Tutorial-x86 status flag

Po-An Tsai
02/20/2015
X86 ISA

• A ISA widely used in Intel products
  • 8080 in 1978
  • Haswell in 2014
X86 ISA

• Four general purpose register
  • Register a (ax)
  • Register b (bx)
  • Register c (cx)
  • Register d (dx)
X86 ISA

• Variable-length instructions
  • `add (Ra+Rb)`: 2 bytes
  • `inc (Ra++)`: 1 byte
  • `cmp (Ra-Mem[imm32])`: 6 bytes
X86 ISA

- CISC:
  - supports high-level programming constructs such as procedure calls, loop control, and complex addressing modes.
  - allow data structure and array accesses to be combined into single instructions
  - **Ex:** xchg Ra Mem[imm32]
    - Exchange the value in R1 and Mem[imm32]
Data move (load/store)

• `mov` destination, source
  • Can be reg/reg
  • Or reg/mem
  • But one mem at most
Arithmetic

- add/sub
  - Ra=Rb+Rc
- inc/dec
  - Ra++ / Ra--
Status flags

- Zero flag (ZF)—destination equals zero
- Sign flag (SF)—destination is negative
- Carry flag (CF)—unsigned value out of range
- Overflow flag (OF)—signed value out of range
Status flags

```
mov ax, 00FFh
add ax, 1 ; AX= 0100h SF= 0 ZF= 0 CF= 0
sub ax, 1 ; AX= 00FFh SF= 0 ZF= 0 CF= 0
add al, 1 ; AL= 00h SF= 0 ZF= 1 CF= 1
mov bh, 6Ch
add bh, 95h ; BH= 01h SF= 0 ZF= 0 CF= 1

mov al, 2
sub al, 3 ; AL= FFh SF= 1 ZF= 0 CF= 1
```
**cmp instruction**

- Temp = R - Mem[imm32]
- Set flags

- Example: destination > source

```
mov al,5
cmp al,-2 ; Sign flag == Overflow flag
```
**cmp instruction**

- Temp = R - Mem[imm32]
- Set flags

**Example: destination < source**

```
mov al,-1
cmp al,5 ; Sign flag != Overflow flag
```
j l  instruction

• Jump if left less than right
  • It checks flags in fact
j1 instruction

If (var1 < var2)
    var 3 = 1;
Else
    var 3 = 0;
...
...
L1: mov var3, 1
L2: ...
...

mov ax, var1
cmp ax, var2
j1 L1
mov var3, 0
jmp L2
What ‘s good?

• No need to use ALU twice
  • In MIPS:
    sub r3, r1, r2
    bnz r3, r0, target
  • In x86:
    cmp r1, r2
    jl target
What ‘s bad?

• All arithmetic operation will influence flags
  • Flags will be set/reset by other instructions

• Stronger dependency
  • jl has to follow cmp