University of Diyala
College of engineering
Department of computer Engineering
Second class

# microprocessor Programming

## *Lecture 3*

## *8086 Internal Architecture*

*Presented by*
*Lecturer :Abdullah Thair Abdalsatir*
*Department of computer Engineering*
*University of Diyala*

# Lecture 3
# 8086 microprocessor

- Internal Architecture (etc..)
- Execution Unit (EU):-
  - Accumulator Register (AX)
  - Base Register (BX)
  - Counter Register (CX)
  - Data register (DX)
  - Stack Pointer (SP) and Base Pointer (BP)
  - Source Index (SI) and Destination Index (DI)
  - Flag register
  - 8086 categorized into4 gropes

**EU decodes and executes instructions.**

**A decoder in the EU control system translates instructions.**

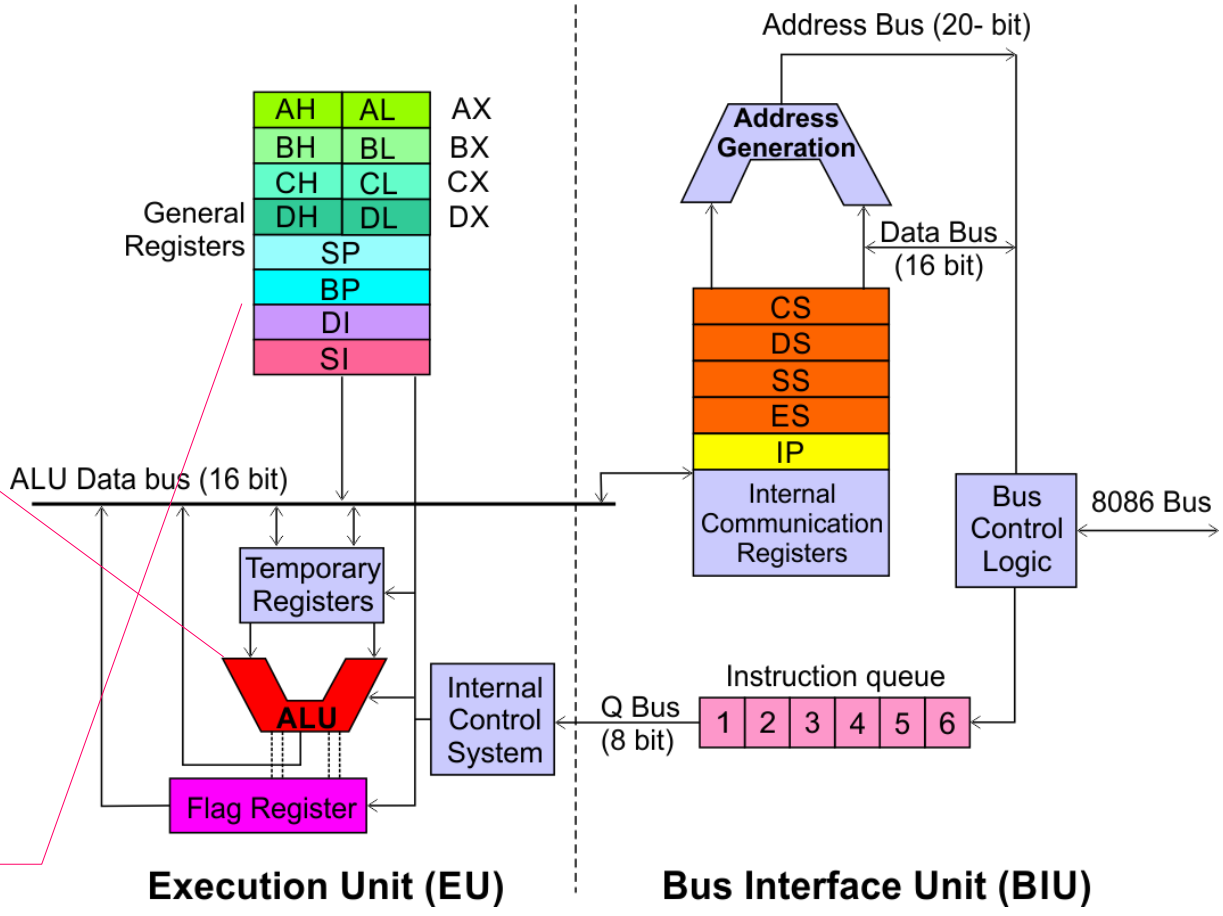
**16-bit ALU for performing arithmetic and logic operation**

**Four general purpose registers (AX, BX, CX, DX);**

**Pointer registers (Stack Pointer, Base Pointer);**

**and**

**Index registers (Source Index, Destination Index) each of 16-bits**



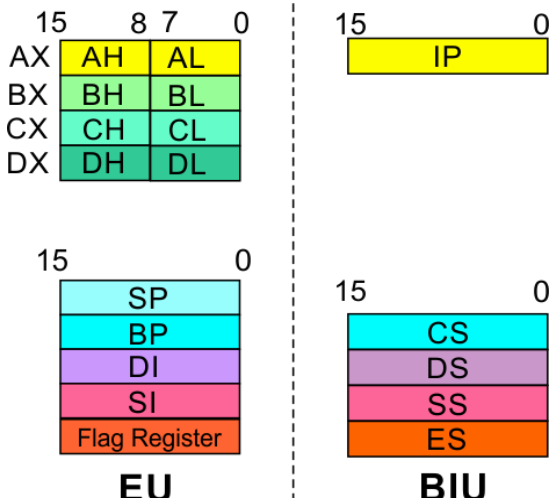

**Some of the 16-bit registers can be used as two 8 bit registers as:**

**AX can be used as AH and AL BX can be used as BH and BL CX can be used as CH and CL DX can be used as DH and DL**

EU Registers

## Accumulator Register (AX)

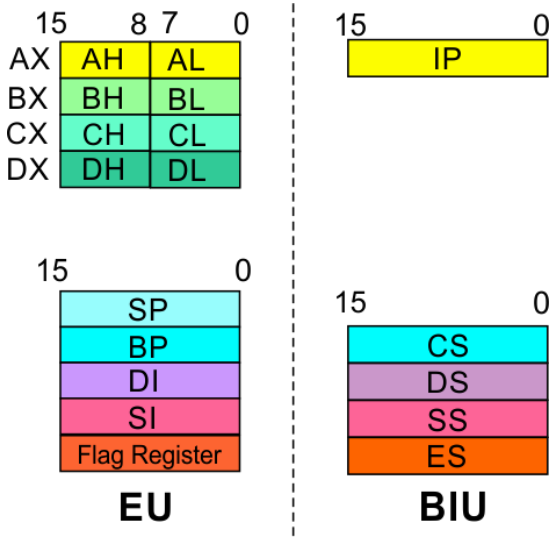
- Consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
- AL in this case contains the low order byte of the word, and AH contains the high-order byte.
- The I/O instructions use the AX or AL for inputting / outputting 16 or 8-bit data to or from an I/O port.
- Multiplication and Division instructions also use the AX or AL.



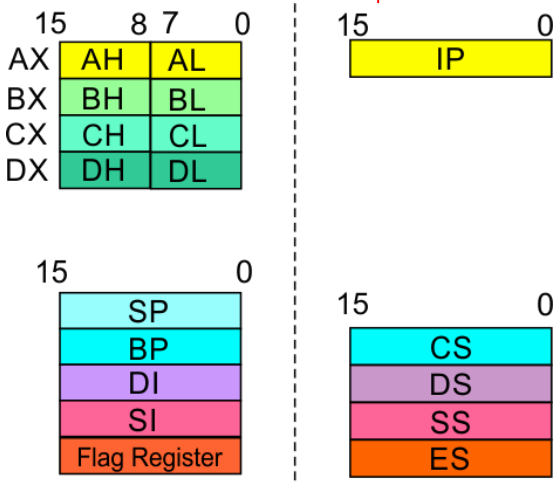
EU Registers

## Base Register (BX)

- Consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BL in this case contains the low-order byte of the word, and BH contains the high-order byte.
- This is the only general purpose register whose contents can be used for addressing the 8086 memory.
- All memory references utilizing this register content for addressing use DS as the default segment register.



EU Registers




## Counter Register (CX)

- Consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX.
- When combined, CL register contains the low order byte of the word, and CH contains the high-order byte.
- Instructions such as **SHIFT**, **ROTATE** and **LOOP** use the contents of CX as a counter.

**Example:**

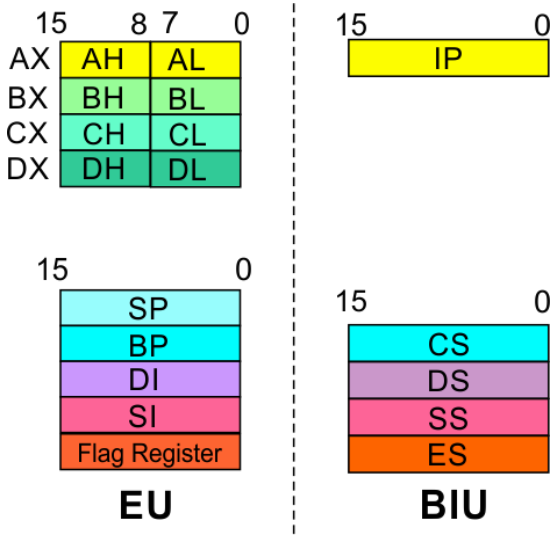
The instruction **LOOP START** automatically decrements CX by 1 without affecting flags and will check if [CX] = 0.

If it is zero, 8086 executes the next instruction; otherwise the 8086 branches to the label START.

EU Registers

## Data Register (DX)

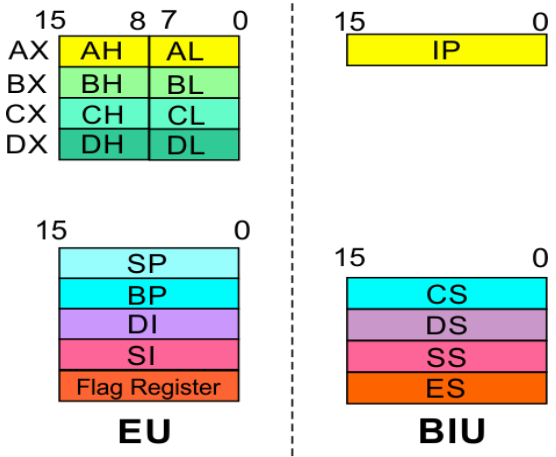
- Consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- When combined, DL register contains the low order byte of the word, and DH contains the high-order byte.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.



EU Registers

## Stack Pointer (SP) and Base Pointer (BP)

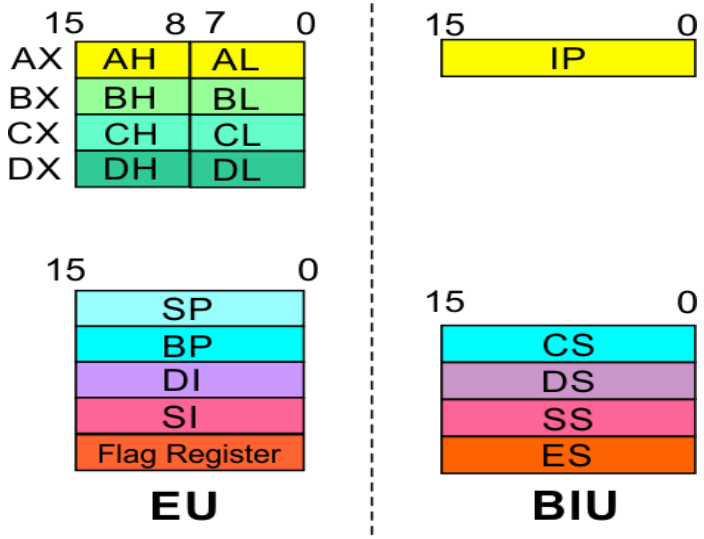
- SP and BP are used to access data in the stack segment.
- SP is used as an offset from the current SS during execution of instructions that involve the stack segment in the external memory.
- SP contents are automatically updated (incremented/ decremented) due to execution of a POP or PUSH instruction.
- BP contains an offset address in the current SS, which is used by instructions utilizing the based addressing mode.



EU Registers

## Source Index (SI) and Destination Index (DI)

- Used in indexed addressing.
- Instructions that process data strings use the SI and DI registers together with DS and ES respectively in order to distinguish between the source and destination addresses.



## Flag Register

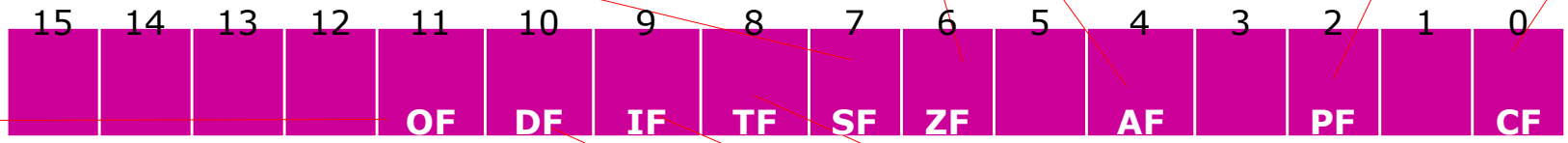
**Auxiliary carry**
This is set, if there is a carry from the lowest nibble, i.e, big three during addition, or borrow for the lowest nibble, i.e, big three, during subtraction.

**Carry Flag**
This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

**Sign Flag**
This flag is set, when the result of any computation is negative

**Zero Flag**
This flag is set, if the result of the computation or comparison performed by an instruction is zero

**Parity Flag**
This flag is set to 1, if the lower byte of the result contains even number of 1's ; for odd number of 1's set to zero.




**Over flow Flag**
This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

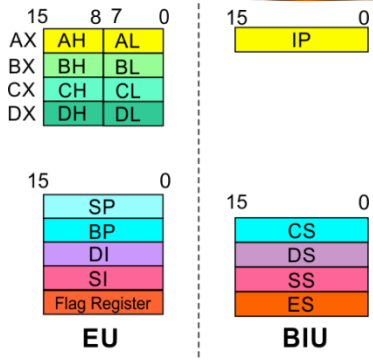
**Tarp Flag**
If this flag is set, the processor enters the single step execution mode by generating internal interrupts after the execution of each instruction

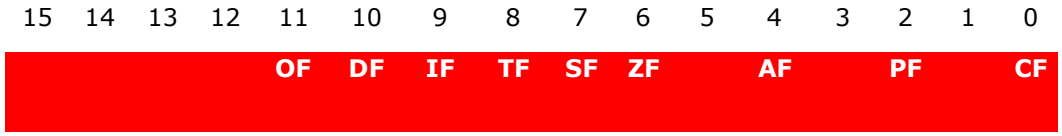
**Direction Flag**
This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.

**Interrupt Flag**
Causes the 8086 to recognize external mask interrupts; clearing IF disables these interrupts.

# Architecture

**8086 registers categorized into 4 groups**




| Sl.No. | Type | Register width | Name of register |
|---|---|---|---|
| 1 | General purpose register | 16 bit | AX, BX, CX, DX |
| | | 8 bit | AL, AH, BL, BH, CL, CH, DL, DH |
| 2 | Pointer register | 16 bit | SP, BP |
| 3 | Index register | 16 bit | SI, DI |
| 4 | Instruction Pointer | 16 bit | IP |
| 5 | Segment register | 16 bit | CS, DS, SS, ES |
| 6 | Flag (PSW) | 16 bit | Flag register |

| Register | Name of the Register | Special Function |
|---|---|---|
| AX | 16-bit Accumulator | Stores the 16-bit results of arithmetic and logic operations |
| AL | 8-bit Accumulator | Stores the 8-bit results of arithmetic and logic operations |
| BX | Base register | Used to hold base value in base addressing mode to access memory data |
| CX | Count Register | Used to hold the count value in SHIFT, ROTATE and LOOP instructions |
| DX | Data Register | Used to hold data for multiplication and division operations |
| SP | Stack Pointer | Used to hold the offset address of top stack memory |
| BP | Base Pointer | Used to hold the base value in base addressing using SS register to access data from stack memory |
| SI | Source Index | Used to hold index value of source operand (data) for string instructions |
| DI | Data Index | Used to hold the index value of destination operand (data) for string operations |

*Thank you so much*

*Any questions?*