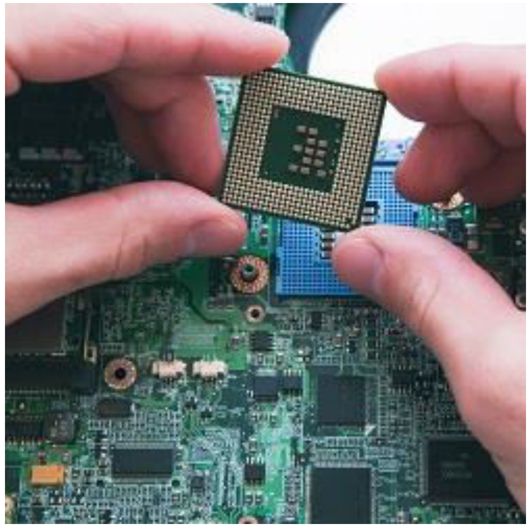
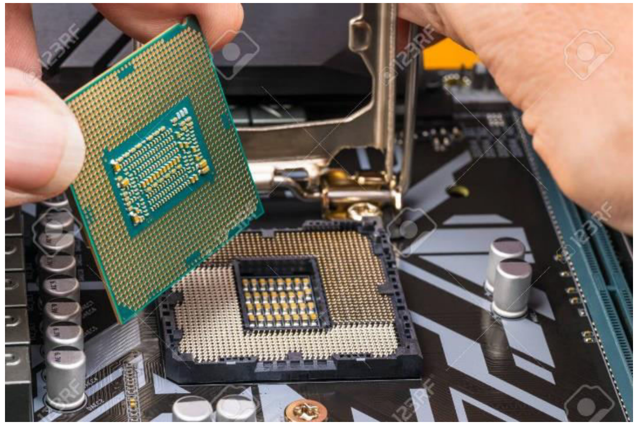
Ch-1
The 8086 microprocessor

Microprocessor

A **microprocessor** is an electronic component that is used by a computer to do its work.
It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors, and diodes that work together.





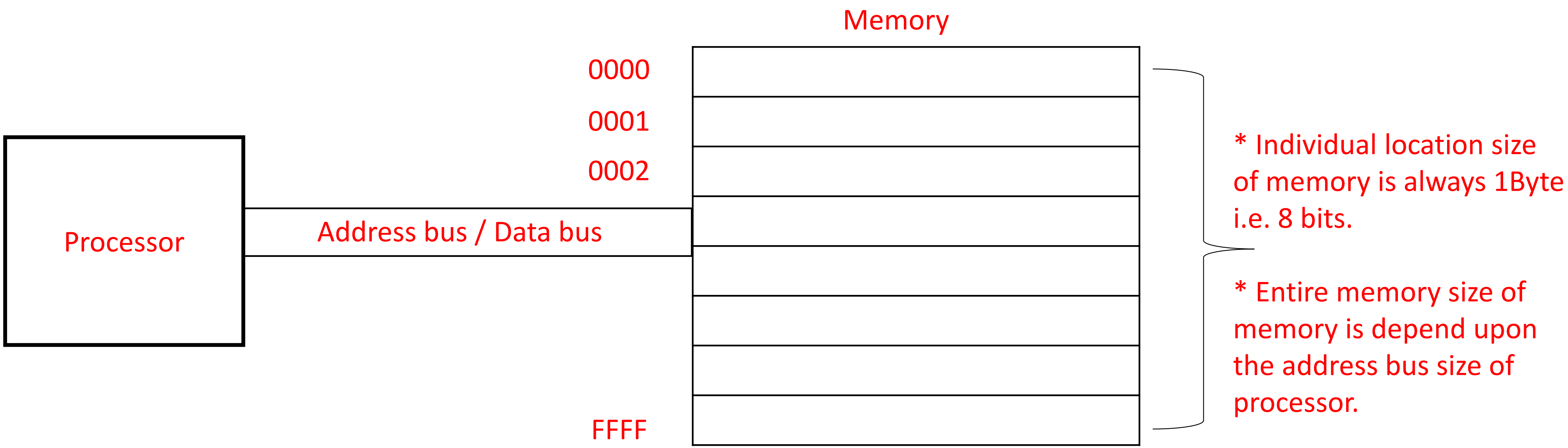
The microprocessor is the central unit of a computer system that performs arithmetic and logic operations, which generally include adding, subtracting, transferring numbers from one area to another, and comparing two numbers. It's often known simply as a processor, a central processing unit, or as a logic chip.

Evolution of Microprocessor

Intel Microprocess				
Name	**Year**	**Transistors**	**Clock speed**	**Data width**
8080	1974	6,000	2 MHz	8 bits
8085	1976	6,500	5 MHz	8 bits
8086	1978	29,000	5 MHz	16 bits
8088	1979	29,000	5 MHz	8 bits
80286	1982	134,000	6 MHz	16 bits
80386	1985	275,000	16 MHz	32 bits
80486	1989	1,200,000	25 MHz	32 bits
Pentium	1993	3,100,000	60 MHz	32/64 bits
Pentium II	1997	7,500,000	233 MHz	64 bits
Pentium III	1999	9,500,000	450 MHz	64 bits
Pentium IV	2000	42,000,000	1.5 GHz	64 bits
Pentium IV "Prescott"	2004	125,000,000	3.6 GHz	64 bits
Intel Core 2	2006	291 million	3 GHz	64 bits
Pentium Dual Core	2007	167 million	2.93 GHz	64 bits
Intel 64 Nchalem	2009	781 million	3.33 GHz	64 bits

Basic Functions of processor

1. Programmer write a program using HLL or ALP(instructions)
2. Assembler will convert HLL/ALP into machine code.(opcode of instructions (hex))
3. Loader is responsible to load program into memory (binary form)



1. Processor calculates the memory address and fetch the content of that memory location.
2. Processor decodes the fetched instruction and execute it.

Content of memory location can be Data/instructions

Features of 8086

Explain features of 8086 (each for 2 marks)

1. Basic Features
2. Special Features
3. Miscellaneous Features

1.Basic Features

- Processor Size
- Speed of processor
- Address bus size for memory
- Address bus size for I/O

(1) It is a 16-bit processor. This implies that

(a) It has a 16-bit ALU that can perform 16-bit operation simultaneously.

(b) It has 16-bit registers and internal data bus.

(c) It has 16-bit external data bus.

(2) It has three versions based on the basis of frequency of operation.

(i) 8086 → 5 MHz.

(ii) 8086-2 → 8 MHz.

(iii) 8086-1 → 10 MHz.

(3) 8086 has 20-bit address lines to access memory, hence it can access

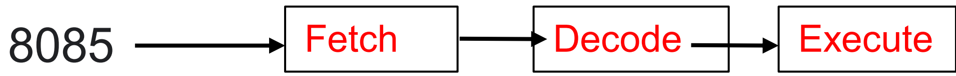
$$2^{20} = 1 \text{ MB memory locations}$$

(4) It has 16-bit address lines to access I/O devices, hence it can access

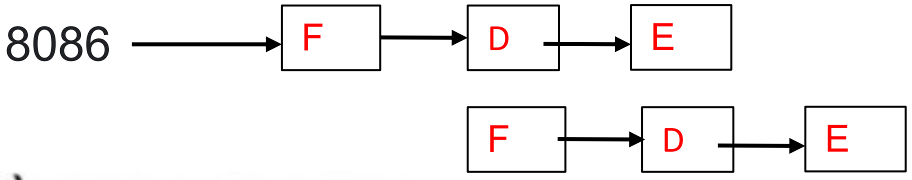
$$2^{16} = 2^6 \times 2^{10} = 64 \times 1\text{ K}$$
$$= 64 \text{ K I/O locations}$$

2. Special Features

- 8086 is a pipelined processor
- 8086 can operate in 2 modes
- 8086 uses memory banks
- 8086 uses memory segmentation



At a time only 1 instruction



At a time 2 instruction

1) 8086 is a pipelined processor (i.e. it supports pipelined architecture)

- It uses a two stage pipelining i.e. **Fetch stage** that pre-fetches up to 6 bytes of instructions stores them in the queue and **Execute stage** that executes these instructions.
- Pipelining improves the performance of the processor i.e. the operations are faster.

2) 8086 can operate in 2 modes

a. **Minimum mode** → A system with only 1 processor i.e. 8086.

b. **Maximum mode** → A system with 8086 and other processors like 8087-(Math Co-processor), 8089-(IO processor) or multiple 8086 processors.

3) 8086 uses memory banks

- The 8086 uses a memory banking system i.e. the entire data is not stored sequentially in a single memory of 1 MB but the memory is divided into two banks of 512KB each.

- The banks are called Lower bank (or even bank, because it stores the data bytes at even locations i.e. 0, 2, 4....) and Higher Bank (or odd bank, because it stores the data bytes at odd locations i.e. 1, 3, 5...).
- The benefit of this is that 16-bit data can be accessed in a single access even though the memory chip can store only 8-bit at a location.

4) 8086 uses memory segmentation

- A 16-bit address in an instruction or a 16-bit address in a register can access a memory location, although 8086 has 20 address lines. This is made possible using the concept of Segmentation that divides the memory into logical components.
- Here the memory is divided into 16 segments of a capacity of 2^{16} (= 65536 B = 64 KB) each and is used as: Code, Stack, Data and Extra Segment.

3. Miscellaneous Features

- Interrupts
- Registers
- Instruction set
- Data size for ALU

(1) **It has 256 vectored interrupts :** There are also non-vectored interrupts in 8086, but they are routed to one of these interrupts.

(2) It has 14, 16-bit registers.

(3) It has a powerful instruction set, that supports multiply and divide operations also. (These operations were not possible in the processors earlier to 8086).

(4) 8086 can perform operations on bit, byte (8-bit), word (16-bit) or a string (block of data) types of data.

Architecture of 8086

Draw and explain architecture of 8086
- Diagram –
- Explanation –

Main task of Processor :

Fetch

Decode

Execute

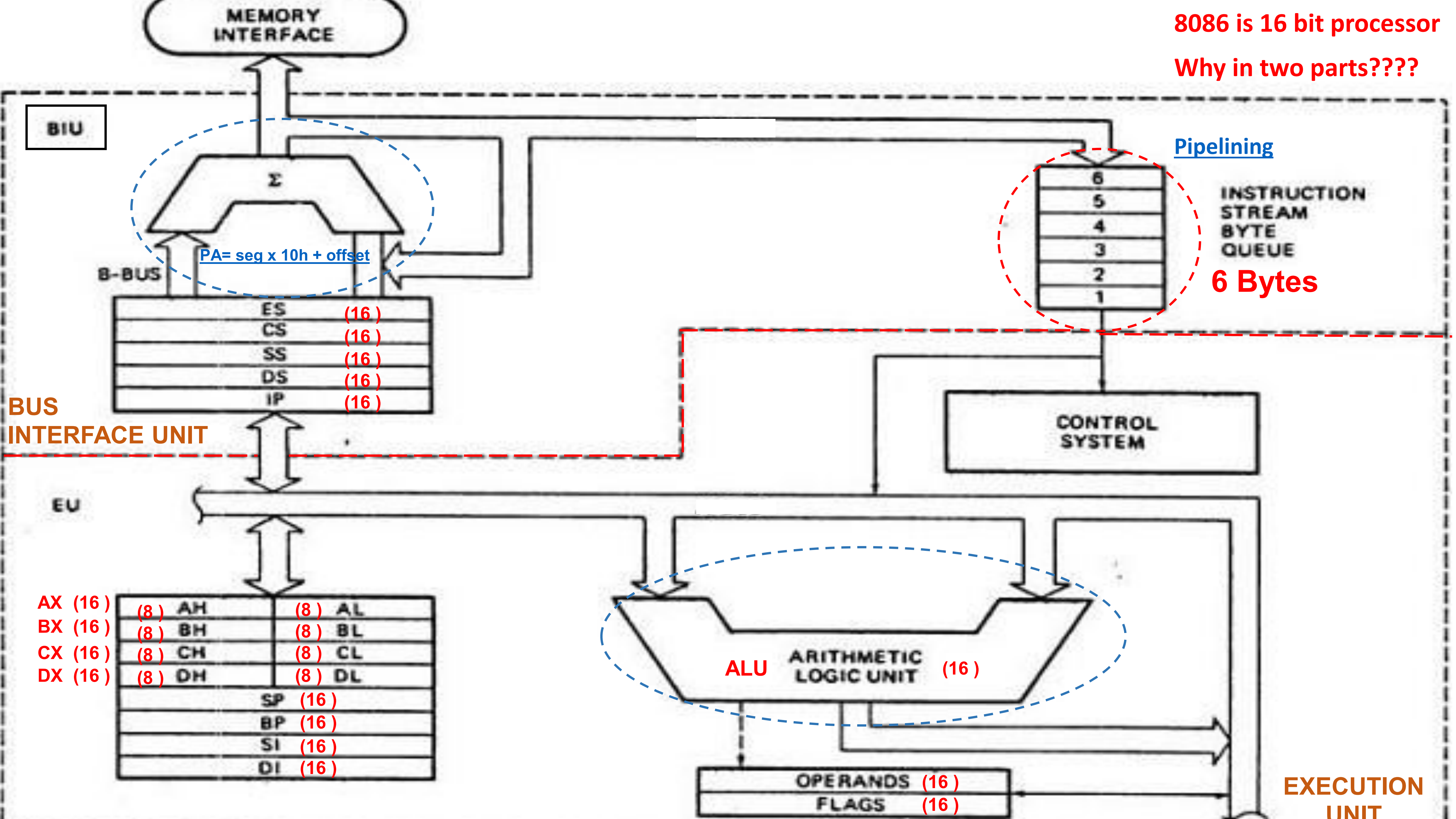
Draw architecture of 8086 and explain any one unit

- Diagram –
- Explanation –

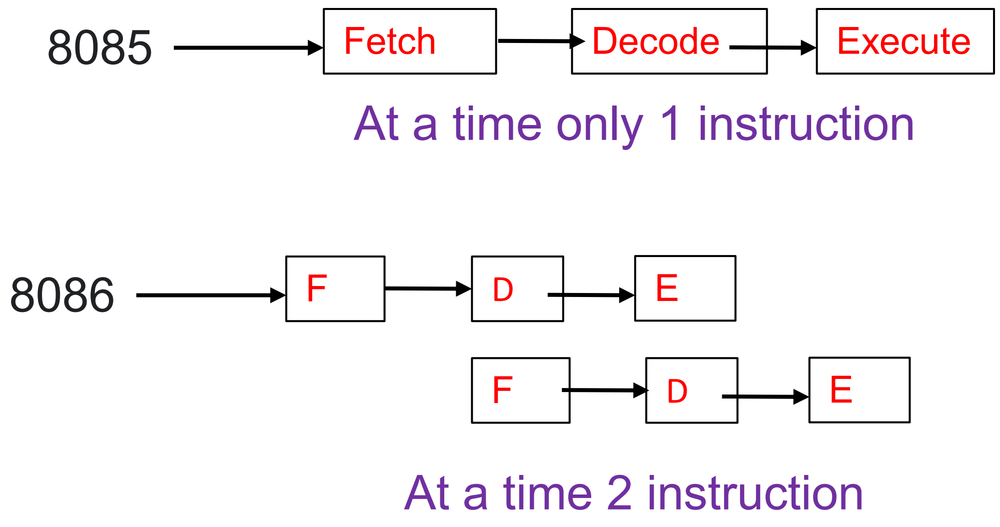
Draw architecture of 8086 (3 marks)

To understand any architecture we must know :

- How do you fetch the instruction?
- Where is getting decoded?
- Where is getting executed?

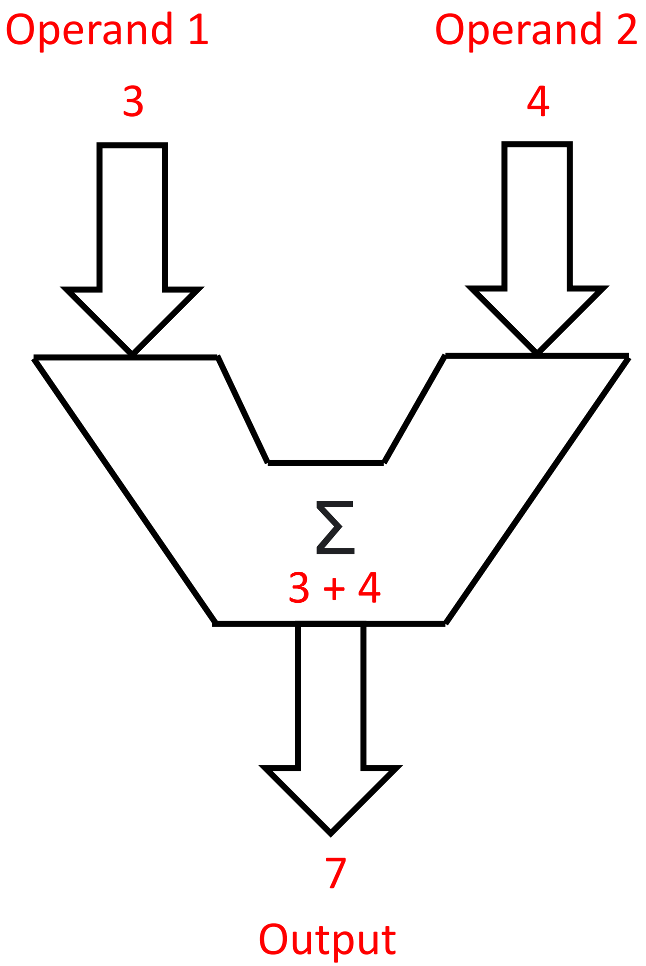

MEMORY INTERFACE
8086 is 16 bit processor
Why in two parts????
BIU
Σ
PA= seg x 10h + offset
B-BUS
ES (16)
CS (16)
SS (16)
DS (16)
IP (16)
Pipelining
6
5
4
3
2
1
INSTRUCTION STREAM BYTE QUEUE
6 Bytes
BUS INTERFACE UNIT
CONTROL SYSTEM
EU
AX (16)
BX (16)
CX (16)
DX (16)
(8) AH
(8) AL
(8) BH
(8) BL
(8) CH
(8) CL
(8) DH
(8) DL
SP (16)
BP (16)
SI (16)
DI (16)
ALU
ARITHMETIC LOGIC UNIT (16)
OPERANDS (16)
FLAGS (16)
EXECUTION UNIT

Pipelining :



- Pipelining is one of the special feature of 8086 processor.
- Due to pipelining 8086 processor architecture is divided into two parts.
- BIU will fetch the instruction from memory and it will pass that fetched instruction to the Execution unit.
- While execution unit executes the current instruction BIU will fetch the next instruction.
- Pipelining helps in order to increase the speed of processor.

Back



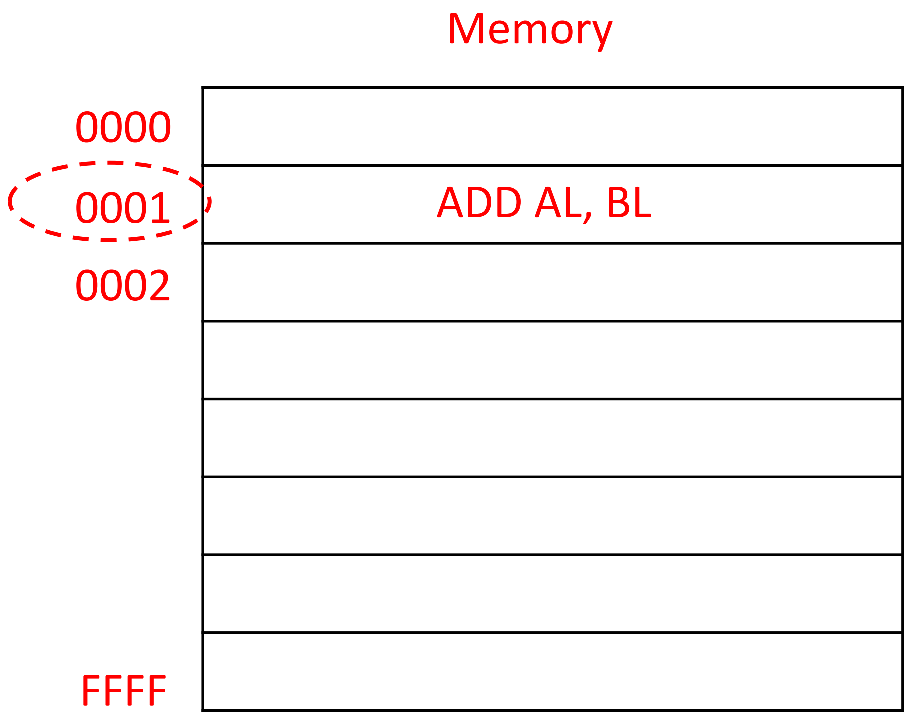
- In the architecture diagram of 8086 everything is in rectangle except two following shapes.
- This the diagram of arithmetic circuit.
- In the given circuit there are two inputs which are used to take operand values.
- operation will be performed on input operands and it will produce output.
- For example 3 + 4

***In the architecture of 8086 there are two such type of circuits are available from that one is **ALU** who is responsible to perform arithmetic operations and another one is to **calculate memory address.**

Back

There is an instruction ADD AL, BL (wants to add content of AL reg with content of BL register)

- This instruction will be loaded into the memory
- Processor will fetch this instruction from memory and to perform fetching processor has to calculate the physical address of that memory location.
- Hence arithmetic circuit in upper section is responsible to calculate the physical address of memory location where instruction is stored.
- ALU which is present at lower section i.e. in execution unit is responsible to perform actual addition of reg AL and BL data.



How to calculate physical address ????????

Formula to calculate physical address of memory location is : **PA= Seg x 10h + offset**

Overview of memory segmentation:

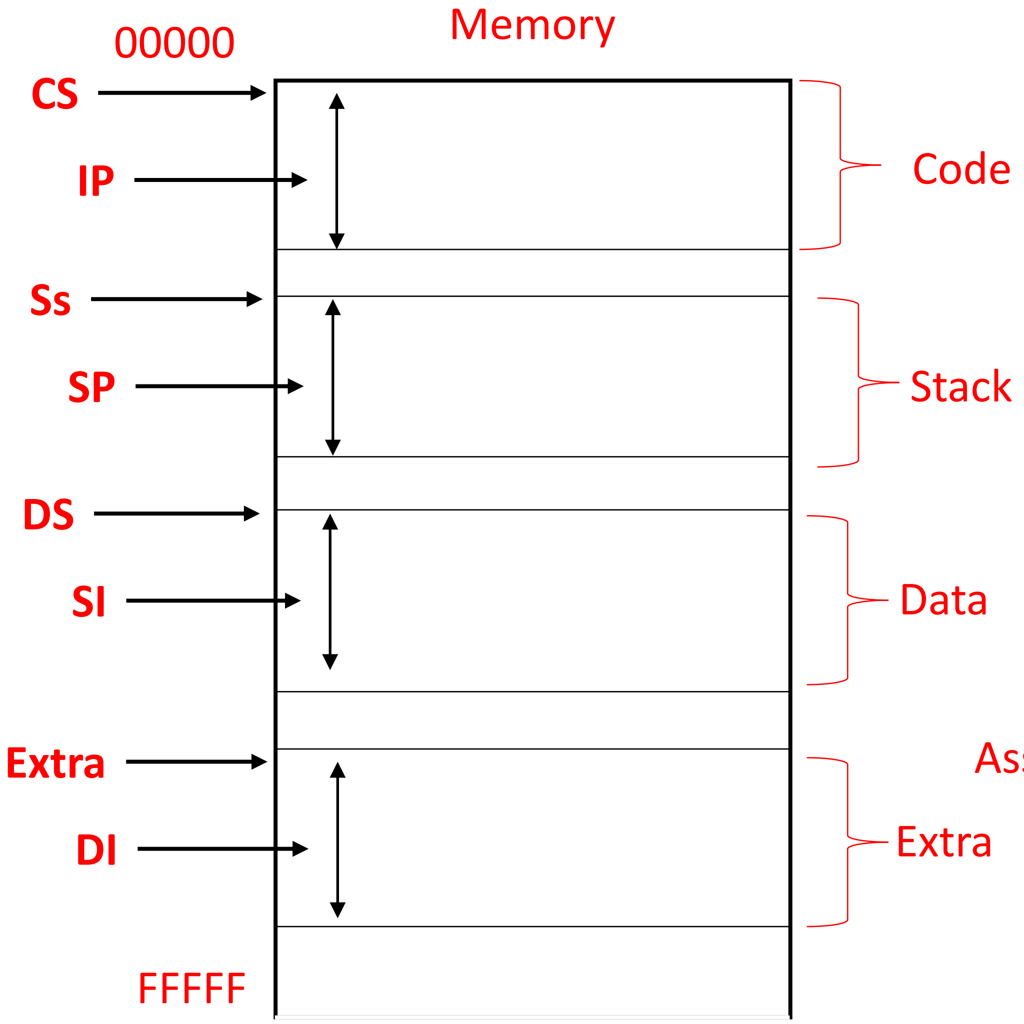
- This is entire memory supported by 8086.
- The amount of memory that you can access is depend upon the address size of processor. 8086 has 20 bit address bus hence entire memory size is 1 MB

Example :

- If you want to access any file from your hard disk you will access that file from file name.
- We don't know the physical address of that file.
- We know the logical address i.e. name

In 8086 memory is divided into 4 part i.e. segments

If you want to access any location of memory which will have its unique physical address.
We give segment address (starting addr of segment) and
offset addr (location within particular segment)



Example :

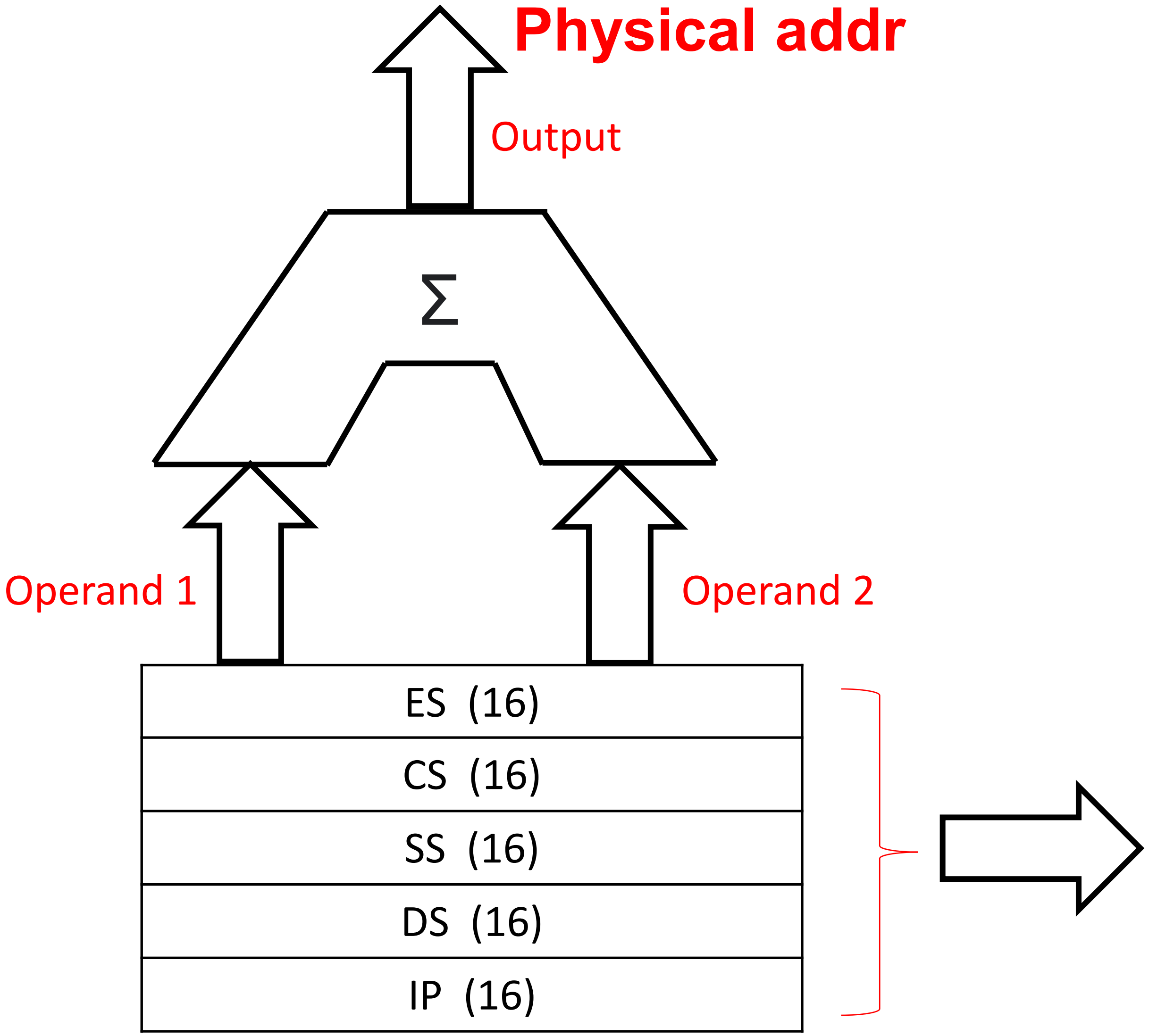
- There is a book which has 1000 pages.
- Assume there are 10 chapters and each chapter has 100 pages.
- We want to go on page no. 564.
- There are two methods :

1 Directly call 564

2 5 th chapter **CS**
64 page **IP**

Assume: CS= 1000 & IP= 2345

PA = Seg x 10H + offset
1000 x 10H + 2345
10000 + 2345
PA = 12345 H

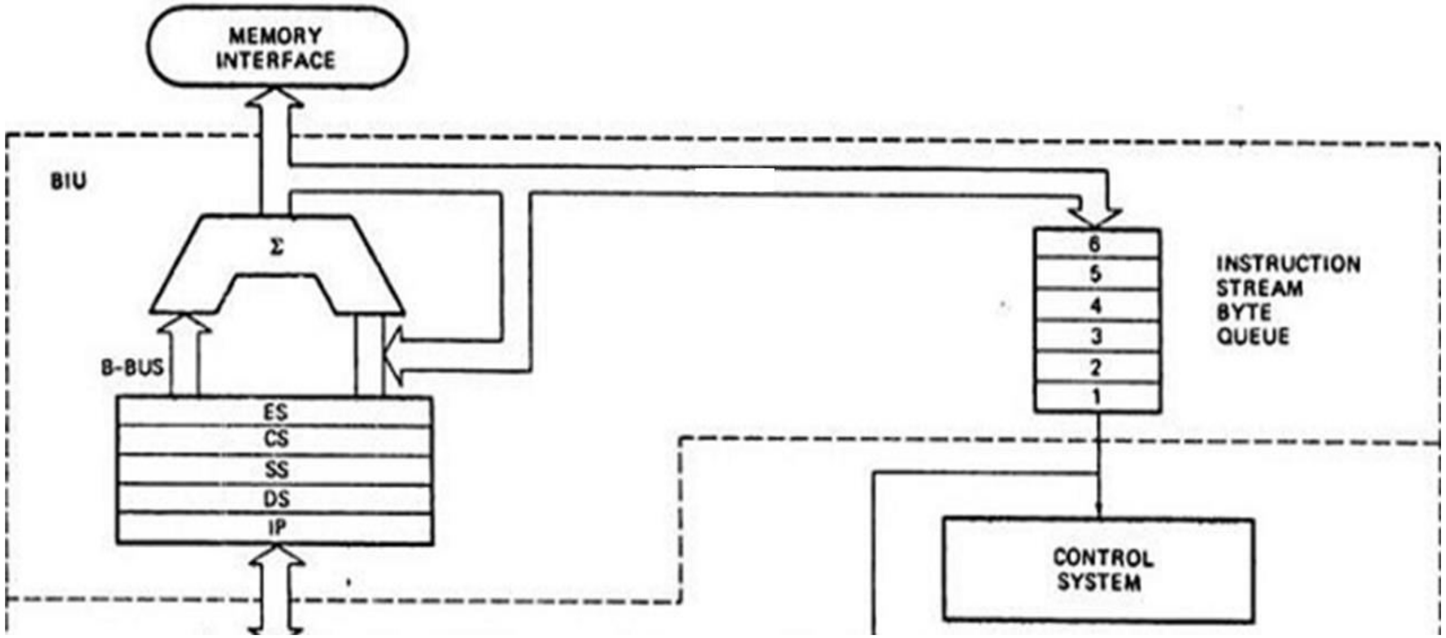


These are the registers which are used to stored starting address of segments

Back

Functions of BIU :

- Fetch the next Instruction from memory
- Calculate the Physical Addr
- Manage the instruction queue.



Managing of queue :

- Processor fetch the next instruction and calculate the physical addr of that instruction.
- After PA calculation processor will transfer that instruction into instruction queue through data bus (16 bit). This phase is called as prefetching phase.
- Maximum size of instruction queue is 6 bytes.

1 ← EI
2
3
4 ← 6 bytes of program
5
6
7
8 when EU executing the next
9 instruction BIU fetch the next
. 6 bytes of program
.

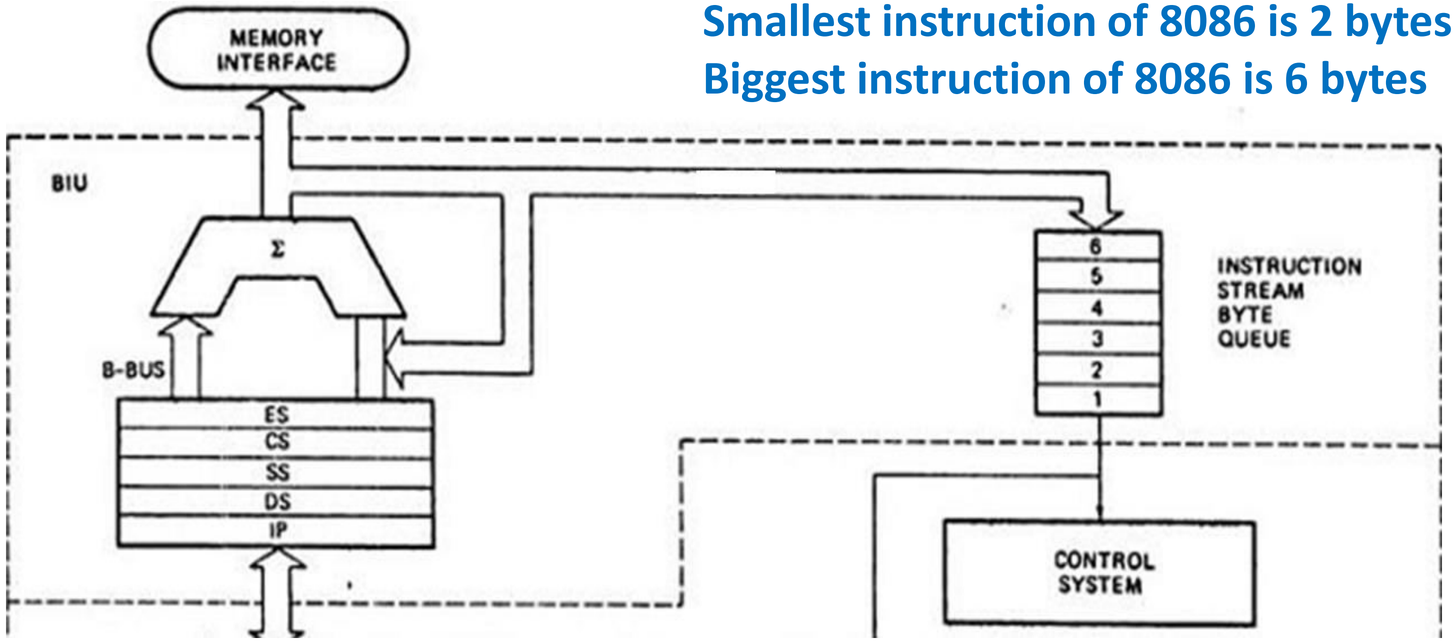
How many Instructions are in the queue????? (nobody knows)

How many bytes are in the queue????? (6 bytes)

All instructions are in different size :

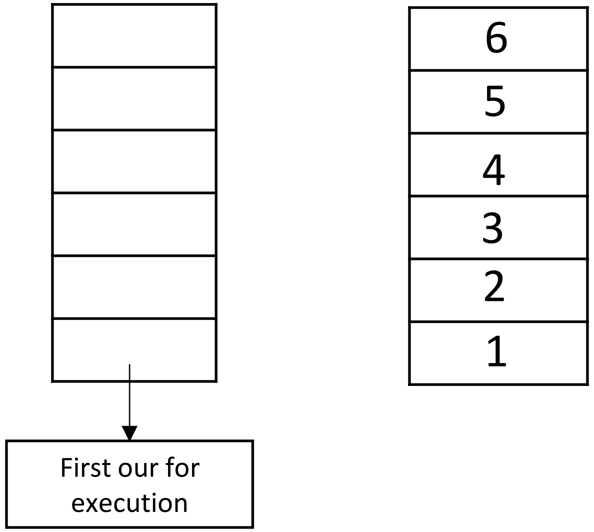
1) ADD BL,CL : 1 byte
2) ADD CL, 02H : 2 bytes
3) ADD AX, 2000H : 3 bytes

Smallest instruction of 8086 is 2 bytes
Biggest instruction of 8086 is 6 bytes



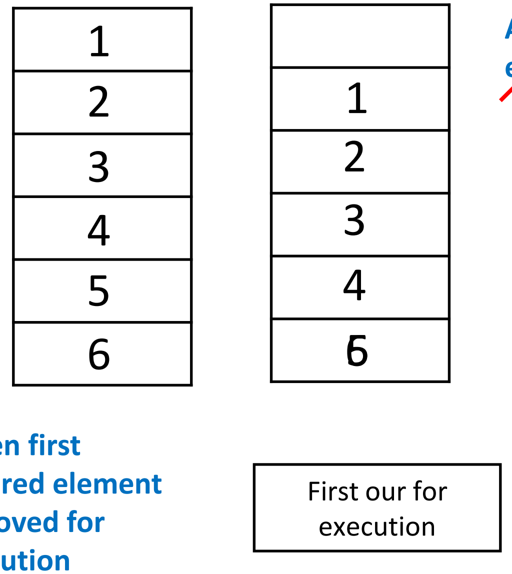
Smallest instruction of 8086 is 2 bytes
Biggest instruction of 8086 is 6 bytes

- 8086 has 6 bytes instruction queue
- It is FIFO type of queue i.e. First In First Out
- Processor will fetch the next **6 bytes** and store into the queue. EU removes one by one each byte for execution



When BIU will refill the queue???

1



When first entered element removed for execution

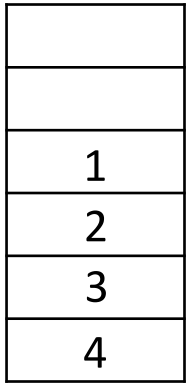
~~**After one byte empty**~~

2

~~**When all 6 bytes are transferred**~~

3

When 2 bytes are transferred for execution.

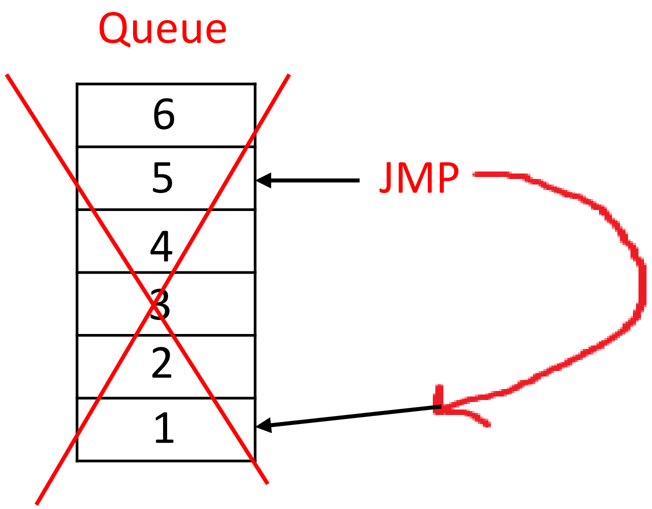


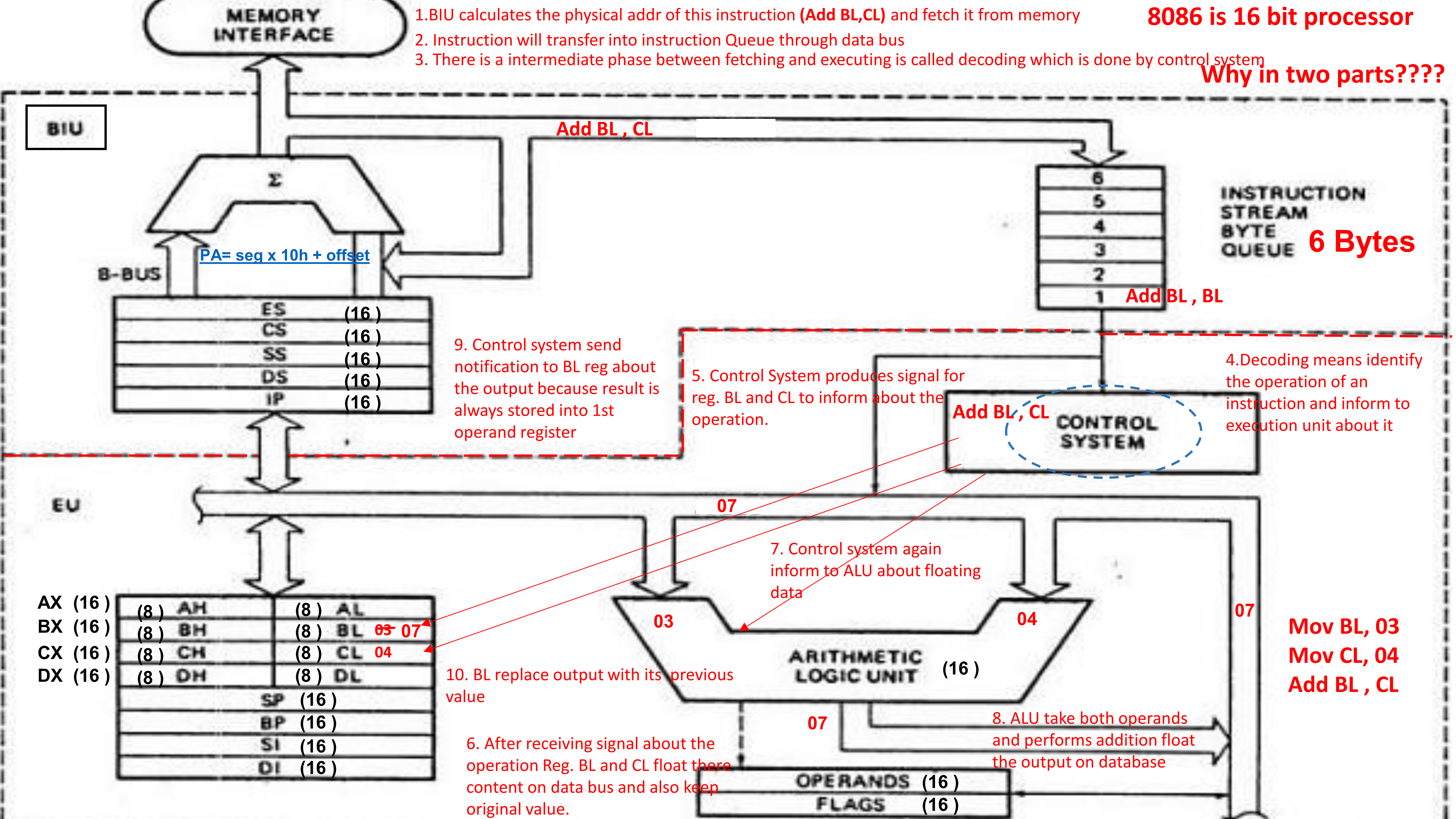
When BIU will refill the queue???

When 2 bytes are transferred for execution. Because the smallest instruction of 8086 is 2 bytes

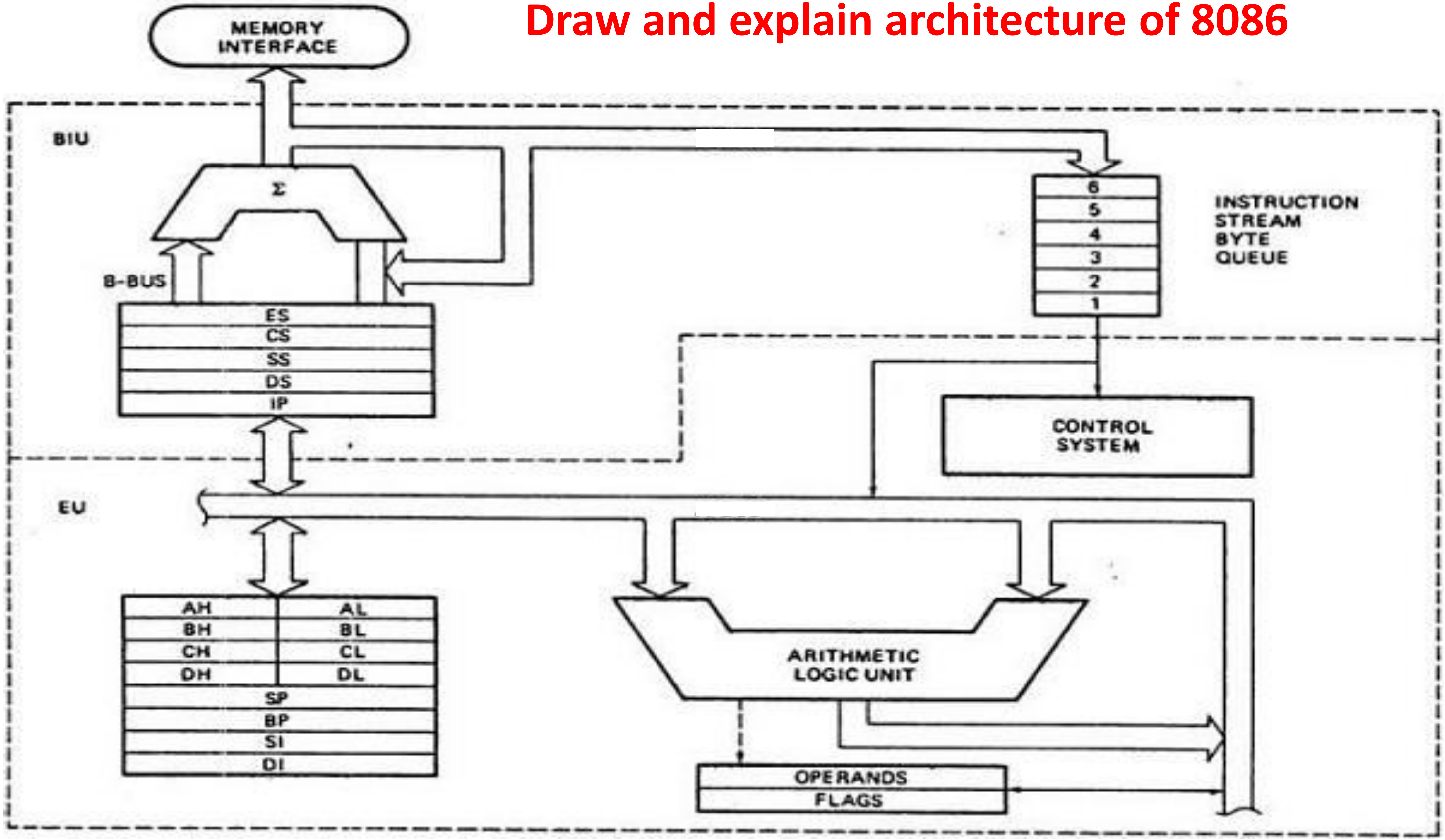
When pipeline fails ???

- If a Jump (JMP) or CALL instruction appears in the main program, then all the existing instruction bytes in the Queue are flushed out and Queue is made empty.
- Then it is reloaded with the new instruction bytes which correspond to the new locations mentioned in the JMP or CALL instructions.
- The refilling of the queue then continues from the new locations corresponding to the main program.




8086 is 16 bit processor
Why in two parts????
1.BIU calculates the physical addr of this instruction (Add BL,CL) and fetch it from memory
2. Instruction will transfer into instruction Queue through data bus
3. There is a intermediate phase between fetching and executing is called decoding which is done by control system
MEMORY INTERFACE
BIU
Add BL , CL
Σ
PA= seg x 10h + offset
B-BUS
ES (16)
CS (16)
SS (16)
DS (16)
IP (16)
6
5
4
3
2
1
INSTRUCTION STREAM BYTE QUEUE
6 Bytes
Add BL , BL
4.Decoding means identify the operation of an instruction and inform to execution unit about it
5. Control System produces signal for reg. BL and CL to inform about the operation.
Add BL , CL
CONTROL SYSTEM
9. Control system send notification to BL reg about the output because result is always stored into 1st operand register
EU
07
7. Control system again inform to ALU about floating data
AX (16)
BX (16)
CX (16)
DX (16)
(8) AH
(8) AL
(8) BH
(8) BL ~~03~~ 07
(8) CH
(8) CL 04
(8) DH
(8) DL
SP (16)
BP (16)
SI (16)
DI (16)
03
04
07
ARITHMETIC LOGIC UNIT (16)
10. BL replace output with its previous value
07
8. ALU take both operands and performs addition float the output on database
6. After receiving signal about the operation Reg. BL and CL float there content on data bus and also keep original value.
OPERANDS (16)
FLAGS (16)
Mov BL, 03
Mov CL, 04
Add BL , CL

Draw and explain architecture of 8086



The Execution Unit (EU) Main function of EU is **decoding** and **execution** of the instructions. In order to carry out its tasks it has the following units :

- Arithmetic Logic Unit (ALU)
- General Purpose Registers. PPI
- Decoder
- Flag Register. → from PPI
- Control Unit
- Pointer and Index Register

Arithmetic Logic Unit (ALU)

- The ALU in the EU is a 16 bit unit i.e. it can perform 16-bit operation simultaneously
- It is capable of performing a variety of arithmetic and logic operations such as add, subtract, AND, OR, NOT, EX-OR, increment, decrement, shift etc.

Control Circuitry

The control circuit is a part of EU. It is used for directing the internal operations.

A Decoder

- "The process of translation from instructions into action is known as decoding"
- A decoder in the execution unit (EU) is used for translating the instructions fetched from the memory into a series of actions.
- The EU will actually carry out these actions.

Pointer and Index Register

The execution unit also contains the following 16 bit registers.

- Base Pointer (BP) register
- Stack Pointer (SP) register
- Source Index (SI) register
- Destination Index (DI) register

These registers can be used as general purpose 16-bit registers. But mainly they are used to hold the 16 bit offset of data word in one of the segments as given below :

Base Pointer (BP) Register

- This is a 16 bit register in the E.U. Its function is to hold the 16 bit offset relative to the stack segment (SS) register.
- But BP has a specific use. BP is used whenever we pass a parameter by way of stack.
- The BP register can also be used as an offset register in the addressing mode called **base addressing mode**.

Stack Pointer (SP) Register

- This is also a 16-bit register in the E.U. Its function is to hold the 16-bit offset address relative to stack segment (SS) register.
- SP is used for sequential access of stack segment.
- It always points to the top of stack.
- It is mainly used during instructions like PUSH, POP, CALL, RETURN etc.

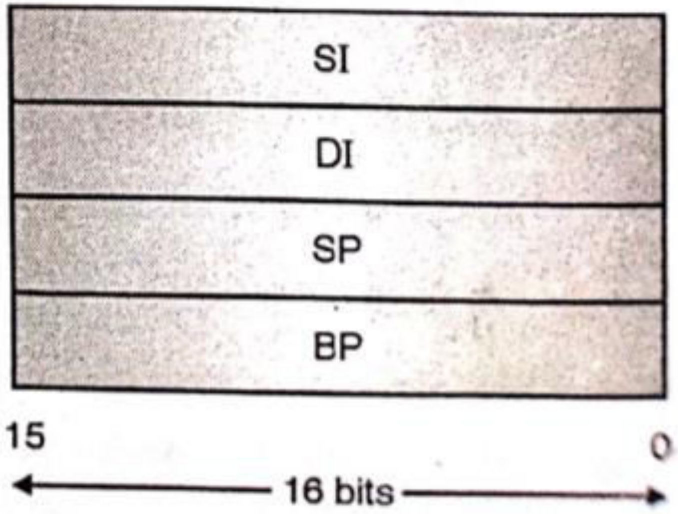
Source Index (SI) Register

- This register is used for holding the offset of a data word in the data segment (DS).
- The physical address of a location in the data segment can be generated by adding a hardwired zero to the segment base (16 bit) held by the data segment (DS) register and then by adding the offset in the SI register to it.

Data segment (DS) Register	→		2	0	0	0	0	←	Hard wired 0
Source index (SI) register (offset)	→	+		1	F	2	3		
20 bit physical address in D.S.	→		2	1	F	2	3		

Destination Index (DI) Register

- This register is used for holding the 16 bit offset of a data word in the extra segment (ES).
- The source index (SI) register and destination index (DI) register are used for the string related instructions.
- For example if we want to move a block of data from memory to memory, then source index (SI) register can be used to point to the source memory address and destination index (DI) register is used to point to the destination memory address.



m(15.3)Fig. 2.4.2 : Pointer register of EU

The Bus Interfacing Unit (BIU)

The bus interface unit performs all the activities related to Bus. Specifically BIU has the following **five functions**

1. Instruction **fetching** (reading) from primary memory. (performed over the system bus)
2. R/W of data operand from/to primary memory. (performed over the system bus)
3. I/O of data from/to peripheral ports. (performed over the system bus)
4. Address generation for memory reference
5. Instruction queuing in an instruction queue.

In order to carry out its functions BIU has the following modules :

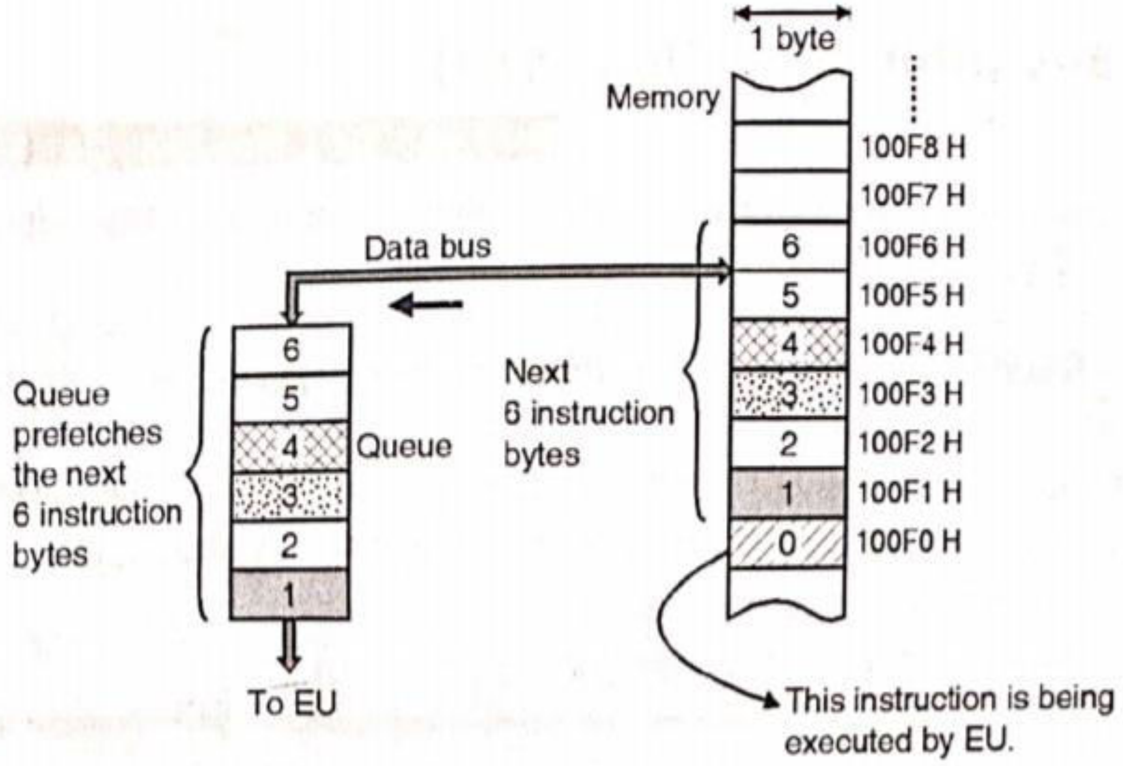
- Instruction queue.
- Segment registers
- An instruction pointer register (IP).
- Address generation and bus control.

The Instruction Queue

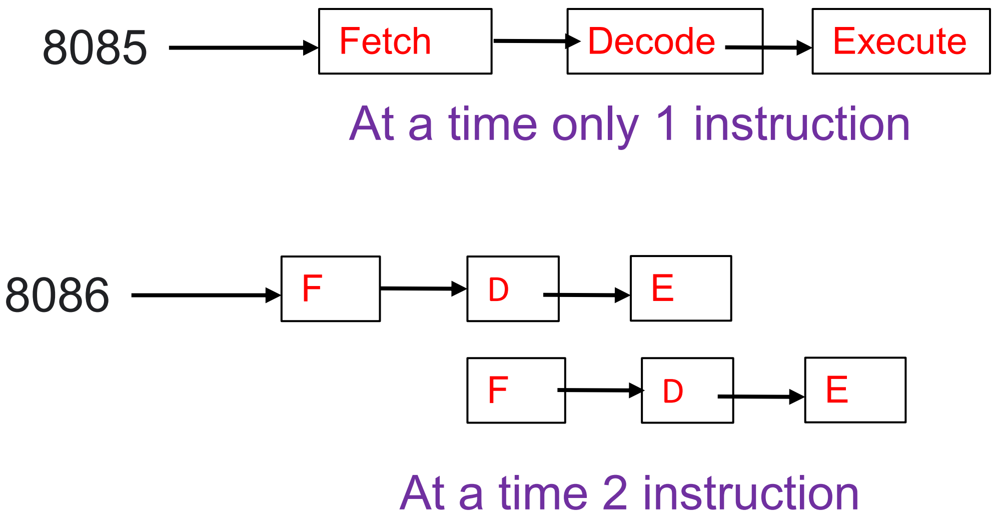
- The execution unit is supposed to decode or execute an instruction. Decoding does not require the use of buses.
- When EU is busy in decoding and executing an instruction, the BIU fetches upto six instruction bytes for the next instructions.
- These bytes are called as the prefetched bytes and they are stored in a first-in-first-out (FIFO) register set, which is called as a "**queue.**"

Significance of Queue

- To understand the significance of the queue, refer Fig. 2.5.1.
- As shown in Fig. 2.5.1, while the EU is busy in decoding the instruction corresponding to memory location 100F0, the BIU fetches the next six instruction bytes from locations 100F1 to 100F6 numbered as 1 to 6.
- These instruction bytes are stored in the 6 byte Queue on the first-in-first-out (FIFO) basis.
- When EU completes the execution of the existing instruction, and becomes ready for the next instruction, it simply reads the instruction bytes in the sequence 1, 2, ... from the Queue.
- Thus the Queue will always hold the instruction bytes of the next instructions to be executed by the EU.



Pipelining :



- Pipelining is one of the special feature of 8086 processor.
- Due to pipelining 8086 processor architecture is divided into two parts.
- BIU will fetch the instruction from memory and it will pass that fetched instruction to the Execution unit.
- While execution unit executes the current instruction BIU will fetch the next instruction.
- Pipelining helps in order to increase the speed of processor.

Advantages of Pipelining :

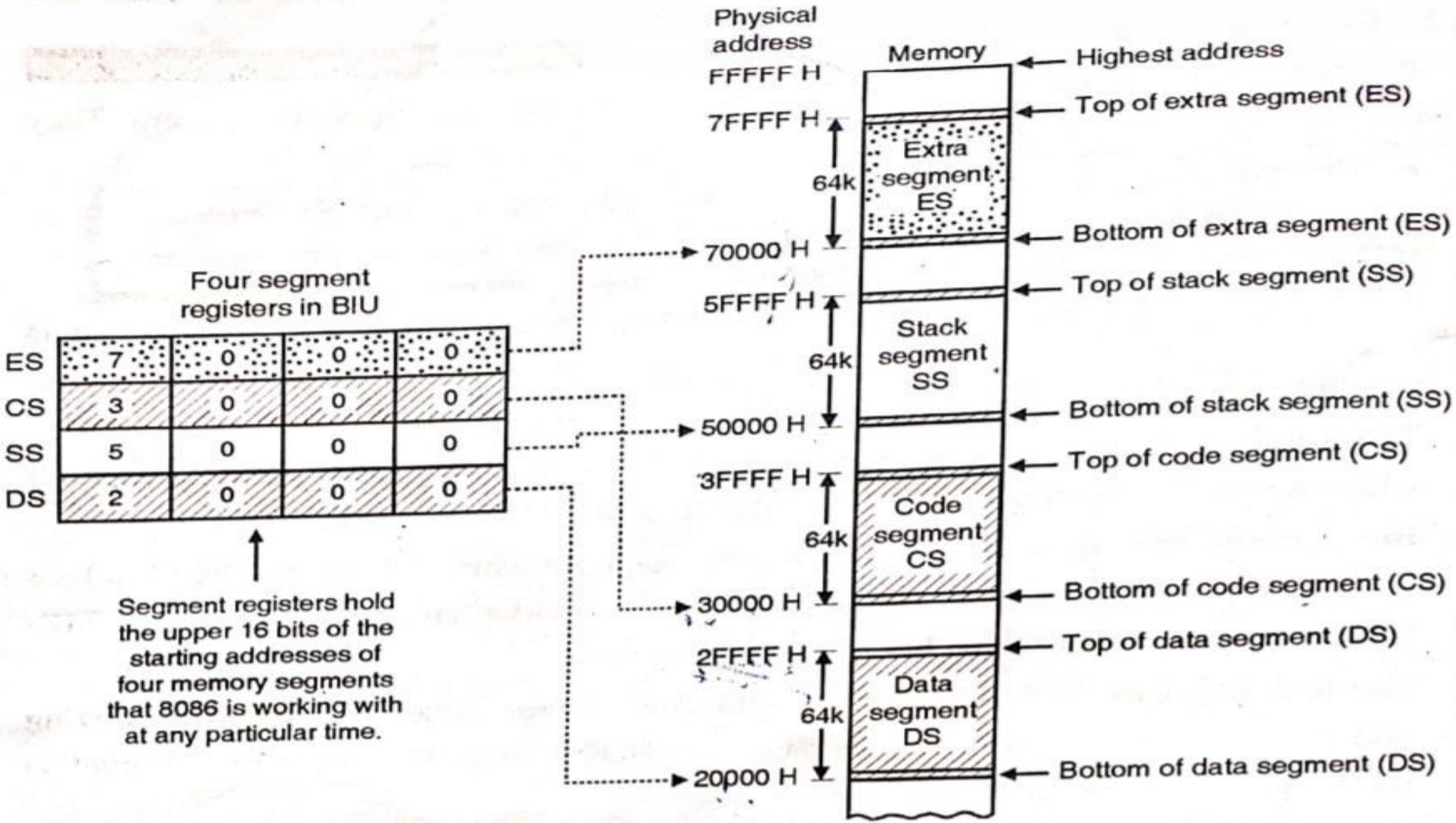
- The EU always reads the next instruction byte from the queue in BIU. This is much faster than sending out an address to the memory and waiting for the next instruction byte to come.
- In short pipelining eliminates the waiting time of EU and speeds up the processing.

Segment Registers

- The BIU contains four special purpose registers called as segment registers. They are :

• The code segment (CS) register	• The stack segment (SS) register
• The extra segment (ES) register and	• The data segment (DS) register

The purpose of using these segment registers and segmentation can be explained as follows :



- All these are 16 bit registers.
- The number of address lines in 8086 is 20. So the 8086 BIU will send out a 20 bit address in order to access one of the 1,048,576 or 1 Mb memory locations.
- But it is interesting to note that the 8086 does not work the whole 1,048,576 byte (1Mbyte) memory at any given time. However it works with only four 65,536 (64K-byte) segments within the whole 1 M-byte memory.
- The four segment registers actually hold (contain) the upper 16 bits of the starting addresses of the four memory segments of 64 K byte each with which the 8086 is working at that instant of time.
- A word is any two consecutive bytes in memory. Word is stored in memory with the most significant byte at the higher memory address. These bytes are stored sequentially from byte 0000H to byte FFFFH.
- Programs view memory space as a group of segments defined by the application.
- A segment is a logical unit of memory that may be upto 64 K bytes long.
- Each segment is made up of contiguous memory locations. It is independent, separately addressable unit.
- Note that these starting addresses will always be changing. They are not fix.
- This concept can be clearly understood by referring to Fig. 2.5.2.
- Fig. 2.5.2 shows one of the possible ways to position the four 64 k byte segments within the 1-M byte memory space of 8086. There is no restriction on the locations of these segments in the memory.
- Note that these segments can be separate from each other as shown in Fig. 2.5.2 or they can overlap.
- Note that the starting address or base address of the data segment is 20000H. The upper 16-bits of this i.e. 2000 are loaded into the data segment register (DS).

Code Segment (CS)

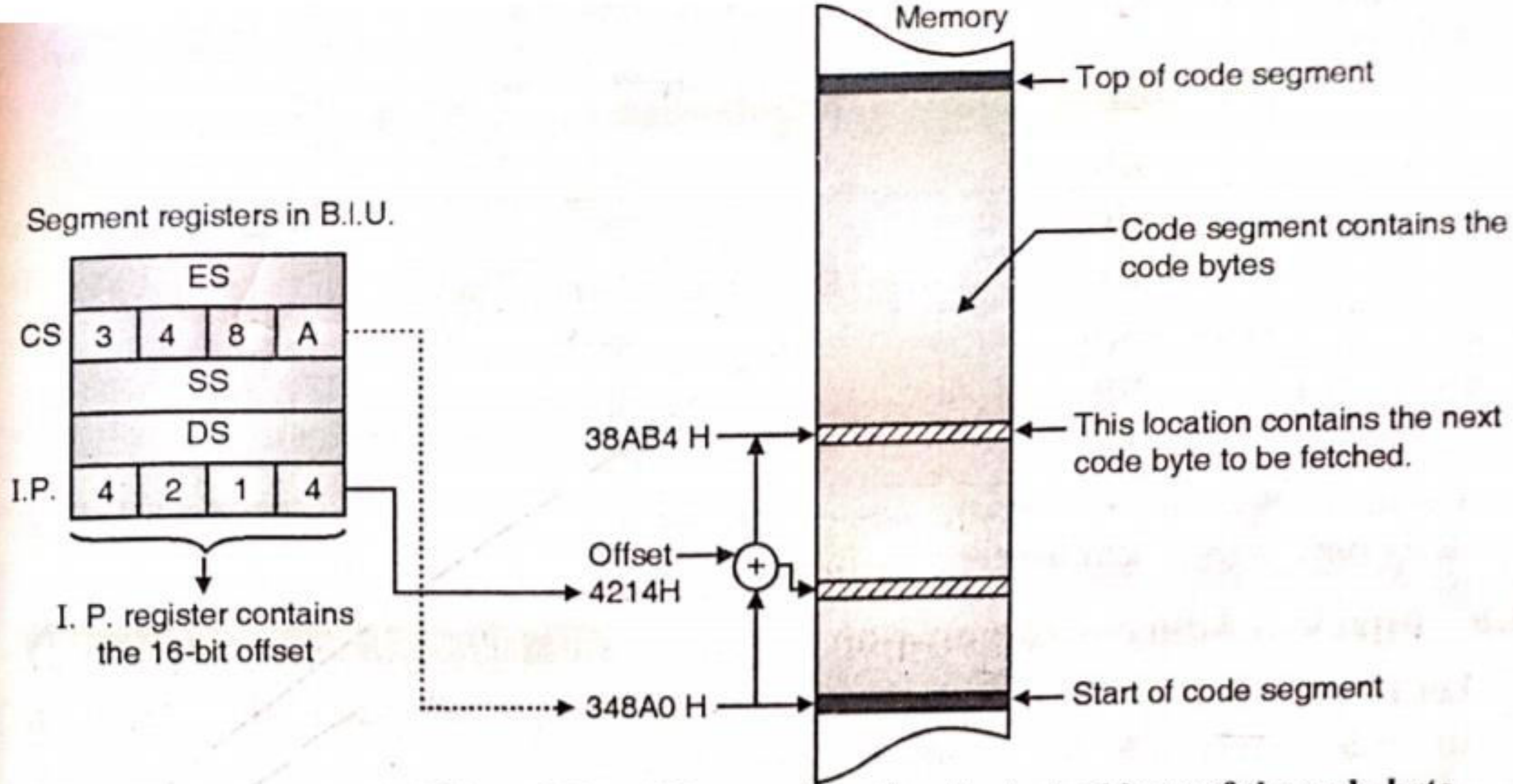
- Code segment (CS) is the part of memory from which the BIU fetches the instruction code bytes.
- The upper 16 bits of the starting address of code segment are held by the code segment (CS) register.
- But the memory address for the base of code segment is 20 bit long. So what about the lower 4-bits ? The answer is that the BIU always inserts zeros for the lowest 4-bits.
- So if the CS register contents are 3428 H then after adding zeros, the physical starting address of code segment becomes 34280 H.
- Therefore a segment will always start at an address with zeros in the lowest 4 bits.

Segment Base : The upper 16-bits of the starting address of a segment, stored in the segment register is called as the segment base.

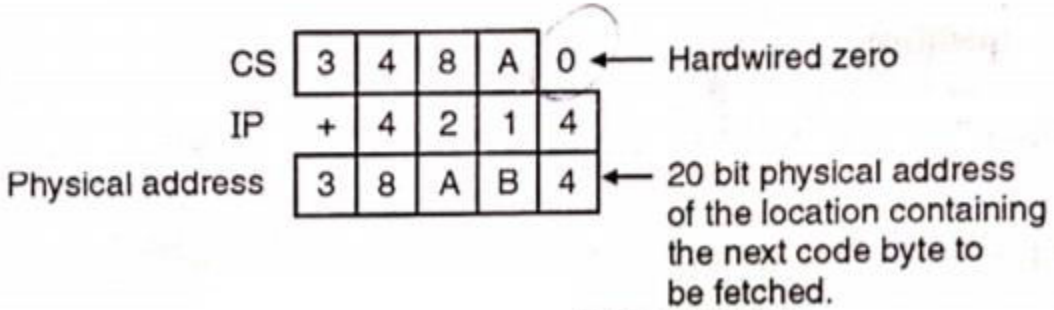
Stack

- In Fig. 2.5.2 we have defined the stack segment from the memory location 50000 H to 5FFFF H.
- The stack is a section of memory which is reserved to store the addresses and data while executing a subroutine program.
- The stack segment (SS) register holds the upper 16 bits of the starting address of the stack area.

Instruction Pointer (IP) Register



Addition of IP to CS to produce the physical address of the code byte



Computation

- Another special purpose register in the BIU is the instruction pointer (IP) register.
- As discussed earlier the code segment (CS) register holds the upper 16 bits of the 20 bit starting address of the code segment. And code segment is the segment from which the BIU is currently fetching the instruction code bytes.
- The instruction pointer (IP) register holds the 16 bit address or **offset** of the next code byte within the code segment. This is illustrated in Fig. 2.5.3(a).

Rules of segmentation

1. If non-overlapping, there can be 16 segment in all, each of 64KB.
   (since, 1MB /64KB = 16)
2. Segments can overlap each other
3. A segment can begin at any location that is a multiple of 10H
4. At any given time a maximum of 4 segments and hence 256KB can be accessed
5. The memory of 8086 is a wrap around memory. This means that once the address generated crosses the 1MB limit, it accesses the location at the top 00000H. For e.g. if CS = FFFF and IP = 0010. Physical address = CS*10H + IP = 100000H. The address lines in 8086 are only 20, so the MSB '1' is discarded and the location being accessed is 00000H.

Physical Address Generation

- Let us now understand the generation of 20 bit physical address of the location in the code segment which contains the next code byte.
- The sequence of operation is as follows.

Physical Address Generation

Step 1 : The CS register contains the upper 16 bits of the starting address of the code segment.

∴ CS Register :

3	4	8	A	Segment base

⇓

Step 2 : The BIU will automatically insert zeros for the lowest four bits of the segment base address to get the 20 bit physical address for the starting of code segment.

∴ Starting address of code segment :

3	4	8	A	0

↑ BIU adds this zero

⇓

Step 3 : The I.P. register contains the offset or distance from this address. The offset here is 4214H.

∴ I.P. Register :

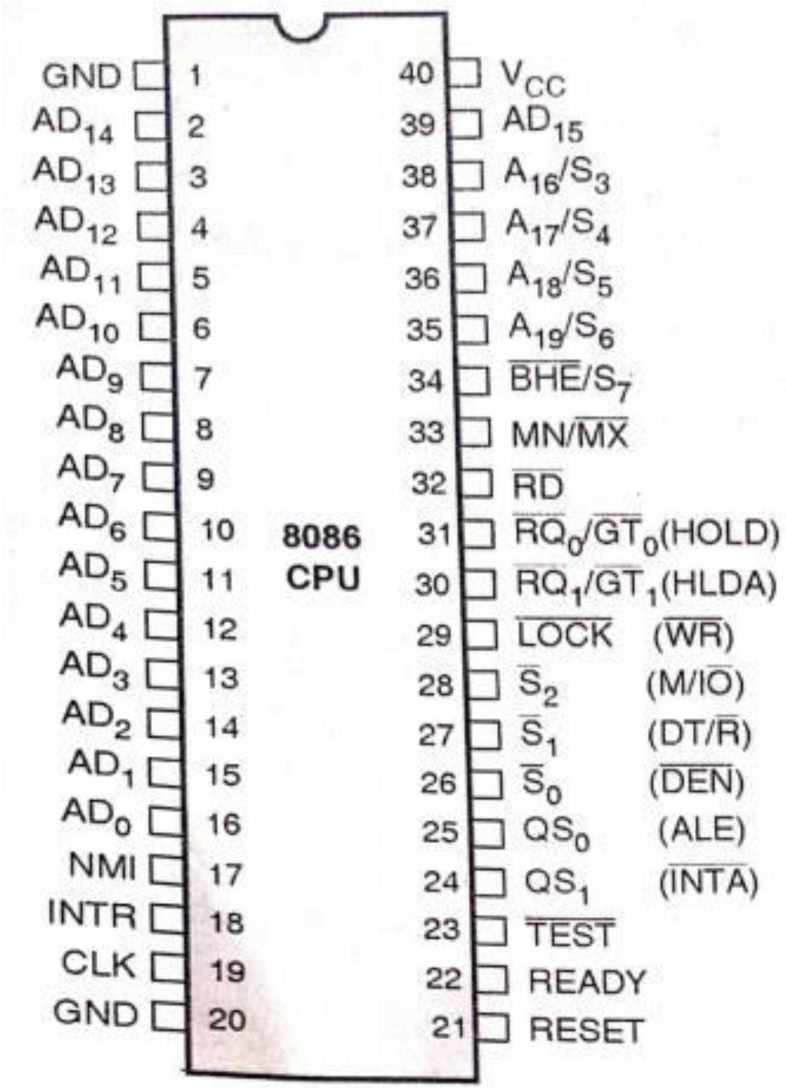
4	2	1	4

⇓

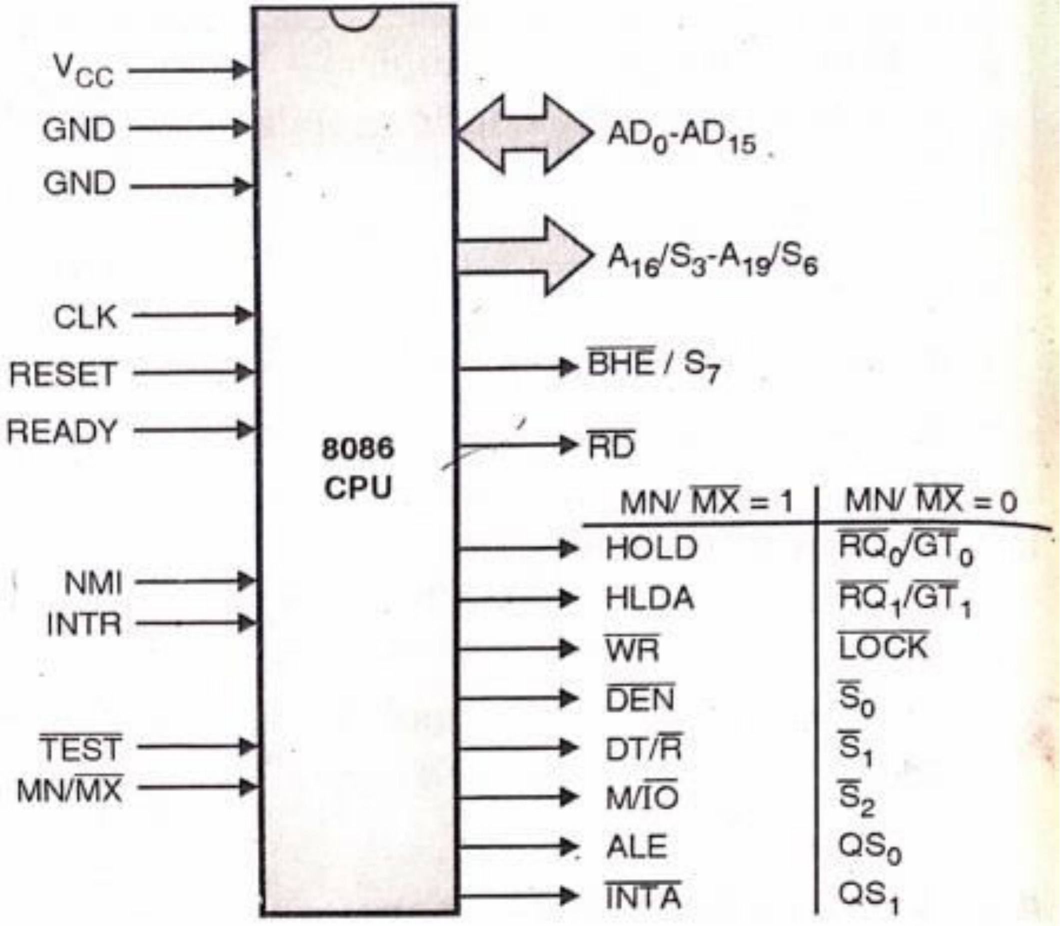
Step 4 : Add the starting address of code segment (20 bit) to the offset to get the physical address of the location containing the next code byte as follows :

Starting address of code segment	→		3	4	8	A	0	← Hard wired 0
Offset in the I.P. Register	→	+		4	2	1	4	
Physical address of the location containing the next code byte	→		3	8	A	B	4	

Pin diagram of 8086



(a) Pin configuration



(b) Functional pin diagram

For Exam

Sr. No	Pins	Name of the pins
1.	Supply pins (3 pins)	V_{CC}, GND, GND
2.	Clock related pins (3 pins)	CLK, RESET, READY
3.	Address and Data pins (21 pins)	$AD_0 - AD_{15}$, $A_{16}/S_3 - A_{19}/S_6$, $\overline{BHE} / S_7$
4.	Interrupt pins (2 pins)	NMI, INTR
5.	Other control (3 pins)	$\overline{TEST}$, MN / $\overline{MX}$, $\overline{RD}$
6.	Mode multiplexed signals (8 pins) (MIN mode – MAX mode signals)	HOLD – $\overline{RQ_0}$ / $\overline{GT_0}$, HLDA – $\overline{RQ_1}$ / $\overline{GT_1}$, $\overline{WR}$ – $\overline{LOCK}$ $\overline{DEN}$ – $\overline{S_0}$, DT / $\overline{R}$ – $\overline{S_1}$ M / $\overline{IO}$ – $\overline{S_2}$, ALE – QS_0 $\overline{INTA}$ – QS_1

1. Supply Pins (3) :

• V_{CC}	• GND	• GND

- Used for power supply i.e. + 5V on V_{CC} w.r.t. GND.
- Two separate GND pins for two layers of 8086 chip, improves the noise rejection.

2. Clock related Pins (3) :

• CLK	• RESET	• READY

CLK

- This pin provides the basic timing for the processor.
- 8086 does not has an on-chip clock generator hence an external clock generator like 8284 is used to provide the clock signal.
- It is asymmetric with 33% duty cycle, TTL clock signal.

RESET

- It causes the processor to immediately terminate its present activity. The 8284 clock generator provides this signal.
- This signal must be active high for atleast 4 clock cycles
- It clears all the flag register, the Instruction Queue, the DS, SS, ES and IP registers and sets the bits of CS register.
- Hence the reset vector address of 8086 is FFFF0H (as CS = FFFFH and IP = 0000H).

READY

- It is an acknowledgement from the addressed memory or I/O that it will complete the data transfer specially meant for slow devices.
- μP samples the READY input between T2 and T3 of a M/C cycle.
- If READY pin is LOW, μP inserts wait-states between T2 and T3, until READY becomes HIGH.

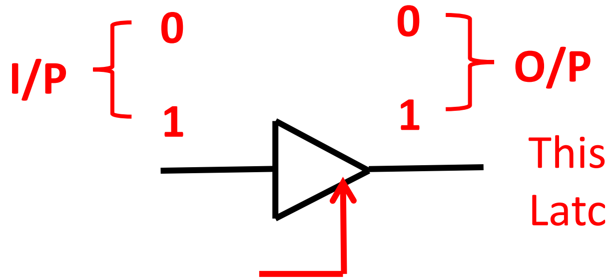
3. Address and Data Pins (3) :

• $AD_{15} - AD_0$	• $A_{16}/S_3 \ldots A_{19}/S_6$	• $\overline{BHE}/S_7$

Buffer

When enable then only output appears on output line

Tristate Buffer



This concept is used in ALE i.e Address Latch Enable (to latch the address

$AD_{15} - AD_0$

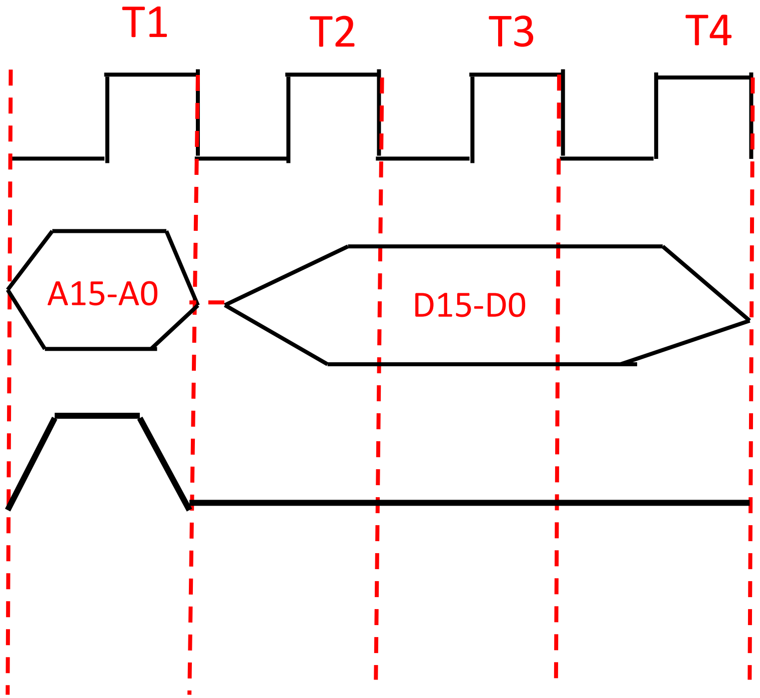
- These are time multiplexed tristate data address lines i.e. for some time they have address and for some time data
- It gives the address $A_{15} - A_0$ during T1 of an Machine Cycle. (When ALE = 1)
- It gives the data $D_{15} - D_0$ after T1 of an M/C Cycle (Machine cycle).

1. Address calculation

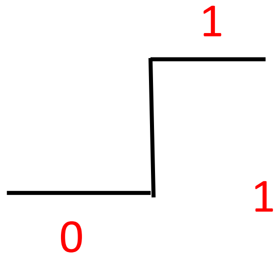
Address Bus

ALE

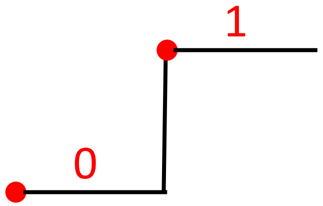
2. Data



How to differentiate between data and address ???



1 clock cycle = 1 T state



Something is high which indicates that bus carries the address.

When ALE = 1 : Address bus (A15-A0)

When ALE = 0 : Data bus (D15-D0)

Pass 0 signal and wait

Pass 1 signal and wait