Fall 2025/26– Lecture Notes # 3

- Introduction to Assembly Programming
- Program Segments

Introduction to Assembly Language Programming

- **ADD *instruction***

ADD destination, source; *dest = dest + source*

mnemonic *operands*

Example:

```
MOV     AL,24H    ;move 24H into  AL
MOV     DL,11H    ;move 11H into  DL
ADD     AL,DL     ;AL=AL+DL       (AL=35H)(DL =11H)
```

```
MOV     CH,24H    ;move 24H into  CH
MOV     BL,11H    ;move 11H into  BL
ADD     CH,BL     ;CH=CH+BL       (CH=35H)
```

```
MOV     CH,24H    ;load one operand into CH
ADD     CH,11H    ;add the second operand to CH  (CH=35H)
```

Introduction to Assembly Language Programming

- **ADD *instruction***

ADD destination, source; *dest = dest + source*

mnemonic *operands*

❖ If destination register is followed by an immediate data as the source, it is called the immediate operand.

```
MOV     CH,24H
ADD     CH,11H
```

❖ 8-bit registers can hold FFH (255) as the maximum value. Addition of larger numbers can be performed by the 16-bit nonsegment registers.

```
MOV     AX,34EH
MOV     DX,6A5H
ADD     DX,AX           ;DX=DX+AX    (DX=9F3H)
MOV     CX,34EH
ADD     CX,6A5H         ;CX=34EH+6A5=9F3H
```

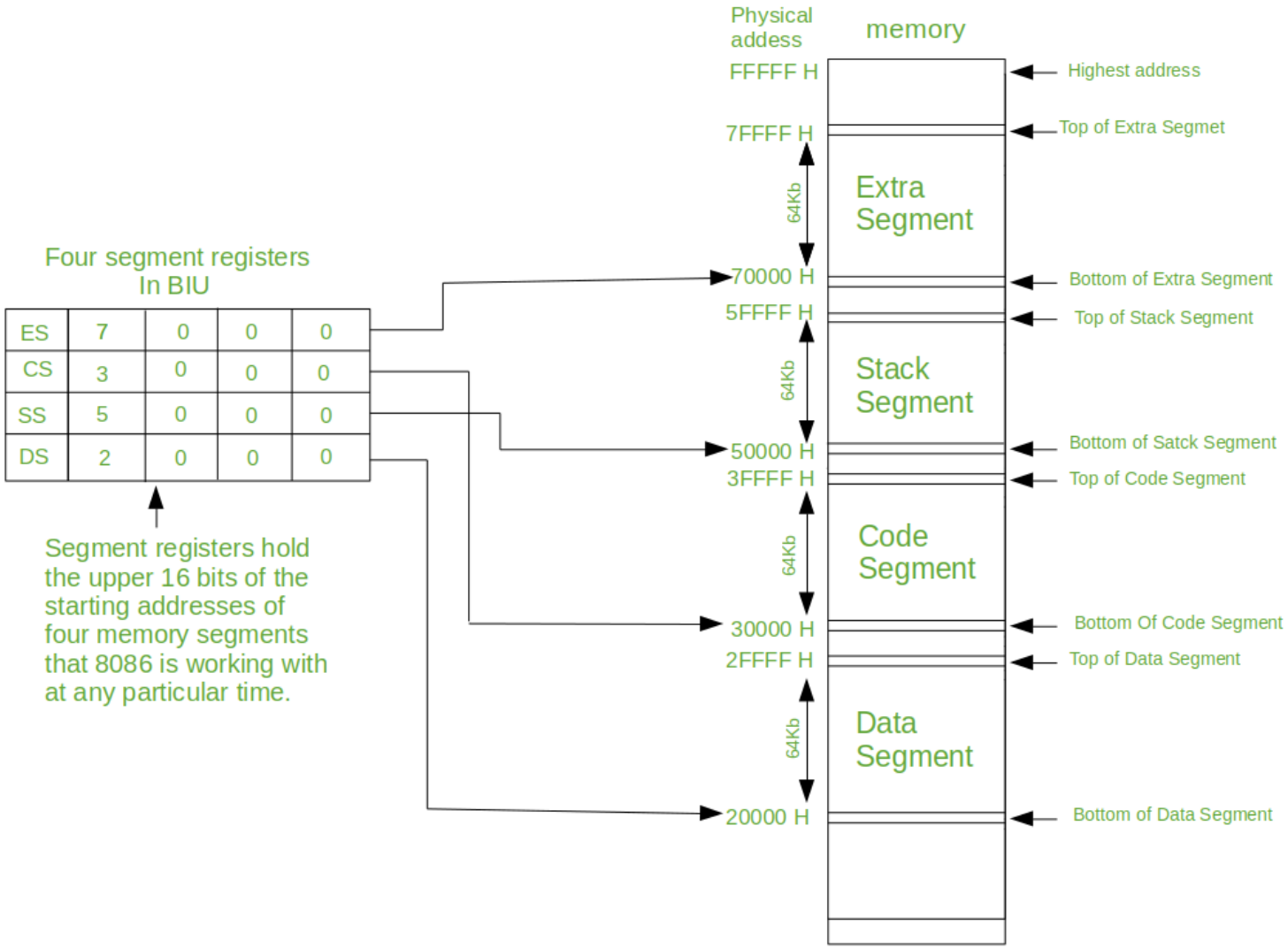
Introduction to Program Segments

- ***Segment***
  - A ***segment*** is an area of memory that includes up to 64K bytes and begins an address evenly divisible by 16 (such an address ends in 0H).
  - Assembly Language Program consists of three segments:
    - ***code segment*** : contains the program code (instructions)
    - ***data segment*** : used to store data to be processed by the program
    - ***stack segment:*** used to store information temporarily.

- ***Logical and Physical Address***
  - ***Physical Address*** is the 20-bit address that actually put on the address bus. (in 8086)
    - Has a range of 00000H - FFFFFH
  - ***Segment Address*** is a 16-bit address of the segment block.
    Each segment is a ***block*** of 64 KB of memory space.
  - ***Offset Address*** is a location ***within*** 64K byte segment range.
    Has a range of 0000H - FFFFH
  - ***Logical Address*** consists of ***segment address*** and ***offset address***.

Introduction to Program Segments

- ***Memory Segmentation in 8086 Microprocessor***



Introduction to Program Segments

- ***Addressing in Code Segment***

- To execute a program, the 8086 fetches the instructions from the ***code segment***.
- The ***logical address*** of an instruction consists **CS** (Code Segment) and **IP**(instruction pointer).
- ***Logical Address*** **in Code segment** is represented by using **segment address in CS** register and **Offset Address in IP** register as follows:

CS:IP **(16 bit CS and 16 bit IP making total of 32 bits)**

Example: If CS register contains 2500H and IP register contains 95F3H. What is the ***Locical Adress*** in the code segment?

CS:IP → **2500:95F3** **(default in adressing is hex. You don't need H)**

Introduction to Program Segments

- ***Addressing in Code Segment***

Physical Address is generated by shifting the CS one hex digit to the left and adding IP. Physical address is ***20 bit address*** which can be generated by using a logical address as follows.

1. **Start with CS**
2. **Shift left CS (insert 0 as the Least significant digit)**
3. **Add IP**

Example: If CS register contains 1980H and IP register contains 78FEH. What is the ***Physical Adress*** in the code segment?

Logical address: CS:IP → **1980:78FE**

1.	**Start with CS**	1980	
2.	**Shift left CS**	19800	
3.	**Add IP**	78FE	(19800+ 78FE =210FE)

Physical address: The microprocessor will retrieve the instruction from the memory locations starting from **210FE** ***(20 bit address).***

Introduction to Program Segments

- ***Addressing in Code Segment***

Example: If CS=24F6H and IP=634AH, determine:

a) The logical address
b) The offset address
c) The physical address
d) The lower range of the code segment
e) The upper range of the code segment

Solution:

a) The logical address is; **24F6:634A**
b) The offset address is; **634A**
c) The Physical address is; **24F60+634A= 2B2AA**
d) The lower range of the code segment: **24F6:0000 → 24F60+0000 = 24F60**
e) The upper range of the code segment: **24F6:FFFF → 24F60+FFFF = 34F5F**

Introduction to Program Segments

- ***Addressing in Data Segment***

- The area of memory allocated strictly for data is called ***data segment.***

- ***Data segment*** contains ***variables*** containing single values and arrays of values, where ***code segment*** only contain program ***instructions***.

- ***Logical Address*** **in Data Segment** is represented by using **segment address in DS** register and **Offset Address in BX, SI or DI** registers.

DS:BX
DS:SI
DS:DI

- At any time ***three locations*** in the data segment are pointed with ***DS:BX***, ***DS:SI*** and ***DS:DI*** respectively.

Introduction to Program Segments

- ***Addressing in Data Segment***

Example: If DS=7FA2H and the offset is 438EH, determine:

a) The physical address
b) The lower range of the data segment
c) The upper range of the data segment
d) Show the logical address

Solution:

a) The Physical address is; **7FA20+438E = 83DAE**
b) The lower range: **7FA20+0000= 7FA20**
c) The upper range: **7FA20+FFFF = 8FA1F**
d) The logical address is; **7FA2:438E**

Introduction to Program Segments

- ***Addressing in Data Segment***

Why do we use data segment?

Assume that a program is needed to add 5 bytes of data (25H, 12H, 15H,1FH and 2BH)

One way:

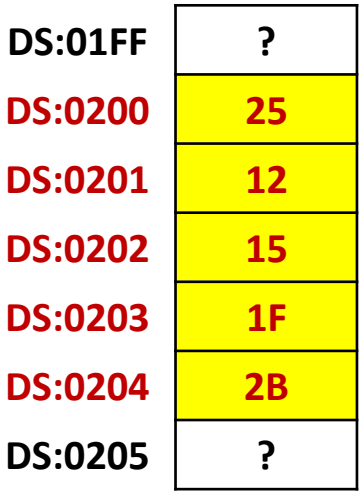
```
MOV  AL,00H        ;initialize AL
ADD  AL,25H
ADD AL,12H
ADD AL,15H
ADD AL,1FH
ADD AL,2BH         ; AL=25+12+15+1F+2B
```

code and data are mixed
(bad programming practice)

Better way: Assume that the Data segment contains the array of bytes starting from **offset address 0200H**.

```
MOV  AL,0 ;clear AL
ADD AL,[0200];add the contents of DS:200 to AL
ADD AL,[0201];add the contents of DS:201 to AL
ADD AL,[0202];add the contents of DS:202 to AL
ADD AL,[0203];add the contents of DS:203 to AL
ADD AL,[0204];add the contents of DS:204 to AL
```

code and data are separated
(good programming practice)



Introduction to Program Segments

- ***Little endian convention***

Given 8-bit (1-byte) data, bytes are stored one after the other in the memory. However given16-bit (2-bytes) of data how are date stored?

Example:

```
MOV AX,35F3H     ;load 35F3H into AX
MOV [1500],AX    ; copy contents of AX to offset 1500H
```

In such a case the low byte goes to the low memory location and high byte goes to the high memory location.

DS:1500 = F3 DS:1501 = 35

This convention is called ***little endian convention***: This convention is used by ***Intel***.

Big endian convention is the opposite, where the high byte goes to the low address and low byte goes to the high address. ***Motorolla*** microprocessor uses this convention.