Princeton University
COS 217: Introduction to Programming Systems
x86-64 Condition Codes

Condition Codes

Bits in the EFLAGS register

`cmp{q,l,w,b}` *`src, dest`*

> Performs the subtraction *dest* - *src*, and sets the condition codes depending upon the difference:

Condition Code	
ZF (zero flag)	**Mathematically**: Set ZF to 1 iff the difference was 0. **Physically**: Set ZF to 1 iff all bits of the difference are 0.
SF (sign flag)	**Mathematically**: Set SF to 1 iff the difference was negative. **Physically**: Set SF to 1 iff the most significant bit of the difference is 1.
CF (carry flag)	**Mathematically**: Set CF to 1 iff the difference is incorrect when we view the operands and difference as **unsigned** integers. **Physically**: Set CF to 1 iff dest<src.
OF (overflow flag)	**Mathematically**: Set OF to 1 iff the difference is incorrect when we view the operands and difference as **signed** integers. **Physically**: Two's complement src. Compute dest+src. Set OF to 1 iff dest>0 and src>0 and sum<0, or dest<0 and src<0 and sum>=0.

Conditional Control Transfer Instructions (Used After Comparing Unsigned Numbers)

Instruction	Jump if and only if:
`je  (jump iff equal)`	`ZF`
`jne (jump iff not equal)`	`~ZF`
`jb  (jump iff below)`	`CF`
`jae (jump iff above or equal)`	`~CF`
`jbe (jump iff below or equal)`	`CF | ZF`
`ja  (jump iff above)`	`~(CF | ZF)`

Why does `jb` jump if and only if `CF`? Informal explanation:

```
(1) largenum - smallnum => correct result => CF=0 => don't jump (not below)

(2) smallnum - largenum => incorrect result => CF=1 => jump (below)
```

So jump if and only if `CF`.

Conditional Control Transfer Instructions
(Used After Comparing Signed Numbers)

Instruction	Jump if and only if:
`je  (jump iff equal)`	`ZF`
`jne (jump iff not equal)`	`~ZF`
`jl  (jump iff less than)`	`OF ^ SF`
`jge (jump iff greater than or equal)`	`~(OF ^ SF)`
`jle (jump iff less than or equal)`	`(OF ^ SF) | ZF`
`jg  (jump iff greater than)`	`~((OF ^ SF) | ZF)`

Why does `jl` jump if and only if `(OF ^ SF)`? Informal explanation:

```
(1) largeposnum - smallposnum
    correct result => OF=0, SF=0 => (OF^SF)==0 => don't jump (not <)

(2) smallposnum - largeposnum
    correct result => OF=0, SF=1 => (OF^SF)== 1 => jump (<)

(3) largenegnum - smallnegnum
    correct result => OF=0, SF=1 => (OF^SF)== 1 => jump (<)

(4) smallnegnum - largenegnum
    correct result => OF=0, SF=0 => (OF^SF)== 0 => don't jump (not <)

(5) posnum - negnum
    correct result => OF=0, SF=0 => (OF^SF)== 0 => don't jump (not <)

(6) posnum - negnum
    incorrect result => OF=1, SF=1 => (OF^SF)==0 => don't jump (not <)

(7) negnum - posnum
    correct result => OF=0, SF=1 => (OF^SF)==1 => jump (<)

(8) negnum - posnum
    incorrect result => OF=1, SF=0 => (OF^SF)== 1 => jump (<)
```

So jump if and only if `(OF ^ SF)`.