

Recap 1

a is in %eax

b is in %edx

cmpl %edx, %eax

setl %al

↓
set if lesser

if (a < b)

%al = 1 (True)

~~if~~ else

%al = 0

(False)

Same for the Zero flag.

ZF = 1 \Rightarrow The last operation yielded a zero

ZF = 0 \Rightarrow The last operation yielded something other than zero

movl \$0x1, %eax

decl %eax

If I check the zero flag now, it will be 1

setz %al

Scaled Index Operand

(, %eax , 4)

Imm (E_b , E_a , S)

Imm (, E_a , S)

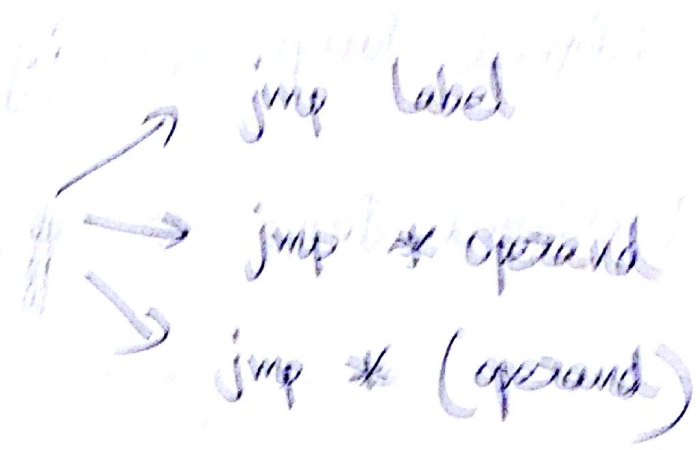
can only be

1, 2, 4 or 8.

and so on.

Jump instruction

① Unconditional Jumps



② ~~Conditional~~ Conditional Jumps

① je Label
(or) jz Label

Jump condition
ZF \Rightarrow If ZF is 0, Jump!

② jne Label
(or) jnz

(ZF) \Rightarrow If ZF is 1, Jump!

③ js Label

SF

④ jns Label

SF

Signed Jump \rightarrow jg, jge, jl, jle

Unsigned Jump \rightarrow ja, jae, jb, jbe

If Statements

Assumptions

C
if (a == b)

a, b are integers

{

a--; // print

a is in %eax

b is in %ebx

}

Assembly

cmpl %ebx, %eax

jne skip-decr: // (~ZF)

dec %eax

If a == b

then a - b = 0

So ZF would be

set to 1

So ~ZF = 0

(False)

Aside : ZF is set (ZF = 1)
Note : ZF is cleared (ZF = 0)

For loops

sum, i are integers.

C
sum = 0 ;

for (i = 1 ; i <= 10 ; i++)

{

sum = sum + i ;

}

Assembly

movl \$10, %ecx

movl \$0, %eax ← sum is in %eax

movl \$1, %edx // ← i is in %edx

jmp Label_1

Label_2:

addl ⁱ %edx, ^{sum} %eax // sum = sum + i

incl %edx // i++

Label_1:

cmpl ¹⁰ %ecx, ⁱ %edx

jump when i <= 10

jle Label_2

Absolute encoding

We give the exact 4-byte address to specify the target.

Relative > Absolute ?

① Compact we may only need 1 or 2 bytes.

② Object code can be shifted to anywhere in memory without worrying about the address.

Relative encoding examples

(Show code on screen)

①

804 83 ae :	<u>75</u>	<u>04</u>	jne	<u>80483b4</u>
+ 2				
80483b0 :				
<u>+ 04</u>				

Diagram: An arrow points from the underlined '04' in the instruction to the underlined '+ 04' in the displacement calculation.

②

804828 f :	<u>74</u>	<u>05</u>	je	<u>8048296</u>
8048291 :				
<u>05</u>				

Diagram: An arrow points from the underlined '05' in the instruction to the underlined '05' in the displacement calculation. Below the displacement is the binary representation: 0000 0101.

③

804 8357 :	72	e7	jb	<u>8048340</u>
804 8359 :				
<u>19</u>				

Diagram: A vertical arrow points from the instruction to the binary representation of the displacement.

16 | 25

1 - 9 ↑

11100111

-128 + 64 + 32 + 4 + 2 + 1

= -25₁₆ = -19₁₆