



SHRI VILEPARLE KELAVANI MANDAL'S
SHRI BHAGUBHAI MAFATLAL POLYTECHNIC



Question Bank for PT1-CH1 – Jun 24 – Dec 24

Program - CSE
Course – FMP

Semester – III
Course code – FMP230806

Ch 1 – The 8086 microprocessor			
	Questions	Answer from	Marks
Q.1	Explain features of 8086 (each 4 mks) – min 4 points in feature	PPT	4
Q.2	Define term Microprocessor - def 2mks , function – 2mks	PPT	4
Q.3	Draw and explain architecture of 8086 Only draw – 4 mks Draw explain any specific block - (each for 2 mks)	PPT	4/6/8
Q.4	Define term pipelining and explain any two advantages of pipelining	PPT	4
Q.5	Draw flag register of 8086 and explain (only draw- 2 mks, any flag explanation with example) each 2 mks	PPT	4/6
Q.6	Explain special functions of general purpose registers (AX,BX,CX,DX) 1. Each for 2 mks (minimum 2 functions),...4/6 2. any one ...3mks (minimum 3 functions),.. 3	PPT	4/6/8
Q.7	Explain use of any pin or signal of 8086 (each for 2 mks)	PPT	4
Q.8	Draw and explain memory bank of 8086. (With all cases)	PPT	8
Q.9	Draw and explain memory segmentation of 8086	---	6
Q.10	Draw and explain significance of queue in 8086	PPT	8
Q.11	When pipeline fails?	PPT	4
Q.12	Explain overflow flag with the help of example – Use of flag – 2mks Range of +ve and -ve numbers – 2mks Example in detail – 2mks	PPT	6
Q.13	Draw functional pin diagram of 8086 only diagram	PPT	4
Q.14	Address calculation		4

Features of 8086

Explain features of 8086 (each for 2 marks)

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

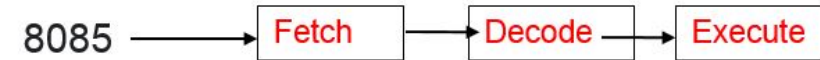
1. Basic Features

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

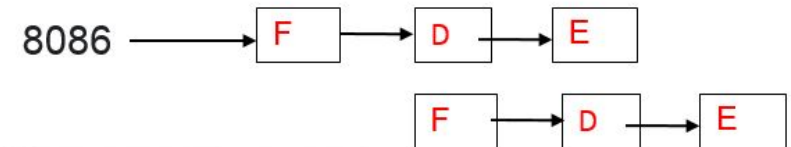
- (1) It is a 16-bit processor. This implies that
 - (a) It has a 16-bit ALU that can perform 16-bit operation simultaneously.
 - (b) It has 16-bit registers and internal data bus.
 - (c) It has 16-bit external data bus.
- (2) It has three versions based on the basis of frequency of operation.
 - (i) 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
$$\begin{aligned} 2^{16} &= 2^6 \times 2^{10} = 64 \times 1 \text{ K} \\ &= 64 \text{ K I/O locations} \end{aligned}$$

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.

3. Miscellaneous Features

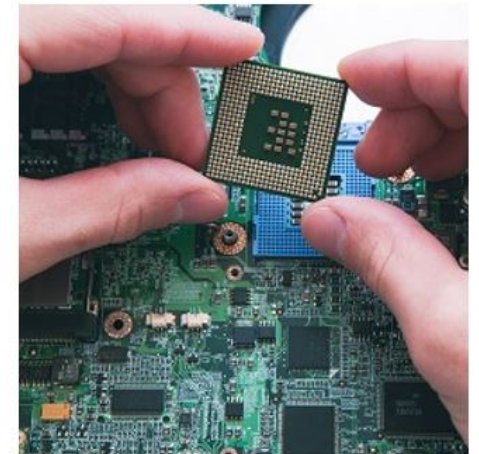
- Interrupts
- Registers
- Instruction set
- Data size for ALU

- (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.

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.

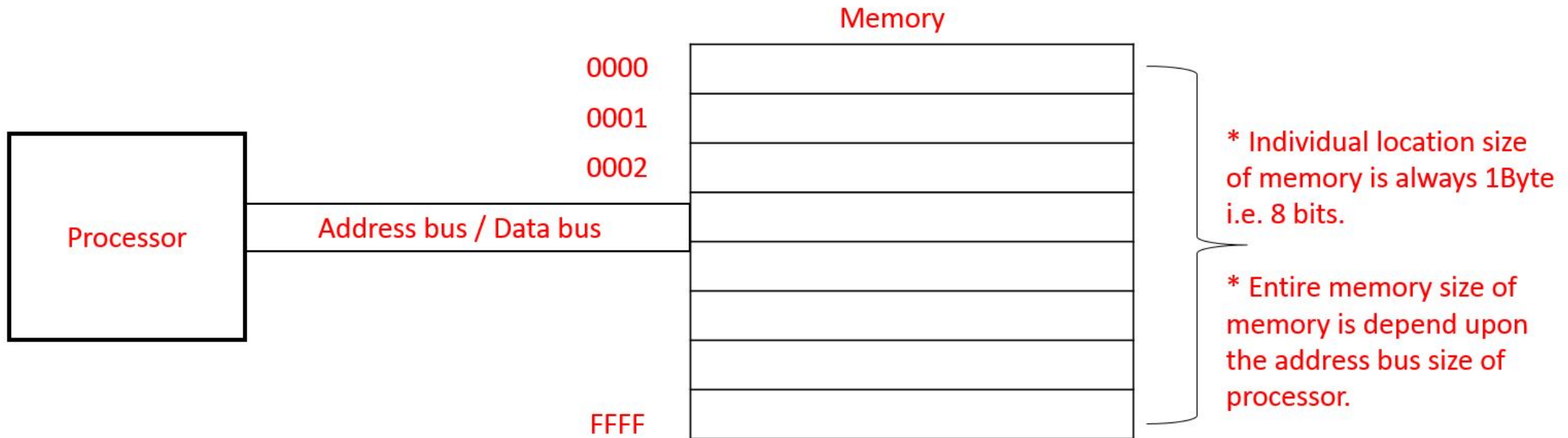


The microprocessor is the central unit of a computer system that performs arithmetic and logic operations, which generally include adding, subtracting, transferring numbers from one area to another, and comparing two numbers. It's often known simply as a processor, a central processing unit, or as a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



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)

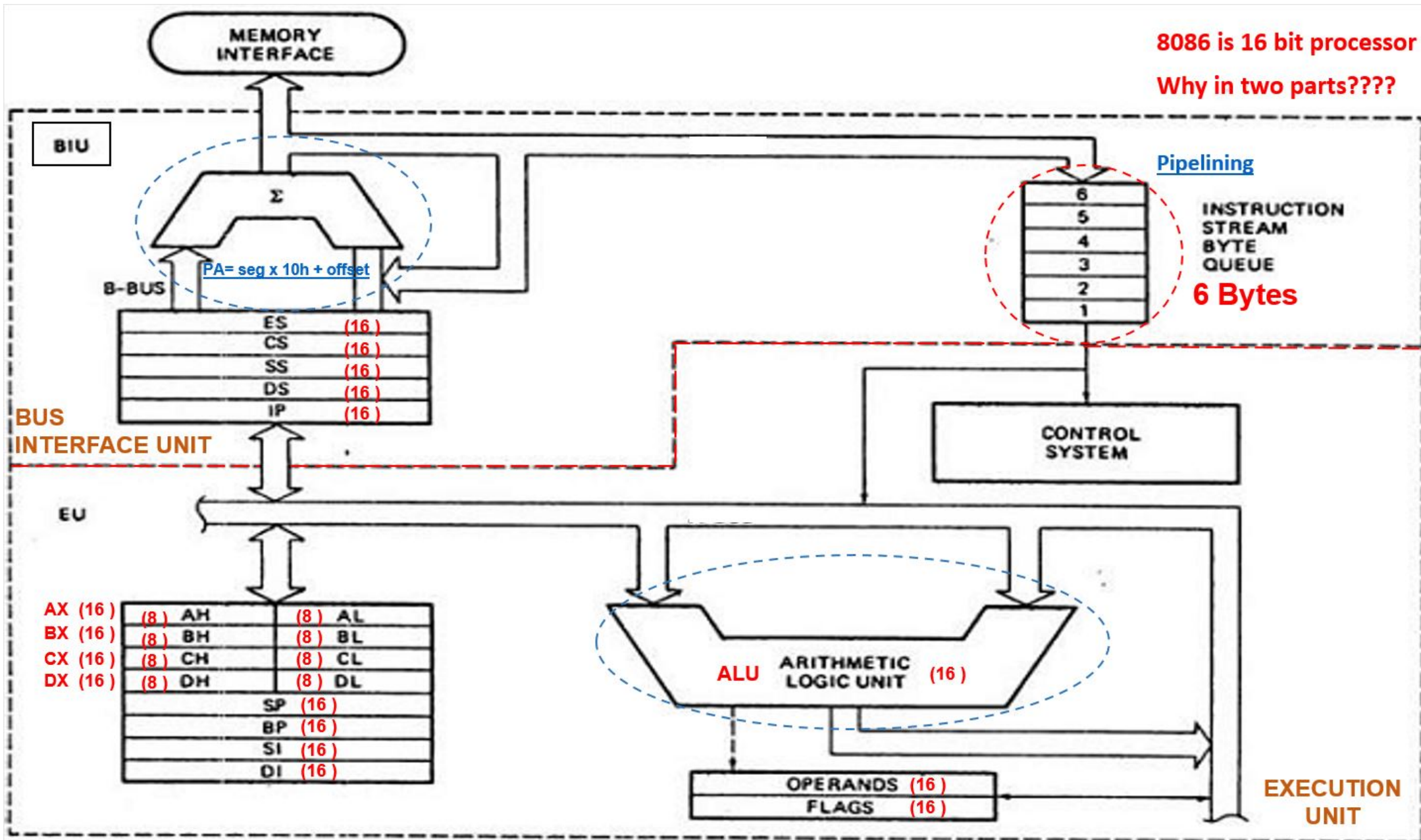


1. Processor calculates the memory address and fetch the content of that memory location.
2. Processor decodes the fetched instruction and execute it.

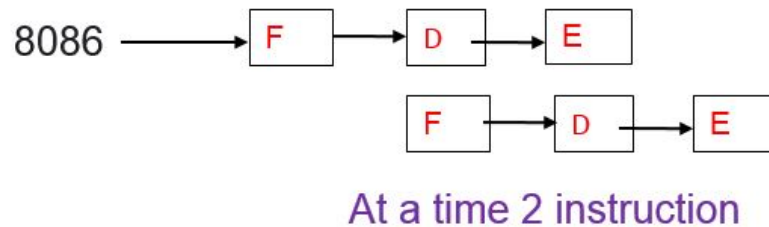
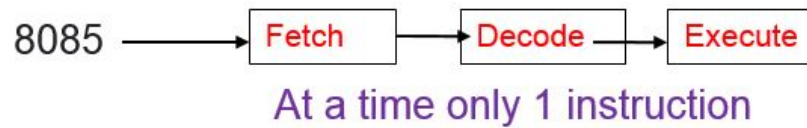
Content of memory location can be Data/instructions

8086 is 16 bit processor

Why in two parts????



Pipelining :



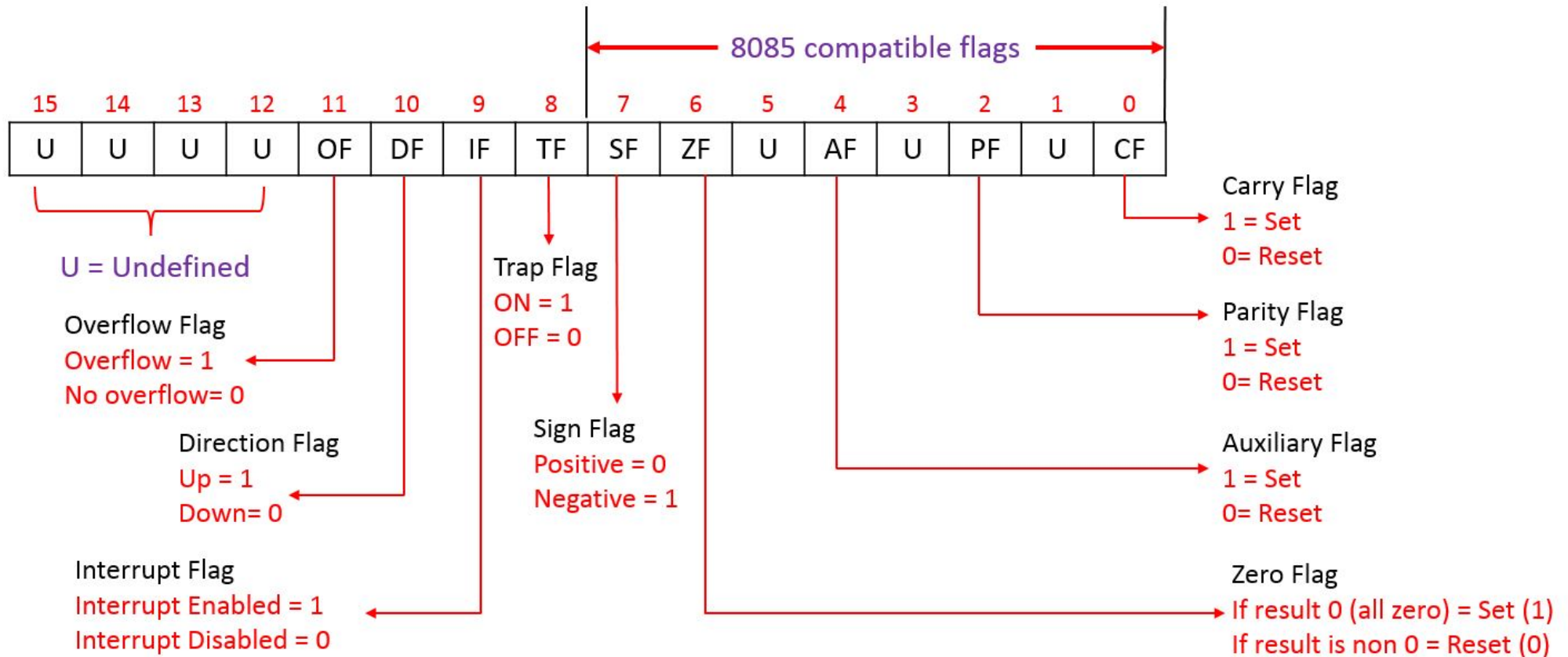
- Pipelining is one of the special feature of 8086 processor.
- Due to pipelining 8086 processor architecture is divided into two parts.
- BIU will fetch the instruction from memory and it will pass that fetched instruction to the Execution unit.
- While execution unit executes the current instruction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

Flag Register of 8086

- Flag register is a part of Execution Unit.
- It is a 16-bit register with each bit corresponding to a flip-flop.
- Flag register is used to give status of operation performed by processor.
- A flag is flip-flop.
- It indicates some condition produced by the execution of an instruction.



Special functions of general purpose registers of 8086

1. Register AX : **Accumulator**

AX is the “16-bit accumulator” while AL is “8-bit accumulator”

Accumulator has the following special functions :

- (i) Some of the operations, such as Multiplication and Division, require that one of the operands be in the accumulator and also the result is stored in accumulator. Some other operations, such as Addition and Subtraction, may be applied to any of the registers (that is, any of the eight general- and special-purpose registers) but are more efficient when working with the accumulator.
- (ii) It works as a via register for I/O accesses i.e. a data is routed through accumulator for the communication of the processor and I/O devices.
For OUT instruction the data in accumulator (AL for 8-bit data and AX for 16-bit data) can only be given to the output device.
For IN instruction the data taken from the input device can be taken only in accumulator (AL for 8-bit data and AX for 16-bit data)
- (iii) It also works as a via register for string instructions. Whenever a data is to be brought from memory or given to memory in case of string operations it is routed through accumulator only.

2. Register BX : **Base**

BX is the “base” register;

- (i) It is the only general-purpose register which may be used for indirect addressing. (various addressing modes are discussed in chapter 4.)
- (ii) For example, the instruction MOV [BX], AX causes the contents of AX to be stored in the memory location whose address is given in BX.

3. Register CX : **Counter**

CX is the “count” register. It works as a default counter register for three instructions viz :

- (i) The looping instructions (LOOP, LOOPE, and LOOPNE), to indicate the number of iterations
- (ii) The shift and rotate instructions (RCL, RCR, ROL, ROR, SHL, SHR, and SAR), to indicate number of shifts or rotations (Here only CL is used and not entire CX)
- (iii) The string instructions (with the prefixes REP, REPE, and REPNE) to indicate the size of the string block.

4. Register DX : **Data**

DX is the "data" register

- (i) It is used together with AX for the word-size MUL and DIV operations, when the operand size is greater than the register AX i.e. operand is 32-bit.
- (ii) It also holds the port number for the IN and OUT instructions. For 16-bit address accesses of I/O ports only DX can be used as a pointer.

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 have 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 at least 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

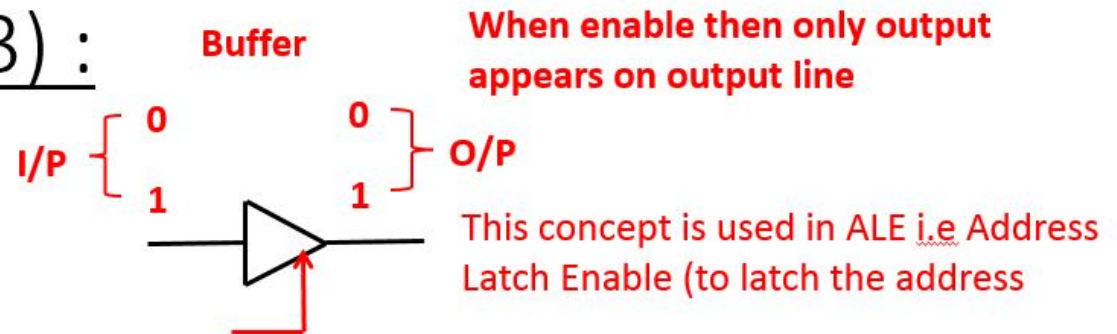
- It is an acknowledgement from the addressed memory or I/O that it will complete the data transfer specially meant for slow devices.
- μ P samples the READY input between T2 and T3 of a M/C cycle.
- If READY pin is LOW, μ P inserts wait-states between T2 and T3, until READY becomes HIGH.

3. Address and Data Pins (3) :

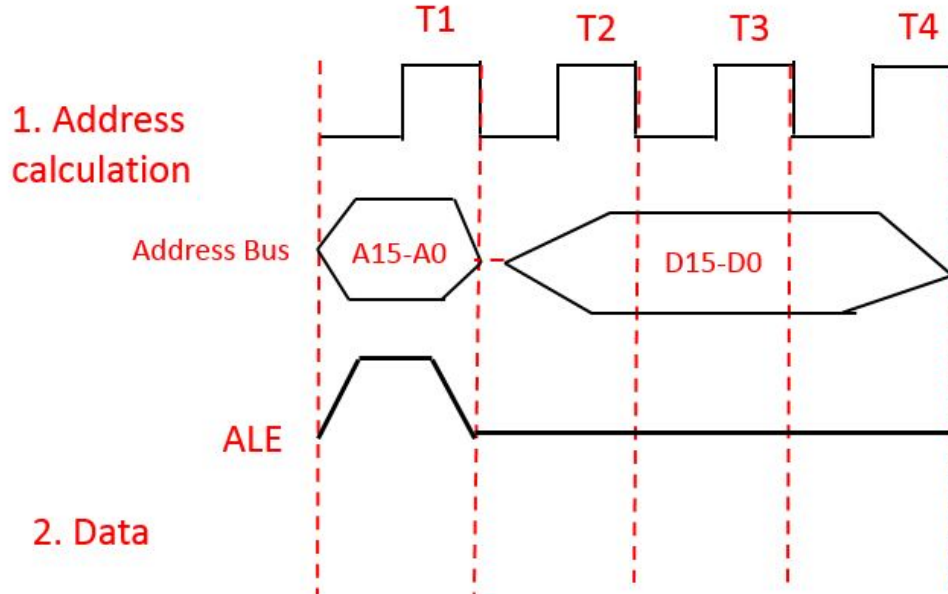
• $AD_{15} - AD_0$	• $A_{16}/S_3 \dots A_{19}/S_6$	• \overline{BHE}/S_7
--------------------	---------------------------------	------------------------

$AD_{15} - AD_0$

Tristate Buffer



- These are time multiplexed 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).

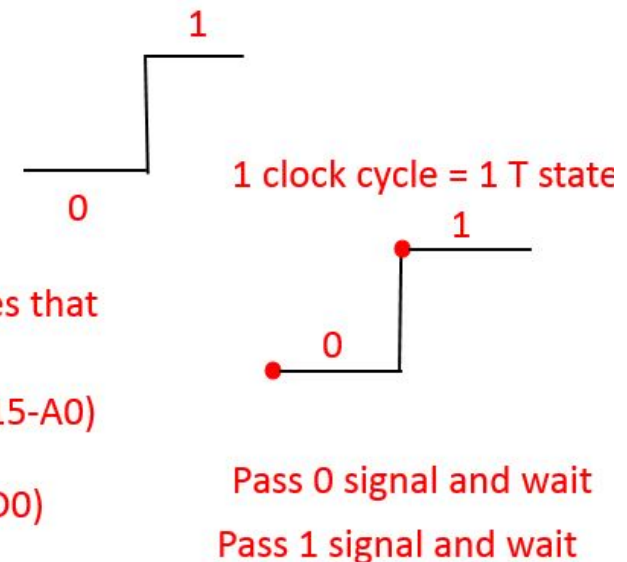


How to differentiate between data and address ???

Something is high which indicates that bus carries the address.

When $ALE = 1$: Address bus ($A_{15}-A_0$)

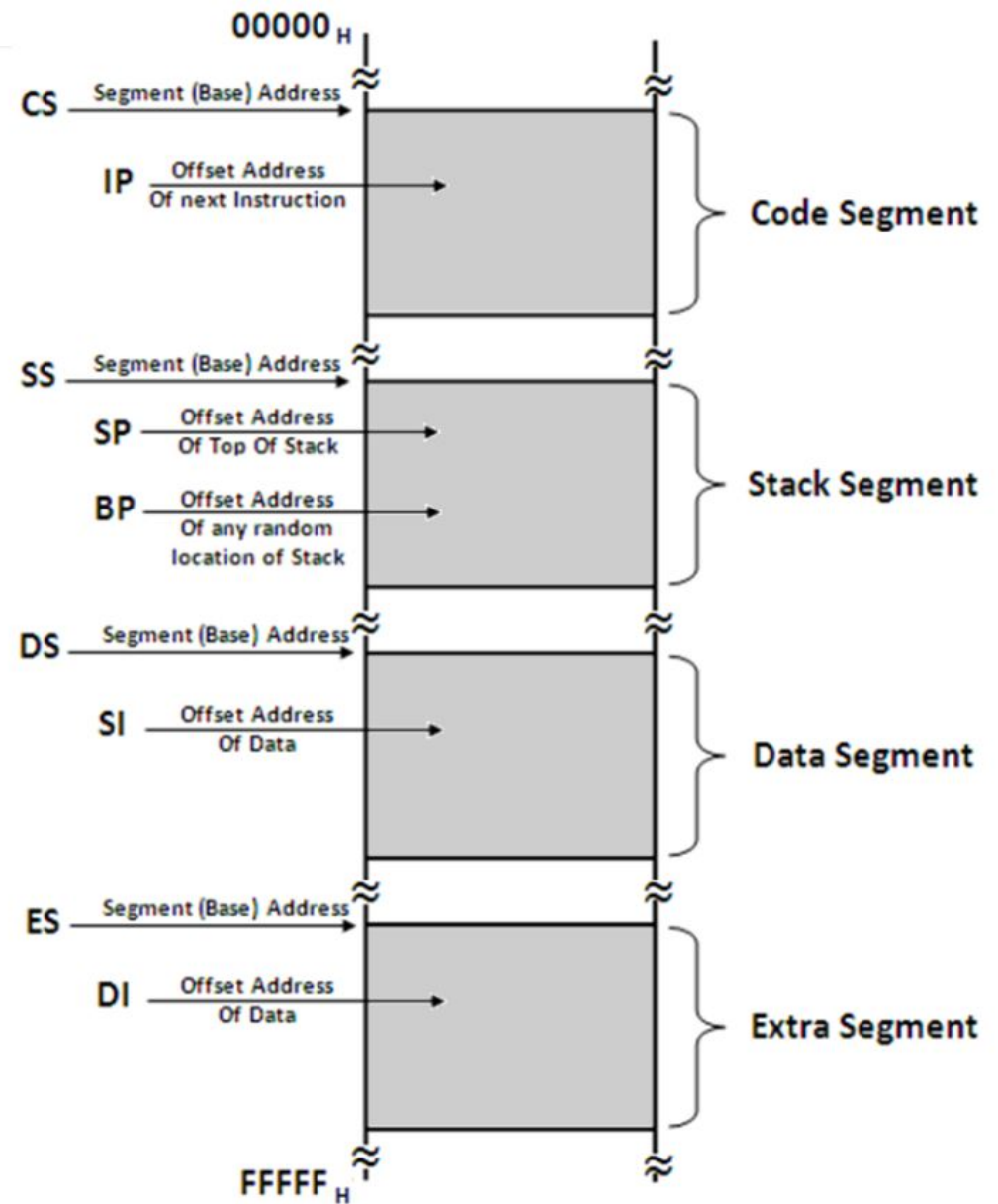
When $ALE = 0$: Data bus ($D_{15}-D_0$)



MEMORY SEGMENTATION IN 8086

NEED FOR SEGMENTATION / CONCEPT OF SEGMENTATION

- 1) Segmentation means **dividing** the memory into **logically different parts called segments**.
- 2) 8086 has a **20-bit address bus**, hence it can access 2^{20} Bytes i.e. **1MB** memory.
- 3) But this also means that **Physical address** will now be **20 bit**.
- 4) It is **not possible** to work with a **20 bit address** as it is **not a byte compatible** number. (20 bits is two and a half bytes).
- 5) To avoid working with this incompatible number, we **create a virtual model** of the memory.
- 6) Here the memory is **divided into 4 segments**: Code, Stack Data and Extra.
- 7) The **max size** of a segment is **64KB** and the **minimum size** is **16 bytes**.
- 8) Now programmer can access each location with a **VIRTUAL ADDRESS**.
- 9) The Virtual Address is a **combination** of **Segment Address** and **Offset Address**.
- 10) **Segment Address** indicates where the segment is located in the memory (base address)
- 11) **Offset Address** gives the offset of the target location within the segment.
- 12) Since both, Segment Address and Offset Address are **16 bits each**, they both are **compatible numbers** and can be easily used by the programmer.
- 13) Moreover, **Segment Address** is given **only in the beginning** of the program, to initialize the segment. Thereafter, we **only give offset address**.
- 14) **Hence we can access 1 MB memory using only a 16 bit offset address for most part of the program. This is the advantage of segmentation.**
- 15) Moreover, dividing Code, stack and Data into different segments, makes the memory **more organized and prevents accidental overwrites** between them.
- 16) The **Maximum Size** of a segment is **64KB** because **offset addresses** are of **16 bits**.
 $2^{16} = 64KB$.
- 17) As max size of a segment is 64KB, programmer can create **multiple Code/Stack/Data segments** till the entire 1 MB is utilized, but **only one of each type** will be **currently active**.
- 18) The physical address is calculated by the microprocessor, using the formula:
PHYSICAL ADDRESS = SEGMENT ADDRESS \times 10H + OFFSET ADDRESS
- 19) Ex: if Segment Address = 1234H and Offset Address is 0005H then
Physical Address = $1234H \times 10H + 0005H = 12345H$
- 20) This formula automatically ensures that the **minimum size of a segment is 10H bytes** (10H = 16 Bytes).

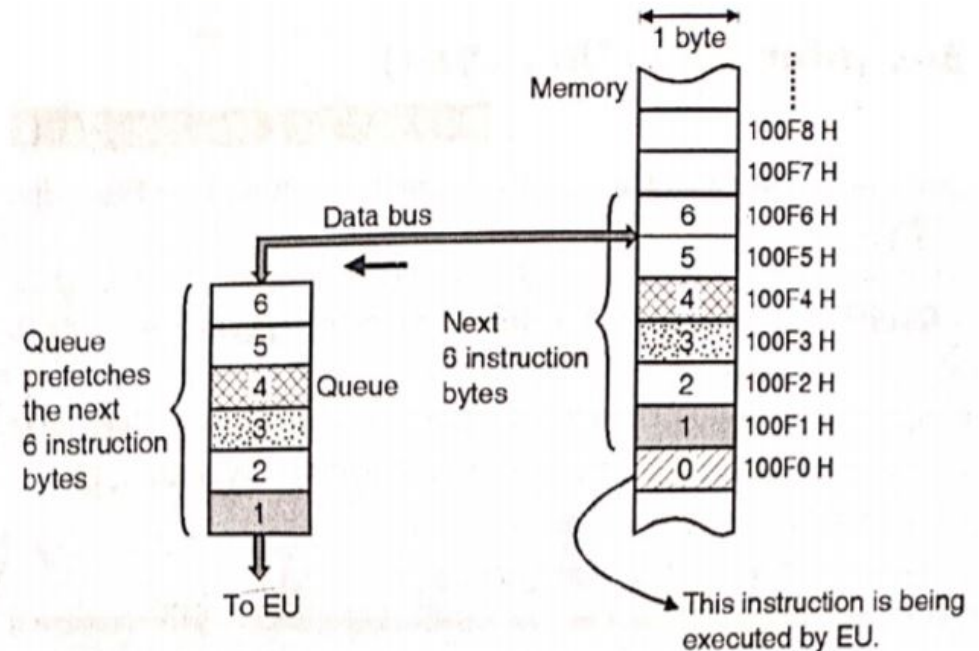


The Instruction Queue

- 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."

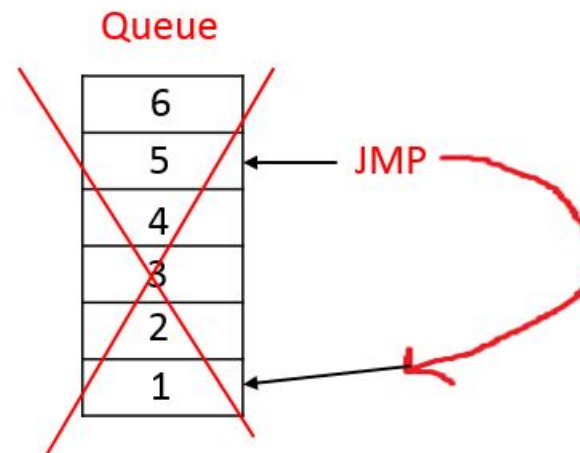
Significance of Queue

- 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.



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.



Overflow Flag :


Overflow flag matters only for signed numbers

Range for Positive numbers (0 to 127)




00	0000 0000
01	0000 0001
7F	0111 1111

Range for Negative numbers (- 128 to -01)



80	1000 0000
FF	1111 1111



01	0000 0001
-01	1111 1111
80	1000 0000
-80	1000 0000

Range for Positive numbers : 00 to 7F

Range for Negative numbers : FF to 80

After addition if result is going beyond above ranges then overflow flag is set i.e. OF = 1

Example for overflow flag:

Overflow flag matters only for signed numbers

$$\begin{array}{r}
 23 \text{ h} \quad 0010 \ 0011 \\
 + 31 \text{ h} \quad 0011 \ 0001 \\
 \hline
 54 \text{ h} \quad 0101 \ 0100 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 0 \quad 0 \quad 0 \quad 0
 \end{array}$$

$$\begin{array}{r}
 -23 \text{ h} \quad 1101 \ 1101 \\
 + -31 \text{ h} \quad 1100 \ 1111 \\
 \hline
 -54 \quad 1010 \ 1100 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 1 \quad 1 \quad 0 \quad 1
 \end{array}$$

$$\begin{array}{r}
 27 \text{ h} \quad 0010 \ 0111 \\
 + 39 \text{ h} \quad 0011 \ 1001 \\
 \hline
 60 \text{ h} \quad 0110 \ 0000 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 0 \quad 1 \quad 0 \quad 0
 \end{array}$$

$$\begin{array}{r}
 -27 \text{ h} \quad 1101 \ 1001 \\
 + -39 \text{ h} \quad 1100 \ 0111 \\
 \hline
 -60 \text{ h} \quad 1010 \ 0000 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 1 \quad 1 \quad 0 \quad 1
 \end{array}$$

$$\begin{array}{r}
 42 \text{ h} \quad 0100 \ 0010 \\
 + 43 \text{ h} \quad 0100 \ 0011 \\
 \hline
 85 \text{ h} \quad 1000 \ 0101 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 0 \quad 0 \quad 1 \quad 0
 \end{array}$$

$$\begin{array}{r}
 -42 \text{ h} \quad 1011 \ 1110 \\
 + -43 \text{ h} \quad 1011 \ 1101 \\
 \hline
 -85 \text{ h} \quad 0111 \ 1011 \\
 \hline
 \text{CY} \quad \text{AC} \quad \text{OF} \quad \text{P} \\
 1 \quad 1 \quad 1 \quad 1
 \end{array}$$

1. Using MSB bit we can identify whether number is +ve or -ve
2. If MSB is 1 it means number is -ve
3. But sometimes it will give wrong sign bit
4. In such cases checking only MSB is not sufficient
5. We have to check range of both numbers
6. If number cross range it means there is overflow problem

Range for Positive numbers : 00 to 7F (0 to 127)

Range for Negative numbers : FF to 80 (-80 to -01)

Example :

$$\begin{array}{r}
 7\text{F h} \quad 0111 \ 1111 \\
 + 01 \text{ h} \quad 0000 \ 0001 \\
 \hline
 80 \text{ h} \quad 1000 \ 0000 \\
 \hline
 \uparrow
 \end{array}$$

Result is positive and answer gives sign bit negative.

