Princeton University
COS 217: Introduction to Programming Systems
IA-32 Condition Codes and
Conditional Control Transfer Instructions

Condition Codes

Bits in the EFLAGS register

```
cmpl src, dest
```

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

Condition Code	Set When
ZF (zero flag)	The difference is 0
SF (sign flag)	The difference is negative, that is, the high order bit of the difference is 1
CF (carry flag)	The difference is mathematically incorrect when we view the operands as **unsigned** integers
OF (overflow flag)	The difference is mathematically incorrect when we view the operands as **signed** integers

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)	SF ^ OF
jge (jump iff greater than or equal)	~(SF ^ OF)
jle (jump iff less than or equal)	(SF ^ OF) | ZF
jg (jump iff greater than)	~((SF ^ OF) | ZF)

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)

Examples (assuming a 5-bit computer for simplicity):

Instruction	Subtraction Performed	Resulting Condition Code Values	Conditional Jump Instructions

`cmpl $6, $12`	`01100`
	`00110`
	`-----`
	`00110`
		`ZF = 0 (diff is not 0)`
		`SF = 0 (diff high order bit is 0)`
		`CF = 0 (unsigned diff is correct)`
		`OF = 0 (signed diff is correct)`
			`jl: (SF ^ OF) == 0`
			`   So don’t jump`
			`jb: CF == 0`
			`   So don’t jump`

`cmpl $12, $6`	`00110`
	`01100`
	`-----`
	`11010`
		`ZF = 0 (diff is not 0)`
		`SF = 1 (diff high order bit is 1)`
		`CF = 1 (unsigned diff is incorrect)`
		`OF = 0 (signed diff is correct)`
			`jl: (SF ^ OF) == 1`
			`   So jump`
			`jb: CF == 1`
			`   So jump`

`cmpl $6, $-12`
`cmpl $6, $20`
	`10100`
	`00110`
	`-----`
	`01110`
		`ZF = 0 (diff is not 0)`
		`SF = 0 (diff high order bit is 0)`
		`CF = 0 (unsigned diff is correct)`
		`OF = 1 (signed diff is incorrect)`
			`jl: (SF ^ OF) == 1`
			`   So jump`
			`jb: CF == 0`
			`   So don’t jump`

`cmpl $-12, $6`
`cmpl $20, $6`
	`00110`
	`10100`
	`-----`
	`10010`
		`ZF = 0 (diff is not 0)`
		`SF = 1 (diff high order bit is 1)`
		`CF = 1 (unsigned diff is incorrect)`
		`OF = 1 (signed diff is incorrect)`
			`jl: (SF ^ OF) == 0`
			`   So don’t jump`
			`jb: CF == 1`
			`   So jump`

`cmpl $-6, $12`
`cmpl $28, $12`
	`01100`
	`11010`
	`-----`
	`10010`
		`ZF = 0 (diff is not 0)`
		`SF = 1 (diff high order bit is 1)`
		`CF = 1 (unsigned diff is incorrect)`
		`OF = 1 (signed diff is incorrect)`
			`jl: (SF ^ OF) == 0`
			`   So don’t jump`
			`jb: CF == 1`
			`   So jump`

`cmpl $12, $-6`
`cmpl $12, $28`
	`11010`
	`01100`
	`-----`
	`01110`
		`ZF = 0 (diff is not 0)`
		`SF = 0 (diff high order bit is 0)`
		`CF = 0 (unsigned diff is correct)`
		`OF = 1 (signed diff is incorrect)`
			`jl: (SF ^ OF) == 1`
			`   So jump`
			`jb: CF == 0`
			`   So don’t jump`

`cmpl $-6, $-12`
`cmpl $28, $20`
	`10100`
	`11010`
	`-----`
	`11010`
		`ZF = 0 (diff is not 0)`
		`SF = 1 (diff high order bit is 1)`
		`CF = 1 (unsigned diff is incorrect)`
		`OF = 0 (signed diff is correct)`
			`jl: (SF ^ OF) == 1`
			`   So jump`
			`jb: CF == 1`
			`   So jump`

`cmpl $-12, $-6`
`cmpl $20, $28`
	`11010`
	`10100`
	`-----`
	`00110`
		`ZF = 0 (diff is not 0)`
		`SF = 0 (diff high order bit is 0)`
		`CF = 0 (unsigned diff is correct)`
		`OF = 1 (signed diff is incorrect)`
			`jl: (SF ^ OF) == 0`
			`   So don’t jump`
			`jb: CF == 0`
			`   So don’t jump`