CS208

W 4 Feb 2026

When execute an instruction

e.g.

addl %eax, %ebx →

effect ebx = eax + ebx

Also sets

ZF = 1 if the sum is 0, else ZF = 0

addl $5, %ebx

ebx = ebx + 5

SF = 1 if sum < 0, else SF = 0

Conditional jumps

```
je .L2
```

"jump if equal (to zero)"

if ZF = 1, jump, otherwise don't.

```
jne .L3
```

jump if ZF = 0

```
int something(int a, int b){
    if(a > b)
```

performs
edi − esi == a − b
+ sets ZF + SF
accordingly

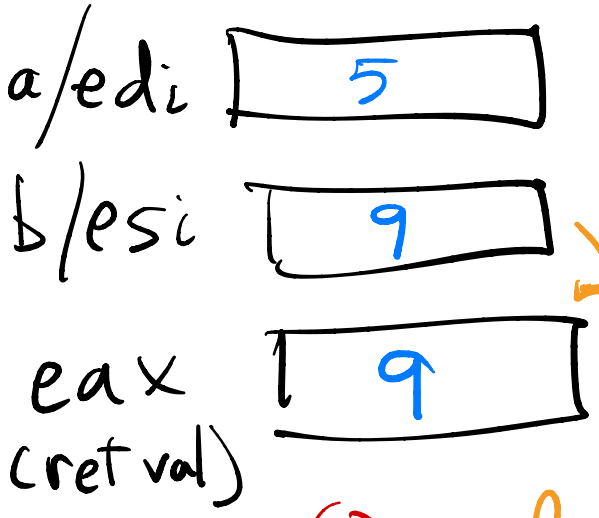
```
a ~ edi   (1st param)
b ~ esi   (2nd)

cmpl %esi, %edi
jle .L2
```

jump if a − b <= 0

max(5, 9)

a b

edi esi



2 movl ①

② cmpl esi, edi

a − b ⇝ −4 ⇝ ZF=0, SF=1

③ jle — take the jump

④ ret

SF is the sign bit of the result of previous instruction

So if result < 0, SF = 1

≥ 0, SF = 0

test is like cmp

but does &

test %eax, %ebx

eax & ebx & sets SF & ZF accordingly

x & x == x

test %edi, %edi

asking: is edi 0?

is edi < 0?

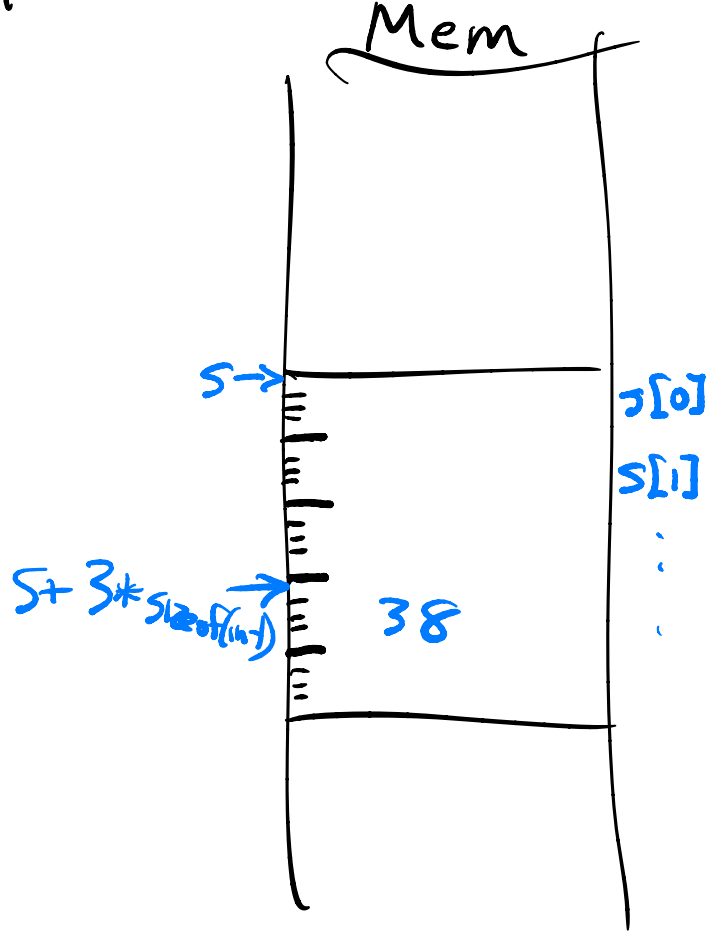
SF = 1 if edi < 0

ZF = 1 if edi == 0

Digression. What happens when I do

```
int s[5];
s[3] = 38;
```

*(s + 3*sizeof(int))



Addressing modes

$0 $5 "immediate"

%eax "register (direct)"

(%rsp) "indirect"

go to memory where rsp is pointing

movl (%rsp), %edx copy int pointed to by rsp into edx