

Introduction to 8086 Assembly

Lecture 5

Jump, Conditional Jump, Looping, Compare instructions

Labels and jumping (the jmp instruction)



K. N. Toosi
University of Technology

```
mov eax, 1
```

```
add eax, eax
```

```
jmp label1
```

```
xor eax, eax
```

```
label1:
```

```
sub eax, 303
```

Labels and jumping (the jmp instruction)



memory

```
mov eax, 1
```

```
add eax, eax
```

```
jmp label1
```

```
xor eax, eax
```

```
label1:
```

```
sub eax, 303
```

1001

```
mov eax, 1
```

1006

```
add eax, eax
```

1008

```
jmp label1
```

1010

```
xor eax, eax
```

1012

```
sub eax, 303
```

Labels and jumping (the jmp instruction)



memory

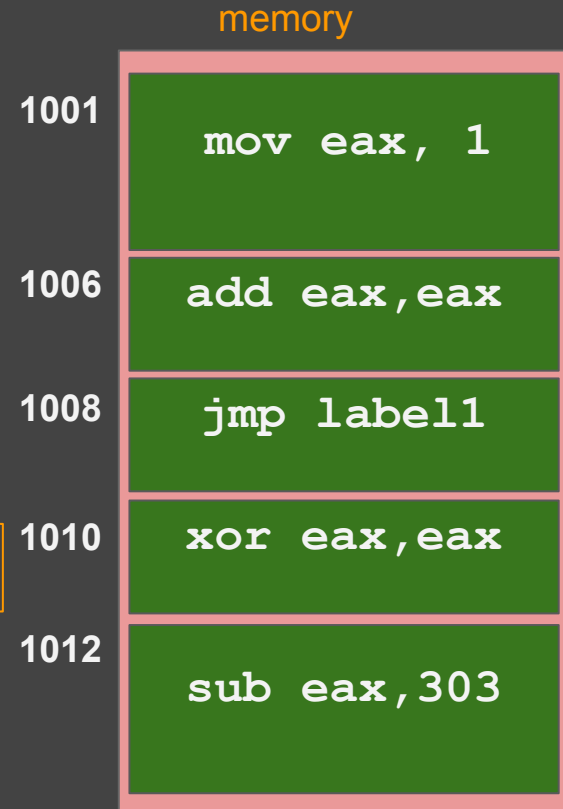
```
mov eax, 1  
add eax, eax  
jmp label1  
xor eax, eax  
label1:  
sub eax, 303
```

1001	mov eax, 1
1006	add eax, eax
1008	jmp label1
1010	xor eax, eax
1012	sub eax, 303

Labels and jumping (the jmp instruction)



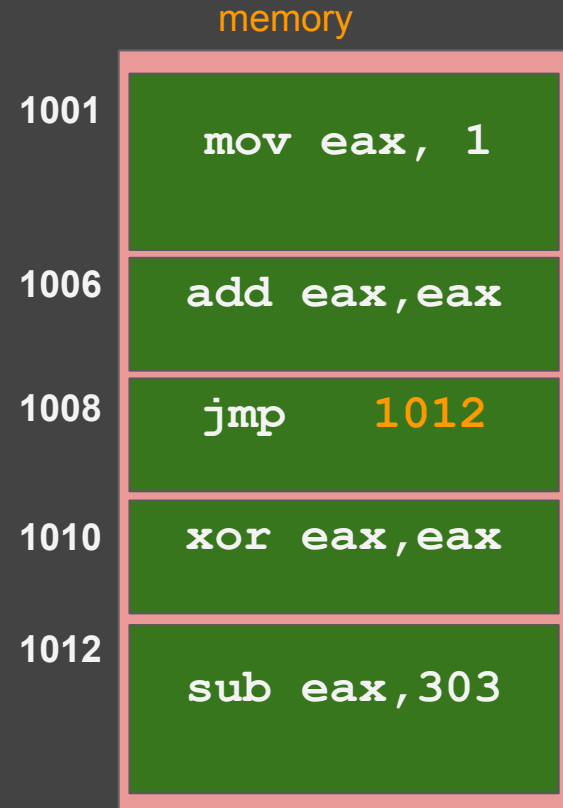
```
mov eax, 1  
add eax, eax  
jmp label1  
xor eax, eax  
label1:  
sub eax, 303
```



Labels and jumping (the jmp instruction)



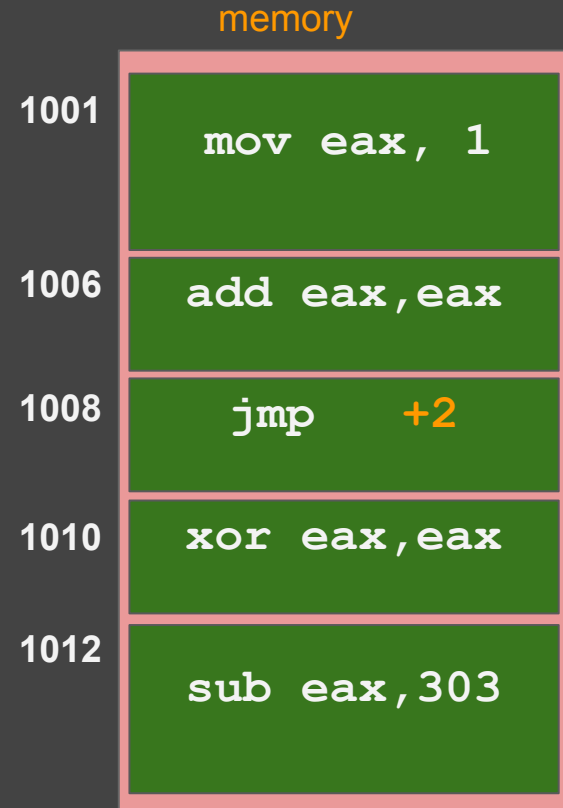
```
mov eax, 1  
add eax, eax  
jmp label1  
xor eax, eax  
label1:  
sub eax, 303
```



Labels and jumping (the jmp instruction)



```
mov eax, 1  
add eax, eax  
jmp label1  
xor eax, eax  
label1:  
sub eax, 303
```



Labels and jumping (the jmp instruction)



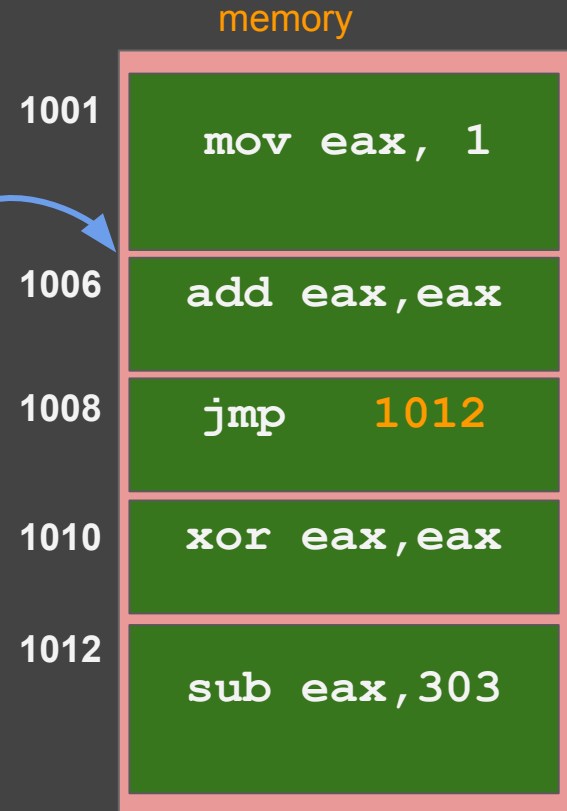
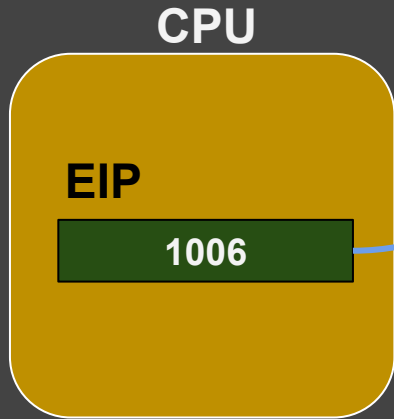
```
mov eax, 1  
add eax, eax  
jmp label1  
xor eax, eax  
label1:  
sub eax, 303
```



Labels and jumping (the jmp instruction)



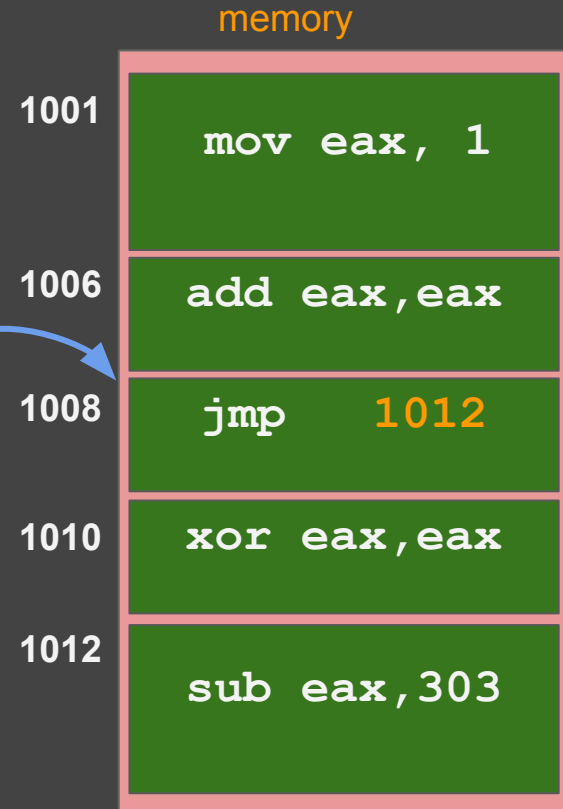
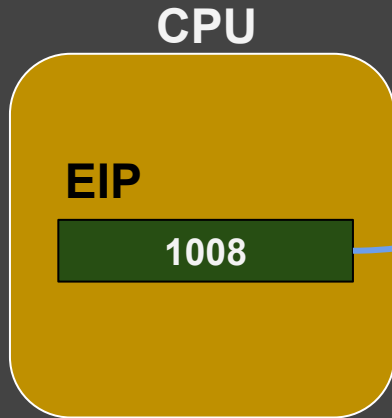
Executing: `mov eax, 1`



Labels and jumping (the jmp instruction)



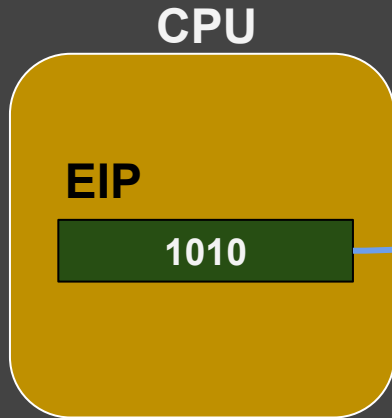
Executing: `add eax, eax`



Labels and jumping (the jmp instruction)



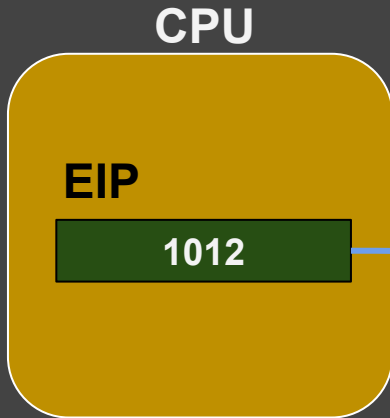
Executing: `jmp 1012`



Labels and jumping (the jmp instruction)



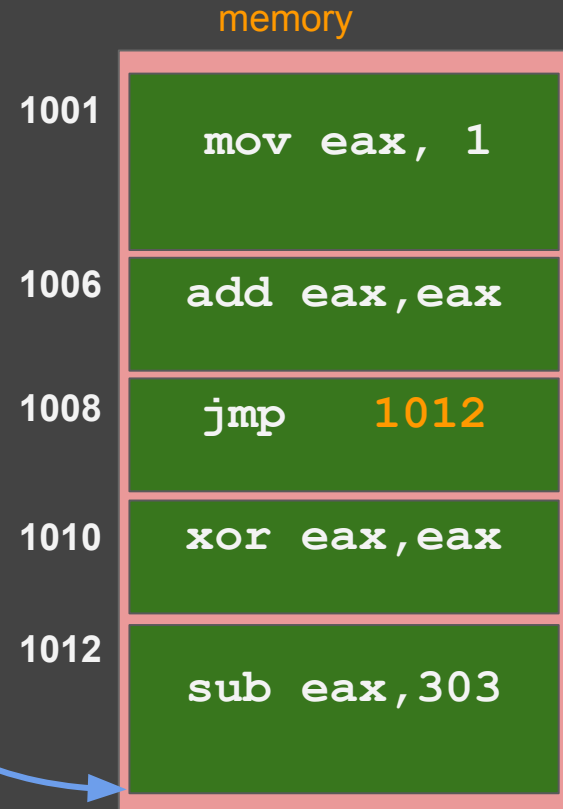
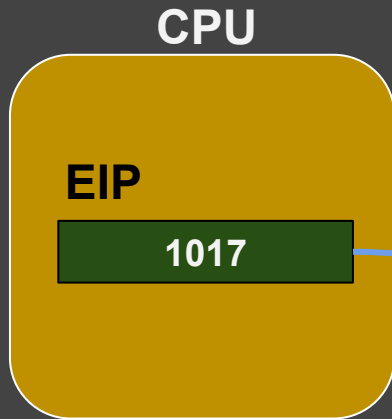
Executing: `jmp 1012`



Labels and jumping (the jmp instruction)



Executing: `sub eax, 303`



Infinite loop



K. N. Toosi
University of Technology

```
mov eax, 0
```

```
loop1:
```

```
call print_int
```

```
call print_nl
```

```
inc eax
```

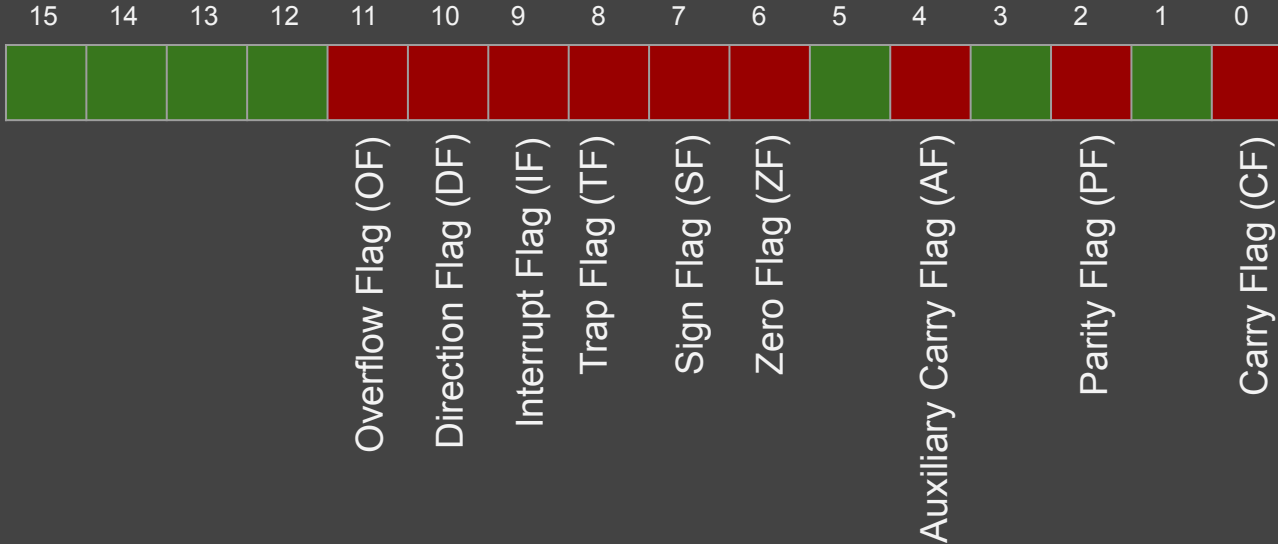
```
jmp loop1
```

```
infinite_loop.asm
```

Remember: the FLAGS Register



K. N. Toosi
University of Technology



CF: carry flag

OF: overflow flag

SF: sign flag

ZF: zero flag

PF: parity flag

DF: direction flag

IF: interrupt flag

Conditional jumps



JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice



unsigned integer:

```
if (eax == ebx)
    esi = 0
```

JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

unsigned integer:

```
if (eax == ebx)
    esi = 0
```

```
sub eax, ebx
jnz next
```

```
mov esi, 0
```

```
next:
```



K. N. Toosi
University of Technology

JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

signed integer:

```
if (eax == ebx)
    esi = 0
```

```
sub eax, ebx
jnz next
```

```
mov esi, 0
```

```
next:
```



JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

signed integer:

```
if (eax == - ebx)
    edi = 4
```



K. N. Toosi
University of Technology

JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

signed integer:

```
if (eax == - ebx)
    edi = 4
```

```
add eax, ebx
jnz next
```

```
mov edi, 4
```

```
next:
```



JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

unsigned integer:

```
if (eax >= ebx)
    esp -= 4
```



JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice

unsigned integer:

```
if (eax >= ebx)
    esp -= 4
```

```
sub eax, ebx
jc next
```

```
sub esp, 4
```

```
next:
```



JZ	Jump if ZF=1
JNZ	Jump if ZF=0
JO	Jump if OF=1
JNO	Jump if OF=0
JS	Jump if SF=1
JNS	Jump if SF=0
JC	Jump if CF=1
JNC	Jump if CF=0
JP	Jump if PF=1
JNP	Jump if PF=0

Practice



K. N. Toosi
University of Technology

signed integer:

```
if (eax < ebx)
    ebp += 8
```

$x - y$

$x < y \Rightarrow SF = 1$

$x \geq y \Rightarrow SF = 0$

Practice



K. N. Toosi
University of Technology

signed integer:

$x - y$

```
if (eax < ebx)
```

```
    ebp += 8
```

OF=0

$x < y \Rightarrow SF = 1$

$x \geq y \Rightarrow SF = 0$

Practice



signed integer:

$x - y$

```
if (eax < ebx)
```

```
    ebp += 8
```

OF=0	$x < y \Rightarrow SF = 1$ $x \geq y \Rightarrow SF = 0$
OF=1	$x < 0 < y \Rightarrow SF = 0$ $x > 0 > y \Rightarrow SF = 1$

Practice



signed integer:

```
if (eax < ebx) ebp += 8
```

```
    sub eax, ebx
```

```
    jo  overflow
```

```
    jns endl
```

```
if_cond:
```

```
    add ebp, 8
```

```
    jmp endl
```

```
overflow:
```

```
    jns if_cond
```

```
endl:
```

$x - y$

OF=0	$x < y \Rightarrow SF = 1$ $x \geq y \Rightarrow SF = 0$
OF=1	$x < 0 < y \Rightarrow SF = 0$ $x > 0 > y \Rightarrow SF = 1$

Practice



signed integer:

```
if (eax < ebx)
    ebp += 8
else
    ebp -= 8
```

$x - y$

OF=0	$x < y \Rightarrow SF = 1$ $x \geq y \Rightarrow SF = 0$
OF=1	$x < 0 < y \Rightarrow SF = 0$ $x > 0 > y \Rightarrow SF = 1$

Practice:



```
    call read_int
    mov ebx, eax

    call read_int

l1:   sub ebx, eax
      jnc l1

      add eax, ebx

      call print_int
      call print_nl
```

Practice:



32 bits

```
call read_int
mov ebx, eax

call read_int

l1:
sub ebx, eax
jnc l1

add eax, ebx

call print_int
call print_nl
```

	binary
0	00000000000000000000000000000000
1	00000000000000000000000000000001
2	00000000000000000000000000000010
:	:
:	:
$2^{32}-3$	111111111111111111111111111111101
$2^{32}-2$	111111111111111111111111111111110
$2^{32}-1$	111111111111111111111111111111111

Other conditional jump commands



sub x, y		unsigned	signed		
JE	label	jump if x == y (same as JZ)	JE	label	jump if x == y (same as JZ)
JNE	label	jump if x != y (same as JNZ)	JNE	label	jump if x != y (same as JNZ)
JA	label	jump if x > y	JG	label	jump if x > y
JNBE	label		JNLE	label	
JB	label	jump if x < y	JL	label	jump if x < y
JNAE	label		JNGE	label	
JAE	label	jump if x >= y	JGE	label	jump if x >= y
JNB	label		JNL	label	
JBE	label	jump if x <= y	JLE	label	jump if x <= y
JNA	label		JNG	label	

Practice:



```
    call read_int
    mov ebx, eax

    call read_int

l1:   sub ebx, eax
      jnc l1

      add eax, ebx

      call print_int
      call print_nl
```


Practice:



```
call read_int  
mov ebx, eax  
  
call read_int
```

l1:

```
sub ebx, eax  
jnc l1  
  
add eax, ebx  
  
call print_int  
call print_nl
```

rem.asm

```
call read_int  
mov ebx, eax  
  
call read_int
```

l1:

```
sub ebx, eax  
jae l1  
  
add eax, ebx  
  
call print_int  
call print_nl
```

rem2.asm

Practice:



K. N. Toosi
University of Technology

Practice: Also print quotient

```
    call read_int
    mov ebx, eax

    call read_int

l1:
    sub ebx, eax
    jnc l1

    add eax, ebx

    call print_int
    call print_nl
```

rem.asm

Practice:



```
    call read_int
    mov ebx, eax

    call read_int

l1:
    sub ebx, eax
    jnc l1

    add eax, ebx

    call print_int
    call print_nl
```

rem.asm

```
    call read_int
    mov ebx, eax

    call read_int
    mov ecx, 0

l1:
    sub ebx, eax
    inc ecx
    jnc l1

    dec ecx
    add eax, ebx
    call print_int
    call print_nl

    mov eax, ecx
    call print_int
    call print_nl
```

div.asm

Practice:



```
    call read_int
    mov ecx, eax

    call read_int

    mov ebx, 0
l1:  add ebx, eax

    dec ecx
    jnz l1

    mov eax, ebx
    call print_int
    call print_nl
```

The LOOP instruction



```
call read_int
mov ecx, eax

call read_int

mov ebx, 0
l1:
add ebx, eax

dec ecx
jnz l1

mov eax, ebx
call print_int
call print_nl
```

```
call read_int
mov ecx, eax

call read_int

mov ebx, 0
l1:
add ebx, eax

loop l1

mov eax, ebx
call print_int
call print_nl
```

The loop commands



K. N. Toosi
University of Technology

```
loop    lbl    ecx--; if (ecx!=0) goto lbl
loopz   lbl    ecx--; if (ecx!=0 && ZF=1) goto lbl
loopnz  lbl    ecx--; if (ecx!=0 && ZF=0) goto lbl
```



The loop commands

```
loop      lbl      ecx--; if (ecx!=0) goto lbl
loopz    lbl      ecx--; if (ecx!=0 && ZF=1) goto lbl
loopnz   lbl      ecx--; if (ecx!=0 && ZF=0) goto lbl

loope    ≡ loopz
loopne   ≡ loopnz
```

Example: Count up to N



```
    call read_int
    mov ebx, eax

    mov eax, 1

l1:  call print_int
     call print_nl

     inc eax

     mov ecx, ebx
     sub ecx, eax
     jnc l1
```


Example: Count up to N



```
    call read_int
    mov  ebx, eax

    mov  eax, 1

l1:  call print_int
     call print_nl

     inc  eax

     mov  ecx, ebx
     sub  ecx, eax
     jnc  l1
```

```
    call read_int
    mov  ebx, eax

    mov  eax, 1

l1:  call print_int
     call print_nl

     inc  eax

     mov  ecx, ebx
     sub  ecx, eax
     jae  l1
```

Example: Count up to N



K. N. Toosi
University of Technology

```
    call read_int
    mov  ebx, eax

    mov  eax, 1

l1:  call print_int
     call print_nl

     inc  eax

     mov  ecx, ebx
     sub  ecx, eax
     jnc  l1
```

```
    call read_int
    mov  ebx, eax

    mov  eax, 1

l1:  call print_int
     call print_nl

     inc  eax

     mov  ecx, ebx
     sub  ecx, eax
     jae  l1
```

```
    call read_int
    mov  ebx, eax

    mov  eax, 1

l1:  call print_int
     call print_nl

     inc  eax

     mov  ecx, ebx
     sub  ecx, eax
     jge  l1
```

using sub before jump; what's wrong?



K. N. Toosi
University of Technology

```
    call read_int
    mov ebx, eax

    mov eax, 1
l1:  call print_int
     call print_nl

     inc eax

     mov ecx, ebx
     sub ecx, eax
     jae l1
```

the cmp instruction



```
    call read_int
    mov ebx, eax

    mov eax, 1
l1:  call print_int
    call print_nl

    inc eax

    mov ecx, ebx
    sub ecx, eax
    jae l1
```

```
    call read_int
    mov ebx, eax

    mov eax, 1
l1:  call print_int
    call print_nl

    inc eax

    cmp ebx, eax
    jae l1
```



The `cmp` instruction

```
sub eax, ebx
```

```
cmp eax, ebx
```

- `cmp x, y`
- subtracts `y` from `x` (like `sub x, y`)
- does not store the result (`x` is not changed)
- flags are set (as though a subtraction has taken place)

The cmp instruction



cmp x, y		unsigned	signed		
JE	label	jump if x == y	JE	label	jump if x == y
JNE	label	jump if x != y	JNE	label	jump if x != y
JA	label	jump if x > y	JG	label	jump if x > y
JNBE	label		JNLE	label	
JB	label	jump if x < y	JL	label	jump if x < y
JNAE	label		JNGE	label	
JAE	label	jump if x >= y	JGE	label	jump if x >= y
JNB	label		JNL	label	
JBE	label	jump if x <= y	JLE	label	jump if x <= y
JNA	label		JNG	label	

Practice



K. N. Toosi
University of Technology

(signed)

```
if (eax > ebx)
```

```
    edi=1
```

```
else
```

```
    edi=2
```

Practice



(signed)

if (eax > ebx)

edi=1

else

edi=2

```
cmp eax, ebx
```

```
jle else_lbl
```

```
mov edi, 1
```

```
jmp endif
```

```
else_lbl:
```

```
mov edi, 2
```

```
endif:
```


Practice



(signed)

if (eax > ebx)

edi=1

else

edi=2

```
cmp eax, ebx
```

```
jle else_lbl
```

```
mov edi, 1
```

```
jmp endif
```

```
else_lbl:
```

```
mov edi, 2
```

```
endif:
```

executed at the
same time?