Basic Assembly

More conditional branching

Assembly language programming
By xorpd

xorpd.net

Objectives

- We will learn about the CMP instruction, which is useful for numbers comparison.

- We will understand how to compare unsigned numbers and signed numbers using specialized instructions.

CMP

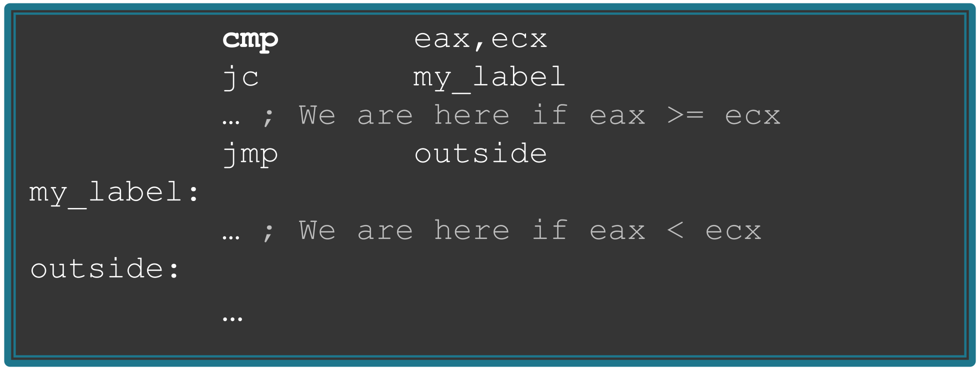
- We want to be able to compare numbers.
- We could use SUB and then JC for example.
  - compare two unsigned numbers.

```
          sub       eax,ecx
          jc        my_label
          … ; We are here if eax >= ecx
          jmp       outside
my_label:
          … ; We are here if eax < ecx
outside:
          …
```

- But SUB overrides our compared values.
- We could use the CMP instruction instead.
  - Just like SUB, but doesn’t store the subtraction’s result!

CMP (Cont.)

- CMP A,B
  - Subtracts: A – B, Changes flags accordingly but doesn’t change A or B.
- Very useful for numbers comparisons.
- Example:



```
          cmp       eax,ecx
          jc        my_label
          … ; We are here if eax >= ecx
          jmp       outside
my_label:
          … ; We are here if eax < ecx
outside:
          …
```

- This time eax is not overridden.

Unsigned vs Signed comparison

- Comparison of Unsigned numbers and Signed numbers is different.
  - `0xffffffff > 0x00000001` considering unsigned numbers.
  - `0xffffffff < 0x00000001` considering signed numbers (Two’s complement).
    - Negative < Positive

- It is our responsibility as programmers to know what is the meaning of the numbers we compare.
- We will learn about specialized instructions for each type of comparison.

Unsigned Comparison

- We would like to compare two unsigned numbers.
- We already know that we could achieve that by combining the CMP instruction with JC (and maybe JZ).

- Instead of dealing with the Carry and Zero flags, we have some ready to use instructions:
  - JB – Jump if Below.
  - JBE – Jump if Below or Equal.
  - JA – Jump if Above.
  - JAE – Jump if Above or Equal.
- These instructions only work for unsigned comparison!

Unsigned Comparison (Cont.)

- How do these instructions work?

Instruction	Condition being checked
`JB   (Jump Below)`	CF = 1
`JBE  (Jump Below Equal)`	CF = 1 or ZF = 1
`JA   (Jump Above)`	CF = 0 and ZF = 0
`JAE  (Jump Above Equal)`	CF = 0

- JB is just a different name for JC.
- JAE is just a different name for JNC.

Unsigned Comparison (Example)

- We can use JB instead of JC.

```
           cmp        eax,ecx
           jc         my_label
           … ; We are here if eax >= ecx
           jmp        outside
my_label:
           … ; We are here if eax < ecx
outside:
           …
```

→

```
           cmp        eax,ecx
           jb         my_label
           … ; We are here if eax >= ecx
           jmp        outside
my_label:
           … ; We are here if eax < ecx
outside:
           …
```

- A bit more readable.
  - But no change in behaviour.

Signed Comparison

- Signed comparison could be a bit trickier, as there are more cases to consider.

- We could use CMP and then check the Sign, Overflow and Zero flags.
  - Each combination of those flags will have some meaning regarding the result of the comparison.

- Instead, we have some ready to use instructions:
  - JL – Jump if Less.
  - JLE – Jump if Less or Equal.
  - JG – Jump if Greater.
  - JGE – Jump if Greater or Equal.

Signed Comparison (Cont.)

- How do these instructions work?

Instruction	Condition being checked
`JG   (Jump Greater)`	SF = OF and ZF = 0
`JGE  (Jump Greater Equal)`	SF = OF
`JL   (Jump Less)`	SF ≠ OF
`JLE  (Jump Less Equal)`	SF ≠ OF or ZF = 1

- What does SF = OF mean?
- If we understand how JGE works, we will understand how all those instructions work.

Signed Comparison – JGE

- Assume that we have just executed the instruction: cmp ecx,edx.
  - We execute ecx – edx and change the flags accordingly.
- If OF=0:
  - No overflow has occurred – The result has the “correct” sign.
  - If SF = 0, the result is positive, hence ecx ≥ edx in the signed sense.
    (OF = SF = 0)
  - If SF = 1, the result is negative, hence ecx < edx in the signed sense.
    (0 = OF ≠ SF = 1)
- If OF = 1:
  - An overflow has occurred – The result has the “wrong” sign.
  - If SF = 0, the result should be negative, hence ecx < edx in the signed sense. (1 = OF ≠ SF = 0)
  - If SF = 1, the result should be positive, hence ecx ≥ edx in the signed sense. (OF = SF = 1)
- ecx ≥ edx in the signed sense iff OF = SF.

Signed Comparison (Example)

- We can use JL instead of JB in our example.
  - Signed comparison instead of unsigned comparison.

```
; Unsigned comparison
           cmp        eax,ecx
           jb         my_label
           … ; We are here if eax >= ecx
           jmp        outside
my_label:
           … ; We are here if eax < ecx
outside:
           …
```

→

```
; Signed comparison
           cmp        eax,ecx
           jl         my_label
           … ; We are here if eax >= ecx
           jmp        outside
my_label:
           … ; We are here if eax < ecx
outside:
           …
```

- The details of checking the flags are hidden from us.
  - We don’t have to remember the details.

Readable code

- Prefer the more meaningful instructions in your programs.
  - Use JB instead of JC if you are comparing unsigned numbers.
  - Use JGE instead of checking that SF=OF on your own.
- The processor doesn’t care.
- Makes life easier for you and for your coworkers.

Summary

- The CMP instruction is just like SUB, but doesn’t change the compared values.
- Unsigned comparison is done using:
  - JA, JAE, JB, JBE.
- Signed comparison is done using:
  - JG, JGE, JL, JLE.
- Prefer meaningful JCC instructions in your code over instructions like JC,JZ,JS,JO.