Architecture of 8086 Intel Microprocessor

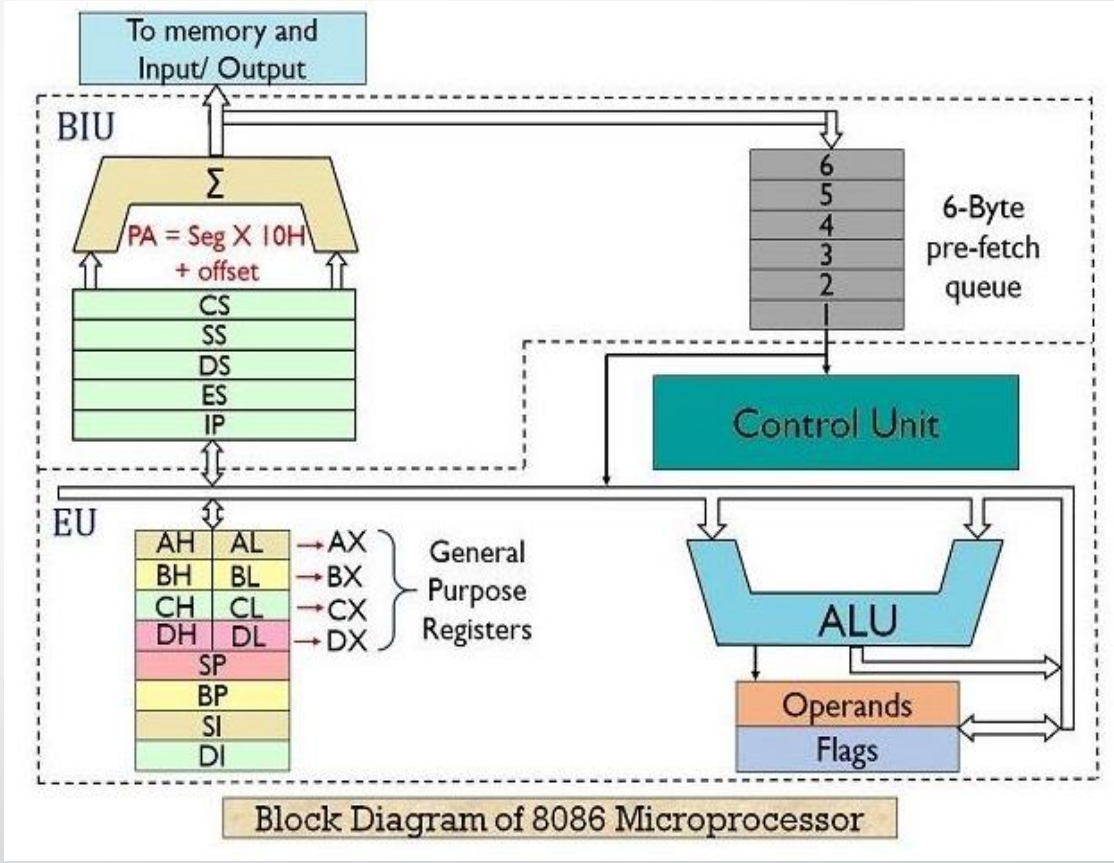
Lecture01- 10/2022

8086 microprocessor Internal Architecture

- The **8086 microprocessor** consists of two main blocks: the **Bus Interface Unit(BIU)** and the **Execution Unit (EU)**. All the components of the 8086 microprocessor are present within these two blocks.

- The BIU has a direct link with the memory. This memory can be directly accessed either by the segment registers, the Instruction Pointer (IP) or the Instruction Queue for fetching up the instructions.

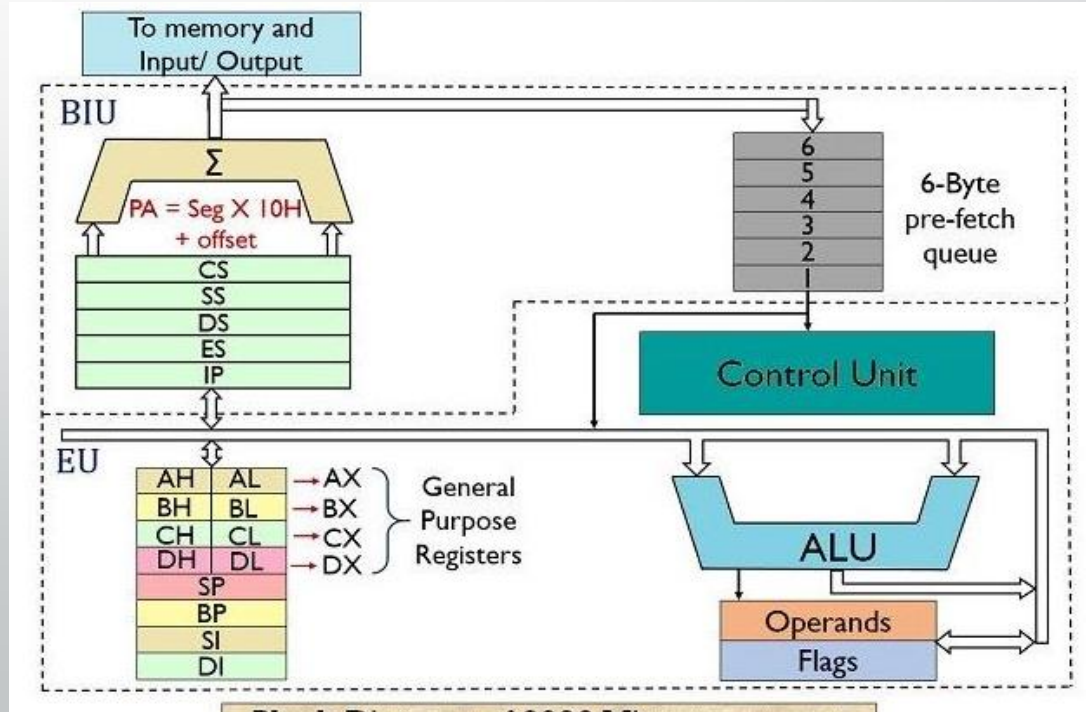
- These instructions are sent into the Control Unit for execution. The control unit takes the help of General Purpose registers, Index registers, Pointers, operands, flags and the ALU. All these are part of the Execution Unit.



Block Diagram of 8086 Microprocessor

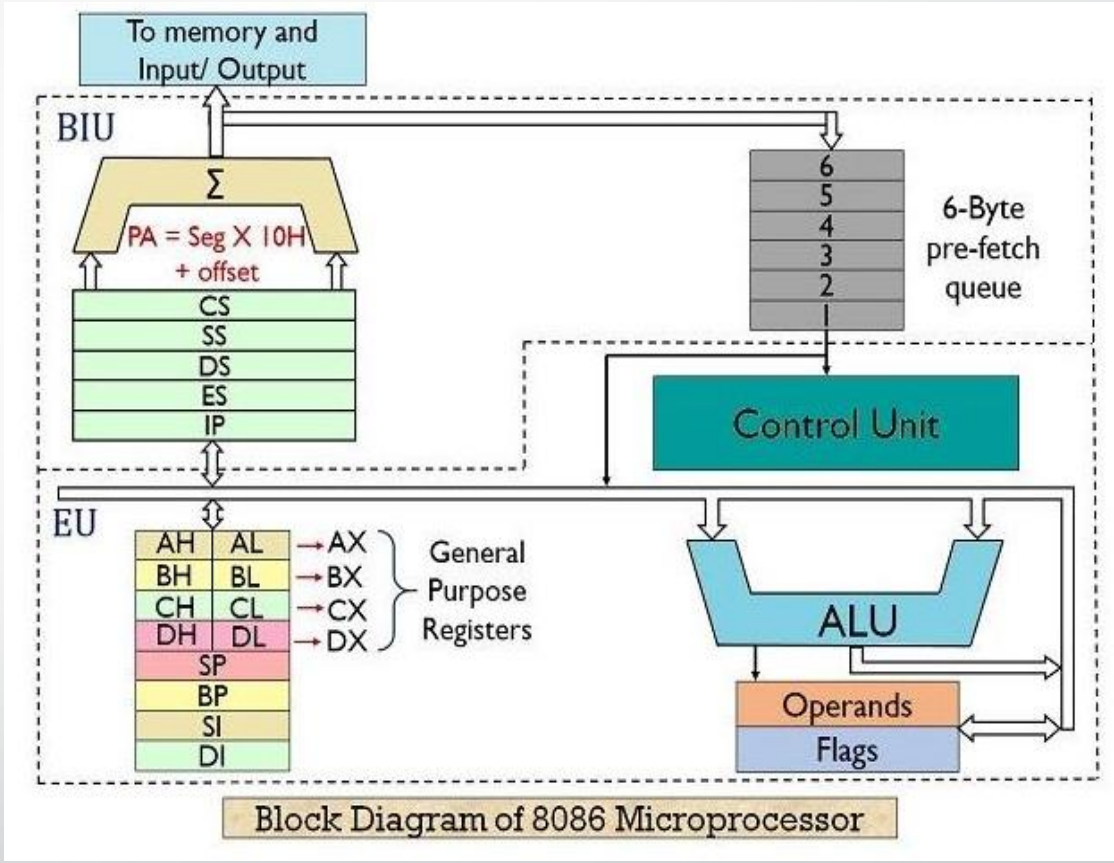
Bus Interface Unit (BIU)

- It generates the 20 bit physical address for memory access.
- It fetches instruction from memory.
- It transfers data to and from the memory and I/O.
- The Instruction Queue contains the set of instruction which is to be executed.
- The 8086 pre-fetches up to 6 (6-byte Instruction Queue) instructions in advance and stores them in the Instruction queue. It supports pipelining using the 6 byte instruction queue.



Bus Interface Unit (BIU)- Continue

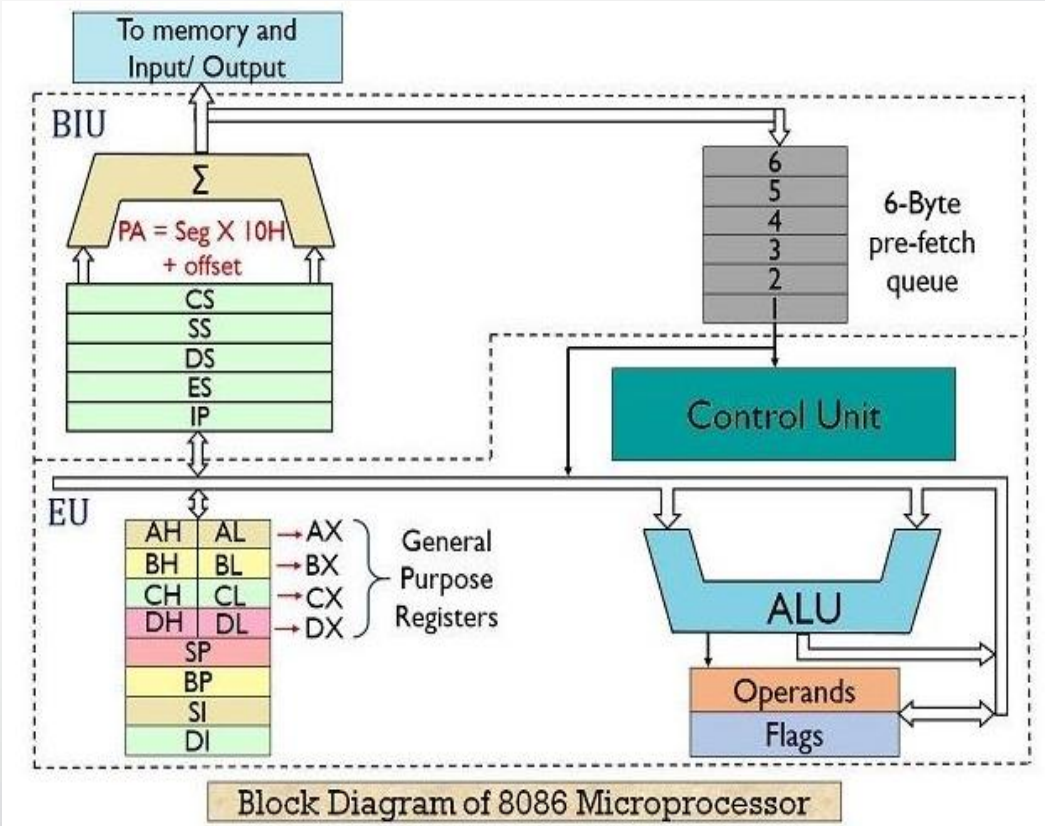
- **The main components of the BIU are as follows:**
- Segment registers: There are 4 segment registers :
  - Code Segment Register(CS): is used for addressing memory location in the code segment of the memory, where the executable program is stored.
  - Data Segment Register(DS): points to the data segment of the memory where the data is stored.
  - Extra Segment Register(ES) also refers to a segment in the memory which is another data segment in the memory.
  - Stack Segment Register(SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.



Block Diagram of 8086 Microprocessor

Bus Interface Unit (BIU)- Continue

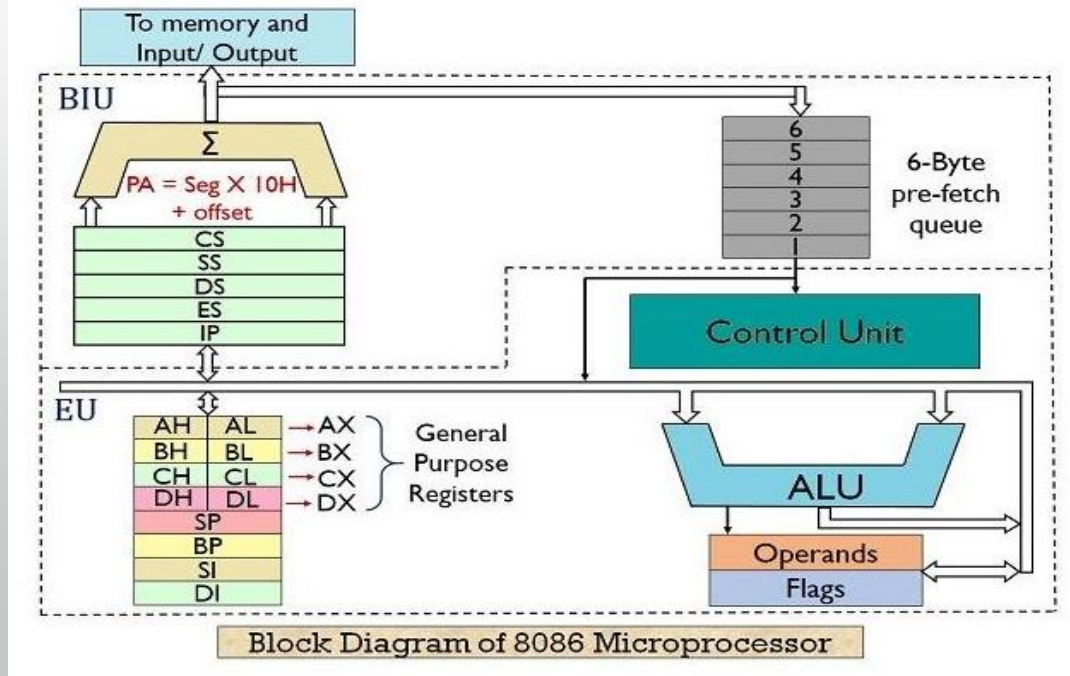
- **Instruction Pointer (IP):**
  - It is a 16 bit register. It holds offset of the next instructions in the Code Segment.
  - Address of the next instruction is calculated as CS x 10H + IP.
  - IP is incremented after every instruction byte is fetched.
- **Address Generation Circuit-**
  - The BIU has a Physical Address Generation Circuit. It generates the 20 bit physical address using Segment and Offset addresses using the formula:
  - Physical Address = Segment Address x 10H + Offset Address



Block Diagram of 8086 Microprocessor

Bus Interface Unit (BIU)- Continue

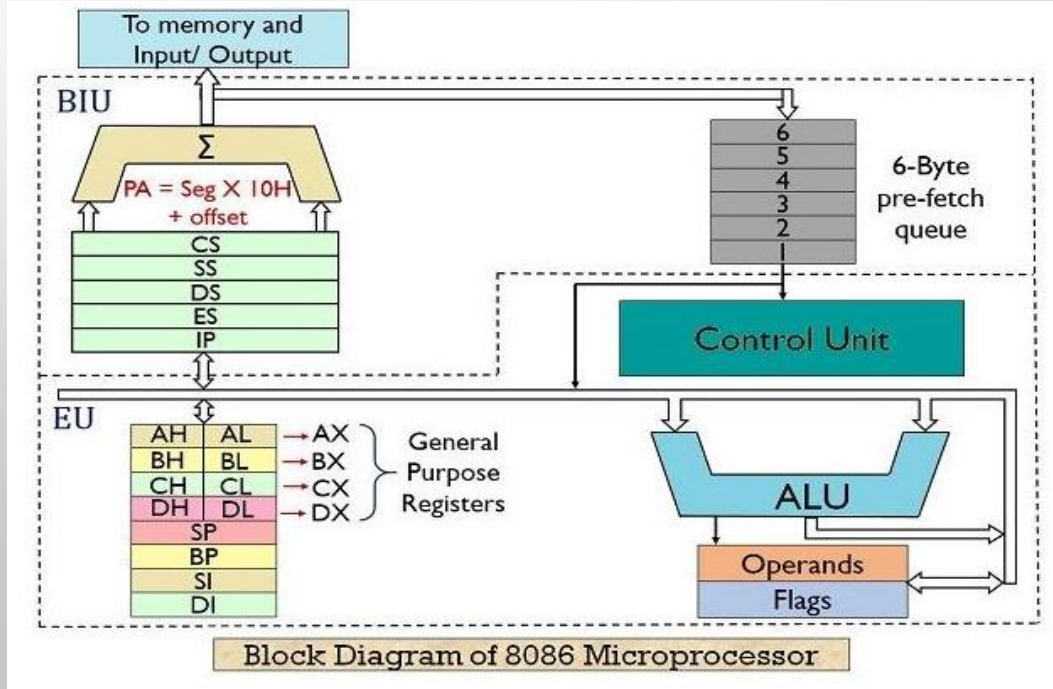
- **6 Byte Pre-fetch Queue:**
  - It is a 6 byte first in first out (FIFO) RAM used to implement pipelining. Fetching the next instruction while executing the current instruction is called pipelining.
  - BIU fetches the next six instruction bytes from the Code Segment and stores it into the queue.



Block Diagram of 8086 Microprocessor

Execution Unit (EU)-

- It fetches instructions from the Queue in BIU, decodes and executes them.
- It performs arithmetic, logic and internal data transfer operations within the microprocessor.
- It sends request signals to the BIU to access the external module.
- It operates with respect to clock cycles and does not depend upon which machine cycle is being performed by the BIU.

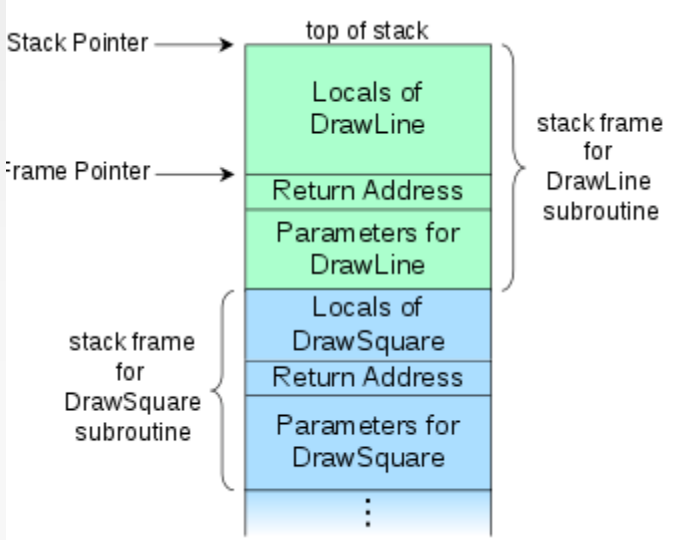


Block Diagram of 8086 Microprocessor

Execution Unit (EU)-

- **General purpose registers:**
  - 8086 microprocessor has four 16 bit general purpose registers AX, BX, CX and DX. These are available to the programmer for storing values during programs. Each of these can be divided into two 8 bit registers such as AH, Al; BH, BL; etc. Beside their general use, these registers also have some specific functions.
  - **AX register "Accumulator register" (16 bits):** It holds operands and results during multiplication and division operations. All I/O data transfers using IN and OUT instructions use A register (AL/AH or AX). It functions as accumulator during string operations.
  - **BX register "Base Register" (16 bits):** It holds the memory address (offset address) in indirect addressing modes.
  - **CX register "Counter Register" (16 bits):** It holds count for instructions like loop, rotate, shift and string operations.
  - **DX register "Data Register" (16 bits):** It is used with AX to hold 32 bit values during multiplication and division. It is used to hold the address of the I/O port in indirect I/O addressing mode.

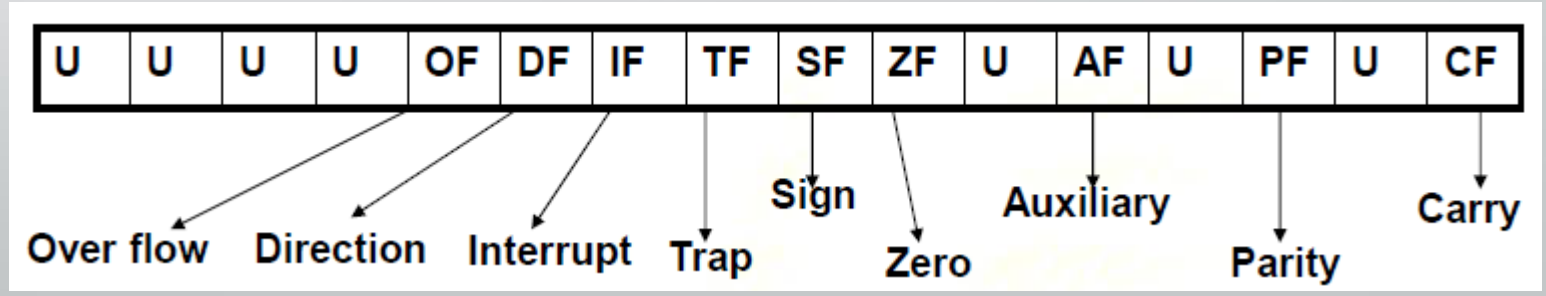
Execution Unit (EU)-Special Purpose Register



- **Special purpose registers-**
  - **Stack Pointer (SP 16 bits):** It holds offset address of the **top of the Stack**. Stack is a set of memory locations operating in Last in, first out (LIFO)manner. Stack is present in the memory in Stack Segment. It is used during instructions like PUSH, POP, CALL, RET etc. The Push instruction is used to load data onto the stack, and the Pull instruction is used to take data from the stack.
  - **Base Pointer (BP 16 bits):** BP can **hold offset address of any location in the stack segment**. It is used to access random locations of the stack.
  - **Source Index (SI 16 bits):** It is normally used to hold the **offset address for Data Segment** but can also be used for other segments using Segment Overriding. It holds offset address of source data in Data Segment during string operations.
  - **Destination Index (DI 16 bits):** It is normally used to hold the **offset address for Extra Segment** but can also be used for other segments using Segment Overriding. It holds offset address of destination in Extra Segment during string operations.

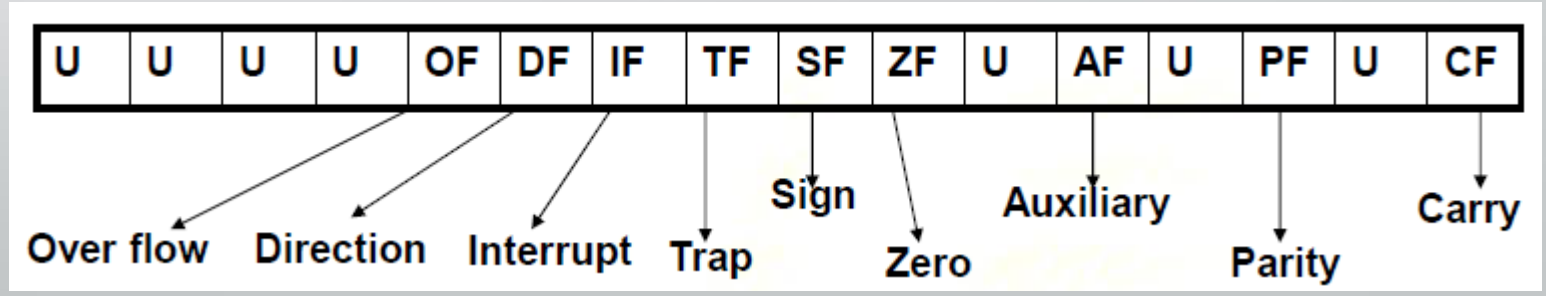
Execution Unit (EU)- ***Flag Register***

- **ALU (Arithmetic Logic Unit):** It has a 16 bit ALU. It performs 8 and 16 bit arithmetic and logic operations.
- **Operand register:** It is a 16 bit register used by the control register to hold the operands temporarily. It is not available to the programmer.
- **Flag register (16 bits):** A flag is a flip flop which indicates some conditions produced by the execution of an instruction or controls. a 16 bit flag register, **9** of the 16 are active flags and remaining **7** are undefined.
  - These flags are of two types: **6 Status flags** namely *carry flag*, *parity flag*, *auxiliary carry flag*, *zero flag*, *sign flag* and **3 Control flags** namely *trap flag*, *interrupt flag* and *direction flag*.
  - **Status flags** are affected by the ALU after every arithmetic or logic operation. They give the status of the current result.
  - **The Control flags** are used to control certain operations. They are changed by the programmer.



Execution Unit (EU)- ***Flag Register***

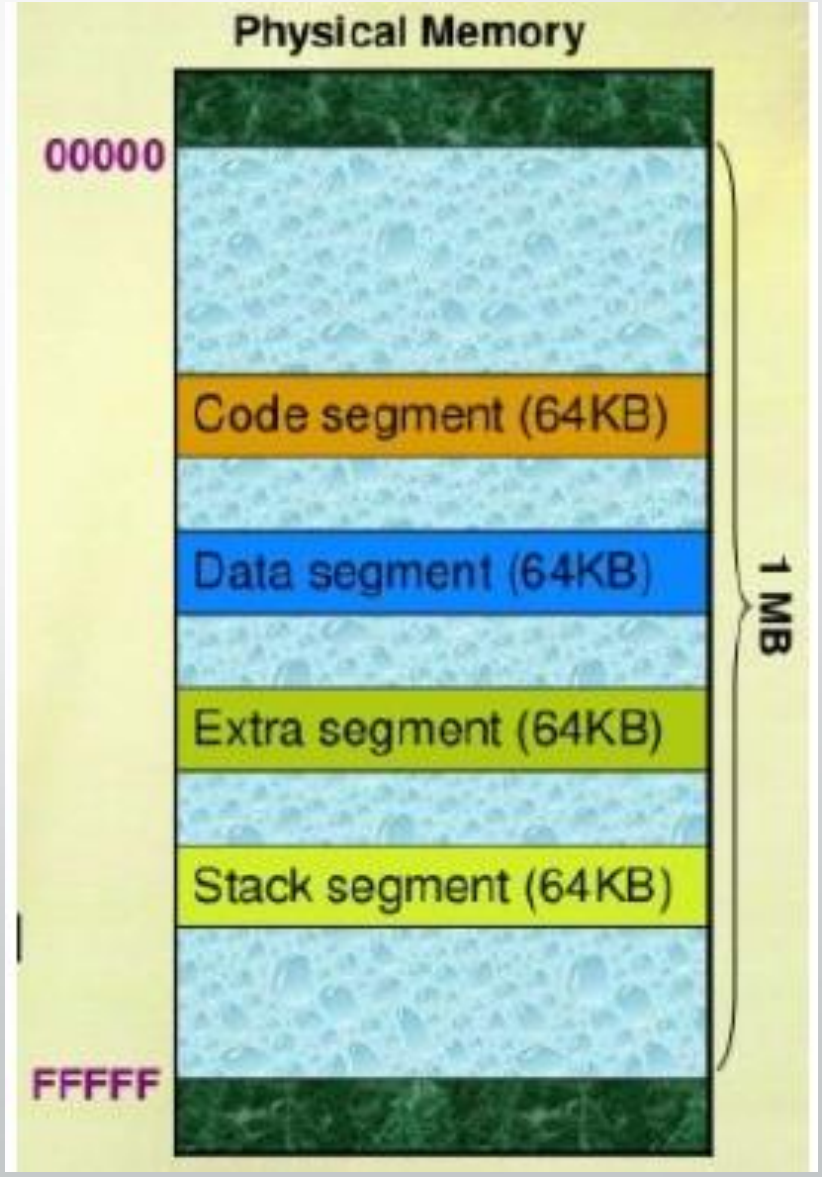
- ***PF*- Parity Flag:** This flag is set to 1, if the lower byte of the result contains even number of 1's.
- ***CF*- Carry Flag:** This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.
- ***AF*- Auxiliary Carry Flag:** This is set, if there is a carry from the lowest nibble, i.e., bit three during addition, or borrow for the lowest nibble, i.e., bit three, during subtraction.
- ***OF*- 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, and then the overflow will be set.
- ***TF*- Trap Flag "A control flag":** If this flag is set, the processor enters the single step execution mode. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.
- ***IF*- Interrupt Flag "A control flag":** If this flag is set, the mask able interrupts are recognized by the CPU, otherwise they are ignored.
- ***D*- Direction Flag "A control 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 decrementing mode.

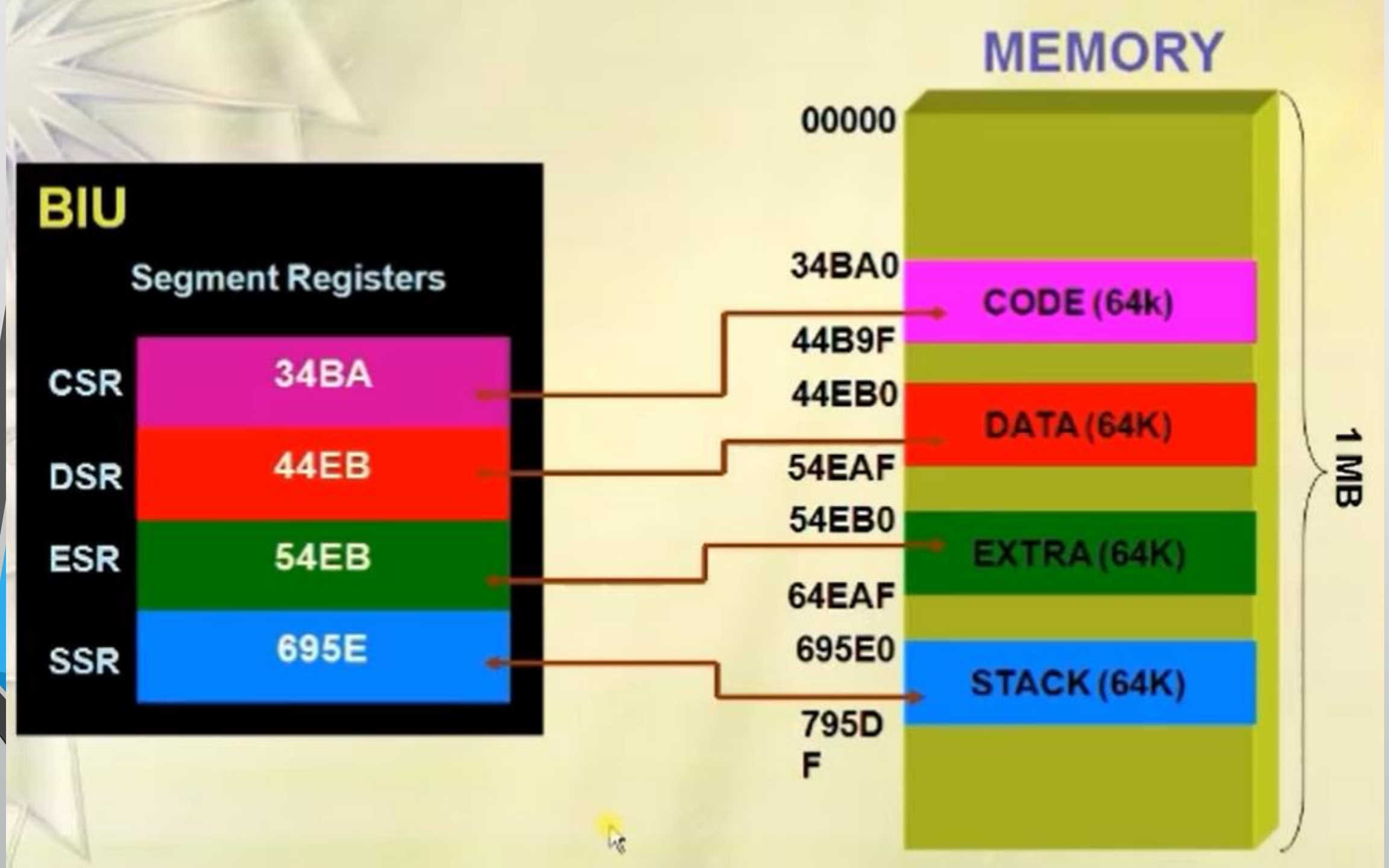


Segmented Memory

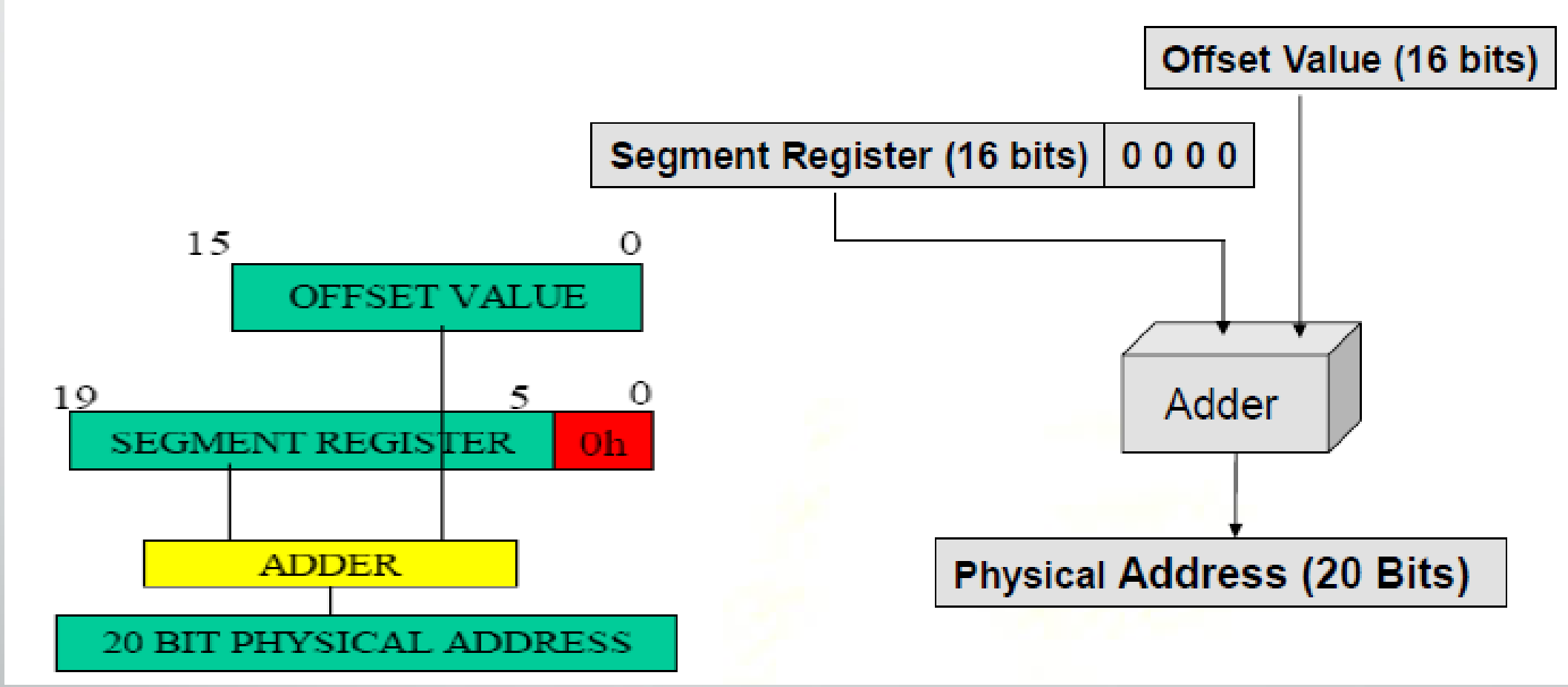
The 8086 architecture uses the concept of segmented memory. 8086 able to address a memory capacity of 1 megabyte and it is byte organized. This 1 megabyte memory is divided into 16 logical segments. Each segment contains 64 k bytes of memory.

- The memory in an 8086/88 based System is organized as segmented memory.
- The Complete physically available memory may be divided into a number of logical segments.
- A segment may be located any where in the memory
- Each of these segments can be used for a specific function.
- The 16 bit contents of the segment registers in the BIU actually point to the starting location of a particular segment. Segments may be overlapped or non-overlapped

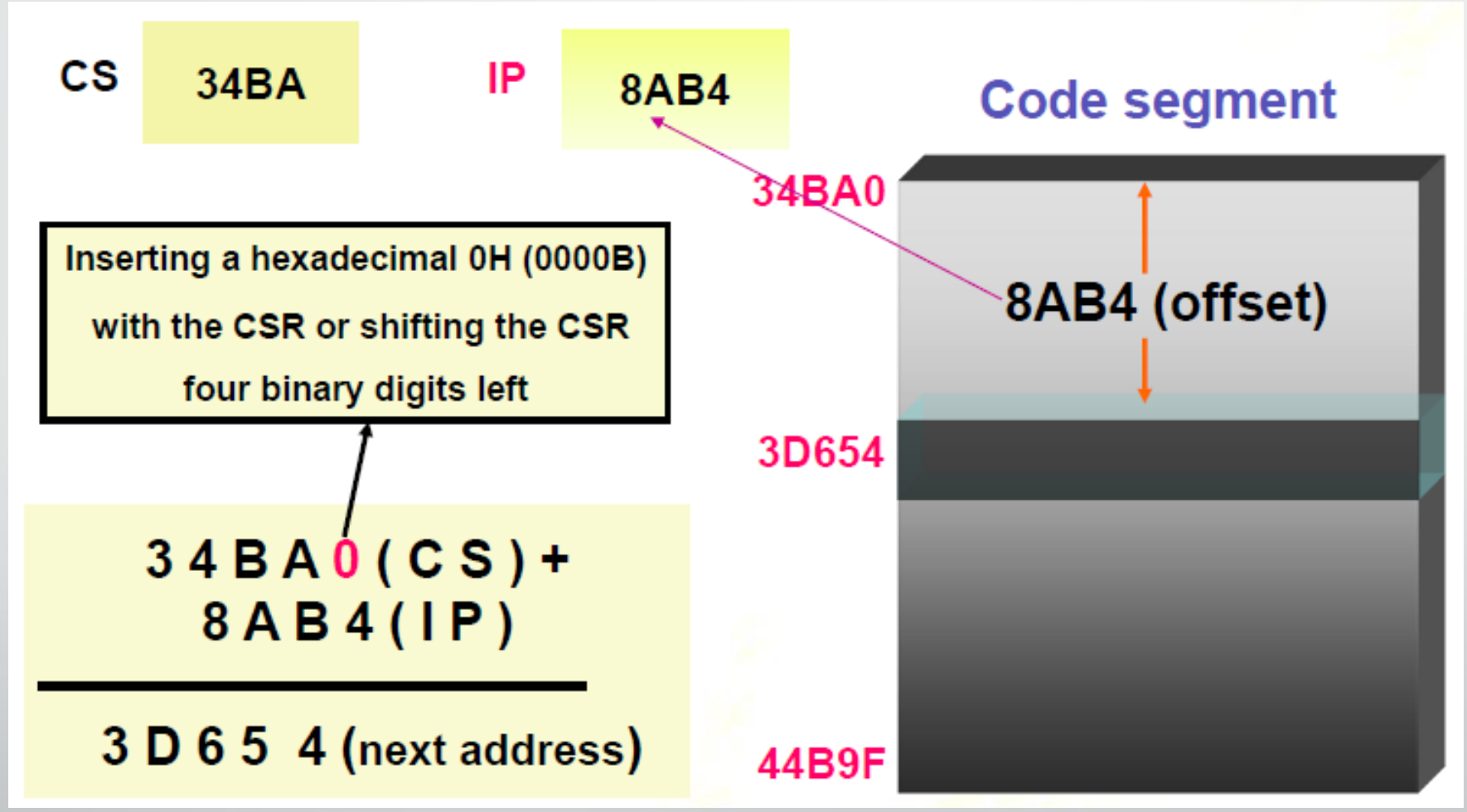



MEMORY
BIU
Segment Registers
CSR
34BA
DSR
44EB
ESR
54EB
SSR
695E
00000
34BA0
CODE (64k)
44B9F
44EB0
DATA (64K)
54EAF
54EB0
EXTRA (64K)
64EAF
695E0
STACK (64K)
795D
F
1 MB

Memory Address Generation



Memory Address Generation



Segment and Address register combination

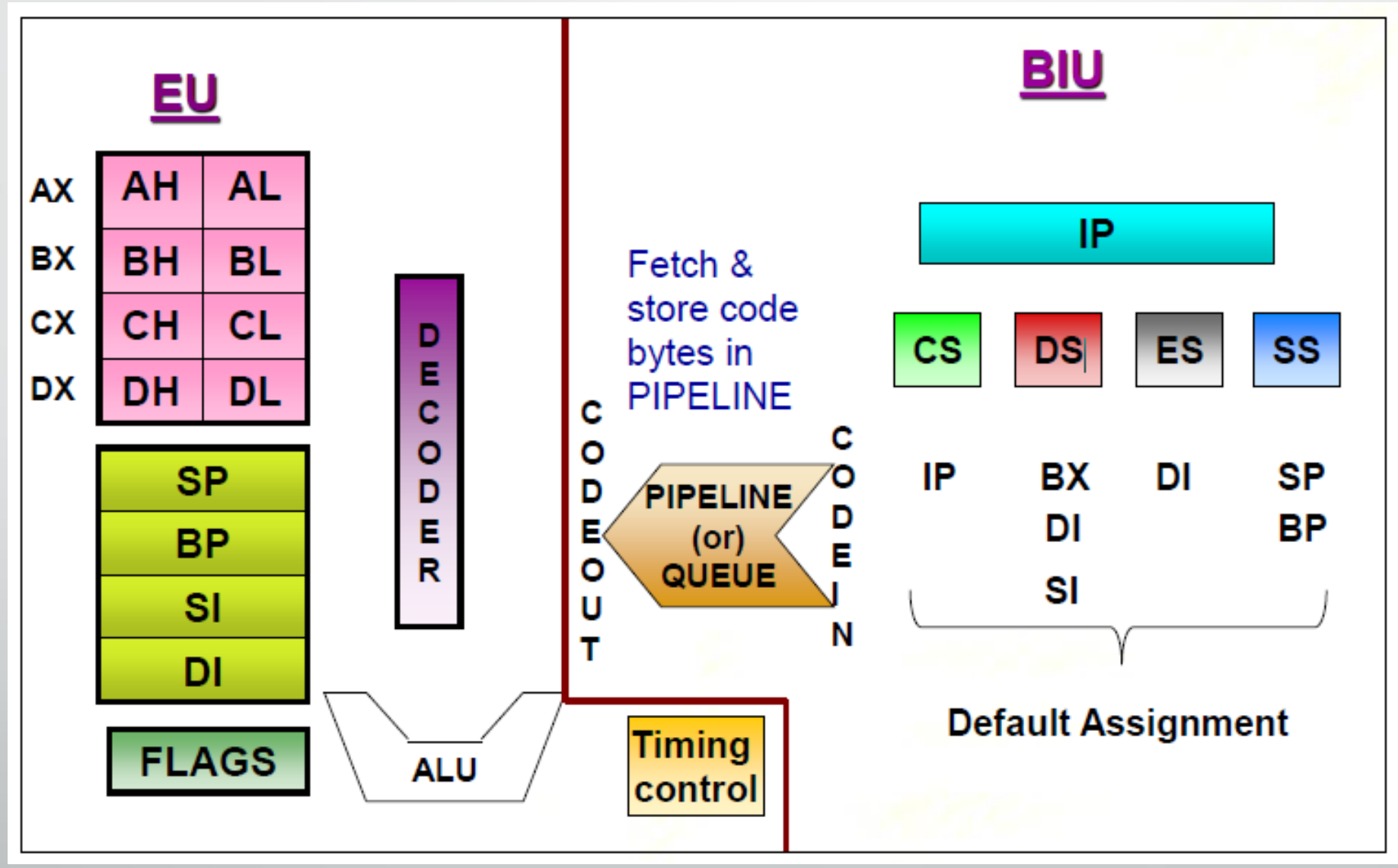
- CS:IP
- SS:SP SS:BP
- DS:BX DS:SI
- DS:DI (for other than string operations)
- ES:DI (for string operations)

segment	Offset	Special Purpose
CS	IP	Instruction address
SS	BP	Stack address
SS	SP	Top of the stack
DS	BX,DI,SI, an 8-bit number, or a 16-bit number	Data address
ES	DI for string instructions	String destination address

Segment and Address register combination

- Note that a program can have more than four segments, but can only access four segments at a time. In the real mode a combinational of A segment address and offset address access a memory location
- The microprocessor has a set of rules that apply to segments whenever memory is addressed.
- For example, the code segment register is always used with the instruction pointer to address the next instruction in a program. This combination is CS:IP.

Summary of Registers & Pipeline of 8086 µP

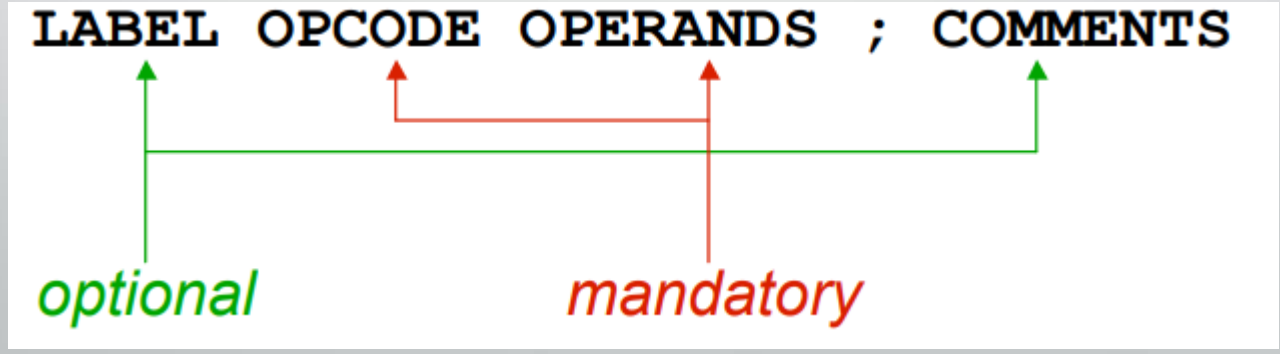


EMu8086

Each line of a program is one of the following:

- an instruction
- an assembler directive (or pseudo-op)
- a comment Whitespace (between symbols) and case are ignored. Comments (beginning with ";") are also ignored.

An instruction has the following format:



Thank you