III. STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING TECHNIQUES

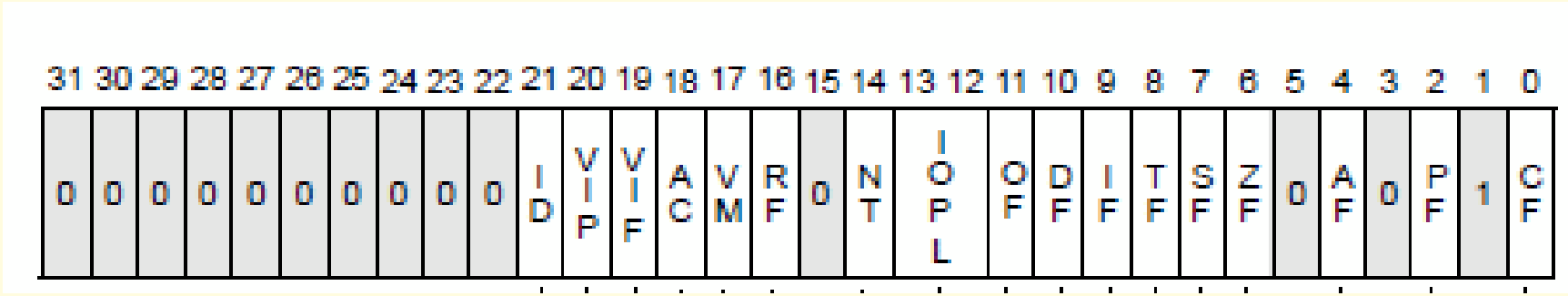
Control Transfer Instructions

More on Conditional Jumps

- Instructions that check the eFLAGs register before jumping
- The FLAGs checked by Conditional jumps
  - Carry
  - Parity
  - Zero
  - Sign
  - Overflow flags

The eFLAGs Register

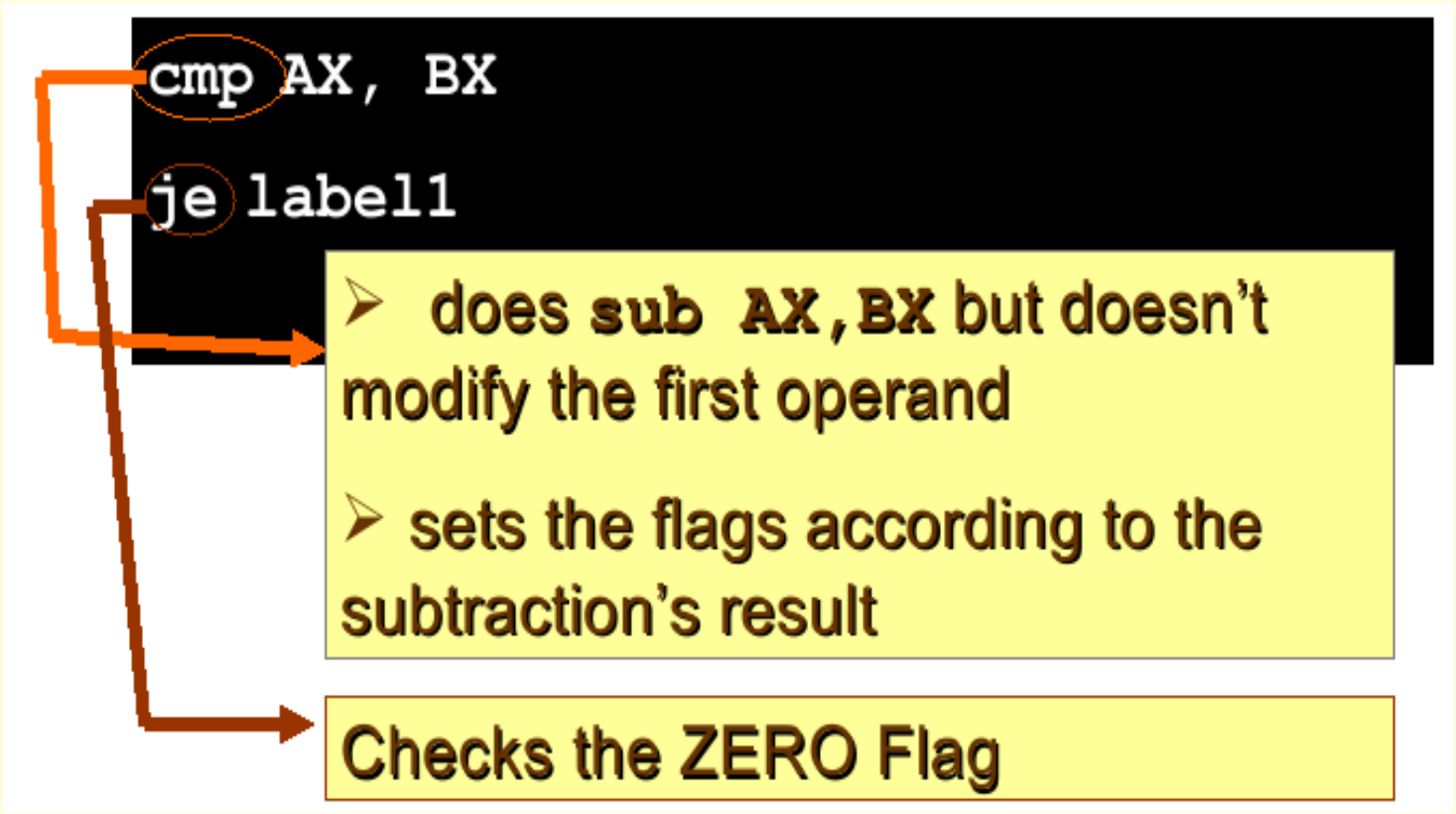
- A special purpose register
- Certain bits in this register serve as Flags



The eFLAGs Register

FLAGS	SET (1)	CLEARED (0)
Overflow	OV [overflow]	NV [no overflow]
Sign	NG [negative]	PL [positive]
Zero	ZR [zero]	NZ [not zero]
Parity	PE [even]	PO [odd]
Carry	CY [carry]	NC [no carry]

Conditional Jumps



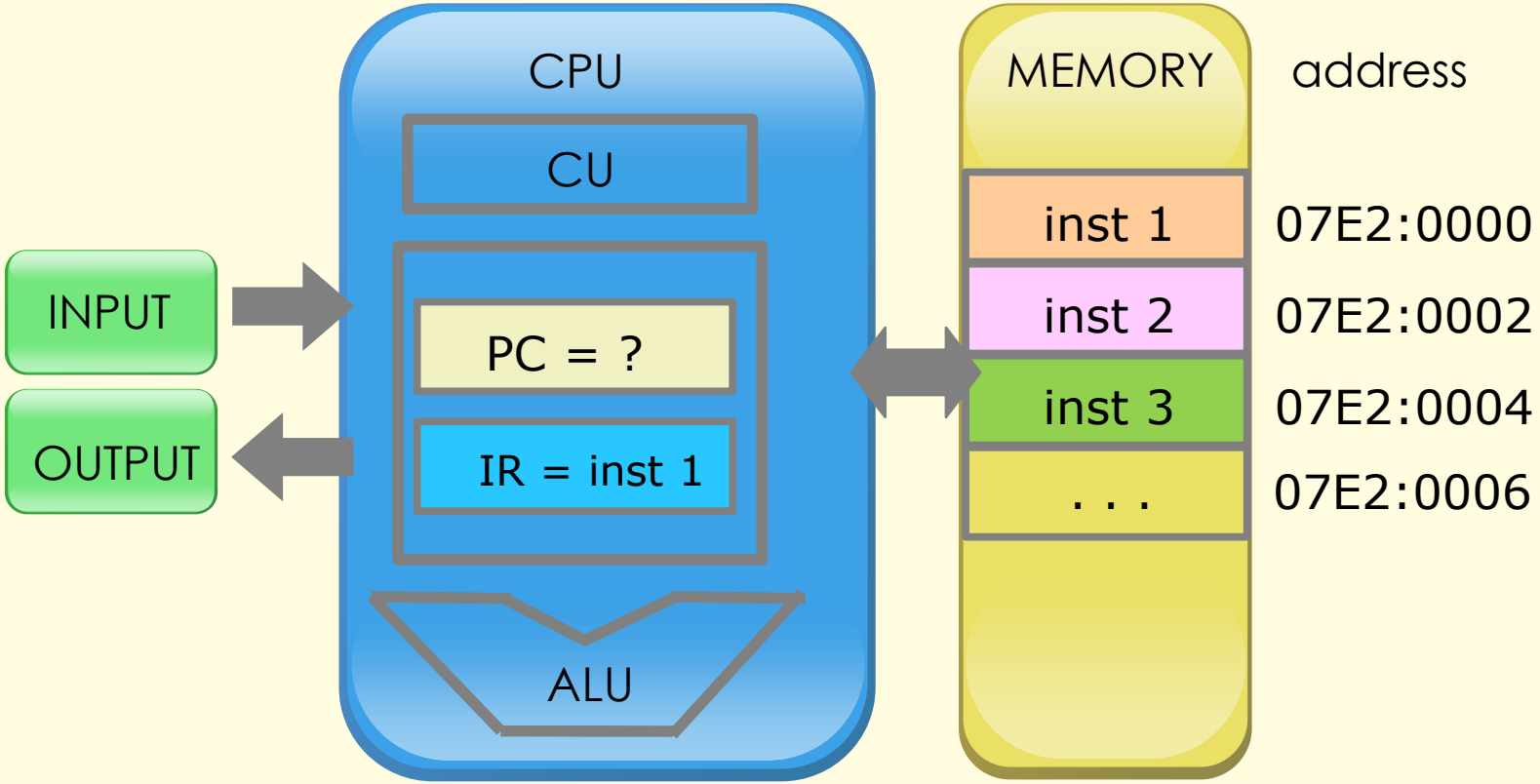
Sequential Statements

- Fetch-Decode-Execute

  CS:EIP is the PC (Program Counter)
- When the fetched instruction is copied into the instruction register, EIP is automatically incremented by X.

  X = instruction length (in bytes)
- Since EIP is automatically INCREMENTED by X, the instructions are executed SEQUENTIALLY by default.

Recall


CPU
CU
PC = ?
IR = inst 1
ALU
INPUT
OUTPUT
MEMORY
address
inst 1
inst 2
inst 3
. . .
07E2:0000
07E2:0002
07E2:0004
07E2:0006

How can we change the default execution?

What will happen to EIP?
High-level code/algorithm:

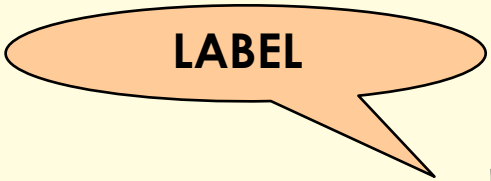
Current instruction ► if (condition) then do1
IP points here ► do1: code 0
code 1
code 2
else
do2: code x
code y
code z

JMP Statement

- This instruction takes one operand: a label
- For example: jmp doon

```
dito_ba:    mov eax, 4
            mov ebx, 1
            mov ecx, msg1
            mov edx, len1
            int 80h

doon:       mov eax, 1
            mov ebx, 0
            int 80h
```



JMP Statement

nasm –f elf sample.asm –l sample.lst

```
00000000 E916000000      jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

JMP Statement

machine code/
opcode of

```
00000000 E916000000      jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

JMP Statement

machine code/
opcode of

```
00000000 E916000000 ?    jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

JMP Statement

jmp

a value in backwords storage format;
therefore actual value is: 00000016

```
00000000 E916000000      jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

So we have:
E9 00000016

JMP Statement

When Instruction Register contains E9 00000016

CS:EIP will 'point' here

So EIP = 00000005 h

```
00000000 E916000000      jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

Note: As the instruction in IR is executed:
EIP will be: EIP+00000016 = 0000001B h

JMP Statement

When Instruction Register contains E9 00000016

CS:EIP will 'point' here

So EIP = 00000005 h

```
00000000 E916000000      jmp doon

                         dito_ba:
00000005 B804000000      mov eax, 4
0000000A BB01000000      mov ebx, 1
0000000F B9[00000000]    mov ecx, msg1
00000014 BA11000000      mov edx, len1
00000019 CD80            int 80h

                         doon:
0000001B B801000000      mov eax, 1
00000020 BB00000000      mov ebx, 0
00000025 CD80            int 80h
```

Note: As the instruction in IR is executed:
EIP will be: EIP+00000016 = 0000001B h

Addition of hex numbers

Decimal:

$$\begin{array}{r} 19 \\ +\ 9 \\ \hline 28 \end{array}$$

Hexadecimal:

$$\begin{array}{r} 19 \\ +\ 9 \\ \hline 22 \end{array}$$

JMP Statement

Reminder:

- The displacement is in Backwords Storage Format.
- A displacement in a jump can either be a positive or a negative value.

JMP Statement

```
                                dito_ba:
00000000 B804000000             mov eax, 4
00000005 BB01000000             mov ebx, 1
0000000A B9[00000000]           mov ecx, msg1
0000000F BA11000000             mov edx, len1
00000014 CD80                   int 80h
00000016 E9E5FFFFFF             jmp dito_ba

                                doon:
0000001B B801000000             mov eax, 1
00000020 BB00000000             mov ebx, 0
00000025 CD80                   int 80h
```

FFFF FFE5

JMP Statement

```
                                dito_ba:
00000000 B804000000             mov eax, 4
00000005 BB01000000             mov ebx, 1
0000000A B9[00000000]           mov ecx, msg1
0000000F BA11000000             mov edx, len1
00000014 CD80                   int 80h
00000016 E9E5FFFFFF             jmp dito_ba

                                doon:
0000001B B801000000             mov eax, 1
00000020 BB0000000              mov ebx,
00000025 CD80                   int 80h
```

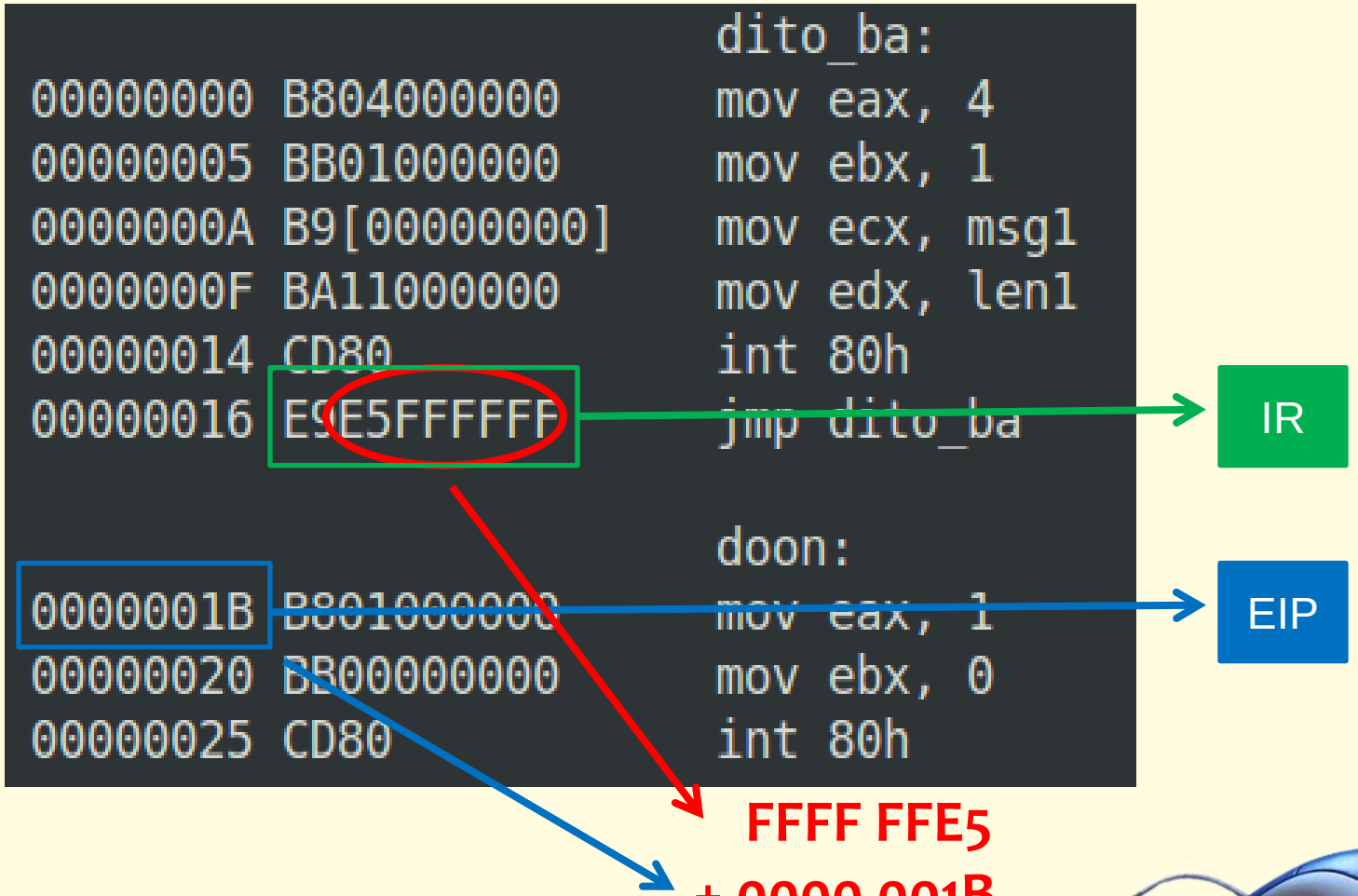
The value is negative, so this jumps to a label 'above'

FFFF FFE5

sign-bit

F = 1111 (binary)

JMP Statement



Quiz

Given the following assembly program, what is its equivalent C code?

```
cmp bl, cl
jng label2
mov al, bl
jmp next
label2:
mov al, cl
next:
```

```
if (_____1_____) {
_____2_____
}
else {
_____3_____
}
```

Quiz

Given the following assembly program, what is its equivalent C code?

```
cmp bl, cl
jng label2
mov al, bl
jmp next
label2:
mov al, cl
next:
```

```
if (   bl <= cl    ) {
    al = cl;
}
else {
    al = bl;
}
```

Quiz

Given the following assembly program, what is its equivalent C code?

```
cmp bl, cl
jng label2
mov al, bl
jmp next
label2:
mov al, cl
next:
```

```
if (   bl > cl    ) {
    al = bl;
}
else {
    al = cl;
}
```