Status Flags

[Ref: Intel® 64 and IA-32 Architectures Software Developer's Manual, §3.4.3]

- Six designated bits within the 64-bit `RFLAGS` register that indicate the results of arithmetic, logical, and shift operations. The lower 32-bit section of this register is called `EFLAGS`.
  - `CF`: Carry Flag (Was a carry generated from MSB?, i.e., unsigned overflow).
  - ~~PF: Parity Flag.~~
  - ~~AF: Auxiliary Carry Flag.~~
  - `ZF`: Zero Flag (Was the result `0x0`?)
  - `SF`: Sign Flag (Was the result negative?)
  - `OF`: Overflow Flag (Did the operation cause a two's-complement overflow, either positive or negative?)

Implicit Setting of Status Flags

- Arithmetic, logic, and shift operations set the flags (mostly) in the expected manner.
  - Logical operations set `CF` and `OF` to 0.
  - Shift operations set `CF` to the last bit shifted out. `OF` is changed only when the shift amount is 1 (complicated rules).
  - `INC` and `DEC` set `OF` and `ZF`, but leave `CF` unchanged.

- `LEAQ` does not alter the status flags.

Explicit Setting of Status Flags

- There are two instructions (one arithmetic, the other logical) whose sole purpose is to set status flags.

- `cmpX` S_1, S_2: Sets status flags based on the value of $S_2 - S_1$ *but discards the value*.

- `testX` S_1, S_2: Sets status flags based on the value of $S_2 \& S_1$ *but discards the value*.

- $X \in \{B, W, L, Q\}$.

Accessing or Using Status Flags

- Usually not accessed directly, but through a set of conditions that are combinations of the status flag values and correspond to typical tests needed in compiling programming language constructs.
- Three forms
  - `setCND D`: Set a single destination byte specified by `D` (which can be either in register or in memory) to 0 or 1 depending on a combination of status flags specified by `CND`.
  - `jCND L`: Jump to label `L` if the combination of status flags specified by `CND` evaluates to true.
  - `cmovCND S, D`: If condition `CND` evaluates to true, move the contents of `S` to `D`; otherwise, do nothing.
    - Operand `S` is R-type or M-type.
    - Operand `D` is R-type (16, 32, or 64 bits only).
    - Transfer width is inferred from destination register name.

Possible CND Combinations

Combination	setCND form	jCND form	cmovCND form	Comments

`ZF`	`sete D`
	`setz D`
		`je L`
		`jz L`
			`cmove S, D`
			`cmovz S, D`
				Equal to zero

`~ZF`	`setne D`
	`setnz D`
		`jne L`
		`jnz L`
			`cmovne S, D`
			`cmovnz S, D`
				Not equal to zero

`SF`	`sets D`	`js L`	`cmovs S, D`	Negative
`~SF`	`setns D`	`jns L`	`cmovns S, D`	Non-negative
`~(SF^OF)&~ZF`	`setg D`	`jg L`	`cmovg S, D`	> (signed)
`~(SF^OF)`	`setge D`	`jge L`	`cmovge S, D`	≥ (signed)
`SF^OF`	`setl D`	`jl L`	`cmovl S, D`	< (signed)
`(SF^OF)|ZF`	`setle D`	`jle L`	`cmovle S, D`	≤ (signed)
`~CF&~ZF`	`seta D`	`ja L`	`cmova S, D`	Above (unsigned >)
`~CF`	`setae D`	`jae L`	`cmovae S, D`	Above or equal (unsigned ≥)
`CF`	`setb D`	`jb L`	`cmovb S, D`	Below (unsigned <)
`CF|ZF`	`setbe D`	`jbe L`	`cmovbe S, D`	Below or equal (unsigned ≤)

Unconditional Jumps

- Two forms at assembly-language level: *direct* and *indirect*.
  - Direct: `jmp L`, where `L` is a label.
  - Indirect: `jmp *D`, where `D` is a register or memory format operand specifier.

- At machine code level, the instruction has a large number of variants, and the label is translated into an absolute or relative offset from `%rip`.

- No indirect conditional jumps.