move i-number to r2 ;*
17622                                  <1>              ; mov (sp),r1 / restore value of r1 from the stack
17623                                  <1>                            ; / which is index to u.fp ; **
17624 00004E89 C683[4E740000]00        <1>       mov    byte [ebx+u.fp], 0
17625                                  <1>              ; clrb u.fp(r1) / clear that entry in the u.fp list
17626 00004E90 8B1D[58740000]          <1>       mov    ebx, [u.fofp]
17627                                  <1>              ; mov u.fofp,r1 / r1 points to 3rd word in fsp entry
17628                                  <1> fclose_0:
17629 00004E96 FE4B04                  <1>       dec    byte [ebx+4] ; 18/06/2015
17630                                  <1>              ; decb 2(r1) / decrement the number of processes
17631                                  <1>                           ; / that have opened the file
17632 00004E99 791F                    <1>       jns    short fclose_2 ; jump if not negative (jump if bit 7 is 0)
17633                                  <1>              ; bge 1f / if all processes haven't closed the file, return
17634                                  <1>       ;
17635 00004E9B 6652                    <1>       push   dx ;*
17636                                  <1>              ; mov r2,-(sp) / put r2 on the stack (i-number)
17637 00004E9D 6631C0                  <1>       xor    ax, ax ; 0
17638 00004EA0 668943FC                <1>       mov    [ebx-4], ax ; 0
17639                                  <1>              ; clr -4(r1) / clear 1st word of fsp entry
17640 00004EA4 8A4305                  <1>       mov    al, [ebx+5] ; 18/06/2015
17641                                  <1>              ; tstb 3(r1) / has this file been deleted
17642 00004EA7 20C0                    <1>       and    al, al
17643 00004EA9 7408                    <1>       jz     short fclose_1
17644                                  <1>              ; beq 2f / no, branch
17645 00004EAB 6689D0                  <1>       mov    ax, dx ; *
17646                                  <1>              ; mov r2,r1 / yes, put i-number back into r1
17647                                  <1>       ; AX = inode number
17648 00004EAE E868040000              <1>       call   anyi
17649                                  <1>              ; jsr r0,anyi / free all blocks related to i-number
17650                                  <1>                            ; / check if file appears in fsp again
17651                                  <1> fclose_1: ; 2:
17652 00004EB3 6658                    <1>       pop    ax ; *
17653                                  <1>              ; mov (sp)+,r1 / put i-number back into r1
17654 00004EB5 E8D8100000              <1>       call   iclose ; close if it is special file
17655                                  <1>              ; jsr r0,iclose / check to see if its a special file
17656                                  <1> fclose_2: ; 1:
17657 00004EBA 6658                    <1>       pop    ax ; ***
17658                                  <1>              ; mov (sp)+,r1 / put index to u.fp back into r1
17659 00004EBC C3                      <1>       retn
17660                                  <1>              ; rts r0
17661                                  <1>
17662                                  <1> getf: ; / get the device number and the i-number of an open file
17663                                  <1>       ; 13/05/2015
17664                                  <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
17665                                  <1>       ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
17666                                  <1>       ;
17667 00004EBD 89C3                    <1>       mov    ebx, eax
17668                                  <1> getf1: ;; Calling point from 'rw1' (23/05/2013)
17669 00004EBF 83FB0A                  <1>       cmp    ebx, 10
17670                                  <1>              ; cmp r1,$10. / user limited to 10 open files
17671 00004EC2 730A                    <1>         jnb  short getf2 ; 13/05/2015
17672                                  <1>       ;jnb     error
17673                                  <1>              ; bhis error3 / u.fp is table of users open files,
17674                                  <1>                            ; / index in fsp table
17675 00004EC4 8A9B[4E740000]          <1>       mov    bl, [ebx+u.fp]
17676                                  <1>              ; movb u.fp(r1),r1 / r1 contains number of entry
17677                                  <1>                                 ; / in fsp table
17678 00004ECA 08DB                    <1>       or     bl, bl
17679 00004ECC 7503                    <1>       jnz    short getf3
17680                                  <1>       ;jz    short getf4
17681                                  <1>              ; beq 1f / if its zero return
17682                                  <1> getf2:
17683                                  <1>       ; 'File not open !' error (ax=0)
17684 00004ECE 29C0                    <1>       sub    eax, eax
17685 00004ED0 C3                      <1>       retn
17686                                  <1> getf3:
17687                                  <1>       ; Retro UNIX 386 v1 modification ! (11/05/2015)
17688                                  <1>       ;
17689                                  <1>       ; 'fsp' table (10 bytes/entry)
17690                                  <1>       ; bit 15                             bit 0
17691                                  <1>       ; ---|-----------------------------------------
17692                                  <1>       ; r/w|       i-number of open file
17693                                  <1>       ; ---|-----------------------------------------
17694                                  <1>       ;               device number
17695                                  <1>       ; -----------------------------------------------
17696                                  <1>       ; offset pointer, r/w pointer to file (bit 0-15)
17697                                  <1>       ; -----------------------------------------------
17698                                  <1>       ; offset pointer, r/w pointer to file (bit 16-31)
17699                                  <1>       ; ----------------------|------------------------
17700                                  <1>       ;  flag that says file    | number of processes
17701                                  <1>       ;   has been deleted      | that have file open
17702                                  <1>       ; ----------------------|------------------------
```

```
17703                                  <1>       ;
17704 00004ED1 B80A000000               <1>       mov   eax, 10
17705 00004ED6 F6E3                     <1>       mul   bl
17706 00004ED8 BB[10720000]             <1>       mov   ebx, fsp - 6 ; the 3rd word in the fsp entry
17707 00004EDD 01C3                     <1>       add   ebx, eax
17708                                  <1>             ; asl r1
17709                                  <1>             ; asl r1 / multiply by 8 to get index into
17710                                  <1>                    ; / fsp table entry
17711                                  <1>             ; asl r1
17712                                  <1>             ; add $fsp-4,r1 / r1 is pointing at the 3rd word
17713                                  <1>                           ; / in the fsp entry
17714 00004EDF 891D[58740000]          <1>       mov   [u.fofp], ebx
17715                                  <1>             ; mov r1,u.fofp / save address of 3rd word
17716                                  <1>                           ; / in fsp entry in u.fofp
17717 00004EE5 4B                      <1>       dec   ebx
17718 00004EE6 4B                      <1>       dec   ebx
17719 00004EE7 668B03                  <1>       mov   ax, [ebx]
17720                                  <1>       ;mov  [cdev], al ; ;;Retro UNIX 8086 v1 !
17721 00004EEA 66A3[2E740000]          <1>       mov   [cdev], ax ; ;;in fact (!)
17722                                  <1>                          ;;dev number is in 1 byte
17723                                  <1>             ; mov -(r1),cdev / remove the device number  cdev
17724 00004EF0 4B                      <1>       dec   ebx
17725 00004EF1 4B                      <1>       dec   ebx
17726 00004EF2 668B03                  <1>       mov   ax, [ebx]
17727                                  <1>             ; mov -(r1),r1 / and the i-number  r1
17728                                  <1> getf4:      ; 1:
17729 00004EF5 C3                      <1>       retn
17730                                  <1>             ; rts r0
17731                                  <1>
17732                                  <1> namei:
17733                                  <1>       ; 04/12/2015 (14 byte file names)
17734                                  <1>       ; 18/10/2015 (nbase, ncount)
17735                                  <1>       ; 12/10/2015
17736                                  <1>       ; 21/08/2015
17737                                  <1>       ; 18/07/2015
17738                                  <1>       ; 02/07/2015
17739                                  <1>       ; 17/06/2015
17740                                  <1>       ; 16/06/2015 (Retro UNIX 386 v1 - Beginning)
17741                                  <1>       ; 24/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
17742                                  <1>       ;
17743                                  <1>       ; 'namei' takes a file path name and returns i-number of
17744                                  <1>       ; the file in the current directory or the root directory
17745                                  <1>       ; (if the first character of the pathname is '/').
17746                                  <1>       ;
17747                                  <1>       ; INPUTS ->
17748                                  <1>       ;    u.namep - points to a file path name
17749                                  <1>       ;    u.cdir - i-number of users directory
17750                                  <1>       ;    u.cdev - device number on which user directory resides
17751                                  <1>       ; OUTPUTS ->
17752                                  <1>       ;    r1 - i-number of file
17753                                  <1>       ;    cdev
17754                                  <1>       ;    u.dirbuf - points to directory entry where a match
17755                                  <1>       ;               occurs in the search for file path name.
17756                                  <1>       ;              If no match u.dirb points to the end of
17757                                  <1>       ;               the directory and r1 = i-number of the current
17758                                  <1>       ;              directory.
17759                                  <1>       ; ((AX = R1))
17760                                  <1>       ;
17761                                  <1>       ; (Retro UNIX Prototype : 07/10/2012 - 05/01/2013, UNIXCOPY.ASM)
17762                                  <1>         ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
17763                                  <1>       ;
17764                                  <1>
17765 00004EF6 66A1[4C740000]          <1>       mov   ax, [u.cdir]
17766                                  <1>             ; mov u.cdir,r1 / put the i-number of current directory
17767                                  <1>                           ; / in r1
17768 00004EFC 668B15[92740000]        <1>       mov   dx, [u.cdrv]
17769 00004F03 668915[2E740000]        <1>       mov   [cdev], dx       ; NOTE: Retro UNIX 8086 v1
17770                                  <1>                              ; device/drive number is in 1 byte,
17771                                  <1>                              ; not in 1 word!
17772                                  <1>             ; mov u.cdev,cdev / device number for users directory
17773                                  <1>                           ; / into cdev
17774                                  <1>       ; 12/10/2015
17775                                  <1>       ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
17776                                  <1>              ; convert virtual (pathname) addr to physical address
17777 00004F0A E82C010000              <1>       call    trans_addr_nmbp ; 12/10/2015
17778                                  <1>             ; esi = physical address of [u.namep]
17779                                  <1>             ; ecx = byte count in the page
17780 00004F0F 803E2F                  <1>       cmp   byte [esi], '/'
17781                                  <1>             ; cmpb *u.namep,$'/ / is first char in file name a /
17782 00004F12 751E                    <1>       jne   short namei_1
17783                                  <1>             ; bne 1f
17784 00004F14 FF05[60740000]          <1>       inc   dword [u.namep]
17785                                  <1>             ; inc u.namep / go to next char
17786 00004F1A 6649                    <1>       dec   cx ; remain byte count in the page
17787 00004F1C 7506                    <1>       jnz   short namei_0
17788                                  <1>       ; 12/10/2015
17789 00004F1E E818010000              <1>       call  trans_addr_nmbp ; convert virtual address to physical
17790                                  <1>             ; esi = physical address (page start + offset)
17791                                  <1>             ; ecx = byte count in the page
17792 00004F23 4E                      <1>       dec   esi
17793                                  <1> namei_0:
17794 00004F24 46                      <1>       inc   esi  ; go to next char
17795 00004F25 66A1[38740000]          <1>       mov   ax, [rootdir] ; 09/07/2013
17796                                  <1>             ; mov rootdir,r1 / put i-number of rootdirectory in r1
17797 00004F2B C605[2E740000]00        <1>       mov   byte [cdev], 0
17798                                  <1>             ; clr cdev / clear device number
17799                                  <1> namei_1: ; 1:
17800 00004F32 F606FF                  <1>       test  byte [esi], 0FFh
17801 00004F35 74BE                    <1>       jz    short getf4
17802                                  <1>       ;jz      nig
17803                                  <1>             ; tstb *u.namep / is the character in file name a nul
17804                                  <1>             ; beq nig / yes, end of file name reached;
17805                                  <1>                  ; / branch to "nig"
17806                                  <1> namei_2: ; 1:
17807                                  <1>       ; 18/10/2015
```

```
17808 00004F37 8935[C0740000]      <1>       mov   [nbase], esi
17809 00004F3D 66890D[C4740000]    <1>       mov   [ncount], cx
17810                              <1>       ;
17811                              <1>       ;mov  dx, 2
17812 00004F44 B202                <1>       mov   dl, 2 ; user flag (read, non-owner)
17813 00004F46 E876080000          <1>       call  access
17814                              <1>             ; jsr r0,access; 2 / get i-node with i-number r1
17815                              <1>       ; 'access' will not return here if user has not "r" permission !
17816 00004F4B 66F705[16710000]00- <1>       test  word [i.flgs], 4000h
17817 00004F53 40                  <1>
17818                              <1>             ; bit $40000,i.flgs / directory i-node?
17819 00004F54 746A                <1>         jz      short namei_err
17820                              <1>             ; beq error3 / no, got an error
17821                              <1>       ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
17822 00004F56 31C0                <1>       xor   eax, eax
17823 00004F58 A3[64740000]        <1>       mov   [u.off], eax ; 0
17824 00004F5D 66A1[1A710000]      <1>       mov   ax, [i.size]
17825 00004F63 A3[5C740000]        <1>       mov   [u.dirp], eax
17826                              <1>             ; mov i.size,u.dirp / put size of directory in u.dirp
17827                              <1>             ; clr u.off / u.off is file offset used by user
17828 00004F68 C705[58740000]-     <1>       mov   dword [u.fofp], u.off
17829 00004F6E [64740000]          <1>
17830                              <1>             ; mov $u.off,u.fofp / u.fofp is a pointer to
17831                              <1>                                 ; / the offset portion of fsp entry
17832                              <1> namei_3: ; 2:
17833 00004F72 C705[68740000]-     <1>       mov   dword [u.base], u.dirbuf
17834 00004F78 [7A740000]          <1>
17835                              <1>             ; mov $u.dirbuf,u.base / u.dirbuf holds a file name
17836                              <1>                                    ; / copied from a directory
17837 00004F7C C705[6C740000]1000- <1>       mov   dword [u.count], 16 ; 04/12/2015 (10 -> 16)
17838 00004F84 0000                <1>
17839                              <1>             ; mov $10.,u.count / u.count is byte count
17840                              <1>                                ; / for reads and writes
17841 00004F86 66A1[2A740000]      <1>       mov   ax, [ii]
17842                              <1>       ; 31/07/2013 ('namei_r') - 16/06/2015 ('u.kcall')
17843 00004F8C FE05[AF740000]      <1>       inc     byte [u.kcall] ; the caller is 'namei' sign
17844 00004F92 E82F090000          <1>       call  readi
17845                              <1>             ; jsr r0,readi / read 10. bytes of file
17846                              <1>                    ; with i-number (r1); i.e. read a directory entry
17847 00004F97 8B0D[70740000]      <1>       mov   ecx, [u.nread]
17848 00004F9D 09C9                <1>       or    ecx, ecx
17849                              <1>             ; tst u.nread
17850 00004F9F 741B                <1>       jz    short nib
17851                              <1>             ; ble nib / gives error return
17852                              <1>       ;
17853 00004FA1 668B1D[7A740000]    <1>       mov   bx, [u.dirbuf]
17854 00004FA8 6621DB              <1>       and   bx, bx
17855                              <1>             ; tst u.dirbuf /
17856 00004FAB 7522                <1>       jnz   short namei_4
17857                              <1>             ; bne 3f / branch when active directory entry
17858                              <1>                    ; / (i-node word in entry non zero)
17859 00004FAD A1[64740000]        <1>       mov   eax, [u.off]
17860 00004FB2 83E810              <1>       sub   eax, 16 ; 04/12/2015 (10 -> 16)
17861 00004FB5 A3[5C740000]        <1>       mov   [u.dirp], eax
17862                              <1>             ; mov u.off,u.dirp
17863                              <1>             ; sub $10.,u.dirp
17864 00004FBA EBB6                <1>       jmp   short namei_3
17865                              <1>             ; br 2b
17866                              <1>
17867                              <1>       ; 18/07/2013
17868                              <1> nib:
17869 00004FBC 31C0                <1>       xor   eax, eax  ; xor ax, ax ; ax = 0 -> file not found
17870 00004FBE F9                  <1>       stc
17871                              <1> nig:
17872 00004FBF C3                  <1>       retn
17873                              <1>
17874                              <1> namei_err:
17875                              <1>       ; 16/06/2015
17876 00004FC0 C705[9D740000]1300- <1>       mov   dword [u.error], ERR_NOT_DIR ; 'not a directory !' error
17877 00004FC8 0000                <1>
17878 00004FCA E96BF0FFFF          <1>       jmp   error
17879                              <1>
17880                              <1> namei_4: ; 3:
17881                              <1>       ; 18/10/2015
17882                              <1>       ; 12/10/2015
17883                              <1>       ; 21/08/2015
17884                              <1>       ; 18/07/2015
17885 00004FCF 8B2D[60740000]      <1>       mov   ebp, [u.namep]
17886                              <1>             ; mov u.namep,r2 / u.namep points into a file name string
17887 00004FD5 BF[7C740000]        <1>       mov   edi, u.dirbuf + 2
17888                              <1>             ; mov $u.dirbuf+2,r3 / points to file name of directory entry
17889                              <1>       ; 18/10/2015
17890 00004FDA 8B35[C0740000]      <1>       mov   esi, [nbase]
17891 00004FE0 668B0D[C4740000]    <1>       mov   cx, [ncount]
17892                              <1>       ;
17893 00004FE7 6621C9              <1>       and   cx, cx
17894 00004FEA 7505                <1>       jnz   short namei_5
17895                              <1>       ;
17896 00004FEC E850000000          <1>       call  trans_addr_nm ; convert virtual address to physical
17897                              <1>             ; esi = physical address (page start + offset)
17898                              <1>             ; ecx = byte count in the page
17899                              <1> namei_5: ; 3:
17900 00004FF1 45                  <1>       inc   ebp ; 18/07/2015
17901 00004FF2 AC                  <1>       lodsb   ; mov al, [esi] ; inc esi (al = r4)
17902                              <1>             ; movb (r2)+,r4 / move a character from u.namep string into r4
17903 00004FF3 08C0                <1>       or    al, al
17904 00004FF5 741D                <1>       jz    short namei_7
17905                              <1>             ; beq 3f / if char is nul, then the last char in string
17906                              <1>                    ; / has been moved
17907 00004FF7 3C2F                <1>       cmp   al, '/'
17908                              <1>             ; cmp r4,$'/ / is char a </>
17909 00004FF9 7419                <1>       je    short namei_7
17910                              <1>             ; beq 3f
17911                              <1>       ; 12/10/2015
17912 00004FFB 6649                <1>       dec   cx ; remain byte count in the page
```

```
17913 00004FFD 7505               <1>       jnz    short namei_6
17914 00004FFF E83D000000         <1>       call   trans_addr_nm ; convert virtual address to physical
17915                             <1>              ; esi = physical address (page start + offset)
17916                             <1>              ; ecx = byte count in the page
17917                             <1> namei_6:
17918 00005004 81FF[8A740000]     <1>         cmp     edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
17919                             <1>              ; cmp r3,$u.dirbuf+10. / have I checked
17920                             <1>                                   ; / all 8 bytes of file name
17921 0000500A 74E5               <1>       je     short namei_5
17922                             <1>              ; beq 3b
17923 0000500C AE                 <1>       scasb
17924                             <1>              ; cmpb (r3)+,r4 / compare char in u.namep string to file name
17925                             <1>                           ; / char read from directory
17926 0000500D 74E2               <1>       je     short namei_5
17927                             <1>              ; beq 3b / branch if chars match
17928                             <1>
17929 0000500F E95EFFFFFF         <1>         jmp    namei_3 ; 2b
17930                             <1>              ; br 2b / file names do not match go to next directory entry
17931                             <1> namei_7: ; 3:
17932 00005014 81FF[8A740000]     <1>       cmp    edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
17933                             <1>              ; cmp r3,$u.dirbuf+10. / if equal all 8 bytes were matched
17934 0000501A 740A               <1>       je     short namei_8
17935                             <1>              ; beq 3f
17936 0000501C 8A27               <1>       mov    ah, [edi]
17937                             <1>       ;inc   edi
17938 0000501E 20E4               <1>       and    ah, ah
17939                             <1>              ; tstb (r3)+ /
17940 00005020 0F854CFFFFFF       <1>         jnz     namei_3
17941                             <1>              ; bne 2b
17942                             <1> namei_8: ; 3
17943 00005026 892D[60740000]     <1>       mov    [u.namep], ebp ; 18/07/2015
17944                             <1>              ; mov r2,u.namep / u.namep points to char
17945                             <1>                           ; / following a / or nul
17946                             <1>       ;mov   bx, [u.dirbuf]
17947                             <1>              ; mov u.dirbuf,r1 / move i-node number in directory
17948                             <1>                           ; / entry to r1
17949 0000502C 20C0               <1>       and    al, al
17950                             <1>              ; tst r4 / if r4 = 0 the end of file name reached,
17951                             <1>                    ;  / if r4 = </> then go to next directory
17952                             <1>       ; mov  ax, bx
17953 0000502E 66A1[7A740000]     <1>       mov    ax, [u.dirbuf] ; 17/06/2015
17954 00005034 0F85FDFEFFFF       <1>         jnz     namei_2
17955                             <1>              ; bne 1b
17956                             <1>       ; AX = i-number of the file
17957                             <1> ;;nig:
17958 0000503A C3                 <1>       retn
17959                             <1>              ; tst (r0)+ / gives non-error return
17960                             <1> ;;nib:
17961                             <1> ;;    xor    ax, ax ; Retro UNIX 8086 v1 modification !
17962                             <1>                     ; ax = 0 -> file not found
17963                             <1> ;;    stc    ; 27/05/2013
17964                             <1> ;;    retn
17965                             <1>              ; rts r0
17966                             <1>
17967                             <1> trans_addr_nmbp:
17968                             <1>       ; 18/10/2015
17969                             <1>       ; 12/10/2015
17970 0000503B 8B2D[60740000]     <1>       mov    ebp, [u.namep]
17971                             <1> trans_addr_nm:
17972                             <1>       ; Convert virtual (pathname) address to physical address
17973                             <1>       ; (Retro UNIX 386 v1 feature only !)
17974                             <1>       ; 18/10/2015
17975                             <1>       ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
17976                             <1>       ; 02/07/2015
17977                             <1>       ; 17/06/2015
17978                             <1>       ; 16/06/2015
17979                             <1>       ;
17980                             <1>       ; INPUTS:
17981                             <1>       ;      ebp = pathname address (virtual) ; [u.namep]
17982                             <1>       ;      [u.pgdir] = user's page directory
17983                             <1>       ; OUTPUT:
17984                             <1>       ;       esi = physical address of the pathname
17985                             <1>       ;      ecx = remain byte count in the page
17986                             <1>       ;
17987                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
17988                             <1>       ;
17989 00005041 833D[A5740000]00   <1>         cmp     dword [u.ppgdir], 0  ; /etc/init ? (sysexec)
17990 00005048 7618               <1>       jna    short trans_addr_nmk ; the caller is os kernel;
17991                             <1>                                   ; it is already physical address
17992 0000504A 50                 <1>       push   eax
17993 0000504B 89EB               <1>       mov    ebx, ebp ; [u.namep] ; pathname address (virtual)
17994 0000504D E836E6FFFF         <1>              call   get_physical_addr ; get physical address
17995 00005052 7204               <1>       jc     short tr_addr_nm_err
17996                             <1>       ; 18/10/2015
17997                             <1>       ; eax = physical address
17998                             <1>       ; cx = remain byte count in page (1-4096)
17999                             <1>              ; 12/10/2015 (cx = [u.pncount])
18000 00005054 89C6               <1>       mov    esi, eax ; 12/10/2015 (esi=[u.pnbase])
18001 00005056 58                 <1>       pop    eax
18002 00005057 C3                 <1>       retn
18003                             <1>
18004                             <1> tr_addr_nm_err:
18005 00005058 A3[9D740000]       <1>       mov    [u.error], eax
18006                             <1>       ;pop   eax
18007 0000505D E9D8EFFFFF         <1>       jmp    error
18008                             <1>
18009                             <1> trans_addr_nmk:
18010                             <1>       ; 12/10/2015
18011                             <1>       ; 02/07/2015
18012 00005062 8B35[60740000]     <1>       mov    esi, [u.namep]  ; [u.pnbase]
18013 00005068 66B90010           <1>       mov    cx, PAGE_SIZE ; 4096 ; [u.pncount]
18014 0000506C C3                 <1>       retn
18015                             <1>
18016                             <1> syschdir:
18017                             <1>       ; / makes the directory specified in the argument
```

```
18018                                  <1>       ; / the current directory
18019                                  <1>       ;
18020                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18021                                  <1>       ; 19/06/2013 (Retro UNIX 8086 v1)
18022                                  <1>       ;
18023                                  <1>       ; 'syschdir' makes the directory specified in its argument
18024                                  <1>       ; the current working directory.
18025                                  <1>       ;
18026                                  <1>       ; Calling sequence:
18027                                  <1>       ;     syschdir; name
18028                                  <1>       ; Arguments:
18029                                  <1>       ;     name - address of the path name of a directory
18030                                  <1>       ;            terminated by nul byte.
18031                                  <1>       ; Inputs: -
18032                                  <1>       ; Outputs: -
18033                                  <1>       ; ..........................................................
18034                                  <1>       ;
18035                                  <1>       ; Retro UNIX 8086 v1 modification:
18036                                  <1>       ;      The user/application program puts address of
18037                                  <1>       ;      the path name in BX register as 'syschdir'
18038                                  <1>       ;      system call argument.
18039                                  <1>
18040 0000506D 891D[60740000]           <1>       mov    [u.namep], ebx
18041                                  <1>              ;jsr r0,arg; u.namep / u.namep points to path name
18042 00005073 E87EFEFFFF               <1>       call   namei
18043                                  <1>              ; jsr r0,namei / find its i-number
18044                                  <1>       ;jc    error
18045                                  <1>              ; br error3
18046 00005078 730F                     <1>       jnc    short syschdir0
18047                                  <1>       ; 'directory not found !' error
18048 0000507A C705[9D740000]0C00-      <1>       mov    dword [u.error], ERR_DIR_NOT_FOUND ; 12
18049 00005082 0000                     <1>
18050 00005084 E9B1EFFFFF               <1>       jmp    error
18051                                  <1> syschdir0:
18052 00005089 E833070000               <1>       call   access
18053                                  <1>              ; jsr r0,access; 2 / get i-node into core
18054 0000508E 66F705[16710000]00-      <1>       test   word [i.flgs], 4000h
18055 00005096 40                       <1>
18056                                  <1>              ; bit $40000,i.flgs / is it a directory?
18057                                  <1>       ;jz    error
18058                                  <1>              ; beq error3 / no error
18059 00005097 750F                     <1>       jnz    short syschdir1
18060 00005099 C705[9D740000]1300-      <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
18061 000050A1 0000                     <1>
18062 000050A3 E992EFFFFF               <1>       jmp    error
18063                                  <1> syschdir1:
18064 000050A8 66A3[4C740000]           <1>       mov    [u.cdir], ax
18065                                  <1>              ; mov r1,u.cdir / move i-number to users
18066                                  <1>                                ; / current directory
18067 000050AE 66A1[2E740000]           <1>       mov    ax, [cdev]
18068 000050B4 66A3[92740000]           <1>       mov    [u.cdrv], ax
18069                                  <1>              ; mov cdev,u.cdev / move its device to users
18070                                  <1>                                  ; / current device
18071 000050BA E99BEFFFFF               <1>       jmp    sysret
18072                                  <1>              ; br sysret3
18073                                  <1>
18074                                  <1> syschmod: ; < change mode of file >
18075                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18076                                  <1>       ; 20/06/2013 - 07/07/2013 (Retro UNIX 8086 v1)
18077                                  <1>       ;
18078                                  <1>       ; 'syschmod' changes mode of the file whose name is given as
18079                                  <1>       ; null terminated string pointed to by 'name' has it's mode
18080                                  <1>       ; changed to 'mode'.
18081                                  <1>       ;
18082                                  <1>       ; Calling sequence:
18083                                  <1>       ;     syschmod; name; mode
18084                                  <1>       ; Arguments:
18085                                  <1>       ;     name - address of the file name
18086                                  <1>       ;            terminated by null byte.
18087                                  <1>       ;     mode - (new) mode/flags < attributes >
18088                                  <1>       ;
18089                                  <1>       ; Inputs: -
18090                                  <1>       ; Outputs: -
18091                                  <1>       ; ..........................................................
18092                                  <1>       ;
18093                                  <1>       ; Retro UNIX 8086 v1 modification:
18094                                  <1>       ;      'syschmod' system call has two arguments; so,
18095                                  <1>       ;     * 1st argument, name is pointed to by BX register
18096                                  <1>       ;     * 2nd argument, mode is in CX register
18097                                  <1>       ;
18098                                  <1>       ; Mode bits (Flags):
18099                                  <1>       ;     bit 0 - write permission for non-owner (1)
18100                                  <1>       ;     bit 1 - read permission for non-owner (2)
18101                                  <1>       ;     bit 2 - write permission for owner (4)
18102                                  <1>       ;     bit 3 - read permission for owner (8)
18103                                  <1>       ;     bit 4 - executable flag (16)
18104                                  <1>       ;     bit 5 - set user ID on execution flag (32)
18105                                  <1>       ;     bit 6,7,8,9,10,11 are not used (undefined)
18106                                  <1>       ;     bit 12 - large file flag (4096)
18107                                  <1>       ;     bit 13 - file has modified flag (always on) (8192)
18108                                  <1>       ;     bit 14 - directory flag (16384)
18109                                  <1>       ;     bit 15 - 'i-node is allocated' flag (32768)
18110                                  <1>
18111                                  <1>       ; / name; mode
18112 000050BF E814000000              <1>       call   isown
18113                                  <1>              ;jsr r0,isown / get the i-node and check user status
18114 000050C4 66F705[16710000]00-      <1>       test   word [i.flgs], 4000h
18115 000050CC 40                       <1>
18116                                  <1>              ; bit  $40000,i.flgs / directory?
18117 000050CD 7402                     <1>       jz     short syschmod1
18118                                  <1>              ; beq 2f / no
18119                                  <1>       ; AL = (new) mode
18120 000050CF 24CF                     <1>       and    al, 0CFh ; 11001111b (clears bit 4 & 5)
18121                                  <1>              ; bic $60,r2 / su & ex / yes, clear set user id and
18122                                  <1>                             ; / executable modes
```

```
18123                                  <1> syschmod1: ; 2:
18124 000050D1 A2[16710000]            <1>       mov    [i.flgs], al
18125                                  <1>              ; movb r2,i.flgs / move remaining mode to i.flgs
18126 000050D6 EB42                    <1>       jmp    short isown1
18127                                  <1>              ; br 1f
18128                                  <1>
18129                                  <1> isown:
18130                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18131                                  <1>       ; 04/05/2013 - 07/07/2013 (Retro UNIX 8086 v1)
18132                                  <1>       ;
18133                                  <1>       ; 'isown' is given a file name (the 1st argument).
18134                                  <1>       ;  It find the i-number of that file via 'namei'
18135                                  <1>       ;  then gets the i-node into core via 'iget'.
18136                                  <1>       ;  It then tests to see if the user is super user.
18137                                  <1>       ;  If not, it cheks to see if the user is owner of
18138                                  <1>       ;  the file. If he is not an error occurs.
18139                                  <1>       ;  If user is the owner 'setimod' is called to indicate
18140                                  <1>       ;  the inode has been modificed and the 2nd argument of
18141                                  <1>       ;  the call is put in r2.
18142                                  <1>       ;
18143                                  <1>       ; INPUTS ->
18144                                  <1>       ;    arguments of syschmod and syschown calls
18145                                  <1>       ; OUTPUTS ->
18146                                  <1>       ;    u.uid - id of user
18147                                  <1>       ;    imod - set to a 1
18148                                  <1>       ;    r2 - contains second argument of the system call

18149                                  <1>       ;
18150                                  <1>       ;   ((AX=R2) output as 2nd argument)
18151                                  <1>       ;
18152                                  <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
18153                                  <1>       ;
18154                                  <1>              ; jsr r0,arg2 / u.namep points to file name
18155                                  <1>       ;; ! 2nd argument on top of stack !
18156                                  <1>       ;; 22/06/2015 - 32 bit modifications
18157                                  <1>       ;; 07/07/2013
18158 000050D8 891D[60740000]          <1>       mov    [u.namep], ebx ;; 1st argument
18159 000050DE 51                      <1>       push   ecx ;; 2nd argument
18160                                  <1>       ;;
18161 000050DF E812FEFFFF              <1>       call   namei
18162                                  <1>              ; jsr r0,namei / get its i-number
18163                                  <1>        ; Retro UNIX 8086 v1 modification !
18164                                  <1>        ; ax = 0 -> file not found
18165                                  <1>       ;and   ax, ax
18166                                  <1>       ;jz    error
18167                                  <1>       ;jc    error ; 27/05/2013
18168                                  <1>              ; br error3
18169 000050E4 730F                    <1>       jnc    short isown0
18170                                  <1>       ; 'file not found !' error
18171 000050E6 C705[9D740000]0C00-     <1>       mov    dword [u.error], ERR_FILE_NOT_FOUND ; 12
18172 000050EE 0000                    <1>
18173 000050F0 E945EFFFFF              <1>       jmp    error
18174                                  <1> isown0:
18175 000050F5 E8EC050000              <1>       call   iget
18176                                  <1>              ; jsr r0,iget / get i-node into core
18177 000050FA A0[94740000]            <1>       mov    al, [u.uid] ; 02/08/2013
18178 000050FF 08C0                    <1>       or     al, al
18179                                  <1>              ; tstb u.uid / super user?
18180 00005101 7417                    <1>       jz     short isown1
18181                                  <1>              ; beq 1f / yes, branch
18182 00005103 3A05[19710000]          <1>       cmp    al, [i.uid]
18183                                  <1>              ; cmpb i.uid,u.uid / no, is this the owner of
18184                                  <1>                                 ; / the file
18185                                  <1>       ;jne   error
18186                                  <1>              ; beq 1f / yes
18187                                  <1>              ; jmp error3 / no, error
18188 00005109 740F                    <1>       je     short isown1
18189                                  <1>
18190 0000510B C705[9D740000]0B00-     <1>       mov    dword [u.error], ERR_NOT_OWNER  ; 11
18191 00005113 0000                    <1>
18192                                  <1>                      ;  'permission denied !' error
18193 00005115 E920EFFFFF              <1>       jmp    error
18194                                  <1> isown1: ; 1:
18195 0000511A E8DA060000              <1>       call   setimod
18196                                  <1>              ; jsr r0,setimod / indicates
18197                                  <1>              ;             ; / i-node has been modified
18198 0000511F 58                      <1>       pop    eax ; 2nd argument
18199                                  <1>              ; mov (sp)+,r2 / mode is put in r2
18200                                  <1>                     ; / (u.off put on stack with 2nd arg)
18201 00005120 C3                      <1>       retn
18202                                  <1>              ; rts r0
18203                                  <1>
18204                                  <1> ;;arg:  ; < get system call arguments >
18205                                  <1>       ; 'arg' extracts an argument for a routine whose call is
18206                                  <1>       ; of form:
18207                                  <1>       ;     sys 'routine' ; arg1
18208                                  <1>       ;           or
18209                                  <1>       ;     sys 'routine' ; arg1 ; arg2
18210                                  <1>       ;           or
18211                                  <1>       ;     sys 'routine' ; arg1;...;arg10 (sys exec)
18212                                  <1>       ;
18213                                  <1>       ; INPUTS ->
18214                                  <1>       ;    u.sp+18 - contains a pointer to one of arg1..argn
18215                                  <1>       ;     This pointers's value is actually the value of
18216                                  <1>       ;     update pc at the the trap to sysent (unkni) is
18217                                  <1>       ;     made to process the sys instruction
18218                                  <1>       ;    r0 - contains the return address for the routine
18219                                  <1>       ;     that called arg. The data in the word pointer
18220                                  <1>       ;     to by the return address is used as address
18221                                  <1>       ;     in which the extracted argument is stored
18222                                  <1>       ;
18223                                  <1>       ; OUTPUTS ->
18224                                  <1>       ;    'address' - contains the extracted argument
18225                                  <1>       ;    u.sp+18 - is incremented by 2
18226                                  <1>       ;    r1 - contains the extracted argument
```

```
18227                                  <1>       ;    r0 - points to the next instruction to be
18228                                  <1>       ;      executed in the calling routine.
18229                                  <1>       ;
18230                                  <1>
18231                                  <1>       ; mov u.sp,r1
18232                                  <1>       ; mov *18.(r1),*(r0)+ / put argument of system call
18233                                  <1>                     ; / into argument of arg2
18234                                  <1>       ; add $2,18.(r1) / point pc on stack
18235                                  <1>                          ; / to next system argument
18236                                  <1>       ; rts r0
18237                                  <1>
18238                                  <1> ;;arg2: ; < get system calls arguments - with file name pointer>
18239                                  <1>       ; 'arg2' takes first argument in system call
18240                                  <1>       ;  (pointer to name of the file) and puts it in location
18241                                  <1>       ;  u.namep; takes second argument and puts it in u.off
18242                                  <1>       ;  and on top of the stack
18243                                  <1>       ;
18244                                  <1>       ; INPUTS ->
18245                                  <1>       ;    u.sp, r0
18246                                  <1>       ;
18247                                  <1>       ; OUTPUTS ->
18248                                  <1>       ;    u.namep
18249                                  <1>       ;    u.off
18250                                  <1>       ;    u.off pushed on stack
18251                                  <1>       ;    r1
18252                                  <1>       ;
18253                                  <1>
18254                                  <1>       ; jsr  r0,arg; u.namep / u.namep contains value of
18255                                  <1>                           ; / first arg in sys call
18256                                  <1>       ; jsr r0,arg; u.off / u.off contains value of
18257                                  <1>                           ; / second arg in sys call
18258                                  <1>       ; mov r0,r1 / r0 points to calling routine
18259                                  <1>       ; mov (sp),r0 / put operation code back in r0
18260                                  <1>       ; mov u.off,(sp) / put pointer to second argument
18261                                  <1>                    ; / on stack
18262                                  <1>       ; jmp (r1) / return to calling routine
18263                                  <1>
18264                                  <1> syschown: ; < change owner of file >
18265                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18266                                  <1>       ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18267                                  <1>       ;
18268                                  <1>       ; 'syschown' changes the owner of the file whose name is given
18269                                  <1>       ; as null terminated string pointed to by 'name' has it's owner
18270                                  <1>       ; changed to 'owner'
18271                                  <1>       ;
18272                                  <1>       ; Calling sequence:
18273                                  <1>       ;     syschown; name; owner
18274                                  <1>       ; Arguments:
18275                                  <1>       ;     name - address of the file name
18276                                  <1>       ;            terminated by null byte.
18277                                  <1>       ;     owner - (new) owner (number/ID)
18278                                  <1>       ;
18279                                  <1>       ; Inputs: -
18280                                  <1>       ; Outputs: -
18281                                  <1>       ; ..........................................................
18282                                  <1>       ;
18283                                  <1>       ; Retro UNIX 8086 v1 modification:
18284                                  <1>       ;       'syschown' system call has two arguments; so,
18285                                  <1>       ;     * 1st argument, name is pointed to by BX register
18286                                  <1>       ;     * 2nd argument, owner number is in CX register
18287                                  <1>       ;
18288                                  <1>       ; / name; owner
18289 00005121 E8B2FFFFFF             <1>       call   isown
18290                                  <1>              ; jsr r0,isown / get the i-node and check user status
18291 00005126 803D[94740000]00       <1>       cmp    byte [u.uid], 0 ; 02/08/2013
18292                                  <1>              ; tstb u.uid / super user
18293 0000512D 7418                   <1>       jz     short syschown1
18294                                  <1>              ; beq 2f / yes, 2f
18295 0000512F F605[16710000]20       <1>         test    byte [i.flgs], 20h ; 32
18296                                  <1>              ; bit $40,i.flgs / no, set userid on execution?
18297                                  <1>       ;jnz   error
18298                                  <1>              ; bne 3f / yes error, could create Trojan Horses
18299 00005136 740F                   <1>       jz     short syschown1
18300                                  <1>       ; 'permission denied !'
18301 00005138 C705[9D740000]0B00-     <1>       mov    dword [u.error], ERR_FILE_ACCESS  ; 11
18302 00005140 0000                   <1>
18303 00005142 E9F3EEFFFF             <1>       jmp    error
18304                                  <1> syschown1: ; 2:
18305                                  <1>       ; AL = owner (number/ID)
18306 00005147 A2[19710000]           <1>       mov    [i.uid], al ; 23/06/2015
18307                                  <1>              ;  movb      r2,i.uid / no, put the new owners id
18308                                  <1>                           ; / in the i-node
18309 0000514C E909EFFFFF             <1>       jmp    sysret
18310                                  <1>       ; 1:
18311                                  <1>              ; jmp sysret4
18312                                  <1>       ; 3:
18313                                  <1>              ; jmp  error
18314                                  <1>
18315                                  <1> systime: ; / get time of year
18316                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18317                                  <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
18318                                  <1>       ;
18319                                  <1>       ; 20/06/2013
18320                                  <1>       ; 'systime' gets the time of the year.
18321                                  <1>       ; The present time is put on the stack.
18322                                  <1>       ;
18323                                  <1>       ; Calling sequence:
18324                                  <1>       ;     systime
18325                                  <1>       ; Arguments: -
18326                                  <1>       ;
18327                                  <1>       ; Inputs: -
18328                                  <1>       ; Outputs: sp+2, sp+4 - present time
18329                                  <1>       ; ..........................................................
18330                                  <1>       ;
18331                                  <1>       ; Retro UNIX 8086 v1 modification:
```

```
18332                                  <1>       ;       'systime' system call will return to the user
18333                                  <1>       ;     with unix time (epoch) in DX:AX register pair
18334                                  <1>       ;
18335                                  <1>       ;     !! Major modification on original Unix v1 'systime'
18336                                  <1>       ;     system call for PC compatibility !!
18337                                  <1>
18338 00005151 E81FEAFFFF             <1>       call  epoch
18339 00005156 A3[48740000]            <1>       mov   [u.r0], eax
18340                                  <1>             ; mov s.time,4(sp)
18341                                  <1>             ; mov s.time+2,2(sp) / put the present time
18342                                  <1>                           ; / on the stack
18343                                  <1>             ; br sysret4
18344 0000515B E9FAEEFFFF             <1>       jmp   sysret
18345                                  <1>
18346                                  <1> sysstime: ; / set time
18347                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18348                                  <1>       ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18349                                  <1>       ;
18350                                  <1>       ; 'sysstime' sets the time. Only super user can use this call.
18351                                  <1>       ;
18352                                  <1>       ; Calling sequence:
18353                                  <1>       ;     sysstime
18354                                  <1>       ; Arguments: -
18355                                  <1>       ;
18356                                  <1>       ; Inputs: sp+2, sp+4 - time system is to be set to.
18357                                  <1>       ; Outputs: -
18358                                  <1>       ; ..........................................................
18359                                  <1>       ;
18360                                  <1>       ; Retro UNIX 8086 v1 modification:
18361                                  <1>       ;     the user calls 'sysstime' with unix (epoch) time
18362                                  <1>       ;     (to be set) is in CX:BX register pair as two arguments.
18363                                  <1>       ;
18364                                  <1>       ;     Retro UNIX 8086 v1 argument transfer method 2 is used
18365                                  <1>       ;     to get sysstime system call arguments from the user;
18366                                  <1>       ;     * 1st argument, lowword of unix time is in BX register
18367                                  <1>       ;     * 2nd argument, highword of unix time is in CX register

18368                                  <1>       ;
18369                                  <1>       ;     !! Major modification on original Unix v1 'sysstime'
18370                                  <1>       ;     system call for PC compatibility !!
18371                                  <1>
18372 00005160 803D[94740000]00        <1>       cmp   byte [u.uid], 0
18373                                  <1>             ; tstb u.uid / is user the super user
18374                                  <1>       ;ja   error
18375                                  <1>             ; bne error4 / no, error
18376 00005167 760F                    <1>       jna   short systime1
18377                                  <1>       ; 'permission denied !'
18378 00005169 C705[9D740000]0B00-     <1>       mov   dword [u.error], ERR_NOT_SUPERUSER  ; 11
18379 00005171 0000                    <1>
18380 00005173 E9C2EEFFFF              <1>       jmp   error
18381                                  <1> systime1:
18382                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - 32 bit version)
18383                                  <1>       ; EBX = unix (epoch) time (from user)
18384 00005178 89D8                    <1>       mov   eax, ebx
18385 0000517A E878EBFFFF              <1>       call  set_date_time
18386                                  <1>             ; mov 4(sp),s.time
18387                                  <1>             ; mov 2(sp),s.time+2 / set the system time
18388 0000517F E9D6EEFFFF              <1>       jmp   sysret
18389                                  <1>             ; br sysret4
18390                                  <1>
18391                                  <1> sysbreak:
18392                                  <1>       ; 18/10/2015
18393                                  <1>       ; 07/10/2015
18394                                  <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18395                                  <1>       ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
18396                                  <1>       ;
18397                                  <1>       ; 'sysbreak' sets the programs break points.
18398                                  <1>       ; It checks the current break point (u.break) to see if it is
18399                                  <1>       ; between "core" and the stack (sp). If it is, it is made an
18400                                  <1>       ; even address (if it was odd) and the area between u.break
18401                                  <1>       ; and the stack is cleared. The new breakpoint is then put
18402                                  <1>       ; in u.break and control is passed to 'sysret'.
18403                                  <1>       ;
18404                                  <1>       ; Calling sequence:
18405                                  <1>       ;     sysbreak; addr
18406                                  <1>       ; Arguments: -
18407                                  <1>       ;
18408                                  <1>       ; Inputs: u.break - current breakpoint
18409                                  <1>       ; Outputs: u.break - new breakpoint
18410                                  <1>       ;     area between old u.break and the stack (sp) is cleared.
18411                                  <1>       ; ..........................................................
18412                                  <1>       ;
18413                                  <1>       ; Retro UNIX 8086 v1 modification:
18414                                  <1>       ;     The user/application program puts breakpoint address
18415                                  <1>       ;      in BX register as 'sysbreak' system call argument.
18416                                  <1>       ;     (argument transfer method 1)
18417                                  <1>       ;
18418                                  <1>       ;  NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
18419                                  <1>       ;     ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))
18420                                  <1>       ;  NOTE:
18421                                  <1>       ;     'sysbreak' clears extended part (beyond of previous
18422                                  <1>       ;     'u.break' address) of user's memory for original unix's
18423                                  <1>       ;     'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
18424                                  <1>
18425                                  <1>             ; mov u.break,r1 / move users break point to r1
18426                                  <1>             ; cmp r1,$core / is it the same or lower than core?
18427                                  <1>             ; blos 1f / yes, 1f
18428                                  <1>       ; 23/06/2015
18429 00005184 8B2D[74740000]          <1>       mov   ebp, [u.break] ; virtual address (offset)
18430                                  <1>       ;and  ebp, ebp
18431                                  <1>       ;jz   short sysbreak_3
18432                                  <1>       ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
18433                                  <1>       ; (Even break point address is not needed for Retro UNIX 386 v1)
18434 0000518A 8B15[40740000]          <1>       mov   edx, [u.sp] ; kernel stack at the beginning of sys call
18435 00005190 83C20C                  <1>       add   edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
```

```
18436                                  <1>       ; 07/10/2015
18437 00005193 891D[74740000]          <1>       mov    [u.break], ebx ; virtual address !!!
18438                                  <1>       ;
18439 00005199 3B1A                    <1>       cmp    ebx, [edx] ; compare new break point with
18440                                  <1>                         ; with top of user's stack (virtual!)
18441 0000519B 7327                    <1>       jnb    short sysbreak_3
18442                                  <1>              ; cmp r1,sp / is it the same or higher
18443                                  <1>                     ; / than the stack?
18444                                  <1>              ; bhis 1f / yes, 1f
18445 0000519D 89DE                    <1>       mov    esi, ebx
18446 0000519F 29EE                    <1>       sub    esi, ebp ; new break point - old break point
18447 000051A1 7621                    <1>       jna    short sysbreak_3
18448                                  <1>       ;push  ebx
18449                                  <1> sysbreak_1:
18450 000051A3 89EB                    <1>       mov    ebx, ebp
18451 000051A5 E8DEE4FFFF              <1>       call   get_physical_addr ; get physical address
18452 000051AA 0F82A8FEFFFF            <1>       jc     tr_addr_nm_err
18453                                  <1>       ; 18/10/2015
18454 000051B0 89C7                    <1>       mov    edi, eax
18455 000051B2 29C0                    <1>       sub    eax, eax ; 0
18456                                  <1>              ; ECX = remain byte count in page (1-4096)
18457 000051B4 39CE                    <1>       cmp    esi, ecx
18458 000051B6 7302                    <1>       jnb    short sysbreak_2
18459 000051B8 89F1                    <1>       mov    ecx, esi
18460                                  <1> sysbreak_2:
18461 000051BA 29CE                    <1>       sub    esi, ecx
18462 000051BC 01CD                    <1>       add    ebp, ecx
18463 000051BE F3AA                    <1>       rep    stosb
18464 000051C0 09F6                    <1>       or     esi, esi
18465 000051C2 75DF                    <1>       jnz    short sysbreak_1
18466                                  <1>       ;
18467                                  <1>              ; bit $1,r1 / is it an odd address
18468                                  <1>              ; beq 2f / no, its even
18469                                  <1>              ; clrb (r1)+ / yes, make it even
18470                                  <1>       ; 2: / clear area between the break point and the stack
18471                                  <1>              ; cmp r1,sp / is it higher or same than the stack
18472                                  <1>              ; bhis 1f / yes, quit
18473                                  <1>              ; clr (r1)+ / clear word
18474                                  <1>              ; br 2b / go back
18475                                  <1>       ;pop   ebx
18476                                  <1> sysbreak_3: ; 1:
18477                                  <1>       ;mov   [u.break], ebx ; virtual address !!!
18478                                  <1>              ; jsr r0,arg; u.break / put the "address"
18479                                  <1>                    ; / in u.break (set new break point)
18480                                  <1>              ; br sysret4 / br sysret
18481 000051C4 E991EEFFFF              <1>       jmp    sysret
18482                                  <1>
18483                                  <1>
18484                                  <1> maknod:
18485                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18486                                  <1>       ; 02/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18487                                  <1>       ;
18488                                  <1>       ; 'maknod' creates an i-node and makes a directory entry
18489                                  <1>       ; for this i-node in the current directory.
18490                                  <1>       ;
18491                                  <1>       ; INPUTS ->
18492                                  <1>       ;    r1 - contains mode
18493                                  <1>       ;    ii - current directory's i-number
18494                                  <1>       ;
18495                                  <1>       ; OUTPUTS ->
18496                                  <1>       ;    u.dirbuf - contains i-number of free i-node
18497                                  <1>       ;    i.flgs - flags in new i-node
18498                                  <1>       ;    i.uid - filled with u.uid
18499                                  <1>       ;    i.nlks - 1 is put in the number of links
18500                                  <1>       ;    i.ctim - creation time
18501                                  <1>       ;    i.ctim+2 - modification time
18502                                  <1>       ;    imod - set via call to setimod
18503                                  <1>       ;
18504                                  <1>       ; ((AX = R1)) input
18505                                  <1>       ;
18506                                  <1>       ; (Retro UNIX Prototype :
18507                                  <1>       ;     30/10/2012 - 01/03/2013, UNIXCOPY.ASM)
18508                                  <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
18509                                  <1>
18510                                  <1>       ; / r1 contains the mode
18511 000051C9 80CC80                  <1>       or     ah, 80h  ; 10000000b
18512                                  <1>              ; bis  $100000,r1 / allocate flag set
18513 000051CC 6650                    <1>       push   ax
18514                                  <1>              ; mov r1,-(sp) / put mode on stack
18515                                  <1>       ; 31/07/2013
18516 000051CE 66A1[2A740000]          <1>       mov    ax, [ii] ; move current i-number to AX/r1
18517                                  <1>              ; mov ii,r1 / move current i-number to r1
18518 000051D4 B201                    <1>       mov    dl, 1 ; owner flag mask
18519 000051D6 E8E6050000              <1>       call   access
18520                                  <1>              ; jsr r0,access; 1 / get its i-node into core
18521 000051DB 6650                    <1>       push   ax
18522                                  <1>              ; mov r1,-(sp) / put i-number on stack
18523 000051DD 66B82800                <1>       mov    ax, 40
18524                                  <1>              ; mov $40.,r1 / r1 = 40
18525                                  <1> maknod1: ; 1: / scan for a free i-node (next 4 instructions)
18526 000051E1 6640                    <1>       inc    ax
18527                                  <1>              ; inc r1 / r1 = r1 + 1
18528 000051E3 E8AA060000              <1>       call   imap
18529                                  <1>              ; jsr r0,imap / get byte address and bit position in
18530                                  <1>                           ; /      inode map in r2 & m
18531                                  <1>           ; DX (MQ) has a 1 in the calculated bit position
18532                                  <1>           ; eBX (R2) has byte address of the byte with allocation bit
18533                                  <1>       ; 22/06/2015 - NOTE for next Retro UNIX version:
18534                                  <1>       ;             Inode count must be checked here
18535                                  <1>       ; (Original UNIX v1 did not check inode count here !?)
18536 000051E8 8413                    <1>       test   [ebx], dl
18537                                  <1>              ; bitb mq,(r2) / is the i-node active
18538 000051EA 75F5                    <1>       jnz    short maknod1
18539                                  <1>              ; bne 1b / yes, try the next one
18540 000051EC 0813                    <1>       or     [ebx], dl
```

```
18541                                  <1>             ; bisb mq,(r2) / no, make it active
18542                                  <1>                             ; / (put a 1 in the bit map)
18543 000051EE E8F3040000             <1>       call  iget
18544                                  <1>             ; jsr r0,iget / get i-node into core
18545 000051F3 66F705[16710000]00-     <1>       test  word [i.flgs], 8000h
18546 000051FB 80                      <1>
18547                                  <1>             ; tst i.flgs / is i-node already allocated
18548 000051FC 75E3                    <1>       jnz   short maknod1
18549                                  <1>             ; blt 1b / yes, look for another one
18550 000051FE 66A3[7A740000]          <1>       mov   [u.dirbuf], ax
18551                                  <1>             ; mov r1,u.dirbuf / no, put i-number in u.dirbuf
18552 00005204 6658                    <1>       pop   ax
18553                                  <1>             ; mov (sp)+,r1 / get current i-number back
18554 00005206 E8DB040000              <1>       call  iget
18555                                  <1>             ; jsr r0,iget / get i-node in core
18556 0000520B E87DF7FFFF              <1>       call  mkdir
18557                                  <1>             ; jsr r0,mkdir / make a directory entry
18558                                  <1>                             ; / in current directory
18559 00005210 66A1[7A740000]          <1>       mov   ax, [u.dirbuf]
18560                                  <1>             ; mov u.dirbuf,r1 / r1 = new inode number
18561 00005216 E8CB040000              <1>       call  iget
18562                                  <1>             ; jsr r0,iget / get it into core
18563                                  <1>             ; jsr r0,copyz; inode; inode+32. / 0 it out
18564 0000521B B908000000              <1>       mov   ecx, 8
18565 00005220 31C0                    <1>       xor   eax, eax ; 0
18566 00005222 BF[16710000]            <1>       mov   edi, inode
18567 00005227 F3AB                    <1>       rep   stosd
18568                                  <1>       ;
18569 00005229 668F05[16710000]        <1>       pop   word [i.flgs]
18570                                  <1>             ; mov (sp)+,i.flgs / fill flags
18571 00005230 8A0D[94740000]          <1>       mov   cl, [u.uid] ; 02/08/2013
18572 00005236 880D[19710000]          <1>       mov   [i.uid], cl
18573                                  <1>             ; movb u.uid,i.uid / user id
18574 0000523C C605[18710000]01        <1>       mov     byte [i.nlks], 1
18575                                  <1>             ; movb $1,i.nlks / 1 link
18576                                  <1>       ;call  epoch ; Retro UNIX 8086 v1 modification !
18577                                  <1>       ;mov   eax, [s.time]
18578                                  <1>       ;mov   [i.ctim], eax
18579                                  <1>             ; mov s.time,i.ctim / time created
18580                                  <1>             ; mov s.time+2,i.ctim+2 / time modified
18581                                  <1>       ; Retro UNIX 8086 v1 modification !
18582                                  <1>       ; i.ctime=0, i.ctime+2=0 and
18583                                  <1>         ; 'setimod' will set ctime of file via 'epoch'
18584 00005243 E8B1050000              <1>       call setimod
18585                                  <1>             ; jsr r0,setimod / set modified flag
18586 00005248 C3                      <1>       retn
18587                                  <1>             ; rts r0 / return
18588                                  <1>
18589                                  <1> sysseek: ; / moves read write pointer in an fsp entry
18590                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18591                                  <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18592                                  <1>       ;
18593                                  <1>       ; 'sysseek' changes the r/w pointer of (3rd word of in an
18594                                  <1>       ; fsp entry) of an open file whose file descriptor is in u.r0.
18595                                  <1>       ; The file descriptor refers to a file open for reading or
18596                                  <1>       ; writing. The read (or write) pointer is set as follows:
18597                                  <1>       ;     * if 'ptrname' is 0, the pointer is set to offset.
18598                                  <1>       ;     * if 'ptrname' is 1, the pointer is set to its
18599                                  <1>       ;       current location plus offset.
18600                                  <1>       ;     * if 'ptrname' is 2, the pointer is set to the
18601                                  <1>       ;       size of file plus offset.
18602                                  <1>       ; The error bit (e-bit) is set for an undefined descriptor.
18603                                  <1>       ;
18604                                  <1>       ; Calling sequence:
18605                                  <1>       ;     sysseek; offset; ptrname
18606                                  <1>       ; Arguments:
18607                                  <1>       ;     offset - number of bytes desired to move
18608                                  <1>       ;              the r/w pointer
18609                                  <1>       ;     ptrname - a switch indicated above
18610                                  <1>       ;
18611                                  <1>       ; Inputs: r0 - file descriptor
18612                                  <1>       ; Outputs: -
18613                                  <1>       ; ..........................................................
18614                                  <1>       ;
18615                                  <1>       ; Retro UNIX 8086 v1 modification:
18616                                  <1>       ;       'sysseek' system call has three arguments; so,
18617                                  <1>       ;     * 1st argument, file descriptor is in BX (BL) register
18618                                  <1>       ;     * 2nd argument, offset is in CX register
18619                                  <1>       ;     * 3rd argument, ptrname/switch is in DX (DL) register
18620                                  <1>       ;
18621                                  <1>
18622 00005249 E823000000              <1>       call  seektell
18623                                  <1>       ; AX = u.count
18624                                  <1>       ; BX = *u.fofp
18625                                  <1>             ; jsr r0,seektell / get proper value in u.count
18626                                  <1>             ; add u.base,u.count / add u.base to it
18627 0000524E 0305[68740000]          <1>       add   eax, [u.base] ; add offset (u.base) to base
18628 00005254 8903                    <1>       mov   [ebx], eax
18629                                  <1>             ; mov u.count,*u.fofp / put result into r/w pointer
18630 00005256 E9FFEDFFFF              <1>       jmp   sysret
18631                                  <1>             ; br sysret4
18632                                  <1>
18633                                  <1> systell: ; / get the r/w pointer
18634                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18635                                  <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18636                                  <1>       ;
18637                                  <1>       ; Retro UNIX 8086 v1 modification:
18638                                  <1>       ; ! 'systell' does not work in original UNIX v1,
18639                                  <1>       ;         it returns with error !
18640                                  <1>       ; Inputs: r0 - file descriptor
18641                                  <1>       ; Outputs: r0 - file r/w pointer
18642                                  <1>
18643                                  <1>       ;xor   ecx, ecx ; 0
18644 0000525B BA01000000              <1>       mov   edx, 1 ; 05/08/2013
18645                                  <1>       ;call  seektell
```

```
18646 00005260 E812000000          <1>       call   seektell0 ; 05/08/2013
18647                              <1>       ;mov   ebx, [u.fofp]
18648 00005265 8B03                <1>       mov    eax, [ebx]
18649 00005267 A3[48740000]        <1>       mov    [u.r0], eax
18650 0000526C E9E9EDFFFF          <1>       jmp    sysret
18651                              <1>
18652                              <1> ; Original unix v1 'systell' system call:
18653                              <1>             ; jsr r0,seektell
18654                              <1>             ; br error4
18655                              <1>
18656                              <1> seektell:
18657                              <1>       ; 03/01/2016
18658                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18659                              <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18660                              <1>       ;
18661                              <1>       ; 'seektell' puts the arguments from sysseek and systell
18662                              <1>       ; call in u.base and u.count. It then gets the i-number of
18663                              <1>       ; the file from the file descriptor in u.r0 and by calling
18664                              <1>       ; getf. The i-node is brought into core and then u.count
18665                              <1>       ; is checked to see it is a 0, 1, or 2.
18666                              <1>       ; If it is 0 - u.count stays the same
18667                              <1>       ;          1 - u.count = offset (u.fofp)
18668                              <1>       ;         2 - u.count = i.size (size of file)
18669                              <1>       ;
18670                              <1>       ; !! Retro UNIX 8086 v1 modification:
18671                              <1>       ;     Argument 1, file descriptor is in BX;
18672                              <1>       ;     Argument 2, offset is in CX;
18673                              <1>       ;     Argument 3, ptrname/switch is in DX register.
18674                              <1>       ;
18675                              <1>       ; mov  ax, 3 ; Argument transfer method 3 (three arguments)
18676                              <1>       ; call       arg
18677                              <1>       ;
18678                              <1>       ; ((Return -> ax = base for offset (position= base+offset))
18679                              <1>       ;
18680 00005271 890D[68740000]      <1>       mov    [u.base], ecx ; offset
18681                              <1>             ; jsr r0,arg; u.base / puts offset in u.base
18682                              <1> seektell0:
18683 00005277 8915[6C740000]      <1>       mov    [u.count], edx
18684                              <1>             ; jsr r0,arg; u.count / put ptr name in u.count
18685                              <1>       ; mov  ax, bx
18686                              <1>             ; mov *u.r0,r1 / file descriptor in r1
18687                              <1>                       ; / (index in u.fp list)
18688                              <1>       ; call getf
18689                              <1>             ; jsr r0,getf / u.fofp points to 3rd word in fsp entry
18690                              <1>       ; BX = file descriptor (file number)
18691 0000527D E83DFCFFFF          <1>       call   getf1
18692 00005282 6609C0              <1>       or     ax, ax ; i-number of the file
18693                              <1>             ; mov r1,-(sp) / r1 has i-number of file,
18694                              <1>                          ; / put it on the stack
18695                              <1>       ;jz    error
18696                              <1>             ; beq error4 / if i-number is 0, not active so error
18697 00005285 750F                <1>       jnz    short seektell1
18698 00005287 C705[9D740000]0A00- <1>       mov    dword [u.error], ERR_FILE_NOT_OPEN  ; 'file not open !'
18699 0000528F 0000                <1>
18700 00005291 E9A4EDFFFF          <1>       jmp    error
18701                              <1> seektell1:
18702                              <1>       ;push  eax
18703 00005296 80FC80              <1>       cmp    ah, 80h
18704 00005299 7203                <1>       jb     short seektell2
18705                              <1>             ; bgt .+4 / if its positive jump
18706 0000529B 66F7D8              <1>       neg    ax
18707                              <1>             ; neg r1 / if not make it positive
18708                              <1> seektell2:
18709 0000529E E843040000          <1>       call   iget
18710                              <1>             ; jsr r0,iget / get its i-node into core
18711 000052A3 8B1D[58740000]      <1>         mov    ebx, [u.fofp] ; 05/08/2013
18712 000052A9 803D[6C740000]01    <1>       cmp    byte [u.count], 1
18713                              <1>             ; cmp u.count,$1 / is ptr name =1
18714 000052B0 7705                <1>       ja     short seektell3
18715                              <1>             ; blt 2f / no its zero
18716 000052B2 740A                <1>       je     short seektell_4
18717                              <1>             ; beq 1f / yes its 1
18718 000052B4 31C0                <1>       xor    eax, eax
18719                              <1>       ;jmp   short seektell_5
18720 000052B6 C3                  <1>       retn
18721                              <1> seektell3:
18722                              <1>       ; 03/01/2016
18723                              <1>       ;movzx eax, word [i.size]
18724 000052B7 66A1[1A710000]      <1>         mov        ax, [i.size]
18725                              <1>                ; mov i.size,u.count /  put number of bytes
18726                              <1>                                     ; / in file in u.count
18727                              <1>       ;jmp   short seektell_5
18728                              <1>             ; br 2f
18729 000052BD C3                  <1>       retn
18730                              <1> seektell_4: ; 1: / ptrname =1
18731                              <1>       ;mov   ebx, [u.fofp]
18732 000052BE 8B03                <1>       mov    eax, [ebx]
18733                              <1>             ; mov *u.fofp,u.count / put offset in u.count
18734                              <1> ;seektell_5: ; 2: / ptrname =0
18735                              <1>       ;mov   [u.count], eax
18736                              <1>       ;pop   eax
18737                              <1>             ; mov (sp)+,r1 / i-number on stack  r1
18738 000052C0 C3                  <1>       retn
18739                              <1>             ; rts r0
18740                              <1>
18741                              <1> sysintr: ; / set interrupt handling
18742                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18743                              <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
18744                              <1>       ;
18745                              <1>       ; 'sysintr' sets the interrupt handling value. It puts
18746                              <1>       ; argument of its call in u.intr then branches into 'sysquit'
18747                              <1>       ; routine. u.tty is checked if to see if a control tty exists.
18748                              <1>       ; If one does the interrupt character in the tty buffer is
18749                              <1>       ; cleared and 'sysret'is called. If one does not exits
18750                              <1>       ; 'sysret' is just called.
```

```
18751                                  <1>       ;
18752                                  <1>       ; Calling sequence:
18753                                  <1>       ;     sysintr; arg
18754                                  <1>       ; Argument:
18755                                  <1>       ;     arg - if 0, interrupts (ASCII DELETE) are ignored.
18756                                  <1>       ;         - if 1, intterupts cause their normal result
18757                                  <1>       ;             i.e force an exit.
18758                                  <1>       ;         - if arg is a location within the program,
18759                                  <1>       ;           control is passed to that location when
18760                                  <1>       ;           an interrupt occurs.
18761                                  <1>       ; Inputs: -
18762                                  <1>       ; Outputs: -
18763                                  <1>       ; ..........................................................
18764                                  <1>       ;
18765                                  <1>       ; Retro UNIX 8086 v1 modification:
18766                                  <1>       ;       'sysintr' system call sets u.intr to value of BX
18767                                  <1>       ;     then branches into sysquit.
18768                                  <1>       ;
18769 000052C1 66891D[8C740000]       <1>       mov   [u.intr], bx
18770                                  <1>             ; jsr r0,arg; u.intr / put the argument in u.intr
18771                                  <1>             ; br 1f / go into quit routine
18772 000052C8 E98DEDFFFF            <1>       jmp   sysret
18773                                  <1>
18774                                  <1> sysquit:
18775                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18776                                  <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
18777                                  <1>       ;
18778                                  <1>       ; 'sysquit' turns off the quit signal. it puts the argument of
18779                                  <1>       ; the call in u.quit. u.tty is checked if to see if a control
18780                                  <1>       ; tty exists. If one does the interrupt character in the tty
18781                                  <1>       ; buffer is cleared and 'sysret'is called. If one does not exits
18782                                  <1>       ; 'sysret' is just called.
18783                                  <1>       ;
18784                                  <1>       ; Calling sequence:
18785                                  <1>       ;     sysquit; arg
18786                                  <1>       ; Argument:
18787                                  <1>       ;     arg - if 0, this call diables quit signals from the
18788                                  <1>       ;           typewriter (ASCII FS)
18789                                  <1>       ;         - if 1, quits are re-enabled and cause execution to
18790                                  <1>       ;           cease and a core image to be produced.
18791                                  <1>       ;            i.e force an exit.
18792                                  <1>       ;         - if arg is an addres in the program,
18793                                  <1>       ;           a quit causes control to sent to that
18794                                  <1>       ;           location.
18795                                  <1>       ; Inputs: -
18796                                  <1>       ; Outputs: -
18797                                  <1>       ; ..........................................................
18798                                  <1>       ;
18799                                  <1>       ; Retro UNIX 8086 v1 modification:
18800                                  <1>       ;       'sysquit' system call sets u.quit to value of BX
18801                                  <1>       ;     then branches into 'sysret'.
18802                                  <1>       ;
18803 000052CD 66891D[8E740000]       <1>       mov   [u.quit], bx
18804 000052D4 E981EDFFFF            <1>       jmp   sysret
18805                                  <1>             ; jsr r0,arg; u.quit / put argument in u.quit
18806                                  <1>       ;1:
18807                                  <1>             ; mov u.ttyp,r1 / move pointer to control tty buffer
18808                                  <1>                             ; / to r1
18809                                  <1>             ; beq sysret4 / return to user
18810                                  <1>             ; clrb 6(r1) / clear the interrupt character
18811                                  <1>                      ; / in the tty buffer
18812                                  <1>             ; br sysret4 / return to user
18813                                  <1>
18814                                  <1> syssetuid: ; / set process id
18815                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18816                                  <1>       ; 07/07/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18817                                  <1>       ;
18818                                  <1>       ; 'syssetuid' sets the user id (u.uid) of the current process
18819                                  <1>       ; to the process id in (u.r0). Both the effective user and
18820                                  <1>       ; u.uid and the real user u.ruid are set to this.
18821                                  <1>       ; Only the super user can make this call.
18822                                  <1>       ;
18823                                  <1>       ; Calling sequence:
18824                                  <1>       ;     syssetuid
18825                                  <1>       ; Arguments: -
18826                                  <1>       ;
18827                                  <1>       ; Inputs: (u.r0) - contains the process id.
18828                                  <1>       ; Outputs: -
18829                                  <1>       ; ..........................................................
18830                                  <1>       ;
18831                                  <1>       ; Retro UNIX 8086 v1 modification:
18832                                  <1>       ;       BL contains the (new) user ID of the current process
18833                                  <1>
18834                                  <1>             ; movb *u.r0,r1 / move process id (number) to r1
18835 000052D9 3A1D[95740000]         <1>       cmp   bl, [u.ruid]
18836                                  <1>             ; cmpb r1,u.ruid / is it equal to the real user
18837                                  <1>                              ; / id number
18838 000052DF 741E                  <1>       je    short setuid1
18839                                  <1>             ; beq 1f / yes
18840 000052E1 803D[94740000]00       <1>       cmp   byte [u.uid], 0 ; 02/08/2013
18841                                  <1>             ; tstb u.uid / no, is current user the super user?
18842                                  <1>       ;ja   error
18843                                  <1>             ; bne error4 / no, error
18844 000052E8 760F                  <1>       jna   short setuid0
18845 000052EA C705[9D740000]0B00-    <1>       mov   dword [u.error], ERR_NOT_SUPERUSER  ; 11
18846 000052F2 0000                  <1>
18847                                  <1>                             ;  'permission denied !' error
18848 000052F4 E941EDFFFF            <1>       jmp   error
18849                                  <1> setuid0:
18850 000052F9 881D[95740000]         <1>       mov   [u.ruid], bl
18851                                  <1> setuid1: ; 1:
18852 000052FF 881D[94740000]         <1>       mov   [u.uid], bl ; 02/08/2013
18853                                  <1>             ; movb r1,u.uid / put process id in u.uid
18854                                  <1>             ; movb r1,u.ruid / put process id in u.ruid
18855 00005305 E950EDFFFF            <1>       jmp   sysret
```

```
18856                                  <1>             ; br sysret4 / system return
18857                                  <1>
18858                                  <1> sysgetuid: ; < get user id >
18859                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18860                                  <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
18861                                  <1>       ;
18862                                  <1>       ; 'sysgetuid' returns the real user ID of the current process.
18863                                  <1>       ; The real user ID identifies the person who is logged in,
18864                                  <1>       ; in contradistinction to the effective user ID, which
18865                                  <1>       ; determines his access permission at each moment. It is thus
18866                                  <1>       ; useful to programs which operate using the 'set user ID'
18867                                  <1>       ; mode, to find out who invoked them.
18868                                  <1>       ;
18869                                  <1>       ; Calling sequence:
18870                                  <1>       ;     syssetuid
18871                                  <1>       ; Arguments: -
18872                                  <1>       ;
18873                                  <1>       ; Inputs: -
18874                                  <1>       ; Outputs: (u.r0) - contains the real user's id.
18875                                  <1>       ; ..........................................................
18876                                  <1>       ;
18877                                  <1>       ; Retro UNIX 8086 v1 modification:
18878                                  <1>       ;      AL contains the real user ID at return.
18879                                  <1>       ;
18880 0000530A 0FB605[95740000]       <1>       movzx  eax, byte [u.ruid]
18881 00005311 A3[48740000]           <1>       mov    [u.r0], eax
18882                                  <1>              ; movb u.ruid,*u.r0 / move the real user id to (u.r0)
18883 00005316 E93FEDFFFF             <1>       jmp    sysret
18884                                  <1>              ; br sysret4 / systerm return, sysret
18885                                  <1>
18886                                  <1> anyi:
18887                                  <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18888                                  <1>       ; 25/04/2013 (Retro UNIX 8086 v1)
18889                                  <1>       ;
18890                                  <1>       ; 'anyi' is called if a file deleted while open.
18891                                  <1>       ; "anyi" checks to see if someone else has opened this file.
18892                                  <1>       ;
18893                                  <1>       ; INPUTS ->
18894                                  <1>       ;    r1 - contains an i-number
18895                                  <1>       ;    fsp - start of table containing open files
18896                                  <1>       ;
18897                                  <1>       ; OUTPUTS ->
18898                                  <1>       ;    "deleted" flag set in fsp entry of another occurrence of
18899                                  <1>       ;         this file and r2 points 1st word of this fsp entry.
18900                                  <1>       ;    if file not found - bit in i-node map is cleared
18901                                  <1>       ;                        (i-node is freed)
18902                                  <1>       ;                  all blocks related to i-node are freed
18903                                  <1>       ;                all flags in i-node are cleared
18904                                  <1>       ; ((AX = R1)) input
18905                                  <1>       ;
18906                                  <1>       ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
18907                                  <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
18908                                  <1>       ;
18909                                  <1>              ; / r1 contains an i-number
18910 0000531B BB[16720000]           <1>       mov    ebx, fsp
18911                                  <1>              ; mov $fsp,r2 / move start of fsp table to r2
18912                                  <1> anyi_1: ; 1:
18913 00005320 663B03                 <1>       cmp    ax, [ebx]
18914                                  <1>              ; cmp r1,(r2) / do i-numbers match?
18915 00005323 7433                   <1>       je     short anyi_3
18916                                  <1>              ; beq 1f / yes, 1f
18917 00005325 66F7D8                 <1>       neg    ax
18918                                  <1>              ; neg r1 / no complement r1
18919 00005328 663B03                 <1>       cmp    ax, [ebx]
18920                                  <1>              ; cmp r1,(r2) / do they match now?
18921 0000532B 742B                   <1>       je     short anyi_3
18922                                  <1>              ; beq 1f / yes, transfer
18923                                  <1>              ; / i-numbers do not match
18924 0000532D 83C30A                 <1>       add    ebx, 10 ; fsp table size is 10 bytes
18925                                  <1>                     ; in Retro UNIX 386 v1 (22/06/2015)
18926                                  <1>              ; add $8,r2 / no, bump to next entry in fsp table
18927 00005330 81FB[0A740000]         <1>       cmp    ebx, fsp + (nfiles*10) ; 22/06/2015
18928                                  <1>              ; cmp r2,$fsp+[nfiles*8]
18929                                  <1>                              ; / are we at last entry in the table
18930 00005336 72E8                   <1>       jb     short anyi_1
18931                                  <1>              ; blt 1b / no, check next entries i-number
18932                                  <1>       ;cmp   ax, 32768
18933 00005338 80FC80                 <1>       cmp    ah, 80h ; negative number check
18934                                  <1>              ; tst r1 / yes, no match
18935                                  <1>              ; bge .+4
18936 0000533B 7203                   <1>       jb     short anyi_2
18937 0000533D 66F7D8                 <1>       neg    ax
18938                                  <1>              ; neg r1 / make i-number positive
18939                                  <1> anyi_2:
18940 00005340 E84D050000             <1>       call   imap
18941                                  <1>              ; jsr r0,imap / get address of allocation bit
18942                                  <1>                           ; / in the i-map in r2
18943                                  <1>       ;; DL/DX (MQ) has a 1 in the calculated bit position
18944                                  <1>         ;; eBX (R2) has address of the byte with allocation bit
18945                                  <1>       ; not  dx
18946 00005345 F6D2                   <1>       not    dl ;; 0 at calculated bit position, other bits are 1
18947                                  <1>         ;and [ebx], dx
18948 00005347 2013                   <1>       and    [ebx], dl
18949                                  <1>              ; bicb mq,(r2) / clear bit for i-node in the imap
18950 00005349 E8CD040000             <1>       call   itrunc
18951                                  <1>              ; jsr r0,itrunc / free all blocks related to i-node
18952 0000534E 66C705[16710000]00-    <1>       mov    word [i.flgs], 0
18953 00005356 00                     <1>
18954                                  <1>              ; clr i.flgs / clear all flags in the i-node
18955 00005357 C3                     <1>       retn
18956                                  <1>              ;rts   r0 / return
18957                                  <1> anyi_3: ; 1: / i-numbers match
18958 00005358 FE4309                 <1>       inc    byte [ebx+9] ; 22/06/2015
18959                                  <1>              ;incb 7(r2) / increment upper byte of the 4th word
18960                                  <1>                 ; / in that fsp entry (deleted flag of fsp entry)
```

```
18961 0000535B C3                <1>       retn
18962                            <1>               ; rts r0
18963                                %include 'u3.s'        ; 10/05/2015
18964                            <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS3.INC
18965                            <1> ; Last Modification: 15/09/2015
18966                            <1> ; ---------------------------------------------------------------------------
18967                            <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
18968                            <1> ; (v0.1 - Beginning: 11/07/2012)
18969                            <1> ;
18970                            <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
18971                            <1> ; (Original) Source Code by Ken Thompson (1971-1972)
18972                            <1> ; <Bell Laboratories (17/3/1972)>
18973                            <1> ; <Preliminary Release of UNIX Implementation Document>
18974                            <1> ;
18975                            <1> ; Retro UNIX 8086 v1 - U3.ASM (08/03/2014) //// UNIX v1 -> u3.s
18976                            <1> ;
18977                            <1> ; ***************************************************************************
18978                            <1>
18979                            <1> tswitch: ; Retro UNIX 386 v1
18980                            <1> tswap:
18981                            <1>       ; 01/09/2015
18982                            <1>       ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
18983                            <1>       ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
18984                            <1>       ; time out swap, called when a user times out.
18985                            <1>       ; the user is put on the low priority queue.
18986                            <1>       ; This is done by making a link from the last user
18987                            <1>       ; on the low priority queue to him via a call to 'putlu'.
18988                            <1>       ; then he is swapped out.
18989                            <1>       ;
18990                            <1>       ; Retro UNIX 386 v1 modification ->
18991                            <1>       ;       swap (software task switch) is performed by changing
18992                            <1>       ;     user's page directory (u.pgdir) instead of segment change
18993                            <1>       ;     as in Retro UNIX 8086 v1.
18994                            <1>       ;
18995                            <1>       ; RETRO UNIX 8086 v1 modification ->
18996                            <1>       ;       'swap to disk' is replaced with 'change running segment'
18997                            <1>       ;     according to 8086 cpu (x86 real mode) architecture.
18998                            <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
18999                            <1>       ;     compatibles was using 1MB segmented memory
19000                            <1>       ;     in 8086/8088 times.
19001                            <1>       ;
19002                            <1>       ; INPUTS ->
19003                            <1>       ;    u.uno - users process number
19004                            <1>       ;    runq+4 - lowest priority queue
19005                            <1>       ; OUTPUTS ->
19006                            <1>       ;    r0 - users process number
19007                            <1>       ;    r2 - lowest priority queue address
19008                            <1>       ;
19009                            <1>       ; ((AX = R0, BX = R2)) output
19010                            <1>       ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
19011                            <1>       ;
19012 0000535C A0[97740000]      <1>       mov   al, [u.uno]
19013                            <1>                   ; movb u.uno,r1 / move users process number to r1
19014                            <1>             ; mov  $runq+4,r2
19015                            <1>                   ; / move lowest priority queue address to r2
19016 00005361 E8CE000000        <1>         call      putlu
19017                            <1>             ; jsr r0,putlu / create link from last user on Q to
19018                            <1>                           ; / u.uno's user
19019                            <1>
19020                            <1> switch: ; Retro UNIX 386 v1
19021                            <1> swap:
19022                            <1>       ; 02/09/2015
19023                            <1>       ; 01/09/2015
19024                            <1>       ; 31/08/2015
19025                            <1>       ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19026                            <1>       ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19027                            <1>       ; 'swap' is routine that controls the swapping of processes
19028                            <1>       ; in and out of core.
19029                            <1>       ;
19030                            <1>       ; Retro UNIX 386 v1 modification ->
19031                            <1>       ;       swap (software task switch) is performed by changing
19032                            <1>       ;     user's page directory (u.pgdir) instead of segment change
19033                            <1>       ;     as in Retro UNIX 8086 v1.
19034                            <1>       ;
19035                            <1>       ; RETRO UNIX 8086 v1 modification ->
19036                            <1>       ;       'swap to disk' is replaced with 'change running segment'
19037                            <1>       ;     according to 8086 cpu (x86 real mode) architecture.
19038                            <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
19039                            <1>       ;     compatibles was using 1MB segmented memory
19040                            <1>       ;     in 8086/8088 times.
19041                            <1>       ;
19042                            <1>       ; INPUTS ->
19043                            <1>       ;    runq table - contains processes to run.
19044                            <1>       ;    p.link - contains next process in line to be run.
19045                            <1>       ;    u.uno - process number of process in core
19046                            <1>       ;    s.stack - swap stack used as an internal stack for swapping.
19047                            <1>       ; OUTPUTS ->
19048                            <1>       ;    (original unix v1 -> present process to its disk block)
19049                            <1>       ;    (original unix v1 -> new process into core ->
19050                            <1>       ;        Retro Unix 8086 v1 -> segment registers changed
19051                            <1>       ;        for new process)
19052                            <1>       ;    u.quant = 3 (Time quantum for a process)
19053                            <1>       ;     ((INT 1Ch count down speed -> 18.2 times per second)
19054                            <1>       ;    RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
19055                            <1>       ;      for now, it will swap the process if there is not
19056                            <1>       ;      a keyboard event (keystroke) (Int 15h, function 4Fh)
19057                            <1>       ;      or will count down from 3 to 0 even if there is a
19058                            <1>       ;        keyboard event locking due to repetitive key strokes.
19059                            <1>       ;      u.quant will be reset to 3 for RETRO UNIX 8086 v1.
19060                            <1>       ;
19061                            <1>       ;    u.pri -points to highest priority run Q.
19062                            <1>       ;    r2 - points to the run queue.
19063                            <1>       ;    r1 - contains new process number
19064                            <1>       ;    r0 - points to place in routine or process that called
19065                            <1>       ;       swap all user parameters
```

```
19066                                  <1>       ;
19067                                  <1>       ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
19068                                  <1>       ;
19069                                  <1> swap_0:
19070                                  <1>             ;mov $300,*$ps / processor priority = 6
19071 00005366 BE[3A740000]            <1>       mov   esi, runq
19072                                  <1>             ; mov $runq,r2 / r2 points to runq table
19073                                  <1> swap_1: ; 1: / search runq table for highest priority process
19074 0000536B 668B06                  <1>       mov   ax, [esi]
19075 0000536E 6621C0                  <1>       and   ax, ax
19076                                  <1>                   ; tst (r2)+ / are there any processes to run
19077                                  <1>                     ; / in this Q entry
19078 00005371 7507                    <1>       jnz   short swap_2
19079                                  <1>                   ; bne 1f / yes, process 1f
19080                                  <1>             ; cmp r2,$runq+6 / if zero compare address
19081                                  <1>                              ; / to end of table
19082                                  <1>             ; bne 1b / if not at end, go back
19083 00005373 E8E1000000              <1>       call  idle
19084                                  <1>             ; jsr r0,idle; s.idlet+2 / wait for interrupt;
19085                                  <1>                                    ; / all queues are empty
19086 00005378 EBF1                    <1>       jmp   short swap_1
19087                                  <1>             ; br swap
19088                                  <1> swap_2: ; 1:
19089 0000537A 0FB6D8                  <1>       movzx ebx, al ; 02/09/2015
19090                                  <1>             ; tst -(r2) / restore pointer to right Q entry
19091                                  <1>             ; mov r2,u.pri / set present user to this run queue
19092                                  <1>              ; movb (r2)+,r1 / move 1st process in queue to r1
19093 0000537D 38E0                    <1>       cmp   al, ah
19094                                  <1>             ; cmpb r1,(r2)+ / is there only 1 process
19095                                  <1>                             ; / in this Q to be run
19096 0000537F 740A                    <1>       je    short swap_3
19097                                  <1>                   ; beq 1f / yes
19098                                  <1>             ; tst -(r2) / no, pt r2 back to this Q entry
19099                                  <1>       ;movzx ebx, al
19100 00005381 8AA3[B5710000]          <1>       mov   ah, [ebx+p.link-1]
19101 00005387 8826                    <1>             mov   [esi], ah
19102                                  <1>             ; movb p.link-1(r1),(r2) / move next process
19103                                  <1>                                    ; / in line into run queue
19104 00005389 EB06                    <1>       jmp   short swap_4
19105                                  <1>                   ; br 2f
19106                                  <1> swap_3: ; 1:
19107 0000538B 6631D2                  <1>       xor   dx, dx
19108 0000538E 668916                  <1>       mov   [esi], dx
19109                                  <1>             ; clr -(r2) / zero the entry; no processes on the Q
19110                                  <1> swap_4: ; / write out core to appropriate disk area and read
19111                                  <1>       ; / in new process if required
19112                                  <1>                   ; clr *$ps / clear processor status
19113 00005391 8A25[97740000]          <1>       mov   ah, [u.uno]
19114 00005397 38C4                    <1>       cmp   ah, al
19115                                  <1>             ; cmpb r1,u.uno / is this process the same as
19116                                  <1>                             ; / the process in core?
19117 00005399 743B                    <1>             je    short swap_8
19118                                  <1>                   ; beq 2f / yes, don't have to swap
19119                                  <1>                   ; mov r0,-(sp) / no, write out core; save r0
19120                                  <1>                      ; / (address in routine that called swap)
19121                                  <1>             ; mov r1,-(sp) / put r1 (new process #) on the stack
19122                                  <1>       ; 01/09/2015
19123                                  <1>       ;mov   [u.usp], esp
19124                                  <1>                   ; mov sp,u.usp / save stack pointer
19125                                  <1>             ; mov $sstack,sp / move swap stack pointer
19126                                  <1>                              ; / to the stack pointer
19127 0000539B 08E4                    <1>       or    ah, ah
19128                                  <1>                   ; tstb u.uno / is the process # = 0
19129 0000539D 740D                    <1>             jz    short swap_6 ; 'sysexit'
19130                                  <1>             ; beq  1f / yes, kill process by overwriting
19131                                  <1>       ; 02/09/2015
19132 0000539F 8925[44740000]          <1>       mov   [u.usp], esp ; return  address for 'syswait' & 'sleep'
19133                                  <1>       ;
19134 000053A5 E834000000              <1>       call  wswap
19135                                  <1>             ;jsr r0,wswap / write out core to disk
19136                                  <1>        ; 31/08/2015
19137                                  <1>       ;movzx ebx, al ; New (running) process number
19138 000053AA EB1C                    <1>       jmp   short swap_7
19139                                  <1> swap_6:
19140                                  <1>       ; 31/08/2015
19141                                  <1>       ; Deallocate memory pages belong to the process
19142                                  <1>       ; which is being terminated
19143                                  <1>       ; 14/05/2015 ('sysexit')
19144                                  <1>       ; Deallocate memory pages of the process
19145                                  <1>       ; (Retro UNIX 386 v1 modification !)
19146                                  <1>       ;
19147                                  <1>       ; movzx ebx, al
19148 000053AC 53                      <1>       push  ebx
19149 000053AD A1[A1740000]            <1>       mov   eax, [u.pgdir]  ; page directory of the process
19150 000053B2 8B1D[A5740000]          <1>       mov   ebx, [u.ppgdir] ; page directory of the parent process
19151 000053B8 E871DDFFFF              <1>       call  deallocate_page_dir
19152 000053BD A1[98740000]            <1>       mov   eax, [u.upage] ; 'user' structure page of the process
19153 000053C2 E806DEFFFF              <1>       call  deallocate_page
19154 000053C7 5B                      <1>       pop   ebx
19155                                  <1> swap_7: ;1:
19156                                  <1>       ; 02/09/2015
19157                                  <1>       ; 31/08/2015
19158                                  <1>       ; 14/05/2015
19159 000053C8 C0E302                  <1>       shl   bl, 2 ; * 4
19160 000053CB 8B83[D2710000]          <1>       mov   eax, [ebx+p.upage-4] ; the 'u' page of the new process
19161                                  <1>       ;cli
19162 000053D1 E831000000              <1>       call  rswap
19163                                  <1>             ; mov (sp)+,r1 / restore r1 to new process number
19164                                  <1>             ; jsr r0,rswap / read new process into core
19165                                  <1>                    ; jsr r0,unpack / unpack the users stack from next
19166                                  <1>                          ; / to his program to its normal
19167                                  <1>       ; 01/09/2015
19168                                  <1>       ;mov   esp, [u.usp]
19169                                  <1>             ; mov u.usp,sp / location; restore stack pointer to
19170                                  <1>                           ; / new process stack
```

```
19171                                  <1>             ; mov (sp)+,r0 / put address of where the process
19172                                  <1>                       ; / that just got swapped in, left off.,
19173                                  <1>                       ; / i.e., transfer control to new process
19174                                  <1>       ;sti
19175                                  <1> swap_8: ;2:
19176                                  <1>       ; RETRO UNIX 8086 v1 modification !
19177 000053D6 C605[8A740000]04       <1>       mov    byte [u.quant], time_count
19178                                  <1>             ; movb    $30.,uquant / initialize process time quantum
19179 000053DD C3                     <1>       retn
19180                                  <1>             ; rts r0 / return
19181                                  <1>
19182                                  <1> wswap:  ; < swap out, swap to disk >
19183                                  <1>       ; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
19184                                  <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19185                                  <1>       ; 'wswap' writes out the process that is in core onto its
19186                                  <1>       ; appropriate disk area.
19187                                  <1>       ;
19188                                  <1>       ; Retro UNIX 386 v1 modification ->
19189                                  <1>       ;       User (u) structure content and the user's register content
19190                                  <1>       ;     will be copied to the process's/user's UPAGE (a page for
19191                                  <1>       ;     saving 'u' structure and user registers for task switching).
19192                                  <1>       ;     u.usp - points to kernel stack address which contains
19193                                  <1>       ;             user's registers while entering system call.
19194                                  <1>       ;     u.sp  - points to kernel stack address
19195                                  <1>       ;             to return from system call -for IRET-.
19196                                  <1>       ;     [u.usp]+32+16 = [u.sp]
19197                                  <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
19198                                  <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
19199                                  <1>       ;
19200                                  <1>       ; Retro UNIX 8086 v1 modification ->
19201                                  <1>       ;       'swap to disk' is replaced with 'change running segment'
19202                                  <1>       ;     according to 8086 cpu (x86 real mode) architecture.
19203                                  <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
19204                                  <1>       ;     compatibles was using 1MB segmented memory
19205                                  <1>       ;     in 8086/8088 times.
19206                                  <1>       ;
19207                                  <1>       ; INPUTS ->
19208                                  <1>       ;    u.break - points to end of program
19209                                  <1>       ;    u.usp - stack pointer at the moment of swap
19210                                  <1>       ;    core - beginning of process program
19211                                  <1>       ;    ecore - end of core
19212                                  <1>       ;    user - start of user parameter area
19213                                  <1>       ;    u.uno - user process number
19214                                  <1>       ;    p.dska - holds block number of process
19215                                  <1>       ; OUTPUTS ->
19216                                  <1>       ;    swp I/O queue
19217                                  <1>       ;    p.break - negative word count of process
19218                                  <1>       ;    r1 - process disk address
19219                                  <1>       ;    r2 - negative word count
19220                                  <1>       ;
19221                                  <1>       ; RETRO UNIX 8086 v1 input/output:
19222                                  <1>       ;
19223                                  <1>       ; INPUTS ->
19224                                  <1>       ;    u.uno - process number (to be swapped out)
19225                                  <1>       ; OUTPUTS ->
19226                                  <1>       ;    none
19227                                  <1>       ;
19228                                  <1>       ;   ((Modified registers: ECX, ESI, EDI))
19229                                  <1>       ;
19230 000053DE 8B3D[98740000]         <1>       mov    edi, [u.upage] ; process's user (u) structure page addr
19231 000053E4 B91E000000             <1>       mov    ecx, (U_SIZE + 3) / 4
19232 000053E9 BE[40740000]           <1>       mov    esi, user ; active user (u) structure
19233 000053EE F3A5                   <1>       rep    movsd
19234                                  <1>       ;
19235 000053F0 8B35[44740000]         <1>       mov    esi, [u.usp] ; esp (system stack pointer,
19236                                  <1>                           ;      points to user registers)
19237 000053F6 8B0D[40740000]         <1>       mov    ecx, [u.sp]  ; return address from the system call
19238                                  <1>                           ; (for IRET)
19239                                  <1>                           ; [u.sp] -> EIP (user)
19240                                  <1>                           ; [u.sp+4]-> CS (user)
19241                                  <1>                           ; [u.sp+8] -> EFLAGS (user)
19242                                  <1>                           ; [u.sp+12] -> ESP (user)
19243                                  <1>                           ; [u.sp+16] -> SS (user)
19244 000053FC 29F1                   <1>       sub    ecx, esi     ; required space for user registers
19245 000053FE 83C114                 <1>       add    ecx, 20           ; +5 dwords to return from system call
19246                                  <1>                           ; (for IRET)
19247 00005401 C1E902                 <1>       shr    ecx, 2
19248 00005404 F3A5                   <1>       rep    movsd
19249 00005406 C3                     <1>       retn
19250                                  <1>
19251                                  <1>       ; Original UNIX v1 'wswap' routine:
19252                                  <1>       ; wswap:
19253                                  <1>             ; mov *$30,u.emt / determines handling of emts
19254                                  <1>             ; mov *$10,u.ilgins / determines handling of
19255                                  <1>                       ; / illegal instructions
19256                                  <1>             ; mov u.break,r2 / put process program break address in r2
19257                                  <1>             ; inc r2 / add 1 to it
19258                                  <1>             ; bic $1,r2 / make it even
19259                                  <1>             ; mov r2,u.break / set break to an even location
19260                                  <1>             ; mov u.usp,r3 / put users stack pointer
19261                                  <1>                       ; / at moment of swap in r3
19262                                  <1>             ; cmp r2,$core / is u.break less than $core
19263                                  <1>             ; blos 2f / yes
19264                                  <1>             ; cmp r2,r3 / no, is (u.break) greater than stack ptr.
19265                                  <1>                   ; bhis 2f / yes
19266                                  <1>       ; 1:
19267                                  <1>                   ; mov (r3)+,(r2)+ / no, pack stack next to users program
19268                                  <1>             ; cmp r3,$ecore / has stack reached end of core
19269                                  <1>             ; bne 1b / no, keep packing
19270                                  <1>             ; br 1f / yes
19271                                  <1>       ; 2:
19272                                  <1>                   ; mov $ecore,r2 / put end of core in r2
19273                                  <1>       ; 1:
19274                                  <1>                   ; sub  $user,r2 / get number of bytes to write out
19275                                  <1>                      ; / (user up to end of stack gets written out)
```

```
19276                                 <1>             ; neg r2 / make it negative
19277                                 <1>             ; asr r2 / change bytes to words (divide by 2)
19278                                 <1>             ; mov r2,swp+4 / word count
19279                                 <1>             ; movb u.uno,r1 / move user process number to r1
19280                                 <1>             ; asl r1 / x2 for index
19281                                 <1>                   ; mov r2,p.break-2(r1) / put negative of word count
19282                                 <1>                               ; / into the p.break table
19283                                 <1>                   ; mov p.dska-2(r1),r1 / move disk address of swap area
19284                                 <1>                               ; /      for process to r1
19285                                 <1>                   ; mov r1,swp+2 / put processes dska address in swp+2
19286                                 <1>                       ; / (block number)
19287                                 <1>             ; bis $1000,swp / set it up to write (set bit 9)
19288                                 <1>                   ; jsr r0,ppoke / write process out on swap area of disk
19289                                 <1>       ; 1:
19290                                 <1>                   ; tstb swp+1 / is lt done writing?
19291                                 <1>                   ; bne 1b / no, wait
19292                                 <1>             ; rts r0 / yes, return to swap
19293                                 <1>
19294                                 <1> rswap:  ; < swap in, swap from disk >
19295                                 <1>       ; 15/09/2015
19296                                 <1>       ; 28/08/2015
19297                                 <1>       ; 14/05/2015
19298                                 <1>       ; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
19299                                 <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19300                                 <1>       ; 'rswap' reads a process whose number is in r1,
19301                                 <1>       ; from disk into core.
19302                                 <1>       ;
19303                                 <1>       ; Retro UNIX 386 v1 modification ->
19304                                 <1>       ;      User (u) structure content and the user's register content
19305                                 <1>       ;     will be restored from process's/user's UPAGE (a page for
19306                                 <1>       ;     saving 'u' structure and user registers for task switching).
19307                                 <1>       ;     u.usp - points to kernel stack address which contains
19308                                 <1>       ;             user's registers while entering system call.
19309                                 <1>       ;     u.sp  - points to kernel stack address
19310                                 <1>       ;             to return from system call -for IRET-.
19311                                 <1>       ;     [u.usp]+32+16 = [u.sp]
19312                                 <1>       ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
19313                                 <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
19314                                 <1>       ;
19315                                 <1>       ; RETRO UNIX 8086 v1 modification ->
19316                                 <1>       ;      'swap to disk' is replaced with 'change running segment'
19317                                 <1>       ;     according to 8086 cpu (x86 real mode) architecture.
19318                                 <1>       ;     pdp-11 was using 64KB uniform memory while IBM PC
19319                                 <1>       ;     compatibles was using 1MB segmented memory
19320                                 <1>       ;     in 8086/8088 times.
19321                                 <1>       ;
19322                                 <1>       ; INPUTS ->
19323                                 <1>       ;    r1 - process number of process to be read in
19324                                 <1>       ;    p.break - negative of word count of process
19325                                 <1>       ;    p.dska - disk address of the process
19326                                 <1>       ;    u.emt - determines handling of emt's
19327                                 <1>       ;    u.ilgins - determines handling of illegal instructions
19328                                 <1>       ; OUTPUTS ->
19329                                 <1>       ;    8 = (u.ilgins)
19330                                 <1>       ;    24 = (u.emt)
19331                                 <1>       ;    swp - bit 10 is set to indicate read
19332                                 <1>       ;             (bit 15=0 when reading is done)
19333                                 <1>       ;    swp+2 - disk block address
19334                                 <1>       ;    swp+4 - negative word count
19335                                 <1>       ;      ((swp+6 - address of user structure))
19336                                 <1>       ;
19337                                 <1>       ; RETRO UNIX 8086 v1 input/output:
19338                                 <1>       ;
19339                                 <1>       ; INPUTS ->
19340                                 <1>       ;    AL      - new process number (to be swapped in)
19341                                 <1>       ; OUTPUTS ->
19342                                 <1>       ;    none
19343                                 <1>       ;
19344                                 <1>       ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
19345                                 <1>       ;
19346                                 <1>       ; Retro UNIX 386 v1 - modification ! 14/05/2015
19347 00005407 89C6                   <1>       mov    esi, eax  ; process's user (u) structure page addr
19348 00005409 B91E000000             <1>       mov    ecx, (U_SIZE + 3) / 4
19349 0000540E BF[40740000]           <1>       mov    edi, user ; active user (u) structure
19350 00005413 F3A5                   <1>       rep    movsd
19351 00005415 58                     <1>       pop    eax ; 15/09/2015, 'rswap' return address
19352 00005416 8B3D[44740000]         <1>       mov    edi, [u.usp] ; esp (system stack pointer,
19353                                 <1>                           ;      points to user registers)
19354 0000541C 8B0D[40740000]         <1>       mov    ecx, [u.sp]  ; return address from the system call
19355                                 <1>                           ; (for IRET)
19356                                 <1>                           ; [u.sp] -> EIP (user)
19357                                 <1>                           ; [u.sp+4]-> CS (user)
19358                                 <1>                           ; [u.sp+8] -> EFLAGS (user)
19359                                 <1>                           ; [u.sp+12] -> ESP (user)
19360                                 <1>                           ; [u.sp+16] -> SS (user)
19361                                 <1>       ; 28/08/2015
19362 00005422 29F9                   <1>       sub    ecx, edi     ; required space for user registers
19363 00005424 83C114                 <1>       add    ecx, 20           ; +5 dwords to return from system call
19364                                 <1>                           ; (for IRET)
19365 00005427 C1E902                 <1>       shr    ecx, 2
19366 0000542A F3A5                   <1>       rep    movsd
19367 0000542C 8B25[44740000]         <1>       mov    esp, [u.usp] ; 15/09/2015
19368 00005432 50                     <1>       push   eax ; 15/09/2015 'rswap' return address
19369 00005433 C3                     <1>       retn
19370                                 <1>
19371                                 <1>       ; Original UNIX v1 'rswap'  and 'unpack' routines:
19372                                 <1>       ;rswap:
19373                                 <1>                     ; asl r1 / process number x2 for index
19374                                 <1>                     ; mov p.break-2(r1), swp+4 / word count
19375                                 <1>                     ; mov p.dska-2(r1),swp+2 / disk address
19376                                 <1>                     ; bis $2000,swp / read
19377                                 <1>                     ; jsr r0,ppoke / read it in
19378                                 <1>       ; 1:
19379                                 <1>                     ; tstb swp+1 / done
19380                                 <1>                     ; bne 1b / no, wait for bit 15 to clear (inhibit bit)
```

```
19381                                  <1>                     ; mov u.emt,*$30 / yes move these
19382                                  <1>                     ; mov u.ilgins,*$10 / back
19383                                  <1>                     ; rts r0 / return
19384                                  <1>
19385                                  <1>       ;unpack: ; / move stack back to its normal place
19386                                  <1>             ; mov u.break,r2 / r2 points to end of user program
19387                                  <1>                   ; cmp r2,$core / at beginning of user program yet?
19388                                  <1>             ; blos 2f / yes, return
19389                                  <1>             ; cmp r2,u.usp / is break_above the stack pointer
19390                                  <1>                           ; / before swapping
19391                                  <1>             ; bhis 2f / yes, return
19392                                  <1>             ; mov $ecore,r3 / r3 points to end of core
19393                                  <1>             ; add r3,r2
19394                                  <1>             ; sub u.usp,r2 / end of users stack is in r2
19395                                  <1>       ; 1:
19396                                  <1>             ; mov -(r2),-(r3) / move stack back to its normal place
19397                                  <1>             ; cmp r2,u.break / in core
19398                                  <1>             ; bne 1b
19399                                  <1>       ; 2:
19400                                  <1>                     ; rts r0
19401                                  <1>
19402                                  <1> putlu:
19403                                  <1>       ; 12/09/2015
19404                                  <1>       ; 02/09/2015
19405                                  <1>       ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19406                                  <1>       ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
19407                                  <1>       ; 'putlu' is called with a process number in r1 and a pointer
19408                                  <1>       ; to lowest priority Q (runq+4) in r2. A link is created from
19409                                  <1>       ; the last process on the queue to process in r1 by putting
19410                                  <1>       ; the process number in r1 into the last process's link.
19411                                  <1>       ;
19412                                  <1>       ; INPUTS ->
19413                                  <1>       ;    r1 - user process number
19414                                  <1>       ;    r2 - points to lowest priority queue
19415                                  <1>       ;    p.dska - disk address of the process
19416                                  <1>       ;    u.emt - determines handling of emt's
19417                                  <1>       ;    u.ilgins - determines handling of illegal instructions
19418                                  <1>       ; OUTPUTS ->
19419                                  <1>       ;    r3 - process number of last process on the queue upon
19420                                  <1>       ;       entering putlu
19421                                  <1>       ;    p.link-1 + r3 - process number in r1
19422                                  <1>       ;    r2 - points to lowest priority queue
19423                                  <1>       ;
19424                                  <1>       ; ((Modified registers: EDX, EBX))
19425                                  <1>       ;
19426                                  <1>       ; / r1 = user process no.; r2 points to lowest priority queue
19427                                  <1>
19428                                  <1>       ; eBX = r2
19429                                  <1>       ; eAX = r1 (AL=r1b)
19430                                  <1>
19431 00005434 BB[3A740000]           <1>       mov    ebx, runq
19432 00005439 0FB613                <1>       movzx         edx, byte [ebx]
19433 0000543C 43                    <1>       inc    ebx
19434 0000543D 20D2                  <1>       and    dl, dl
19435                                <1>             ; tstb (r2)+ / is queue empty?
19436 0000543F 740A                  <1>             jz     short putlu_1
19437                                <1>             ; beq 1f / yes, branch
19438 00005441 8A13                  <1>       mov    dl, [ebx] ; 12/09/2015
19439                                <1>             ; movb (r2),r3 / no, save the "last user" process number
19440                                <1>                         ; / in r3
19441 00005443 8882[B5710000]        <1>             mov    [edx+p.link-1], al
19442                                <1>             ; movb r1,p.link-1(r3) / put pointer to user on
19443                                <1>                         ; / "last users" link
19444 00005449 EB03                  <1>       jmp    short putlu_2
19445                                <1>             ; br 2f /
19446                                <1> putlu_1: ; 1:
19447 0000544B 8843FF                <1>       mov    [ebx-1], al
19448                                <1>                     ; movb r1,-1(r2) / user is only user;
19449                                <1>                         ; / put process no. at beginning and at end
19450                                <1> putlu_2: ; 2:
19451 0000544E 8803                  <1>       mov    [ebx], al
19452                                <1>                     ; movb r1,(r2) / user process in r1 is now the last entry
19453                                <1>                         ; / on the queue
19454 00005450 88C2                  <1>       mov    dl, al
19455 00005452 88B2[B5710000]        <1>         mov     [edx+p.link-1], dh ; 0
19456                                <1>             ; dec r2 / restore r2
19457 00005458 C3                    <1>         retn
19458                                <1>             ; rts r0
19459                                <1>
19460                                <1> ;copyz:
19461                                <1> ;       mov     r1,-(sp) / put r1 on stack
19462                                <1> ;       mov     r2,-(sp) / put r2 on stack
19463                                <1> ;       mov     (r0)+,r1
19464                                <1> ;       mov     (r0)+,r2
19465                                <1> ;1:
19466                                <1> ;       clr     (r1)+ / clear all locations between r1 and r2
19467                                <1> ;       cmp     r1,r2
19468                                <1> ;       blo     1b
19469                                <1> ;       mov     (sp)+,r2 / restore r2
19470                                <1> ;       mov     (sp)+,r1 / restore r1
19471                                <1> ;       rts     r0
19472                                <1>
19473                                <1> idle:
19474                                <1>       ; 01/09/2015
19475                                <1>       ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19476                                <1>       ; 10/04/2013 - 23/10/2013 (Retro UNIX 8086 v1)
19477                                <1>       ; (idle & wait loop)
19478                                <1>       ; Retro Unix 8086 v1 modification on original UNIX v1
19479                                <1>       ; idle procedure!
19480                                <1>             ;
19481                                <1>       ; 01/09/2015
19482 00005459 FB                    <1>       sti
19483                                <1>             ; 29/07/2013
19484 0000545A F4                    <1>             hlt
19485 0000545B 90                    <1>             nop ; 10/10/2013
```

```
19486 0000545C 90                <1>             nop
19487 0000545D 90                <1>             nop
19488                            <1>             ; 23/10/2013
19489 0000545E 90                <1>             nop
19490 0000545F 90                <1>             nop
19491 00005460 90                <1>             nop
19492 00005461 90                <1>             nop
19493 00005462 C3                <1>             retn
19494                            <1>
19495                            <1>       ;mov *$ps,-(sp) / save ps on stack
19496                            <1>       ;clr *$ps / clear ps
19497                            <1>       ;mov clockp,-(sp) / save clockp on stack
19498                            <1>       ;mov (r0)+,clockp / arg to idle in clockp
19499                            <1>       ;1 / wait for interrupt
19500                            <1>       ;mov (sp)+,clockp / restore clockp, ps
19501                            <1>       ;mov (sp)+,*$ps
19502                            <1>       ;rts r0
19503                            <1>
19504                            <1> clear:
19505                            <1>       ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19506                            <1>       ; 09/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
19507                            <1>       ; 'clear' zero's out of a block (whose block number is in r1)
19508                            <1>       ; on the current device (cdev)
19509                            <1>       ;
19510                            <1>       ; INPUTS ->
19511                            <1>       ;    r1 - block number of block to be zeroed
19512                            <1>       ;    cdev - current device number
19513                            <1>       ; OUTPUTS ->
19514                            <1>       ;    a zeroed I/O buffer onto the current device
19515                            <1>       ;    r1 - points to last entry in the I/O buffer
19516                            <1>       ;
19517                            <1>       ; ((AX = R1)) input/output
19518                            <1>       ;    (Retro UNIX Prototype : 18/11/2012 - 14/11/2012, UNIXCOPY.ASM)
19519                            <1>         ;    ((Modified registers: EDX, ECX, EBX, ESI, EDI, EBP))
19520                            <1>
19521 00005463 E8AB0D0000        <1>       call  wslot
19522                            <1>             ; jsr r0,wslot / get an I/O buffer set bits 9 and 15 in first
19523                            <1>                   ; / word of I/O queue r5 points to first data word in buffer
19524 00005468 89DF              <1>       mov   edi, ebx ; r5
19525 0000546A 89C2              <1>       mov   edx, eax
19526 0000546C B980000000        <1>       mov   ecx, 128
19527                            <1>             ; mov $256.,r3
19528 00005471 31C0              <1>       xor   eax, eax
19529 00005473 F3AB              <1>       rep   stosd
19530 00005475 89D0              <1>       mov   eax, edx
19531                            <1> ; 1:
19532                            <1>                   ; clr (r5)+ / zero data word in buffer
19533                            <1>                   ; dec r3
19534                            <1>                   ; bgt 1b / branch until all data words in buffer are zero
19535 00005477 E8B30D0000        <1>       call  dskwr
19536                            <1>             ; jsr r0,dskwr / write zeroed buffer area out onto physical
19537                            <1>                                ; / block specified in r1
19538                            <1>       ; eAX (r1) = block number
19539 0000547C C3                <1>       retn
19540                            <1>             ; rts r0
19541                                %include 'u4.s'        ; 15/04/2015
19542                            <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS4.INC
19543                            <1> ; Last Modification: 14/10/2015
19544                            <1> ; ----------------------------------------------------------------------------
19545                            <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
19546                            <1> ; (v0.1 - Beginning: 11/07/2012)
19547                            <1> ;
19548                            <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
19549                            <1> ; (Original) Source Code by Ken Thompson (1971-1972)
19550                            <1> ; <Bell Laboratories (17/3/1972)>
19551                            <1> ; <Preliminary Release of UNIX Implementation Document>
19552                            <1> ;
19553                            <1> ; Retro UNIX 8086 v1 - U4.ASM (04/07/2014) //// UNIX v1 -> u4.s
19554                            <1> ;
19555                            <1> ; ****************************************************************************
19556                            <1>
19557                            <1> ;setisp:
19558                            <1>        ;mov     r1,-(sp)
19559                            <1>        ;mov     r2,-(sp)
19560                            <1>        ;mov     r3,-(sp)
19561                            <1>        ;mov     clockp,-(sp)
19562                            <1>        ;mov     $s.syst+2,clockp
19563                            <1>        ;jmp     (r0)
19564                            <1>
19565                            <1> clock: ; / interrupt from 60 cycle clock
19566                            <1>
19567                            <1>       ; 14/10/2015
19568                            <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
19569                            <1>       ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
19570                            <1>
19571                            <1>        ;mov     r0,-(sp) / save r0
19572                            <1>        ;tst     *$lks / restart clock?
19573                            <1>        ;mov     $s.time+2,r0 / increment the time of day
19574                            <1>        ;inc     (r0)
19575                            <1>        ;bne     1f
19576                            <1>        ;inc     -(r0)
19577                            <1> ;1:
19578                            <1>        ;mov     clockp,r0 / increment appropriate time category
19579                            <1>        ;inc     (r0)
19580                            <1>        ;bne     1f
19581                            <1>        ;inc     -(r0)
19582                            <1> ;1:
19583                            <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
19584                            <1>
19585 0000547D 803D[8A740000]00  <1>       cmp    byte [u.quant], 0
19586 00005484 772C              <1>       ja     short clk_1
19587                            <1>       ;
19588 00005486 803D[3F740000]FF  <1>         cmp     byte [sysflg], 0FFh ; user or system space ?
19589 0000548D 7529              <1>       jne    short clk_2 ; system space (sysflg <> 0FFh)
19590 0000548F 803D[97740000]01  <1>       cmp     byte [u.uno], 1 ; /etc/init ?
```

```
19591 00005496 761A                <1>       jna    short clk_1 ; yes, do not swap out
19592 00005498 66833D[8C740000]00  <1>       cmp    word [u.intr], 0
19593 000054A0 7616                <1>       jna    short clk_2
19594                              <1> clk_0:
19595                              <1>       ; 14/10/2015
19596 000054A2 FE05[3F740000]      <1>       inc    byte [sysflg]       ; Now, we are in system space
19597 000054A8 58                  <1>       pop    eax ; return address to the timer interrupt
19598                              <1>       ;
19599 000054A9 B020                <1>       MOV    AL,EOI              ; GET END OF INTERRUPT MASK
19600                              <1>       ;CLI                       ; DISABLE INTERRUPTS TILL STACK CLEARED
19601 000054AB E620                <1>       OUT    INTA00,AL           ; END OF INTERRUPT TO 8259 - 1
19602                              <1>       ;
19603 000054AD E904ECFFFF          <1>       jmp     sysrelease ; 'sys release' by clock/timer
19604                              <1> clk_1:
19605 000054B2 FE0D[8A740000]      <1>       dec    byte [u.quant]
19606                              <1> clk_2:
19607 000054B8 C3                  <1>       retn   ; return to (hardware) timer interrupt routine
19608                              <1>
19609                              <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
19610                              <1>
19611                              <1>        ;mov     $uquant,r0 / decrement user time quantum
19612                              <1>        ;decb    (r0)
19613                              <1>        ;bge     1f / if less than 0
19614                              <1>        ;clrb    (r0) / make it 0
19615                              <1> ;1: / decrement time out counts return now if priority was not 0
19616                              <1>        ;cmp     4(sp),$200 / ps greater than or equal to 200
19617                              <1>        ;bge     2f / yes, check time outs
19618                              <1>        ;tstb    (r0) / no, user timed out?
19619                              <1>        ;bne     1f / no
19620                              <1>        ;cmpb    sysflg,$-1 / yes, are we outside the system?
19621                              <1>        ;bne     1f / no, 1f
19622                              <1>        ;mov     (sp)+,r0 / yes, put users r0 in r0
19623                              <1>        ;sys     0 / sysrele
19624                              <1>        ;rti
19625                              <1> ;2: / priority is high so just decrement time out counts
19626                              <1>        ;mov     $toutt,r0 / r0 points to beginning of time out table
19627                              <1> ;2:
19628                              <1>        ;tstb    (r0) / is the time out?
19629                              <1>        ;beq     3f / yes, 3f (get next entry)
19630                              <1>        ;decb    (r0) / no, decrement the time
19631                              <1>        ;bne     3f / isit zero now?
19632                              <1>        ;incb    (r0) / yes, increment the time
19633                              <1> ;3:
19634                              <1>        ;inc     r0 / next entry
19635                              <1>        ;cmp     r0,$touts / end of toutt table?
19636                              <1>        ;blo     2b / no, check this entry
19637                              <1>        ;mov     (sp)+,r0 / yes, restore r0
19638                              <1>        ;rti / return from interrupt
19639                              <1> ;1: / decrement time out counts; if 0 call subroutine
19640                              <1>        ;mov     (sp)+,r0 / restore r0
19641                              <1>        ;mov     $240,*$ps / set processor priority to 5
19642                              <1>        ;jsr     r0,setisp / save registers
19643                              <1>        ;mov     $touts-toutt-1,r0 / set up r0 as index to decrement thru
19644                              <1>                                 ;  / the table
19645                              <1> ;1:
19646                              <1>        ;tstb    toutt(r0) / is the time out for this entry
19647                              <1>        ;beq     2f / yes
19648                              <1>        ;decb    toutt(r0) / no, decrement the time
19649                              <1>        ;bne     2f / is the time 0, now
19650                              <1>        ;asl     r0 / yes, 2 x r0 to get word index for tout entry
19651                              <1>        ;jsr     r0,*touts(r0) / go to appropriate routine specified in this
19652                              <1>        ;asr     r0 / touts entry; set r0 back to toutt index
19653                              <1> ;2:
19654                              <1>        ;dec     r0 / set up r0 for next entry
19655                              <1>        ;bge     1b / finished? , no, go back
19656                              <1>        ;br      retisp / yes, restore registers and do a rti
19657                              <1>
19658                              <1> ;retisp:
19659                              <1>        ;mov     (sp)+,clockp / pop values before interrupt off the stack
19660                              <1>        ;mov     (sp)+,r3
19661                              <1>        ;mov     (sp)+,r2
19662                              <1>        ;mov     (sp)+,r1
19663                              <1>        ;mov     (sp)+,r0
19664                              <1>        ;rti     / return from interrupt
19665                              <1>
19666                              <1>
19667                              <1> wakeup: ; / wakeup processes waiting for an event
19668                              <1>       ; / by linking them to the queue
19669                              <1>       ;
19670                              <1>       ; 15/09/2015
19671                              <1>       ; 29/06/2015
19672                              <1>       ; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
19673                              <1>       ;
19674                              <1>       ; 15/05/2013 - 02/06/2014
19675                              <1>       ; Retro UNIX 8086 v1 modification !
19676                              <1>       ; (Process/task switching routine by using
19677                              <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.)
19678                              <1>       ;
19679                              <1>       ; In original UNIX v1, 'wakeup' is called to wake the process
19680                              <1>       ; sleeping in the specified wait channel by creating a link
19681                              <1>       ; to it from the last user process on the run queue.
19682                              <1>       ; If there is no process to wake up, nothing happens.
19683                              <1>       ;
19684                              <1>       ; In Retro UNIX 8086 v1, Int 09h keyboard interrupt will set
19685                              <1>       ; 'switching' status of the current process (owns current tty)
19686                              <1>       ; (via alt + function keys) to a process which has highest
19687                              <1>       ; priority (on run queue) on the requested tty (0 to 7, except
19688                              <1>       ; 8 and 9 which are tty identifiers of COM1, COM2 serial ports)
19689                              <1>       ; as it's console tty. (NOTE: 'p.ttyc' is used to set console
19690                              <1>       ; tty for tty switching by keyboard.)
19691                              <1>       ;
19692                              <1>       ; INPUT ->
19693                              <1>       ;        AL = wait channel (r3) ('tty number' for now)
19694                              <1>       ;        ;;EBX = Run queue (r2) offset
19695                              <1>       ;
```

```
19696                                  <1>       ; ((modified registers: EAX, EBX))
19697                                  <1>       ;
19698 000054B9 0FB6D8                  <1>       movzx  ebx, al ; 29/06/2015
19699 000054BC 81C3[C8700000]          <1>       add    ebx, wlist
19700 000054C2 8A03                    <1>       mov    al, [ebx] ; waiting list (waiting process number)
19701 000054C4 20C0                    <1>       and    al, al
19702 000054C6 7424                    <1>       jz     short wa0 ; nothing to wakeup
19703                                  <1>       ;
19704 000054C8 30E4                    <1>       xor    ah, ah
19705 000054CA 8825[8A740000]          <1>       mov    [u.quant], ah ; 0 ; time quantum = 0
19706 000054D0 8823                    <1>       mov    [ebx], ah ; 0 ; zero wait channel entry
19707                                  <1>       ; 15/09/2015
19708 000054D2 0FB6D8                  <1>       movzx  ebx, al
19709 000054D5 88A3[A5710000]          <1>       mov    [ebx+p.waitc-1], ah ; 0
19710 000054DB FEC4                    <1>       inc    ah
19711 000054DD 88A3[C5710000]          <1>       mov    byte [ebx+p.stat-1], ah ; 1 ; SRUN
19712                                  <1>       ;
19713 000054E3 57                      <1>       push   edi
19714 000054E4 52                      <1>       push   edx
19715 000054E5 E84AFFFFFF              <1>       call   putlu
19716 000054EA 5A                      <1>       pop    edx
19717 000054EB 5F                      <1>       pop    edi
19718                                  <1> wa0:
19719 000054EC C3                      <1>       retn
19720                                  <1>
19721                                  <1> sleep:
19722                                  <1>       ; 15/09/2015
19723                                  <1>       ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
19724                                  <1>       ;
19725                                  <1>       ; 09/05/2013 - 20/03/2014
19726                                  <1>       ;
19727                                  <1>       ; Retro UNIX 8086 v1 modification !
19728                                  <1>       ; (Process/task switching and quit routine by using
19729                                  <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.))
19730                                  <1>       ;
19731                                  <1>       ; In original UNIX v1, 'sleep' is called to wait for
19732                                  <1>       ; tty and tape output or input becomes available
19733                                  <1>       ; and process is put on waiting channel and swapped out,
19734                                  <1>       ; then -when the tty or tape is ready to write or read-
19735                                  <1>       ; 'wakeup' gets process back to active swapped-in status.)
19736                                  <1>       ;
19737                                  <1>       ; In Retro UNIX 8086 v1, Int 1Bh ctrl+brk interrupt and
19738                                  <1>       ; Int 09h keyboard interrupt will set 'quit' or 'switching'
19739                                  <1>       ; status of the current process also INT 1Ch will count down
19740                                  <1>       ; 'uquant' value and INT 09h will redirect scancode of keystroke
19741                                  <1>       ; to tty buffer of the current process and kernel will get
19742                                  <1>       ; user input by using tty buffer of the current process
19743                                  <1>       ; (instead of standard INT 16h interrupt).
19744                                  <1>       ; TTY output will be redirected to related video page of text mode
19745                                  <1>       ; (INT 10h will be called with different video page depending
19746                                  <1>       ; on tty assignment of the active process: 0 to 7 for
19747                                  <1>       ; pseudo screens.)
19748                                  <1>       ;
19749                                  <1>       ; In Retro UNIX 8086 v1, 'sleep' will be called to wait for
19750                                  <1>       ; a keystroke from keyboard or wait for reading or writing
19751                                  <1>       ; characters/data on serial port(s).
19752                                  <1>       ;
19753                                  <1>       ; Character/Terminal input/output through COM1 and COM2 will be
19754                                  <1>       ; performed by related routines in addition to pseudo TTY routines.
19755                                  <1>       ;
19756                                  <1>       ; R1 = AH = wait channel (0-9 for TTYs) ; 05/10/2013 (22/09/2013)
19757                                  <1>       ;
19758                                  <1>       ;; 05/10/2013
19759                                  <1>         ;10/12/2013
19760                                  <1>       ;cmp   byte [u.uno], 1
19761                                  <1>         ;ja    short sleep0
19762                                  <1>       ;retn
19763                                  <1>
19764                                  <1>       ; 20/03/2014
19765                                  <1>       ;mov   bx, [runq]
19766                                  <1>       ;cmp   bl, bh
19767                                  <1>       ;jne   short sleep0
19768                                  <1>       ; 25/02/2014
19769                                  <1>       ;cmp word ptr [runq], 0
19770                                  <1>       ;ja short sleep0
19771                                  <1>       ;retn
19772                                  <1> sleep0:
19773                                  <1>       ;
19774 000054ED E854000000              <1>       call   isintr
19775 000054F2 0F8562EBFFFF            <1>       jnz    sysret
19776                                  <1>              ; / wait for event
19777                                  <1>                     ; jsr r0,isintr / check to see if interrupt
19778                                  <1>                           ; / or quit from user
19779                                  <1>                            ; br 2f / something happened
19780                                  <1>                           ; / yes, his interrupt so return
19781                                  <1>                                 ;      / to user
19782                                  <1>
19783                                  <1>       ; 30/06/2015
19784 000054F8 0FB6DC                  <1>       movzx  ebx, ah ; 30/06/2015
19785 000054FB 81C3[C8700000]          <1>       add    ebx, wlist
19786 00005501 8A03                    <1>       mov    al, [ebx]
19787 00005503 20C0                    <1>       and    al, al
19788 00005505 7407                    <1>       jz     short sleep1
19789 00005507 53                      <1>       push   ebx
19790 00005508 E827FFFFFF              <1>       call   putlu
19791 0000550D 5B                      <1>       pop    ebx
19792                                  <1> sleep1:
19793 0000550E A0[97740000]            <1>       mov    al, [u.uno]
19794 00005513 8803                    <1>       mov    [ebx], al    ; put the process number
19795                                  <1>                           ; in the wait channel
19796                                  <1>              ; mov (r0)+,r1 / put number of wait channel in r1
19797                                  <1>              ; movb wlist(r1),-(sp) / put old process number in there,
19798                                  <1>                                   ; / on the stack
19799                                  <1>                    ; movb u.uno,wlist(r1) / put process number of process
19800                                  <1>                                   ; / to put to sleep in there
```

```
19801                                  <1>         ; 15/09/2015
19802 00005515 0FB6D8                  <1>       movzx  ebx, al
19803 00005518 C683[C5710000]04        <1>         mov     byte [ebx+p.stat-1], 4 ; SSLEEP
19804 0000551F FEC4                    <1>       inc    ah
19805 00005521 88A3[A5710000]          <1>       mov    [ebx+p.waitc-1], ah ; wait channel + 1
19806                                  <1>       ;
19807 00005527 66FF35[2E740000]        <1>       push    word [cdev]
19808                                  <1>             ; mov cdev,-(sp) / nothing happened in isintr so
19809 0000552E E833FEFFFF              <1>       call   swap
19810                                  <1>                    ; jsr r0,swap / swap out process that needs to sleep
19811 00005533 668F05[2E740000]        <1>         pop     word [cdev]
19812                                  <1>             ; mov (sp)+,cdev / restore device
19813 0000553A E807000000              <1>       call   isintr
19814                                  <1>       ; 22/09/2013
19815 0000553F 0F8515EBFFFF            <1>       jnz    sysret
19816                                  <1>             ; jsr r0,isintr / check for interrupt of new process
19817                                  <1>                          ; br 2f / yes, return to new user
19818                                  <1>             ; movb (sp)+,r1 / no, r1 = old process number that was
19819                                  <1>                          ; / originally on the wait channel
19820                                  <1>                  ; beq 1f / if 0 branch
19821                                  <1>             ; mov $runq+4,r2 / r2 points to lowest priority queue
19822                                  <1>                  ; mov $300,*$ps / processor priority = 6
19823                                  <1>             ; jsr r0,putlu / create link to old process number
19824                                  <1>                  ; clr *$ps / clear the status; process priority = 0
19825                                  <1>     ;1:
19826 00005545 C3                      <1>       retn
19827                                  <1>             ; rts r0 / return
19828                                  <1>     ;2:
19829                                  <1>         ;;jmp      sysret
19830                                  <1>             ; jmp sysret / return to user
19831                                  <1>
19832                                  <1> isintr:
19833                                  <1>       ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
19834                                  <1>       ;
19835                                  <1>       ; 09/05/2013 - 30/05/2014
19836                                  <1>       ;
19837                                  <1>       ; Retro UNIX 8086 v1 modification !
19838                                  <1>       ; (Process/task switching and quit routine by using
19839                                  <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.))
19840                                  <1>       ;
19841                                  <1>       ; Retro UNIX 8086 v1 modification:
19842                                  <1>       ; 'isintr' checks if user interrupt request is enabled
19843                                  <1>       ;  and there is a 'quit' request by user;
19844                                  <1>       ;  otherwise, 'isintr' will return with zf=1 that means
19845                                  <1>       ;  "nothing to do". (20/10/2013)
19846                                  <1>       ;
19847                                  <1>       ; 20/10/2013
19848 00005546 66833D[78740000]00      <1>       cmp    word [u.ttyp], 0 ; has process got a tty ?
19849 0000554E 7622                    <1>       jna    short isintr2 ; retn
19850                                  <1>       ; 03/09/2013
19851                                  <1>       ; (nothing to do)
19852                                  <1>       ;retn
19853                                  <1>       ; 22/09/2013
19854 00005550 66833D[8C740000]00      <1>       cmp    word [u.intr], 0
19855 00005558 7618                    <1>       jna    short isintr2 ; retn
19856                                  <1>       ; 30/05/2014
19857 0000555A 6650                    <1>       push   ax
19858 0000555C 66A1[8E740000]          <1>       mov    ax, [u.quit]
19859 00005562 6609C0                  <1>       or     ax, ax ; 0 ?
19860 00005565 7409                    <1>       jz     short isintr1 ; zf = 1
19861 00005567 6683F8FE                <1>       cmp    ax, 0FFFEh  ; 'ctrl + brk' check
19862 0000556B 7703                    <1>       ja     short isintr1 ; 0FFFFh, zf = 0
19863 0000556D 6631C0                  <1>       xor    ax, ax ; zf = 1
19864                                  <1> isintr1:
19865 00005570 6658                    <1>       pop    ax
19866                                  <1> isintr2: ; 22/09/2013
19867                                  <1>       ; zf=1 -> nothing to do
19868 00005572 C3                      <1>       retn
19869                                  <1>
19870                                  <1>       ; UNIX v1 original 'isintr' routine...
19871                                  <1>             ;mov     r1,-(sp) / put number of wait channel on the stack
19872                                  <1>             ;mov     r2,-(sp) / save r2
19873                                  <1>             ;mov     u.ttyp,r1 / r1 = pointer to buffer of process control
19874                                  <1>         ;                / typewriter
19875                                  <1>             ;beq     1f / if 0, do nothing except skip return
19876                                  <1>             ;movb    6(r1),r1 / put interrupt char in the tty buffer in r1
19877                                  <1>             ;beq     1f / if its 0 do nothing except skip return
19878                                  <1>             ;cmp     r1,$177 / is interrupt char = delete?
19879                                  <1>             ;bne     3f / no, so it must be a quit (fs)
19880                                  <1>             ;tst     u.intr / yes, value of u.intr determines handling
19881                                  <1>         ;              / of interrupts
19882                                  <1>             ;bne     2f / if not 0, 2f. If zero do nothing.
19883                                  <1>     ;1:
19884                                  <1>             ;tst     (r0)+ / bump r0 past system return (skip)
19885                                  <1>     ;4:
19886                                  <1>             ;mov     (sp)+,r2 / restore r1 and r2
19887                                  <1>             ;mov     (sp)+,r1
19888                                  <1>             ;rts     r0
19889                                  <1>     ;3: / interrupt char = quit (fs)
19890                                  <1>             ;tst     u.quit / value of u.quit determines handling of quits
19891                                  <1>             ;beq     1b / u.quit = 0 means do nothing
19892                                  <1>     ;2: / get here because either u.intr <> 0 or u.qult <> 0
19893                                  <1>             ;mov     $tty+6,r1 / move pointer to tty block into r1
19894                                  <1>     ;1: / find process control tty entry in tty block
19895                                  <1>             ;cmp     (r1),u.ttyp / is this the process control tty buffer?
19896                                  <1>             ;beq     1f / block found go to 1f
19897                                  <1>             ;add     $8,r1 / look at next tty block
19898                                  <1>             ;cmp     r1,$tty+[ntty*8]+6 / are we at end of tty blocks
19899                                  <1>             ;blo     1b / no
19900                                  <1>             ;br      4b / no process control tty found so go to 4b
19901                                  <1>     ;1:
19902                                  <1>             ;mov     $240,*$ps / set processor priority to 5
19903                                  <1>             ;movb    -3(r1),0f / load getc call argument; character llst
19904                                  <1>         ;                / identifier
19905                                  <1>             ;inc     0f / increment
```

```
19906                                  <1>      ;1:
19907                                  <1>              ;jsr     r0,getc; 0:.. / erase output char list for control
19908                                  <1>         ;        br 4b / process tty. This prevents a line of stuff
19909                                  <1>         ;             / being typed out after you hit the interrupt
19910                                  <1>         ;             / key
19911                                  <1>              ;br      1b
19912                                      %include 'u5.s'        ; 03/06/2015
19913                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS5.INC
19914                                  <1> ; Last Modification: 14/11/2015
19915                                  <1> ; ---------------------------------------------------------------------------
19916                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
19917                                  <1> ; (v0.1 - Beginning: 11/07/2012)
19918                                  <1> ;
19919                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
19920                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
19921                                  <1> ; <Bell Laboratories (17/3/1972)>
19922                                  <1> ; <Preliminary Release of UNIX Implementation Document>
19923                                  <1> ;
19924                                  <1> ; Retro UNIX 8086 v1 - U5.ASM (07/08/2013) //// UNIX v1 -> u5.s
19925                                  <1> ;
19926                                  <1> ; ***************************************************************************
19927                                  <1>
19928                                  <1> mget:
19929                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
19930                                  <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
19931                                  <1>       ;
19932                                  <1>       ; Get existing or (allocate) a new disk block for file
19933                                  <1>       ;
19934                                  <1>       ; INPUTS ->
19935                                  <1>       ;    u.fofp (file offset pointer)
19936                                  <1>       ;    inode
19937                                  <1>       ;    u.off (file offset)
19938                                  <1>       ; OUTPUTS ->
19939                                  <1>       ;    r1 (physical block number)
19940                                  <1>       ;    r2, r3, r5 (internal)
19941                                  <1>       ;
19942                                  <1>       ; ((AX = R1)) output
19943                                  <1>       ;    (Retro UNIX Prototype : 05/03/2013 - 14/11/2012, UNIXCOPY.ASM)
19944                                  <1>         ;    ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
19945                                  <1>
19946                                  <1>             ; mov *u.fofp,mq / file offset in mq
19947                                  <1>             ; clr ac / later to be high sig
19948                                  <1>             ; mov $-8,lsh   / divide ac/mq by 256.
19949                                  <1>             ; mov mq,r2
19950                                  <1>             ; bit $10000,i.flgs / lg/sm is this a large or small file
19951                                  <1>             ; bne 4f / branch for large file
19952                                  <1> mget_0:
19953 00005573 8B35[58740000]        <1>         mov     esi, [u.fofp]
19954 00005579 0FB65E01              <1>         movzx   ebx, byte [esi+1]
19955                                  <1>         ; BX = r2
19956 0000557D 66F705[16710000]00-   <1>         test       word [i.flgs], 4096 ; 1000h
19957 00005585 10                    <1>
19958                                  <1>                                ; is this a large or small file
19959 00005586 756F                  <1>       jnz    short mget_5 ; 4f ; large file
19960                                  <1>
19961 00005588 F6C3F0                <1>         test    bl, 0F0h ; !0Fh
19962                                  <1>             ; bit $!17,r2
19963 0000558B 7526                  <1>       jnz    short mget_2
19964                                  <1>             ; bne 3f / branch if r2 greater than or equal to 16
19965 0000558D 80E30E                <1>         and     bl, 0Eh
19966                                  <1>             ; bic $!16,r2 / clear all bits but bits 1,2,3
19967 00005590 0FB783[1C710000]      <1>       movzx  eax, word [ebx+i.dskp] ; AX = R1, physical block number
19968                                  <1>             ; mov i.dskp(r2),r1 / r1 has physical block number
19969 00005597 6609C0                <1>       or     ax, ax
19970 0000559A 7516                  <1>       jnz    short mget_1
19971                                  <1>             ; bne 2f / if physical block num is zero then need a new block
19972                                  <1>                    ; / for file
19973 0000559C E8AB000000            <1>       call   alloc
19974                                  <1>             ; jsr r0,alloc / allocate a new block
19975                                  <1>          ; eAX (r1) = Physical block number
19976 000055A1 668983[1C710000]      <1>       mov    [ebx+i.dskp], ax
19977                                  <1>             ; mov r1,i.dskp(r2) / physical block number stored in i-node
19978 000055A8 E84C020000            <1>       call   setimod
19979                                  <1>             ; jsr r0,setimod / set inode modified byte (imod)
19980 000055AD E8B1FEFFFF            <1>       call   clear
19981                                  <1>             ; jsr r0,clear / zero out disk/drum block just allocated
19982                                  <1> mget_1: ; 2:
19983                                  <1>         ; eAX (r1) = Physical block number
19984 000055B2 C3                    <1>       retn
19985                                  <1>             ; rts r0
19986                                  <1> mget_2: ; 3: / adding on block which changes small file to a large file
19987 000055B3 E894000000            <1>       call   alloc
19988                                  <1>             ; jsr r0,alloc / allocate a new block for this file;
19989                                  <1>                            ; / block number in r1
19990                                  <1>         ; eAX (r1) = Physical block number
19991 000055B8 E8560C0000            <1>       call   wslot
19992                                  <1>             ; jsr r0,wslot / set up I/O buffer for write, r5 points to
19993                                  <1>                          ; / first data word in buffer
19994                                  <1>         ; eAX (r1) = Physical block number
19995 000055BD B908000000            <1>       mov    ecx, 8  ; R3, transfer old physical block pointers
19996                                  <1>                    ; into new indirect block area for the new
19997                                  <1>                    ; large file
19998 000055C2 89DF                  <1>       mov    edi, ebx ; r5
19999 000055C4 BE[1C710000]          <1>       mov    esi, i.dskp
20000                                  <1>             ; mov $8.,r3 / next 6 instructions transfer old physical
20001                                  <1>                      ; / block pointers
20002                                  <1>             ; mov $i.dskp,r2 / into new indirect block for the new
20003                                  <1>                     ; / large file
20004 000055C9 6631C0                <1>       xor    ax, ax ; mov ax, 0
20005                                  <1> mget_3: ;1:
20006 000055CC 66A5                  <1>       movsw
20007                                  <1>             ; mov (r2),(r5)+
20008 000055CE 668946FE              <1>       mov    [esi-2], ax
20009                                  <1>             ; clr (r2)+
20010 000055D2 E2F8                  <1>       loop   mget_3 ; 1b
```

```
20011                                  <1>             ; dec r3
20012                                  <1>             ; bgt 1b
20013                                  <1>
20014 000055D4 B1F8                    <1>       mov   cl, 256-8
20015                                  <1>             ; mov $256.-8.,r3 / clear rest of data buffer
20016                                  <1> mget_4:     ; 1
20017 000055D6 F366AB                  <1>       rep   stosw
20018                                  <1>             ; clr (r5)+
20019                                  <1>             ; dec r3
20020                                  <1>             ; bgt 1b
20021                                  <1>       ; 24/03/2013
20022                                  <1>         ; AX (r1) = Physical block number
20023 000055D9 E8510C0000              <1>       call  dskwr
20024                                  <1>             ; jsr r0,dskwr / write new indirect block on disk
20025                                  <1>         ; eAX (r1) = Physical block number
20026 000055DE 66A3[1C710000]          <1>       mov   [i.dskp], ax
20027                                  <1>             ; mov r1,i.dskp / put pointer to indirect block in i-node
20028 000055E4 66810D[16710000]00-     <1>       or    word [i.flgs], 4096 ; 1000h
20029 000055EC 10                      <1>
20030                                  <1>             ; bis $10000,i.flgs / set large file bit
20031                                  <1>                           ; / in i.flgs word of i-node
20032 000055ED E807020000              <1>       call  setimod
20033                                  <1>             ; jsr r0,setimod / set i-node modified flag
20034 000055F2 E97CFFFFFF              <1>         jmp     mget_0
20035                                  <1>             ; br mget
20036                                  <1>
20037                                  <1> mget_5:  ; 4 ; large file
20038                                  <1>             ; mov $-8,lsh / divide byte number by 256.
20039                                  <1>             ; bic $!776,r2 / zero all bits but 1,2,3,4,5,6,7,8; gives offset
20040                                  <1>                       ; / in indirect block
20041                                  <1>             ; mov r2,-(sp) / save on stack (*)
20042                                  <1>             ; mov mq,r2 / calculate offset in i-node for pointer to proper
20043                                  <1>                           ; / indirect block
20044                                  <1>           ; bic $!16,r2
20045 000055F7 80E3FE                 <1>         and     bl, 0FEh ; bh = 0
20046 000055FA 53                      <1>         push    ebx  ; i-node pointer offset in indirect block  (*)
20047                                  <1>         ; 01/03/2013 Max. possible BX (offset) value is 127 (65535/512)
20048                                  <1>       ;          for this file system (offset 128 to 255 not in use)
20049                                  <1>       ; There is always 1 indirect block for this file system
20050 000055FB 0FB705[1C710000]        <1>       movzx  eax, word [i.dskp] ; i.dskp[0]
20051                                  <1>             ; mov i.dskp(r2),r1
20052 00005602 6609C0                  <1>       or    ax, ax ; R1
20053 00005605 7515                    <1>       jnz   short mget_6 ; 2f
20054                                  <1>             ; bne 2f / if no indirect block exists
20055 00005607 E840000000              <1>       call  alloc
20056                                  <1>             ; jsr r0,alloc / allocate a new block
20057 0000560C 66A3[1C710000]          <1>       mov   [i.dskp], ax  ; 03/03/2013
20058                                  <1>             ; mov r1,i.dskp(r2) / put block number of new block in i-node
20059 00005612 E8E2010000              <1>       call  setimod
20060                                  <1>             ; jsr r0,setimod / set i-node modified byte
20061                                  <1>       ; eAX = new block number
20062 00005617 E847FEFFFF              <1>       call  clear
20063                                  <1>             ; jsr r0,clear / clear new block
20064                                  <1> mget_6: ;2
20065                                  <1>       ; 05/03/2013
20066                                  <1>       ; eAX = r1, physical block number (of indirect block)
20067 0000561C E8920B0000              <1>       call  dskrd ; read indirect block
20068                                  <1>             ; jsr r0,dskrd / read in indirect block
20069 00005621 5A                      <1>       pop   edx  ; R2, get offset (*)
20070                                  <1>             ; mov (sp)+,r2 / get offset
20071                                  <1>       ; eAX = r1, physical block number (of indirect block)
20072 00005622 50                      <1>       push  eax ; ** ; 24/03/2013
20073                                  <1>             ; mov r1,-(sp) / save block number of indirect block on stack
20074                                  <1>       ; eBX (r5) = pointer to buffer (indirect block)
20075 00005623 01D3                    <1>       add   ebx, edx ; / r5 points to first word in indirect block, r2
20076                                  <1>             ; add r5,r2 / r5 points to first word in indirect block, r2
20077                                  <1>                         ; / points to location of inter
20078 00005625 0FB703                  <1>       movzx  eax, word [ebx] ; put physical block no of block
20079                                  <1>                          ; in file sought in R1 (AX)
20080                                  <1>             ; mov (r2),r1 / put physical block no of block in file
20081                                  <1>                           ; / sought in r1
20082 00005628 6609C0                  <1>       or    ax, ax
20083 0000562B 751D                    <1>         jnz  short mget_7 ; 2f
20084                                  <1>             ; bne 2f / if no block exists
20085 0000562D E81A000000              <1>       call  alloc
20086                                  <1>             ; jsr r0,alloc / allocate a new block
20087 00005632 668903                  <1>       mov   [ebx], ax ; R1
20088                                  <1>             ; mov r1,(r2) / put new block number into proper location in
20089                                  <1>                           ; / indirect block
20090 00005635 5A                      <1>       pop   edx ; ** ; 24/03/2013
20091                                  <1>             ; mov (sp)+,r1 / get block number of indirect block
20092 00005636 52                      <1>       push  edx ; ** ; 31/07/2013
20093 00005637 50                      <1>       push  eax ; * ; 24/03/2013, 31/07/2013 (new block number)
20094 00005638 89D0                    <1>       mov   eax, edx ; 24/03/2013
20095                                  <1>             ; mov (r2),-(sp) / save block number of new block
20096                                  <1>       ; eAX (r1) = physical block number (of indirect block)
20097 0000563A E8D40B0000              <1>       call  wslot
20098                                  <1>             ; jsr r0,wslot
20099                                  <1>         ; eAX (r1) = physical block number
20100                                  <1>       ; eBX (r5) = pointer to buffer (indirect block)
20101 0000563F E8EB0B0000              <1>       call  dskwr
20102                                  <1>       ; eAX = r1 = physical block number (of indirect block)
20103                                  <1>             ; jsr r0,dskwr / write newly modified indirect block
20104                                  <1>                          ; / back out on disk
20105 00005644 58                      <1>       pop   eax ; *  ; 31/07/2013
20106                                  <1>             ; mov (sp),r1 / restore block number of new block
20107                                  <1>       ; eAX (r1) = physical block number of new block
20108 00005645 E819FEFFFF              <1>       call  clear
20109                                  <1>             ; jsr r0,clear / clear new block
20110                                  <1> mget_7: ; 2
20111 0000564A 5A                      <1>       pop   edx ; **
20112                                  <1>             ; tst (sp)+ / bump stack pointer
20113                                  <1>       ; eAX (r1) = Block number of new block
20114 0000564B C3                      <1>       retn
20115                                  <1>             ; rts r0
```

```
20116                                  <1>
20117                                  <1> alloc:
20118                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20119                                  <1>       ; 01/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20120                                  <1>       ;
20121                                  <1>       ; get a free block and
20122                                  <1>       ; set the corresponding bit in the free storage map
20123                                  <1>       ;
20124                                  <1>       ; INPUTS ->
20125                                  <1>       ;    cdev (current device)
20126                                  <1>       ;    r2
20127                                  <1>       ;    r3
20128                                  <1>       ; OUTPUTS ->
20129                                  <1>       ;    r1 (physical block number of block assigned)
20130                                  <1>       ;    smod, mmod, systm (super block), mount (mountable super block)
20131                                  <1>       ;
20132                                  <1>       ; ((AX = R1)) output
20133                                  <1>       ;    (Retro UNIX Prototype : 14/11/2012 - 21/07/2012, UNIXCOPY.ASM)
20134                                  <1>         ;    ((Modified registers: DX, CX))
20135                                  <1>
20136                                  <1>             ;mov r2,-(sp) / save r2, r3 on stack
20137                                  <1>             ;mov r3,-(sp)
20138                                  <1>       ;push  ecx
20139 0000564C 53                      <1>       push   ebx ; R2
20140                                  <1>       ;push  edx ; R3
20141 0000564D BB[08810000]            <1>       mov    ebx, systm ; SuperBlock
20142                                  <1>              ; mov $systm,r2 / start of inode and free storage map for drum
20143 00005652 803D[2E740000]00        <1>       cmp    byte [cdev], 0
20144                                  <1>              ; tst cdev
20145 00005659 7605                    <1>       jna    short alloc_1
20146                                  <1>              ; beq 1f / drum is device
20147 0000565B BB[10830000]            <1>       mov    ebx, mount
20148                                  <1>              ; mov $mount,r2 / disk or tape is device, start of inode and
20149                                  <1>                              ; / free storage map
20150                                  <1> alloc_1: ; 1
20151 00005660 668B0B                  <1>         mov  cx, [ebx]
20152                                  <1>              ; mov (r2)+,r1 / first word contains number of bytes in free
20153                                  <1>                             ; / storage map
20154 00005663 66C1E103                <1>       shl    cx, 3
20155                                  <1>              ; asl r1 / multiply r1 by eight gives
20156                                  <1>              ; number of blocks in device
20157                                  <1>              ; asl r1
20158                                  <1>              ; asl r1
20159                                  <1>       ;; push cx ;; 01/08/2013
20160                                  <1>              ; mov r1,-(sp) / save # of blocks in device on stack
20161 00005667 31C0                    <1>       xor    eax, eax ; 0
20162                                  <1>              ; clr r1 / r1 contains bit count of free storage map
20163                                  <1> alloc_2: ; 1
20164 00005669 43                      <1>       inc    ebx ; 18/8/2012
20165 0000566A 43                      <1>       inc    ebx ;
20166 0000566B 668B13                  <1>       mov    dx, [ebx]
20167                                  <1>              ; mov (r2)+,r3 / word of free storage map in r3
20168 0000566E 6609D2                  <1>       or     dx, dx
20169 00005671 750E                    <1>       jnz    short alloc_3 ; 1f
20170                                  <1>              ; bne 1f / branch if any free blocks in this word
20171 00005673 6683C010                <1>       add    ax, 16
20172                                  <1>              ; add $16.,r1
20173 00005677 6639C8                  <1>       cmp    ax, cx
20174                                  <1>              ; cmp r1 ,(sp) / have we examined all free storage bytes
20175 0000567A 72ED                    <1>       jb     short alloc_2
20176                                  <1>              ; blo 1b
20177                                  <1>       ; 14/11/2015
20178                                  <1>       ; Note: If the super block buffer has wrong content (zero bytes)
20179                                  <1>       ;      because of a (DMA or another) r/w error,
20180                                  <1>       ;      we will be here, at 'jmp panic' code address,
20181                                  <1>       ;      even if the (disk) file system space is not full !!!
20182                                  <1>       ;      (cx = 0)
20183                                  <1>       ;
20184 0000567C E94FE2FFFF              <1>       jmp     panic
20185                                  <1>              ; jmp panic / found no free storage
20186                                  <1> alloc_3: ; 1
20187 00005681 66D1EA                  <1>       shr    dx, 1
20188                                  <1>              ; asr r3 / find a free block
20189 00005684 7204                    <1>       jc     short alloc_4 ; 1f
20190                                  <1>              ; bcs 1f / branch when free block found; bit for block k
20191                                  <1>                     ; / is in byte k/8 / in bit k (mod 8)
20192 00005686 6640                    <1>       inc    ax
20193                                  <1>              ; inc r1 / increment bit count in bit k (mod8)
20194 00005688 EBF7                    <1>       jmp    short alloc_3
20195                                  <1>              ; br 1b
20196                                  <1> alloc_4: ; 1:
20197                                  <1>       ;; pop cx ;; 01/08/2013
20198                                  <1>              ; tst (sp)+ / bump sp
20199                                  <1>       ; 02/04/2013
20200 0000568A E829000000              <1>       call   free3
20201                                  <1>              ; jsr r0,3f / have found a free block
20202                                  <1>       ; 21/8/2012
20203 0000568F 66F7D2                  <1>       not    dx ; masking bit is '0' and others are '1'
20204 00005692 662113                  <1>       and    [ebx], dx   ;; 0 -> allocated
20205                                  <1>              ; bic r3,(r2) / set bit for this block
20206                                  <1>                            ; / i.e. assign block
20207                                  <1>              ; br 2f
20208 00005695 EB09                    <1>       jmp    short alloc_5
20209                                  <1>
20210                                  <1> free:
20211                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20212                                  <1>       ; 07/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20213                                  <1>       ;
20214                                  <1>       ; calculates byte address and bit position for given block number
20215                                  <1>       ; then sets the corresponding bit in the free storage map
20216                                  <1>       ;
20217                                  <1>       ; INPUTS ->
20218                                  <1>       ;    r1 - block number for a block structured device
20219                                  <1>       ;    cdev - current device
20220                                  <1>       ; OUTPUTS ->
```

```
20221                                  <1>       ;    free storage map is updated
20222                                  <1>       ;    smod is incremented if cdev is root device (fixed disk)
20223                                  <1>       ;    mmod is incremented if cdev is a removable disk
20224                                  <1>       ;
20225                                  <1>       ;  (Retro UNIX Prototype : 01/12/2012, UNIXCOPY.ASM)
20226                                  <1>         ;  ((Modified registers: DX, CX))
20227                                  <1>
20228                                  <1>             ;mov r2,-(sp) / save r2, r3
20229                                  <1>             ;mov r3,-(sp)
20230                                  <1>       ;push  ecx
20231 00005697 53                     <1>       push   ebx ; R2
20232                                  <1>       ;push  edx ; R3
20233                                  <1>
20234 00005698 E81B000000            <1>         call    free3
20235                                  <1>             ; jsr r0,3f  / set up bit mask and word no.
20236                                  <1>                          ; / in free storage map for block
20237 0000569D 660913                <1>       or     [ebx], dx
20238                                  <1>             ; bis r3, (r2) / set free storage block bit;
20239                                  <1>                       ;  / indicates free block
20240                                  <1>       ; 0 -> allocated, 1 -> free
20241                                  <1>
20242                                  <1> alloc_5:
20243                                  <1>       ; 07/04/2013
20244                                  <1> free_1: ; 2:
20245                                  <1>       ; pop  edx
20246                                  <1>             ; mov (sp)+,r3 / restore r2, r3
20247 000056A0 5B                     <1>       pop    ebx
20248                                  <1>             ; mov (sp)+,r2
20249                                  <1>       ; pop  ecx
20250 000056A1 803D[2E740000]00      <1>       cmp    byte [cdev], 0
20251                                  <1>             ; tst cdev / cdev = 0, block structured, drum;
20252                                  <1>                    ; / cdev = 1, mountable device
20253 000056A8 7707                  <1>       ja     short alloc_6 ; 1f
20254                                  <1>             ; bne 1f
20255                                  <1>       ;mov   byte [smod], 1
20256 000056AA FE05[3D740000]        <1>       inc    byte [smod]
20257                                  <1>             ; incb smod / set super block modified for drum
20258                                  <1>       ; eAX (r1) = block number
20259 000056B0 C3                     <1>       retn
20260                                  <1>             ; rts r0
20261                                  <1> free_2:
20262                                  <1> alloc_6: ; 1:
20263                                  <1>       ;mov   byte [mmod], 1
20264 000056B1 FE05[3E740000]        <1>       inc    byte [mmod]
20265                                  <1>             ; incb mmod
20266                                  <1>               ; / set super block modified for mountable device
20267                                  <1>       ; eAX (r1) = block number
20268 000056B7 C3                     <1>       retn
20269                                  <1>             ; rts r0
20270                                  <1> free3:
20271                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20272                                  <1>       ; 02/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20273                                  <1>       ;
20274                                  <1>       ; free3 is called from 'alloc' and 'free' procedures
20275                                  <1>       ;
20276                                  <1> alloc_free_3: ; 3
20277 000056B8 66BA0100             <1>       mov    dx, 1
20278 000056BC 88C1                  <1>       mov    cl, al
20279                                  <1>             ; mov r1,r2 / block number, k, = 1
20280 000056BE 80E10F                <1>       and    cl, 0Fh  ; 0Fh <-- (k) mod 16
20281                                  <1>             ; bic $!7,r2 / clear all bits but 0,1,2; r2 = (k) mod (8)
20282 000056C1 7403                  <1>       jz     short free4
20283                                  <1>             ; bisb 2f(r2),r3 / use mask to set bit in r3 corresponding to
20284                                  <1>                              ; / (k) mod 8
20285 000056C3 66D3E2                <1>       shl    dx, cl
20286                                  <1> free4:
20287 000056C6 0FB7D8                <1>       movzx  ebx, ax
20288                                  <1>             ; mov r1,r2 / divide block number by 16
20289 000056C9 66C1EB04              <1>       shr    bx, 4
20290                                  <1>             ; asr r2
20291                                  <1>             ; asr r2
20292                                  <1>             ; asr r2
20293                                  <1>             ; asr r2
20294                                  <1>             ; bcc 1f / branch if bit 3 in r1 was 0 i.e.,
20295                                  <1>                    ; / bit for block is in lower half of word
20296                                  <1>             ; swab r3 / swap bytes in r3; bit in upper half of word in free
20297                                  <1>                      ; / storage map
20298                                  <1> alloc_free_4: ; 1
20299 000056CD 66D1E3                <1>       shl    bx, 1
20300                                  <1>             ; asl r2 / multiply block number by 2; r2 = k/8
20301 000056D0 81C3[0A810000]        <1>       add    ebx, systm+2 ; SuperBlock+2
20302                                  <1>             ; add $systm+2,r2 / address of word of free storage map for drum
20303                                  <1>                               ; / with block bit in it
20304 000056D6 803D[2E740000]00      <1>       cmp    byte [cdev], 0
20305                                  <1>             ; tst cdev
20306 000056DD 7606                  <1>       jna    short alloc_free_5
20307                                  <1>             ; beq 1f / cdev = 0 indicates device is drum
20308 000056DF 81C308020000          <1>       add    ebx, mount - systm
20309                                  <1>             ; add $mount-systm,r2 / address of word of free storage map for
20310                                  <1>                                   ; / mountable device with bit of block to be
20311                                  <1>                                   ; / freed
20312                                  <1> alloc_free_5: ; 1
20313 000056E5 C3                     <1>       retn
20314                                  <1>             ; rts r0 / return to 'free'
20315                                  <1>            ; 2
20316                                  <1>              ; .byte    1,2,4,10,20,40,100,200 / masks for bits 0,...,7
20317                                  <1>
20318                                  <1> iget:
20319                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20320                                  <1>       ; 07/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
20321                                  <1>       ;
20322                                  <1>       ; get a new i-node whose i-number in r1 and whose device is in cdev
20323                                  <1>       ;
20324                                  <1>       ; ('iget' returns current i-number in r1, if input value of r1 is 0)
20325                                  <1>       ;
```

```
20326                                  <1>       ; INPUTS ->
20327                                  <1>       ;    ii - current i-number, rootdir
20328                                  <1>       ;    cdev - new i-node device
20329                                  <1>       ;    idev - current i-node device
20330                                  <1>       ;    imod - current i-node modified flag
20331                                  <1>       ;    mnti - cross device file i-number
20332                                  <1>       ;    r1 - i-numbe rof new i-node
20333                                  <1>       ;    mntd - mountable device number
20334                                  <1>       ;
20335                                  <1>       ; OUTPUTS ->
20336                                  <1>       ;    cdev, idev, imod, ii, r1
20337                                  <1>       ;
20338                                  <1>       ; ((AX = R1)) input/output
20339                                  <1>       ;
20340                                  <1>       ;  (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)
20341                                  <1>         ;  ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
20342                                  <1>
20343 000056E6 8A15[2E740000]         <1>       mov    dl, [cdev] ; 18/07/2013
20344 000056EC 8A35[2C740000]         <1>       mov    dh, [idev] ; 07/08/2013
20345                                  <1>       ;
20346 000056F2 663B05[2A740000]       <1>       cmp    ax, [ii]
20347                                  <1>              ; cmp r1,ii / r1 = i-number of current file
20348 000056F9 7504                   <1>       jne    short iget_1
20349                                  <1>              ; bne 1f
20350 000056FB 38F2                   <1>       cmp    dl, dh
20351                                  <1>              ; cmp idev,cdev
20352                                  <1>                      ; / is device number of i-node = current device
20353 000056FD 7476                   <1>         je      short iget_5
20354                                  <1>              ; beq 2f
20355                                  <1> iget_1: ; 1:
20356 000056FF 30DB                   <1>       xor    bl, bl
20357 00005701 381D[3C740000]         <1>       cmp    [imod], bl ; 0
20358                                  <1>              ; tstb imod / has i-node of current file
20359                                  <1>                      ; / been modified i.e., imod set
20360 00005707 762D                   <1>       jna    short iget_2
20361                                  <1>              ; beq 1f
20362 00005709 881D[3C740000]         <1>       mov    [imod], bl ; 0
20363                                  <1>              ;  clrb      imod / if it has,
20364                                  <1>                       ; / we must write the new i-node out on disk
20365 0000570F 6650                   <1>       push   ax
20366                                  <1>              ; mov r1,-(sp)
20367                                  <1>       ;mov   dl, [cdev]
20368 00005711 6652                   <1>       push   dx
20369                                  <1>              ; mov cdev,-(sp)
20370 00005713 66A1[2A740000]         <1>       mov    ax, [ii]
20371                                  <1>              ; mov ii,r1
20372                                  <1>       ;mov   dh, [idev]
20373 00005719 8835[2E740000]         <1>       mov    [cdev], dh
20374                                  <1>              ; mov idev,cdev
20375 0000571F FEC3                   <1>       inc    bl ; 1
20376                                  <1>       ; 31/07/2013
20377 00005721 881D[CC740000]         <1>       mov     [rw], bl ; 1 == write
20378                                  <1>       ;;28/07/2013 rw -> u.rw
20379                                  <1>         ;;mov   [u.rw], bl ; 1 == write
20380 00005727 E84A000000             <1>       call   icalc
20381                                  <1>              ; jsr r0,icalc; 1
20382 0000572C 665A                   <1>       pop    dx
20383 0000572E 8815[2E740000]         <1>       mov    [cdev], dl
20384                                  <1>              ; mov (sp)+,cdev
20385 00005734 6658                   <1>       pop    ax
20386                                  <1>              ; mov (sp)+,r1
20387                                  <1> iget_2: ; 1:
20388 00005736 6621C0                 <1>       and    ax, ax
20389                                  <1>              ; tst r1 / is new i-number non zero
20390 00005739 7434                   <1>       jz     short iget_4 ; 2f
20391                                  <1>              ; beq 2f / branch if r1=0
20392                                  <1>
20393                                  <1>       ; mov  dl, [cdev]
20394 0000573B 08D2                   <1>       or     dl, dl
20395                                  <1>              ; tst cdev / is the current device number non zero
20396                                  <1>                     ; / (i.e., device =/ drum)
20397 0000573D 7517                   <1>       jnz    short iget_3 ;  1f
20398                                  <1>              ; bne 1f / branch 1f cdev =/ 0  ;; (cdev != 0)
20399 0000573F 663B05[34740000]       <1>       cmp    ax, [mnti]
20400                                  <1>              ; cmp r1,mnti / mnti is the i-number of the cross device
20401                                  <1>                       ; / file (root directory of mounted device)
20402 00005746 750E                   <1>       jne    short iget_3 ; 1f
20403                                  <1>              ; bne 1f
20404                                  <1>         ;mov    bl, [mntd]
20405 00005748 FEC2                   <1>       inc    dl ; mov dl, 1 ; 17/07/2013
20406 0000574A 8815[2E740000]         <1>         mov  [cdev], dl ; 17/07/2013 - 09/07/2013
20407                                  <1>              ; mov mntd,cdev / make mounted device the current device
20408 00005750 66A1[38740000]         <1>       mov    ax, [rootdir]
20409                                  <1>              ; mov rootdir,r1
20410                                  <1> iget_3: ; 1:
20411 00005756 66A3[2A740000]         <1>       mov    [ii], ax
20412                                  <1>              ; mov r1,ii
20413 0000575C 8815[2C740000]         <1>       mov    [idev], dl ; cdev
20414                                  <1>              ; mov cdev,idev
20415 00005762 30DB                   <1>       xor    bl, bl
20416                                  <1>         ; 31/07/2013
20417 00005764 881D[CC740000]         <1>       mov     [rw], bl ; 0 == read
20418                                  <1>       ;;28/07/2013 rw -> u.rw
20419                                  <1>         ;;mov   [u.rw], bl ; 0 = read
20420 0000576A E807000000             <1>       call   icalc
20421                                  <1>              ; jsr r0,icalc; 0 / read in i-node ii
20422                                  <1> iget_4: ; 2:
20423 0000576F 66A1[2A740000]         <1>       mov    ax, [ii]
20424                                  <1>              ; mov ii,r1
20425                                  <1> iget_5:
20426 00005775 C3                     <1>       retn
20427                                  <1>              ; rts r0
20428                                  <1>
20429                                  <1> icalc:
20430                                  <1>       ; 02/07/2015
```

```
20431                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20432                                  <1>       ; 07/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
20433                                  <1>       ;
20434                                  <1>       ; calculate physical block number from i-number then
20435                                  <1>       ; read or write that block
20436                                  <1>       ;
20437                                  <1>       ; 'icalc' is called from 'iget'
20438                                  <1>       ;
20439                                  <1>       ; for original unix v1:
20440                                  <1>       ; / i-node i is located in block (i+31.)/16. and begins 32.*
20441                                  <1>            ; / (i+31.) mod 16. bytes from its start
20442                                  <1>       ;
20443                                  <1>       ; for retro unix 8086 v1:
20444                                  <1>       ;  i-node is located in block (i+47)/16 and
20445                                  <1>       ;  begins 32*(i+47) mod 16 bytes from its start
20446                                  <1>       ;
20447                                  <1>       ; INPUTS ->
20448                                  <1>       ;    r1 - i-number of i-node
20449                                  <1>       ;
20450                                  <1>       ; OUTPUTS ->
20451                                  <1>       ;    inode r/w
20452                                  <1>       ;
20453                                  <1>       ; ((AX = R1)) input
20454                                  <1>       ;
20455                                  <1>       ;  (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)
20456                                  <1>         ;  ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))
20457                                  <1>       ;
20458 00005776 0FB7D0                  <1>       movzx  edx, ax
20459 00005779 6683C22F                <1>       add    dx, 47
20460 0000577D 89D0                    <1>       mov    eax, edx
20461                                  <1>       ;add   ax, 47 ; add 47 to inode number
20462                                  <1>              ;  add $31.,r1 / add 31. to i-number
20463 0000577F 50                      <1>       push   eax
20464                                  <1>              ; mov r1,-(sp) / save i+31. on stack
20465 00005780 66C1E804                <1>       shr    ax, 4
20466                                  <1>              ; asr r1 / divide by 16.
20467                                  <1>              ; asr r1
20468                                  <1>              ; asr r1
20469                                  <1>              ; asr r1 / r1 contains block number of block
20470                                  <1>                    ; / in which i-node exists
20471 00005784 E82A0A0000              <1>       call   dskrd
20472                                  <1>              ; jsr r0,dskrd / read in block containing i-node i.
20473                                  <1>       ; 31/07/2013
20474 00005789 803D[CC740000]00        <1>         cmp     byte [rw], 0 ; Retro Unix 8086 v1 feature !
20475                                  <1>       ;; 28/07/2013 rw -> u.rw
20476                                  <1>         ;;cmp     byte [u.rw], 0 ; Retro Unix 8086 v1 feature !
20477                                  <1>              ; tst (r0)
20478 00005790 7605                    <1>       jna    short icalc_1
20479                                  <1>              ; beq 1f / branch to wslot when argument
20480                                  <1>                    ; / in icalc call = 1
20481                                  <1>       ; eAX = r1  = block number
20482 00005792 E87C0A0000              <1>       call   wslot
20483                                  <1>              ; jsr r0,wslot / set up data buffer for write
20484                                  <1>                           ; / (will be same buffer as dskrd got)
20485                                  <1>       ; eBX = r5 points to first word in data area for this block
20486                                  <1> icalc_1: ; 1:
20487 00005797 5A                      <1>       pop    edx
20488 00005798 83E20F                  <1>       and    edx, 0Fh ; (i+47) mod 16
20489                                  <1>              ; bic $!17,(sp) / zero all but last 4 bits;
20490                                  <1>                     ; / gives (i+31.) mod 16
20491 0000579B C1E205                  <1>       shl    edx, 5
20492                                  <1>       ; eDX = 32 * ((i+47) mod 16)
20493 0000579E 89DE                    <1>       mov    esi, ebx  ; ebx points 1st word of the buffer
20494 000057A0 01D6                    <1>       add    esi, edx  ; edx is inode offset in the buffer
20495                                  <1>              ; eSI (r5) points to first word in i-node i.
20496                                  <1>              ; mov (sp)+,mq / calculate offset in data buffer;
20497                                  <1>                           ; / 32.*(i+31.)mod16
20498                                  <1>              ; mov $5,lsh / for i-node i.
20499                                  <1>              ; add mq,r5 / r5 points to first word in i-node i.
20500 000057A2 BF[16710000]            <1>       mov    edi, inode
20501                                  <1>              ; mov $inode,r1 / inode is address of first word
20502                                  <1>                           ; / of current i-node
20503 000057A7 B908000000              <1>       mov    ecx, 8 ; 02/07/2015(32 bit modification)
20504                                  <1>              ; mov $16.,r3
20505                                  <1>         ; 31/07/2013
20506 000057AC 382D[CC740000]          <1>       cmp     [rw], ch ; 0  ;; Retro Unix 8086 v1 feature !
20507                                  <1>         ;;28/07/2013 rw -> u.rw
20508                                  <1>         ;;cmp    [u.rw], ch ; 0  ;; Retro Unix 8086 v1 feature !
20509                                  <1>              ; tst (r0)+ / branch to 2f when argument in icalc call = 0
20510 000057B2 760A                    <1>       jna    short icalc_3
20511                                  <1>              ; beq 2f / r0 now contains proper return address
20512                                  <1>                    ; / for rts r0
20513                                  <1> icalc_2: ; 1:
20514 000057B4 87F7                    <1>       xchg   esi, edi
20515                                  <1>       ; overwrite old i-node (in buffer to be written)
20516 000057B6 F3A5                    <1>       rep    movsd
20517                                  <1>              ; mov (r1)+,(r5)+ / over write old i-node
20518                                  <1>              ; dec r3
20519                                  <1>              ; bgt 1b
20520 000057B8 E8720A0000              <1>       call   dskwr
20521                                  <1>              ; jsr r0,dskwr / write inode out on device
20522 000057BD C3                      <1>       retn
20523                                  <1>              ; rts r0
20524                                  <1> icalc_3: ; 2:
20525                                  <1>       ; copy new i-node into inode area of (core) memory
20526 000057BE F3A5                    <1>       rep    movsd
20527                                  <1>              ; mov (r5)+,(r1)+ / read new i-node into
20528                                  <1>                                ; / "inode" area of core
20529                                  <1>              ; dec r3
20530                                  <1>              ; bgt 2b
20531 000057C0 C3                      <1>       retn
20532                                  <1>              ; rts r0
20533                                  <1>
20534                                  <1> access:
20535                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
```

```
20536                                  <1>       ; 24/04/2013 - 29/04/2013 (Retro UNIX 8086 v1)
20537                                  <1>       ;
20538                                  <1>       ; check whether user is owner of file or user has read or write
20539                                  <1>       ; permission (based on i.flgs).
20540                                  <1>       ;
20541                                  <1>       ; INPUTS ->
20542                                  <1>       ;    r1 - i-number of file
20543                                  <1>       ;    u.uid
20544                                  <1>       ; arg0 -> (owner flag mask)
20545                                  <1>       ;     Retro UNIX 8086 v1 feature -> owner flag mask in DL (DX)
20546                                  <1>       ; OUTPUTS ->
20547                                  <1>       ;    inode (or jump to error)
20548                                  <1>       ;
20549                                  <1>       ; ((AX = R1)) input/output
20550                                  <1>       ;
20551                                  <1>         ;  ((Modified registers: eCX, eBX, eDX, eSI, eDI, eBP))
20552                                  <1>       ;
20553 000057C1 6652                   <1>       push   dx  ; save flags (DL)
20554 000057C3 E81EFFFFFF             <1>       call   iget
20555                                  <1>              ; jsr r0,iget / read in i-node for current directory
20556                                  <1>                        ; / (i-number passed in r1)
20557 000057C8 8A0D[16710000]         <1>       mov    cl, [i.flgs]
20558                                  <1>              ; mov i.flgs,r2
20559 000057CE 665A                   <1>       pop    dx  ; restore flags (DL)
20560 000057D0 8A35[94740000]         <1>       mov    dh, [u.uid]
20561 000057D6 3A35[19710000]         <1>       cmp    dh, [i.uid]
20562                                  <1>              ; cmpb i.uid,u.uid / is user same as owner of file
20563 000057DC 7503                   <1>       jne    short access_1
20564                                  <1>              ; bne 1f / no, then branch
20565 000057DE C0E902                 <1>       shr    cl, 2
20566                                  <1>              ; asrb r2 / shift owner read write bits into non owner
20567                                  <1>                      ; / read/write bits
20568                                  <1>              ; asrb r2
20569                                  <1> access_1: ; 1:
20570 000057E1 20D1                   <1>       and    cl, dl
20571                                  <1>              ; bit r2,(r0)+ / test read-write flags against argument
20572                                  <1>                          ; / in access call
20573 000057E3 7513                   <1>       jnz    short access_2
20574                                  <1>              ; bne 1f
20575 000057E5 08F6                   <1>       or     dh, dh ; super user  (root) ?
20576                                  <1>              ; tstb u.uid
20577 000057E7 740F                   <1>       jz     short access_2 ; yes, super user
20578                                  <1>       ;jnz   error
20579                                  <1>              ; beq 1f
20580                                  <1>              ; jmp error
20581 000057E9 C705[9D740000]0B00-    <1>       mov    dword [u.error], ERR_FILE_ACCESS
20582 000057F1 0000                   <1>
20583                                  <1>                     ; 'permission denied !' error
20584 000057F3 E942E8FFFF             <1>       jmp    error
20585                                  <1>
20586                                  <1> access_2: ; 1:
20587                                  <1>       ; DL = flags
20588 000057F8 C3                     <1>       retn
20589                                  <1>              ; rts r0
20590                                  <1>
20591                                  <1> setimod:
20592                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20593                                  <1>       ; 09/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
20594                                  <1>       ;
20595                                  <1>       ; 'setimod' sets byte at location 'imod' to 1; thus indicating that
20596                                  <1>       ; the inode has been modified. Also puts the time of modification
20597                                  <1>       ; into the inode.
20598                                  <1>       ;
20599                                  <1>       ;  (Retro UNIX Prototype : 14/07/2012 - 23/02/2013, UNIXCOPY.ASM)
20600                                  <1>         ;  ((Modified registers: eDX, eCX, eBX))
20601                                  <1>       ;
20602                                  <1>
20603                                  <1>       ; push       edx
20604 000057F9 50                     <1>       push   eax
20605                                  <1>
20606 000057FA C605[3C740000]01       <1>       mov    byte [imod], 1
20607                                  <1>              ; movb $1,imod / set current i-node modified bytes
20608                                  <1>       ; Erdogan Tan 14-7-2012
20609 00005801 E86FE3FFFF             <1>       call   epoch
20610                                  <1>              ; mov s.time,i.mtim
20611                                  <1>                       ; / put present time into file modified time
20612                                  <1>              ; mov s.time+2,i.mtim+2
20613                                  <1>
20614 00005806 A3[30710000]           <1>       mov    [i.mtim], eax
20615                                  <1>
20616                                  <1>       ; Retro UNIX 386 v1 modification ! (cmp)
20617                                  <1>       ; Retro UNIX 8086 v1 modification ! (test)
20618 0000580B 833D[2C710000]00       <1>       cmp    dword [i.ctim], 0
20619 00005812 7505                   <1>       jnz    short setimod_ok
20620                                  <1>
20621 00005814 A3[2C710000]           <1>       mov    [i.ctim], eax
20622                                  <1>
20623                                  <1> setimod_ok: ; 31/07/2013
20624 00005819 58                     <1>       pop    eax
20625                                  <1>       ;pop   edx
20626                                  <1>
20627 0000581A C3                     <1>       retn
20628                                  <1>              ; rts r0
20629                                  <1>
20630                                  <1> itrunc:
20631                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20632                                  <1>       ; 23/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20633                                  <1>       ;
20634                                  <1>       ; 'itrunc' truncates a file whose i-number is given in r1
20635                                  <1>       ;  to zero length.
20636                                  <1>       ;
20637                                  <1>       ; INPUTS ->
20638                                  <1>       ;    r1 - i-number of i-node
20639                                  <1>       ;    i.dskp - pointer to contents or indirect block in an i-node
20640                                  <1>       ;    i.flgs - large file flag
```

```
20641                                  <1>       ;    i.size - size of file
20642                                  <1>       ;
20643                                  <1>       ; OUTPUTS ->
20644                                  <1>       ;    i.flgs - large file flag is cleared
20645                                  <1>       ;    i.size - set to 0
20646                                  <1>       ;    i.dskp .. i.dskp+16 - entire list is cleared
20647                                  <1>       ;    setimod - set to indicate i-node has been modified
20648                                  <1>       ;    r1 - i-number of i-node
20649                                  <1>       ;
20650                                  <1>       ; ((AX = R1)) input/output
20651                                  <1>       ;
20652                                  <1>       ;  (Retro UNIX Prototype : 01/12/2012 - 10/03/2013, UNIXCOPY.ASM)
20653                                  <1>         ;  ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
20654                                  <1>
20655 0000581B E8C6FEFFFF             <1>       call  iget
20656                                  <1>             ; jsr r0,iget
20657 00005820 BE[1C710000]           <1>       mov   esi, i.dskp
20658                                  <1>             ; mov $i.dskp,r2 / address of block pointers in r2
20659 00005825 31C0                   <1>       xor   eax, eax
20660                                  <1> itrunc_1: ; 1:
20661 00005827 66AD                   <1>       lodsw
20662                                  <1>             ; mov (r2)+,r1 / move physical block number into r1
20663 00005829 6609C0                 <1>       or    ax, ax
20664 0000582C 743B                   <1>       jz    short itrunc_5
20665                                  <1>             ; beq 5f
20666 0000582E 56                     <1>       push  esi
20667                                  <1>             ; mov r2,-(sp)
20668 0000582F 66F705[16710000]00-    <1>       test   word [i.flgs], 1000h
20669 00005837 10                     <1>
20670                                  <1>             ; bit $10000,i.flgs / test large file bit?
20671 00005838 7429                   <1>       jz    short itrunc_4
20672                                  <1>             ; beq 4f / if clear, branch
20673 0000583A 50                     <1>       push  eax
20674                                  <1>             ; mov r1,-(sp) / save block number of indirect block
20675 0000583B E873090000             <1>       call  dskrd
20676                                  <1>             ; jsr r0,dskrd / read in block, 1st data word
20677                                  <1>                         ; / pointed to by r5
20678                                  <1>       ; eBX = r5 = Buffer data address (the 1st word)
20679 00005840 B900010000             <1>       mov   ecx, 256
20680                                  <1>             ; mov $256.,r3 / move word count into r3
20681 00005845 89DE                   <1>       mov   esi, ebx
20682                                  <1> itrunc_2: ; 2:
20683 00005847 66AD                   <1>       lodsw
20684                                  <1>             ; mov (r5)+,r1 / put 1st data word in r1;
20685                                  <1>                         ; / physical block number
20686 00005849 6621C0                 <1>       and   ax, ax
20687 0000584C 7409                   <1>       jz    short itrunc_3
20688                                  <1>             ; beq 3f / branch if zero
20689                                  <1>       ;push ecx
20690 0000584E 6651                   <1>       push  cx
20691                                  <1>             ; mov r3,-(sp) / save r3, r5 on stack
20692                                  <1>       ;push esi
20693                                  <1>             ; mov r5,-(sp)
20694 00005850 E842FEFFFF             <1>       call  free
20695                                  <1>             ; jsr r0,free / free block in free storage map
20696                                  <1>       ;pop  esi
20697                                  <1>             ; mov(sp)+,r5
20698 00005855 6659                   <1>       pop   cx
20699                                  <1>       ;pop  ecx
20700                                  <1>             ; mov (sp)+,r3
20701                                  <1> itrunc_3: ; 3:
20702 00005857 E2EE                   <1>       loop  itrunc_2
20703                                  <1>             ; dec r3 / decrement word count
20704                                  <1>             ; bgt 2b / branch if positive
20705 00005859 58                     <1>       pop   eax
20706                                  <1>             ; mov (sp)+,r1 / put physical block number of
20707                                  <1>                         ; / indirect block
20708                                  <1>       ; 01/08/2013
20709 0000585A 668125[16710000]FF-    <1>         and     word [i.flgs], 0EFFFh ; 1110111111111111b
20710 00005862 EF                     <1>
20711                                  <1> itrunc_4: ; 4:
20712 00005863 E82FFEFFFF             <1>       call  free
20713                                  <1>             ; jsr r0,free / free indirect block
20714 00005868 5E                     <1>       pop   esi
20715                                  <1>             ; mov (sp)+,r2
20716                                  <1> itrunc_5: ; 5:
20717 00005869 81FE[2C710000]         <1>       cmp   esi, i.dskp+16
20718                                  <1>             ; cmp r2,$i.dskp+16.
20719 0000586F 72B6                   <1>       jb    short itrunc_1
20720                                  <1>             ; bne 1b / branch until all i.dskp entries check
20721                                  <1>       ; 01/08/2013
20722                                  <1>       ;and     word [i.flgs], 0EFFFh ; 1110111111111111b
20723                                  <1>             ; bic $10000,i.flgs / clear large file bit
20724 00005871 BF[1C710000]           <1>       mov   edi, i.dskp
20725 00005876 66B90800               <1>       mov   cx, 8
20726 0000587A 6631C0                 <1>       xor   ax, ax
20727 0000587D 66A3[1A710000]         <1>       mov   [i.size], ax ; 0
20728                                  <1>             ; clr i.size / zero file size
20729 00005883 F366AB                 <1>       rep   stosw
20730                                  <1>             ; jsr r0,copyz; i.dskp; i.dskp+16.
20731                                  <1>                       ; / zero block pointers
20732 00005886 E86EFFFFFF             <1>       call  setimod
20733                                  <1>             ; jsr r0,setimod / set i-node modified flag
20734 0000588B 66A1[2A740000]         <1>       mov   ax, [ii]
20735                                  <1>             ; mov ii,r1
20736 00005891 C3                     <1>       retn
20737                                  <1>             ; rts r0
20738                                  <1>
20739                                  <1> imap:
20740                                  <1>       ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20741                                  <1>       ; 26/04/2013 (Retro UNIX 8086 v1)
20742                                  <1>       ;
20743                                  <1>       ; 'imap' finds the byte in core (superblock) containing
20744                                  <1>       ; allocation bit for an i-node whose number in r1.
20745                                  <1>       ;
```

```
20746                                  <1>       ; INPUTS ->
20747                                  <1>       ;    r1 - contains an i-number
20748                                  <1>       ;    fsp - start of table containing open files
20749                                  <1>       ;
20750                                  <1>       ; OUTPUTS ->
20751                                  <1>       ;    r2 - byte address of byte with the allocation bit
20752                                  <1>       ;    mq - a mask to locate the bit position.
20753                                  <1>       ;        (a 1 is in calculated bit posisiton)
20754                                  <1>       ;
20755                                  <1>       ; ((AX = R1)) input/output
20756                                  <1>       ; ((DL/DX = MQ)) output
20757                                  <1>       ; ((BX = R2)) output
20758                                  <1>       ;
20759                                  <1>       ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
20760                                  <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI))
20761                                  <1>       ;
20762                                  <1>             ; / get the byte that has the allocation bit for
20763                                  <1>             ; / the i-number contained in r1
20764                                  <1>       ;mov   dx, 1
20765 00005892 B201                   <1>       mov    dl, 1
20766                                  <1>             ; mov $1,mq / put 1 in the mq
20767 00005894 0FB7D8                 <1>       movzx  ebx, ax
20768                                  <1>             ; mov r1,r2 / r2 now has i-number whose byte
20769                                  <1>                         ; / in the map we must find
20770 00005897 6683EB29               <1>       sub    bx, 41
20771                                  <1>             ; sub $41.,r2 / r2 has i-41
20772 0000589B 88D9                   <1>       mov    cl, bl
20773                                  <1>             ; mov r2,r3 / r3 has i-41
20774 0000589D 80E107                 <1>       and    cl, 7
20775                                  <1>             ; bic $!7,r3 / r3 has (i-41) mod 8 to get
20776                                  <1>                       ; / the bit position
20777 000058A0 7402                   <1>       jz     short imap1
20778                                  <1>       ;shl   dx, cl
20779 000058A2 D2E2                   <1>       shl    dl, cl
20780                                  <1>             ; mov r3,lsh / move the 1 over (i-41) mod 8 positions
20781                                  <1> imap1:                     ; / to the left to mask the correct bit
20782 000058A4 66C1EB03               <1>       shr    bx, 3
20783                                  <1>             ; asr r2
20784                                  <1>             ; asr r2
20785                                  <1>             ; asr r2 / r2 has (i-41) base 8 of the byte number
20786                                  <1>                    ; / from the start of the map
20787                                  <1>             ; mov r2,-(sp) / put (i-41) base 8 on the stack
20788 000058A8 BE[08810000]           <1>       mov    esi, systm
20789                                  <1>             ; mov $systm,r2 / r2 points to the in-core image of
20790                                  <1>                           ; / the super block for drum
20791                                  <1>       ;cmp   word [cdev], 0
20792 000058AD 803D[2E740000]00       <1>       cmp    byte [cdev], 0
20793                                  <1>             ; tst cdev / is the device the disk
20794 000058B4 7606                   <1>       jna    short imap2
20795                                  <1>             ; beq 1f / yes
20796 000058B6 81C608020000           <1>       add    esi, mount - systm
20797                                  <1>             ; add $mount-systm,r2 / for mounted device,
20798                                  <1>                    ; / r2 points to 1st word of its super block
20799                                  <1> imap2: ; 1:
20800 000058BC 66031E                 <1>       add    bx, [esi] ;; add free map size to si
20801                                  <1>             ; add (r2)+,(sp) / get byte address of allocation bit
20802 000058BF 6683C304               <1>       add    bx, 4
20803 000058C3 01F3                   <1>       add    ebx, esi
20804                                  <1>             ; add (sp)+,r2 / ?
20805                                  <1>       ;add   ebx, 4 ;; inode map offset in superblock
20806                                  <1>                    ;; (2 + free map size + 2)
20807                                  <1>             ; add $2,r2 / ?
20808                                  <1>       ; DL/DX (MQ) has a 1 in the calculated bit position
20809                                  <1>         ; BX (R2) has byte address of the byte with allocation bit
20810 000058C5 C3                     <1>       retn
20811                                  <1>             ; rts r0
20812                                      %include 'u6.s'        ; 31/05/2015
20813                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS6.INC
20814                                  <1> ; Last Modification: 18/11/2015
20815                                  <1> ; -----------------------------------------------------------------------------
20816                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
20817                                  <1> ; (v0.1 - Beginning: 11/07/2012)
20818                                  <1> ;
20819                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
20820                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
20821                                  <1> ; <Bell Laboratories (17/3/1972)>
20822                                  <1> ; <Preliminary Release of UNIX Implementation Document>
20823                                  <1> ;
20824                                  <1> ; Retro UNIX 8086 v1 - U6.ASM (23/07/2014) //// UNIX v1 -> u6.s
20825                                  <1> ;
20826                                  <1> ; ***************************************************************************
20827                                  <1>
20828                                  <1> readi:
20829                                  <1>       ; 20/05/2015
20830                                  <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20831                                  <1>       ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
20832                                  <1>       ;
20833                                  <1>       ; Reads from an inode whose number in R1
20834                                  <1>       ;
20835                                  <1>       ; INPUTS ->
20836                                  <1>       ;    r1 - inode number
20837                                  <1>       ;    u.count - byte count user desires
20838                                  <1>       ;    u.base - points to user buffer
20839                                  <1>       ;    u.fofp - points to word with current file offset
20840                                  <1>       ; OUTPUTS ->
20841                                  <1>       ;    u.count - cleared
20842                                  <1>       ;    u.nread - accumulates total bytes passed back
20843                                  <1>       ;
20844                                  <1>       ; ((AX = R1)) input/output
20845                                  <1>       ;    (Retro UNIX Prototype : 01/03/2013 - 14/12/2012, UNIXCOPY.ASM)
20846                                  <1>         ;    ((Modified registers: edx, ebx, ecx, esi, esi, ebp))
20847                                  <1>
20848 000058C6 31D2                   <1>       xor    edx, edx ; 0
20849 000058C8 8915[70740000]         <1>       mov    [u.nread], edx ; 0
20850                                  <1>              ; clr u.nread / accumulates number of bytes transmitted
```

```
20851 000058CE 668915[AD740000]    <1>       mov    [u.pcount], dx ; 19/05/2015
20852 000058D5 3915[6C740000]      <1>       cmp    [u.count], edx ; 0
20853                              <1>                ; tst u.count / is number of bytes to be read greater than 0
20854 000058DB 7701                <1>       ja     short readi_1 ; 1f
20855                              <1>                ; bgt 1f / yes, branch
20856 000058DD C3                  <1>       retn
20857                              <1>                ; rts r0 / no, nothing to read; return to caller
20858                              <1> readi_1: ; 1:
20859                              <1>                  ; mov r1,-(sp) / save i-number on stack
20860 000058DE 6683F828            <1>       cmp    ax, 40
20861                              <1>                ; cmp r1,$40. / want to read a special file
20862                              <1>                ;             / (i-nodes 1,...,40 are for special files)
20863 000058E2 0F87D3000000        <1>         ja      dskr
20864                              <1>                ; ble 1f / yes, branch
20865                              <1>                ; jmp dskr / no, jmp to dskr;
20866                              <1>                ;         / read file with i-node number (r1)
20867                              <1>                ;    / starting at byte ((u.fofp)), read in u.count bytes
20868                              <1>       ; (20/05/2015)
20869 000058E8 50                  <1>       push   eax ; because subroutines will jump to 'ret_'
20870                              <1>       ; 1:
20871 000058E9 0FB6D8              <1>       movzx  ebx, al
20872 000058EC 66C1E302            <1>       shl    bx, 2
20873                              <1>                ; asl r1 / multiply inode number by 2
20874 000058F0 81C3[F4580000]      <1>       add    ebx, readi_2 - 4
20875 000058F6 FF23                <1>       jmp    dword [ebx]
20876                              <1>                ; jmp *1f-2(r1)
20877                              <1> readi_2: ; 1:
20878 000058F8 [44590000]          <1>       dd     rtty ; tty, AX = 1 (runix)
20879                              <1>                ;rtty / tty; r1=2
20880                              <1>                ;rppt / ppt; r1=4
20881 000058FC [97590000]          <1>       dd     rmem ; mem, AX = 2 (runix)
20882                              <1>                ;rmem / mem; r1=6
20883                              <1>                ;rrf0 / rf0
20884                              <1>                ;rrk0 / rk0
20885                              <1>                ;rtap / tap0
20886                              <1>                ;rtap / tap1
20887                              <1>                ;rtap / tap2
20888                              <1>                ;rtap / tap3
20889                              <1>                ;rtap / tap4
20890                              <1>                ;rtap / tap5
20891                              <1>                ;rtap / tap6
20892                              <1>                ;rtap / tap7
20893 00005900 [59600000]         <1>       dd     rfd ; fd0, AX = 3 (runix only)
20894 00005904 [59600000]         <1>       dd     rfd ; fd1, AX = 4 (runix only)
20895 00005908 [59600000]         <1>       dd     rhd ; hd0, AX = 5 (runix only)
20896 0000590C [59600000]         <1>       dd     rhd ; hd1, AX = 6 (runix only)
20897 00005910 [59600000]         <1>       dd     rhd ; hd2, AX = 7 (runix only)
20898 00005914 [59600000]         <1>       dd     rhd ; hd3, AX = 8 (runix only)
20899 00005918 [AC590000]         <1>       dd     rlpr ; lpr, AX = 9 (invalid, write only device !?)
20900 0000591C [93590000]         <1>       dd     rcvt ; tty0, AX = 10 (runix)
20901                             <1>                ;rcvt / tty0
20902 00005920 [93590000]         <1>       dd     rcvt ; tty1, AX = 11 (runix)
20903                             <1>                ;rcvt / tty1
20904 00005924 [93590000]         <1>       dd     rcvt ; tty2, AX = 12 (runix)
20905                             <1>                ;rcvt / tty2
20906 00005928 [93590000]         <1>       dd     rcvt ; tty3, AX = 13 (runix)
20907                             <1>                ;rcvt / tty3
20908 0000592C [93590000]         <1>       dd     rcvt ; tty4, AX = 14 (runix)
20909                             <1>                ;rcvt / tty4
20910 00005930 [93590000]         <1>       dd     rcvt ; tty5, AX = 15 (runix)
20911                             <1>                ;rcvt / tty5
20912 00005934 [93590000]         <1>       dd     rcvt ; tty6, AX = 16 (runix)
20913                             <1>                ;rcvt / tty6
20914 00005938 [93590000]         <1>       dd     rcvt ; tty7, AX = 17 (runix)
20915                             <1>                ;rcvt / tty7
20916 0000593C [93590000]         <1>       dd     rcvt ; COM1, AX = 18 (runix only)
20917                             <1>                ;rcrd / crd
20918 00005940 [93590000]         <1>       dd     rcvt ; COM2, AX = 19 (runix only)
20919                             <1>
20920                             <1> rtty: ; / read from console tty
20921                             <1>       ; 17/10/2015 - 16/07/2015 (Retro UNIX 8086 v1)
20922                             <1>       ;          (Only 1 byte is read, by ignoring byte count!)
20923                             <1>       ;          WHAT FOR: Every character from Keyboard input
20924                             <1>       ;          must be written immediate on video page (screen)
20925                             <1>       ;          when it is required.
20926                             <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20927                             <1>       ; 11/03/2013 - 19/06/2014 (Retro UNIX 8086 v1)
20928                             <1>       ;
20929                             <1>       ; Console tty buffer is PC keyboard buffer
20930                             <1>       ; and keyboard-keystroke handling is different than original
20931                             <1>       ; unix (PDP-11) here. TTY/Keyboard procedures here are changed
20932                             <1>       ; according to IBM PC compatible ROM BIOS keyboard functions.
20933                             <1>       ;
20934                             <1>       ; 06/12/2013
20935 00005944 0FB61D[97740000]   <1>       movzx  ebx, byte [u.uno] ; process number
20936 0000594B 8A83[95710000]     <1>       mov    al, [ebx+p.ttyc-1] ; current/console tty
20937                             <1> rttys:
20938                             <1>              ; mov tty+[8*ntty]-8+6,r5 / r5 is the address of the 4th word of
20939                             <1>                      ; / of the control and status block
20940                             <1>              ; tst 2(r5) / for the console tty; this word points to the console
20941                             <1>                     ; / tty buffer
20942                             <1>       ; 28/07/2013
20943 00005951 A2[9C740000]       <1>       mov    [u.ttyn], al
20944                             <1>       ; 13/01/2014
20945 00005956 FEC0               <1>       inc    al
20946 00005958 A2[78740000]       <1>       mov    [u.ttyp], al ; tty number + 1
20947                             <1> rtty_nc: ; 01/02/2014
20948                             <1>       ; 29/09/2013
20949 0000595D B90A000000         <1>       mov    ecx, 10
20950                             <1> rtty_1:     ; 01/02/2014
20951 00005962 6651               <1>       push   cx ; 29/09/2013
20952                             <1>       ; byte [u.ttyn] = tty number (0 to 9)
20953 00005964 B001               <1>       mov    al, 1
20954 00005966 E8330B0000         <1>       call   getc
20955 0000596B 6659               <1>       pop    cx ; 29/09/2013
```

```
20956 0000596D 7516                <1>       jnz    short rtty_2
20957                              <1>              ; bne 1f / 2nd word of console tty buffer contains number
20958                              <1>                       ; / of chars. Is this number non-zero?
20959 0000596F E20D                <1>       loop   rtty_idle ; 01/02/2014
20960                              <1>       ; 05/10/2013
20961 00005971 8A25[9C740000]      <1>       mov    ah, [u.ttyn]
20962                              <1>       ; 29/09/2013
20963 00005977 E871FBFFFF          <1>       call   sleep
20964                              <1>              ; jsr r0,canon; ttych / if 0, call 'canon' to get a line
20965                              <1>                 ;          / (120 chars.)
20966                              <1>       ;byte [u.ttyn] = tty number (0 to 9)
20967 0000597C EBDF                <1>       jmp    short rtty_nc ; 01/02/2014
20968                              <1>
20969                              <1> rtty_idle:
20970                              <1>       ; 29/07/2013
20971 0000597E E8D6FAFFFF          <1>       call   idle
20972 00005983 EBDD                <1>       jmp    short rtty_1 ; 01/02/2014
20973                              <1>       ;1:
20974                              <1>              ; tst 2(r5) / is the number of characters zero
20975                              <1>              ; beq ret1 / yes, return to caller via 'ret1'
20976                              <1>              ; movb *4(r5),r1 / no, put character in r1
20977                              <1>              ; inc 4(r5) / 3rd word of console tty buffer points to byte which
20978                              <1>                        ; / contains the next char.
20979                              <1>              ; dec 2(r5) / decrement the character count
20980                              <1> rtty_2:
20981 00005985 30C0                <1>       xor    al, al
20982 00005987 E8120B0000          <1>       call   getc
20983 0000598C E892000000          <1>       call   passc
20984                              <1>              ; jsr r0,passc / move the character to core (user)
20985                              <1>       ;; 17/10/2015 - 16/07/2015
20986                              <1>       ; 19/06/2014
20987                              <1>       ;;jnz  short rtty_nc
20988 00005991 58                  <1>       pop    eax  ; (20/05/2015)
20989 00005992 C3                  <1>       retn
20990                              <1> ;ret1:
20991                              <1>              ; jmp ret / return to caller via 'ret'
20992                              <1>
20993                              <1> rcvt:   ; < receive/read character from tty >
20994                              <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20995                              <1>       ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
20996                              <1>       ;
20997                              <1>       ; Retro UNIX 8086 v1 modification !
20998                              <1>       ;
20999                              <1>       ; In original UNIX v1, 'rcvt' routine
21000                              <1>       ;            (exactly different than this one)
21001                              <1>       ;     was in 'u9.s' file.
21002                              <1>       ;
21003 00005993 2C0A                <1>       sub    al, 10
21004                              <1>       ; AL = tty number (0 to 9), (COM1=8, COM2=9)
21005                              <1>       ; 16/07/2013
21006                              <1>       ; 21/05/2013
21007 00005995 EBBA                <1>         jmp     short rttys
21008                              <1>
21009                              <1> ;rppt: / read paper tape
21010                              <1> ;     jsr    r0,pptic / gets next character in clist for ppt input and
21011                              <1> ;                     / places
21012                              <1> ;            br ret / it in r1; if there 1s no problem with reader, it
21013                              <1> ;                     / also enables read bit in prs
21014                              <1> ;     jsr    r0,passc / place character in users buffer area
21015                              <1> ;     br     rppt
21016                              <1>
21017                              <1> rmem: ; / transfer characters from memory to a user area of core
21018                              <1>       ; 17/10/2015
21019                              <1>       ; 11/06/2015
21020                              <1>       ; 24/05/2015
21021                              <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21022                              <1>       ;
21023 00005997 8B35[58740000]      <1>       mov     esi, [u.fofp]
21024                              <1> rmem_1:
21025 0000599D 8B1E                <1>         mov     ebx, [esi]
21026                              <1>              ; mov *u.fofp,r1 / save file offset which points to the char
21027                              <1>                               ; / to be transferred to user
21028 0000599F FF06                <1>         inc     dword [esi] ; 17/10/2015
21029                              <1>             ; inc *u.fofp / increment file offset to point to 'next'
21030                              <1>                             ; / char in memory file
21031 000059A1 8A03                <1>       mov    al, [ebx]
21032                              <1>              ; movb (r1),r1 / get character from memory file,
21033                              <1>                             ; / put it in r1
21034 000059A3 E87B000000          <1>       call   passc          ; jsr r0,passc / move this character to
21035                              <1>                            ;  / the next byte of the users core area
21036                              <1>              ; br rmem / continue
21037 000059A8 75F3                <1>       jnz    short rmem_1
21038                              <1> ret_:
21039 000059AA 58                  <1>       pop    eax ; 09/06/2015
21040 000059AB C3                  <1>       retn
21041                              <1>
21042                              <1> rlpr:
21043                              <1> ;1:
21044                              <1> ;rcrd:
21045 000059AC C705[9D740000]0F00- <1>         mov     dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
21046 000059B4 0000                <1>
21047 000059B6 E97FE6FFFF          <1>       jmp    error
21048                              <1>              ;jmp   error / see 'error' routine
21049                              <1>
21050                              <1> dskr:
21051                              <1>       ; 12/10/2015
21052                              <1>       ; 21/08/2015
21053                              <1>       ; 25/07/2015
21054                              <1>       ; 10/07/2015
21055                              <1>       ; 16/06/2015
21056                              <1>       ; 31/05/2015
21057                              <1>       ; 24/05/2015 (Retro UNIX 386 v1 - Beginning)
21058                              <1>       ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
21059                              <1> dskr_0:
21060 000059BB 50                  <1>       push   eax
```

```
21061                                  <1>             ; mov (sp),r1 / i-number in r1
21062                                  <1>       ; AX = i-number
21063 000059BC E825FDFFFF            <1>       call  iget
21064                                  <1>             ; jsr r0,iget / get i-node (r1) into i-node section of core
21065 000059C1 0FB715[1A710000]      <1>         movzx   edx, word [i.size] ; 16/06/2015
21066                                  <1>             ; mov i.size,r2 / file size in bytes in r2
21067 000059C8 8B1D[58740000]        <1>       mov   ebx, [u.fofp]
21068 000059CE 2B13                  <1>       sub   edx, [ebx]
21069                                  <1>             ; sub *u.fofp,r2 / subtract file offset
21070                                  <1>         ; 12/10/2015
21071                                  <1>       ; jna     short ret_
21072                                  <1>             ; blos ret
21073 000059D0 7709                  <1>       ja    short dskr_1
21074                                  <1>       ;
21075                                  <1> dskr_retn: ; 12/10/2015
21076 000059D2 58                    <1>       pop   eax
21077 000059D3 C605[AF740000]00      <1>       mov   byte [u.kcall], 0
21078 000059DA C3                    <1>       retn
21079                                  <1> dskr_1:
21080 000059DB 3B15[6C740000]        <1>       cmp     edx, [u.count]
21081                                  <1>             ; cmp r2,u.count / are enough bytes left in file
21082                                  <1>                             ; / to carry out read
21083 000059E1 7306                  <1>       jnb   short dskr_2
21084                                  <1>             ; bhis 1f
21085 000059E3 8915[6C740000]        <1>       mov   [u.count], edx
21086                                  <1>             ; mov r2,u.count / no, just read to end of file
21087                                  <1> dskr_2: ; 1:
21088                                  <1>       ; AX = i-number
21089 000059E9 E885FBFFFF            <1>       call  mget
21090                                  <1>             ; jsr r0,mget / returns physical block number of block
21091                                  <1>                           ; / in file where offset points
21092                                  <1>       ; eAX = physical block number
21093 000059EE E8C0070000            <1>       call  dskrd
21094                                  <1>             ; jsr r0,dskrd / read in block, r5 points to
21095                                  <1>                            ; / 1st word of data in buffer
21096                                  <1>       ; 09/06/2015
21097 000059F3 803D[AF740000]00      <1>       cmp   byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
21098 000059FA 770F                  <1>       ja    short dskr_4   ; zf=0 -> the caller is 'namei'
21099 000059FC 66833D[AD740000]00    <1>       cmp   word [u.pcount], 0
21100 00005A04 7705                  <1>       ja    short dskr_4
21101                                  <1> dskr_3:
21102                                  <1>       ; [u.base] = virtual address to transfer (as destination address)
21103 00005A06 E853000000            <1>       call  trans_addr_w ; translate virtual address to physical (w)
21104                                  <1> dskr_4:
21105                                  <1>       ; eBX (r5) = system (I/O) buffer address -physical-
21106 00005A0B E8C7020000            <1>       call  sioreg
21107                                  <1>             ; jsr r0,sioreg
21108 00005A10 87F7                  <1>       xchg  esi, edi
21109                                  <1>       ; eDI = file (user data) offset
21110                                  <1>       ; eSI = sector (I/O) buffer offset
21111                                  <1>       ; eCX = byte count
21112 00005A12 F3A4                  <1>       rep   movsb
21113                                  <1>             ; movb (r2)+,(r1)+ / move data from buffer into working core
21114                                  <1>                                ; / starting at u.base
21115                                  <1>             ; dec r3
21116                                  <1>             ; bne 2b / branch until proper number of bytes are transferred
21117                                  <1>       ; 25/07/2015
21118                                  <1>       ; eax = remain bytes in buffer
21119                                  <1>         ;       (check if remain bytes in the buffer > [u.pcount])
21120 00005A14 09C0                  <1>       or    eax, eax
21121 00005A16 75EE                  <1>       jnz   short dskr_3 ; (page end before system buffer end!)
21122                                  <1>       ; 03/08/2013
21123                                  <1>       ;pop  eax
21124 00005A18 390D[6C740000]        <1>       cmp   [u.count], ecx ; 0
21125                                  <1>             ; tst u.count / all bytes read off disk
21126                                  <1>             ; bne dskr
21127                                  <1>             ; br ret
21128                                  <1>         ;ja     short dskr_0
21129                                  <1>       ;mov  [u.kcall], cl ; 0 ; 09/06/2015
21130                                  <1>       ;retn
21131                                  <1>       ; 12/10/2015
21132 00005A1E 76B2                  <1>       jna   short dskr_retn
21133 00005A20 58                    <1>       pop   eax  ; (i-node number)
21134 00005A21 EB98                  <1>       jmp   short dskr_0
21135                                  <1>
21136                                  <1> passc:
21137                                  <1>       ; 18/10/2015
21138                                  <1>       ; 10/07/2015
21139                                  <1>       ; 01/07/2015
21140                                  <1>       ; 08/06/2015
21141                                  <1>       ; 04/06/2015
21142                                  <1>       ; 20/05/2015
21143                                  <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21144                                  <1>       ;
21145                                  <1>       ;(Retro UNIX 386 v1 - translation from user's virtual address
21146                                  <1>       ;                     to physical address
21147 00005A23 66833D[AD740000]00    <1>       cmp   word [u.pcount], 0 ; byte count in page = 0 (initial value)
21148                                  <1>                              ; 1-4095 --> use previous physical base address
21149                                  <1>                              ; in [u.pbase]
21150 00005A2B 7705                  <1>       ja    short passc_3
21151                                  <1>       ; 08/06/2015 - 10/07/2015
21152 00005A2D E82C000000            <1>       call  trans_addr_w
21153                                  <1> passc_3:
21154                                  <1>       ; 19/05/2015
21155 00005A32 66FF0D[AD740000]      <1>       dec   word [u.pcount]
21156                                  <1>       ;
21157 00005A39 8B1D[A9740000]        <1>       mov   ebx, [u.pbase]
21158 00005A3F 8803                  <1>       mov   [ebx], al
21159                                  <1>             ; movb r1,*u.base / move a character to the next byte of the
21160                                  <1>                               ; / users buffer
21161 00005A41 FF05[68740000]        <1>       inc   dword [u.base]
21162                                  <1>             ; inc u.base / increment the pointer to point to
21163                                  <1>                      ; / the next byte in users buffer
21164 00005A47 FF05[A9740000]        <1>       inc   dword [u.pbase] ; 04/06/2015
21165 00005A4D FF05[70740000]        <1>       inc   dword [u.nread]
```

```
21166                                  <1>             ; inc u.nread / increment the number of bytes read
21167 00005A53 FF0D[6C740000]          <1>       dec   dword [u.count]
21168                                  <1>             ; dec u.count / decrement the number of bytes to be read
21169                                  <1>             ; bne 1f / any more bytes to read?; yes, branch
21170 00005A59 C3                      <1>       retn
21171                                  <1>             ; mov (sp)+,r0 / no, do a non-local return to the caller of
21172                                  <1>                             ; / 'readi' by:
21173                                  <1>             ;/ (1) pop the return address off the stack into r0
21174                                  <1>             ; mov (sp)+,r1 / (2) pop the i-number off the stack into r1
21175                                  <1>       ;1:
21176                                  <1>             ; clr  *$ps / clear processor status
21177                                  <1>             ; rts r0 / return to address currently on top of stack
21178                                  <1>
21179                                  <1> trans_addr_r:
21180                                  <1>       ; Translate virtual address to physical address
21181                                  <1>       ; for reading from user's memory space
21182                                  <1>       ; (Retro UNIX 386 v1 feature only !)
21183                                  <1>       ; 18/10/2015
21184                                  <1>       ; 10/07/2015
21185                                  <1>       ; 09/06/2015
21186                                  <1>       ; 08/06/2015
21187                                  <1>       ; 04/06/2015
21188                                  <1>       ;
21189                                  <1>       ; 18/10/2015
21190 00005A5A 31D2                    <1>       xor   edx, edx ; 0 (read access sign)
21191 00005A5C EB04                    <1>       jmp   short trans_addr_rw
21192                                  <1>
21193                                  <1>       ;push  eax
21194                                  <1>       ;push  ebx
21195                                  <1>       ;mov   ebx, [u.base]
21196                                  <1>       ;call  get_physical_addr ; get physical address
21197                                  <1>       ;;jnc  short cpass_0
21198                                  <1>       ;jnc   short passc_1
21199                                  <1>       ;mov   [u.error], eax
21200                                  <1>       ;;pop  ebx
21201                                  <1>       ;;pop  eax
21202                                  <1>       ;jmp   error
21203                                  <1> ;cpass_0:
21204                                  <1>       ; 18/10/2015
21205                                  <1>       ; 20/05/2015
21206                                  <1>       ;mov   [u.pbase], eax ; physical address
21207                                  <1>       ;mov   [u.pcount], cx ; remain byte count in page (1-4096)
21208                                  <1>       ;pop   ebx
21209                                  <1>       ;pop   eax
21210                                  <1>       ;retn  ; 08/06/2015
21211                                  <1>
21212                                  <1> trans_addr_w:
21213                                  <1>       ; Translate virtual address to physical address
21214                                  <1>       ; for writing to user's memory space
21215                                  <1>       ; (Retro UNIX 386 v1 feature only !)
21216                                  <1>       ; 18/10/2015
21217                                  <1>       ; 29/07/2015
21218                                  <1>       ; 10/07/2015
21219                                  <1>       ; 09/06/2015
21220                                  <1>       ; 08/06/2015
21221                                  <1>       ; 04/06/2015 (passc)
21222                                  <1>       ;
21223                                  <1>       ; 18/10/2015
21224 00005A5E 29D2                    <1>       sub   edx, edx
21225 00005A60 FEC2                    <1>       inc   dl ; 1 (write access sign)
21226                                  <1> trans_addr_rw:
21227 00005A62 50                      <1>       push  eax
21228 00005A63 53                      <1>       push  ebx
21229                                  <1>       ; 18/10/2015
21230 00005A64 52                      <1>       push  edx ; r/w sign (in DL)
21231                                  <1>       ;
21232 00005A65 8B1D[68740000]          <1>       mov   ebx, [u.base]
21233 00005A6B E818DCFFFF              <1>       call  get_physical_addr ; get physical address
21234 00005A70 730A                    <1>       jnc   short passc_0
21235 00005A72 A3[9D740000]            <1>       mov   [u.error], eax
21236                                  <1>       ;pop  edx
21237                                  <1>       ;pop  ebx
21238                                  <1>       ;pop  eax
21239 00005A77 E9BEE5FFFF              <1>       jmp   error
21240                                  <1> passc_0:
21241 00005A7C F6C202                  <1>       test  dl, PTE_A_WRITE ; writable page ; 18/10/2015
21242 00005A7F 5A                      <1>       pop   edx ; 18/10/2015
21243 00005A80 751C                    <1>       jnz   short passc_1
21244                                  <1>       ; 18/10/2015
21245 00005A82 20D2                    <1>       and   dl, dl
21246 00005A84 7418                    <1>       jz    short passc_1
21247                                  <1>       ; 20/05/2015
21248                                  <1>       ; read only (duplicated) page -must be copied to a new page-
21249                                  <1>       ; EBX = linear address
21250 00005A86 51                      <1>       push  ecx
21251 00005A87 E811D9FFFF              <1>       call  copy_page
21252 00005A8C 59                      <1>       pop   ecx
21253 00005A8D 721E                    <1>       jc    short passc_2
21254 00005A8F 50                      <1>       push  eax ; physical address of the new/allocated page
21255 00005A90 E81DDBFFFF              <1>       call  add_to_swap_queue
21256 00005A95 58                      <1>       pop   eax
21257                                  <1>       ; 18/10/2015
21258 00005A96 81E3FF0F0000            <1>       and   ebx, PAGE_OFF ; 0FFFh
21259                                  <1>       ;mov  ecx, PAGE_SIZE
21260                                  <1>       ;sub  ecx, ebx
21261 00005A9C 01D8                    <1>       add   eax, ebx
21262                                  <1> passc_1:
21263                                  <1>       ; 18/10/2015
21264                                  <1>       ; 20/05/2015
21265 00005A9E A3[A9740000]            <1>       mov   [u.pbase], eax ; physical address
21266 00005AA3 66890D[AD740000]        <1>       mov   [u.pcount], cx ; remain byte count in page (1-4096)
21267 00005AAA 5B                      <1>       pop   ebx
21268 00005AAB 58                      <1>       pop   eax
21269 00005AAC C3                      <1>       retn  ; 08/06/2015
21270                                  <1> passc_2:
```

```
21271 00005AAD C705[9D740000]0100- <1>       mov   dword [u.error], ERR_MINOR_IM ; "Insufficient memory !" error
21272 00005AB5 0000               <1>
21273                             <1>       ;pop   ebx
21274                             <1>       ;pop   eax
21275 00005AB7 E97EE5FFFF         <1>       jmp   error
21276                             <1>
21277                             <1> writei:
21278                             <1>       ; 20/05/2015
21279                             <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21280                             <1>       ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
21281                             <1>       ;
21282                             <1>       ; Write data to file with inode number in R1
21283                             <1>       ;
21284                             <1>       ; INPUTS ->
21285                             <1>       ;    r1 - inode number
21286                             <1>       ;    u.count - byte count to be written
21287                             <1>       ;    u.base - points to user buffer
21288                             <1>       ;    u.fofp - points to word with current file offset
21289                             <1>       ; OUTPUTS ->
21290                             <1>       ;    u.count - cleared
21291                             <1>       ;    u.nread - accumulates total bytes passed back
21292                             <1>       ; ((AX = R1))
21293                             <1>       ;    (Retro UNIX Prototype : 18/11/2012 - 11/11/2012, UNIXCOPY.ASM)
21294                             <1>       ;    ((Modified registers: DX, BX, CX, SI, DI, BP))
21295                             <1>
21296 00005ABC 31C9              <1>       xor   ecx, ecx
21297 00005ABE 890D[70740000]     <1>       mov   [u.nread], ecx  ; 0
21298                             <1>             ; clr u.nread / clear the number of bytes transmitted during
21299                             <1>                           ; / read or write calls
21300 00005AC4 66890D[AD740000]   <1>       mov   [u.pcount], cx ; 19/05/2015
21301 00005ACB 390D[6C740000]     <1>       cmp   [u.count], ecx
21302                             <1>       ;     ; tst u.count / test the byte count specified by the user
21303 00005AD1 7701               <1>       ja    short writei_1 ; 1f
21304                             <1>             ; bgt 1f / any bytes to output; yes, branch
21305 00005AD3 C3                <1>       retn
21306                             <1>       ;     ; rts r0 / no, return - no writing to do
21307                             <1> writei_1: ;1:
21308                             <1>             ; mov r1 ,-(sp) / save the i-node number on the stack
21309 00005AD4 6683F828           <1>       cmp   ax, 40
21310                             <1>             ; cmp r1,$40.
21311                             <1>             ; / does the i-node number indicate a special file?
21312 00005AD8 0F87F2000000       <1>         ja      dskw
21313                             <1>             ; bgt dskw / no, branch to standard file output
21314                             <1>       ; (20/05/2015)
21315 00005ADE 50                 <1>       push  eax ; because subroutines will jump to 'ret_'
21316 00005ADF 0FB6D8             <1>       movzx ebx, al
21317 00005AE2 66C1E302           <1>       shl   bx, 2
21318                             <1>             ; asl r1 / yes, calculate the index into the special file
21319 00005AE6 81C3[EA5A0000]     <1>       add   ebx, writei_2 - 4
21320 00005AEC FF23               <1>       jmp   dword [ebx]
21321                             <1>             ; jmp *1f-2(r1)
21322                             <1>             ; / jump table and jump to the appropriate routine
21323                             <1> writei_2: ;1:
21324 00005AEE [3A5B0000]         <1>       dd    wtty ; tty, AX = 1 (runix)
21325                             <1>              ;wtty / tty; r1=2
21326                             <1>              ;wppt / ppt; r1=4
21327 00005AF2 [A05B0000]         <1>       dd    wmem ; mem, AX = 2 (runix)
21328                             <1>              ;wmem / mem; r1=6
21329                             <1>              ;wrf0 / rf0
21330                             <1>              ;wrk0 / rk0
21331                             <1>              ;wtap / tap0
21332                             <1>              ;wtap / tap1
21333                             <1>              ;wtap / tap2
21334                             <1>              ;wtap / tap3
21335                             <1>              ;wtap / tap4
21336                             <1>              ;wtap / tap5
21337                             <1>              ;wtap / tap6
21338                             <1>              ;wtap / tap7
21339 00005AF6 [DB600000]         <1>       dd    wfd ; fd0, AX = 3 (runix only)
21340 00005AFA [DB600000]         <1>       dd    wfd ; fd1, AX = 4 (runix only)
21341 00005AFE [DB600000]         <1>       dd    whd ; hd0, AX = 5 (runix only)
21342 00005B02 [DB600000]         <1>       dd    whd ; hd1, AX = 6 (runix only)
21343 00005B06 [DB600000]         <1>       dd    whd ; hd2, AX = 7 (runix only)
21344 00005B0A [DB600000]         <1>       dd    whd ; hd3, AX = 8 (runix only)
21345 00005B0E [915B0000]         <1>       dd    wlpr ; lpr, AX = 9   (runix)
21346 00005B12 [8B5B0000]         <1>       dd    xmtt ; tty0, AX = 10 (runix)
21347                             <1>              ;xmtt / tty0
21348 00005B16 [8B5B0000]         <1>       dd    xmtt ; tty1, AX = 11 (runix)
21349                             <1>              ;xmtt / tty1
21350 00005B1A [8B5B0000]         <1>       dd    xmtt ; tty2, AX = 12 (runix)
21351                             <1>              ;xmtt / tty2
21352 00005B1E [8B5B0000]         <1>       dd    xmtt ; tty3, AX = 13 (runix)
21353                             <1>              ;xmtt / tty3
21354 00005B22 [8B5B0000]         <1>       dd    xmtt ; tty4, AX = 14 (runix)
21355                             <1>              ;xmtt / tty4
21356 00005B26 [8B5B0000]         <1>       dd    xmtt ; tty5, AX = 15 (runix)
21357                             <1>              ;xmtt / tty5
21358 00005B2A [8B5B0000]         <1>       dd    xmtt ; tty6, AX = 16 (runix)
21359                             <1>              ;xmtt / tty6
21360 00005B2E [8B5B0000]         <1>       dd    xmtt ; tty7, AX = 17 (runix)
21361                             <1>              ;xmtt / tty7
21362 00005B32 [8B5B0000]         <1>       dd    xmtt ; COM1, AX = 18 (runix only)
21363                             <1>             ; / wlpr / lpr
21364 00005B36 [8B5B0000]         <1>       dd    xmtt ; COM2, AX = 19 (runix only)
21365                             <1>
21366                             <1> wtty: ; write to console tty (write to screen)
21367                             <1>       ; 18/11/2015
21368                             <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21369                             <1>       ; 12/03/2013 - 07/07/2014 (Retro UNIX 8086 v1)
21370                             <1>       ;
21371                             <1>       ; Console tty output is on current video page
21372                             <1>       ; Console tty character output procedure is changed here
21373                             <1>       ; acconding to IBM PC compatible ROM BIOS video (text mode) functions.
21374                             <1>       ;
21375 00005B3A 0FB61D[97740000]   <1>       movzx  ebx, byte [u.uno] ; process number
```

```
21376 00005B41 8AA3[95710000]      <1>       mov    ah, [ebx+p.ttyc-1] ; current/console tty
21377 00005B47 88E0                <1>       mov    al, ah ; 07/07/2014
21378                              <1> wttys:
21379                              <1>       ; 10/10/2013
21380 00005B49 8825[9C740000]      <1>       mov    [u.ttyn], ah
21381                              <1>       ; 13/01/2014
21382 00005B4F FEC0                <1>       inc    al
21383 00005B51 A2[79740000]        <1>       mov    [u.ttyp+1], al ; tty number + 1
21384                              <1> wtty_nc: ; 15/05/2013
21385                              <1>       ; AH = [u.ttyn] = tty number ; 28/07/2013
21386 00005B56 E81C010000          <1>       call   cpass
21387                              <1>              ; jsr r0,cpass / get next character from user buffer area; if
21388                              <1>                             ; / none go to return address in syswrite
21389                              <1>              ; tst r1 / is character = null
21390                              <1>              ; beq wtty / yes, get next character
21391                              <1>       ; 10/10/2013
21392 00005B5B 742C                <1>       jz     short wret
21393                              <1>       ;1 :
21394                              <1>              ;mov   $240,*$ps / no, set processor priority to five
21395                              <1>              ;cmpb  cc+1,$20. / is character count for console tty greater
21396                              <1>              ;                / than 20
21397                              <1>              ;bhis  2f / yes; branch to put process to sleep
21398                              <1>       ; 27/06/2014
21399                              <1> wtty_1:
21400                              <1>       ; AH = tty number
21401                              <1>       ; AL = ASCII code of the character
21402                              <1>       ; 15/04/2014
21403 00005B5D 6650                <1>       push   ax
21404 00005B5F E8A80A0000          <1>       call   putc ; 14/05/2013
21405 00005B64 731F                <1>       jnc    short wtty_2
21406                              <1>       ; 18/11/2015
21407 00005B66 E8EEF8FFFF          <1>       call   idle
21408 00005B6B 668B0424            <1>       mov    ax, [esp]
21409 00005B6F E8980A0000          <1>       call   putc
21410 00005B74 730F                <1>       jnc    short wtty_2
21411                              <1>       ; 02/06/2014
21412 00005B76 8A25[9C740000]      <1>       mov    ah, [u.ttyn]
21413 00005B7C E86CF9FFFF          <1>       call   sleep
21414 00005B81 6658                <1>       pop    ax
21415 00005B83 EBD8                <1>       jmp    short wtty_1
21416                              <1>              ; jc   error ; 15/05/2013 (COM1 or COM2 serial port error)
21417                              <1>              ; jsr  r0,putc; 1 / find place in freelist to assign to
21418                              <1>                             ; / console tty and
21419                              <1>              ; br   2f / place character in list; if none available
21420                              <1>                ; / branch to put process to sleep
21421                              <1>              ; jsr  r0,startty / attempt to output character on tty
21422                              <1> wtty_2:
21423                              <1>       ; 15/04/2014
21424 00005B85 6658                <1>       pop    ax
21425 00005B87 EBCD                <1>       jmp    short wtty_nc
21426                              <1>              ; br wtty
21427                              <1> wret: ; 10/10/2013 (20/05/2015)
21428 00005B89 58                  <1>       pop    eax
21429 00005B8A C3                  <1>       retn
21430                              <1>       ;2:
21431                              <1>              ;mov   r1,-(sp) / place character on stack
21432                              <1>              ;jsr   r0,sleep; 1 / put process to sleep
21433                              <1>              ;mov   (sp)+,r1 / remove character from stack
21434                              <1>              ;br    1b / try again to place character in clist and output
21435                              <1>
21436                              <1> xmtt:   ; < send/write character to tty >
21437                              <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21438                              <1>       ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
21439                              <1>       ;
21440                              <1>       ; Retro UNIX 8086 v1 modification !
21441                              <1>       ;
21442                              <1>       ; In original UNIX v1, 'xmtt' routine
21443                              <1>       ;            (exactly different than this one)
21444                              <1>       ;     was in 'u9.s' file.
21445                              <1>       ;
21446 00005B8B 2C0A                <1>       sub    al, 10
21447                              <1>       ; AL = tty number (0 to 9), (COM1=8, COM2=9)
21448                              <1>        ; 10/10/2013
21449 00005B8D 88C4                <1>       mov    ah, al
21450                              <1>       ; 28/07/2013
21451 00005B8F EBB8                <1>       jmp    short wttys
21452                              <1>
21453                              <1> ;wppt:
21454                              <1> ;     jsr    r0,cpass / get next character from user buffer area,
21455                              <1> ;                     / if none return to writei's calling routine
21456                              <1> ;     jsr    r0,pptoc / output character on ppt
21457                              <1> ;     br     wppt
21458                              <1> wlpr:
21459 00005B91 C705[9D740000]0F00- <1>         mov     dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
21460 00005B99 0000                <1>
21461 00005B9B E99AE4FFFF          <1>       jmp    error   ; ... Printing procedure will be located here ...
21462                              <1>              ;/     jsr    r0,cpass
21463                              <1>              ;/     cmp    r0,$'a
21464                              <1>              ;/     blo    1f
21465                              <1>              ;/     cmp    r1,$'z
21466                              <1>              ;/     bhi    1f
21467                              <1>              ;/     sub    $40,r1
21468                              <1>              ;/1:
21469                              <1>              ;/     jsr    r0,lptoc
21470                              <1>              ;/     br     wlpr
21471                              <1>              ; br rmem / continue
21472                              <1>
21473                              <1> wmem: ; / transfer characters from a user area of core to memory file
21474                              <1>       ; 17/10/2015
21475                              <1>       ; 11/06/2015
21476                              <1>       ; 24/05/2015
21477                              <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21478                              <1>       ;
21479 00005BA0 813D[D8070000]-     <1>       cmp    dword [x_timer], clock ; multi tasking clock/timer
21480 00005BA6 [7D540000]          <1>
```

```
21481 00005BAA 7415                <1>         je      short wmem_acc_err
21482                              <1>       ;
21483 00005BAC 8B35[58740000]      <1>         mov     esi, [u.fofp]
21484                              <1> wmem_1:
21485 00005BB2 E8C0000000          <1>       call  cpass
21486                              <1>             ; jsr r0,cpass / get next character from users area of
21487                              <1>                         ; / core and put it in r1
21488                              <1>             ; mov r1,-(sp) / put character on the stack
21489                              <1>       ; 20/09/2013
21490 00005BB7 74D0                <1>       jz    short wret ; wmem_2
21491 00005BB9 8B1E                <1>         mov     ebx, [esi]
21492                              <1>             ; mov *u.fofp,r1 / save file offset in r1
21493 00005BBB FF06                <1>         inc     dword [esi] ; 17/10/2015
21494                              <1>             ; inc *u.fofp / increment file offset to point to next
21495                              <1>                       ; / available location in file
21496 00005BBD 8803                <1>       mov   [ebx], al
21497                              <1>             ; movb (sp)+,(r1) / pop char off stack, put in memory loc
21498                              <1>                           ; / assigned to it
21499 00005BBF EBF1                <1>       jmp   short wmem_1
21500                              <1>             ; br wmem / continue
21501                              <1>       ;1:
21502                              <1>       ;jmp  error / ?
21503                              <1> ;wmem_2:
21504                              <1> ;     ; 20/09/2013
21505                              <1> ;     pop   ax
21506                              <1> ;     retn
21507                              <1>
21508                              <1> wmem_acc_err:
21509 00005BC1 C705[9D740000]0B00- <1>       mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
21510 00005BC9 0000                <1>
21511 00005BCB E96AE4FFFF          <1>       jmp   error
21512                              <1>
21513                              <1>
21514                              <1> dskw: ; / write routine for non-special files
21515                              <1>       ;
21516                              <1>       ; 25/07/2015
21517                              <1>       ; 16/06/2015
21518                              <1>       ; 09/06/2015
21519                              <1>       ; 31/05/2015 (Retro UNIX 386 v1 - Beginning)
21520                              <1>       ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
21521                              <1>       ;
21522                              <1>       ; 01/08/2013 (mkdir_w check)
21523 00005BD0 6650                <1>       push  ax ; 26/04/2013
21524                              <1>             ; mov (sp),r1 / get an i-node number from the stack into r1
21525                              <1>       ; AX = inode number
21526 00005BD2 E80FFBFFFF          <1>       call  iget
21527                              <1>             ; jsr r0,iget / write i-node out (if modified),
21528                              <1>                         ; / read i-node 'r1' into i-node area of core
21529 00005BD7 8B1D[58740000]      <1>         mov     ebx, [u.fofp]
21530 00005BDD 8B13                <1>       mov   edx, [ebx]
21531                              <1>             ; mov *u.fofp,r2 / put the file offset [(u.off) or the offset
21532                              <1>                           ; / in the fsp entry for this file] in r2
21533 00005BDF 0315[6C740000]      <1>       add   edx, [u.count]
21534                              <1>             ; add u.count,r2 / no. of bytes to be written
21535                              <1>                           ; / + file offset is put in r2
21536                              <1>       ; 16/06/2015
21537 00005BE5 81FAFFFF0000        <1>       cmp   edx, 65535 ; file size limit (for UNIX v1 file system)
21538 00005BEB 760F                <1>       jna   short dskw_0
21539 00005BED C705[9D740000]1400- <1>       mov   dword [u.error], ERR_FILE_SIZE ; 'file size error !'
21540 00005BF5 0000                <1>
21541 00005BF7 E93EE4FFFF          <1>       jmp   error
21542                              <1> dskw_0:
21543 00005BFC 663B15[1A710000]    <1>       cmp     dx, [i.size]
21544                              <1>             ; cmp r2,i.size / is this greater than the present size of
21545                              <1>                           ; / the file?
21546 00005C03 760C                <1>       jna   short dskw_1
21547                              <1>             ; blos 1f / no, branch
21548 00005C05 668915[1A710000]    <1>         mov     [i.size], dx
21549                              <1>             ; mov r2,i.size / yes, increase the file size to
21550                              <1>                         ; / file offset + no. of data bytes
21551 00005C0C E8E8FBFFFF          <1>       call  setimod
21552                              <1>             ; jsr r0,setimod / set imod=1 (i.e., core inode has been
21553                              <1>                       ; / modified), stuff time of modification into
21554                              <1>                     ; / core image of i-node
21555                              <1> dskw_1: ; 1:
21556 00005C11 E85DF9FFFF          <1>       call  mget
21557                              <1>       ; eAX = Block number
21558                              <1>             ; jsr r0,mget / get the block no. in which to write
21559                              <1>                       ; /      the next data byte
21560                              <1>       ; eax = block number
21561 00005C16 8B1D[58740000]      <1>       mov     ebx, [u.fofp]
21562 00005C1C 8B13                <1>       mov   edx, [ebx]
21563 00005C1E 81E2FF010000        <1>       and   edx, 1FFh
21564                              <1>             ; bit *u.fofp,$777 / test the lower 9 bits of the file offset
21565 00005C24 750C                <1>       jnz   short dskw_2
21566                              <1>             ; bne 2f / if its non-zero, branch; if zero, file offset = 0,
21567                              <1>                     ; / 512, 1024,...(i.e., start of new block)
21568 00005C26 813D[6C740000]0002- <1>       cmp   dword [u.count], 512
21569 00005C2E 0000                <1>
21570                              <1>             ; cmp u.count,$512. / if zero, is there enough data to fill
21571                              <1>                           ; / an entire block? (i.e., no. of
21572 00005C30 7305                <1>       jnb   short dskw_3
21573                              <1>             ; bhis 3f / bytes to be written greater than 512.?
21574                              <1>                   ; / Yes, branch. Don't have to read block
21575                              <1> dskw_2: ; 2: / in as no past info. is to be saved (the entire block will be
21576                              <1>             ; / overwritten).
21577 00005C32 E87C050000          <1>       call  dskrd
21578                              <1>             ; jsr r0,dskrd / no, must retain old info..
21579                              <1>                         ; / Hence, read block 'r1' into an I/O buffer
21580                              <1> dskw_3: ; 3:
21581                              <1>       ; eAX (r1) = block/sector number
21582 00005C37 E8D7050000          <1>       call  wslot
21583                              <1>             ; jsr r0,wslot / set write and inhibit bits in I/O queue,
21584                              <1>                       ; / proc. status=0, r5 points to 1st word of data
21585 00005C3C 803D[AF740000]00    <1>       cmp   byte [u.kcall], 0
```

```
21586 00005C43 770F                <1>       ja    short dskw_5 ; zf=0 -> the caller is 'mkdir'
21587                              <1>       ;
21588 00005C45 66833D[AD740000]00  <1>       cmp   word [u.pcount], 0
21589 00005C4D 7705                <1>       ja    short dskw_5
21590                              <1> dskw_4:
21591                              <1>       ; [u.base] = virtual address to transfer (as source address)
21592 00005C4F E806FEFFFF          <1>       call  trans_addr_r ; translate virtual address to physical (r)
21593                              <1> dskw_5:
21594                              <1>       ; eBX (r5) = system (I/O) buffer address
21595 00005C54 E87E000000          <1>       call  sioreg
21596                              <1>             ; jsr r0,sioreg / r3 = no. of bytes of data,
21597                              <1>                         ; / r1 = address of data, r2 points to location
21598                              <1>                         ; / in buffer in which to start writing data
21599                              <1>       ; eSI = file (user data) offset
21600                              <1>       ; eDI = sector (I/O) buffer offset
21601                              <1>       ; eCX = byte count
21602                              <1>       ;
21603 00005C59 F3A4                <1>       rep   movsb
21604                              <1>             ; movb (r1 )+,(r2)+
21605                              <1>                      ; / transfer a byte of data to the I/O buffer
21606                              <1>             ; dec r3 / decrement no. of bytes to be written
21607                              <1>             ; bne 2b / have all bytes been transferred? No, branch
21608                              <1>       ; 25/07/2015
21609                              <1>       ; eax = remain bytes in buffer
21610                              <1>         ;       (check if remain bytes in the buffer > [u.pcount])
21611 00005C5B 09C0                <1>       or    eax, eax
21612 00005C5D 75F0                <1>       jnz   short dskw_4 ; (page end before system buffer end!)
21613                              <1> dskw_6:
21614 00005C5F E8CB050000          <1>       call  dskwr
21615                              <1>             ; jsr r0,dskwr / yes, write the block and the i-node
21616 00005C64 833D[6C740000]00    <1>         cmp     dword [u.count], 0
21617                              <1>             ; tst u.count / any more data to write?
21618 00005C6B 77A4                <1>       ja    short dskw_1
21619                              <1>             ; bne 1b / yes, branch
21620                              <1>       ; 03/08/2013
21621 00005C6D C605[AF740000]00    <1>       mov   byte [u.kcall], 0
21622                              <1>       ; 20/09/2013 (;;)
21623 00005C74 6658                <1>       pop   ax
21624 00005C76 C3                  <1>       retn
21625                              <1>       ;;jmp  short dskw_ret
21626                              <1>              ; jmp ret / no, return to the caller via 'ret'
21627                              <1>
21628                              <1> cpass: ; / get next character from user area of core and put it in r1
21629                              <1>       ; 18/10/2015
21630                              <1>       ; 10/10/2015
21631                              <1>       ; 10/07/2015
21632                              <1>       ; 02/07/2015
21633                              <1>       ; 01/07/2015
21634                              <1>       ; 24/06/2015
21635                              <1>       ; 08/06/2015
21636                              <1>       ; 04/06/2015
21637                              <1>       ; 20/05/2015
21638                              <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21639                              <1>       ;
21640                              <1>       ; INPUTS ->
21641                              <1>       ;     [u.base] = virtual address in user area
21642                              <1>       ;     [u.count] = byte count (max.)
21643                              <1>       ;     [u.pcount] = byte count in page (0 = reset)
21644                              <1>       ; OUTPUTS ->
21645                              <1>       ;     AL = the character which is pointed by [u.base]
21646                              <1>       ;     zf = 1 -> transfer count has been completed
21647                              <1>         ;
21648                              <1>       ; ((Modified registers:  EAX, EDX, ECX))
21649                              <1>       ;
21650                              <1>       ;
21651 00005C77 833D[6C740000]00    <1>       cmp   dword [u.count], 0  ; 14/08/2013
21652                              <1>             ; tst u.count / have all the characters been transferred
21653                              <1>                       ; / (i.e., u.count, # of chars. left
21654 00005C7E 763F                <1>       jna   short cpass_3
21655                              <1>             ; beq 1f / to be transferred = 0?) yes, branch
21656 00005C80 FF0D[6C740000]      <1>       dec   dword [u.count]
21657                              <1>             ; dec u.count / no, decrement u.count
21658                              <1>         ; 19/05/2015
21659                              <1>       ;(Retro UNIX 386 v1 - translation from user's virtual address
21660                              <1>       ;                  to physical address
21661 00005C86 66833D[AD740000]00  <1>       cmp   word [u.pcount], 0 ; byte count in page = 0 (initial value)
21662                              <1>                         ; 1-4095 --> use previous physical base address
21663                              <1>                         ; in [u.pbase]
21664 00005C8E 770E                <1>       ja    short cpass_1
21665                              <1>       ; 02/07/2015
21666 00005C90 833D[A5740000]00    <1>         cmp     dword [u.ppgdir], 0  ; is the caller os kernel
21667 00005C97 7427                <1>         je      short cpass_k       ; (sysexec, '/etc/init') ?
21668                              <1>       ; 08/06/2015 - 10/07/2015
21669 00005C99 E8BCFDFFFF          <1>       call  trans_addr_r
21670                              <1> cpass_1:
21671                              <1>       ; 02/07/2015
21672                              <1>       ; 24/06/2015
21673 00005C9E 66FF0D[AD740000]    <1>       dec   word [u.pcount]
21674                              <1> cpass_2:
21675                              <1>        ;10/10/2015
21676                              <1>       ; 02/07/2015
21677 00005CA5 8B15[A9740000]      <1>       mov   edx, [u.pbase]
21678 00005CAB 8A02                <1>       mov   al, [edx] ; 10/10/2015
21679                              <1>             ; movb *u.base,r1 / take the character pointed to
21680                              <1>                          ; / by u.base and put it in r1
21681 00005CAD FF05[70740000]      <1>       inc   dword [u.nread]
21682                              <1>             ; inc u.nread / increment no. of bytes transferred
21683 00005CB3 FF05[68740000]      <1>       inc   dword [u.base]
21684                              <1>             ; inc u.base / increment the buffer address to point to the
21685                              <1>                        ; / next byte
21686 00005CB9 FF05[A9740000]      <1>       inc   dword [u.pbase] ; 04/06/2015
21687                              <1> cpass_3:
21688 00005CBF C3                  <1>       retn
21689                              <1>             ; rts  r0 / next byte
21690                              <1>       ; 1:
```

```
21691                                  <1>             ; mov (sp)+,r0
21692                                  <1>                     ; / put return address of calling routine into r0
21693                                  <1>             ; mov (sp)+,r1 / i-number in r1
21694                                  <1>             ; rts r0 / non-local return
21695                                  <1> cpass_k:
21696                                  <1>       ; 02/07/2015
21697                                  <1>       ; The caller is os kernel
21698                                  <1>       ; (get sysexec arguments from kernel's memory space)
21699                                  <1>       ;
21700 00005CC0 8B1D[68740000]          <1>       mov    ebx, [u.base]
21701 00005CC6 66C705[AD740000]00-     <1>         mov     word [u.pcount], PAGE_SIZE ; 4096
21702 00005CCE 10                      <1>
21703 00005CCF 891D[A9740000]          <1>       mov    [u.pbase], ebx
21704 00005CD5 EBCE                    <1>       jmp    short cpass_2
21705                                  <1>
21706                                  <1> sioreg:
21707                                  <1>       ; 25/07/2015
21708                                  <1>       ; 18/07/2015
21709                                  <1>       ; 02/07/2015
21710                                  <1>       ; 17/06/2015
21711                                  <1>       ; 09/06/2015
21712                                  <1>       ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21713                                  <1>       ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
21714                                  <1>       ;
21715                                  <1>       ; INPUTS ->
21716                                  <1>       ;     eBX = system buffer (data) address (r5)
21717                                  <1>       ;     [u.fofp] = pointer to file offset pointer
21718                                  <1>       ;     [u.base] = virtual address of the user buffer
21719                                  <1>       ;     [u.pbase] = physical address of the user buffer
21720                                  <1>       ;     [u.count] = byte count
21721                                  <1>       ;     [u.pcount] = byte count within page frame
21722                                  <1>       ; OUTPUTS ->
21723                                  <1>       ;     eSI = user data offset (r1)
21724                                  <1>       ;     eDI = system (I/O) buffer offset (r2)
21725                                  <1>       ;     eCX = byte count (r3)
21726                                  <1>       ;     EAX = remain bytes after byte count within page frame
21727                                  <1>       ;      (If EAX > 0, transfer will continue from the next page)
21728                                  <1>         ;
21729                                  <1>       ; ((Modified registers:  EDX))
21730                                  <1>
21731 00005CD7 8B35[58740000]          <1>         mov     esi, [u.fofp]
21732 00005CDD 8B3E                    <1>         mov     edi, [esi]
21733                                  <1>             ; mov *u.fofp,r2 / file offset (in bytes) is moved to r2
21734 00005CDF 89F9                    <1>       mov    ecx, edi
21735                                  <1>             ; mov r2,r3 / and also to r3
21736 00005CE1 81C900FEFFFF            <1>       or     ecx, 0FFFFFE00h
21737                                  <1>             ; bis $177000,r3 / set bits 9,...,15 of file offset in r3
21738 00005CE7 81E7FF010000            <1>       and    edi, 1FFh
21739                                  <1>             ; bic $!777,r2 / calculate file offset mod 512.
21740 00005CED 01DF                    <1>       add    edi, ebx ; EBX = system buffer (data) address
21741                                  <1>             ; add r5,r2 / r2 now points to 1st byte in system buffer
21742                                  <1>                     ; / where data is to be placed
21743                                  <1>                ; mov u.base,r1 / address of data is in r1
21744 00005CEF F7D9                    <1>       neg    ecx
21745                                  <1>             ; neg r3 / 512 - file offset (mod512.) in r3
21746                                  <1>                    ; / (i.e., the no. of free bytes in the file block)
21747 00005CF1 3B0D[6C740000]          <1>       cmp    ecx, [u.count]
21748                                  <1>             ; cmp r3,u.count / compare this with the no. of data bytes
21749                                  <1>                             ; / to be written to the file
21750 00005CF7 7606                    <1>       jna    short sioreg_0
21751                                  <1>             ; blos 2f / if less than branch. Use the no. of free bytes
21752                                  <1>                    ; / in the file block as the number to be written
21753 00005CF9 8B0D[6C740000]          <1>       mov    ecx, [u.count]
21754                                  <1>             ; mov u.count,r3 / if greater than, use the no. of data
21755                                  <1>                             ; / bytes as the number to be written
21756                                  <1> sioreg_0:
21757                                  <1>       ; 17/06/2015
21758 00005CFF 803D[AF740000]00        <1>       cmp    byte [u.kcall], 0
21759 00005D06 7613                    <1>       jna    short sioreg_1
21760                                  <1>       ; 25/07/2015
21761                                  <1>        ; the caller is 'mkdir' or 'namei'
21762 00005D08 A1[68740000]            <1>       mov    eax, [u.base] ; 25/07/2015
21763 00005D0D A3[A9740000]            <1>       mov    [u.pbase], eax ; physical address = virtual address
21764 00005D12 66890D[AD740000]        <1>       mov    word [u.pcount], cx ; remain bytes in buffer (1 sector)
21765 00005D19 EB0B                    <1>       jmp    short sioreg_2
21766                                  <1> sioreg_1:
21767                                  <1>       ; 25/07/2015
21768                                  <1>       ; 18/07/2015
21769                                  <1>       ; 09/06/2015
21770 00005D1B 0FB715[AD740000]        <1>       movzx  edx, word [u.pcount]
21771                                  <1>             ; ecx and [u.pcount] are always > 0, here
21772 00005D22 39D1                    <1>       cmp    ecx, edx
21773 00005D24 772A                    <1>       ja     short sioreg_4 ; transfer count > [u.pcount]
21774                                  <1> sioreg_2: ; 2:
21775 00005D26 31C0                    <1>       xor    eax, eax ; 25/07/2015
21776                                  <1> sioreg_3:
21777 00005D28 010D[70740000]          <1>       add    [u.nread], ecx
21778                                  <1>             ; add r3,u.nread / r3 + number of bytes xmitted
21779                                  <1>                              ; / during write is put into u.nread
21780 00005D2E 290D[6C740000]          <1>       sub    [u.count], ecx
21781                                  <1>             ; sub r3,u.count / u.count = no. of bytes that still
21782                                  <1>                              ; / must be written or read
21783 00005D34 010D[68740000]          <1>       add    [u.base], ecx
21784                                  <1>             ; add r3,u.base / u.base points to the 1st of the remaining
21785                                  <1>                             ; / data bytes
21786 00005D3A 010E                    <1>         add  [esi], ecx
21787                                  <1>             ; add r3,*u.fofp / new file offset = number of bytes done
21788                                  <1>                              ; / + old file offset
21789                                  <1>       ; 25/07/2015
21790 00005D3C 8B35[A9740000]          <1>       mov    esi, [u.pbase]
21791 00005D42 66290D[AD740000]        <1>       sub    [u.pcount], cx
21792 00005D49 010D[A9740000]          <1>       add    [u.pbase], ecx
21793 00005D4F C3                      <1>         retn
21794                                  <1>             ; rts r0
21795                                  <1>             ; transfer count > [u.pcount]
```

```
21796                                  <1> sioreg_4:
21797                                  <1>       ; 25/07/2015
21798                                  <1>       ; transfer count > [u.pcount]
21799                                  <1>       ; (ecx > edx)
21800 00005D50 89C8                     <1>       mov    eax, ecx
21801 00005D52 29D0                     <1>       sub    eax, edx ; remain bytes for 1 sector (block) transfer
21802 00005D54 89D1                     <1>       mov    ecx, edx ; current transfer count = [u.pcount]
21803 00005D56 EBD0                     <1>       jmp    short sioreg_3
21804                                      %include 'u7.s'        ; 18/04/2015
21805                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS7.INC
21806                                  <1> ; Last Modification: 14/11/2015
21807                                  <1> ; ---------------------------------------------------------------------------
21808                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
21809                                  <1> ; (v0.1 - Beginning: 11/07/2012)
21810                                  <1> ;
21811                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
21812                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
21813                                  <1> ; <Bell Laboratories (17/3/1972)>
21814                                  <1> ; <Preliminary Release of UNIX Implementation Document>
21815                                  <1> ;
21816                                  <1> ; Retro UNIX 8086 v1 - U7.ASM (13/07/2014) //// UNIX v1 -> u7.s
21817                                  <1> ;
21818                                  <1> ; ***************************************************************************
21819                                  <1>
21820                                  <1> sysmount: ; / mount file system; args special; name
21821                                  <1>       ; 14/11/2015
21822                                  <1>       ; 24/10/2015
21823                                  <1>       ; 13/10/2015
21824                                  <1>       ; 10/07/2015
21825                                  <1>       ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
21826                                  <1>       ; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
21827                                  <1>       ;
21828                                  <1>       ; 'sysmount' anounces to the system that a removable
21829                                  <1>       ; file system has been mounted on a special file.
21830                                  <1>       ; The device number of the special file is obtained via
21831                                  <1>       ; a call to 'getspl'. It is put in the I/O queue entry for
21832                                  <1>       ; dismountable file system (sb1) and the I/O queue entry is
21833                                  <1>       ; set up to read (bit 10 is set). 'ppoke' is then called to
21834                                  <1>       ; to read file system into core, i.e. the first block on the
21835                                  <1>       ; mountable file system is read in. This block is super block
21836                                  <1>       ; for the file system. This call is super user restricted.
21837                                  <1>       ;
21838                                  <1>       ; Calling sequence:
21839                                  <1>       ;     sysmount; special; name
21840                                  <1>       ; Arguments:
21841                                  <1>       ;     special - pointer to name of special file (device)
21842                                  <1>       ;     name -  pointer to name of the root directory of the
21843                                  <1>       ;            newly mounted file system. 'name' should
21844                                  <1>       ;            always be a directory.
21845                                  <1>       ; Inputs: -
21846                                  <1>       ; Outputs: -
21847                                  <1>       ; ..........................................................
21848                                  <1>       ;
21849                                  <1>       ; Retro UNIX 8086 v1 modification:
21850                                  <1>       ;      'sysmount' system call has two arguments; so,
21851                                  <1>       ;     * 1st argument, special is pointed to by BX register
21852                                  <1>       ;     * 2nd argument, name is in CX register
21853                                  <1>       ;
21854                                  <1>       ;     NOTE: Device numbers, names and related procedures are
21855                                  <1>       ;            already modified for IBM PC compatibility and
21856                                  <1>       ;            Retro UNIX 8086 v1 device configuration.
21857                                  <1>
21858                                  <1>       ;call  arg2
21859                                  <1>             ; jsr r0,arg2 / get arguments special and name
21860 00005D58 891D[60740000]           <1>       mov    [u.namep], ebx
21861 00005D5E 51                       <1>       push   ecx ; directory name
21862 00005D5F 66833D[34740000]00       <1>       cmp    word [mnti], 0
21863                                  <1>             ; tst mnti / is the i-number of the cross device file
21864                                  <1>                    ; / zero?
21865                                  <1>       ;ja    error
21866                                  <1>             ; bne errora / no, error
21867 00005D67 0F87E9000000             <1>       ja     sysmnt_err0
21868                                  <1>       ;
21869 00005D6D E8CC000000               <1>       call   getspl
21870                                  <1>             ; jsr r0,getspl / get special files device number in r1
21871                                  <1>       ; 13/10/2015
21872 00005D72 0FB7D8                   <1>       movzx  ebx, ax ; ; Retro UNIX 8086 v1 device number (0 to 5)
21873 00005D75 F683[6A6B0000]80         <1>         test    byte [ebx+drv.status], 80h ; 24/10/2015
21874 00005D7C 750F                     <1>       jnz    short sysmnt_1
21875                                  <1> sysmnt_err1:
21876 00005D7E C705[9D740000]0F00-      <1>         mov     dword [u.error], ERR_DRV_NOT_RDY ; drive not ready !
21877 00005D86 0000                     <1>
21878 00005D88 E9ADE2FFFF               <1>       jmp    error
21879                                  <1> sysmnt_1:
21880 00005D8D 8F05[60740000]           <1>       pop    dword [u.namep]
21881                                  <1>             ; mov (sp)+,u.namep / put the name of file to be placed
21882                                  <1>                                ; / on the device
21883                                  <1>       ; 14/11/2015
21884 00005D93 53                       <1>       push   ebx ; 13/10/2015
21885                                  <1>             ; mov r1,-(sp) / save the device number
21886                                  <1>         ;
21887 00005D94 E85DF1FFFF               <1>       call   namei
21888                                  <1>       ;or    ax, ax ; Retro UNIX 8086 v1 modification !
21889                                  <1>                     ; ax = 0 -> file not found
21890                                  <1>       ;jz    error
21891                                  <1>       ;jc    error
21892                                  <1>             ; jsr r0,namei / get the i-number of the file
21893                                  <1>                    ; br errora
21894 00005D99 730F                     <1>       jnc    short sysmnt_2
21895                                  <1> sysmnt_err2:
21896 00005D9B C705[9D740000]0C00-      <1>         mov     dword [u.error], ERR_FILE_NOT_FOUND ; drive not ready !
21897 00005DA3 0000                     <1>
21898 00005DA5 E990E2FFFF               <1>       jmp    error
21899                                  <1> sysmnt_2:
21900 00005DAA 66A3[34740000]           <1>       mov    [mnti], ax
```

```
21901                                  <1>             ; mov r1,mnti / put it in mnti
21902                                  <1> ;     mov   ebx, sb1 ; super block buffer (of mounted disk)
21903                                  <1> sysmnt_3: ;1:
21904                                  <1>           ;cmp byte [ebx+1], 0
21905                                  <1>             ; tstb sb1+1 / is 15th bit of I/O queue entry for
21906                                  <1>                          ; / dismountable device set?
21907                                  <1>           ;jna short sysmnt_4
21908                                  <1>             ; bne 1b / (inhibit bit) yes, skip writing
21909                                  <1>           ;call idle  ; (wait for hardware interrupt)
21910                                  <1>           ;jmp  short sysmnt_3
21911                                  <1> sysmnt_4:
21912 00005DB0 58                     <1>           pop   eax ; Retro UNIX 8086 v1 device number/ID (0 to 5)
21913 00005DB1 A2[31740000]           <1>           mov   [mdev], al
21914                                  <1>             ; mov  (sp),mntd / no, put the device number in mntd
21915 00005DB6 8803                   <1>           mov   [ebx], al
21916                                  <1>             ; movb (sp),sb1 / put the device number in the lower byte
21917                                  <1>                           ; / of the I/O queue entry
21918                                  <1>           ;mov  byte [cdev], 1 ; mounted device/drive
21919                                  <1>             ; mov (sp)+,cdev / put device number in cdev
21920 00005DB8 66810B0004             <1>           or    word [ebx], 400h ; Bit 10, 'read' flag/bit
21921                                  <1>             ; bis $2000,sb1 / set the read bit
21922                                  <1>           ; Retro UNIX 386 v1 modification :
21923                                  <1>           ;     32 bit block number at buffer header offset 4
21924 00005DBD C7430401000000         <1>           mov   dword [ebx+4], 1 ; physical block number = 1
21925 00005DC4 E8A3050000             <1>           call  diskio
21926 00005DC9 731C                   <1>           jnc   short sysmnt_5
21927 00005DCB 31C0                   <1>           xor   eax, eax
21928 00005DCD 66A3[34740000]         <1>           mov   [mnti], ax ; 0
21929 00005DD3 A2[31740000]           <1>           mov   [mdev], al ; 0
21930                                  <1>           ;mov  [cdev], al ; 0
21931                                  <1> sysmnt_invd:
21932                                  <1>           ; 14/11/2015
21933 00005DD8 FEC8                   <1>           dec   al
21934 00005DDA 8903                   <1>           mov   [ebx], eax ; 000000FFh
21935 00005DDC FEC0                   <1>           inc   al
21936 00005DDE 48                     <1>           dec   eax
21937 00005DDF 894304                 <1>           mov   [ebx+4], eax ; 0FFFFFFFFh
21938 00005DE2 E953E2FFFF             <1>           jmp   error
21939                                  <1> sysmnt_5:
21940                                  <1>           ; 14/11/2015 (Retro UNIX 386 v1 modification)
21941                                  <1>           ; (Following check is needed to prevent mounting an
21942                                  <1>           ; in valid valid file system (in valid super block).
21943                                  <1>           ;
21944 00005DE7 0FB603                 <1>           movzx eax, byte [ebx] ; device number
21945 00005DEA C0E002                 <1>           shl   al, 2 ; 4*index
21946 00005DED 8B88[4E6B0000]         <1>           mov   ecx, [eax+drv.size] ; volume (fs) size
21947 00005DF3 C1E103                 <1>           shl   ecx, 3
21948 00005DF6 0FB715[0C830000]       <1>           movzx edx, word [sb1+4] ; the 1st data word
21949 00005DFD 39D1                   <1>           cmp   ecx, edx ; compare free map bits and volume size
21950                                  <1>                          ; (in sectors), if they are not equal
21951                                  <1>                          ; the disk to be mounted is an...
21952 00005DFF 75D7                   <1>           jne   short sysmnt_invd ; invalid disk !
21953                                  <1>                          ; (which has not got a valid super block)
21954                                  <1>           ;
21955 00005E01 C6430100               <1>           mov   byte [ebx+1], 0
21956                                  <1>                         ; jsr r0,ppoke / read in entire file system
21957                                  <1> ;sysmnt_6: ;1:
21958                                  <1>           ;;cmp byte [sb1+1], 0
21959                                  <1>                 ; tstb   sb1+1 / done reading?
21960                                  <1>           ;;jna sysret
21961                                  <1>           ;;call idle ; (wait for hardware interrupt)
21962                                  <1>           ;;jmp short sysmnt_6
21963                                  <1>                 ;bne 1b / no, wait
21964                                  <1>                 ;br sysreta / yes
21965 00005E05 E950E2FFFF             <1>           jmp   sysret
21966                                  <1>
21967                                  <1> sysumount: ; / special dismount file system
21968                                  <1>           ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
21969                                  <1>           ; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
21970                                  <1>           ;
21971                                  <1>           ; 04/11/2013
21972                                  <1>           ; 09/07/2013
21973                                  <1>           ; 'sysumount' anounces to the system that the special file,
21974                                  <1>           ; indicated as an argument is no longer contain a removable
21975                                  <1>           ; file system. 'getspl' gets the device number of the special
21976                                  <1>           ; file. If no file system was mounted on that device an error
21977                                  <1>           ; occurs. 'mntd' and 'mnti' are cleared and control is passed
21978                                  <1>           ; to 'sysret'.
21979                                  <1>           ;
21980                                  <1>           ; Calling sequence:
21981                                  <1>           ;     sysmount; special
21982                                  <1>           ; Arguments:
21983                                  <1>           ;     special - special file to dismount (device)
21984                                  <1>           ;
21985                                  <1>           ; Inputs: -
21986                                  <1>           ; Outputs: -
21987                                  <1>           ; ..........................................................
21988                                  <1>           ;
21989                                  <1>           ; Retro UNIX 8086 v1 modification:
21990                                  <1>           ;      'sysumount' system call has one argument; so,
21991                                  <1>           ;     * Single argument, special is pointed to by BX register
21992                                  <1>           ;
21993                                  <1>
21994                                  <1>           ;mov  ax, 1 ; one/single argument, put argument in BX
21995                                  <1>           ;call arg
21996                                  <1>                 ; jsr r0,arg; u.namep / point u.namep to special
21997 00005E0A 891D[60740000]         <1>             mov [u.namep], ebx
21998 00005E10 E829000000             <1>           call  getspl
21999                                  <1>                 ; jsr r0,getspl / get the device number in r1
22000 00005E15 3A05[31740000]         <1>           cmp   al, [mdev]
22001                                  <1>                 ; cmp r1,mntd / is it equal to the last device mounted?
22002 00005E1B 7539                   <1>           jne   short sysmnt_err0 ; 'permission denied !' error
22003                                  <1>           ;jne  error
22004                                  <1>                 ; bne errora / no error
22005 00005E1D 30C0                   <1>           xor   al, al ; ah = 0
```

```
22006                                  <1> sysumnt_0: ;1:
22007 00005E1F 3805[09830000]          <1>       cmp   [sb1+1], al ; 0
22008                                  <1>             ; tstb sb1+1 / yes, is the device still doing I/O
22009                                  <1>                       ; / (inhibit bit set)?
22010 00005E25 7607                    <1>       jna   short sysumnt_1
22011                                  <1>             ; bne 1b / yes, wait
22012 00005E27 E82DF6FFFF              <1>       call  idle ; (wait for hardware interrupt)
22013 00005E2C EBF1                    <1>       jmp   short sysumnt_0
22014                                  <1> sysumnt_1:
22015 00005E2E A2[31740000]            <1>       mov   [mdev], al
22016                                  <1>             ; clr mntd / no, clear these
22017 00005E33 66A3[34740000]          <1>       mov   [mnti], ax
22018                                  <1>             ; clr mnti
22019 00005E39 E91CE2FFFF              <1>         jmp sysret
22020                                  <1>             ; br sysreta / return
22021                                  <1>
22022                                  <1> getspl: ; / get device number from a special file name
22023 00005E3E E8B3F0FFFF              <1>       call  namei
22024                                  <1>       ;or   ax, ax ; Retro UNIX 8086 v1 modification !
22025                                  <1>                    ; ax = 0 -> file not found
22026 00005E43 0F8252FFFFFF            <1>         jc      sysmnt_err2 ; 'file not found !' error
22027                                  <1>       ;jz   error
22028                                  <1>       ;jc   error
22029                                  <1>             ; jsr r0,namei / get the i-number of the special file
22030                                  <1>                ; br errora / no such file
22031 00005E49 6683E803                <1>         sub ax, 3 ; Retro UNIX 8086 v1 modification !
22032                                  <1>                   ;      i-number-3, 0 = fd0, 5 = hd3
22033                                  <1>             ; sub $4,r1 / i-number-4 rk=1,tap=2+n
22034 00005E4D 7207                    <1>         jc  short sysmnt_err0 ; 'permission denied !' error
22035                                  <1>       ;jc   error
22036                                  <1>             ; ble errora / less than 0?  yes, error
22037 00005E4F 6683F805                <1>         cmp ax, 5 ;
22038                                  <1>             ; cmp  r1,$9. / greater than 9  tap 7
22039 00005E53 7701                    <1>       ja    short sysmnt_err0 ; 'permission denied !' error
22040                                  <1>       ;ja   error
22041                                  <1>             ; bgt errora / yes, error
22042                                  <1>           ; AX = Retro UNIX 8086 v1 Device Number (0 to 5)
22043                                  <1> iopen_retn:
22044 00005E55 C3                      <1>       retn
22045                                  <1>             ; rts    r0 / return with device number in r1
22046                                  <1> sysmnt_err0:
22047 00005E56 C705[9D740000]0B00-     <1>       mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
22048 00005E5E 0000                    <1>
22049 00005E60 E9D5E1FFFF              <1>       jmp   error
22050                                  <1> iopen:
22051                                  <1>       ; 19/05/2015
22052                                  <1>       ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22053                                  <1>       ; 21/05/2013 - 27/08/2013 (Retro UNIX 8086 v1)
22054                                  <1>       ;
22055                                  <1>       ; open file whose i-number is in r1
22056                                  <1>       ;
22057                                  <1>       ; INPUTS ->
22058                                  <1>       ;    r1 - inode number
22059                                  <1>       ; OUTPUTS ->
22060                                  <1>       ;    file's inode in core
22061                                  <1>       ;    r1 - inode number (positive)
22062                                  <1>       ;
22063                                  <1>       ; ((AX = R1))
22064                                  <1>         ; ((Modified registers: edx, ebx, ecx, esi, edi, ebp))
22065                                  <1>       ;
22066                                  <1> ; / open file whose i-number is in r1
22067 00005E65 F6C480                  <1>       test  ah, 80h ; Bit 15 of AX
22068                                  <1>             ;tst r1 / write or read access?
22069 00005E68 756A                    <1>         jnz short iopen_2
22070                                  <1>             ;blt 2f / write, go to 2f
22071 00005E6A B202                    <1>       mov   dl, 2 ; read access
22072 00005E6C E850F9FFFF              <1>       call  access
22073                                  <1>             ; jsr r0,access; 2
22074                                  <1>       ; / get inode into core with read access
22075                                  <1>       ; DL=2
22076                                  <1> iopen_0:
22077 00005E71 6683F828                <1>         cmp ax, 40
22078                                  <1>             ; cmp r1,$40. / is it a special file
22079 00005E75 77DE                    <1>         ja  short iopen_retn
22080                                  <1>             ;bgt  3f / no. 3f
22081 00005E77 6650                    <1>       push  ax
22082                                  <1>             ; mov r1,-(sp) / yes, figure out
22083 00005E79 0FB6D8                  <1>       movzx ebx, al
22084 00005E7C 66C1E302                <1>       shl   bx, 2
22085                                  <1>             ; asl r1
22086 00005E80 81C3[845E0000]          <1>         add     ebx, iopen_1 - 4
22087 00005E86 FF23                    <1>       jmp   dword [ebx]
22088                                  <1>             ; jmp *1f-2(r1) / which one and transfer to it
22089                                  <1> iopen_1: ; 1:
22090 00005E88 [EE5E0000]              <1>       dd    otty ; tty, AX = 1 (runix)
22091                                  <1>              ;otty / tty ; r1=2
22092                                  <1>              ;oppt / ppt ; r1=4
22093 00005E8C [8B5F0000]              <1>       dd    sret ; mem, AX = 2 (runix)
22094                                  <1>              ;sret / mem ; r1=6
22095                                  <1>              ;sret / rf0
22096                                  <1>              ;sret / rk0
22097                                  <1>              ;sret / tap0
22098                                  <1>              ;sret / tap1
22099                                  <1>              ;sret / tap2
22100                                  <1>              ;sret / tap3
22101                                  <1>              ;sret / tap4
22102                                  <1>              ;sret / tap5
22103                                  <1>              ;sret / tap6
22104                                  <1>              ;sret / tap7
22105 00005E90 [8B5F0000]              <1>         dd  sret ; fd0, AX = 3 (runix only)
22106 00005E94 [8B5F0000]              <1>         dd  sret ; fd1, AX = 4 (runix only)
22107 00005E98 [8B5F0000]              <1>         dd  sret ; hd0, AX = 5 (runix only)
22108 00005E9C [8B5F0000]              <1>         dd  sret ; hd1, AX = 6 (runix only)
22109 00005EA0 [8B5F0000]              <1>         dd  sret ; hd2, AX = 7 (runix only)
22110 00005EA4 [8B5F0000]              <1>         dd  sret ; hd3, AX = 8 (runix only)
```

```
22111                                  <1>       ;dd    error ; lpr, AX = 9 (error !)
22112 00005EA8 [8B5F0000]              <1>         dd      sret ; lpr, AX = 9 (runix)
22113 00005EAC [FD5E0000]              <1>       dd     ocvt ; tty0, AX = 10 (runix)
22114                                  <1>              ;ocvt / tty0
22115 00005EB0 [FD5E0000]              <1>       dd     ocvt ; tty1, AX = 11 (runix)
22116                                  <1>              ;ocvt / tty1
22117 00005EB4 [FD5E0000]              <1>       dd     ocvt ; tty2, AX = 12 (runix)
22118                                  <1>              ;ocvt / tty2
22119 00005EB8 [FD5E0000]              <1>       dd     ocvt ; tty3, AX = 13 (runix)
22120                                  <1>              ;ocvt / tty3
22121 00005EBC [FD5E0000]              <1>       dd     ocvt ; tty4, AX = 14 (runix)
22122                                  <1>              ;ocvt / tty4
22123 00005EC0 [FD5E0000]              <1>       dd     ocvt ; tty5, AX = 15 (runix)
22124                                  <1>              ;ocvt / tty5
22125 00005EC4 [FD5E0000]              <1>       dd     ocvt ; tty6, AX = 16 (runix)
22126                                  <1>              ;ocvt / tty6
22127 00005EC8 [FD5E0000]              <1>       dd     ocvt ; tty7, AX = 17 (runix)
22128                                  <1>              ;ocvt / tty7
22129 00005ECC [FD5E0000]              <1>       dd     ocvt ; COM1, AX = 18 (runix only)
22130                                  <1>              ;error / crd
22131 00005ED0 [FD5E0000]              <1>       dd     ocvt ; COM2, AX = 19 (runix only)
22132                                  <1>
22133                                  <1> iopen_2: ; 2: / check open write access
22134 00005ED4 66F7D8                  <1>       neg    ax
22135                                  <1>              ;neg r1 / make inode number positive
22136 00005ED7 B201                    <1>       mov    dl, 1 ; write access
22137 00005ED9 E8E3F8FFFF              <1>       call   access
22138                                  <1>              ;jsr r0,access; 1 / get inode in core
22139                                  <1>       ; DL=1
22140 00005EDE 66F705[16710000]00-     <1>       test   word [i.flgs], 4000h ; Bit 14 : Directory flag
22141 00005EE6 40                      <1>
22142                                  <1>              ;bit $40000,i.flgs / is it a directory?
22143 00005EE7 7488                    <1>       jz     short iopen_0
22144                                  <1>       ;mov   [u.error], ERR_DIR_ACCESS
22145                                  <1>       ;jmp   error ; permission denied !
22146 00005EE9 E968FFFFFF              <1>       jmp    sysmnt_err0
22147                                  <1>       ;;jnz  error
22148                                  <1>                      ; bne 2f / yes, transfer (error)
22149                                  <1>         ;;jmp     short iopen_0
22150                                  <1>       ;cmp   ax, 40
22151                                  <1>              ; cmp r1,$40. / no, is it a special file?
22152                                  <1>         ;ja  short iopen_2
22153                                  <1>               ;bgt 3f / no, return
22154                                  <1>       ;push  ax
22155                                  <1>              ;mov r1,-(sp) / yes
22156                                  <1>       ;movzx ebx, al
22157                                  <1>       ;shl   bx, 1
22158                                  <1>              ; asl r1
22159                                  <1>       ;add   ebx, ipen_3 - 2
22160                                  <1>       ;jmp   dword [ebx]
22161                                  <1>              ; jmp *1f-2(r1) / figure out
22162                                  <1>                     ; / which special file it is and transfer
22163                                  <1> ;iopen_3: ; 1:
22164                                  <1> ;     dd     otty ; tty, AX = 1 (runix)
22165                                  <1>               ;otty / tty ; r1=2
22166                                  <1>               ;leadr / ppt ; r1=4
22167                                  <1> ;     dd     sret ; mem, AX = 2 (runix)
22168                                  <1>               ;sret / mem ; r1=6
22169                                  <1>               ;sret / rf0
22170                                  <1>               ;sret / rk0
22171                                  <1>               ;sret / tap0
22172                                  <1>               ;sret / tap1
22173                                  <1>               ;sret / tap2
22174                                  <1>               ;sret / tap3
22175                                  <1>               ;sret / tap4
22176                                  <1>               ;sret / tap5
22177                                  <1>               ;sret / tap6
22178                                  <1>               ;sret / tap7
22179                                  <1> ;     dd     sret ; fd0, AX = 3 (runix only)
22180                                  <1> ;     dd     sret ; fd1, AX = 4 (runix only)
22181                                  <1> ;     dd     sret ; hd0, AX = 5 (runix only)
22182                                  <1> ;     dd     sret ; hd1, AX = 6 (runix only)
22183                                  <1> ;     dd     sret ; hd2, AX = 7 (runix only)
22184                                  <1> ;     dd     sret ; hd3, AX = 8 (runix only)
22185                                  <1> ;     dd     sret ; lpr, AX = 9  (runix)
22186                                  <1>       ;dd    ejec ; lpr, AX = 9  (runix)
22187                                  <1> ;     dd     sret ; tty0, AX = 10 (runix)
22188                                  <1>               ;ocvt / tty0
22189                                  <1> ;     dd     sret ; tty1, AX = 11 (runix)
22190                                  <1>               ;ocvt / tty1
22191                                  <1> ;     dd     sret ; tty2, AX = 12 (runix)
22192                                  <1>               ;ocvt / tty2
22193                                  <1> ;     dd     sret ; tty3, AX = 13 (runix)
22194                                  <1>               ;ocvt / tty3
22195                                  <1> ;     dd     sret ; tty4, AX = 14 (runix)
22196                                  <1>               ;ocvt / tty4
22197                                  <1> ;     dd     sret ; tty5, AX = 15 (runix)
22198                                  <1>               ;ocvt / tty5
22199                                  <1> ;     dd     sret ; tty6, AX = 16 (runix)
22200                                  <1>               ;ocvt / tty6
22201                                  <1> ;     dd     sret ; tty7, AX = 17 (runix)
22202                                  <1>               ;ocvt / tty7
22203                                  <1> ;     dd     ocvt ; COM1, AX = 18 (runix only)
22204                                  <1>               ;/ ejec / lpr
22205                                  <1> ;     dd     ocvt ; COM2, AX = 19 (runix only)
22206                                  <1>
22207                                  <1>
22208                                  <1> otty: ;/ open console tty for reading or writing
22209                                  <1>       ; 16/11/2015
22210                                  <1>       ; 12/11/2015
22211                                  <1>       ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22212                                  <1>       ; 21/05/2013 - 13/07/2014 (Retro UNIX 8086 v1)
22213                                  <1>       ; 16/07/2013
22214                                  <1>       ; Retro UNIX 8086 v1 modification:
22215                                  <1>       ;  If a tty is open for read or write by
```

```
22216                                  <1>       ;     a process (u.uno), only same process can open
22217                                  <1>       ;     same tty to write or read (R->R&W or W->W&R).
22218                                  <1>       ;
22219                                  <1>       ; (INPUT: DL=2 for Read, DL=1 for Write, DL=0 for sysstty)
22220                                  <1>       ;
22221 00005EEE 0FB61D[97740000]       <1>       movzx  ebx, byte [u.uno] ; process number
22222 00005EF5 8A83[95710000]         <1>       mov    al, [ebx+p.ttyc-1] ; current/console tty
22223                                  <1>       ; 13/01/2014
22224 00005EFB EB02                   <1>       jmp    short ottyp
22225                                  <1> ocvt:
22226 00005EFD 2C0A                   <1>       sub    al, 10
22227                                  <1> ottyp:
22228                                  <1>       ; 16/11/2015
22229                                  <1>       ; 12/11/2015
22230                                  <1>       ; 18/05/2015 (32 bit modifications)
22231                                  <1>       ; 06/12/2013 - 13/07/2014
22232 00005EFF 88C6                   <1>       mov    dh, al ; tty number
22233 00005F01 0FB6D8                 <1>       movzx  ebx, al ; AL = tty number (0 to 9), AH = 0
22234 00005F04 D0E3                   <1>       shl    bl, 1  ; aligned to word
22235                                  <1>       ;26/01/2014
22236 00005F06 81C3[B4700000]         <1>       add    ebx, ttyl
22237 00005F0C 668B0B                 <1>       mov    cx, [ebx]
22238                                  <1>              ; CL = lock value (0 or process number)
22239                                  <1>              ; CH = open count
22240 00005F0F 20C9                   <1>       and    cl, cl
22241                                  <1>       ; 13/01/2014
22242 00005F11 7439                   <1>       jz     short otty_ret
22243                                  <1>       ;
22244                                  <1>       ; 16/11/2015
22245 00005F13 3A0D[97740000]         <1>       cmp    cl, [u.uno]
22246 00005F19 745C                   <1>       je     short ottys_3
22247                                  <1>       ;
22248 00005F1B 0FB6D9                 <1>       movzx  ebx, cl ; the process which has locked the tty
22249 00005F1E D0E3                   <1>       shl    bl, 1
22250 00005F20 668B83[34710000]       <1>       mov    ax, [ebx+p.pid-2]
22251                                  <1>       ;movzx       ebx, byte [u.uno]
22252 00005F27 8A1D[97740000]         <1>       mov    bl, [u.uno]
22253 00005F2D D0E3                   <1>       shl    bl, 1
22254 00005F2F 663B83[54710000]       <1>       cmp    ax, [ebx+p.ppid-2]
22255 00005F36 743F                   <1>       je     short ottys_3  ; 16/11/2015
22256                                  <1>       ;
22257                                  <1>       ; the tty is locked by another process
22258                                  <1>       ; except the parent process (p.ppid)
22259                                  <1>         ;
22260 00005F38 C705[9D740000]0B00-     <1>       mov     dword [u.error], ERR_DEV_ACCESS
22261 00005F40 0000                   <1>
22262                                  <1>                     ; permission denied ! error
22263                                  <1> otty_err: ; 13/01/2014
22264 00005F42 08D2                   <1>       or     dl, dl ; DL = 0 -> called by sysstty
22265 00005F44 0F85F0E0FFFF           <1>       jnz    error
22266 00005F4A F9                     <1>       stc
22267 00005F4B C3                     <1>       retn
22268                                  <1> otty_ret:
22269                                  <1>       ; 13/01/2014
22270 00005F4C 80FE07                 <1>       cmp    dh, 7
22271 00005F4F 761C                   <1>       jna    short ottys_2
22272                                  <1>       ; 16/11/2015
22273                                  <1> com_port_check:
22274 00005F51 BE[D2700000]           <1>       mov    esi, com1p
22275 00005F56 80FE08                 <1>       cmp    dh, 8  ; COM1 (tty8) ?
22276 00005F59 7601                   <1>       jna    short ottys_1 ; yes, it is COM1
22277 00005F5B 46                     <1>       inc    esi    ; no, it is COM2 (tty9)
22278                                  <1> ottys_1:
22279                                  <1>       ; 12/11/2015
22280 00005F5C 803E00                 <1>       cmp    byte [esi], 0 ; E3h (or 23h)
22281 00005F5F 770C                   <1>       ja     short com_port_ready
22282                                  <1>       ;
22283 00005F61 C705[9D740000]0F00-     <1>         mov     dword [u.error], ERR_DEV_NOT_RDY
22284 00005F69 0000                   <1>
22285                                  <1>                       ; device not ready ! error
22286 00005F6B EBD5                   <1>       jmp    short otty_err
22287                                  <1> com_port_ready:
22288                                  <1> ottys_2:
22289 00005F6D 08C9                   <1>       or     cl, cl  ; cl = lock/owner, ch = open count
22290 00005F6F 7506                   <1>       jnz    short ottys_3
22291 00005F71 8A0D[97740000]         <1>       mov    cl, [u.uno]
22292                                  <1> ottys_3:
22293 00005F77 FEC5                   <1>       inc    ch
22294 00005F79 66890B                 <1>       mov    [ebx], cx ; set tty lock again
22295                                  <1>       ; 06/12/2013
22296 00005F7C FEC6                   <1>       inc    dh ; tty number + 1
22297 00005F7E BB[78740000]           <1>       mov    ebx, u.ttyp
22298                                  <1>       ; 13/01/2014
22299 00005F83 F6C202                 <1>       test   dl, 2 ; open for read sign
22300 00005F86 7501                   <1>       jnz    short ottys_4
22301 00005F88 43                     <1>       inc    ebx
22302                                  <1> ottys_4:
22303                                  <1>       ; Set 'u.ttyp' ('the recent TTY') value
22304 00005F89 8833                   <1>       mov    [ebx], dh ; tty number + 1
22305                                  <1> sret:
22306 00005F8B 08D2                   <1>       or     dl, dl ; sysstty system call check (DL=0)
22307 00005F8D 7402                   <1>       jz     short iclose_retn
22308 00005F8F 6658                   <1>       pop    ax
22309                                  <1> iclose_retn:
22310 00005F91 C3                     <1>       retn
22311                                  <1>
22312                                  <1>       ;
22313                                  <1>       ; Original UNIX v1 'otty' routine:
22314                                  <1>       ;
22315                                  <1>       ;mov    $100,*$tks / set interrupt enable bit (zero others) in
22316                                  <1>         ;                  / reader status reg
22317                                  <1>         ;mov    $100,*$tps / set interrupt enable bit (zero others) in
22318                                  <1>         ;                  / punch status reg
22319                                  <1>         ;mov    tty+[ntty*8]-8+6,r5 / r5 points to the header of the
22320                                  <1>         ;                         / console tty buffer
```

```
22321                           <1>         ;incb   (r5) / increment the count of processes that opened the
22322                           <1>         ;           / console tty
22323                           <1>         ;tst    u.ttyp / is there a process control tty (i.e., has a tty
22324                           <1>         ;             / buffer header
22325                           <1>         ;bne    sret / address been loaded into u.ttyp yet)?  yes, branch
22326                           <1>         ;mov    r5,u.ttyp / no, make the console tty the process control
22327                           <1>         ;                  / tty
22328                           <1>         ;br     sret / ?
22329                           <1> ;sret:
22330                           <1>             ;clr *$ps / set processor priority to zero
22331                           <1> ;     pop    ax
22332                           <1>             ;mov (sp)+,r1 / pop stack to r1
22333                           <1> ;3:
22334                           <1> ;     retn
22335                           <1>             ;rts r0
22336                           <1>
22337                           <1> ;ocvt:      ; < open tty >
22338                           <1>       ; 13/01/2014
22339                           <1>       ; 06/12/2013 (major modification: p.ttyc, u.ttyp)
22340                           <1>       ; 24/09/2013 consistency check -> ok
22341                           <1>       ; 16/09/2013
22342                           <1>       ; 03/09/2013
22343                           <1>       ; 27/08/2013
22344                           <1>       ; 16/08/2013
22345                           <1>       ; 16/07/2013
22346                           <1>       ; 27/05/2013
22347                           <1>       ; 21/05/2013
22348                           <1>       ;
22349                           <1>       ; Retro UNIX 8086 v1 modification !
22350                           <1>       ;
22351                           <1>       ; In original UNIX v1, 'ocvt' routine
22352                           <1>       ;            (exactly different than this one)
22353                           <1>       ;     was in 'u9.s' file.
22354                           <1>       ;
22355                           <1>       ; 16/07/2013
22356                           <1>       ; Retro UNIX 8086 v1 modification:
22357                           <1>       ;  If a tty is open for read or write by
22358                           <1>       ;     a process (u.uno), only same process can open
22359                           <1>       ;     same tty to write or read (R->R&W or W->W&R).
22360                           <1>       ;
22361                           <1>       ; INPUT: DL=2 for Read DL=1 for Write
22362                           <1>
22363                           <1>       ; 16/09/2013
22364                           <1>       ; sub  al, 10
22365                           <1>
22366                           <1>       ; 06/12/2013
22367                           <1>       ;cmp   al, 7
22368                           <1>         ;jna     short ottyp
22369                           <1>       ; 13/01/2014
22370                           <1>       ;jmp   short ottyp
22371                           <1>
22372                           <1>
22373                           <1> ;oppt: / open paper tape for reading or writing
22374                           <1> ;       mov    $100,*$prs / set reader interrupt enable bit
22375                           <1> ;       tstb   pptiflg / is file already open
22376                           <1> ;       bne    2f / yes, branch
22377                           <1> ;1:
22378                           <1> ;       mov    $240,*$ps / no, set processor priority to 5
22379                           <1> ;       jsr    r0,getc; 2 / remove all entries in clist
22380                           <1> ;              br .+4 / for paper tape input and place in free list
22381                           <1> ;       br     1b
22382                           <1> ;       movb   $2,pptiflg / set pptiflg to indicate file just open
22383                           <1> ;       movb   $10.,toutt+1 / place 10 in paper tape input tout entry
22384                           <1> ;       br     sret
22385                           <1> ;2:
22386                           <1> ;       jmp    error / file already open
22387                           <1>
22388                           <1> iclose:
22389                           <1>       ; 19/05/2015
22390                           <1>       ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22391                           <1>       ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
22392                           <1>       ;
22393                           <1>       ; close file whose i-number is in r1
22394                           <1>       ;
22395                           <1>       ; INPUTS ->
22396                           <1>       ;    r1 - inode number
22397                           <1>       ; OUTPUTS ->
22398                           <1>       ;    file's inode in core
22399                           <1>       ;    r1 - inode number (positive)
22400                           <1>       ;
22401                           <1>       ; ((AX = R1))
22402                           <1>         ;    ((Modified registers: -ebx-, edx))
22403                           <1>       ;
22404                           <1> ;/ close file whose i-number is in r1
22405 00005F92 B202             <1>       mov    dl, 2 ; 12/01/2014
22406 00005F94 F6C480           <1>       test   ah, 80h ; Bit 15 of AX
22407                           <1>              ;tst r1 / test i-number
22408                           <1>         ;jnz short iclose_2
22409                           <1>              ;blt 2f / if neg., branch
22410 00005F97 7405             <1>       jz     short iclose_0 ; 30/07/2013
22411                           <1>       ; 16/07/2013
22412 00005F99 66F7D8           <1>       neg    ax ; make it positive
22413                           <1>       ; 12/01/2014
22414 00005F9C FECA             <1>       dec    dl ; dl = 1 (open for write)
22415                           <1> iclose_0:
22416 00005F9E 6683F828         <1>       cmp    ax, 40
22417                           <1>              ;cmp r1,$40. / is it a special file
22418 00005FA2 77ED             <1>         ja   short iclose_retn  ; 13/01/2014
22419                           <1>              ;bgt 3b / no, return
22420                           <1>       ; 12/01/2014
22421                           <1>       ; DL=2 -> special file was opened for reading
22422                           <1>       ; DL=1 -> special file was opened for writing
22423 00005FA4 6650             <1>       push   ax
22424                           <1>              ;mov r1,-(sp) / yes, save r1 on stack
22425 00005FA6 0FB6D8           <1>       movzx  ebx, al
```

```
22426 00005FA9 66C1E302            <1>       shl    bx, 2
22427                              <1>              ; asl r1
22428 00005FAD 81C3[B15F0000]      <1>       add    ebx, iclose_1 - 4
22429 00005FB3 FF23                <1>       jmp    dword [ebx]
22430                              <1>              ; jmp *1f-2(r1) / compute jump address and transfer
22431                              <1> iclose_1 :
22432 00005FB5 [01600000]          <1>       dd     ctty ; tty, AX = 1 (runix)
22433 00005FB9 [52600000]          <1>       dd     cret ; mem, AX = 2 (runix)
22434 00005FBD [52600000]          <1>       dd     cret ; fd0, AX = 3 (runix only)
22435 00005FC1 [52600000]          <1>       dd     cret ; fd1, AX = 4 (runix only)
22436 00005FC5 [52600000]          <1>       dd     cret ; hd0, AX = 5 (runix only)
22437 00005FC9 [52600000]          <1>       dd     cret ; hd1, AX = 6 (runix only)
22438 00005FCD [52600000]          <1>       dd     cret ; hd2, AX = 7 (runix only)
22439 00005FD1 [52600000]          <1>       dd     cret ; hd3, AX = 8 (runix only)
22440 00005FD5 [52600000]          <1>       dd     cret ; lpr, AX = 9 (runix)
22441                              <1>       ;dd    error; lpr, AX = 9 (error !)
22442                              <1>       ;;dd   offset ejec ;;lpr, AX = 9
22443 00005FD9 [10600000]          <1>       dd     ccvt ; tty0, AX = 10 (runix)
22444 00005FDD [10600000]          <1>       dd     ccvt ; tty1, AX = 11 (runix)
22445 00005FE1 [10600000]          <1>       dd     ccvt ; tty2, AX = 12 (runix)
22446 00005FE5 [10600000]          <1>       dd     ccvt ; tty3, AX = 13 (runix)
22447 00005FE9 [10600000]          <1>       dd     ccvt ; tty4, AX = 14 (runix)
22448 00005FED [10600000]          <1>       dd     ccvt ; tty5, AX = 15 (runix)
22449 00005FF1 [10600000]          <1>       dd     ccvt ; tty6, AX = 16 (runix)
22450 00005FF5 [10600000]          <1>       dd     ccvt ; tty7, AX = 17 (runix)
22451 00005FF9 [10600000]          <1>       dd     ccvt ; COM1, AX = 18 (runix only)
22452 00005FFD [10600000]          <1>       dd     ccvt ; COM2, AX = 19 (runix only)
22453                              <1>
22454                              <1>       ; 1:
22455                              <1>       ;        ctty   / tty
22456                              <1>       ;        cppt   / ppt
22457                              <1>       ;        sret   / mem
22458                              <1>       ;        sret   / rf0
22459                              <1>       ;        sret   / rk0
22460                              <1>       ;        sret   / tap0
22461                              <1>       ;        sret   / tap1
22462                              <1>       ;        sret   / tap2
22463                              <1>       ;        sret   / tap3
22464                              <1>       ;        sret   / tap4
22465                              <1>       ;        sret   / tap5
22466                              <1>       ;        sret   / tap6
22467                              <1>       ;        sret   / tap7
22468                              <1>       ;        ccvt   / tty0
22469                              <1>       ;        ccvt   / tty1
22470                              <1>       ;        ccvt   / tty2
22471                              <1>       ;        ccvt   / tty3
22472                              <1>       ;        ccvt   / tty4
22473                              <1>       ;        ccvt   / tty5
22474                              <1>       ;        ccvt   / tty6
22475                              <1>       ;        ccvt   / tty7
22476                              <1>       ;        error / crd
22477                              <1>
22478                              <1> ;iclose_2: ; 2: / negative i-number
22479                              <1>       ;neg   ax
22480                              <1>              ;neg r1 / make it positive
22481                              <1>       ;cmp   ax, 40
22482                              <1>              ;cmp r1,$40. / is it a special file?
22483                              <1>         ;ja  short @b
22484                              <1>              ;bgt    3b / no. return
22485                              <1>       ;push  ax
22486                              <1>              ;mov r1,-(sp)
22487                              <1>       ;movzx ebx, al
22488                              <1>       ;shl   bx, 1
22489                              <1>              ;asl r1 / yes. compute jump address and transfer
22490                              <1>       ;add   ebx, iclose_3 - 2
22491                              <1>       ;jmp   dword [ebx]
22492                              <1>              ;jmp *1f-2(r1) / figure out
22493                              <1> ;iclose_3:
22494                              <1>       ;dd    ctty ; tty, AX = 1 (runix)
22495                              <1>       ;dd    sret ; mem, AX = 2 (runix)
22496                              <1>       ;dd    sret ; fd0, AX = 3 (runix only)
22497                              <1>       ;dd    sret ; fd1, AX = 4 (runix only)
22498                              <1>       ;dd    sret ; hd0, AX = 5 (runix only)
22499                              <1>       ;dd    sret ; hd1, AX = 6 (runix only)
22500                              <1>       ;dd    sret ; hd2, AX = 7 (runix only)
22501                              <1>       ;dd    sret ; hd3, AX = 8 (runix only)
22502                              <1>        ;dd   sret ; lpr, AX = 9
22503                              <1>       ;dd    ejec ; lpr, AX = 9  (runix)
22504                              <1>       ;dd    ccvt ; tty0, AX = 10 (runix)
22505                              <1>       ;dd    ccvt ; tty1, AX = 11 (runix)
22506                              <1>       ;dd    ccvt ; tty2, AX = 12 (runix)
22507                              <1>       ;dd    ccvt ; tty3, AX = 13 (runix)
22508                              <1>       ;dd    ccvt ; tty4, AX = 14 (runix)
22509                              <1>       ;dd    ccvt ; tty5, AX = 15 (runix)
22510                              <1>       ;dd    ccvt ; tty6, AX = 16 (runix)
22511                              <1>       ;dd    ccvt ; tty7, AX = 17 (runix)
22512                              <1>       ;dd    ccvt ; COM1, AX = 18 (runix only)
22513                              <1>       ;dd    ccvt ; COM2, AX = 19 (runix only)
22514                              <1>
22515                              <1>       ;1:
22516                              <1>       ;            ctty   / tty
22517                              <1>       ;       leadr  / ppt
22518                              <1>       ;       sret   / mem
22519                              <1>       ;       sret   / rf0
22520                              <1>       ;       sret   / rk0
22521                              <1>       ;       sret   / tap0
22522                              <1>       ;       sret   / tap1
22523                              <1>       ;       sret   / tap2
22524                              <1>       ;       sret   / tap3
22525                              <1>       ;       sret   / tap4
22526                              <1>       ;       sret   / tap5
22527                              <1>       ;       sret   / tap6
22528                              <1>       ;       sret   / tap7
22529                              <1>       ;       ccvt   / tty0
22530                              <1>       ;       ccvt   / tty1
```

```
22531                                  <1>       ;       ccvt   / tty2
22532                                  <1>       ;       ccvt   / tty3
22533                                  <1>       ;       ccvt   / tty4
22534                                  <1>       ;       ccvt   / tty5
22535                                  <1>       ;       ccvt   / tty6
22536                                  <1>       ;       ccvt   / tty7
22537                                  <1>       ;/       ejec / lpr
22538                                  <1>
22539                                  <1> ctty: ; / close console tty
22540                                  <1>       ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22541                                  <1>       ; 21/05/2013 - 26/01/2014 (Retro UNIX 8086 v1)
22542                                  <1>       ;
22543                                  <1>       ; Retro UNIX 8086 v1 modification !
22544                                  <1>       ; (DL = 2 -> it is open for reading)
22545                                  <1>       ; (DL = 1 -> it is open for writing)
22546                                  <1>       ; (DL = 0 -> it is open for sysstty system call)
22547                                  <1>       ;
22548                                  <1>       ; 06/12/2013
22549 00006001 0FB61D[97740000]       <1>         movzx   ebx, byte [u.uno] ; process number
22550 00006008 8A83[95710000]         <1>         mov     al, [ebx+p.ttyc-1]
22551                                  <1>       ; 13/01/2014
22552 0000600E EB02                   <1>       jmp    short cttyp
22553                                  <1> ccvt:
22554 00006010 2C0A                   <1>       sub    al, 10
22555                                  <1> cttyp:
22556                                  <1>       ; 18/05/2015 (32 bit modifications)
22557                                  <1>       ; 16/08/2013 - 26/01/2014
22558 00006012 0FB6D8                 <1>       movzx  ebx, al ; tty number (0 to 9)
22559 00006015 D0E3                   <1>       shl    bl, 1  ; aligned to word
22560                                  <1>       ; 26/01/2014
22561 00006017 81C3[B4700000]         <1>       add    ebx, ttyl
22562 0000601D 88C6                   <1>       mov    dh, al ; tty number
22563 0000601F 668B03                 <1>       mov    ax, [ebx]
22564                                  <1>              ; AL = lock value (0 or process number)
22565                                  <1>              ; AH = open count
22566 00006022 20E4                   <1>       and    ah, ah
22567 00006024 750F                   <1>       jnz    short ctty_ret
22568 00006026 C705[9D740000]0A00-    <1>         mov     dword [u.error], ERR_DEV_NOT_OPEN
22569 0000602E 0000                   <1>
22570                                  <1>                     ; device not open ! error
22571                                  <1>       ;jmp   short ctty_err ; open count = 0, it is not open !
22572 00006030 E905E0FFFF             <1>       jmp    error
22573                                  <1>       ; 26/01/2014
22574                                  <1> ctty_ret:
22575 00006035 FECC                   <1>       dec    ah ; decrease open count
22576 00006037 7502                   <1>       jnz    short ctty_1
22577 00006039 30C0                   <1>       xor    al, al ; unlock/free tty
22578                                  <1> ctty_1:
22579 0000603B 668903                 <1>       mov    [ebx], ax ; close tty instance
22580                                  <1>       ;
22581 0000603E BB[78740000]           <1>       mov    ebx, u.ttyp
22582 00006043 F6C201                 <1>       test   dl, 1 ; open for write sign
22583 00006046 7401                   <1>       jz     short ctty_2
22584 00006048 43                     <1>       inc    ebx
22585                                  <1> ctty_2:
22586 00006049 FEC6                   <1>       inc    dh ; tty number + 1
22587 0000604B 3A33                   <1>       cmp    dh, [ebx]
22588 0000604D 7503                   <1>       jne    short cret
22589                                  <1>       ; Reset/Clear 'u.ttyp' ('the recent TTY') value
22590 0000604F C60300                 <1>       mov    byte [ebx], 0
22591                                  <1> cret:
22592 00006052 08D2                   <1>       or     dl, dl ; sysstty system call check (DL=0)
22593 00006054 7402                   <1>       jz     short ctty_3
22594 00006056 6658                   <1>       pop    ax
22595                                  <1> ctty_3:
22596 00006058 C3                     <1>       retn
22597                                  <1>
22598                                  <1> ;ctty_err: ; 13/01/2014
22599                                  <1> ;     or     dl, dl ; DL = 0 -> called by sysstty
22600                                  <1> ;     jnz    error
22601                                  <1> ;     stc
22602                                  <1> ;     retn
22603                                  <1>
22604                                  <1>
22605                                  <1>       ; Original UNIX v1 'ctty' routine:
22606                                  <1>       ;
22607                                  <1>         ;mov    tty+[ntty*8]-8+6,r5
22608                                  <1>       ;            ;/ point r5 to the console tty buffer
22609                                  <1>         ;decb   (r5) / dec number of processes using console tty
22610                                  <1>         ;br     sret / return via sret
22611                                  <1>
22612                                  <1> ;ccvt:      ; < close tty >
22613                                  <1>       ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
22614                                  <1>       ;
22615                                  <1>       ; Retro UNIX 8086 v1 modification !
22616                                  <1>       ;
22617                                  <1>       ; In original UNIX v1, 'ccvt' routine
22618                                  <1>       ;            (exactly different than this one)
22619                                  <1>       ;     was in 'u9.s' file.
22620                                  <1>       ;
22621                                  <1>       ; DL = 2 -> it is open for reading
22622                                  <1>       ; DL = 1 -> it is open for writing
22623                                  <1>       ;
22624                                  <1>       ; 17/09/2013
22625                                  <1>       ;sub   al, 10
22626                                  <1>       ;cmp   al, 7
22627                                  <1>       ;jna   short cttyp
22628                                  <1>       ; 13/01/2014
22629                                  <1>       ;jmp   short cttyp
22630                                  <1>
22631                                  <1> ;cppt: / close paper tape
22632                                  <1> ;       clrb   pptiflg / set pptiflg to indicate file not open
22633                                  <1> ;1:
22634                                  <1> ;       mov    $240,*$ps /set process or priority to 5
22635                                  <1> ;       jsr    r0,getc; 2 / remove all ppt input entries from clist
```

```
22636                                   <1> ;                           / and assign to free list
22637                                   <1> ;                br sret
22638                                   <1> ;        br     1b
22639                                   <1>
22640                                   <1> ;ejec:
22641                                   <1> ;     jmp    error
22642                                   <1> ;/ejec:
22643                                   <1> ;/       mov    $100,*$lps / set line printer interrupt enable bit
22644                                   <1> ;/       mov    $14,r1 / 'form feed' character in r1 (new page).
22645                                   <1> ;/       jsr    r0,lptoc / space the printer to a new page
22646                                   <1> ;/       br     sret / return to caller via 'sret'
22647                                       %include 'u8.s'        ; 11/06/2015
22648                                   <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS8.INC
22649                                   <1> ; Last Modification: 24/10/2015
22650                                   <1> ; -----------------------------------------------------------------------------
22651                                   <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
22652                                   <1> ; (v0.1 - Beginning: 11/07/2012)
22653                                   <1> ;
22654                                   <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
22655                                   <1> ; (Original) Source Code by Ken Thompson (1971-1972)
22656                                   <1> ; <Bell Laboratories (17/3/1972)>
22657                                   <1> ; <Preliminary Release of UNIX Implementation Document>
22658                                   <1> ;
22659                                   <1> ; Retro UNIX 8086 v1 - U8.ASM (18/01/2014) //// UNIX v1 -> u8.s
22660                                   <1> ;
22661                                   <1> ; ***************************************************************************
22662                                   <1>
22663                                   <1> ;; I/O Buffer - Retro UNIX 386 v1 modification
22664                                   <1> ;;     (8+512 bytes, 8 bytes header, 512 bytes data)
22665                                   <1> ;; Word 1, byte 0 = device id
22666                                   <1> ;; Word 1, byte 1 = status bits (bits 8 to 15)
22667                                   <1> ;;          bit 9 = write bit
22668                                   <1> ;;        bit 10 = read bit
22669                                   <1> ;;        bit 12 = waiting to write bit
22670                                   <1> ;;        bit 13 = waiting to read bit
22671                                   <1> ;;        bit 15 = inhibit bit
22672                                   <1> ;; Word 2 (byte 2 & byte 3) = reserved (for now - 07/06/2015)
22673                                   <1> ;; Word 3 + Word 4 (byte 4,5,6,7) = physical block number
22674                                   <1> ;;             (In fact, it is 32 bit LBA for Retro UNIX 386 v1)
22675                                   <1> ;;
22676                                   <1> ;; I/O Buffer ((8+512 bytes in original Unix v1))
22677                                   <1> ;;          ((4+512 bytes in Retro UNIX 8086 v1))
22678                                   <1> ;;
22679                                   <1> ;; I/O Queue Entry (of original UNIX operating system v1)
22680                                   <1> ;; Word 1, Byte 0 = device id
22681                                   <1> ;; Word 1, Byte 1 = (bits 8 to 15)
22682                                   <1> ;;          bit 9 = write bit
22683                                   <1> ;;        bit 10 = read bit
22684                                   <1> ;;        bit 12 = waiting to write bit
22685                                   <1> ;;        bit 13 = waiting to read bit
22686                                   <1> ;;        bit 15 = inhibit bit
22687                                   <1> ;; Word 2 = physical block number (In fact, it is LBA for Retro UNIX 8086 v1)
22688                                   <1> ;;
22689                                   <1> ;; Original UNIX v1 ->
22690                                   <1> ;;          Word 3 = number of words in buffer (=256)
22691                                   <1> ;; Original UNIX v1 ->
22692                                   <1> ;;          Word 4 = bus address (addr of first word of data buffer)
22693                                   <1> ;;
22694                                   <1> ;; Retro UNIX 8086 v1 -> Buffer Header (I/O Queue Entry) size is 4 bytes !
22695                                   <1> ;;
22696                                   <1> ;; Device IDs (of Retro Unix 8086 v1)
22697                                   <1> ;;          0 = fd0
22698                                   <1> ;;        1 = fd1
22699                                   <1> ;;        2 = hd0
22700                                   <1> ;;        3 = hd1
22701                                   <1> ;;        4 = hd2
22702                                   <1> ;;        5 = hd3
22703                                   <1>
22704                                   <1> ; Retro UNIX 386 v1 - 32 bit modifications (rfd, wfd, rhd, whd) - 09/06/2015
22705                                   <1>
22706                                   <1> rfd:  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22707                                   <1>       ; 26/04/2013
22708                                   <1>       ; 13/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
22709                                   <1>       ;sub   ax, 3 ; zero based device number (Floppy disk)
22710                                   <1>              ;jmp   short bread ; **** returns to routine that called readi

22711                                   <1>
22712                                   <1> rhd:  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22713                                   <1>       ; 26/04/2013
22714                                   <1>       ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
22715                                   <1>       ;sub   ax, 3 ; zero based device number (Hard disk)
22716                                   <1>       ;jmp   short bread ; **** returns to routine that called readi
22717                                   <1>
22718                                   <1> bread:
22719                                   <1>       ; 14/07/2015
22720                                   <1>       ; 10/07/2015
22721                                   <1>       ; 09/06/2015
22722                                   <1>       ; 07/06/2015 (Retro UNIX 386 v1 - Beginning)
22723                                   <1>       ; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
22724                                   <1>       ;
22725                                   <1>       ; / read a block from a block structured device
22726                                   <1>       ;
22727                                   <1>       ; INPUTS ->
22728                                   <1>       ;    [u.fopf] points to the block number
22729                                   <1>       ;    CX = maximum block number allowed on device
22730                                   <1>       ;       ; that was an arg to bread, in original Unix v1, but
22731                                   <1>       ;       ; CX register is used instead of arg in Retro Unix 8086 v1

22732                                   <1>       ;    [u.count]      number of bytes to read in
22733                                   <1>       ; OUTPUTS ->
22734                                   <1>       ;    [u.base] starting address of data block or blocks in user area
22735                                   <1>       ;    [u.fopf] points to next consecutive block to be read
22736                                   <1>       ;
22737                                   <1>       ; ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))
22738                                   <1>       ;
```

```
22739                                  <1>       ; NOTE: Original UNIX v1 has/had a defect/bug here, even if read
22740                                  <1>       ;       byte count is less than 512, block number in *u.fofp (u.off)
22741                                  <1>       ;     is increased by 1. For example: If user/program request
22742                                  <1>       ;       to read 16 bytes in current block, 'sys read' increases
22743                                  <1>       ;     the next block number just as 512 byte reading is done.
22744                                  <1>       ;       This wrong is done in 'bread'. So, in Retro UNIX 8086 v1,
22745                                  <1>       ;       for user (u) structure compatibility (because 16 bit is not
22746                                  <1>       ;       enough to keep byte position/offset of the disk), this
22747                                  <1>       ;     defect will not be corrected, user/program must request
22748                                  <1>       ;     512 byte read per every 'sys read' call to block devices
22749                                  <1>       ;       for achieving correct result. In future version(s),
22750                                  <1>       ;     this defect will be corrected by using different
22751                                  <1>       ;       user (u) structure.  26/07/2013 - Erdogan Tan
22752                                  <1>
22753                                  <1>             ; jsr r0,tstdeve / error on special file I/O
22754                                  <1>                         ; / (only works on tape)
22755                                  <1>             ; mov *u.fofp,r1 / move block number to r1
22756                                  <1>             ; mov $2.-cold,-(sp) / "2-cold" to stack
22757                                  <1> ;1:
22758                                  <1>             ; cmp r1,(r0) / is this block # greater than or equal to
22759                                  <1>                       ; / maximum block # allowed on device
22760                                  <1>             ; jnb short @f
22761                                  <1>             ; bhis 1f / yes, 1f (error)
22762                                  <1>             ; mov r1,-(sp) / no, put block # on stack
22763                                  <1>             ; jsr r0,preread / read in the block into an I/O buffer
22764                                  <1>             ; mov (sp)+,r1 / return block # to r1
22765                                  <1>             ; inc r1 / bump block # to next consecutive block
22766                                  <1>             ; dec (sp) / "2-1-cold" on stack
22767                                  <1>             ; bgt 1b / 2-1-cold = 0?  No, go back and read in next block
22768                                  <1> ;1:
22769                                  <1>             ; tst (sp)+ / yes, pop stack to clear off cold calculation
22770                                  <1>       ;push  ecx ; **
22771                                  <1>       ;26/04/2013
22772                                  <1>       ;sub   ax, 3 ; 3 to 8 -> 0 to 5
22773 00006059 2C03                   <1>       sub    al, 3
22774                                  <1>             ; AL = Retro Unix 8086 v1 disk (block device) number
22775 0000605B A2[B0740000]           <1>       mov    [u.brwdev], al
22776                                  <1>       ; 09/06/2015
22777 00006060 0FB6D8                 <1>       movzx  ebx, al
22778 00006063 8B8B[4E6B0000]         <1>       mov    ecx, [ebx+drv.size] ; disk size (in sectors)
22779                                  <1> bread_0:
22780 00006069 51                     <1>       push   ecx ; ** ; 09/06/2015
22781                                  <1>       ; 10/07/2015 (Retro UNIX 386 v1 modification!)
22782                                  <1>       ; [u.fopf] points to byte position in disk, not sector/block !
22783 0000606A 8B1D[58740000]         <1>       mov    ebx, [u.fofp]
22784 00006070 8B03                   <1>       mov    eax, [ebx]
22785 00006072 C1E809                 <1>       shr    eax, 9 ; convert byte position to block/sector number
22786                                  <1>             ; mov *u.fofp,r1 / restore r1 to initial value of the
22787                                  <1>                              ; / block #
22788 00006075 39C8                   <1>       cmp    eax, ecx
22789                                  <1>             ; cmp r1,(r0)+ / block # greater than or equal to maximum
22790                                  <1>                                ; / block number allowed
22791                                  <1>       ;jnb   error       ; 18/04/2013
22792                                  <1>             ; bhis error10 / yes, error
22793 00006077 720F                   <1>       jb     short bread_1
22794 00006079 C705[9D740000]1000-     <1>       mov    dword [u.error], ERR_DEV_VOL_SIZE  ; 'out of volume' error
22795 00006081 0000                   <1>
22796 00006083 E9B2DFFFFF             <1>       jmp    error
22797                                  <1> bread_1:
22798                                  <1>       ; inc  dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
22799                                  <1>             ; inc *u.fofp / no, *u.fofp has next block number
22800                                  <1>       ; eAX = Block number (zero based)
22801                                  <1>             ;;jsr r0,preread / read in the block whose number is in r1
22802                                  <1> preread: ;; call preread
22803 00006088 BF[B0740000]           <1>       mov    edi, u.brwdev ; block device number for direct I/O
22804 0000608D E864020000             <1>       call   bufaloc_0 ; 26/04/2013
22805                                  <1>       ;; jc  error
22806                                  <1>       ; eBX = Buffer (Header) Address -Physical-
22807                                  <1>         ; eAX = Block/Sector number (r1)
22808                                  <1>              ; jsr r0,bufaloc / get a free I/O buffer (r1 has block number)
22809                                  <1>       ; 14/03/2013
22810 00006092 740A                   <1>         jz   short bread_2 ; Retro UNIX 8086 v1 modification
22811                                  <1>                     ; br 1f / branch if block already in a I/O buffer
22812 00006094 66810B0004             <1>       or     word [ebx], 400h ; set read bit (10) in I/O Buffer
22813                                  <1>             ; bis $2000,(r5) / set read bit (bit 10 in I/O buffer)
22814 00006099 E8B3010000             <1>       call   poke
22815                                  <1>             ; jsr r0,poke / perform the read
22816                                  <1>       ;;jc   error ;2 0/07/2013
22817                                  <1> ; 1:
22818                                  <1>             ; clr *$ps / ps = 0
22819                                  <1>             ; rts r0
22820                                  <1> ;; return from preread
22821                                  <1> bread_2:
22822 0000609E 66810B0040             <1>       or     word [ebx], 4000h
22823                                  <1>             ; bis $40000,(r5)
22824                                  <1>                     ; / set bit 14 of the 1st word of the I/O buffer
22825                                  <1> bread_3: ; 1:
22826 000060A3 66F7030024             <1>       test   word [ebx], 2400h
22827                                  <1>             ; bit $22000,(r5) / are 10th and 13th bits set (read bits)
22828 000060A8 7407                   <1>       jz     short bread_4
22829                                  <1>             ; beq 1f / no
22830                                  <1>             ; cmp cdev,$1 / disk or drum?
22831                                  <1>             ; ble 2f / yes
22832                                  <1>             ; tstb uquant / is the time quantum = 0?
22833                                  <1>             ; bne 2f / no, 2f
22834                                  <1>             ; mov r5,-(sp) / yes, save r5 (buffer address)
22835                                  <1>             ; jsr r0,sleep; 31.
22836                                  <1>                     ; / put process to sleep in channel 31 (tape)
22837                                  <1>             ; mov (sp)+,r5 / restore r5
22838                                  <1>             ; br 1b / go back
22839                                  <1> ; 2: / drum or disk
22840                                  <1>         ;; mov     cx, [s.wait_]+2 ;; 29/07/2013
22841 000060AA E8AAF3FFFF             <1>       call   idle
22842                                  <1>             ; jsr r0,idle; s.wait+2 / wait
22843 000060AF EBF2                   <1>       jmp    short bread_3
```

```
22844                                  <1>                     ; br 1b
22845                                  <1> bread_4: ; 1: / 10th and 13th bits not set
22846 000060B1 668123FFBF              <1>       and   word [ebx], 0BFFFh ; 1011111111111111b
22847                                  <1>             ; bic $40000,(r5) / clear bit 14
22848                                  <1>                     ; jsr r0,tstdeve / test device for error (tape)
22849 000060B6 83C308                  <1>       add   ebx, 8
22850                                  <1>             ; add $8,r5 / r5 points to data in I/O buffer
22851                                  <1>       ; 09/06/2015
22852 000060B9 66833D[AD740000]00      <1>       cmp   word [u.pcount], 0
22853 000060C1 7705                    <1>       ja    short bread_5
22854 000060C3 E896F9FFFF              <1>       call  trans_addr_w ; translate virtual address to physical (w)
22855                                  <1> bread_5:
22856                                  <1>       ; eBX = system (I/O) buffer address
22857 000060C8 E870000000              <1>       call  dioreg
22858                                  <1>                     ; jsr r0,dioreg / do bookkeeping on u.count etc.
22859                                  <1>       ; esi =  start address of the transfer (in the buffer)
22860                                  <1>       ; edi =  [u.pbase], destination address in user's memory space
22861                                  <1>       ; ecx =  transfer count (in bytes)
22862                                  <1>       ;
22863                                  <1> ;1: / r5 points to beginning of data in I/O buffer, r2 points to beginning
22864                                  <1> ;   / of users data
22865 000060CD F3A4                    <1>       rep   movsb
22866                                  <1>             ; movb (r5)+,(r2)+ / move data from the I/O buffer
22867                                  <1>                     ; dec r3 / to the user's area in core starting at u.base
22868                                  <1>                     ; bne 1b
22869 000060CF 59                      <1>       pop   ecx ; **
22870 000060D0 833D[6C740000]00        <1>       cmp   dword [u.count], 0
22871                                  <1>             ; tst u.count / done
22872 000060D7 7790                    <1>       ja    short bread_0 ; 09/06/2015
22873                                  <1>                     ; beq 1f / yes, return
22874                                  <1>             ; tst -(r0) / no, point r0 to the argument again
22875                                  <1>                     ; br bread / read some more
22876                                  <1> ; 1:
22877 000060D9 58                      <1>       pop   eax ; ****
22878                                  <1>                     ; mov (sp)+,r0
22879 000060DA C3                      <1>       retn        ; 09/06/2015
22880                                  <1>       ;jmp     ret_
22881                                  <1>             ;jmp ret  / jump to routine that called readi
22882                                  <1>
22883                                  <1> wfd:    ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22884                                  <1>       ; 26/04/2013
22885                                  <1>       ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
22886                                  <1>       ;sub   ax, 3 ; zero based device number (Hard disk)
22887                                  <1>       ;jmp   short bwrite ; **** returns to routine that called writei
22888                                  <1>
22889                                  <1> whd:  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22890                                  <1>       ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
22891                                  <1>       ;sub   ax, 3 ; zero based device number (Hard disk)
22892                                  <1>       ;jmp   short bwrite ; **** returns to routine that called writei ('jmp ret')
22893                                  <1>
22894                                  <1> bwrite:
22895                                  <1>       ; 14/07/2015
22896                                  <1>       ; 10/07/2015
22897                                  <1>       ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22898                                  <1>       ; 14/03/2013 - 20/07/2013 (Retro UNIX 8086 v1)
22899                                  <1>       ;
22900                                  <1>       ;; / write on block structured device
22901                                  <1>       ;
22902                                  <1>       ; INPUTS ->
22903                                  <1>       ;    [u.fopf] points to the block number
22904                                  <1>       ;    CX = maximum block number allowed on device
22905                                  <1>       ;       ; that was an arg to bwrite, in original Unix v1, but
22906                                  <1>       ;       ; CX register is used instead of arg in Retro Unix 8086 v1

22907                                  <1>       ;    [u.count]      number of bytes to user desires to write
22908                                  <1>       ; OUTPUTS ->
22909                                  <1>       ;    [u.fopf] points to next consecutive block to be written into
22910                                  <1>       ;
22911                                  <1>       ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
22912                                  <1>       ;
22913                                  <1>       ; NOTE: Original UNIX v1 has/had a defect/bug here, even if write
22914                                  <1>       ;       byte count is less than 512, block number in *u.fofp (u.off)
22915                                  <1>       ;      is increased by 1. For example: If user/program request
22916                                  <1>       ;       to write 16 bytes in current block, 'sys write' increases
22917                                  <1>       ;      the next block number just as 512 byte writing is done.
22918                                  <1>       ;       This wrong is done in 'bwrite'. So, in Retro UNIX 8086 v1,
22919                                  <1>       ;       for user (u) structure compatibility (because 16 bit is not
22920                                  <1>       ;       enough to keep byte position/offset of the disk), this
22921                                  <1>       ;      defect will not be corrected, user/program must request
22922                                  <1>       ;      512 byte write per every 'sys write' call to block devices
22923                                  <1>       ;       for achieving correct result. In future version(s),
22924                                  <1>       ;      this defect will be corrected by using different
22925                                  <1>       ;       user (u) structure.  26/07/2013 - Erdogan Tan
22926                                  <1>
22927                                  <1>                     ; jsr r0,tstdeve / test the device for an error
22928                                  <1>       ;push  ecx ; **
22929                                  <1>       ;26/04/2013
22930                                  <1>       ;sub   ax, 3 ; 3 to 8 -> 0 to 5
22931 000060DB 2C03                    <1>       sub   al, 3
22932                                  <1>             ; AL = Retro Unix 8086 v1 disk (block device) number
22933 000060DD A2[B0740000]            <1>       mov   [u.brwdev], al
22934                                  <1>       ; 09/06/2015
22935 000060E2 0FB6D8                  <1>       movzx ebx, al
22936 000060E5 8B8B[4E6B0000]          <1>       mov   ecx, [ebx+drv.size] ; disk size (in sectors)
22937                                  <1> bwrite_0:
22938 000060EB 51                      <1>       push  ecx ; ** ; 09/06/2015
22939                                  <1>       ; 10/07/2015 (Retro UNIX 386 v1 modification!)
22940                                  <1>       ; [u.fopf] points to byte position in disk, not sector/block !
22941 000060EC 8B1D[58740000]          <1>       mov   ebx, [u.fofp]
22942 000060F2 8B03                    <1>       mov   eax, [ebx]
22943 000060F4 C1E809                  <1>       shr   eax, 9 ; convert byte position to block/sector number
22944                                  <1>             ; mov *u.fofp,r1 / put the block number in r1
22945 000060F7 39C8                    <1>       cmp   eax, ecx
22946                                  <1>             ; cmp r1,(r0)+ / does block number exceed maximum allowable #
22947                                  <1>                                     ; / block number allowed
```

```
22948                                  <1>       ;jnb   error       ; 18/04/2013
22949                                  <1>             ; bhis error10 / yes, error
22950 000060F9 720F                    <1>       jb    short bwrite_1
22951 000060FB C705[9D740000]1000-     <1>       mov   dword [u.error], ERR_DEV_VOL_SIZE  ; 'out of volume' error
22952 00006103 0000                    <1>
22953 00006105 E930DFFFFF              <1>       jmp   error
22954                                  <1> bwrite_1:
22955                                  <1>       ; inc  dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
22956                                  <1>             ; inc *u.fofp / no, increment block number
22957                                  <1>       ; 09/06/2015 - 10/07/2015
22958 0000610A 66833D[AD740000]00      <1>       cmp   word [u.pcount], 0
22959 00006112 7705                    <1>       ja    short bwrite_2
22960 00006114 E841F9FFFF              <1>       call  trans_addr_r ; translate virtual address to physical (r)
22961                                  <1> bwrite_2:
22962 00006119 BF[B0740000]            <1>       mov   edi, u.brwdev  ; block device number for direct I/O
22963 0000611E E8C4000000              <1>             call  bwslot ; 26/04/2013 (wslot -> bwslot)

22964                                  <1>             ; jsr r0,wslot / get an I/O buffer to write into
22965                                  <1>             ; add $8,r5 / r5 points to data in I/O buffer
22966 00006123 E815000000              <1>         call dioreg
22967                                  <1>             ; jsr r0,dioreg / do the necessary bookkeeping
22968                                  <1>       ; esi =  destination address (in the buffer)
22969                                  <1>       ; edi =  [u.pbase], start address of transfer in user's memory space
22970                                  <1>       ; ecx =  transfer count (in bytes)
22971                                  <1> ; 1: / r2 points to the users data; r5 points to the I/O buffers data area
22972 00006128 87F7                    <1>       xchg  esi, edi ; 14/07/2015
22973 0000612A F3A4                    <1>       rep   movsb
22974                                  <1>             ; movb (r2)+,(r5)+ / ; r3, has the byte count
22975                                  <1>                   ; dec r3 / area to the I/O buffer
22976                                  <1>                   ; bne 1b
22977 0000612C E8FE000000              <1>       call  dskwr
22978                                  <1>             ; jsr r0,dskwr / write it out on the device
22979 00006131 59                      <1>       pop   ecx ; **
22980 00006132 833D[6C740000]00        <1>         cmp     dword [u.count], 0
22981                                  <1>             ; tst u.count / done
22982 00006139 77B0                    <1>       ja    short bwrite_0 ; 09/06/2015
22983                                  <1>             ; beq 1f / yes, 1f
22984                                  <1>             ; tst -(r0) / no, point r0 to the argument of the call
22985                                  <1>                   ; br bwrite / go back and write next block
22986                                  <1> ; 1:
22987 0000613B 58                      <1>       pop   eax ; ****
22988                                  <1>                   ; mov (sp)+,r0
22989 0000613C C3                      <1>       retn        ; 09/06/2015
22990                                  <1>         ;jmp     ret_
22991                                  <1>             ; jmp ret / return to routine that called writei
22992                                  <1> ;error10:
22993                                  <1> ;       jmp     error  ; / see 'error' routine
22994                                  <1>
22995                                  <1> dioreg:
22996                                  <1>       ; 14/07/2015
22997                                  <1>       ; 10/07/2015 (UNIX v1 bugfix - [u.fofp]: byte pos., not block)
22998                                  <1>       ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22999                                  <1>       ; 14/03/2013 (Retro UNIX 8086 v1)
23000                                  <1>       ;
23001                                  <1>       ; bookkeeping on block transfers of data
23002                                  <1>       ;
23003                                  <1>       ; * returns value of u.pbase before it gets updated, in EDI
23004                                  <1>       ; * returns byte count (to transfer) in ECX (<=512)
23005                                  <1>       ; 10/07/2015
23006                                  <1>       ; * returns byte offset from beginning of current sector buffer
23007                                  <1>       ; (beginning of data) in ESI
23008                                  <1>       ;
23009 0000613D 8B0D[6C740000]          <1>       mov   ecx, [u.count]
23010                                  <1>             ; mov u.count,r3 / move char count to r3
23011 00006143 81F900020000            <1>             cmp   ecx, 512
23012                                  <1>             ; cmp r3,$512. / more than 512. char?
23013 00006149 7605                    <1>       jna   short dioreg_0
23014                                  <1>             ; blos 1f / no, branch
23015 0000614B B900020000              <1>       mov   ecx, 512
23016                                  <1>             ; mov $512.,r3 / yes, just take 512.
23017                                  <1> dioreg_0:
23018                                  <1>       ; 09/06/2015
23019 00006150 663B0D[AD740000]        <1>       cmp   cx, [u.pcount]
23020 00006157 7607                    <1>       jna   short dioreg_1
23021 00006159 668B0D[AD740000]        <1>       mov   cx, [u.pcount]
23022                                  <1> dioreg_1:
23023                                  <1> ; 1:
23024 00006160 8B15[68740000]          <1>       mov   edx, [u.base] ; 09/06/2015 (eax -> edx)
23025                                  <1>             ; mov u.base,r2 / put users base in r2
23026 00006166 010D[70740000]          <1>       add   [u.nread], ecx
23027                                  <1>             ; add r3,u.nread / add the number to be read to u.nread
23028 0000616C 290D[6C740000]          <1>       sub   [u.count], ecx
23029                                  <1>             ; sub r3,u.count / update count
23030 00006172 010D[68740000]          <1>       add   [u.base], ecx
23031                                  <1>             ; add r3,u.base / update base
23032                                  <1>       ; 10/07/2015
23033                                  <1>       ; Retro UNIX 386 v1 - modification !
23034                                  <1>       ; (File pointer points to byte position, not block/sector no.)
23035                                  <1>       ; (It will point to next byte position instead of next block no.)
23036 00006178 8B35[58740000]          <1>       mov   esi, [u.fofp] ; u.fopf points to byte position pointer
23037 0000617E 8B06                    <1>       mov   eax, [esi] ; esi points to current byte pos. on the disk
23038 00006180 010E                    <1>       add   [esi], ecx ; ecx is added to set the next byte position
23039 00006182 25FF010000              <1>       and   eax, 1FFh  ; get offset from beginning of current block
23040 00006187 89DE                    <1>       mov   esi, ebx   ; beginning of data in sector/block buffer
23041 00006189 01C6                    <1>       add   esi, eax   ; esi contains start address of the transfer
23042                                  <1>       ; 09/06/2015 - 10/07/2015
23043 0000618B 66290D[AD740000]        <1>       sub   [u.pcount], cx
23044 00006192 81E2FF0F0000            <1>       and   edx, PAGE_OFF ; 0FFFh
23045 00006198 8B3D[A9740000]          <1>       mov   edi, [u.pbase]
23046 0000619E 81E700F0FFFF            <1>       and   edi, ~PAGE_OFF
23047 000061A4 01D7                    <1>       add   edi, edx
23048 000061A6 893D[A9740000]          <1>       mov   [u.pbase], edi
23049 000061AC 010D[A9740000]          <1>       add   [u.pbase], ecx ; 14/07/2015
23050 000061B2 C3                      <1>       retn
23051                                  <1>             ; rts r0 / return
```

```
23052                                  <1>
23053                                  <1> dskrd:
23054                                  <1>       ; 18/08/2015
23055                                  <1>       ; 02/07/2015
23056                                  <1>       ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23057                                  <1>       ; 14/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
23058                                  <1>       ;
23059                                  <1>       ; 'dskrd' acquires an I/O buffer, puts in the proper
23060                                  <1>       ; I/O queue entries (via bufaloc) then reads a block
23061                                  <1>       ; (number specified in r1) in the acquired buffer.)
23062                                  <1>       ; If the device is busy at the time dskrd is called,
23063                                  <1>       ; dskrd calls idle.
23064                                  <1>       ;
23065                                  <1>       ; INPUTS ->
23066                                  <1>       ;    r1 - block number
23067                                  <1>       ;    cdev - current device number
23068                                  <1>       ; OUTPUTS ->
23069                                  <1>       ;    r5 - points to first data word in I/O buffer
23070                                  <1>       ;
23071                                  <1>       ; ((AX = R1)) input/output
23072                                  <1>       ; ((BX = R5)) output
23073                                  <1>       ;
23074                                  <1>         ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
23075                                  <1>       ;
23076 000061B3 E830010000             <1>       call  bufaloc
23077                                  <1>             ; jsr r0,bufaloc / shuffle off to bufaloc;
23078                                  <1>                              ; / get a free I/O buffer
23079                                  <1>       ;;jc  error ; 20/07/2013
23080 000061B8 741B                  <1>       jz    short dskrd_1 ; Retro UNIX 8086 v1 modification
23081                                  <1>                     ; br 1f / branch if block already in a I/O buffer
23082                                  <1> dskrd_0: ; 10/07/2015 (wslot)
23083 000061BA 66810B0004            <1>       or    word [ebx], 400h ; set read bit (10) in I/O Buffer
23084                                  <1>             ; bis $2000,(r5) / set bit 10 of word 1 of
23085                                  <1>                              ; / I/O queue entry for buffer
23086 000061BF E88D000000            <1>       call  poke
23087                                  <1>             ; jsr r0,poke / just assigned in bufaloc,
23088                                  <1>                         ; /      bit 10=1 says read
23089                                  <1>       ; 09/06/2015
23090 000061C4 730F                  <1>       jnc   short dskrd_1
23091                                  <1>       ;
23092 000061C6 C705[9D740000]1100-   <1>       mov   dword [u.error], ERR_DRV_READ ; disk read error !
23093 000061CE 0000                  <1>
23094 000061D0 E965DEFFFF            <1>       jmp   error
23095                                  <1> dskrd_1: ; 1:
23096                                  <1>                     ;clr *$ps
23097 000061D5 66F7030024            <1>             test  word [ebx], 2400h
23098                                  <1>             ; bit $22000,(r5) / if either bits 10, or 13 are 1;
23099                                  <1>                               ; / jump to idle
23100 000061DA 7407                  <1>             jz    short dskrd_2
23101                                  <1>             ; beq 1f
23102                                  <1>           ;;mov   ecx, [s.wait_]
23103 000061DC E878F2FFFF            <1>             call  idle
23104                                  <1>             ; jsr r0,idle; s.wait+2
23105 000061E1 EBF2                  <1>       jmp   short dskrd_1
23106                                  <1>                     ; br 1b
23107                                  <1> dskrd_2: ; 1:
23108 000061E3 83C308                <1>           add  ebx, 8
23109                                  <1>             ; add $8,r5 / r5 points to first word of data in block
23110                                  <1>                       ; / just read in
23111 000061E6 C3                    <1>             retn
23112                                  <1>             ; rts r0
23113                                  <1>
23114                                  <1> bwslot:
23115                                  <1>       ; 10/07/2015
23116                                  <1>       ;     If the block/sector is not placed in a buffer
23117                                  <1>       ;     before 'wslot', it must be read before
23118                                  <1>       ;     it is written! (Otherwise transfer counts less
23119                                  <1>       ;     than 512 bytes will be able to destroy existing
23120                                  <1>       ;     data on disk.)
23121                                  <1>       ;
23122                                  <1>       ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23123                                  <1>       ; 26/04/2013(Retro UNIX 8086 v1)
23124                                  <1>       ; Retro UNIX 8086 v1 modification !
23125                                  <1>       ; ('bwslot' will be called from 'bwrite' only!)
23126                                  <1>       ; INPUT -> eDI - points to device id (in u.brwdev)
23127                                  <1>       ;     -> eAX = block number
23128                                  <1>       ;
23129 000061E7 E80A010000            <1>       call  bufaloc_0
23130 000061EC 742A                  <1>       jz    short wslot_0 ; block/sector already is in the buffer
23131                                  <1> bwslot_0:
23132                                  <1>       ; 10/07/2015
23133 000061EE 8B35[58740000]        <1>       mov   esi, [u.fofp]
23134 000061F4 8B06                  <1>       mov   eax, [esi]
23135 000061F6 25FF010000            <1>       and   eax, 1FFh ; offset from beginning of the sector/block
23136 000061FB 750C                  <1>       jnz   short bwslot_1 ; it is not a full sector write
23137                                  <1>                     ; recent disk data must be placed in the buffer
23138 000061FD 813D[6C740000]0002-   <1>       cmp   dword [u.count], 512
23139 00006205 0000                  <1>
23140 00006207 730F                  <1>       jnb   short wslot_0
23141                                  <1> bwslot_1:
23142 00006209 E8ACFFFFFF            <1>       call  dskrd_0
23143 0000620E 83EB08                <1>       sub   ebx, 8 ; set ebx to the buffer header address again
23144 00006211 EB05                  <1>       jmp   short wslot_0
23145                                  <1>
23146                                  <1> wslot:
23147                                  <1>       ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23148                                  <1>       ;            (32 bit modifications)
23149                                  <1>       ; 14/03/2013 - 29/07/2013(Retro UNIX 8086 v1)
23150                                  <1>       ;
23151                                  <1>       ; 'wslot' calls 'bufaloc' and obtains as a result, a pointer
23152                                  <1>       ; to the I/O queue of an I/O buffer for a block structured
23153                                  <1>       ; device. It then checks the first word of I/O queue entry.
23154                                  <1>       ; If bits 10 and/or 13 (read bit, waiting to read bit) are set,
23155                                  <1>       ; wslot calls 'idle'. When 'idle' returns, or if bits 10
23156                                  <1>       ; and/or 13 are not set, 'wslot' sets bits 9 and 15 of the first
```

```
23157                           <1>       ; word of the I/O queue entry (write bit, inhibit bit).
23158                           <1>       ;
23159                           <1>       ; INPUTS ->
23160                           <1>       ;    r1 - block number
23161                           <1>       ;    cdev - current (block/disk) device number
23162                           <1>       ;
23163                           <1>       ; OUTPUTS ->
23164                           <1>       ;    bufp - bits 9 and 15 are set,
23165                           <1>       ;           the remainder of the word left unchanged
23166                           <1>       ;    r5 - points to first data word in I/O buffer
23167                           <1>       ;
23168                           <1>       ; ((AX = R1)) input/output
23169                           <1>       ; ((BX = R5)) output
23170                           <1>       ;
23171                           <1>         ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
23172                           <1>
23173 00006213 E8D0000000        <1>       call  bufaloc
23174                           <1>       ; 10/07/2015
23175                           <1>             ; jsr r0,bufaloc / get a free I/O buffer; pointer to first
23176                           <1>             ; br 1f / word in buffer in r5
23177                           <1>       ; eBX = Buffer (Header) Address (r5) (ES=CS=DS, system/kernel segment)
23178                           <1>         ; eAX = Block/Sector number (r1)
23179                           <1> wslot_0: ;1:
23180 00006218 66F7030024        <1>       test  word [ebx], 2400h
23181                           <1>             ; bit $22000,(r5) / check bits 10, 13 (read, waiting to read)
23182                           <1>                               ; / of I/O queue entry
23183 0000621D 7407              <1>       jz    short wslot_1
23184                           <1>               ; beq 1f  / branch if 10, 13 zero (i.e., not reading,
23185                           <1>                 ; / or not waiting to read)
23186                           <1>
23187                           <1>         ;; mov     ecx, [s.wait_] ; 29/07/2013
23188 0000621F E835F2FFFF        <1>       call  idle
23189                           <1>             ; jsr r0,idle; / if buffer is reading or writing to read,
23190                           <1>                              ; / idle
23191 00006224 EBF2              <1>       jmp   short wslot_0
23192                           <1>             ; br 1b / till finished
23193                           <1> wslot_1: ;1:
23194 00006226 66810B0082        <1>         or      word [ebx], 8200h
23195                           <1>                 ; bis $101000,(r5) / set bits 9, 15 in 1st word of I/O queue
23196                           <1>                                    ; / (write, inhibit bits)
23197                           <1>                 ; clr     *$ps / clear processor status
23198 0000622B 83C308            <1>         add  ebx, 8 ; 11/06/2015
23199                           <1>             ; add $8,r5 / r5 points to first word in data area
23200                           <1>                       ; / for this block
23201 0000622E C3                <1>             retn
23202                           <1>             ; rts r0
23203                           <1> dskwr:
23204                           <1>       ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23205                           <1>       ; 14/03/2013 - 03/08/2013 (Retro UNIX 8086 v1)
23206                           <1>       ;
23207                           <1>       ; 'dskwr' writes a block out on disk, via ppoke. The only
23208                           <1>       ; thing dskwr does is clear bit 15 in the first word of I/O queue
23209                           <1>       ; entry pointed by 'bufp'. 'wslot' which must have been called
23210                           <1>       ; previously has supplied all the information required in the
23211                           <1>       ; I/O queue entry.
23212                           <1>       ;
23213                           <1>       ; (Modified registers: eCX, eDX, eBX, eSI, eDI)
23214                           <1>       ;
23215                           <1>       ;
23216 0000622F 8B1D[0A740000]    <1>       mov   ebx, [bufp]
23217 00006235 668123FF7F        <1>       and   word [ebx], 7FFFh ; 0111111111111111b
23218                           <1>             ; bic $100000,*bufp / clear bit 15 of I/O queue entry at
23219                           <1>                                 ; / bottom of queue
23220 0000623A E812000000        <1>       call  poke
23221                           <1>       ; 09/06/2015
23222 0000623F 730F              <1>       jnc   short dskwr_1
23223 00006241 C705[9D740000]1200- <1>     mov   dword [u.error], ERR_DRV_WRITE ; disk write error !
23224 00006249 0000              <1>
23225 0000624B E9EADDFFFF        <1>       jmp   error
23226                           <1> dskwr_1:
23227 00006250 C3                <1>       retn
23228                           <1>
23229                           <1>
23230                           <1> ;ppoke:
23231                           <1>                 ; mov $340,*$ps
23232                           <1>                 ; jsr r0,poke
23233                           <1>                 ; clr *$ps
23234                           <1>             ; rts r0
23235                           <1> poke:
23236                           <1>       ; 24/10/2015
23237                           <1>       ; 20/08/2015
23238                           <1>       ; 18/08/2015
23239                           <1>       ; 02/07/2015
23240                           <1>       ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23241                           <1>       ; 15/03/2013 - 18/01/2014 (Retro UNIX 8086 v1)
23242                           <1>       ;
23243                           <1>       ; (NOTE: There are some disk I/O code modifications & extensions
23244                           <1>       ; & exclusions on original 'poke' & other device I/O procedures of
23245                           <1>       ; UNIX v1 OS for performing disk I/O functions by using IBM PC
23246                           <1>       ; compatible rombios calls in Retro UNIX 8086 v1 kernel.)
23247                           <1>       ;
23248                           <1>       ; Basic I/O functions for all block structured devices
23249                           <1>       ;
23250                           <1>         ; (Modified registers: eCX, eDX, eSI, eDI)
23251                           <1>       ;
23252                           <1>       ; 20/07/2013 modifications
23253                           <1>       ;           (Retro UNIX 8086 v1 features only !)
23254                           <1>       ; INPUTS ->
23255                           <1>       ;       (EBX = buffer header address)
23256                           <1>       ; OUTPUTS ->
23257                           <1>       ;      cf=0 -> successed r/w (at least, for the caller's buffer)
23258                           <1>       ;      cf=1 -> error, word [eBX] = 0FFFFh
23259                           <1>       ;            (drive not ready or r/w error!)
23260                           <1>       ;      (dword [EBX+4] <> 0FFFFFFFFh indicates r/w success)
23261                           <1>       ;      (dword [EBx+4] = 0FFFFFFFFh means RW/IO error)
```

```
23262                                <1>       ;        (also it indicates invalid buffer data)
23263                                <1>       ;
23264 00006251 53                    <1>       push   ebx
23265                                <1>                    ; mov r1,-(sp)
23266                                <1>                    ; mov r2,-(sp)
23267                                <1>                    ; mov r3,-(sp)
23268 00006252 50                    <1>       push   eax ; Physical Block Number (r1) (mget)
23269                                <1>       ;
23270                                <1>       ; 09/06/2015
23271                                <1>       ; (permit read/write after a disk  R/W error)
23272 00006253 8A0B                  <1>       mov    cl, [ebx] ; device id (0 to 5)
23273 00006255 B001                  <1>       mov    al, 1
23274 00006257 D2E0                  <1>       shl    al, cl
23275 00006259 8405[32740000]        <1>       test   al, [active] ; busy ? (error)
23276 0000625F 7408                  <1>       jz     short poke_0
23277 00006261 F6D0                  <1>       not    al
23278 00006263 2005[32740000]        <1>       and    [active], al ; reset busy bit for this device only
23279                                <1> poke_0:
23280 00006269 BE[2A740000]          <1>         mov     esi, bufp + (4*(nbuf+2))
23281                                <1>             ; mov $bufp+nbuf+nbuf+6,r2 / r2 points to highest priority
23282                                <1>                                ; / I/O queue pointer
23283                                <1> poke_1: ; 1:
23284 0000626E 83EE04                <1>         sub  esi, 4
23285 00006271 8B1E                  <1>       mov    ebx, [esi]
23286                                <1>             ; mov -(r2),r1 / r1 points to an I/O queue entry
23287 00006273 668B03                <1>       mov    ax, [ebx] ; 17/07/2013
23288 00006276 F6C406                <1>             test   ah, 06h
23289                                <1>       ;test  word [ebx], 600h ; 0000011000000000b
23290                                <1>             ; bit $3000,(r1) / test bits 9 and 10 of word 1 of I/O
23291                                <1>                          ; / queue entry
23292 00006279 745E                  <1>         jz     short poke_5
23293                                <1>             ; beq 2f / branch to 2f if both are clear
23294                                <1>       ; 31/07/2013
23295                                <1>       ;test  ah, 0B0h ; (*)
23296                                <1>       ;;test word [ebx], 0B000h ; 1011000000000000b
23297                                <1>             ; bit $130000,(r1) / test bits 12, 13, and 15
23298                                <1>         ;jnz     short poke_5 ; 31/07/2013 (*)
23299                                <1>             ; bne 2f / branch if any are set
23300                                <1>       ;movzx ecx, byte [ebx] ; 09/06/2015 ; Device Id
23301                                <1>             ; movb (r1),r3 / get device id
23302 0000627B 0FB6C8               <1>       movzx  ecx, al ; 18/08/2015
23303                                <1>       ;mov   edi, ecx ; 26/04/2013
23304 0000627E 31C0                  <1>       xor    eax, eax ; 0
23305                                <1>       ;cmp   [edi+drv.error], al ; 0
23306                                <1>             ; tstb deverr(r3) / test for errors on this device
23307                                <1>             ;jna   short poke_2
23308                                <1>             ; beq 3f / branch if no errors
23309                                <1>       ; 02/07/2015
23310                                <1>       ;dec   eax
23311                                <1>       ;mov   [ebx+4], ax ; 0FFFFFFFFh ; -1
23312                                <1>                    ; mov $-1,2(r1) / destroy associativity
23313                                <1>       ;shr   eax, 24
23314                                <1>       ;mov   [ebx], eax ; 000000FFh, reset
23315                                <1>             ; clrb 1(r1) / do not do I/O
23316                                <1>       ;jmp     short poke_5
23317                                <1>         ;       ; br 2f
23318                                <1>                 ; rts r0
23319                                <1> poke_2: ; 3:
23320                                <1>       ; 02/07/2015
23321 00006280 FEC1                  <1>       inc    cl ; 0FFh -> 0
23322 00006282 7455                  <1>       jz     short poke_5
23323 00006284 FEC0                  <1>       inc    al ; mov ax, 1
23324 00006286 FEC9                  <1>       dec    cl
23325 00006288 7402                  <1>       jz     short poke_3
23326                                <1>       ; 26/04/2013 Modification
23327                                <1>       ;inc   al ; mov ax, 1
23328                                <1>       ;or    cl, cl ; Retro UNIX 8086 v1 device id.
23329                                <1>       ;jz    short poke_3 ; cl = 0
23330 0000628A D2E0                  <1>       shl    al, cl ; shl ax, cl
23331                                <1> poke_3:
23332                                <1>       ;test  [active], ax
23333 0000628C 8405[32740000]        <1>       test   [active], al
23334                                <1>             ; bit $2,active / test disk busy bit
23335 00006292 7545                  <1>       jnz     short poke_5
23336                                <1>             ; bne 2f / branch if bit is set
23337                                <1>       ;or    [active], ax
23338 00006294 0805[32740000]        <1>       or     [active], al
23339                                <1>             ; bis $2,active / set disk busy bit
23340 0000629A 6650                  <1>       push   ax
23341 0000629C E8CB000000            <1>       call   diskio ; Retro UNIX 8086 v1 Only !
23342                                <1>       ;mov    [edi+drv.error], ah
23343 000062A1 6658                  <1>       pop    ax
23344 000062A3 730E                  <1>       jnc    short poke_4 ; 20/07/2013
23345                                <1>       ;cmp   [edi+drv.error], al ; 0
23346                                <1>       ;jna   short poke_4
23347                                <1>             ; tstb deverr(r3) / test for errors on this device
23348                                <1>                    ; beq 3f / branch if no errors
23349                                <1>       ; 02/07/2015 (32 bit modification)
23350                                <1>       ; 20/07/2013
23351 000062A5 C74304FFFFFFFF       <1>       mov    dword [ebx+4], 0FFFFFFFFh ; -1
23352                                <1>                    ; mov $-1,2(r1) / destroy associativity
23353 000062AC 66C703FF00            <1>       mov    word [ebx], 0FFh ; 20/08/2015
23354                                <1>             ; clrb 1(r1) / do not do I/O
23355 000062B1 EB26                  <1>       jmp     short poke_5
23356                                <1> poke_4:     ; 20/07/2013
23357                                <1>       ; 17/07/2013
23358 000062B3 F6D0                  <1>       not    al
23359 000062B5 2005[32740000]        <1>       and    [active], al ; reset, not busy
23360                                <1>       ; eBX = system I/O buffer header (queue entry) address
23361                                <1> seta: ; / I/O queue bookkeeping; set read/write waiting bits.
23362 000062BB 668B03                <1>       mov    ax, [ebx]
23363                                <1>                    ; mov (r1),r3 / move word 1 of I/O queue entry into r3
23364 000062BE 66250006              <1>         and  ax, 600h
23365                                <1>             ; bic $!3000,r3 / clear all bits except 9 and 10
23366 000062C2 668123FFF9            <1>       and    word [ebx], 0F9FFh
```

```
23367                                  <1>                     ; bic $3000,(r1) / clear only bits 9 and 10
23368 000062C7 C0E403                  <1>       shl   ah, 3
23369                                  <1>                     ; rol r3
23370                                  <1>                   ; rol r3
23371                                  <1>                   ; rol r3
23372 000062CA 660903                  <1>       or    [ebx], ax
23373                                  <1>             ; bis r3,(r1) / or old value of bits 9 and 10 with
23374                                  <1>                           ; bits 12 and 13
23375 000062CD E887F1FFFF              <1>       call  idle ; 18/01/2014
23376                                  <1>       ;; sti
23377                                  <1>       ;hlt   ; wait for a hardware interrupt
23378                                  <1>       ;; cli
23379                                  <1>       ; NOTE: In fact, disk controller's 'disk I/O completed'
23380                                  <1>         ; interrupt would be used to reset busy bits, but INT 13h
23381                                  <1>       ; returns when disk I/O is completed. So, here, as temporary
23382                                  <1>       ; method, this procedure will wait for a time according to
23383                                  <1>       ; multi tasking and time sharing concept.
23384                                  <1>       ;
23385                                  <1>       ; 24/10/2015
23386                                  <1>       ;not   ax
23387 000062D2 66B8FF00               <1>       mov   ax, 0FFh ; 24/10/2015 (temporary)
23388 000062D6 662103                  <1>       and   [ebx], ax ; clear bits 12 and 13
23389                                  <1> poke_5: ;2:
23390 000062D9 81FE[0A740000]          <1>         cmp     esi, bufp
23391                                  <1>               ; cmp r2,$bufp / test to see if entire I/O queue
23392                                  <1>                              ; / has been scanned
23393 000062DF 778D                    <1>       ja      short poke_1
23394                                  <1>                ; bhi 1b
23395                                  <1>       ; 24/03/2013
23396                                  <1>                     ; mov (sp)+,r3
23397                                  <1>                     ; mov (sp)+,r2
23398                                  <1>                     ; mov (sp)+,r1
23399 000062E1 58                      <1>         pop  eax  ; Physical Block Number (r1) (mget)
23400 000062E2 5B                      <1>       pop   ebx
23401                                  <1>       ; 02/07/2015 (32 bit modification)
23402                                  <1>       ; 20/07/2013
23403                                  <1>       ;cmp   dword [ebx+4], 0FFFFFFFFh
23404 000062E3 803BFF                  <1>       cmp   byte [ebx], 0FFh ; 20/08/2015
23405                                  <1>       ;
23406                                  <1>       ; 'poke' returns with cf=0 if the requested buffer is read
23407                                  <1>       ; or written succesfully; even if an error occurs while
23408                                  <1>       ; reading to or writing from other buffers. 20/07/2013
23409                                  <1>       ;
23410                                  <1>       ; 09/06/2015
23411 000062E6 F5                      <1>       cmc
23412 000062E7 C3                      <1>       retn
23413                                  <1>                ; rts r0
23414                                  <1>
23415                                  <1> bufaloc:
23416                                  <1>       ; 20/08/2015
23417                                  <1>       ; 19/08/2015
23418                                  <1>       ; 02/07/2015
23419                                  <1>       ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23420                                  <1>       ;          (32 bit modifications)
23421                                  <1>       ; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
23422                                  <1>       ;
23423                                  <1>       ; bufaloc - Block device I/O buffer allocation
23424                                  <1>       ;
23425                                  <1>       ; INPUTS ->
23426                                  <1>       ;    r1 - block number
23427                                  <1>       ;    cdev - current (block/disk) device number
23428                                  <1>       ;    bufp+(2*n)-2 --- n = 1 ... nbuff
23429                                  <1>       ; OUTPUTS ->
23430                                  <1>       ;    r5 - pointer to buffer allocated
23431                                  <1>       ;    bufp ... bufp+12 --- (bufp), (bufp)+2
23432                                  <1>       ;
23433                                  <1>       ; ((AX = R1)) input/output
23434                                  <1>       ; ((BX = R5)) output
23435                                  <1>         ;    ((Modified registers: DX, CX, BX, SI, DI, BP))
23436                                  <1>       ;    zf=1 -> block already in a I/O buffer
23437                                  <1>       ;    zf=0 -> a new I/O buffer has been allocated
23438                                  <1>       ;    ((DL = Device ID))
23439                                  <1>       ;    (((DH = 0 or 1)))
23440                                  <1>       ;    (((CX = previous value of word ptr [bufp])))
23441                                  <1>       ;    ((CX and DH will not be used after return)))
23442                                  <1>
23443                                  <1>       ;;push       esi ; ***
23444                                  <1>             ; mov r2,-(sp) / save r2 on stack
23445                                  <1>                     ; mov $340,*$ps / set processor priority to 7
23446                                  <1>       ; 20/07/2013
23447                                  <1>       ; 26/04/2013
23448 000062E8 0FB61D[2E740000]        <1>       movzx  ebx, byte [cdev] ; 0 or 1
23449 000062EF BF[30740000]            <1>       mov    edi, rdev  ; offset mdev = offset rdev + 1
23450 000062F4 01DF                    <1>       add    edi, ebx
23451                                  <1> bufaloc_0: ; 26/04/2013 !! here is called from bread or bwrite !!
23452                                  <1>                     ;; eDI points to device id.
23453 000062F6 0FB61F                  <1>       movzx  ebx, byte [edi] ; [EDI] -> rdev/mdev or brwdev
23454                                  <1>       ; 11/06/20215
23455 000062F9 80BB[6A6B0000]F0        <1>       cmp    byte [ebx+drv.status], 0F0h ; Drive not ready !
23456 00006300 720F                    <1>       jb     short bufaloc_9
23457 00006302 C705[9D740000]0F00-     <1>       mov    dword [u.error], ERR_DRV_NOT_RDY
23458 0000630A 0000                    <1>
23459 0000630C E929DDFFFF              <1>       jmp    error
23460                                  <1> bufaloc_9:
23461 00006311 89DA                    <1>       mov    edx, ebx ; dh = 0, dl = device number (0 to 5)
23462                                  <1> bufaloc_10: ; 02/07/2015
23463 00006313 31ED                    <1>       xor    ebp, ebp ; 0
23464 00006315 55                      <1>       push   ebp ; 0
23465 00006316 89E5                    <1>         mov     ebp, esp
23466                                  <1>       ;
23467                                  <1> bufaloc_1: ;1:
23468                                  <1>             ; clr -(sp) / vacant buffer
23469 00006318 BE[0A740000]            <1>         mov  esi, bufp
23470                                  <1>             ; mov $bufp,r2 / bufp contains pointers to I/O queue
23471                                  <1>                          ; / entrys in buffer area
```

```
23472                                  <1> bufaloc_2: ;2:
23473 0000631D 8B1E                    <1>       mov    ebx, [esi]
23474                                  <1>                    ; mov (r2)+,r5 / move pointer to word 1 of an I/O
23475                                  <1>                        ; queue entry into r5
23476 0000631F 66F70300F6              <1>       test   word [ebx], 0F600h
23477                                  <1>              ; bit $173000,(r5) / lock+keep+active+outstanding
23478 00006324 7503                    <1>         jnz  short bufaloc_3
23479                                  <1>              ; bne 3f / branch when
23480                                  <1>                     ; / any of bits 9,10,12,13,14,15 are set
23481                                  <1>                         ; / (i.e., buffer busy)
23482 00006326 897500                  <1>         mov     [ebp], esi ; pointer to I/0 queue entry
23483                                  <1>                     ; mov  r2,(sp) ;/ save pointer to last non-busy buffer
23484                                  <1>                        ; / found points to word 2 of I/O queue entry)
23485                                  <1> bufaloc_3: ;3:
23486                                  <1>       ;mov  dl, [edi] ; 26/04/2013
23487                                  <1>       ;
23488 00006329 3813                    <1>       cmp    [ebx], dl
23489                                  <1>              ; cmpb (r5),cdev / is device in I/O queue entry same
23490                                  <1>                           ; / as current device
23491 0000632B 7508                    <1>       jne    short bufaloc_4
23492                                  <1>                     ; bne 3f
23493 0000632D 394304                  <1>       cmp    [ebx+4], eax
23494                                  <1>                     ; cmp 2(r5),r1 / is block number in I/O queue entry,
23495                                  <1>                          ; / same as current block number
23496 00006330 7503                    <1>              jne    short bufaloc_4
23497                                  <1>              ; bne 3f
23498                                  <1>       ;add   esp, 4
23499 00006332 59                      <1>       pop    ecx
23500                                  <1>                     ; tst (sp)+ / bump stack pointer
23501 00006333 EB20                    <1>       jmp    short bufaloc_7 ; Retro Unix 8086 v1 modification
23502                                  <1>                           ; jump to bufaloc_6 in original Unix v1
23503                                  <1>                     ; br 1f / use this buffer
23504                                  <1> bufaloc_4: ;3:
23505 00006335 83C604                  <1>       add    esi, 4 ; 20/08/2015
23506                                  <1>       ;
23507 00006338 81FE[22740000]          <1>       cmp    esi, bufp + (nbuf*4)
23508                                  <1>              ; cmp r2,$bufp+nbuf+nbuf
23509 0000633E 72DD                    <1>       jb     short bufaloc_2
23510                                  <1>              ; blo 2b / go to 2b if r2 less than bufp+nbuf+nbuf (all
23511                                  <1>                       ; / buffers not checked)
23512 00006340 5E                      <1>         pop  esi
23513                                  <1>              ; mov (sp)+,r2 / once all bufs are examined move pointer
23514                                  <1>                           ; / to last free block
23515 00006341 09F6                    <1>              or     esi, esi
23516 00006343 7507                    <1>       jnz    short bufaloc_5
23517                                  <1>              ; bne 2f / if (sp) is non zero, i.e.,
23518                                  <1>                ; / if a free buffer is found branch to 2f
23519                                  <1>         ;; mov  ecx, [s.wait_]
23520 00006345 E80FF1FFFF              <1>       call   idle
23521                                  <1>              ; jsr r0,idle; s.wait+2 / idle if no free buffers
23522 0000634A EBC7                    <1>       jmp    short bufaloc_10 ; 02/07/2015
23523                                  <1>                     ; br 1b
23524                                  <1> bufaloc_5: ;2:
23525                                  <1>              ; tst (r0)+ / skip if warmed over buffer
23526 0000634C FEC6                    <1>       inc    dh ; Retro UNIX 8086 v1 modification
23527                                  <1> bufaloc_6: ;1:
23528 0000634E 8B1E                    <1>         mov        ebx, [esi]
23529                                  <1>              ; mov -(r2),r5 / put pointer to word 1 of I/O queue
23530                                  <1>                         ; / entry in r5
23531                                  <1>       ;; 26/04/2013
23532                                  <1>         ;mov dl, [edi] ; byte [rdev] or byte [mdev]
23533 00006350 8813                    <1>       mov    [ebx], dl
23534                                  <1>              ; movb cdev,(r5) / put current device number
23535                                  <1>                           ; / in I/O queue entry
23536 00006352 894304                  <1>       mov    [ebx+4], eax
23537                                  <1>              ; mov r1,2(r5) / move block number into word 2
23538                                  <1>                         ; / of I/O queue entry
23539                                  <1> bufaloc_7: ;1:
23540 00006355 81FE[0A740000]          <1>         cmp  esi, bufp
23541                                  <1>              ; cmp r2,$bufp / bump all entrys in bufp
23542                                  <1>                         ; / and put latest assigned
23543 0000635B 760A                    <1>       jna    short bufaloc_8
23544                                  <1>                     ; blos 1f / buffer on the top
23545                                  <1>                     ; / (this makes if the lowest priority)
23546 0000635D 83EE04                  <1>       sub    esi, 4
23547 00006360 8B0E                    <1>       mov    ecx, [esi]
23548 00006362 894E04                  <1>       mov    [esi+4], ecx
23549                                  <1>              ; mov -(r2),2(r2) / job for a particular device
23550 00006365 EBEE                    <1>       jmp    short bufaloc_7
23551                                  <1>              ; br 1b
23552                                  <1> bufaloc_8: ;1:
23553 00006367 891E                    <1>         mov  [esi], ebx
23554                                  <1>              ; mov r5,(r2)
23555                                  <1>       ;;pop  esi ; ***
23556                                  <1>                     ; mov (sp)+,r2 / restore r2
23557 00006369 08F6                    <1>              or     dh, dh ; 0 or 1 ?
23558                                  <1>              ; Retro UNIX 8086 v1 modification
23559                                  <1>              ; zf=1 --> block already is in an I/O buffer
23560                                  <1>              ; zf=0 --> a new I/O buffer has been allocated
23561 0000636B C3                      <1>       retn
23562                                  <1>              ; rts r0
23563                                  <1>
23564                                  <1> diskio:
23565                                  <1>       ; 10/07/2015
23566                                  <1>       ; 02/07/2015
23567                                  <1>       ; 16/06/2015
23568                                  <1>       ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23569                                  <1>       ;          (80386 protected mode modifications)
23570                                  <1>       ; 15/03/2013 - 29/04/2013 (Retro UNIX 8086 v1)
23571                                  <1>       ;
23572                                  <1>       ; Retro UNIX 8086 v1 feature only !
23573                                  <1>       ;
23574                                  <1>       ; Derived from proc_chs_read procedure of TRDOS DISKIO.ASM (2011)
23575                                  <1>       ; 04/07/2009 - 20/07/2011
23576                                  <1>       ;
```

```
23577                                  <1>       ; NOTE: Reads only 1 block/sector (sector/block size is 512 bytes)
23578                                  <1>       ;
23579                                  <1>         ; INPUTS ->
23580                                  <1>       ;         eBX = System I/O Buffer header address
23581                                  <1>       ;
23582                                  <1>         ; OUTPUTS -> cf=0 --> done
23583                                  <1>       ;           cf=1 ---> error code in AH
23584                                  <1>       ;
23585                                  <1>       ; (Modified registers: eAX, eCX, eDX)
23586                                  <1>
23587                                  <1> ;rw_disk_sector:
23588                                  <1>       ; 10/07/2015
23589                                  <1>       ; 02/07/2015
23590                                  <1>       ; 11/06/2015 - Retro UNIX 386 v1 - 'u8.s'
23591                                  <1>       ; 21/02/2015 ('dsectpm.s', 'read_disk_sector')
23592                                  <1>       ; 16/02/2015 (Retro UNIX 386 v1 test - 'unix386.s')
23593                                  <1>       ; 01/12/2014 - 18/01/2015 ('dsectrm2.s')
23594                                  <1>       ;
23595                                  <1>       ;mov   dx, 0201h ; Read 1 sector/block
23596 0000636C B602                   <1>       mov    dh, 2
23597 0000636E 668B03                 <1>       mov    ax, [ebx]
23598                                  <1>       ;
23599 00006371 56                     <1>       push   esi ; ****
23600 00006372 53                     <1>       push   ebx ; ***
23601                                  <1>       ;
23602 00006373 0FB6C8                 <1>       movzx  ecx, al
23603 00006376 89CE                   <1>       mov    esi, ecx
23604                                  <1>       ;
23605 00006378 38F1                   <1>       cmp    cl, dh ; 2
23606 0000637A 7202                   <1>       jb     short rwdsk0
23607 0000637C 047E                   <1>       add    al, 7Eh  ; 80h, 81h, 82h, 83h
23608                                  <1> rwdsk0:
23609 0000637E A2[1B6B0000]           <1>       mov    [drv], al
23610 00006383 81C6[6A6B0000]         <1>       add    esi, drv.status
23611                                  <1>       ; 11/06/2015
23612 00006389 803EF0                 <1>       cmp    byte [esi], 0F0h
23613 0000638C 720F                   <1>       jb      short rwdsk1
23614                                  <1>       ; 'drive not ready' error
23615 0000638E C705[9D740000]0F00-     <1>       mov    dword [u.error], ERR_DRV_NOT_RDY
23616 00006396 0000                   <1>
23617 00006398 E99DDCFFFF             <1>       jmp    error
23618                                  <1> rwdsk1:
23619 0000639D F6C402                 <1>       test   ah, 2
23620                                  <1>       ;test  ax, 200h ; Bit 9 of word 0 (status word)
23621                                  <1>                       ; write bit
23622 000063A0 7402                   <1>       jz     short rwdsk2
23623                                  <1>       ;test  ah, 4
23624                                  <1>       ;;test ax, 400h ; Bit 10 of word 0 (status word)
23625                                  <1>       ;             ; read bit
23626                                  <1>       ;jz    short diskio_ret
23627 000063A2 FEC6                   <1>       inc    dh ; 03h = write
23628                                  <1> rwdsk2:
23629 000063A4 88C2                   <1>       mov    dl, al
23630 000063A6 83C304                 <1>       add    ebx, 4 ; sector/block address/number pointer
23631 000063A9 8B03                   <1>       mov    eax, [ebx] ; sector/block number (LBA)
23632 000063AB C0E102                 <1>       shl    cl, 2
23633 000063AE 81C1[4E6B0000]         <1>       add    ecx, drv.size ; disk size
23634 000063B4 3B01                   <1>       cmp    eax, [ecx] ; Last sector + 1 (number of secs.)
23635 000063B6 720F                   <1>       jb      short rwdsk3
23636                                  <1>       ; 'out of volume' error
23637 000063B8 C705[9D740000]1000-     <1>       mov    dword [u.error], ERR_DEV_VOL_SIZE
23638 000063C0 0000                   <1>
23639 000063C2 E973DCFFFF             <1>       jmp    error
23640                                  <1> rwdsk3:
23641                                  <1>       ; 11/06/2015
23642 000063C7 83C304                 <1>       add    ebx, 4 ; buffer address
23643 000063CA C605[CE740000]04       <1>       mov    byte [retry_count], 4
23644 000063D1 F60601                 <1>       test   byte [esi], 1 ; LBA ready ?
23645 000063D4 7432                   <1>         jz     short rwdsk_chs
23646                                  <1> rwdsk_lba:
23647                                  <1>       ; LBA read/write (with private LBA function)
23648                                  <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
23649 000063D6 83C607                 <1>         add     esi, drv.error - drv.status ; 10/07/2015
23650 000063D9 89C1                   <1>       mov    ecx, eax ; sector number
23651                                  <1>       ; ebx = buffer (data) address
23652                                  <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
23653                                  <1> rwdsk_lba_retry:
23654                                  <1>       ;mov   dl, [drv]
23655                                  <1>              ; Function 1Bh = LBA read, 1Ch = LBA write
23656 000063DB B419                   <1>       mov    ah, 1Ch - 3h ; LBA write function number - 3
23657 000063DD 00F4                   <1>       add    ah, dh
23658 000063DF B001                   <1>       mov    al, 1
23659                                  <1>       ;int   13h
23660 000063E1 E8DCC3FFFF             <1>       call   int13h
23661 000063E6 8826                   <1>       mov    [esi], ah ; error code ; 10/07/2015
23662 000063E8 730E                   <1>       jnc    short rwdsk_lba_ok
23663 000063EA 80FC80                 <1>       cmp    ah, 80h ; time out ?
23664 000063ED 7408                   <1>         je     short rwdsk_lba_fails
23665 000063EF FE0D[CE740000]         <1>       dec    byte [retry_count]
23666 000063F5 7504                   <1>         jnz     short rwdsk_lba_reset ; 10/07/2015
23667                                  <1> rwdsk_lba_fails:
23668 000063F7 F9                     <1>       stc
23669                                  <1> rwdsk_lba_ok:
23670 000063F8 5B                     <1>       pop    ebx ; ***
23671 000063F9 5E                     <1>       pop    esi ; ****
23672 000063FA C3                     <1>       retn
23673                                  <1> rwdsk_lba_reset:
23674 000063FB B40D                   <1>       mov    ah, 0Dh ; Alternate reset
23675                                  <1>       ;int   13h
23676 000063FD E8C0C3FFFF             <1>         call int13h
23677 00006402 73D7                   <1>       jnc     short rwdsk_lba_retry
23678 00006404 8826                   <1>       mov    [esi], ah ; error code ; 10/07/2015
23679 00006406 EBF0                   <1>       jmp    short rwdsk_lba_ok
23680                                  <1>       ;
23681                                  <1>       ; CHS read (convert LBA address to CHS values)
```

```
23682                                   <1> rwdsk_chs:
23683                                   <1>       ; 10/07/2015
23684 00006408 81EE[6A6B0000]           <1>       sub    esi, drv.status
23685 0000640E 89F1                     <1>       mov    ecx, esi
23686 00006410 81C6[716B0000]           <1>       add    esi, drv.error
23687                                   <1>       ; 02/07/2015
23688                                   <1>       ; 16/06/2015
23689                                   <1>       ; 11/06/2015
23690 00006416 53                       <1>       push   ebx ; ** ; buffer
23691 00006417 D1E1                     <1>       shl    ecx, 1
23692 00006419 51                       <1>       push   ecx ; *
23693                                   <1>       ;
23694 0000641A 89CB                     <1>       mov    ebx, ecx
23695 0000641C 8835[CD740000]           <1>       mov    [rwdsk], dh ; 02/07/2015
23696 00006422 31D2                     <1>       xor    edx, edx ; 0
23697 00006424 29C9                     <1>       sub    ecx, ecx
23698 00006426 81C3[406B0000]           <1>         add     ebx, drv.spt
23699 0000642C 668B0B                   <1>       mov    cx, [ebx] ; sector per track
23700                                   <1>              ; EDX:EAX = LBA
23701 0000642F F7F1                     <1>       div    ecx
23702 00006431 88D1                     <1>       mov    cl, dl ; sector number - 1
23703 00006433 FEC1                     <1>       inc    cl     ; sector number (1 based)
23704 00006435 5B                       <1>       pop    ebx ; * ; 11/06/2015
23705 00006436 6651                     <1>       push   cx
23706 00006438 81C3[326B0000]           <1>         add     ebx, drv.heads
23707 0000643E 668B0B                   <1>       mov    cx, [ebx] ; heads
23708 00006441 31D2                     <1>       xor    edx, edx
23709                                   <1>              ; EAX = cylinders * heads + head
23710 00006443 F7F1                     <1>       div    ecx
23711 00006445 6659                     <1>       pop    cx     ; sector number
23712 00006447 88D6                     <1>       mov    dh, dl ; head number
23713 00006449 8A15[1B6B0000]           <1>       mov    dl, [drv]
23714 0000644F 88C5                     <1>       mov    ch, al ; cylinder (bits 0-7)
23715 00006451 C0E406                   <1>       shl    ah, 6
23716 00006454 08E1                     <1>       or     cl, ah ; cylinder (bits 8-9)
23717                                   <1>                     ; sector (bits 0-7)
23718 00006456 5B                       <1>       pop    ebx ; ** ; buffer ; 11/06/2015
23719                                   <1>              ; CL = sector (bits 0-5)
23720                                   <1>              ;      cylinder (bits 8-9 -> bits 6-7)
23721                                   <1>              ; CH = cylinder (bits 0-7)
23722                                   <1>              ; DH = head
23723                                   <1>              ; DL = drive
23724                                   <1>       ;
23725 00006457 C605[CE740000]04         <1>       mov    byte [retry_count], 4
23726                                   <1> rwdsk_retry:
23727 0000645E 8A25[CD740000]           <1>       mov    ah, [rwdsk] ; 02h = read, 03h = write
23728 00006464 B001                     <1>       mov    al, 1 ; sector count
23729                                   <1>       ;int   13h
23730 00006466 E857C3FFFF               <1>       call   int13h
23731 0000646B 8826                     <1>       mov    [esi], ah ; error code ; 10/07/2015
23732 0000646D 730E                     <1>       jnc    short rwdsk_ok ; ah = 0
23733 0000646F 80FC80                   <1>       cmp    ah, 80h ; time out ?
23734 00006472 7408                     <1>       je     short rwdsk_fails
23735 00006474 FE0D[CE740000]           <1>       dec    byte [retry_count]
23736 0000647A 7504                     <1>       jnz    short rwdsk_reset
23737                                   <1> rwdsk_fails:
23738 0000647C F9                       <1>       stc
23739                                   <1> rwdsk_ok:
23740 0000647D 5B                       <1>       pop    ebx ; ***
23741 0000647E 5E                       <1>       pop    esi ; ****
23742 0000647F C3                       <1>       retn
23743                                   <1> rwdsk_reset:
23744                                   <1>       ; 02/02/2015
23745 00006480 28E4                     <1>       sub    ah, ah
23746 00006482 80FA80                   <1>       cmp    dl, 80h
23747 00006485 7202                     <1>       jb     short rwdsk_fd_reset
23748 00006487 B40D                     <1>       mov    ah, 0Dh ; Alternate reset
23749                                   <1> rwdsk_fd_reset:
23750                                   <1>       ;int   13h
23751 00006489 E834C3FFFF               <1>         call int13h
23752 0000648E 73CE                     <1>       jnc    short rwdsk_retry
23753 00006490 8826                     <1>       mov    [esi], ah ; error code ; 10/07/2015
23754 00006492 EBE9                     <1>       jmp    short rwdsk_ok
23755                                   <1>
23756                                   <1>
23757                                   <1> ; Original UNIX v1 - drum (& disk) interrupt routine
23758                                   <1> ;     (Equivalent to IRQ 14 & IRQ 15 disk/hardware interrupts)
23759                                   <1> ;
23760                                   <1> ; This feature is not used in Retro UNIX 386 (& 8086) for now.
23761                                   <1> ; Because, current Retro UNIX 386 disk I/O -INT13H- routine is
23762                                   <1> ; derived from IBM PC AT -infact: XT286- BIOS source code, int 13h
23763                                   <1> ; that uses hardware -transfer has been completed-  interrupt inside it.
23764                                   <1> ; In a next Retro UNIX 386 version, these interrupts
23765                                   <1> ; (fdc_int, hdc1_int, hdc2_int) will be handled by a separate routine
23766                                   <1> ; as in original unix v1.
23767                                   <1> ; I am not removing IBM BIOS source code derivatives -compatible code-
23768                                   <1> ; for now, regarding the new/next 32 bit TRDOS project by me
23769                                   <1> ; (to keep source code files easy adaptable to 32 bit TRDOS.)
23770                                   <1> ;
23771                                   <1> ; Erdogan tan (10/07/2015)
23772                                   <1>
23773                                   <1> ;drum: / interrupt handler
23774                                   <1> ;       jsr     r0,setisp / save r1,r2,r3, and clockp on the stack
23775                                   <1> ;       jsr     r0,trapt; dcs; rfap; 1 / check for stray interrupt or
23776                                   <1> ;                                      / error
23777                                   <1> ;               br 3f / no, error
23778                                   <1> ;       br      2f / error
23779                                   <1> ;
23780                                   <1> ;disk:
23781                                   <1> ;       jsr     r0,setisp / save r1,r2,r3, and clockp on the stack
23782                                   <1> ;       jmp     *$0f
23783                                   <1> ;0:
23784                                   <1> ;       jsr     r0,trapt; rkcs; rkap; 2
23785                                   <1> ;                   br 3f / no, errors
23786                                   <1> ;       mov     $115,(r2) / drive reset, errbit was set
```

```
23787                                  <1> ;       mov     $1f,0b-2 / next time jmp *$0f is executed jmp will be
23788                                  <1> ;                        / to 1f
23789                                  <1> ;       br      4f
23790                                  <1> ;1:
23791                                  <1> ;       bit     $20000,rkcs
23792                                  <1> ;       beq     4f / wait for seek complete
23793                                  <1> ;       mov     $0b,0b-2
23794                                  <1> ;       mov     rkap,r1
23795                                  <1> ;2:
23796                                  <1> ;       bit     $3000,(r1) / are bits 9 or 10 set in the 1st word of
23797                                  <1> ;                          / the disk buffer
23798                                  <1> ;       bne     3f / no, branch ignore error if outstanding
23799                                  <1> ;       inc     r1
23800                                  <1> ;       asr     (r1)
23801                                  <1> ;       asr     (r1)
23802                                  <1> ;       asr     (r1) / reissue request
23803                                  <1> ;       dec     r1
23804                                  <1> ;3:
23805                                  <1> ;       bic     $30000,(r1) / clear bits 12 and 13 in 1st word of buffer
23806                                  <1> ;       mov     ac,-(sp)
23807                                  <1> ;       mov     mq,-(sp) / put these on the stack
23808                                  <1> ;       mov     sc,-(sp)
23809                                  <1> ;       jsr     r0,poke
23810                                  <1> ;       mov     (sp)+,sc
23811                                  <1> ;       mov     (sp)+,mq / pop them off stack
23812                                  <1> ;       mov     (sp)+,ac
23813                                  <1> ;4:
23814                                  <1> ;       jmp     retisp / u4-3
23815                                  <1> ;
23816                                  <1> ;trapt:                 / r2 points to the
23817                                  <1> ;       mov     (r0)+,r2 / device control register
23818                                  <1> ;       mov     *(r0)+,r1 / transaction pointer points to buffer
23819                                  <1> ;       tst     (sp)+
23820                                  <1> ;       tstb    (r2) / is ready bit of dcs set?
23821                                  <1> ;       bge     4b / device still active so branch
23822                                  <1> ;       bit     (r0),active / was device busy?
23823                                  <1> ;       beq     4b / no, stray interrupt
23824                                  <1> ;       bic     (r0)+,active / yes, set active to zero
23825                                  <1> ;       tst     (r2) / test the err(bit is) of dcs
23826                                  <1> ;       bge     2f / if no error jump to 2f
23827                                  <1> ;       tst     (r0)+ / skip on error
23828                                  <1> ; 2:
23829                                  <1> ;       jmp     (r0)
23830                                      %include 'u9.s'        ; 29/06/2015
23831                                  <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
23832                                  <1> ; Last Modification: 09/12/2015
23833                                  <1> ; ----------------------------------------------------------------------------
23834                                  <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
23835                                  <1> ; (v0.1 - Beginning: 11/07/2012)
23836                                  <1> ;
23837                                  <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
23838                                  <1> ; (Original) Source Code by Ken Thompson (1971-1972)
23839                                  <1> ; <Bell Laboratories (17/3/1972)>
23840                                  <1> ; <Preliminary Release of UNIX Implementation Document>
23841                                  <1> ;
23842                                  <1> ; Retro UNIX 8086 v1 - U9.ASM (01/09/2014) //// UNIX v1 -> u9.s
23843                                  <1> ;
23844                                  <1> ; ****************************************************************************
23845                                  <1>
23846                                  <1> getch:
23847                                  <1>       ; 30/06/2015
23848                                  <1>       ; 18/02/2015 - Retro UNIX 386 v1 - feature only!
23849 00006494 28C0                   <1>       sub    al, al ; 0
23850                                  <1> getch_q: ; 06/08/2015
23851 00006496 8A25[96700000]        <1>       mov    ah, [ptty] ; active (current) video page
23852 0000649C EB06                  <1>         jmp     short getc_n
23853                                  <1>
23854                                  <1> getc:
23855                                  <1>       ; 12/11/2015
23856                                  <1>       ; 15/09/2015
23857                                  <1>       ; 01/07/2015
23858                                  <1>       ; 30/06/2015
23859                                  <1>       ; 18/02/2015 (Retro UNIX 386 v1 - Beginning)
23860                                  <1>       ; 13/05/2013 - 04/07/2014 (Retro UNIX 8086 v1)
23861                                  <1>       ;
23862                                  <1>       ; Retro UNIX 8086 v1 modification !
23863                                  <1>       ;
23864                                  <1>       ; 'getc' gets (next) character
23865                                  <1>       ;      from requested TTY (keyboard) buffer
23866                                  <1>       ; INPUTS ->
23867                                  <1>       ;     [u.ttyn] = tty number (0 to 7) (8 is COM1, 9 is COM2)
23868                                  <1>       ;     AL=0 -> Get (next) character from requested TTY buffer
23869                                  <1>       ;      (Keyboard buffer will point to
23870                                  <1>       ;                    next character at next call)
23871                                  <1>       ;     AL=1 -> Test a key is available in requested TTY buffer
23872                                  <1>       ;      (Keyboard buffer will point to
23873                                  <1>       ;                    current character at next call)
23874                                  <1>       ; OUTPUTS ->
23875                                  <1>       ;     (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
23876                                  <1>       ;                     ZF=0 -> AX has (current) character
23877                                  <1>       ;      AL = ascii code
23878                                  <1>       ;      AH = scan code     (AH = line status for COM1 or COM2)
23879                                  <1>       ;                      (cf=1 -> error code/flags in AH)
23880                                  <1>       ; Original UNIX V1 'getc':
23881                                  <1>       ;            get a character off character list
23882                                  <1>       ;
23883                                  <1>       ; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
23884                                  <1>       ;
23885                                  <1>       ; 30/06/20045 (32 bit modifications)
23886                                  <1>       ; 16/07/2013
23887                                  <1>       ; mov  [getctty], ah
23888                                  <1>       ;
23889                                  <1>
23890 0000649E 8A25[9C740000]        <1>       mov    ah, [u.ttyn] ; 28/07/2013
```

```
23891                                  <1> getc_n:
23892                                  <1>       ; 30/06/2015
23893 000064A4 08E4                    <1>       or     ah, ah
23894 000064A6 740D                    <1>       jz     short getc0
23895 000064A8 D0E4                    <1>       shl    ah, 1
23896 000064AA 0FB6DC                  <1>       movzx  ebx, ah
23897 000064AD 81C3[98700000]          <1>       add    ebx, ttychr
23898 000064B3 EB05                    <1>       jmp    short getc1
23899                                  <1> getc0:
23900 000064B5 BB[98700000]            <1>       mov    ebx, ttychr
23901                                  <1> getc1:
23902 000064BA 668B0B                  <1>       mov    cx, [ebx]    ; ascii & scan code
23903                                  <1>                             ; (by kb_int)
23904 000064BD 6609C9                  <1>       or     cx, cx
23905 000064C0 7508                    <1>       jnz    short getc2
23906 000064C2 20C0                    <1>       and    al, al
23907 000064C4 7416                    <1>       jz     short getc_s
23908 000064C6 6631C0                  <1>       xor    ax, ax
23909 000064C9 C3                      <1>       retn
23910                                  <1> getc2:
23911 000064CA 20C0                    <1>       and    al, al
23912 000064CC 6689C8                  <1>       mov    ax, cx
23913 000064CF 66B90000                <1>       mov    cx, 0
23914 000064D3 7506                    <1>       jnz    short getc3
23915                                  <1> getc_sn:
23916 000064D5 66890B                  <1>       mov    [ebx], cx ; 0, reset
23917 000064D8 6639C8                  <1>       cmp    ax, cx  ; zf = 0
23918                                  <1> getc3:
23919 000064DB C3                      <1>       retn
23920                                  <1> getc_s:
23921                                  <1>       ; 12/11/2015
23922                                  <1>       ; 15/09/2015
23923                                  <1>       ; 01/07/2015
23924                                  <1>       ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
23925                                  <1>       ; 16/07/2013 - 14/02/2014 (Retro UNIX 8086 v1)
23926                                  <1>       ;
23927                                  <1>       ; tty  of the current process is not
23928                                  <1>       ; current tty (ptty); so, current process only
23929                                  <1>       ; can use keyboard input when its tty becomes
23930                                  <1>       ; current tty (ptty).
23931                                  <1>       ; 'sleep' is for preventing an endless lock
23932                                  <1>       ; during this tty input request.
23933                                  <1>       ; (Because, the user is not looking at the video page
23934                                  <1>       ; of the process to undersand there is a keyboard
23935                                  <1>       ; input request.)
23936                                  <1>       ;
23937                                  <1>       ;((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
23938                                  <1>       ;
23939                                  <1>       ; 05/10/2013
23940                                  <1>       ; ah = byte ptr [u.ttyn] ; (tty number)
23941                                  <1>       ;
23942                                  <1>       ; 10/10/2013
23943                                  <1> gcw0:
23944 000064DC B10A                    <1>       mov    cl, 10 ; ch = 0
23945                                  <1> gcw1:
23946                                  <1>       ; 12/11/2015
23947 000064DE E859DCFFFF              <1>       call intract ; jumps to 'sysexit' if [u.quit] = FFFFh
23948                                  <1>       ; 10/10/2013
23949 000064E3 E871EFFFFF              <1>       call   idle
23950 000064E8 668B03                  <1>       mov    ax, [ebx]    ; ascii & scan code
23951                                  <1>                             ; (by kb_int)
23952 000064EB 6609C0                  <1>       or     ax, ax
23953                                  <1> ;     jnz    short gcw3
23954 000064EE 7519                    <1>       jnz    short gcw2 ; 15/09/2015
23955                                  <1>       ; 30/06/2015
23956 000064F0 FEC9                    <1>       dec    cl
23957 000064F2 75EA                    <1>       jnz    short gcw1
23958                                  <1>       ;
23959 000064F4 8A25[9C740000]          <1>       mov    ah, [u.ttyn] ; 20/10/2013
23960                                  <1> ;     ; 10/12/2013
23961                                  <1> ;     cmp    ah, [ptty]
23962                                  <1> ;     jne    short gcw2
23963                                  <1> ;     ; 14/02/2014
23964                                  <1> ;     cmp    byte [u.uno], 1
23965                                  <1> ;     jna    short gcw0
23966                                  <1> ;gcw2:
23967 000064FA E8EEEFFFFF              <1>       call   sleep
23968                                  <1>       ;
23969                                  <1>       ; 20/09/2013
23970 000064FF 8A25[9C740000]          <1>       mov    ah, [u.ttyn]
23971 00006505 30C0                    <1>       xor    al, al
23972 00006507 EB9B                    <1>       jmp    short getc_n
23973                                  <1> ;gcw3:
23974                                  <1> gcw2:       ; 15/09/2015
23975                                  <1>       ; 10/10/2013
23976 00006509 30C9                    <1>       xor    cl, cl
23977 0000650B EBC8                    <1>       jmp    short getc_sn
23978                                  <1>
23979                                  <1> sndc:   ; <Send character>
23980                                  <1>       ;
23981                                  <1>       ; 17/11/2015
23982                                  <1>       ; 16/11/2015
23983                                  <1>       ; 11/11/2015
23984                                  <1>       ; 10/11/2015
23985                                  <1>       ; 09/11/2015
23986                                  <1>       ; 08/11/2015
23987                                  <1>       ; 07/11/2015
23988                                  <1>       ; 06/11/2015 (serial4.asm, 'sendchr')
23989                                  <1>       ; 29/10/2015
23990                                  <1>       ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
23991                                  <1>       ; 14/05/2013 - 28/07/2014 (Retro UNIX 8086 v1)
23992                                  <1>       ;
23993                                  <1>       ; Retro UNIX 8086 v1 feature only !
23994                                  <1>       ;
23995                                  <1>       ; ah = [u.ttyn]
```

```
23996                                  <1>       ;
23997                                  <1>       ; 30/06/2015
23998 0000650D 80EC08                  <1>       sub    ah, 8 ; ; 0 = tty8 or 1 = tty9
23999                                  <1>       ; 07/11/2015
24000 00006510 0FB6DC                  <1>       movzx  ebx, ah ; serial port index (0 or 1)
24001                                  <1> sndc0:
24002                                  <1>       ; 07/11/2015
24003 00006513 E82EF0FFFF              <1>       call   isintr ; quit (ctrl+break) check
24004 00006518 7405                    <1>       jz     short sndc1
24005 0000651A E81DDCFFFF              <1>       call   intract ; quit (ctrl+break) check
24006                                  <1>       ; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
24007                                  <1> sndc1:
24008                                  <1>       ; 16/11/2015
24009 0000651F 6689C1                  <1>       mov    cx, ax ; *** al = character (to be sent)
24010                                  <1> sndcx:
24011 00006522 8A83[DA700000]          <1>       mov    al, [ebx+schar] ; last sent character
24012 00006528 8AA3[D8700000]          <1>       mov    ah, [ebx+rchar] ; last received character
24013                                  <1>       ;
24014                                  <1>       ; 17/11/2015
24015                                  <1>       ; check 'request for response' status
24016 0000652E 80BB[D4700000]00        <1>       cmp    byte [ebx+req_resp], 0
24017 00006535 740A                    <1>       jz     short query
24018                                  <1> response:
24019 00006537 FE05[D7700000]          <1>       inc    byte [comqr] ; query or response status
24020 0000653D B0FF                    <1>       mov    al, 0FFh
24021 0000653F EB14                    <1>       jmp    short sndc3
24022                                  <1> query:
24023 00006541 08C0                    <1>       or     al, al  ; 0 = query (also end of text)
24024 00006543 750E                    <1>       jnz    short sndc2 ; normal character
24025                                  <1>       ;cmp   ah, 0FFh     ; is it responded by terminal ?
24026                                  <1>       ;je    short sndc2  ; yes, already responded
24027                                  <1>       ; 16/11/2015
24028                                  <1>       ; query: request for response (again)
24029 00006545 8883[D8700000]          <1>       mov    [ebx+rchar], al ; 0 ; reset
24030 0000654B FE05[D7700000]          <1>       inc    byte [comqr] ; query or response status
24031 00006551 EB02                    <1>       jmp    short sndc3
24032                                  <1> sndc2:
24033 00006553 88C8                    <1>       mov    al, cl       ; *** character (to be sent)
24034                                  <1> sndc3:
24035 00006555 8883[DA700000]          <1>       mov    [ebx+schar], al ; current character (to be sent)
24036 0000655B 88D8                    <1>       mov    al, bl ; 0 or 1 (serial port index)
24037                                  <1>       ; 30/06/2015
24038 0000655D E87DD5FFFF              <1>       call   sp_status ; get serial port status
24039                                  <1>       ; AL = Line status, AH = Modem status
24040                                  <1>       ; 07/11/2015
24041 00006562 A880                    <1>       test   al, 80h
24042 00006564 7504                    <1>       jnz    short sndc4
24043 00006566 A820                    <1>       test   al, 20h      ; Transmitter holding register empty ?
24044 00006568 751D                    <1>       jnz    short sndc5
24045                                  <1> sndc4:       ; Check line status again
24046                                  <1>       ; 16/11/2015
24047 0000656A 6651                    <1>       push   cx
24048 0000656C B906000000              <1>       mov    ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
24049 00006571 E873B0FFFF              <1>       call   WAITF
24050 00006576 6659                    <1>       pop    cx
24051                                  <1>       ;
24052 00006578 88D8                    <1>       mov    al, bl ; 0 or 1 (serial port index)
24053 0000657A E860D5FFFF              <1>       call   sp_status ; get serial port status
24054                                  <1>       ; 16/11/2015
24055                                  <1>       ; 09/11/2015
24056                                  <1>       ; 08/11/2015
24057 0000657F A880                    <1>       test   al, 80h      ; time out error
24058 00006581 756C                    <1>         jnz     short sndc7
24059 00006583 A820                    <1>       test   al, 20h      ; Transmitter holding register empty ?
24060 00006585 7468                    <1>         jz   short sndc7
24061                                  <1> sndc5:
24062 00006587 8A83[DA700000]          <1>       mov    al, [ebx+schar] ; character (to be sent)
24063 0000658D 66BAF803                <1>       mov    dx, 3F8h   ; data port (COM2)
24064 00006591 28DE                    <1>       sub    dh, bl
24065 00006593 EE                      <1>       out    dx, al    ; send on serial port
24066                                  <1>       ; 10/11/2015
24067                                  <1>       ; delay for 3*30 (3*(15..80)) micro seconds
24068                                  <1>       ; (to improve text flow to the terminal)
24069                                  <1>       ; ('diskette.inc': 'WAITF')
24070                                  <1>       ; Uses port 61h, bit 4 to have CPU speed independent waiting.
24071                                  <1>       ; (refresh periods = 1 per 30 microseconds on most machines)
24072 00006594 6651                    <1>       push   cx
24073 00006596 B906000000              <1>       mov    ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
24074 0000659B E849B0FFFF              <1>       call   WAITF
24075 000065A0 6659                    <1>       pop    cx
24076                                  <1>       ;
24077                                  <1>       ; 07/11/2015
24078 000065A2 88D8                    <1>       mov    al, bl ; al = 0 (tty8) or 1 (tty9)
24079                                  <1>       ;
24080 000065A4 E836D5FFFF              <1>       call   sp_status ; get serial port status
24081                                  <1>       ; AL = Line status, AH = Modem status
24082                                  <1>       ;
24083 000065A9 E898EFFFFF              <1>       call   isintr ; quit (ctrl+break) check
24084 000065AE 7405                    <1>       jz     short sndc6
24085 000065B0 E887DBFFFF              <1>       call   intract ; quit (ctrl+break) check
24086                                  <1>       ; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
24087                                  <1> sndc6:
24088 000065B5 3C80                    <1>       cmp    al, 80h
24089 000065B7 7336                    <1>       jnb    short sndc7
24090                                  <1>       ;
24091 000065B9 803D[D7700000]01        <1>       cmp    byte [comqr], 1 ; 'query or response' ?
24092 000065C0 7248                    <1>       jb     short sndc8  ; no, normal character
24093 000065C2 883D[D7700000]          <1>       mov    byte [comqr], bh ; 0 ; reset
24094                                  <1>       ; 17/11/2015
24095 000065C8 E88CEEFFFF              <1>       call   idle
24096                                  <1>       ;
24097 000065CD 38BB[DA700000]          <1>       cmp    [ebx+schar], bh ; 0 ; query ?
24098 000065D3 0F877AFFFFFF            <1>         ja      sndc2       ; response (will be followed by
24099                                  <1>                         ; a normal character)
24100                                  <1>       ; Query request must be responded by the terminal
```

```
24101                                  <1>       ; before sending a normal character !
24102 000065D9 53                      <1>       push   ebx
24103 000065DA 6651                    <1>       push   cx ; *** cl = character (to be sent)
24104 000065DC 8A25[9C740000]          <1>       mov    ah, [u.ttyn]
24105 000065E2 E806EFFFFF              <1>       call   sleep ; this process will be awakened by
24106                                  <1>                    ; received data available interrupt
24107 000065E7 6659                    <1>       pop    cx ; *** cl = character (to be sent)
24108 000065E9 5B                      <1>       pop    ebx
24109 000065EA E933FFFFFF              <1>         jmp  sndcx
24110                                  <1> sndc7:
24111                                  <1>        ; 16/11/2015
24112 000065EF 803D[D7700000]01        <1>       cmp    byte [comqr], 1 ; 'query or response' ?
24113 000065F6 7213                    <1>       jb     short sndc9  ; no
24114                                  <1>       ;
24115 000065F8 88BB[D8700000]          <1>       mov    [ebx+rchar], bh ; 0 ; reset
24116 000065FE 88BB[DA700000]          <1>       mov    [ebx+schar], bh ; 0 ; reset
24117                                  <1>       ;
24118 00006604 883D[D7700000]          <1>       mov    byte [comqr], bh ; 0 ; reset
24119                                  <1> sndc8:
24120 0000660A F5                      <1>       cmc  ; jnc -> jc, jb -> jnb
24121                                  <1> sndc9:
24122                                  <1>       ; AL = Line status, AH = Modem status
24123 0000660B C3                      <1>       retn
24124                                  <1>
24125                                  <1> putc:
24126                                  <1>       ; 13/08/2015
24127                                  <1>       ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
24128                                  <1>       ; 15/05/2013 - 27/07/2014 (Retro UNIX 8086 v1)
24129                                  <1>       ;
24130                                  <1>       ; Retro UNIX 8086 v1 modification !
24131                                  <1>       ;
24132                                  <1>       ; 'putc' puts a character
24133                                  <1>       ;      onto requested (tty) video page or
24134                                  <1>       ;      serial port
24135                                  <1>       ; INPUTS ->
24136                                  <1>       ;     AL = ascii code of the character
24137                                  <1>       ;     AH = video page (tty) number (0 to 7)
24138                                  <1>       ;                  (8 is COM1, 9 is COM2)
24139                                  <1>       ; OUTPUTS ->
24140                                  <1>       ;    (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
24141                                  <1>       ;                       ZF=0 -> AX has (current) character
24142                                  <1>       ;     cf=0 and AH = 0 -> no error
24143                                  <1>       ;     cf=1 and AH > 0 -> error (only for COM1 and COM2)

24144                                  <1>       ;
24145                                  <1>       ; Original UNIX V1 'putc':
24146                                  <1>       ;     put a character at the end of character list
24147                                  <1>       ;
24148                                  <1>       ; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
24149                                  <1>       ;
24150 0000660C 80FC07                  <1>       cmp    ah, 7
24151 0000660F 0F87F8FEFFFF            <1>         ja      sndc
24152                                  <1>       ; 30/06/2015
24153 00006615 0FB6DC                  <1>       movzx  ebx, ah
24154                                  <1>       ; 13/08/2015
24155 00006618 B407                    <1>       mov    ah, 07h ; black background, light gray character color
24156 0000661A E9BBAEFFFF              <1>       jmp    write_tty ; 'video.inc'
24157                                  <1>
24158                                  <1> get_cpos:
24159                                  <1>       ; 29/06/2015 (Retro UNIX 386 v1)
24160                                  <1>       ; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
24161                                  <1>       ;
24162                                  <1>       ; INPUT -> bl = video page number
24163                                  <1>       ; RETURN -> dx = cursor position
24164                                  <1>
24165 0000661F 53                      <1>       push   ebx
24166 00006620 83E30F                  <1>       and    ebx, 0Fh ; 07h ; tty0 to tty7
24167 00006623 D0E3                    <1>       shl    bl, 1
24168 00006625 81C3[86700000]          <1>       add    ebx, cursor_posn
24169 0000662B 668B13                  <1>       mov    dx, [ebx]
24170 0000662E 5B                      <1>       pop    ebx
24171 0000662F C3                      <1>       retn
24172                                  <1>
24173                                  <1> read_ac_current:
24174                                  <1>       ; 29/06/2015 (Retro UNIX 386 v1)
24175                                  <1>       ; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
24176                                  <1>       ;
24177                                  <1>       ; INPUT -> bl = video page number
24178                                  <1>       ; RETURN -> ax = character (al) and attribute (ah)
24179                                  <1>
24180 00006630 E826B0FFFF              <1>       call   find_position ; 'video.inc'
24181                                  <1>       ; dx = status port
24182                                  <1>       ; esi = cursor location/address
24183 00006635 81C600800B00            <1>       add    esi, 0B8000h ; 30/08/2014 (Retro UNIX 386 v1)
24184 0000663B 668B06                  <1>       mov    ax, [esi]    ; get the character and attribute
24185 0000663E C3                      <1>       retn
24186                                  <1>
24187                                  <1> syssleep:
24188                                  <1>       ; 29/06/2015 - (Retro UNIX 386 v1)
24189                                  <1>       ; 11/06/2014 - (Retro UNIX 8086 v1)
24190                                  <1>       ;
24191                                  <1>       ; Retro UNIX 8086 v1 feature only
24192                                  <1>       ; (INPUT -> none)
24193                                  <1>       ;
24194 0000663F 0FB61D[97740000]        <1>       movzx  ebx, byte [u.uno] ; process number
24195 00006646 8AA3[95710000]          <1>       mov    ah, [ebx+p.ttyc-1] ; current/console tty
24196 0000664C E89CEEFFFF              <1>       call   sleep
24197 00006651 E904DAFFFF              <1>       jmp    sysret
24198                                  <1>
24199                                  <1> vp_clr:
24200                                  <1>       ; Reset/Clear Video Page
24201                                  <1>       ;
24202                                  <1>       ; 30/06/2015 - (Retro UNIX 386 v1)
24203                                  <1>       ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
24204                                  <1>       ;
```

```
24205                                  <1>       ; Retro UNIX 8086 v1 feature only !
24206                                  <1>       ;
24207                                  <1>       ; INPUTS ->
24208                                  <1>       ;   BL = video page number
24209                                  <1>       ;
24210                                  <1>       ; OUTPUT ->
24211                                  <1>       ;   none
24212                                  <1>       ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
24213                                  <1>       ;
24214                                  <1>       ; 04/12/2013
24215 00006656 28C0                   <1>       sub   al, al
24216                                  <1>       ; al = 0 (clear video page)
24217                                  <1>       ; bl = video page
24218 00006658 B407                   <1>       mov   ah, 07h
24219                                  <1>       ; ah = 7 (attribute/color)
24220 0000665A 6631C9                 <1>       xor   cx, cx ; 0, left upper column (cl) & row (cl)
24221 0000665D 66BA4F18               <1>       mov   dx, 184Fh ; right lower column & row (dl=24, dh=79)
24222 00006661 E821B0FFFF             <1>       call  scroll_up
24223                                  <1>       ; bl = video page
24224 00006666 6631D2                 <1>       xor   dx, dx ; 0 (cursor position)
24225 00006669 E990AFFFFF             <1>       jmp   set_cpos
24226                                  <1>
24227                                  <1> sysmsg:
24228                                  <1>       ; 11/11/2015
24229                                  <1>       ; 01/07/2015 - (Retro UNIX 386 v1 feature only!)
24230                                  <1>       ; Print user-application message on user's console tty
24231                                  <1>       ;
24232                                  <1>       ; Input -> EBX = Message address
24233                                  <1>       ;         ECX = Message length (max. 255)
24234                                  <1>       ;         DL = Color (IBM PC Rombios color attributes)
24235                                  <1>       ;
24236 0000666E 81F9FF000000           <1>       cmp   ecx, MAX_MSG_LEN ; 255
24237 00006674 0F87E0D9FFFF           <1>       ja    sysret ; nothing to do with big message size
24238 0000667A 08C9                   <1>       or    cl, cl
24239 0000667C 0F84D8D9FFFF           <1>       jz    sysret
24240 00006682 20D2                   <1>       and   dl, dl
24241 00006684 7502                   <1>       jnz   short sysmsg0
24242 00006686 B207                   <1>       mov   dl, 07h ; default color
24243                                  <1>             ; (black background, light gray character)
24244                                  <1> sysmsg0:
24245 00006688 891D[68740000]         <1>       mov   [u.base], ebx
24246 0000668E 8815[97700000]         <1>       mov   [ccolor], dl ; color attributes
24247 00006694 89E5                   <1>       mov   ebp, esp
24248 00006696 31DB                   <1>       xor   ebx, ebx ; 0
24249 00006698 891D[70740000]         <1>       mov   [u.nread], ebx ; 0
24250                                  <1>       ;
24251 0000669E 381D[AF740000]         <1>       cmp   [u.kcall], bl ; 0
24252 000066A4 7769                   <1>       ja    short sysmsgk ; Temporary (01/07/2015)
24253                                  <1>       ;
24254 000066A6 890D[6C740000]         <1>       mov   [u.count], ecx
24255 000066AC 41                     <1>       inc   ecx ; + 00h ; ASCIZZ
24256 000066AD 29CC                   <1>       sub   esp, ecx
24257 000066AF 89E7                   <1>       mov   edi, esp
24258 000066B1 89E6                   <1>       mov   esi, esp
24259 000066B3 66891D[AD740000]       <1>       mov   [u.pcount], bx ; reset page (phy. addr.) counter
24260                                  <1>       ; 11/11/2015
24261 000066BA 8A25[78740000]         <1>       mov   ah, [u.ttyp] ; recent open tty
24262                                  <1>       ; 0 = none
24263 000066C0 FECC                   <1>       dec   ah
24264 000066C2 790C                   <1>       jns   short sysmsg1
24265 000066C4 8A1D[97740000]         <1>       mov   bl, [u.uno] ; process number
24266 000066CA 8AA3[95710000]         <1>       mov   ah, [ebx+p.ttyc-1] ; user's (process's) console tty
24267                                  <1> sysmsg1:
24268 000066D0 8825[9C740000]         <1>       mov   [u.ttyn], ah
24269                                  <1> sysmsg2:
24270 000066D6 E89CF5FFFF             <1>       call  cpass
24271 000066DB 7416                   <1>       jz    short sysmsg5
24272 000066DD AA                     <1>       stosb
24273 000066DE 20C0                   <1>       and   al, al
24274 000066E0 75F4                   <1>       jnz   short sysmsg2
24275                                  <1> sysmsg3:
24276 000066E2 80FC07                 <1>       cmp   ah, 7 ; tty number
24277 000066E5 7711                   <1>       ja    short sysmsg6 ; serial port
24278 000066E7 E83E000000             <1>       call  print_cmsg
24279                                  <1> sysmsg4:
24280 000066EC 89EC                   <1>       mov   esp, ebp
24281 000066EE E967D9FFFF             <1>       jmp   sysret
24282                                  <1> sysmsg5:
24283 000066F3 C60700                 <1>       mov   byte [edi], 0
24284 000066F6 EBEA                   <1>       jmp   short sysmsg3
24285                                  <1> sysmsg6:
24286 000066F8 8A06                   <1>       mov   al, [esi]
24287 000066FA E80EFEFFFF             <1>       call  sndc
24288 000066FF 72EB                   <1>       jc    short sysmsg4
24289 00006701 803E00                 <1>       cmp   byte [esi], 0  ; 0 is stop character
24290 00006704 76E6                   <1>       jna   short sysmsg4
24291 00006706 46                     <1>       inc   esi
24292 00006707 8A25[9C740000]         <1>       mov   ah, [u.ttyn]
24293 0000670D EBE9                   <1>       jmp   short sysmsg6
24294                                  <1>
24295                                  <1> sysmsgk: ; Temporary (01/07/2015)
24296                                  <1>       ; The message has been sent by Kernel (ASCIIZ string)
24297                                  <1>       ; (ECX -character count- will not be considered)
24298 0000670F 8B35[68740000]         <1>       mov   esi, [u.base]
24299 00006715 8A25[96700000]         <1>       mov   ah, [ptty] ; present/current screen (video page)
24300 0000671B 8825[9C740000]         <1>       mov   [u.ttyn], ah
24301 00006721 C605[AF740000]00       <1>       mov   byte [u.kcall], 0
24302 00006728 EBB8                   <1>       jmp   short sysmsg3
24303                                  <1>
24304                                  <1>
24305                                  <1> print_cmsg:
24306                                  <1>       ; 01/07/2015 (retro UNIX 386 v1 feature only !)
24307                                  <1>       ;
24308                                  <1>       ; print message (on user's console tty)
24309                                  <1>       ;     with requested color
```

```
24310                                  <1>       ;
24311                                  <1>       ; INPUTS:
24312                                  <1>       ;     esi = message address
24313                                  <1>       ;     [u.ttyn] = tty number (0 to 7)
24314                                  <1>       ;     [ccolor] = color attributes (IBM PC BIOS colors)
24315                                  <1>       ;
24316 0000672A AC                       <1>       lodsb
24317                                  <1> pcmsg1:
24318 0000672B 56                       <1>       push   esi
24319 0000672C 0FB61D[9C740000]         <1>         movzx   ebx, byte [u.ttyn]
24320 00006733 8A25[97700000]           <1>       mov    ah, [ccolor]
24321 00006739 E89CADFFFF               <1>       call   write_tty
24322 0000673E 5E                       <1>       pop    esi
24323 0000673F AC                       <1>       lodsb
24324 00006740 20C0                     <1>       and    al, al  ; 0
24325 00006742 75E7                     <1>       jnz    short pcmsg1
24326 00006744 C3                       <1>       retn
24327                                  <1>
24328                                  <1> sysgeterr:
24329                                  <1>       ; 09/12/2015
24330                                  <1>       ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
24331                                  <1>       ; Get last error number or page fault count
24332                                  <1>       ; (for debugging)
24333                                  <1>       ;
24334                                  <1>       ; Input -> EBX = return type
24335                                  <1>       ;        0 = last error code (which is in 'u.error')
24336                                  <1>       ;        FFFFFFFFh = page fault count for running process
24337                                  <1>       ;        FFFFFFFEh = total page fault count
24338                                  <1>       ;        1 .. FFFFFFFDh = undefined
24339                                  <1>       ;
24340                                  <1>       ; Output -> EAX = last error number or page fault count
24341                                  <1>       ;        (depending on EBX input)
24342                                  <1>       ;
24343 00006745 21DB                     <1>       and    ebx, ebx
24344 00006747 750B                     <1>       jnz    short glerr_2
24345                                  <1> glerr_0:
24346 00006749 A1[9D740000]             <1>       mov    eax, [u.error]
24347                                  <1> glerr_1:
24348 0000674E A3[48740000]             <1>       mov    [u.r0], eax
24349 00006753 C3                       <1>       retn
24350                                  <1> glerr_2:
24351 00006754 43                       <1>       inc    ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
24352 00006755 74FD                     <1>       jz     short glerr_2 ; page fault count for process
24353 00006757 43                       <1>       inc    ebx ; FFFFFFFFh -> 0
24354 00006758 75EF                     <1>       jnz    short glerr_0
24355 0000675A A1[30850000]             <1>       mov    eax, [PF_Count] ; total page fault count
24356 0000675F EBED                     <1>         jmp     short glerr_1
24357                                  <1> glerr_3:
24358 00006761 A1[B1740000]             <1>       mov    eax, [u.pfcount]
24359 00006766 EBE6                     <1>       jmp    short glerr_1
24360
24361                                      ; 07/03/2015
24362                                      ; Temporary Code
24363                                      display_disks:
24364 00006768 803D[1E6B0000]00                 cmp    byte [fd0_type], 0
24365 0000676F 7605                             jna    short ddsks1
24366 00006771 E87D000000                       call   pdskm
24367                                      ddsks1:
24368 00006776 803D[1F6B0000]00                 cmp    byte [fd1_type], 0
24369 0000677D 760C                             jna    short ddsks2
24370 0000677F C605[4B6D0000]31                 mov    byte [dskx], '1'
24371 00006786 E868000000                       call   pdskm
24372                                      ddsks2:
24373 0000678B 803D[206B0000]00                 cmp    byte [hd0_type], 0
24374 00006792 7654                             jna    short ddsk6
24375 00006794 66C705[496D0000]68-              mov    word [dsktype], 'hd'
24376 0000679C 64
24377 0000679D C605[4B6D0000]30                 mov    byte [dskx], '0'
24378 000067A4 E84A000000                       call   pdskm
24379                                      ddsks3:
24380 000067A9 803D[216B0000]00                 cmp    byte [hd1_type], 0
24381 000067B0 7636                             jna    short ddsk6
24382 000067B2 C605[4B6D0000]31                 mov    byte [dskx], '1'
24383 000067B9 E835000000                       call   pdskm
24384                                      ddsks4:
24385 000067BE 803D[226B0000]00                 cmp    byte [hd2_type], 0
24386 000067C5 7621                             jna    short ddsk6
24387 000067C7 C605[4B6D0000]32                 mov    byte [dskx], '2'
24388 000067CE E820000000                       call   pdskm
24389                                      ddsks5:
24390 000067D3 803D[236B0000]00                 cmp    byte [hd3_type], 0
24391 000067DA 760C                             jna    short ddsk6
24392 000067DC C605[4B6D0000]33                 mov    byte [dskx], '3'
24393 000067E3 E80B000000                       call   pdskm
24394                                      ddsk6:
24395 000067E8 BE[5A6D0000]                     mov    esi, nextline
24396 000067ED E806000000                       call   pdskml
24397                                      pdskm_ok:
24398 000067F2 C3                               retn
24399                                      pdskm:
24400 000067F3 BE[476D0000]                     mov    esi, dsk_ready_msg
24401                                      pdskml:
24402 000067F8 AC                               lodsb
24403 000067F9 08C0                             or     al, al
24404 000067FB 74F5                             jz     short pdskm_ok
24405 000067FD 56                               push   esi
24406 000067FE 31DB                             xor    ebx, ebx ; 0
24407                                                          ; Video page 0 (bl=0)
24408 00006800 B407                             mov    ah, 07h ; Black background,
24409                                                          ; light gray forecolor
24410 00006802 E8D3ACFFFF                       call   write_tty
24411 00006807 5E                               pop    esi
24412 00006808 EBEE                             jmp    short pdskml
24413
24414 0000680A 90<rept>                  align 16
```

```
24415
24416                                   gdt:  ; Global Descriptor Table
24417                                         ; (30/07/2015, conforming cs)
24418                                         ; (26/03/2015)
24419                                         ; (24/03/2015, tss)
24420                                         ; (19/03/2015)
24421                                         ; (29/12/2013)
24422                                         ;
24423 00006810 0000000000000000              dw 0, 0, 0, 0       ; NULL descriptor
24424                                         ; 18/08/2014
24425                                                   ; 8h kernel code segment, base = 00000000h
24426 00006818 FFFF0000009ACF00              dw 0FFFFh, 0, 9A00h, 00CFh; KCODE
24427                                                   ; 10h kernel data segment, base = 00000000h
24428 00006820 FFFF00000092CF00              dw 0FFFFh, 0, 9200h, 00CFh; KDATA
24429                                                   ; 1Bh user code segment, base address = 400000h ; CORE
24430 00006828 FFFB000040FACF00              dw 0FBFFh, 0, 0FA40h, 00CFh      ; UCODE
24431                                                   ; 23h user data segment, base address = 400000h ; CORE
24432 00006830 FFFB000040F2CF00              dw 0FBFFh, 0, 0F240h, 00CFh      ; UDATA
24433                                                   ; Task State Segment
24434 00006838 6700                          dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
24435                                                    ;  no IO permission in ring 3)
24436                                   gdt_tss0:
24437 0000683A 0000                          dw 0  ; TSS base address, bits 0-15
24438                                   gdt_tss1:
24439 0000683C 00                            db 0  ; TSS base address, bits 16-23
24440                                                   ; 49h
24441 0000683D E9                            db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
24442 0000683E 00                            db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
24443                                   gdt_tss2:
24444 0000683F 00                            db 0  ; TSS base address, bits 24-31
24445
24446                                   gdt_end:
24447                                         ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
24448                                                               ;; Type= 1 (code)/C=0/R=1/A=0
24449                                               ; P= Present, DPL=0=ring 0,  1= user (0= system)
24450                                               ; 1= Code C= non-Conforming, R= Readable, A = Accessed
24451
24452                                         ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
24453                                                               ;; Type= 0 (data)/E=0/W=1/A=0
24454                                               ; P= Present, DPL=0=ring 0,  1= user (0= system)
24455                                               ; 0= Data E= Expansion direction (1= down, 0= up)
24456                                               ; W= Writeable, A= Accessed
24457
24458                                         ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
24459                                                               ;; Type= 1 (code)/C=0/R=1/A=0
24460                                               ; P= Present, DPL=3=ring 3,  1= user (0= system)
24461                                               ; 1= Code C= non-Conforming, R= Readable, A = Accessed
24462
24463                                         ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
24464                                                               ;; Type= 0 (data)/E=0/W=1/A=0
24465                                               ; P= Present, DPL=3=ring 3,  1= user (0= system)
24466                                               ; 0= Data E= Expansion direction (1= down, 0= up)
24467
24468                                         ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)
24469
24470                                               ;; Limit = FFFFFh (=> FFFFFh+1= 100000h) // bits 0-15, 48-51 //
24471                                               ;      = 100000h * 1000h (G=1) = 4GB
24472                                               ;; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //
24473                                               ;      = FFC00h * 1000h (G=1) = 4GB - 4MB
24474                                               ; G= Granularity (1= 4KB), B= Big (32 bit),
24475                                               ; AVL= Available to programmers
24476
24477                                   gdtd:
24478 00006840 2F00                             dw gdt_end - gdt - 1    ; Limit (size)
24479 00006842 [10680000]                       dd gdt                  ; Address of the GDT
24480
24481                                         ; 20/08/2014
24482                                   idtd:
24483 00006846 FF01                             dw idt_end - idt - 1    ; Limit (size)
24484 00006848 [006E0000]                       dd idt                  ; Address of the IDT
24485
24486                                   Align 4
24487
24488                                         ; 21/08/2014
24489                                   ilist:
24490                                               ;times       32 dd cpu_except ; INT 0 to INT 1Fh
24491                                               ;
24492                                               ; Exception list
24493                                               ; 25/08/2014
24494 0000684C [F1080000]                           dd     exc0   ; 0h,  Divide-by-zero Error
24495 00006850 [F8080000]                           dd     exc1
24496 00006854 [FF080000]                           dd     exc2
24497 00006858 [06090000]                           dd     exc3
24498 0000685C [0A090000]                           dd     exc4
24499 00006860 [0E090000]                           dd     exc5
24500 00006864 [12090000]                           dd     exc6   ; 06h,  Invalid Opcode
24501 00006868 [16090000]                           dd     exc7
24502 0000686C [1A090000]                           dd     exc8
24503 00006870 [1E090000]                           dd     exc9
24504 00006874 [22090000]                           dd     exc10
24505 00006878 [26090000]                           dd     exc11
24506 0000687C [2A090000]                           dd     exc12
24507 00006880 [2E090000]                           dd     exc13  ; 0Dh, General Protection Fault
24508 00006884 [32090000]                           dd     exc14  ; 0Eh, Page Fault
24509 00006888 [36090000]                           dd     exc15
24510 0000688C [3A090000]                           dd     exc16
24511 00006890 [3E090000]                           dd     exc17
24512 00006894 [42090000]                           dd     exc18
24513 00006898 [46090000]                           dd     exc19
24514 0000689C [4A090000]                           dd     exc20
24515 000068A0 [4E090000]                           dd     exc21
24516 000068A4 [52090000]                           dd     exc22
24517 000068A8 [56090000]                           dd     exc23
24518 000068AC [5A090000]                           dd     exc24
24519 000068B0 [5E090000]                           dd     exc25
```

```
24520 000068B4 [62090000]                     dd    exc26
24521 000068B8 [66090000]                     dd    exc27
24522 000068BC [6A090000]                     dd    exc28
24523 000068C0 [6E090000]                     dd    exc29
24524 000068C4 [72090000]                     dd    exc30
24525 000068C8 [76090000]                     dd    exc31
24526                                         ; Interrupt list
24527 000068CC [27070000]                     dd    timer_int    ; INT 20h
24528                                               ;dd   irq0
24529 000068D0 [300C0000]                     dd    keyb_int     ; 27/08/2014
24530                                               ;dd   irq1
24531 000068D4 [47080000]                     dd    irq2
24532                                               ; COM2 int
24533 000068D8 [4B080000]                     dd    irq3
24534                                               ; COM1 int
24535 000068DC [56080000]                     dd    irq4
24536 000068E0 [61080000]                     dd    irq5
24537                                       ;DISKETTE_INT: ;06/02/2015
24538 000068E4 [6D270000]                           dd    fdc_int            ; 16/02/2015, IRQ 6 handler
24539                                               ;dd   irq6
24540                                       ; Default IRQ 7 handler against spurious IRQs (from master PIC)
24541                                       ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
24542 000068E8 [DD0B0000]                           dd    default_irq7 ; 25/02/2015
24543                                               ;dd   irq7
24544                                       ; Real Time Clock Interrupt
24545 000068EC [800A0000]                           dd    rtc_int            ; 23/02/2015, IRQ 8 handler
24546                                               ;dd   irq8   ; INT 28h
24547 000068F0 [71080000]                     dd    irq9
24548 000068F4 [75080000]                     dd    irq10
24549 000068F8 [79080000]                     dd    irq11
24550 000068FC [7D080000]                     dd    irq12
24551 00006900 [81080000]                     dd    irq13
24552                                         ;HDISK_INT1:  ;06/02/2015
24553 00006904 [A82F0000]                           dd    hdc1_int     ; 21/02/2015, IRQ 14 handler
24554                                               ;dd   irq14
24555                                         ;HDISK_INT2:  ;06/02/2015
24556 00006908 [CF2F0000]                           dd    hdc2_int     ; 21/02/2015, IRQ 15 handler
24557                                               ;dd   irq15  ; INT 2Fh
24558                                               ; 14/08/2015
24559 0000690C [3E3F0000]                           dd    sysent       ; INT 30h (system calls)
24560
24561                                               ;dd   ignore_int
24562 00006910 00000000                             dd    0
24563
24564                                         ;;;
24565                                         ;;; 11/03/2015
24566                                         %include 'kybdata.inc'    ; KEYBOARD (BIOS) DATA
24567                                   <1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC
24568                                   <1> ; Last Modification: 11/03/2015
24569                                   <1> ;            (Data Section for 'KEYBOARD.INC')
24570                                   <1> ;
24571                                   <1> ; ///////// KEYBOARD DATA ///////////////
24572                                   <1>
24573                                   <1> ; 05/12/2014
24574                                   <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
24575                                   <1> ; 03/06/86  KEYBOARD BIOS
24576                                   <1>
24577                                   <1> ;--------------------------------------------------------------------------------
24578                                   <1> ;     KEY IDENTIFICATION SCAN TABLES
24579                                   <1> ;--------------------------------------------------------------------------------
24580                                   <1>
24581                                   <1> ;-----      TABLES FOR ALT CASE ------------
24582                                   <1> ;-----      ALT-INPUT-TABLE
24583 00006914 524F50514B              <1> K30:  db    82,79,80,81,75
24584 00006919 4C4D474849              <1>       db    76,77,71,72,73            ; 10 NUMBER ON KEYPAD
24585                                   <1> ;-----      SUPER-SHIFT-TABLE
24586 0000691E 101112131415            <1>       db    16,17,18,19,20,21   ; A-Z TYPEWRITER CHARS
24587 00006924 161718191E1F            <1>       db    22,23,24,25,30,31
24588 0000692A 202122232425            <1>       db    32,33,34,35,36,37
24589 00006930 262C2D2E2F30            <1>       db    38,44,45,46,47,48
24590 00006936 3132                    <1>       db    49,50
24591                                   <1>
24592                                   <1> ;-----      TABLE OF SHIFT KEYS AND MASK VALUES
24593                                   <1> ;-----      KEY_TABLE
24594 00006938 52                      <1> _K6:    db      INS_KEY                 ; INSERT KEY
24595 00006939 3A4546381D              <1>         db    CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
24596 0000693E 2A36                    <1>           db    LEFT_KEY,RIGHT_KEY
24597                                   <1> _K6L    equ     $-_K6
24598                                   <1>
24599                                   <1> ;-----      MASK_TABLE
24600 00006940 80                      <1> _K7:    db      INS_SHIFT               ; INSERT MODE SHIFT
24601 00006941 4020100804              <1>         db    CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
24602 00006946 0201                    <1>         db    LEFT_SHIFT,RIGHT_SHIFT
24603                                   <1>
24604                                   <1> ;-----      TABLES FOR CTRL CASE            ;---- CHARACTERS ------
24605 00006948 1BFF00FFFFFF            <1> _K8:  db    27,-1,0,-1,-1,-1    ; Esc, 1, 2, 3, 4, 5
24606 0000694E 1EFFFFFFFF1F            <1>       db    30,-1,-1,-1,-1,31   ; 6, 7, 8, 9, 0, -
24607 00006954 FF7FFF111705            <1>       db    -1,127,-1,17,23,5   ; =, Bksp, Tab, Q, W, E
24608 0000695A 12141915090F            <1>       db    18,20,25,21,9,15    ; R, T, Y, U, I, O
24609 00006960 101B1D0AFF01            <1>       db    16,27,29,10,-1,1    ; P, [, ], Enter, Ctrl, A
24610 00006966 13040607080A            <1>       db    19,4,6,7,8,10       ; S, D, F, G, H, J
24611 0000696C 0B0CFFFFFFFF            <1>       db    11,12,-1,-1,-1,-1   ; K, L, :, ', `, LShift
24612 00006972 1C1A18031602            <1>       db    28,26,24,3,22,2           ; Bkslash, Z, X, C, V, B
24613 00006978 0E0DFFFFFFFF            <1>       db    14,13,-1,-1,-1,-1   ; N, M, ,, ., /, RShift
24614 0000697E 96FF20FF                <1>       db    150,-1,' ',-1       ; *, ALT, Spc, CL
24615                                   <1>       ;                         ;----- FUNCTIONS ------
24616 00006982 5E5F60616263            <1>       db    94,95,96,97,98,99   ; F1 - F6
24617 00006988 64656667FFFF            <1>       db    100,101,102,103,-1,-1     ; F7 - F10, NL, SL
24618 0000698E 778D848E738F            <1>       db    119,141,132,142,115,143   ; Home, Up, PgUp, -, Left, Pad5
24619 00006994 749075917692            <1>       db    116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
24620 0000699A 93FFFFF898A             <1>       db    147,-1,-1,-1,137,138      ; Del, SysReq, Undef, WT, F11, F12
24621                                   <1>
24622                                   <1> ;-----      TABLES FOR LOWER CASE ----------
24623 000069A0 1B3132333435363738-     <1> K10:  db    27,'1234567890-=',8,9
24624 000069A9 39302D3D0809            <1>
```

```
24625 000069AF 71776572747975696F- <1>       db    'qwertyuiop[]',13,-1,'asdfghjkl;',39
24626 000069B8 705B5D0DFF61736466- <1>
24627 000069C1 67686A6B6C3B27      <1>
24628 000069C8 60FF5C7A786376626E- <1>       db    96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
24629 000069D1 6D2C2E2FFF2AFF20FF  <1>
24630                              <1> ;-----       LC TABLE SCAN
24631 000069DA 3B3C3D3E3F          <1>       db    59,60,61,62,63            ; BASE STATE OF F1 - F10
24632 000069DF 4041424344          <1>       db    64,65,66,67,68
24633 000069E4 FFFF                <1>       db    -1,-1               ; NL, SL
24634                              <1>
24635                              <1> ;-----       KEYPAD TABLE
24636 000069E6 474849FF4BFF        <1> K15:  db    71,72,73,-1,75,-1   ; BASE STATE OF KEYPAD KEYS
24637 000069EC 4DFF4F50515253      <1>       db    77,-1,79,80,81,82,83
24638 000069F3 FFFF5C8586          <1>       db    -1,-1,92,133,134    ; SysRq, Undef, WT, F11, F12
24639                              <1>
24640                              <1> ;-----       TABLES FOR UPPER CASE ----------
24641 000069F8 1B21402324255E262A- <1> K11:  db    27,'!@#$%',94,'&*()_+',8,0
24642 00006A01 28295F2B0800        <1>
24643 00006A07 5157455255955494F- <1>       db    'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
24644 00006A10 507B7D0DFF41534446- <1>
24645 00006A19 47484A4B4C3A22      <1>
24646 00006A20 7EFF7C5A584356424E- <1>       db    126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
24647 00006A29 4D3C3E3FFF2AFF20FF  <1>
24648                              <1> ;-----       UC TABLE SCAN
24649 00006A32 5455565758          <1> K12:  db    84,85,86,87,88            ; SHIFTED STATE OF F1 - F10
24650 00006A37 595A5B5C5D          <1>       db    89,90,91,92,93
24651 00006A3C FFFF                <1>       db    -1,-1               ; NL, SL
24652                              <1>
24653                              <1> ;-----       NUM STATE TABLE
24654 00006A3E 3738392D3435362B31- <1> K14:  db    '789-456+1230.'           ; NUMLOCK STATE OF KEYPAD KEYS
24655 00006A47 3233302E            <1>
24656                              <1>       ;
24657 00006A4B FFFF7C8788          <1>       db    -1,-1,124,135,136   ; SysRq, Undef, WT, F11, F12
24658                              <1>
24659                              <1> Align 4
24660                              <1> ;----------------------------------------
24661                              <1> ;     VIDEO DISPLAY DATA AREA          ;
24662                              <1> ;----------------------------------------
24663 00006A50 03                  <1> CRT_MODE     db     3      ; CURRENT DISPLAY MODE (TYPE)
24664 00006A51 29                  <1> CRT_MODE_SET db     29h    ; CURRENT SETTING OF THE 3X8 REGISTER
24665                              <1>                            ; (29h default setting for video mode 3)
24666                              <1>                            ; Mode Select register Bits
24667                              <1>                            ;   BIT 0 - 80x25 (1), 40x25 (0)
24668                              <1>                            ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
24669                              <1>                            ;   BIT 2 - COLOR (0), BW (1)
24670                              <1>                            ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
24671                              <1>                            ;   BIT 4 - 640x200 B&W Graphics Mode (1)
24672                              <1>                            ;   BIT 5 - ALPHA mode BLINKING (1)
24673                              <1>                            ;   BIT 6, 7 - Not Used
24674                              <1>
24675                              <1> ; Mode 0 - 2Ch = 101100b  ; 40x25 text, 16 gray colors
24676                              <1> ; Mode 1 - 28h = 101000b  ; 40x25 text, 16 fore colors, 8 back colors
24677                              <1> ; Mode 2 - 2Dh = 101101b  ; 80x25 text, 16 gray colors
24678                              <1> ; MODE 3 - 29h = 101001b  ; 80x25 text, 16 fore color, 8 back color
24679                              <1> ; Mode 4 - 2Ah = 101010b  ; 320x200 graphics, 4 colors
24680                              <1> ; Mode 5 - 2Eh = 101110b  ; 320x200 graphics, 4 gray colors
24681                              <1> ; Mode 6 - 1Eh = 011110b  ; 640x200 graphics, 2 colors
24682                              <1> ; Mode 7 - 29h = 101001b  ; 80x25 text, black & white colors
24683                              <1> ; Mode & 37h = Video signal OFF
24684                              <1>
24685                              <1>
24686                              <1> ; 26/08/2014
24687                              <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
24688                              <1> ; Derived from IBM "pc-at"
24689                              <1> ; rombios source code (06/10/1985)
24690                              <1> ; 'dseg.inc'
24691                              <1>
24692                              <1> ;----------------------------------------;
24693                              <1> ;     SYSTEM DATA AREA          ;
24694                              <1> ;----------------------------------------
24695 00006A52 00                  <1> BIOS_BREAK   db     0           ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
24696                              <1>
24697                              <1> ;----------------------------------------
24698                              <1> ;     KEYBOARD DATA AREAS       ;
24699                              <1> ;----------------------------------------
24700                              <1>
24701 00006A53 00                  <1> KB_FLAG            db     0           ; KEYBOARD SHIFT STATE AND STATUS FLAGS
24702 00006A54 00                  <1> KB_FLAG_1    db    0           ; SECOND BYTE OF KEYBOARD STATUS
24703 00006A55 00                  <1> KB_FLAG_2    db    0           ; KEYBOARD LED FLAGS
24704 00006A56 00                  <1> KB_FLAG_3    db    0           ; KEYBOARD MODE STATE AND TYPE FLAGS
24705 00006A57 00                  <1> ALT_INPUT    db    0           ; STORAGE FOR ALTERNATE KEY PAD ENTRY
24706 00006A58 [686A0000]          <1> BUFFER_START dd    KB_BUFFER    ; OFFSET OF KEYBOARD BUFFER START
24707 00006A5C [886A0000]          <1> BUFFER_END   dd    KB_BUFFER + 32      ; OFFSET OF END OF BUFFER
24708 00006A60 [686A0000]          <1> BUFFER_HEAD  dd    KB_BUFFER    ; POINTER TO HEAD OF KEYBOARD BUFFER
24709 00006A64 [686A0000]          <1> BUFFER_TAIL  dd    KB_BUFFER    ; POINTER TO TAIL OF KEYBOARD BUFFER
24710                              <1> ; ------     HEAD = TAIL  INDICATES THAT THE BUFFER IS EMPTY
24711 00006A68 0000<rept>          <1> KB_BUFFER    times  16 dw 0            ; ROOM FOR 16 SCAN CODE ENTRIES
24712                              <1>
24713                              <1> ; /// End Of KEYBOARD DATA ///
24714                                  %include 'vidata.inc'     ; VIDEO (BIOS) DATA
24715                              <1> ; Retro UNIX 386 v1 Kernel - VIDATA.INC
24716                              <1> ; Last Modification: 11/03/2015
24717                              <1> ;               (Data section for 'VIDEO.INC')
24718                              <1> ;
24719                              <1> ; ///////// VIDEO DATA ///////////////
24720                              <1>
24721                              <1> video_params:
24722                              <1>       ; 02/09/2014 (Retro UNIX 386 v1)
24723                              <1>       ;ORGS.ASM ----- 06/10/85   COMPATIBILITY MODULE
24724                              <1>       ; VIDEO MODE 3
24725 00006A88 71505A0A1F0619      <1>       db    71h,50h,5Ah,0Ah,1Fh,6,19h ; SET UP FOR 80X25
24726 00006A8F 1C02070607          <1>       db    1Ch,2,7,6,7  ; cursor start = 6, cursor stop = 7
24727 00006A94 00000000            <1>       db    0,0,0,0
24728                              <1>
24729                              <1> ; /// End Of VIDEO DATA ///
```

```
24730                                   %include 'diskdata.inc'   ; DISK (BIOS) DATA (initialized)
24731                                 <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC
24732                                 <1> ; Last Modification: 11/03/2015
24733                                 <1> ;     (Initialized Disk Parameters Data section for 'DISKIO.INC')
24734                                 <1> ;
24735                                 <1> ; ///////////////////////////////////////////////////////////////////////////
24736                                 <1>
24737                                 <1> ;---------------------------------------
24738                                 <1> ;     80286 INTERRUPT LOCATIONS :
24739                                 <1> ;     REFERENCED BY POST & BIOS :
24740                                 <1> ;---------------------------------------
24741                                 <1>
24742 00006A98 [FB6A0000]           <1> DISK_POINTER:      dd     MD_TBL6            ; Pointer to Diskette Parameter Table
24743                                 <1>
24744                                 <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
24745                                 <1> ;-------------------------------------------------------------
24746                                 <1> ; DISK_BASE                                    :
24747                                 <1> ;     THIS IS THE SET OF PARAMETERS REQUIRED FOR          :
24748                                 <1> ;     DISKETTE OPERATION. THEY ARE POINTED AT BY THE            :
24749                                 <1> ;     DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,    :
24750                                 <1> ;     BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT       :
24751                                 <1> ;-------------------------------------------------------------
24752                                 <1>
24753                                 <1> ;DISK_BASE:
24754                                 <1> ;     DB     11011111B     ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24755                                 <1> ;     DB     2             ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24756                                 <1> ;     DB     MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24757                                 <1> ;     DB     2             ; 512 BYTES/SECTOR
24758                                 <1> ;     ;DB    15            ; EOT (LAST SECTOR ON TRACK)
24759                                 <1> ;     db     18            ; (EOT for 1.44MB diskette)
24760                                 <1> ;     DB     01BH          ; GAP LENGTH
24761                                 <1> ;     DB     0FFH          ; DTL
24762                                 <1> ;     ;DB    054H          ; GAP LENGTH FOR FORMAT
24763                                 <1> ;     db     06ch          ; (for 1.44MB dsikette)
24764                                 <1> ;     DB     0F6H          ; FILL BYTE FOR FORMAT
24765                                 <1> ;     DB     15            ; HEAD SETTLE TIME (MILLISECONDS)
24766                                 <1> ;     DB     8             ; MOTOR START TIME (1/8 SECONDS)
24767                                 <1>
24768                                 <1> ;---------------------------------------
24769                                 <1> ;     ROM BIOS DATA AREAS       :
24770                                 <1> ;---------------------------------------
24771                                 <1>
24772                                 <1> ;DATA       SEGMENT AT 40H            ; ADDRESS= 0040:0000
24773                                 <1>
24774                                 <1> ;@EQUIP_FLAG DW    ?            ; INSTALLED HARDWARE FLAGS
24775                                 <1>
24776                                 <1> ;---------------------------------------
24777                                 <1> ;     DISKETTE DATA AREAS       :
24778                                 <1> ;---------------------------------------
24779                                 <1>
24780                                 <1> ;@SEEK_STATUS      DB     ?            ; DRIVE RECALIBRATION STATUS
24781                                 <1> ;                                     ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
24782                                 <1> ;                                     ; BEFORE NEXT SEEK IF BIT IS = 0
24783                                 <1> ;@MOTOR_STATUS     DB     ?            ; MOTOR STATUS
24784                                 <1> ;                                     ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
24785                                 <1> ;                                     ; BIT 7 = CURRENT OPERATION IS A WRITE
24786                                 <1> ;@MOTOR_COUNT      DB     ?            ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
24787                                 <1> ;@DSKETTE_STATUS DB       ?            ; RETURN CODE STATUS BYTE
24788                                 <1> ;                                     ; CMD_BLOCK  IN STACK FOR DISK OPERATION
24789                                 <1> ;@NEC_STATUS DB    7 DUP(?)     ; STATUS BYTES FROM DISKETTE OPERATION
24790                                 <1>
24791                                 <1> ;---------------------------------------
24792                                 <1> ;     POST AND BIOS WORK DATA AREA    :
24793                                 <1> ;---------------------------------------
24794                                 <1>
24795                                 <1> ;@INTR_FLAG  DB    ?            ; FLAG INDICATING AN INTERRUPT HAPPENED
24796                                 <1>
24797                                 <1> ;---------------------------------------
24798                                 <1> ;     TIMER DATA AREA           :
24799                                 <1> ;---------------------------------------
24800                                 <1>
24801                                 <1> ; 17/12/2014  (IRQ 0 - INT 08H)
24802                                 <1> ;TIMER_LOW  equ    46Ch          ; Timer ticks (counter)  @ 40h:006Ch
24803                                 <1> ;TIMER_HIGH equ    46Eh          ; (18.2 timer ticks per second)
24804                                 <1> ;TIMER_OFL  equ    470h          ; Timer - 24 hours flag  @ 40h:0070h
24805                                 <1>
24806                                 <1> ;---------------------------------------
24807                                 <1> ;     ADDITIONAL MEDIA DATA           :
24808                                 <1> ;---------------------------------------
24809                                 <1>
24810                                 <1> ;@LASTRATE   DB    ?            ; LAST DISKETTE DATA RATE SELECTED
24811                                 <1> ;@DSK_STATE  DB    ?            ; DRIVE 0 MEDIA STATE
24812                                 <1> ;            DB    ?            ; DRIVE 1 MEDIA STATE
24813                                 <1> ;            DB    ?            ; DRIVE 0 OPERATION START STATE
24814                                 <1> ;            DB    ?            ; DRIVE 1 OPERATION START STATE
24815                                 <1> ;@DSK_TRK    DB    ?            ; DRIVE 0 PRESENT CYLINDER
24816                                 <1> ;            DB    ?            ; DRIVE 1 PRESENT CYLINDER
24817                                 <1>
24818                                 <1> ;DATA       ENDS                ; END OF BIOS DATA SEGMENT
24819                                 <1>
24820                                 <1> ;-------------------------------------------------------
24821                                 <1> ;     DRIVE TYPE TABLE                        :
24822                                 <1> ;-------------------------------------------------------
24823                                 <1>             ; 16/02/2015 (unix386.s, 32 bit modifications)
24824                                 <1> DR_TYPE:
24825 00006A9C 01                   <1>             DB     01           ;DRIVE TYPE, MEDIA TABLE
24826                                 <1>                ;DW      MD_TBL1
24827 00006A9D [BA6A0000]           <1>             dd     MD_TBL1
24828 00006AA1 82                   <1>             DB     02+BIT7ON
24829                                 <1>             ;DW      MD_TBL2
24830 00006AA2 [C76A0000]           <1>                dd      MD_TBL2
24831 00006AA6 02                   <1> DR_DEFAULT:  DB    02
24832                                 <1>                ;DW      MD_TBL3
24833 00006AA7 [D46A0000]           <1>             dd      MD_TBL3
24834 00006AAB 03                   <1>             DB     03
```

```
24835                                  <1>                 ;DW      MD_TBL4
24836 00006AAC [E16A0000]              <1>             dd      MD_TBL4
24837 00006AB0 84                      <1>             DB    04+BIT7ON
24838                                  <1>                 ;DW      MD_TBL5
24839 00006AB1 [EE6A0000]              <1>             dd      MD_TBL5
24840 00006AB5 04                      <1>             DB    04
24841                                  <1>                 ;DW      MD_TBL6
24842 00006AB6 [FB6A0000]              <1>             dd      MD_TBL6
24843                                  <1> DR_TYPE_E       equ $                   ; END OF TABLE
24844                                  <1> ;DR_CNT           EQU   (DR_TYPE_E-DR_TYPE)/3
24845                                  <1> DR_CNT            equ   (DR_TYPE_E-DR_TYPE)/5
24846                                  <1> ;---------------------------------------------------
24847                                  <1> ;     MEDIA/DRIVE PARAMETER TABLES                :
24848                                  <1> ;---------------------------------------------------
24849                                  <1> ;---------------------------------------------------
24850                                  <1> ;     360 KB MEDIA IN 360 KB DRIVE                :
24851                                  <1> ;---------------------------------------------------
24852                                  <1> MD_TBL1:
24853 00006ABA DF                      <1>       DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24854 00006ABB 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24855 00006ABC 25                      <1>       DB    MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24856 00006ABD 02                      <1>       DB    2            ; 512 BYTES/SECTOR
24857 00006ABE 09                      <1>       DB    09           ; EOT (LAST SECTOR ON TRACK)
24858 00006ABF 2A                      <1>       DB    02AH         ; GAP LENGTH
24859 00006AC0 FF                      <1>       DB    0FFH         ; DTL
24860 00006AC1 50                      <1>       DB    050H         ; GAP LENGTH FOR FORMAT
24861 00006AC2 F6                      <1>       DB    0F6H         ; FILL BYTE FOR FORMAT
24862 00006AC3 0F                      <1>       DB    15           ; HEAD SETTLE TIME (MILLISECONDS)
24863 00006AC4 08                      <1>       DB    8            ; MOTOR START TIME (1/8 SECONDS)
24864 00006AC5 27                      <1>       DB    39           ; MAX. TRACK NUMBER
24865 00006AC6 80                      <1>       DB    RATE_250     ; DATA TRANSFER RATE
24866                                  <1> ;---------------------------------------------------
24867                                  <1> ;     360 KB MEDIA IN 1.2 MB DRIVE                :
24868                                  <1> ;---------------------------------------------------
24869                                  <1> MD_TBL2:
24870 00006AC7 DF                      <1>       DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24871 00006AC8 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24872 00006AC9 25                      <1>       DB    MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24873 00006ACA 02                      <1>       DB    2            ; 512 BYTES/SECTOR
24874 00006ACB 09                      <1>       DB    09           ; EOT (LAST SECTOR ON TRACK)
24875 00006ACC 2A                      <1>       DB    02AH         ; GAP LENGTH
24876 00006ACD FF                      <1>       DB    0FFH         ; DTL
24877 00006ACE 50                      <1>       DB    050H         ; GAP LENGTH FOR FORMAT
24878 00006ACF F6                      <1>       DB    0F6H         ; FILL BYTE FOR FORMAT
24879 00006AD0 0F                      <1>       DB    15           ; HEAD SETTLE TIME (MILLISECONDS)
24880 00006AD1 08                      <1>       DB    8            ; MOTOR START TIME (1/8 SECONDS)
24881 00006AD2 27                      <1>       DB    39           ; MAX. TRACK NUMBER
24882 00006AD3 40                      <1>       DB    RATE_300     ; DATA TRANSFER RATE
24883                                  <1> ;---------------------------------------------------
24884                                  <1> ;     1.2 MB MEDIA IN 1.2 MB DRIVE                :
24885                                  <1> ;---------------------------------------------------
24886                                  <1> MD_TBL3:
24887 00006AD4 DF                      <1>       DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24888 00006AD5 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24889 00006AD6 25                      <1>       DB    MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24890 00006AD7 02                      <1>       DB    2            ; 512 BYTES/SECTOR
24891 00006AD8 0F                      <1>       DB    15           ; EOT (LAST SECTOR ON TRACK)
24892 00006AD9 1B                      <1>       DB    01BH         ; GAP LENGTH
24893 00006ADA FF                      <1>       DB    0FFH         ; DTL
24894 00006ADB 54                      <1>       DB    054H         ; GAP LENGTH FOR FORMAT
24895 00006ADC F6                      <1>       DB    0F6H         ; FILL BYTE FOR FORMAT
24896 00006ADD 0F                      <1>       DB    15           ; HEAD SETTLE TIME (MILLISECONDS)
24897 00006ADE 08                      <1>       DB    8            ; MOTOR START TIME (1/8 SECONDS)
24898 00006ADF 4F                      <1>       DB    79           ; MAX. TRACK NUMBER
24899 00006AE0 00                      <1>       DB    RATE_500     ; DATA TRANSFER RATE
24900                                  <1> ;---------------------------------------------------
24901                                  <1> ;     720 KB MEDIA IN 720 KB DRIVE                :
24902                                  <1> ;---------------------------------------------------
24903                                  <1> MD_TBL4:
24904 00006AE1 DF                      <1>       DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24905 00006AE2 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24906 00006AE3 25                      <1>       DB    MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24907 00006AE4 02                      <1>       DB    2            ; 512 BYTES/SECTOR
24908 00006AE5 09                      <1>       DB    09           ; EOT (LAST SECTOR ON TRACK)
24909 00006AE6 2A                      <1>       DB    02AH         ; GAP LENGTH
24910 00006AE7 FF                      <1>       DB    0FFH         ; DTL
24911 00006AE8 50                      <1>       DB    050H         ; GAP LENGTH FOR FORMAT
24912 00006AE9 F6                      <1>       DB    0F6H         ; FILL BYTE FOR FORMAT
24913 00006AEA 0F                      <1>       DB    15           ; HEAD SETTLE TIME (MILLISECONDS)
24914 00006AEB 08                      <1>       DB    8            ; MOTOR START TIME (1/8 SECONDS)
24915 00006AEC 4F                      <1>       DB    79           ; MAX. TRACK NUMBER
24916 00006AED 80                      <1>       DB    RATE_250     ; DATA TRANSFER RATE
24917                                  <1> ;---------------------------------------------------
24918                                  <1> ;     720 KB MEDIA IN 1.44 MB DRIVE               :
24919                                  <1> ;---------------------------------------------------
24920                                  <1> MD_TBL5:
24921 00006AEE DF                      <1>       DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24922 00006AEF 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24923 00006AF0 25                      <1>       DB    MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24924 00006AF1 02                      <1>       DB    2            ; 512 BYTES/SECTOR
24925 00006AF2 09                      <1>       DB    09           ; EOT (LAST SECTOR ON TRACK)
24926 00006AF3 2A                      <1>       DB    02AH         ; GAP LENGTH
24927 00006AF4 FF                      <1>       DB    0FFH         ; DTL
24928 00006AF5 50                      <1>       DB    050H         ; GAP LENGTH FOR FORMAT
24929 00006AF6 F6                      <1>       DB    0F6H         ; FILL BYTE FOR FORMAT
24930 00006AF7 0F                      <1>       DB    15           ; HEAD SETTLE TIME (MILLISECONDS)
24931 00006AF8 08                      <1>       DB    8            ; MOTOR START TIME (1/8 SECONDS)
24932 00006AF9 4F                      <1>       DB    79           ; MAX. TRACK NUMBER
24933 00006AFA 80                      <1>       DB    RATE_250     ; DATA TRANSFER RATE
24934                                  <1> ;---------------------------------------------------
24935                                  <1> ;     1.44 MB MEDIA IN 1.44 MB DRIVE              :
24936                                  <1> ;---------------------------------------------------
24937                                  <1> MD_TBL6:
24938 00006AFB AF                      <1>       DB    10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
24939 00006AFC 02                      <1>       DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
```

```
24940 00006AFD 25                <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24941 00006AFE 02                <1>       DB     2            ; 512 BYTES/SECTOR
24942 00006AFF 12                <1>       DB     18           ; EOT (LAST SECTOR ON TRACK)
24943 00006B00 1B                <1>       DB     01BH         ; GAP LENGTH
24944 00006B01 FF                <1>       DB     0FFH         ; DTL
24945 00006B02 6C                <1>       DB     06CH         ; GAP LENGTH FOR FORMAT
24946 00006B03 F6                <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
24947 00006B04 0F                <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
24948 00006B05 08                <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
24949 00006B06 4F                <1>       DB     79           ; MAX. TRACK NUMBER
24950 00006B07 00                <1>       DB     RATE_500     ; DATA TRANSFER RATE
24951                            <1>
24952                            <1>
24953                            <1> ; << diskette.inc >>
24954                            <1> ; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
24955                            <1> ;
24956                            <1> ;----------------------------------------
24957                            <1> ;     ROM BIOS DATA AREAS       :
24958                            <1> ;----------------------------------------
24959                            <1>
24960                            <1> ;DATA       SEGMENT AT 40H            ; ADDRESS= 0040:0000
24961                            <1>
24962                            <1> ;----------------------------------------
24963                            <1> ;     FIXED DISK DATA AREAS             :
24964                            <1> ;----------------------------------------
24965                            <1>
24966                            <1> ;DISK_STATUS1:     DB     0            ; FIXED DISK STATUS
24967                            <1> ;HF_NUM:           DB     0            ; COUNT OF FIXED DISK DRIVES
24968                            <1> ;CONTROL_BYTE:     DB     0            ; HEAD CONTROL BYTE
24969                            <1> ;@PORT_OFF   DB    ?            ;  RESERVED (PORT OFFSET)
24970                            <1>
24971                            <1> ;----------------------------------------
24972                            <1> ;     ADDITIONAL MEDIA DATA             :
24973                            <1> ;----------------------------------------
24974                            <1>
24975                            <1> ;@LASTRATE   DB     ?            ; LAST DISKETTE DATA RATE SELECTED
24976                            <1> ;HF_STATUS   DB     0            ; STATUS REGISTER
24977                            <1> ;HF_ERROR    DB     0            ; ERROR REGISTER
24978                            <1> ;HF_INT_FLAG DB     0            ; FIXED DISK INTERRUPT FLAG
24979                            <1> ;HF_CNTRL    DB     0            ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
24980                            <1> ;@DSK_STATE  DB     ?            ; DRIVE 0 MEDIA STATE
24981                            <1> ;            DB     ?            ; DRIVE 1 MEDIA STATE
24982                            <1> ;            DB     ?            ; DRIVE 0 OPERATION START STATE
24983                            <1> ;            DB     ?            ; DRIVE 1 OPERATION START STATE
24984                            <1> ;@DSK_TRK    DB     ?            ; DRIVE 0 PRESENT CYLINDER
24985                            <1> ;            DB     ?            ; DRIVE 1 PRESENT CYLINDER
24986                            <1>
24987                            <1> ;DATA       ENDS                 ; END OF BIOS DATA SEGMENT
24988                            <1> ;
24989                            <1> ; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
24990                            <1>
24991                            <1> ERR_TBL:
24992 00006B08 E0                <1>       db     NO_ERR
24993 00006B09 024001BB          <1>       db     BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
24994 00006B0D 04BB100A          <1>       db     RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
24995                            <1>
24996                            <1> ; 17/12/2014 (mov ax, [cfd])
24997                            <1> ; 11/12/2014
24998 00006B11 00                <1> cfd:        db 0                 ; current floppy drive (for GET_PARM)
24999                            <1> ; 17/12/2014                     ; instead of 'DISK_POINTER'
25000 00006B12 01                <1> pfd:        db 1                 ; previous floppy drive (for GET_PARM)
25001                            <1>                                  ; (initial value of 'pfd
25002                            <1>                                  ; must be different then 'cfd' value
25003                            <1>                                  ; to force updating/initializing
25004                            <1>                                  ; current drive parameters)
25005 00006B13 90                <1> align 2
25006                            <1>
25007 00006B14 F001              <1> HF_PORT:    dw     1F0h  ; Default = 1F0h
25008                            <1>                           ; (170h)
25009 00006B16 F603              <1> HF_REG_PORT: dw    3F6h  ; HF_PORT + 206h
25010                            <1>
25011                            <1> ; 05/01/2015
25012 00006B18 00                <1> hf_m_s:         db      0     ; (0 = Master, 1 = Slave)
25013                            <1>
25014                            <1> ; ////////////////////////////////////////////////////////////////////////////
25015                                ;;;
25016
25017 00006B19 90                    Align 2
25018
25019                                ; 12/11/2014 (Retro UNIX 386 v1)
25020 00006B1A 00                    boot_drv:    db 0 ; boot drive number (physical)
25021                                ; 24/11/2014
25022 00006B1B 00                    drv:       db 0
25023 00006B1C 00                    last_drv:    db 0 ; last hdd
25024 00006B1D 00                    hdc:         db 0  ; number of hard disk drives
25025                                             ; (present/detected)
25026                                ;
25027                                ; 24/11/2014 (Retro UNIX 386 v1)
25028                                ; Physical drive type & flags
25029 00006B1E 00                    fd0_type:    db 0  ; floppy drive type
25030 00006B1F 00                    fd1_type:    db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
25031                                             ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
25032                                             ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
25033                                             ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
25034                                             ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
25035 00006B20 00                    hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
25036 00006B21 00                    hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
25037 00006B22 00                    hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
25038 00006B23 00                    hd3_type:    db 0  ; EDD status for hd3 (bit 7 = present flag)
25039                                             ; bit 0 - Fixed disk access subset supported
25040                                             ; bit 1 - Drive locking and ejecting
25041                                             ; bit 2 - Enhanced disk drive support
25042                                             ; bit 3 = Reserved (64 bit EDD support)
25043                                             ; (If bit 0 is '1' Retro UNIX 386 v1
25044                                             ; will interpret it as 'LBA ready'!)
```

```
25045
25046                                   ; 11/03/2015 - 10/07/2015
25047 00006B24 000000000000000000-      drv.cylinders: dw 0,0,0,0,0,0,0
25048 00006B2D 0000000000
25049 00006B32 000000000000000000-      drv.heads:     dw 0,0,0,0,0,0,0
25050 00006B3B 0000000000
25051 00006B40 000000000000000000-      drv.spt:       dw 0,0,0,0,0,0,0
25052 00006B49 0000000000
25053 00006B4E 000000000000000000-      drv.size:      dd 0,0,0,0,0,0,0
25054 00006B57 000000000000000000-
25055 00006B60 000000000000000000-
25056 00006B69 00
25057 00006B6A 00000000000000           drv.status:    db 0,0,0,0,0,0,0
25058 00006B71 00000000000000           drv.error:     db 0,0,0,0,0,0,0
25059                                   ;
25060
25061                                   ; 27/08/2014
25062                                   scr_row:
25063 00006B78 E0810B00                       dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
25064                                   scr_col:
25065 00006B7C 00000000                       dd 0
25066
25067                                   ;; 14/08/2015
25068                                   ;;msgPM:
25069                                   ;;      db "Protected mode and paging are ENABLED ... ", 0
25070                                   msgKVER:
25071 00006B80 526574726F20554E49-            db "Retro UNIX 386 v1.1 - Kernel v0.2.1.0 [04/02/2016]", 0
25072 00006B89 58203338362076312E-
25073 00006B92 31202D204B65726E65-
25074 00006B9B 6C2076302E322E312E-
25075 00006BA4 30205B30342F30322F-
25076 00006BAD 323031365D00
25077
25078 00006BB3 90                       Align 2
25079
25080                                   ; 20/08/2014
25081                                     ; /* This is the default interrupt "handler" :-) */
25082                                     ; Linux v0.12 (head.s)
25083                                   int_msg:
25084 00006BB4 556E6B6E6F776E2069-            db "Unknown interrupt ! ", 0
25085 00006BBD 6E7465727275707420-
25086 00006BC6 212000
25087
25088 00006BC9 90                       Align 2
25089
25090                                   ; 21/08/2014
25091                                   timer_msg:
25092 00006BCA 49525120302028494E-            db "IRQ 0 (INT 20h) ! Timer Interrupt : "
25093 00006BD3 542032306829202120-
25094 00006BDC 54696D657220496E74-
25095 00006BE5 657272757074203A20
25096                                   tcountstr:
25097 00006BEE 303030303020                   db "00000 "
25098 00006BF4 00                             db 0
25099
25100 00006BF5 90                       Align 2
25101                                         ; 21/08/2014
25102                                   exc_msg:
25103 00006BF6 435055206578636570-            db "CPU exception ! "
25104 00006BFF 74696F6E202120
25105                                   excnstr:          ; 25/08/2014
25106 00006C06 3F3F68202045495020-            db "??h", "  EIP : "
25107 00006C0F 3A20
25108                                   EIPstr: ; 29/08/2014
25109 00006C11 00<rept>                       times 12 db 0
25110                                   rtc_msg:
25111 00006C1D 5265616C2054696D65-            db "Real Time Clock - "
25112 00006C26 20436C6F636B202D20
25113                                   datestr:
25114 00006C2F 30302F30302F303030-            db "00/00/0000"
25115 00006C38 30
25116 00006C39 20                             db " "
25117                                   daystr:
25118 00006C3A 44415920                       db "DAY "
25119                                   timestr:
25120 00006C3E 30303A30303A3030               db "00:00:00"
25121 00006C46 20                             db " "
25122 00006C47 00                             db 0
25123
25124                                   daytmp:
25125                                         ; 28/02/2015
25126 00006C48 3F3F3F2053554E204D-            db "??? SUN MON TUE WED THU FRI SAT "
25127 00006C51 4F4E20545545205745-
25128 00006C5A 442054485520465249-
25129 00006C63 2053415420
25130
25131 00006C68 FF                       ptime_seconds: db 0FFh
25132
25133                                         ; 23/02/2015
25134                                         ; 25/08/2014
25135                                   ;scounter:
25136                                   ;     db 5
25137                                   ;     db 19
25138
25139                                   ; 05/11/2014
25140                                   msg_out_of_memory:
25141 00006C69 070D0A                         db    07h, 0Dh, 0Ah
25142 00006C6C 496E73756666696369-              db      'Insufficient memory ! (Minimum 2 MB memory is needed.)'
25143 00006C75 656E74206D656D6F72-
25144 00006C7E 79202120284D696E69-
25145 00006C87 6D756D2032204D4220-
25146 00006C90 6D656D6F7279206973-
25147 00006C99 206E65656465642E29
25148 00006CA2 0D0A00                         db    0Dh, 0Ah, 0
25149                                         ;
```

```
25150                                   setup_error_msg:
25151 00006CA5 0D0A                           db 0Dh, 0Ah
25152 00006CA7 4469736B2053657475-            db 'Disk Setup Error!'
25153 00006CB0 70204572726F7221
25154 00006CB8 0D0A00                         db 0Dh, 0Ah,0
25155
25156                                   ; 02/09/2014 (Retro UNIX 386 v1)
25157                                   ;crt_ulc : db 0 ; upper left column (for scroll)
25158                                   ;       db 0 ; upper left row (for scroll)
25159
25160                                   ;crt_lrc : db 79 ; lower right column (for scroll)
25161                                   ;       db 24 ; lower right row (for scroll)
25162
25163
25164                                   ; 06/11/2014 (Temporary Data)
25165                                   ; Memory Information message
25166                                   ; 14/08/2015
25167                                   msg_memory_info:
25168 00006CBB 07                             db    07h
25169 00006CBC 0D0A                           db    0Dh, 0Ah
25170                                         ;db   "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
25171 00006CBE 546F74616C206D656D-            db    "Total memory : "
25172 00006CC7 6F7279203A20
25173                                   mem_total_b_str: ; 10 digits
25174 00006CCD 303030303030303030-            db    "0000000000 bytes", 0Dh, 0Ah
25175 00006CD6 302062797465730D0A
25176 00006CDF 2020202020202020-              db    "               ", 20h, 20h, 20h
25177 00006CE8 2020202020202020
25178                                   mem_total_p_str: ; 7 digits
25179 00006CF1 3030303030303070-              db    "0000000 pages", 0Dh, 0Ah
25180 00006CFA 616765730D0A
25181 00006D00 0D0A                           db    0Dh, 0Ah
25182 00006D02 46726565206D656D6F-            db    "Free memory  : "
25183 00006D0B 727920203A20
25184                                   free_mem_b_str:  ; 10 digits
25185 00006D11 3F3F3F3F3F3F3F3F3F-            db    "?????????? bytes", 0Dh, 0Ah
25186 00006D1A 3F2062797465730D0A
25187 00006D23 2020202020202020-              db    "               ", 20h, 20h, 20h
25188 00006D2C 2020202020202020
25189                                   free_mem_p_str:  ; 7 digits
25190 00006D35 3F3F3F3F3F3F3F2070-            db    "??????? pages", 0Dh, 0Ah
25191 00006D3E 616765730D0A
25192 00006D44 0D0A00                         db    0Dh, 0Ah, 0
25193
25194                                   dsk_ready_msg:
25195 00006D47 0D0A                           db    0Dh, 0Ah
25196                                   dsktype:
25197 00006D49 6664                           db    'fd'
25198                                   dskx:
25199 00006D4B 30                             db    '0'
25200 00006D4C 20                             db    20h
25201 00006D4D 697320524541445920-            db    'is READY ...'
25202 00006D56 2E2E2E
25203 00006D59 00                             db    0
25204                                   nextline:
25205 00006D5A 0D0A00                         db    0Dh, 0Ah, 0
25206
25207                                   ; KERNEL - SYSINIT Messages
25208                                   ; 24/08/2015
25209                                   ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
25210                                   ; 14/07/2013
25211                                   ;kernel_init_err_msg:
25212                                   ;     db 0Dh, 0Ah
25213                                   ;     db 07h
25214                                   ;     db 'Kernel initialization ERROR !'
25215                                   ;     db 0Dh, 0Ah, 0
25216                                   ; 24/08/2015
25217                                   ;;; (temporary kernel init message has been removed
25218                                   ;;;  from 'sys_init' code)
25219                                   ;kernel_init_ok_msg:
25220                                   ;     db 0Dh, 0Ah
25221                                   ;     db 07h
25222                                   ;     db 'Welcome to Retro UNIX 386 v1.1 Operating System !'
25223                                   ;     db 0Dh, 0Ah
25224                                   ;       db 'by Erdogan Tan - 04/02/2016 (v0.2.1.0)'
25225                                   ;     db 0Dh, 0Ah, 0
25226                                   panic_msg:
25227 00006D5D 0D0A07                         db 0Dh, 0Ah, 07h
25228 00006D60 4552524F523A204B65-            db 'ERROR: Kernel Panic !'
25229 00006D69 726E656C2050616E69-
25230 00006D72 632021
25231 00006D75 0D0A00                         db 0Dh, 0Ah, 0
25232                                   etc_init_err_msg:
25233 00006D78 0D0A                           db 0Dh, 0Ah
25234 00006D7A 07                             db 07h
25235 00006D7B 4552524F523A202F65-            db 'ERROR: /etc/init !?'
25236 00006D84 74632F696E69742021-
25237 00006D8D 3F
25238 00006D8E 0D0A00                         db 0Dh, 0Ah, 0
25239
25240                                   ; 10/05/2015
25241                                   badsys_msg:
25242 00006D91 0D0A                           db 0Dh, 0Ah
25243 00006D93 07                             db 07h
25244 00006D94 496E76616C69642053-            db 'Invalid System Call !'
25245 00006D9D 797374656D2043616C-
25246 00006DA6 6C2021
25247 00006DA9 0D0A                           db 0Dh, 0Ah
25248 00006DAB 4541583A20                     db 'EAX: '
25249                                   bsys_msg_eax:
25250 00006DB0 303030303030303068             db '00000000h'
25251 00006DB9 0D0A                           db 0Dh, 0Ah
25252 00006DBB 4549503A20                     db 'EIP: '
25253                                   bsys_msg_eip:
25254 00006DC0 303030303030303068             db '00000000h'
```

```
25255 00006DC9 0D0A00                      db 0Dh, 0Ah, 0
25256
25257                                  BSYS_M_SIZE equ $ - badsys_msg
25258
25259
25260                                  align 2
25261
25262                                  ; EPOCH Variables
25263                                  ; 13/04/2015 - Retro UNIX 386 v1 Beginning
25264                                  ; 09/04/2013 epoch variables
25265                                  ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
25266                                  ;
25267 00006DCC B207                   year:       dw 1970
25268 00006DCE 0100                   month:      dw 1
25269 00006DD0 0100                   day:  dw 1
25270 00006DD2 0000                   hour:       dw 0
25271 00006DD4 0000                   minute: dw 0
25272 00006DD6 0000                   second: dw 0
25273
25274                                  DMonth:
25275 00006DD8 0000                         dw 0
25276 00006DDA 1F00                         dw 31
25277 00006DDC 3B00                         dw 59
25278 00006DDE 5A00                         dw 90
25279 00006DE0 7800                         dw 120
25280 00006DE2 9700                         dw 151
25281 00006DE4 B500                         dw 181
25282 00006DE6 D400                         dw 212
25283 00006DE8 F300                         dw 243
25284 00006DEA 1101                         dw 273
25285 00006DEC 3001                         dw 304
25286 00006DEE 4E01                         dw 334
25287
25288                                  ; 04/11/2014 (Retro UNIX 386 v1)
25289 00006DF0 0000                   mem_1m_1k:   dw 0  ; Number of contiguous KB between
25290                                                    ; 1 and 16 MB, max. 3C00h = 15 MB.
25291 00006DF2 0000                   mem_16m_64k: dw 0  ; Number of contiguous 64 KB blocks
25292                                                ;   between 16 MB and 4 GB.
25293 00006DF4 90<rept>                align 16
25294
25295                                  bss_start:
25296
25297                                  ABSOLUTE bss_start
25298
25299                                        ; 11/03/2015
25300                                        ; Interrupt Descriptor Table (20/08/2014)
25301                                  idt:
25302 00006E00 <res 00000200>               resb   64*8 ; INT 0 to INT 3Fh
25303                                  idt_end:
25304
25305                                  ;alignb 4
25306
25307                                  task_state_segment:
25308                                        ; 24/03/2015
25309 00007000 <res 00000002>         tss.link:   resw 1
25310 00007002 <res 00000002>                     resw 1
25311                                  ; tss offset 4
25312 00007004 <res 00000004>         tss.esp0:   resd 1
25313 00007008 <res 00000002>         tss.ss0:    resw 1
25314 0000700A <res 00000002>                     resw 1
25315 0000700C <res 00000004>         tss.esp1:   resd 1
25316 00007010 <res 00000002>         tss.ss1:    resw 1
25317 00007012 <res 00000002>                     resw 1
25318 00007014 <res 00000004>         tss.esp2:   resd 1
25319 00007018 <res 00000002>         tss.ss2:    resw 1
25320 0000701A <res 00000002>                     resw 1
25321                                  ; tss offset 28
25322 0000701C <res 00000004>         tss.CR3:    resd 1
25323 00007020 <res 00000004>         tss.eip:    resd 1
25324 00007024 <res 00000004>         tss.eflags: resd 1
25325                                  ; tss offset 40
25326 00007028 <res 00000004>         tss.eax:    resd 1
25327 0000702C <res 00000004>         tss.ecx:    resd 1
25328 00007030 <res 00000004>         tss.edx:    resd 1
25329 00007034 <res 00000004>         tss.ebx:    resd 1
25330 00007038 <res 00000004>         tss.esp:    resd 1
25331 0000703C <res 00000004>         tss.ebp:    resd 1
25332 00007040 <res 00000004>         tss.esi:    resd 1
25333 00007044 <res 00000004>         tss.edi:    resd 1
25334                                  ; tss offset 72
25335 00007048 <res 00000002>         tss.ES:     resw 1
25336 0000704A <res 00000002>                     resw 1
25337 0000704C <res 00000002>         tss.CS:           resw 1
25338 0000704E <res 00000002>                     resw 1
25339 00007050 <res 00000002>         tss.SS:           resw 1
25340 00007052 <res 00000002>                     resw 1
25341 00007054 <res 00000002>         tss.DS:           resw 1
25342 00007056 <res 00000002>                     resw 1
25343 00007058 <res 00000002>         tss.FS:           resw 1
25344 0000705A <res 00000002>                     resw 1
25345 0000705C <res 00000002>         tss.GS:           resw 1
25346 0000705E <res 00000002>                     resw 1
25347 00007060 <res 00000002>         tss.LDTR:   resw 1
25348 00007062 <res 00000002>                     resw 1
25349                                  ; tss offset 100
25350 00007064 <res 00000002>                     resw 1
25351 00007066 <res 00000002>         tss.IOPB:   resw 1
25352                                  ; tss offset 104
25353                                  tss_end:
25354
25355 00007068 <res 00000004>         k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
25356                                                   ;   (Physical address = Virtual address)
25357 0000706C <res 00000004>         memory_size: resd 1 ; memory size in pages
25358 00007070 <res 00000004>         free_pages:  resd 1 ; number of free pages
25359 00007074 <res 00000004>         next_page:   resd 1 ; offset value in M.A.T. for
```

```
25360                                           ;   first free page search
25361 00007078 <res 00000004>           last_page:   resd 1 ; offset value in M.A.T. which
25362                                           ;   next free page search will be
25363                                           ; stopped after it. (end of M.A.T.)
25364 0000707C <res 00000004>           first_page:  resd 1 ;   offset value in M.A.T. which
25365                                           ; first free page search
25366                                           ;   will be started on it. (for user)
25367 00007080 <res 00000004>           mat_size:    resd 1 ; Memory Allocation Table size in pages
25368
25369                                   ;;;
25370                                   ; 02/09/2014 (Retro UNIX 386 v1)
25371                                   ; 04/12/2013 (Retro UNIX 8086 v1)
25372 00007084 <res 00000002>           CRT_START:   resw 1         ; starting address in regen buffer
25373                                                        ; NOTE: active page only
25374 00007086 <res 00000010>           cursor_posn: resw 8         ; cursor positions for video pages
25375                                   active_page:
25376 00007096 <res 00000001>           ptty:            resb 1     ; current tty
25377                                   ; 01/07/2015
25378 00007097 <res 00000001>           ccolor:          resb 1     ; current color attributes ('sysmsg')
25379                                   ; 26/10/2015
25380                                   ; 07/09/2014
25381 00007098 <res 00000014>           ttychr:      resw ntty+2  ; Character buffer (multiscreen)
25382
25383                                   ; 21/08/2014
25384 000070AC <res 00000004>           tcount:          resd 1
25385
25386                                   ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
25387 000070B0 <res 00000004>           p_time:      resd 1     ; present time (for systime & sysmdate)
25388
25389                                   ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
25390                                   ; (open mode locks for pseudo TTYs)
25391                                   ; [ major tty locks (return error in any conflicts) ]
25392 000070B4 <res 00000014>           ttyl:        resw ntty+2 ; opening locks for TTYs.
25393
25394                                   ; 15/04/2015 (Retro UNIX 386 v1)
25395                                   ; 22/09/2013 (Retro UNIX 8086 v1)
25396 000070C8 <res 0000000A>           wlist:       resb ntty+2 ; wait channel list (0 to 9 for TTYs)
25397                                   ; 15/04/2015 (Retro UNIX 386 v1)
25398                                   ;; 12/07/2014 -> sp_init set comm. parameters as 0E3h
25399                                   ;; 0 means serial port is not available
25400                                   ;;comprm: ; 25/06/2014
25401 000070D2 <res 00000001>           com1p:       resb 1  ;;0E3h
25402 000070D3 <res 00000001>           com2p:       resb 1  ;;0E3h
25403
25404                                   ; 17/11/2015
25405                                   ; request for response (from the terminal)
25406 000070D4 <res 00000002>           req_resp:     resw 1
25407                                   ; 07/11/2015
25408 000070D6 <res 00000001>           ccomport:    resb 1 ; current COM (serial) port
25409                                                   ; (0= COM1, 1= COM2)
25410                                   ; 09/11/2015
25411 000070D7 <res 00000001>           comqr:           resb 1 ; 'query or response' sign (u9.s, 'sndc')
25412                                   ; 07/11/2015
25413 000070D8 <res 00000002>           rchar:           resw 1 ; last received char for COM 1 and COM 2
25414 000070DA <res 00000002>           schar:           resw 1 ; last sent char for COM 1 and COM 2
25415
25416                                   ; 23/10/2015
25417                                   ; SERIAL PORTS - COMMUNICATION MODES
25418                                   ; (Retro UNIX 386 v1 feature only!)
25419                                   ; 0 - command mode (default/initial mode)
25420                                   ; 1 - terminal mode (Retro UNIX 386 v1 terminal, ascii chars)
25421                                   ;;; communication modes for futre versions:
25422                                   ; // 2 - keyboard mode (ascii+scancode input)
25423                                   ; // 3 - mouse mode
25424                                   ; // 4 - device control (output) mode
25425                                   ; VALID COMMANDS for current version:
25426                                   ;     'LOGIN'
25427                                   ;  Login request: db 0FFh, 'LOGIN', 0
25428                                   ;      ("Retro UNIX 386 v1 terminal requests login")
25429                                   ;  Login response: db 0FFh, 'login', 0
25430                                   ;      ("login request accepted, wait for login prompt")
25431                                   ; When a login requests is received and acknowledged (by
25432                                   ; serial port interrupt handler (communication procedure),
25433                                   ; Retro UNIX 386 v1 operating system will start terminal mode
25434                                   ; (login procedure) by changing comm. mode to 1 (terminal mode)
25435                                   ; and then running 'etc/getty' for tty8 (COM1) or tty9 (COM2)
25436                                   ;
25437                                   ; 'sys connect' system call is used to change communication mode
25438                                   ; except 'LOGIN' command which is used to start terminal mode
25439                                   ; by using (COM port) terminal.
25440
25441                                   ;com1own:     resb 1 ; COM1 owner (u.uno)
25442                                   ;com2own:     resb 1 ; COM2 owner (u.uno)
25443                                   ;com1mode:    resb 1 ; communication mode for COM1
25444                                   ;com1com:     resb 1 ; communication command for COM1
25445                                   ;com2mode:    resb 1 ; communication mode for COM1
25446                                   ;com2com      resb 1 ; communication command for COM1
25447                                   ;com1cbufp:   resb 8 ; COM1 command buffer char pointer
25448                                   ;com2cbufp:   resb 8 ; COM2 command buffer char pointer
25449                                   ;com1cbuf:    resb 8 ; COM2 command buffer
25450                                   ;com2cbuf:    resb 8 ; COM2 command buffer
25451
25452                                   ; 22/08/2014 (RTC)
25453                                   ; (Packed BCD)
25454 000070DC <res 00000001>           time_seconds: resb 1
25455 000070DD <res 00000001>           time_minutes: resb 1
25456 000070DE <res 00000001>           time_hours:   resb 1
25457 000070DF <res 00000001>           date_wday:    resb 1
25458 000070E0 <res 00000001>           date_day:     resb 1
25459 000070E1 <res 00000001>           date_month:   resb 1
25460 000070E2 <res 00000001>           date_year:    resb 1
25461 000070E3 <res 00000001>           date_century: resb 1
25462
25463                                   %include 'diskbss.inc'    ; UNINITIALIZED DISK (BIOS) DATA
25464                                 <1> ; Retro UNIX 386 v1 Kernel - DISKBSS.INC
```

```
25465                           <1> ; Last Modification: 10/07/2015
25466                           <1> ;     (Unnitialized Disk Parameters Data section for 'DISKIO.INC')
25467                           <1> ;
25468                           <1> ; ///////////////////////////////////////////////////////////////////////////
25469                           <1>
25470                           <1> alignb 2
25471                           <1>
25472                           <1> ;----------------------------------------
25473                           <1> ;     TIMER DATA AREA           :
25474                           <1> ;----------------------------------------
25475                           <1>
25476                           <1> TIMER_LH:    ; 16/02/205
25477 000070E4 <res 00000002>     <1> TIMER_LOW:      resw     1               ; LOW WORD OF TIMER COUNT
25478 000070E6 <res 00000002>     <1> TIMER_HIGH:     resw     1               ; HIGH WORD OF TIMER COUNT
25479 000070E8 <res 00000001>     <1> TIMER_OFL:      resb     1               ; TIMER HAS ROLLED OVER SINCE LAST READ
25480                           <1>
25481                           <1> ;----------------------------------------
25482                           <1> ;     DISKETTE DATA AREAS       :
25483                           <1> ;----------------------------------------
25484                           <1>
25485 000070E9 <res 00000001>     <1> SEEK_STATUS: resb   1
25486 000070EA <res 00000001>     <1> MOTOR_STATUS:      resb   1
25487 000070EB <res 00000001>     <1> MOTOR_COUNT: resb   1
25488 000070EC <res 00000001>     <1> DSKETTE_STATUS:    resb   1
25489 000070ED <res 00000007>     <1> NEC_STATUS:  resb   7
25490                           <1>
25491                           <1> ;----------------------------------------
25492                           <1> ;     ADDITIONAL MEDIA DATA           :
25493                           <1> ;----------------------------------------
25494                           <1>
25495 000070F4 <res 00000001>     <1> LASTRATE:    resb   1
25496 000070F5 <res 00000001>     <1> HF_STATUS:   resb   1
25497 000070F6 <res 00000001>     <1> HF_ERROR:    resb   1
25498 000070F7 <res 00000001>     <1> HF_INT_FLAG: resb   1
25499 000070F8 <res 00000001>     <1> HF_CNTRL:    resb   1
25500 000070F9 <res 00000004>     <1> DSK_STATE:   resb   4
25501 000070FD <res 00000002>     <1> DSK_TRK:     resb   2
25502                           <1>
25503                           <1> ;----------------------------------------
25504                           <1> ;     FIXED DISK DATA AREAS           :
25505                           <1> ;----------------------------------------
25506                           <1>
25507 000070FF <res 00000001>     <1> DISK_STATUS1:      resb   1            ; FIXED DISK STATUS
25508 00007100 <res 00000001>     <1> HF_NUM:            resb   1            ; COUNT OF FIXED DISK DRIVES
25509 00007101 <res 00000001>     <1> CONTROL_BYTE:      resb   1            ; HEAD CONTROL BYTE
25510                           <1> ;@PORT_OFF   resb   1           ; RESERVED (PORT OFFSET)
25511                           <1> ;port1_off   resb   1           ; Hard disk controller 1 - port offset
25512                           <1> ;port2_off   resb   1           ; Hard idsk controller 2 - port offset
25513                           <1>
25514 00007102 <res 00000002>     <1> alignb 4
25515                           <1>
25516                           <1> ;HF_TBL_VEC: resd   1           ; Primary master disk param. tbl. pointer
25517                           <1> ;HF1_TBL_VEC:      resd   1            ; Primary slave disk param. tbl. pointer
25518                           <1> HF_TBL_VEC: ; 22/12/2014
25519 00007104 <res 00000004>     <1> HDPM_TBL_VEC:      resd   1            ; Primary master disk param. tbl. pointer
25520 00007108 <res 00000004>     <1> HDPS_TBL_VEC:      resd   1            ; Primary slave disk param. tbl. pointer
25521 0000710C <res 00000004>     <1> HDSM_TBL_VEC:      resd   1            ; Secondary master disk param. tbl. pointer
25522 00007110 <res 00000004>     <1> HDSS_TBL_VEC:      resd   1            ; Secondary slave disk param. tbl. pointer
25523                           <1>
25524                           <1> ; 03/01/2015
25525 00007114 <res 00000001>     <1> LBAMode:           resb   1
25526                           <1>
25527                           <1> ; ///////////////////////////////////////////////////////////////////////////
25528
25529                               ;;; Real Mode Data (10/07/2015 - BSS)
25530
25531                               ;alignb 2
25532
25533                               %include 'ux.s' ; 12/04/2015 (unix system/user/process data)
25534                           <1> ; Retro UNIX 386 v1 Kernel - ux.s
25535                           <1> ; Last Modification: 04/12/2015
25536                           <1> ;
25537                           <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////
25538                           <1> ; (Modified from
25539                           <1> ;     Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
25540                           <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
25541                           <1> ; ------------------------------------------------------------------------------
25542                           <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
25543                           <1> ; (Original) Source Code by Ken Thompson (1971-1972)
25544                           <1> ; <Bell Laboratories (17/3/1972)>
25545                           <1> ; <Preliminary Release of UNIX Implementation Document>
25546                           <1> ; (Section E10 (17/3/1972) - ux.s)
25547                           <1> ; ///////////////////////////////////////////////////////////////////////////
25548                           <1>
25549 00007115 <res 00000001>     <1> alignb 2
25550                           <1>
25551                           <1> inode:
25552                           <1>       ; 11/03/2013.
25553                           <1>       ;Derived from UNIX v1 source code 'inode' structure (ux).
25554                           <1>       ;i.
25555                           <1>
25556 00007116 <res 00000002>     <1>       i.flgs:       resw 1
25557 00007118 <res 00000001>     <1>       i.nlks:       resb 1
25558 00007119 <res 00000001>     <1>       i.uid:  resb 1
25559 0000711A <res 00000002>     <1>         i.size:  resw 1 ; size
25560 0000711C <res 00000010>     <1>       i.dskp:       resw 8 ; 16 bytes
25561 0000712C <res 00000004>     <1>       i.ctim:       resd 1
25562 00007130 <res 00000004>     <1>       i.mtim:       resd 1
25563 00007134 <res 00000002>     <1>       i.rsvd:  resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
25564                           <1>
25565                           <1> I_SIZE      equ $ - inode
25566                           <1>
25567                           <1> process:
25568                           <1>       ; 06/05/2015
25569                           <1>       ; 11/03/2013 - 05/02/2014
```

```
25570                                  <1>       ;Derived from UNIX v1 source code 'proc' structure (ux).
25571                                  <1>       ;p.
25572                                  <1>
25573 00007136 <res 00000020>          <1>         p.pid:   resw nproc
25574 00007156 <res 00000020>          <1>         p.ppid:  resw nproc
25575 00007176 <res 00000020>          <1>         p.break: resw nproc
25576 00007196 <res 00000010>          <1>         p.ttyc:  resb nproc ; console tty in Retro UNIX 8086 v1.
25577 000071A6 <res 00000010>          <1>       p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
25578 000071B6 <res 00000010>          <1>       p.link:       resb nproc
25579 000071C6 <res 00000010>          <1>       p.stat:       resb nproc
25580                                  <1>
25581                                  <1>       ; 06/05/2015 (Retro UNIX 386 v1 fetaure only !)
25582 000071D6 <res 00000040>          <1>       p.upage: resd nproc ; Physical address of the process's
25583                                  <1>                           ; 'user' structure
25584                                  <1>
25585                                  <1>
25586                                  <1> P_SIZE      equ $ - process
25587                                  <1>
25588                                  <1>
25589                                  <1> ; fsp table (original UNIX v1)
25590                                  <1> ;
25591                                  <1> ;Entry
25592                                  <1> ;          15                                      0
25593                                  <1> ;  1     |---|-----------------------------------|
25594                                  <1> ;        |r/w|       i-number of open file       |
25595                                  <1> ;        |---|-----------------------------------|
25596                                  <1> ;        |               device number           |
25597                                  <1> ;        |---------------------------------------|
25598                                  <1> ;    (*) | offset pointer, i.e., r/w pointer to file |
25599                                  <1> ;        |---------------------------------------|
25600                                  <1> ;        |  flag that says    | number of processes  |
25601                                  <1> ;        |   file deleted     | that have file open  |
25602                                  <1> ;        |---------------------------------------|
25603                                  <1> ;  2     |                                       |
25604                                  <1> ;        |---------------------------------------|
25605                                  <1> ;        |                                       |
25606                                  <1> ;        |---------------------------------------|
25607                                  <1> ;        |                                       |
25608                                  <1> ;        |---------------------------------------|
25609                                  <1> ;        |                                       |
25610                                  <1> ;        |---------------------------------------|
25611                                  <1> ;  3     |                                       |
25612                                  <1> ;        |                                       |
25613                                  <1> ;
25614                                  <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
25615                                  <1>
25616                                  <1>
25617                                  <1> ; 15/04/2015
25618 00007216 <res 000001F4>          <1> fsp:   resb nfiles * 10 ; 11/05/2015 (8 -> 10)
25619 0000740A <res 00000020>          <1> bufp:  resd (nbuf+2) ; will be initialized
25620 0000742A <res 00000002>          <1> ii:    resw 1
25621 0000742C <res 00000002>          <1> idev:  resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
25622 0000742E <res 00000002>          <1> cdev:    resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
25623                                  <1> ; 18/05/2015
25624                                  <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
25625                                  <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
25626                                  <1> ;     0 -> root device (which has Retro UNIX 8086 v1 file system)
25627                                  <1> ;     1 -> mounted device (which has Retro UNIX 8086 v1 file system)
25628                                  <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
25629                                  <1> ;     0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
25630                                  <1> ;     1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
25631                                  <1> ;     2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
25632                                  <1> ;     3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
25633                                  <1> ;     4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
25634                                  <1> ;     5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
25635 00007430 <res 00000001>          <1> rdev:  resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
25636                                  <1>               ; as above, for physical drives numbers in following table
25637 00007431 <res 00000001>          <1> mdev:  resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
25638                                  <1> ; 15/04/2015
25639 00007432 <res 00000001>          <1> active:      resb 1
25640 00007433 <res 00000001>          <1>        resb 1 ; 09/06/2015
25641 00007434 <res 00000002>          <1> mnti:  resw 1
25642 00007436 <res 00000002>          <1> mpid:  resw 1
25643 00007438 <res 00000002>          <1> rootdir: resw 1
25644                                  <1> ; 14/02/2014
25645                                  <1> ; Major Modification: Retro UNIX 8086 v1 feature only!
25646                                  <1> ;                 Single level run queue
25647                                  <1> ;                 (in order to solve sleep/wakeup lock)
25648 0000743A <res 00000002>          <1> runq:  resw 1
25649 0000743C <res 00000001>          <1> imod:  resb 1
25650 0000743D <res 00000001>          <1> smod:  resb 1
25651 0000743E <res 00000001>          <1> mmod:  resb 1
25652 0000743F <res 00000001>          <1> sysflg:      resb 1
25653                                  <1>
25654                                  <1> alignb 4
25655                                  <1>
25656                                  <1> user:
25657                                  <1>       ; 04/12/2015
25658                                  <1>       ; 18/10/2015
25659                                  <1>       ; 12/10/2015
25660                                  <1>       ; 21/09/2015
25661                                  <1>       ; 24/07/2015
25662                                  <1>       ; 16/06/2015
25663                                  <1>       ; 09/06/2015
25664                                  <1>       ; 11/05/2015
25665                                  <1>       ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
25666                                  <1>       ; 10/10/2013
25667                                  <1>       ; 11/03/2013.
25668                                  <1>       ;Derived from UNIX v1 source code 'user' structure (ux).
25669                                  <1>       ;u.
25670                                  <1>
25671 00007440 <res 00000004>          <1>       u.sp:    resd 1 ; esp (kernel stack at the beginning of 'sysent')
25672 00007444 <res 00000004>          <1>       u.usp:   resd 1 ; esp (kernel stack points to user's registers)
25673 00007448 <res 00000004>          <1>       u.r0:    resd 1 ; eax
25674 0000744C <res 00000002>          <1>       u.cdir:       resw 1
```

```
25675 0000744E <res 0000000A>      <1>       u.fp:    resb 10
25676 00007458 <res 00000004>      <1>       u.fofp:         resd 1
25677 0000745C <res 00000004>      <1>       u.dirp:         resd 1
25678 00007460 <res 00000004>      <1>       u.namep:  resd 1
25679 00007464 <res 00000004>      <1>       u.off:   resd 1
25680 00007468 <res 00000004>      <1>       u.base:         resd 1
25681 0000746C <res 00000004>      <1>       u.count:  resd 1
25682 00007470 <res 00000004>      <1>       u.nread:  resd 1
25683 00007474 <res 00000004>      <1>       u.break:  resd 1 ; break
25684 00007478 <res 00000002>      <1>       u.ttyp:         resw 1
25685 0000747A <res 00000010>      <1>       u.dirbuf: resb 16 ; 04/12/2015 (10 -> 16)
25686                              <1>       ;u.pri:         resw 1 ; 14/02/2014
25687 0000748A <res 00000001>      <1>       u.quant:  resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
25688 0000748B <res 00000001>      <1>       u.pri:   resb 1 ;
25689 0000748C <res 00000002>      <1>       u.intr:         resw 1
25690 0000748E <res 00000002>      <1>       u.quit:         resw 1
25691                              <1>       ;u.emt:         resw 1 ; 10/10/2013
25692 00007490 <res 00000002>      <1>       u.ilgins: resw 1
25693 00007492 <res 00000002>      <1>       u.cdrv:         resw 1 ; cdev
25694 00007494 <res 00000001>      <1>       u.uid:   resb 1 ; uid
25695 00007495 <res 00000001>      <1>       u.ruid:         resb 1
25696 00007496 <res 00000001>      <1>       u.bsys:         resb 1
25697 00007497 <res 00000001>      <1>       u.uno:   resb 1
25698 00007498 <res 00000004>      <1>         u.upage:  resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
25699                              <1>       ; tty number (rtty, rcvt, wtty)
25700 0000749C <res 00000001>      <1>       u.ttyn:         resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
25701                              <1>       ; last error number
25702 0000749D <res 00000004>      <1>       u.error:  resd 1 ; 28/07/2013 - 09/03/2015
25703                              <1>                        ; Retro UNIX 8086/386 v1 feature only!
25704 000074A1 <res 00000004>      <1>       u.pgdir:  resd 1 ; 09/03/2015 (page dir addr of process)
25705 000074A5 <res 00000004>      <1>       u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
25706 000074A9 <res 00000004>      <1>       u.pbase:  resd 1 ; 20/05/2015 (physical base/transfer address)
25707 000074AD <res 00000002>      <1>       u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
25708                              <1>       ;u.pncount: resw 1
25709                              <1>              ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
25710                              <1>       ;u.pnbase:  resd 1
25711                              <1>              ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
25712                              <1>                     ; 09/06/2015
25713 000074AF <res 00000001>      <1>       u.kcall:  resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign

25714 000074B0 <res 00000001>      <1>       u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
25715                              <1>                     ; 24/07/2015 - 24/06/2015
25716                              <1>       ;u.args:  resd 1 ; arguments list (line) offset from start of [u.upage]
25717                              <1>                     ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
25718                              <1>                     ; ([u.args] points to argument count -argc- address offset)
25719                              <1>                     ; 24/06/2015
25720                              <1>       ;u.core:  resd 1 ; physical start address of user's memory space (for sys
exec)
25721                              <1>       ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
25722                              <1>                     ; 21/09/2015 (debugging - page fault analyze)
25723 000074B1 <res 00000004>      <1>       u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
25724                              <1>
25725 000074B5 <res 00000003>      <1> alignb 4
25726                              <1>
25727                              <1> U_SIZE      equ $ - user
25728                              <1>
25729                              <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
25730 000074B8 <res 00000004>      <1> pcore:  resd 1 ; physical start address of user's memory space (for sys exec)
25731 000074BC <res 00000004>      <1> ecore:  resd 1 ; physical start address of user's memory space (for sys exec)
25732 000074C0 <res 00000004>      <1> nbase:      resd 1 ; physical base address for 'namei' & 'sysexec'
25733 000074C4 <res 00000002>      <1> ncount: resw 1     ; remain byte count in page for 'namei' & 'sysexec'
25734 000074C6 <res 00000002>      <1> argc: resw 1 ; argument count for 'sysexec'
25735 000074C8 <res 00000004>      <1> argv: resd 1 ; argument list (recent) address for 'sysexec'
25736                              <1>
25737                              <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
25738                              <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
25739 000074CC <res 00000001>      <1> rw:    resb 1 ;; Read/Write sign (iget)
25740 000074CD <res 00000001>      <1> rwdsk:       resb 1 ;; Read/Write function number (diskio) - 16/06/2015
25741 000074CE <res 00000001>      <1> retry_count: resb 1 ; Disk I/O retry count - 11/06/2015
25742 000074CF <res 00000001>      <1>       resb 1 ;; Reserved (16/06/2015)
25743                              <1>
25744                              <1> ;alignb 4
25745                              <1>
25746                              <1> ; 22/08/2015
25747 000074D0 <res 00000C30>      <1> buffer: resb nbuf * 520
25748                              <1>
25749 00008100 <res 00000008>      <1> sb0:  resd 2
25750                              <1> ;s:
25751                              <1> ; (root disk) super block buffer
25752                              <1> systm:
25753                              <1>       ; 13/11/2015 (Retro UNIX 386 v1)
25754                              <1>       ; 11/03/2013.
25755                              <1>       ;Derived from UNIX v1 source code 'systm' structure (ux).
25756                              <1>       ;s.
25757                              <1>
25758 00008108 <res 00000002>      <1>       resw 1
25759 0000810A <res 00000168>      <1>       resb 360 ; 2880 sectors ; original UNIX v1 value: 128
25760 00008272 <res 00000002>      <1>       resw 1
25761 00008274 <res 00000020>      <1>       resb 32       ; 256+40 inodes ; original UNIX v1 value: 64
25762 00008294 <res 00000004>      <1>       s.time:       resd 1
25763 00008298 <res 00000004>      <1>       s.syst:       resd 1
25764 0000829C <res 00000004>      <1>         s.wait_: resd 1 ; wait
25765 000082A0 <res 00000004>      <1>       s.idlet: resd 1
25766 000082A4 <res 00000004>      <1>       s.chrgt: resd 1
25767 000082A8 <res 00000002>      <1>       s.drerr: resw 1
25768                              <1>
25769                              <1> S_SIZE      equ $ - systm
25770                              <1>
25771 000082AA <res 0000005E>      <1>       resb 512-S_SIZE ; 03/06/2015
25772                              <1>
25773 00008308 <res 00000008>      <1> sb1:  resd 2
25774                              <1> ; (mounted disk) super block buffer
25775                              <1> mount:
25776 00008310 <res 00000200>      <1>       resb 512  ; 03/06/2015
25777                              <1>
```

```
25778                           <1> ;/ ux -- unix
25779                           <1> ;
25780                           <1> ;systm:
25781                           <1> ;
25782                           <1> ;     .=.+2
25783                           <1> ;     .=.+128.
25784                           <1> ;     .=.+2
25785                           <1> ;     .=.+64.
25786                           <1> ;     s.time: .=.+4
25787                           <1> ;     s.syst: .=.+4
25788                           <1> ;     s.wait: .=.+4
25789                           <1> ;     s.idlet:.=.+4
25790                           <1> ;     s.chrgt:.=.+4
25791                           <1> ;     s.drerr:.=.+2
25792                           <1> ;inode:
25793                           <1> ;     i.flgs: .=.+2
25794                           <1> ;     i.nlks: .=.+1
25795                           <1> ;     i.uid:  .=.+1
25796                           <1> ;     i.size: .=.+2
25797                           <1> ;     i.dskp: .=.+16.
25798                           <1> ;     i.ctim: .=.+4
25799                           <1> ;     i.mtim: .=.+4
25800                           <1> ;     . = inode+32.
25801                           <1> ;mount:     .=.+1024.
25802                           <1> ;proc:
25803                           <1> ;     p.pid:  .=.+[2*nproc]
25804                           <1> ;     p.dska: .=.+[2*nproc]
25805                           <1> ;     p.ppid: .=.+[2*nproc]
25806                           <1> ;     p.break:.=.+[2*nproc]
25807                           <1> ;     p.link: .=.+nproc
25808                           <1> ;     p.stat: .=.+nproc
25809                           <1> ;tty:
25810                           <1> ;     . = .+[ntty*8.]
25811                           <1> ;fsp: .=.+[nfiles*8.]
25812                           <1> ;bufp:      .=.+[nbuf*2]+6
25813                           <1> ;sb0: .=.+8
25814                           <1> ;sb1: .=.+8
25815                           <1> ;swp: .=.+8
25816                           <1> ;ii:  .=.+2
25817                           <1> ;idev:      .=.+2
25818                           <1> ;cdev:      .=.+2
25819                           <1> ;deverr: .=.+12.
25820                           <1> ;active: .=.+2
25821                           <1> ;rfap:      .=.+2
25822                           <1> ;rkap:      .=.+2
25823                           <1> ;tcap:      .=.+2
25824                           <1> ;tcstate:.=.+2
25825                           <1> ;tcerrc: .=.+2
25826                           <1> ;mnti:      .=.+2
25827                           <1> ;mntd:      .=.+2
25828                           <1> ;mpid:      .=.+2
25829                           <1> ;clockp: .=.+2
25830                           <1> ;rootdir:.=.+2
25831                           <1> ;toutt:     .=.+16.
25832                           <1> ;touts: .=.+32.
25833                           <1> ;runq:      .=.+6
25834                           <1> ;
25835                           <1> ;wlist:     .=.+40.
25836                           <1> ;cc:  .=.+30.
25837                           <1> ;cf:  .=.+31.
25838                           <1> ;cl:  .=.+31.
25839                           <1> ;clist:     .=.+510.
25840                           <1> ;imod:      .=.+1
25841                           <1> ;smod:      .=.+1
25842                           <1> ;mmod:      .=.+1
25843                           <1> ;uquant: .=.+1
25844                           <1> ;sysflg: .=.+1
25845                           <1> ;pptiflg:.=.+1
25846                           <1> ;ttyoch: .=.+1
25847                           <1> ; .even
25848                           <1> ; .=.+100.; sstack:
25849                           <1> ;buffer: .=.+[ntty*140.]
25850                           <1> ;     .=.+[nbuf*520.]
25851                           <1> ;
25852                           <1> ; . = core-64.
25853                           <1> ;user:
25854                           <1> ;     u.sp:    .=.+2
25855                           <1> ;     u.usp:   .=.+2
25856                           <1> ;     u.r0:    .=.+2
25857                           <1> ;     u.cdir:  .=.+2
25858                           <1> ;     u.fp:    .=.+10.
25859                           <1> ;     u.fofp:  .=.+2
25860                           <1> ;     u.dirp:  .=.+2
25861                           <1> ;     u.namep: .=.+2
25862                           <1> ;     u.off:   .=.+2
25863                           <1> ;     u.base:  .=.+2
25864                           <1> ;     u.count: .=.+2
25865                           <1> ;     u.nread: .=.+2
25866                           <1> ;     u.break: .=.+2
25867                           <1> ;     u.ttyp:  .=.+2
25868                           <1> ;     u.dirbuf:.=.+10.
25869                           <1> ;     u.pri:   .=.+2
25870                           <1> ;     u.intr:  .=.+2
25871                           <1> ;     u.quit:  .=.+2
25872                           <1> ;     u.emt:   .=.+2
25873                           <1> ;     u.ilgins:.=.+2
25874                           <1> ;     u.cdev:  .=.+2
25875                           <1> ;     u.uid:   .=.+1
25876                           <1> ;     u.ruid:  .=.+1
25877                           <1> ;     u.bsys:  .=.+1
25878                           <1> ;     u.uno:   .=.+1
25879                           <1> ;. = core
25880
25881                               ;; Memory (swap) Data (11/03/2015)
25882                               ; 09/03/2015
```

```
25883 00008510 <res 00000002>          swpq_count: resw 1 ; count of pages on the swap que
25884 00008512 <res 00000004>          swp_drv:    resd 1 ; logical drive description table address of the swap drive/disk
25885 00008516 <res 00000004>          swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

25886 0000851A <res 00000004>          swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
25887 0000851E <res 00000004>          swpd_next:  resd 1 ; next free page block
25888 00008522 <res 00000004>          swpd_last:  resd 1 ; last swap page block
25889
25890 00008526 <res 00000002>          alignb 4
25891
25892                                    ; 10/07/2015
25893                                    ; 28/08/2014
25894 00008528 <res 00000004>          error_code: resd 1
25895                                    ; 29/08/2014
25896 0000852C <res 00000004>          FaultOffset:       resd 1
25897                                    ; 21/09/2015
25898 00008530 <res 00000004>          PF_Count:   resd 1 ; total page fault count
25899                                                       ; (for debugging - page fault analyze)
25900                                                       ; 'page _fault_handler' (memory.inc)
25901                                                       ; 'sysgeterr' (u9.s)
25902                                    ;; 21/08/2015
25903                                    ;;buffer: resb (nbuf*520) ;; sysdefs.inc, ux.s
25904
25905                                    bss_end:
25906
25907                                    ; 27/12/2013
25908                                    _end:  ; end of kernel code (and read only data, just before bss)
```