IOPL

From Wikipedia, the free encyclopedia

The **IOPL** (I/O Privilege level) flag is a flag found on all IA-32 compatible x86 CPUs. It occupies bits 12 and 13 in the FLAGS register. In protected mode and long mode, it shows the I/O privilege level of the current program or task. The CPL (Current Privilege Level) (CPL0, CPL1, CPL2, CPL3) of the task or program must be less than or equal to the IOPL in order for the task or program to access I/O ports.

The IOPL can be changed using POPF(D) and IRET(D) only when the current privilege level is Ring 0.

Besides IOPL, the I/O Port Permissions in the TSS also take part in determining the ability of a task to access an I/O port.

FLAGS register

The **FLAGS** register is the status register in Intel x86 microprocessors that contains the current state of the processor. This register is 16 bits wide. Its successors, the **EFLAGS** and **RFLAGS** registers, are 32 bits and 64 bits wide, respectively. The wider registers retain compatibility with their smaller predecessors.

Flags

Intel x86 FLAGS register[1]			
Bit #	**Abbreviation**	**Description**	**Category**
FLAGS			
0	CF	Carry flag	Status
1		Reserved	
2	PF	Parity flag	Status
3		Reserved	
4	AF	Adjust flag	Status
5		Reserved	
6	ZF	Zero flag	Status
7	SF	Sign flag	Status
8	TF	Trap flag (single step)	Control
9	IF	Interrupt enable flag	Control
10	DF	Direction flag	Control
11	OF	Overflow flag	Status
12-13	IOPL	I/O privilege level (286+ only), always 1 on 8086 and 186	System
14	NT	Nested task flag (286+ only), always 1 on 8086 and 186	System
15		Reserved, always 1 on 8086 and 186, always 0 on later models	
EFLAGS			
16	RF	Resume flag (386+ only)	System
17	VM	Virtual 8086 mode flag (386+ only)	System
18	AC	Alignment check (486SX+ only)	System
19	VIF	Virtual interrupt flag (Pentium+)	System
20	VIP	Virtual interrupt pending (Pentium+)	System
21	ID	Able to use CPUID instruction (Pentium+)	System
22		Reserved	
23		Reserved	
24		Reserved	
25		Reserved	
26		Reserved	
27		Reserved	
28		Reserved	
29		Reserved	
30		Reserved	
31		Reserved	
RFLAGS			
32-63		Reserved	

Use

The POPF, POPFD, and POPFQ instructions read from the stack the first 16, 32, and 64 bits of the flags register, respectively. POPFD was introduced with the i386 architecture and POPFQ with the x64 architecture. In 64-bit mode, PUSHF/POPF and PUSHFQ/POPFQ are available but not PUSHFD/POPFD.[2]

The following assembly code changes the direction flag (DF):

```
pushf ; Pushes the current flags onto the stack
pop ax ; Pop the flags from the stack into ax register
push ax ; Push them back onto the stack for storage
xor ax, 400h ; toggle the DF flag only, keep the rest of the flags
push ax ; Push again to add the new value to the stack
popf ; Pop the newly pushed into the FLAGS register
; ... Code here ...
popf ; Pop the old FLAGS back into place
```

In practical software, the `cld` and `std` instructions are used to clear and set the direction flag, respectively. Some instructions in assembly language use the FLAGS register. The conditional jump instructions use certain flags to compute. For example, `jz` uses the zero flag, `jc` uses the carry flag and `jo` uses the overflow flag. Other conditional instructions look at combinations of several flags.

Determination of processor type

Testing if certain bits in the FLAGS register are changeable allows determining what kind of processor is installed. For example, the alignment flag can only be changed on the 486 and above, so if it can be changed then the CPU is a 486 or higher. These methods of processor detection were made obsolete by the CPUID instruction, which was first included in theIntel Pentium.