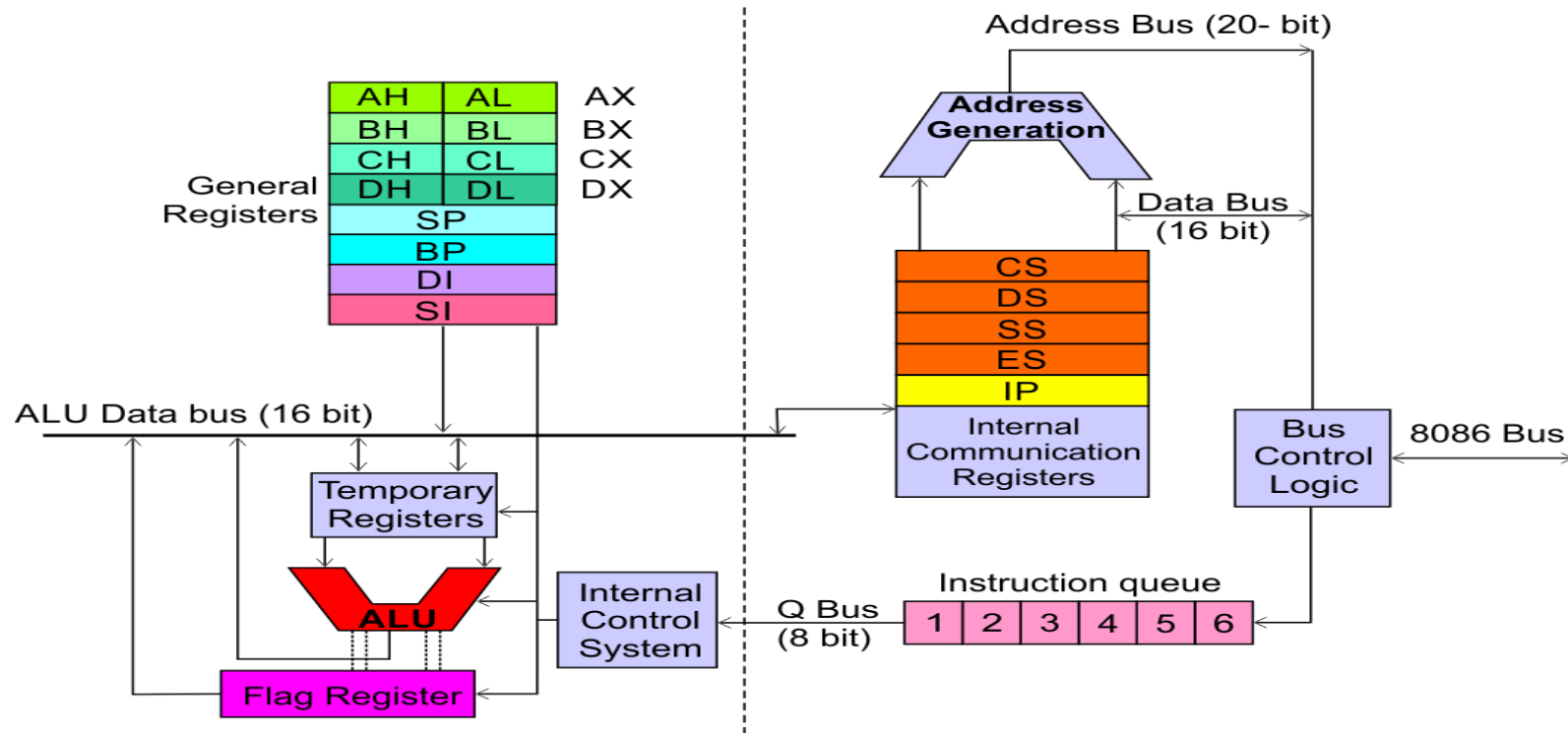


## 8086 Architecture

Prepared by  
 Subhasish Das  
 Assistant Professor  
 Electrical Engineering Department  
 B.P. Poddar Institute Of Management And Technology

# Architecture



## Execution Unit (EU)

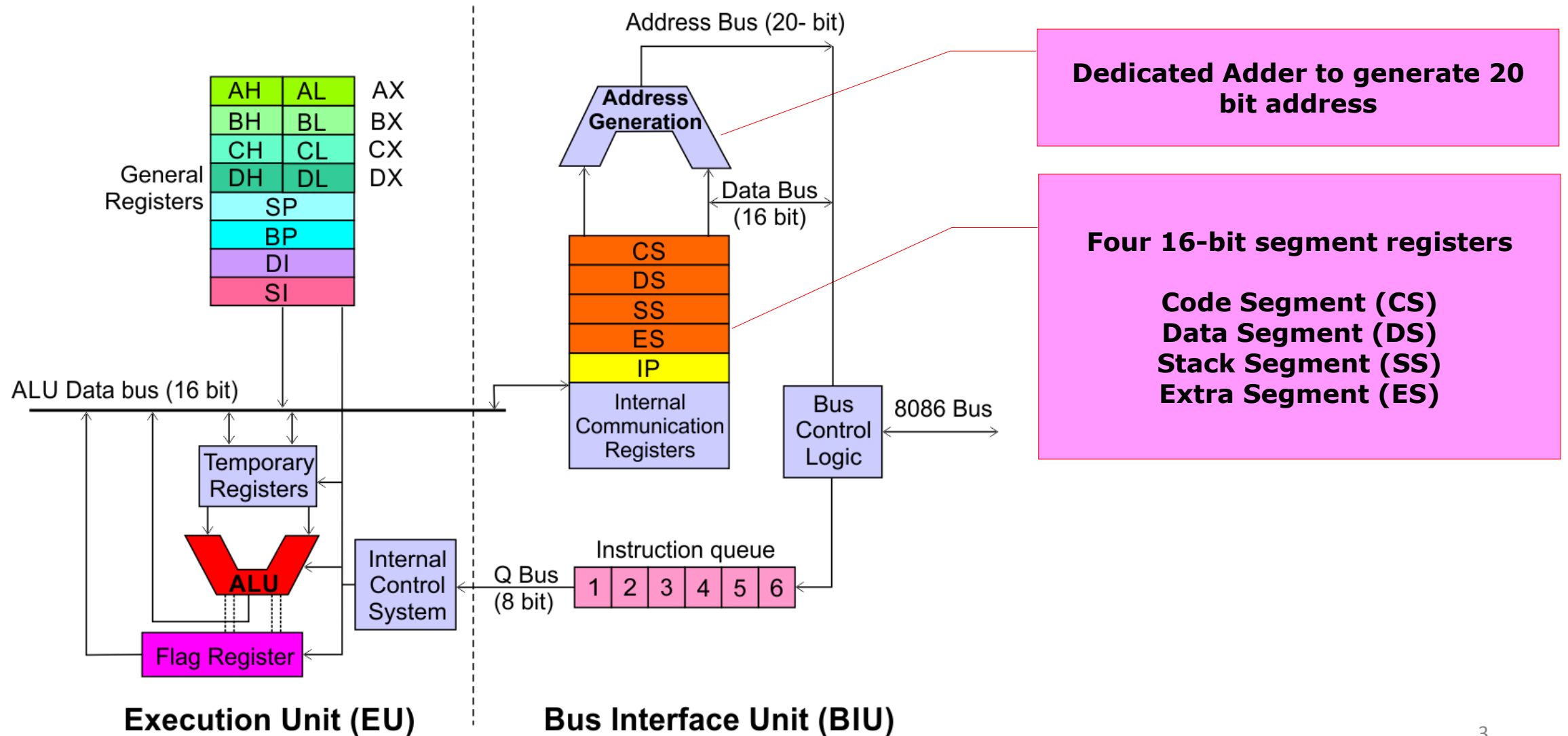
EU executes instructions that have already been fetched by the BIU.

BIU and EU functions separately.

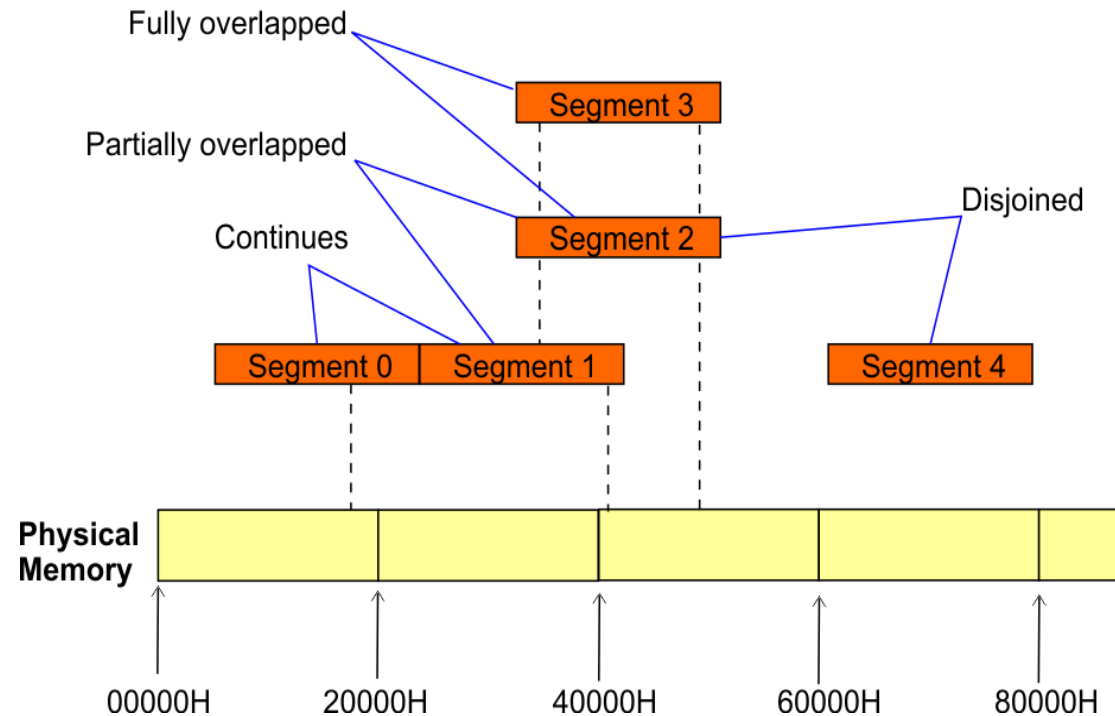
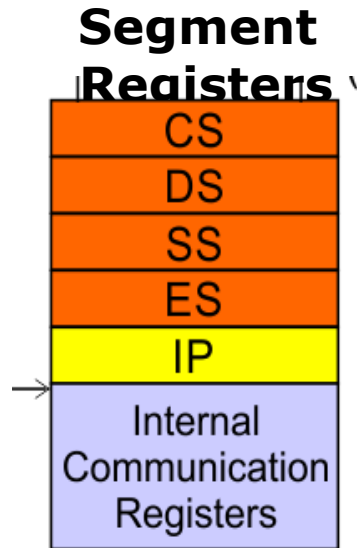
## Bus Interface Unit (BIU)

BIU fetches instructions, reads data from memory and I/O ports, writes data to memory and I/O ports.

# Architecture



# Bus Interface Unit (BIU)

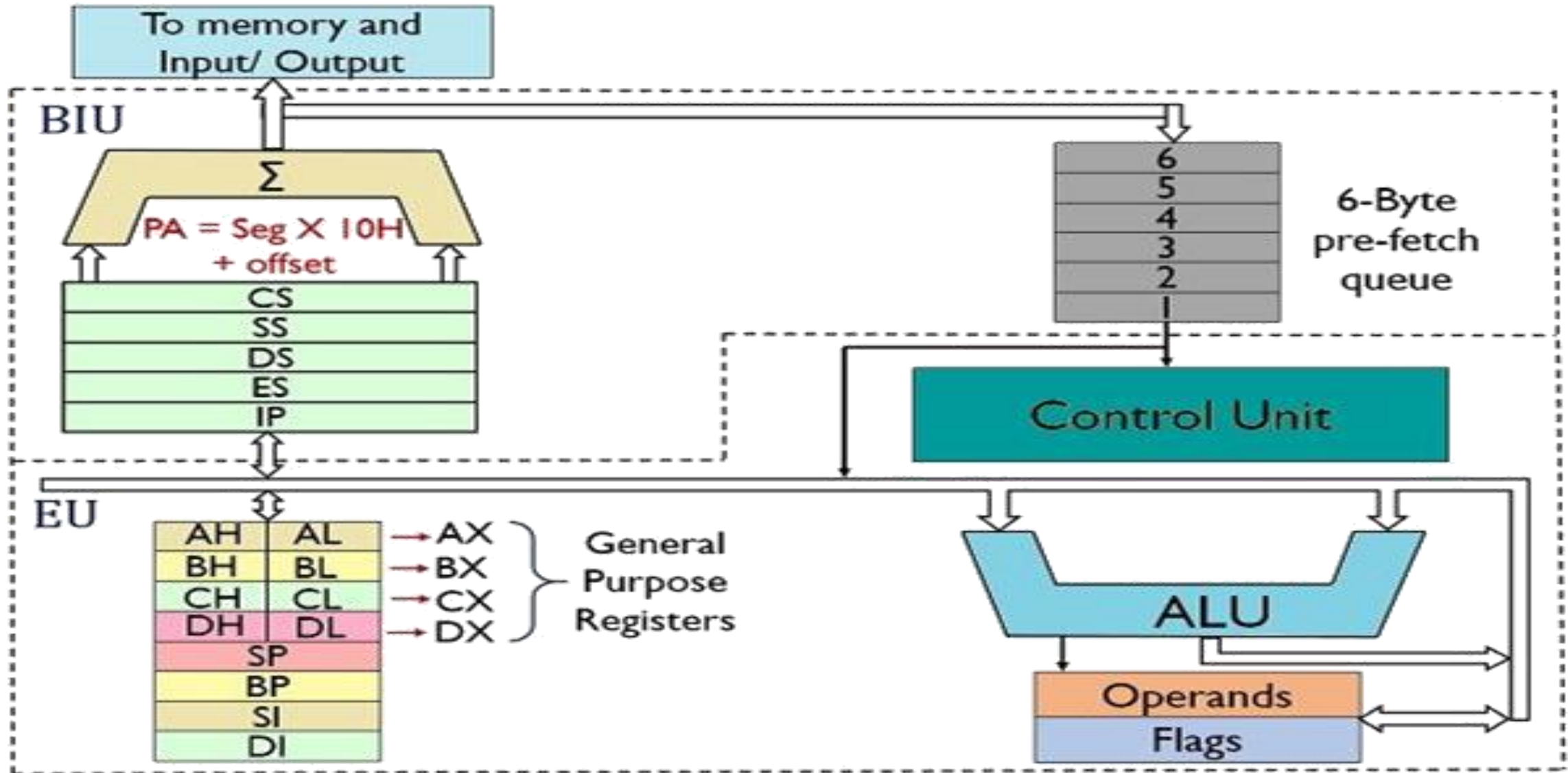


- 8086's 1-megabyte memory is divided into segments of up to 64K bytes each.

- The 8086 can directly address four segments (256 K bytes within the 1 M byte of memory) at a particular time.

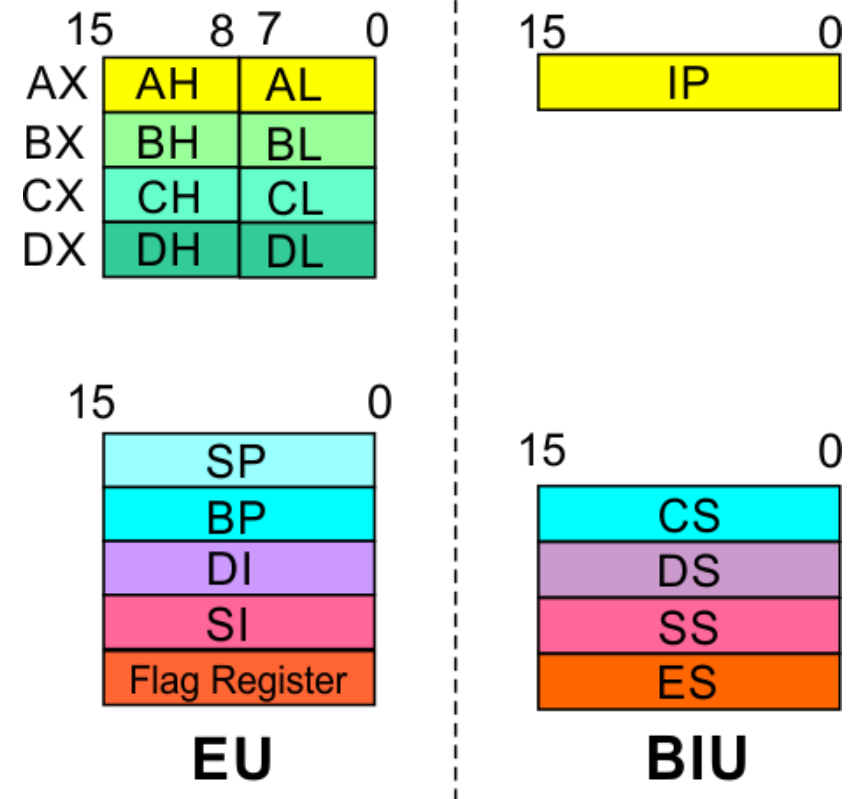
- Programs obtain access to code and data in the segments by changing the segment register content to point to the desired segments.

# 8086 block diagram



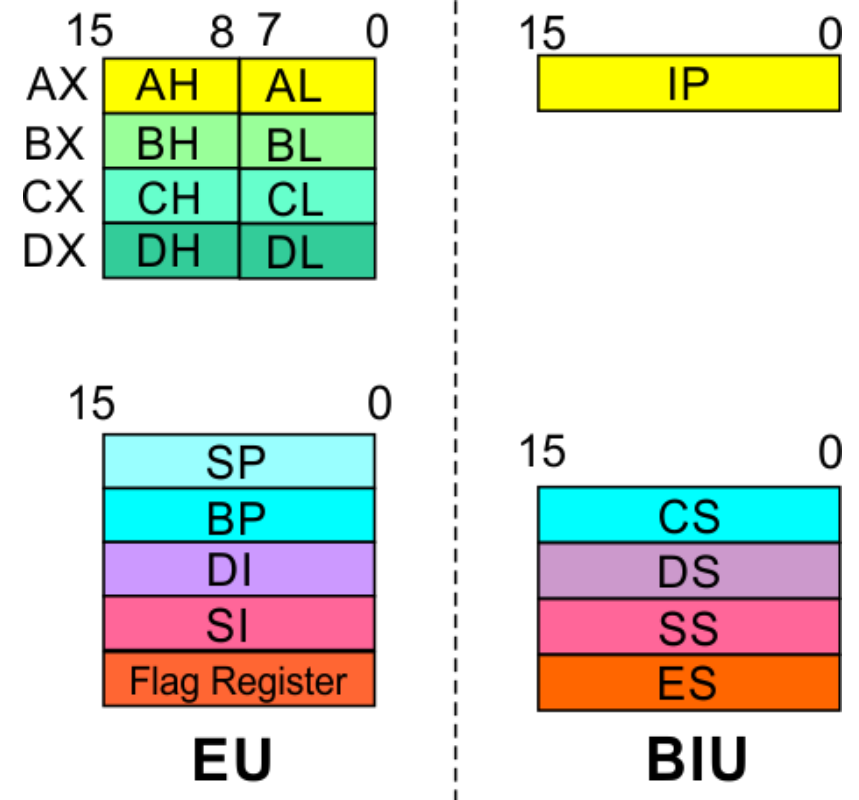
# Code Segment Register

- ❖ 16-bit
- ❖ CS contains the base or start of the current code segment; IP contains the distance or offset from this address to the next instruction byte to be fetched.
- ❖ BIU computes the 20-bit physical address by logically shifting the contents of CS 4-bits to the left and then adding the 16-bit contents of IP.
- ❖ That is, all instructions of a program are relative to the contents of the CS register multiplied by 16 and then offset is added provided by the IP.



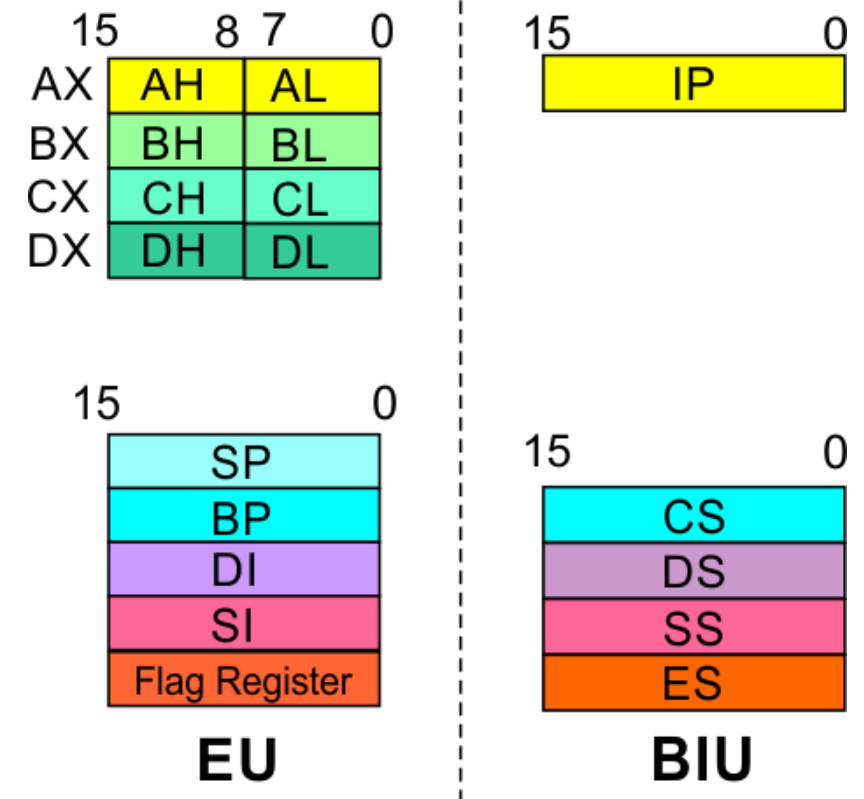
# Data Segment Register

- ❖ 16-bit
- ❖ Points to the current data segment; operands for most instructions are fetched from this segment.
- ❖ The 16-bit contents of the Source Index (SI) or Destination Index (DI) or a 16-bit displacement are used as offset for computing the 20-bit physical address.



# Stack Segment Register

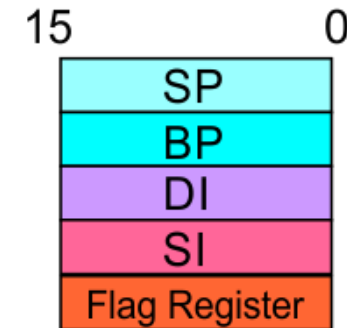
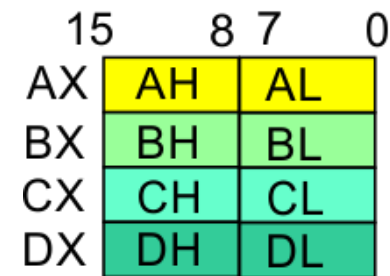
- ❖ 16-bit
- ❖ Points to the current stack.
- ❖ The 20-bit physical stack address is calculated from the Stack Segment (SS) and the Stack Pointer (SP) for stack instructions such as PUSH and POP.
- ❖ In based addressing mode, the 20-bit physical stack address is calculated from the Stack segment (SS) and the Base Pointer (BP).



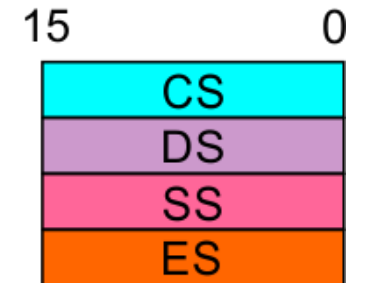


# Extra Segment Register

- ❖ 16-bit
- ❖ Points to the extra segment in which data (in excess of 64K pointed to by the DS) is stored.
- ❖ String instructions use the ES and DI to determine the 20-bit physical address for the destination.



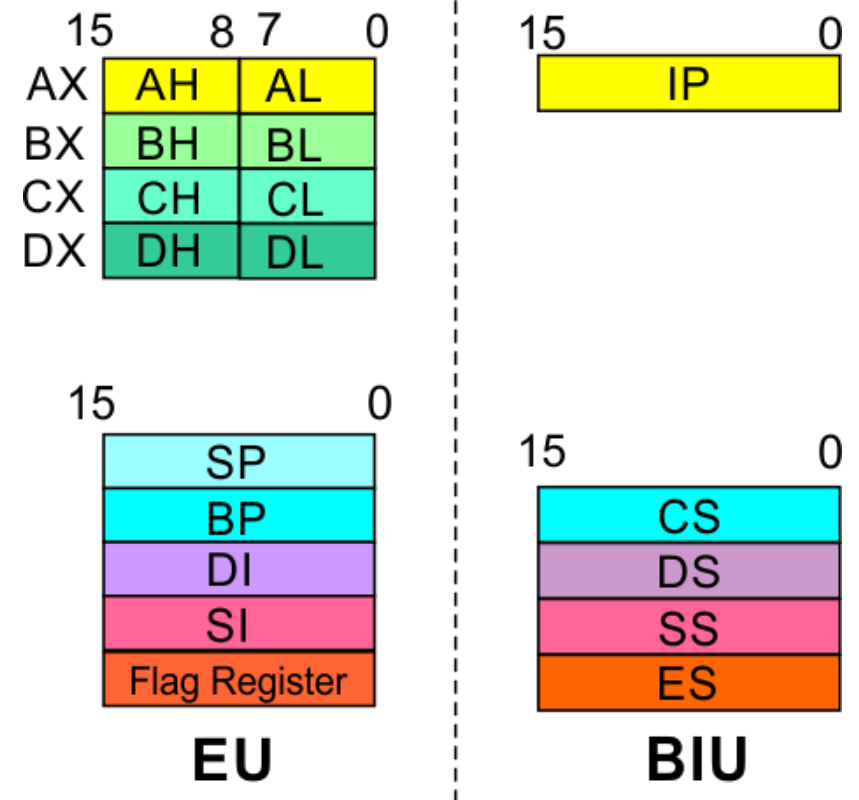
**EU**



**BIU**

# Instruction Pointer

- ❖ 16-bit
- ❖ Always points to the next instruction to be executed within the currently executing code segment.
- ❖ So, this register contains the 16-bit offset address pointing to the next instruction code within the 64Kb of the code segment area.
- ❖ Its content is automatically incremented as the execution of the next instruction takes place.



# Execution Unit (EU)

**EU decodes and executes instructions.**

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

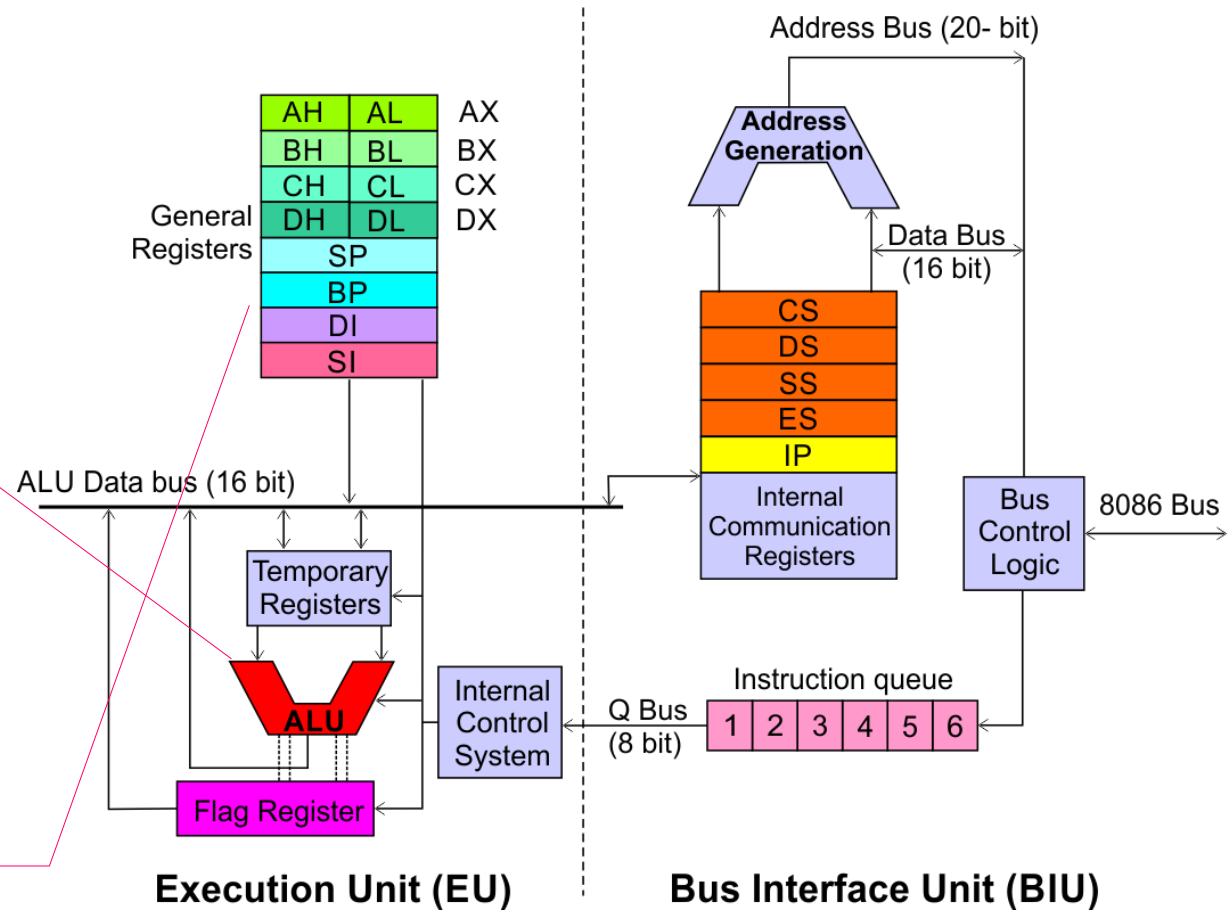
**16-bit ALU for performing arithmetic and logic operation**

**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**

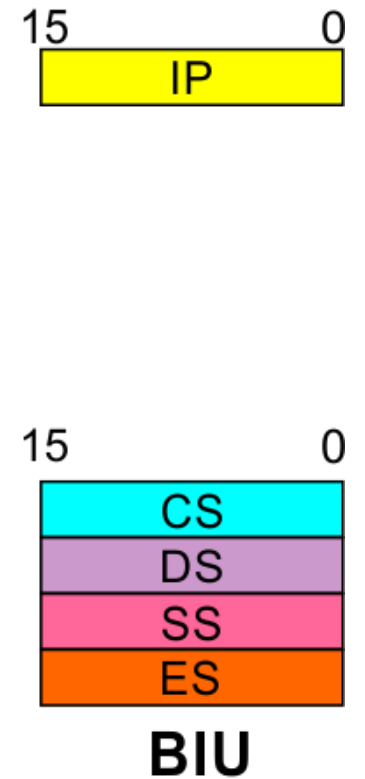
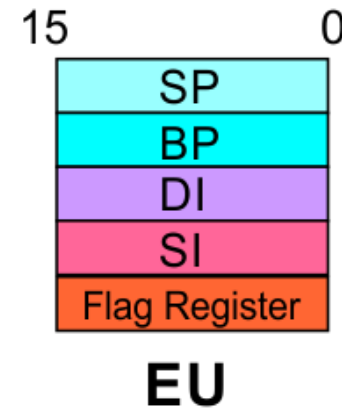
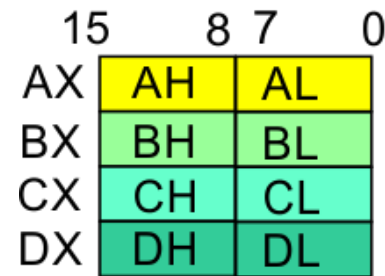


**Some of the 16 bit registers can be used as two 8 bit registers as :**

**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**

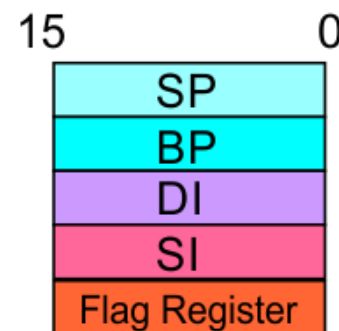
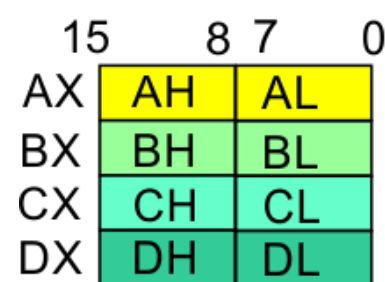
# Accumulator Register (AX)

- ❖ 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.

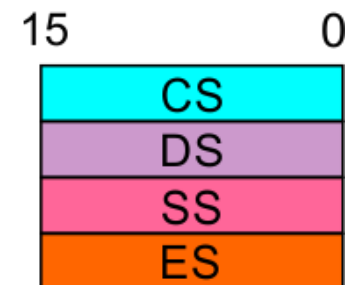


# Base Register (BX)

- ❖ 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**



**BIU**

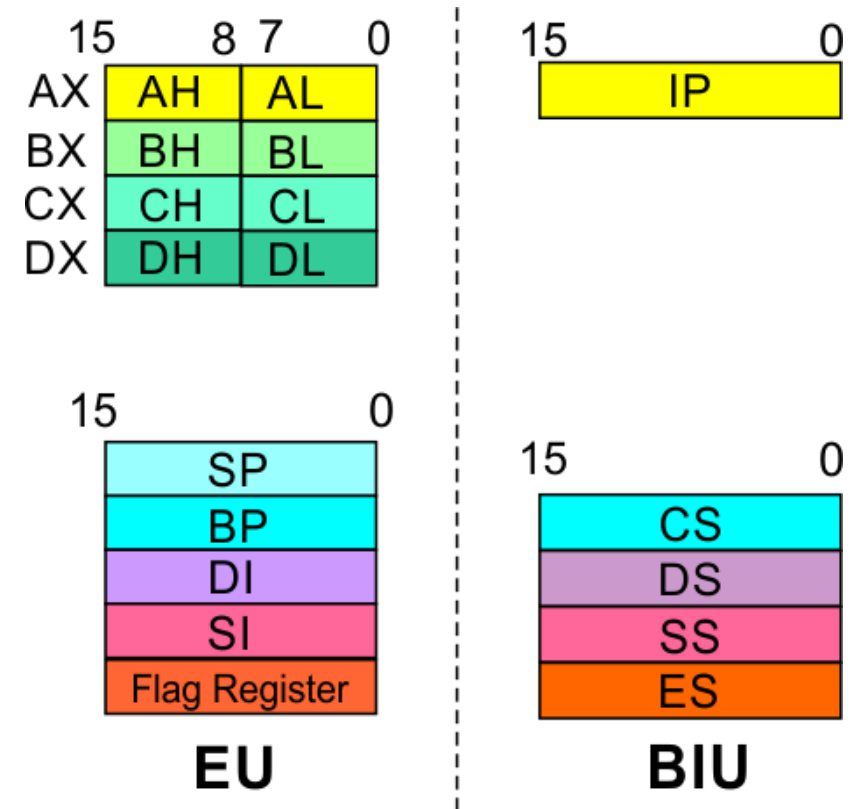
# Counter Register (CX)

- ❖ 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.



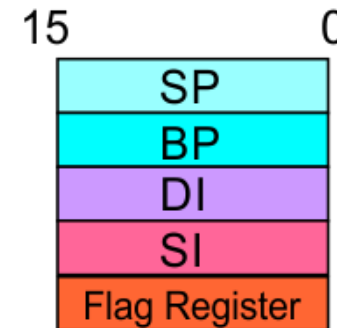
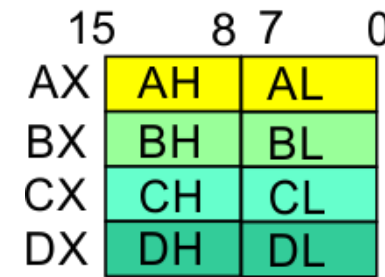
# Counter Register (CX)

- ❖ 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.



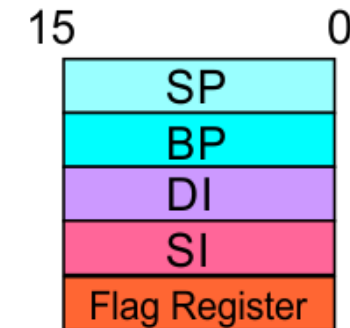
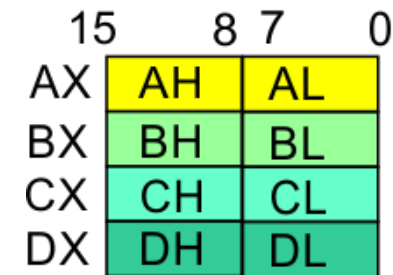
EU



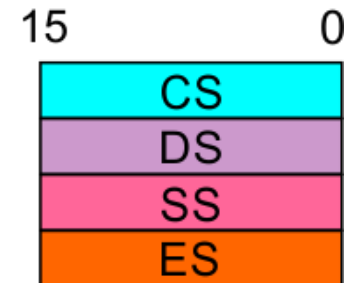
BIU

# 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 remainder after division.



**EU**

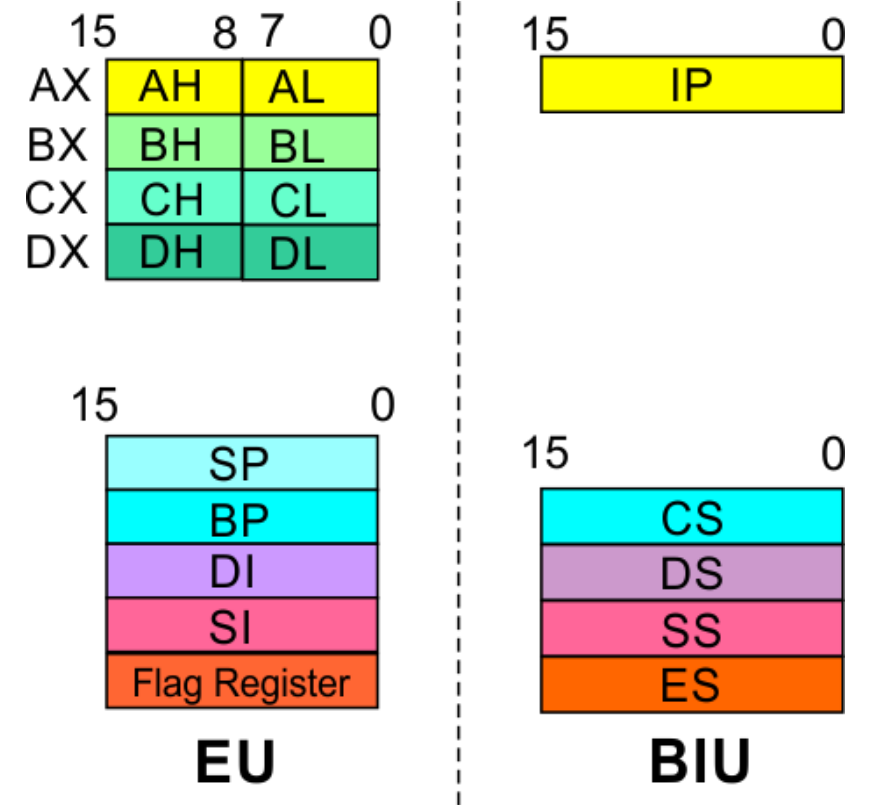


**BIU**



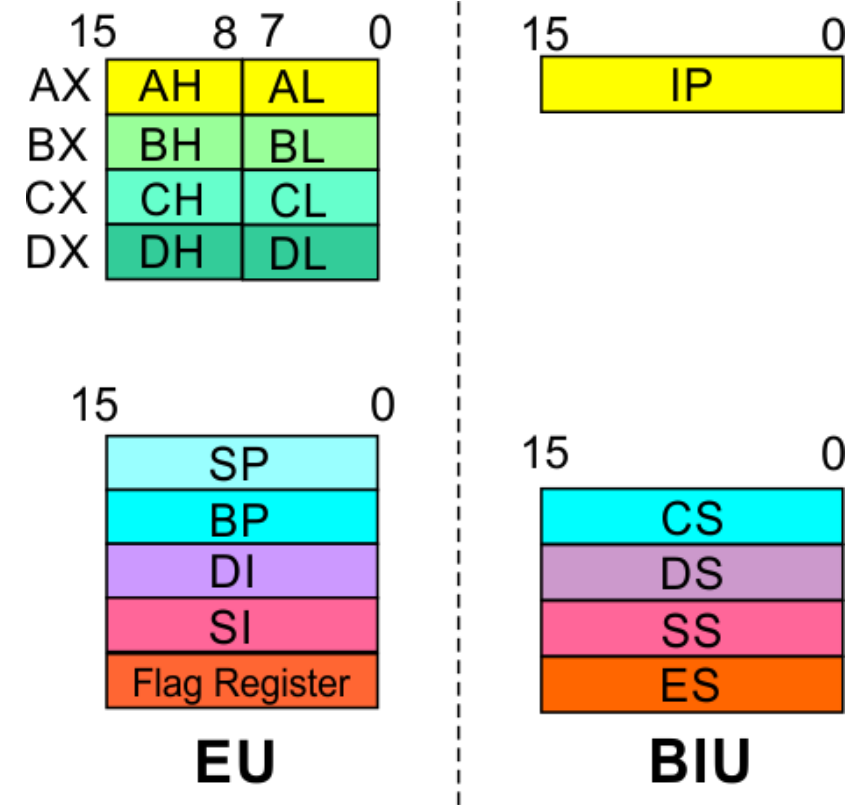
# Stack Pointer (SP) and Base Pointer (BP)

- ❖ 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.

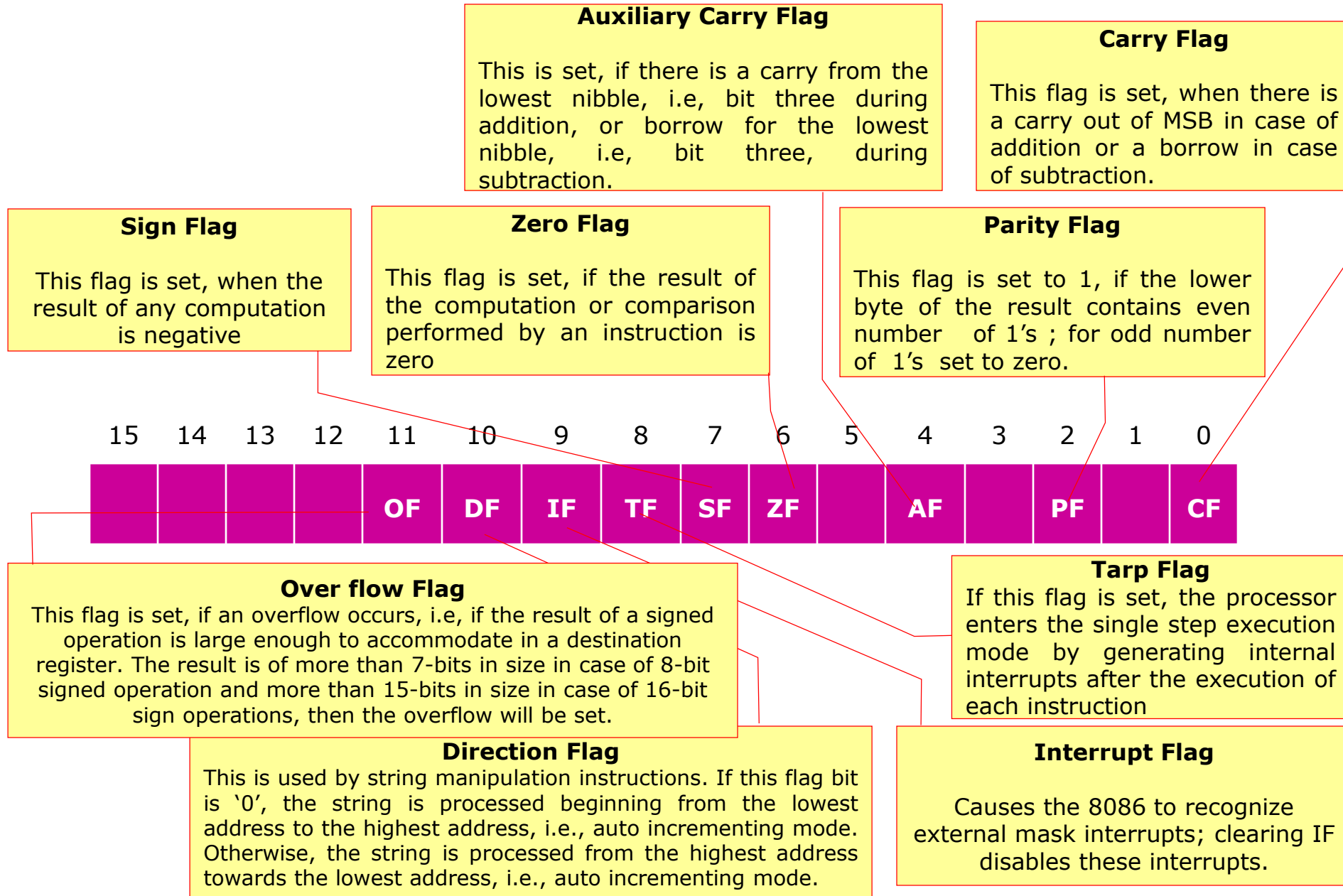


# Source Index (SI) and Destination Index (DI)

- ❖ 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.

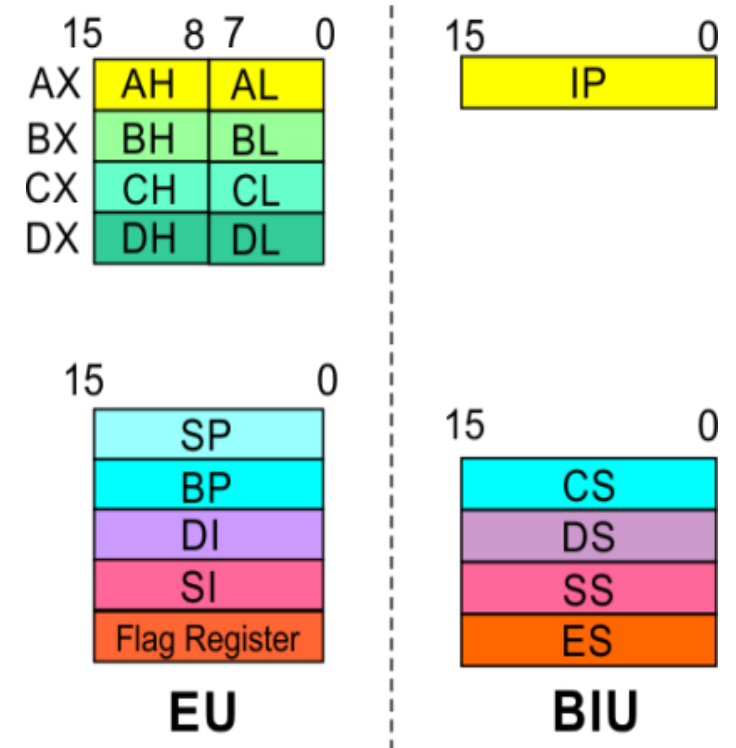


# Flag Register



# Different Register in 8086

S.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





Thank  
you!



# Contacts

Email id-

[subhasish.das@bppimt.ac.in](mailto:subhasish.das@bppimt.ac.in)

[subhasish.bppimt@gmail.com](mailto:subhasish.bppimt@gmail.com)

Mobile No.-

9038796596

