Ajay Kumar Garg Engineering College, Ghaziabad

Department of Electrical & Electronics Engg.

ST-2 Model Solution

Course: B.Tech
Session: 2017-18
Subject: Micro-Processor & its applications

Semester: V
Section: EN-1, EN-2
Sub. Code: NEE-504

SECTION-A

A.(1) Identify the contents of the Accumulator & flag status as the following Instructions are executed.

```
MVI A, 7FH
ORA A
CPI A2
```

Solution: Here A = 7FH = 0111 1111 H

Now Making OR Operation with the contents of Accumulator

Provides A = 0111 1111 H After second Instruction.

In third Instruction, Accumulator is to be Compared with Immediate Value A2H ie. 1010 0010 H which is Higher then 7F ie 0111 1111 H.

Hence: Accumulator Contains = 7FH.

& flags affected. is carry flag which is to be set By this Comparision in PSW.

Hence CY = 1 P(even) = 0, AC = 0, Z = 0

S = 0

(2) If the CS registus Contains the Number 5ACEH & IP Contain = FA3CH, what will Be the Address of Instruction.

Solution: Physical Address = 10H segment Value + IP Value.

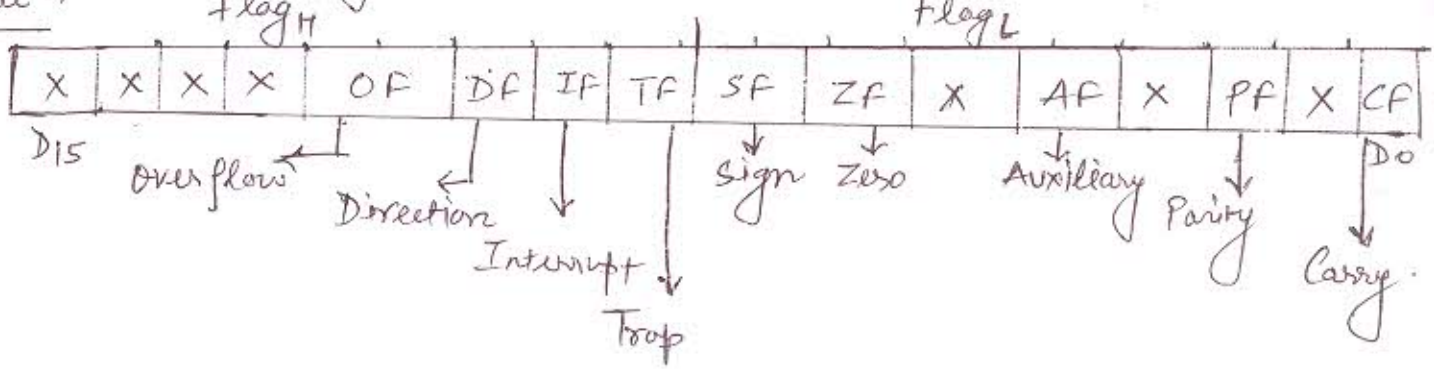
Hence $5ACE \times 10_H + FA3CH = P.A.$

$5ACE0 + FA3C = P.A.$

Physical Address = 6A71CH Ans

(3) Draw the flag register of 8086 MPU.

Solu^{n}:



(4) Differentiate b/w PC & SP.

Ans:- PC stands for Program Counter, which stores the Address of next Instruction to be fetched/Executed by Processor (8085), It is a 16 Bit register.

While SP stands for Stack Pointer, which is also a 16 Bit Register & used for temporarily storing the current value of Program Counter / PSW, in case of Subroutine call/ Branching Instructions.

(5) What is Instruction Queue? & what happens if the Branch Instruction comes?

Ans:- In 8086 Processor's we do have special feature of Pipelining of Data, this Pipelining is made possible with this Queue, which is a type of stack but here in this case it is operating As first in first out (FIFO).

In 8086 Queue is of 6 Bytes length ie while executing the current Instruction Processor Prefetches the next 6 Bytes of Code.

But when Branching Instruction is executing it is causing a change in sequence of execution of Program, Hence the Prefetched data in Queue is to be Dumped at this Instant.

PRASANT KESARWANI	Checked By	DR. BHUPAL SINGH
(Name & Signature)	(Name & Signature)	Hod,EN

Section B

(B)

(6) Sixteen Bytes of Data are stored in memory locations at 0050H to 0050F, Transfer Entire Block of Data to new Memory locations starting at 0070H.

Solution:

```
        MOV SI, 0050H
        MOV DI, 0070H
        MOV CX, 10H
NEXT:   MOV AL, [SI]
        MOV [DI], AL
        INC SI
        INC DI
        LOOP NEXT
        HLT
```

(7) Explain Various Addressing Modes of 8085 µP with Suitable Examples.

Ans: **Addressing Modes:** Every Instruction performs an operation on the specified data called operand.

Every Instruction in particulars requires an operand to be specified for that Instruction to be executed.

this operand can be available in a general purpose Reg, in Accumulator or in a Memory Location. the way in which the the operand is specified for an Instruction is called addressing Mode.

Various addressing Modes in 8085 µP Are:

1. Direct Addressing Mode
2. Registus Addressing Mode.
3. Registus Indirect Addressing Mode.
4. Immediate Addressing Mode.
5. Implicit Addressing Modes.

1. Direct Addressing Mode:- In this mode the address of operand (Data) is given in the instruction itself.

for example:-

STA 2400H — store the content of Acc. to memory location 2400H

IN 02 — Read Data from Port C, 02 is the address of the port C, of an I/O Port from where the data is to be read.

2. Registers Addressing Mode:- In Registers addressing Mode the operands are available in the general purpose Registers.

for ex:- MOV A, B Move (Copy) Contents of B to Acc.

or ADD B Add Contents of B to ACC.

2. Register Indirect Addressing Mode:-

Here in this mode of addressing, the address of operand is specified by a register/register pair.

for example:- LXI H, 2500H Load HL Pair with 2500H

MOV A, M Move Contents of Memory Location to Acc. whose Address is given in HL-Pair Reg.

4. Immediate Addressing:- In this mode the operands are specified within the instruction itself.

for ex.:- MVI A, 05 MOV 05 in the Accumulator.

ADI 06 Add 06 to the Contents of Accumulator.

5. Implicit Addressing:- there are certain instructions which operates directly on the Contents of Accumulator. & do not requires the address of the operands,

for examples:-

1. CMA — Compliment the Contents of Acc.
2. RAR — Rotate Acc Right with Carry By one Bit with carry
3. RAL. — Rotate Contents of Acc Left By one Bit.

(8) Explain the following Instructions :-

(1) MOV CS: Total [BP], AX —:
this Instruction Copies AX to two memory Locations, AL to first Location, AH to second, effective address, EA is the sum of displacement represented By total & Contents of BP.
Physical Address = EA + CS

(2) MOVSB —: Mov string Byte [or string Byte wise]
Syntax: MOVSB Destination string Name, Source string Name.
this Instruction Copies a Byte from a location in DS to a location in ES. for this the offset of destination must Be Contained in DI Register & offset Source must Be Contained in SI Register.

(3) XCHG —: Syntax: XCHG AX, 1600
this Instruction exchanges the Contents of destination (first) & Source (second) operands, these operands could Be Two GPR's or a register and a memory Location, If a memory operand is referred, the Processor's have a restriction for this Instruction that segment registers can not Be used with this Instruction.

(4) AAA —: (ASCII Adjustment After Addition)
this Instruction Converts the result of the addition of two valid unpacked BCD digits to a valid 2-digit BCD Numbers & takes the AL register as its Implicit operand.
the two operands of the addition must have its lower 4 Bits Contain a Numbers in the range from 0-9, the AAA Instruction then Adjust AL so that, it Contains a Correct BCD digit, If the Addition Produces a Carry (CF = 1), the AH register is Incremented and the Carry flag CF & Auxiliary Carry flag AF Are set to 1, If the addition does not Produces a decimal Carry, CF & AF Are cleared to 0 & AH is not altered, In Both Cases the Higher 4 Bits of AL Are cleared to '0'.

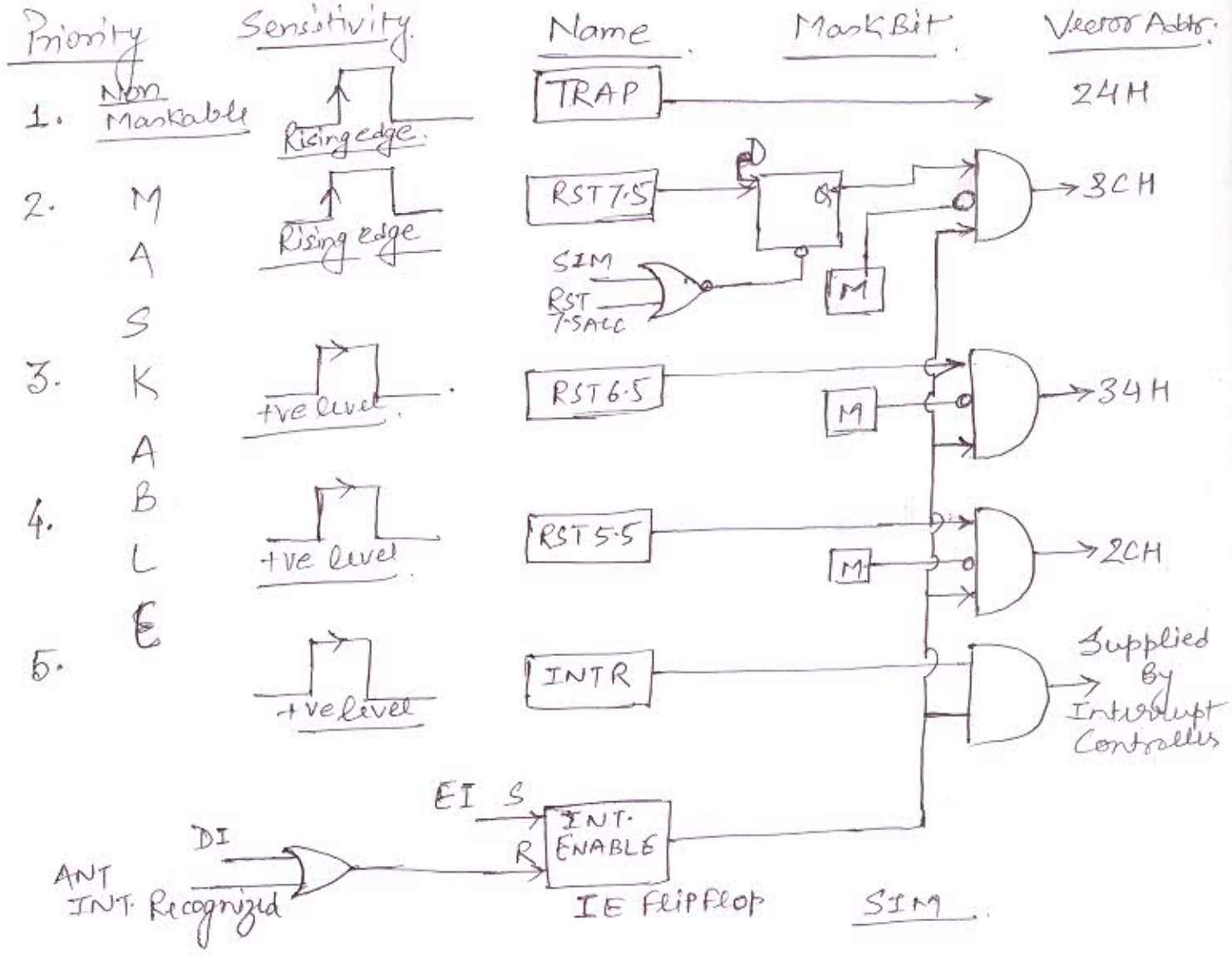
Ex. MOV AH, 00 — clear AH for MSD
MOV AL, 06 — BCD 6 in AL
ADD AL, 05 — Add BCD 5 in AL
AAA — AH = 1 & AL = 1 representing BCD 11.

(5) OR BL, AL :-

It performs the logical OR operation b/w the contents of two registers BL & AL & result of OR operation is stored in BL registers (ie destination) registers.

(9) Explain all the Vectored Interrupts of 8085 MPU & provide their Vector Addresses.

Ans:- Interrupt structure of 8085 :- there Are Basically 5 Interrupts for 8085 MPU, each of the Interrupts Are having a Specified Address & Mask Bit except TRAP. the Interrupt Structure of 8085 is as Below:



(10) Explain the function of following pins of 8086 µP.

Ans :- (i) $\overline{S_3}$ & $\overline{S_4}$:- these are pin No 38 & 37 Respectively in 8086 Pin Discription. also these pin may carry Address Bit $A_{16}-A_{17}$ during ALE=1, But when ALE=0, then S_3 & S_4 Indicates the segment Being Accessed during Current Bus cycle.

S_4	S_3	Function
0	0	Extra segment
0	1	Stack Segment
1	0	Code Segment
1	1	Data Segment

(ii) M/$\overline{IO}$:- this pin is specified in minimum Mode & is 28th pin on 8086 DIP.
this pin Indicates whether the address Bus Contains a memory Address or an I/O Port Address. ie 1 for memory & 0 for I/O Port.

(iii) $\overline{DEN}$:- this is 26th pin on 8086 DIP. & is a minimum Mode operating Pin. this DEN ie Data Enable Bus is Active Low in nature & when there is 0 Available on this pin it Activ -ates External data Bus Buffers.

(iv) $\overline{TEST}$:- this is 23rd pin on 8086 DIP, test pin is an i/p that is tested by the wait Instruction. If the test Pin is at logic 0 the wait Instruction functions as NOP. If test Pin is at logic 1, the wait Instruction waits for test to Become Logic 0.

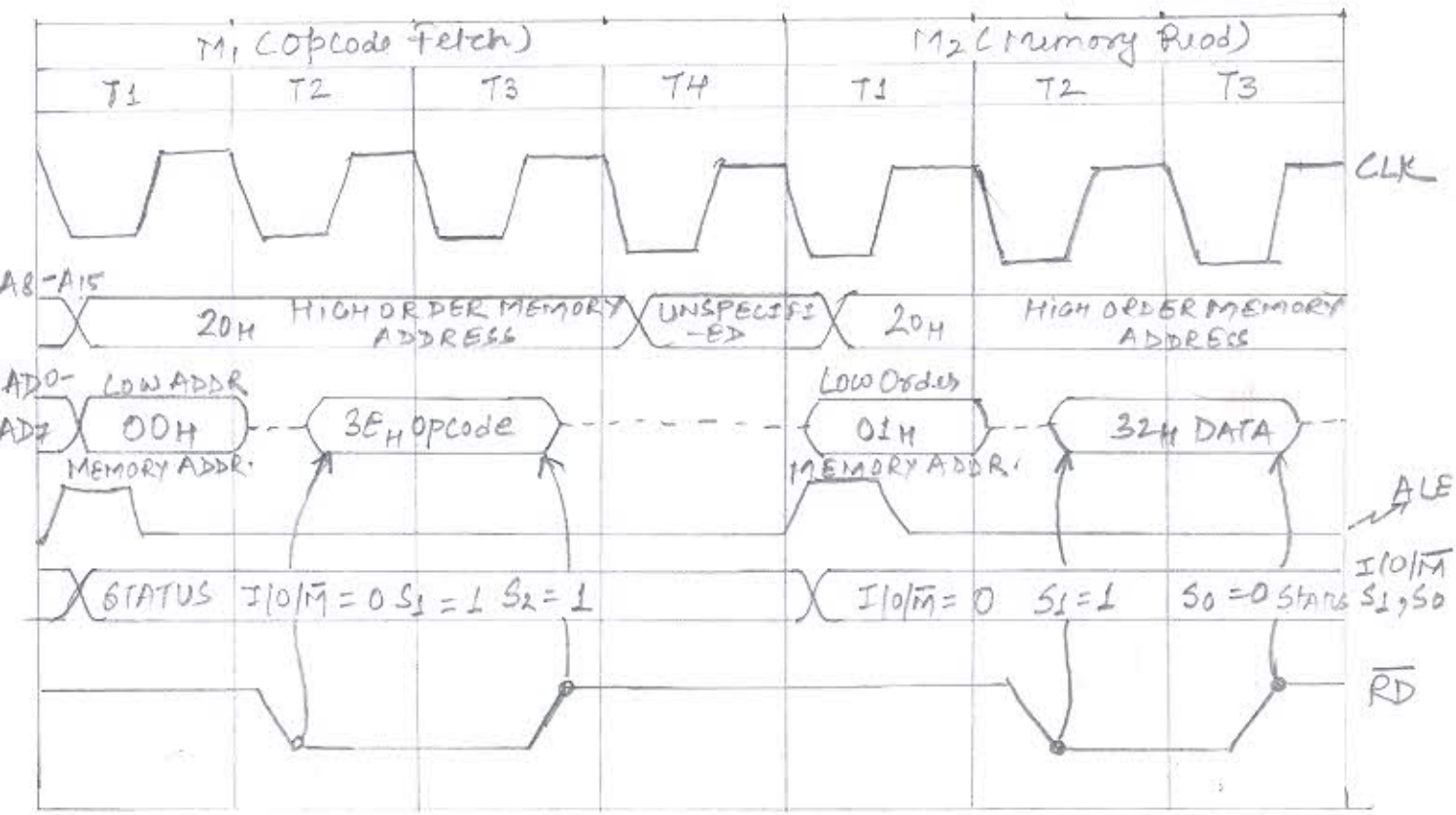
(V) LOCK :- Lock output is 29th pin in Maximum Mode and is used for locking peripheral of system.

Section C

(11) List the sequence of events that occurs when 8085 MPU reads from memory & draw the timing of the memory Read cycle.

Ans:- for Explaining this Memory Read operation let us consider two machine codes - 0011 1110 (3EH) & 0011 0010 (32H) - Are stored in memory locations 2000H & 2001H Respectively. Here the first m/c Code (3EH) represents opcode to load data Byte in Accumulator & second Code (32H) represents data Byte to be loaded in Accumulator & the Instruction is given as

MVI A, 32H



8085 Timing for Execution of the Instruction MVI A, 32H

Sequence of events to take place during Memory Read:-

Here the Instruction Consists of two Bytes; the first is opcode & second is the data Byte, 8085 needs to read these Bytes from Memory & thus requires at least two machine cycles. first M/C is opcode Fetch & second m/c cycle is Memory Read, Here 7 T-States will be involved & sequence of events taking place is as follows:-

1. the first M/C cycle M1 (opcode fetch) is identical in Bus timing with M1C cycle for transfering the Data Byte from memory to MPU.

IN M1 Cycle :-

At T1 :- The processor Identifies that it is an opcode fetch cycle By Placing 011 On status Signal ($IO/\overline{M} = 0, S_1 = 1, S_0 = 1$). It places the memory Address (2000H) from program counter to the Address Bus. Here 20H on A15-A8 & 00H on $AD_7 - AD_0$ & Increments the PC to 2001H, for pointing on to next Instr. Here with as ALE signal goes High (ie ALE=1) during T1, Hence Address (lower) is latched from Bus ($AD_7 - AD_0$).

At T2 :- 8085 Asserts the $\overline{RD}$ control signal which enables the memory O/P Buffer & memory places Data Byte 3EH from Location 2000H on to the data Bus. then 8085 places the opcode in the Instruction Register & disables $\overline{RD}$ Signal, Here the fetch (opcode) is completed in stat T3.

at T4 :- during T4 the 8085 decodes the opcode & finds out that a second Byte needs to be read Yet. after T3 state, the contents of the BUS A15-A8 are unknown & data Bus $AD_7 - AD_0$ goes into High-Impedance state.

IN M2 :- (at T1) After completing the opcode fetch 8085 places the address 2001H on the address Bus & Increments the Program Counter to next Address 2002H, the second M/C Cycle M_2 is Identified as Memory read Cycle (ie. $IO/\overline{M} = 0, S_1 = 1, S_0 = 0$) & ALE is Asserted at T_2, the $\overline{RD}$ Signal Becomes active & enables the chip (Memory)

At T2 :- At rising edge of T_2, the 8085 Activates the data BUS as an I/P Bus, memory places the data Byte 32H on the Data Bus & 8085 Reads & Stores the Byte in Accumulator during T3.

the execution times of Memory read M/C & Instruction cycle Can Be Calculated, If the frequency (CLOCK) is given.

Q.12:- Draw the Architecture of 8086 MPU & Explain its Registers Organization

Ans:- Basically Architecture of 8086 is subdivided into two main units called BIU (Bus Interface Unit) & EU (Execution Unit)

Lets Understand Each one Separately.

BIU (BUS Interface Unit) :-

Bus Interface Unit specifically Deals with memory & I/O Ports ie from where the Processor is fetching the data

As shown in figure BIU is comprising of An Arithmatic ckt Interfacing Buses, 6 Bytes Queue & some segment Registers. like CS (Code segment), DS (Data Segment), ES (Extra segment) SS (Stack Segment) & IPC (Instruction Pointer).

Since 8086 is a 16 Bit Processor the data Bus is of 16 Bit word length & Address Bus is of 20 Bits. Hence it can Access upto 2^{20} = 1MB Physical Memory locations.

Role this Arithmatic Unit is to Calculate the physical Address of the location from where the data is to Be fetched.

While 6 Bytes Queue provides a special feature to this Proc -essor as pipelining. By Prefetching the next 6 Bytes of data & storing in the Buffer.

EU (Execution Unit) :- this Unit Separately performs its task of executing the current Instruction. the main feature of this 8086 Architecture is that the reduced execution time, due to prefetching of data while execution.

Hence for this to Acomplish following Are main components of EU :-

(1) Control System.

(2) General purpose Registers
(3) Arithmatic & Logic Unit (ALU)
(4) flags Registers.
(5) Internal BUS.

Registes Organization of 8086: 8086 is a 16-Bit MPU, Hence Most of the Registers Involved Here A-x of 16-Bit, But as fast as concerned with Application data, Can Be Used for 8 Bit data as well as 32 Bit data as well.

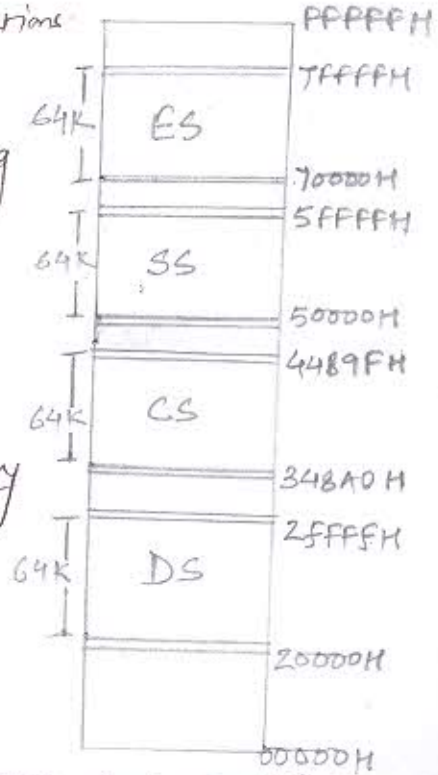
following Are main Registers Involved:

(1) General purpose Registers:

1-	AX(16) — Accumulator	⟷	AH (8) & AL (8)
2-	BX(16) — Base Registes.	⟷	BH(8) & BL(8)
3-	CX(16) — Count Registes.	⟷	CH (8) & CL(8)
4 —	DX(16) — Data Registes	⟶	DH(8) & DL(8)

(2) Segment Registers:

- At Any Instant 8086 works with only 65536 Locations ie 64K segment in total 1MB locations.
- CS is used to Hold upper 16 Bits of the staring Address of segment from which the BIU is currently fetching the Code Byte.
- The Stack segment to Hold upper 16 Bits of starting address of Program Stack
- The DS Registes points the Data Segment Memory where the data resides.
- the ES also refers to the segment, which is Essentialy another data segment of Memory



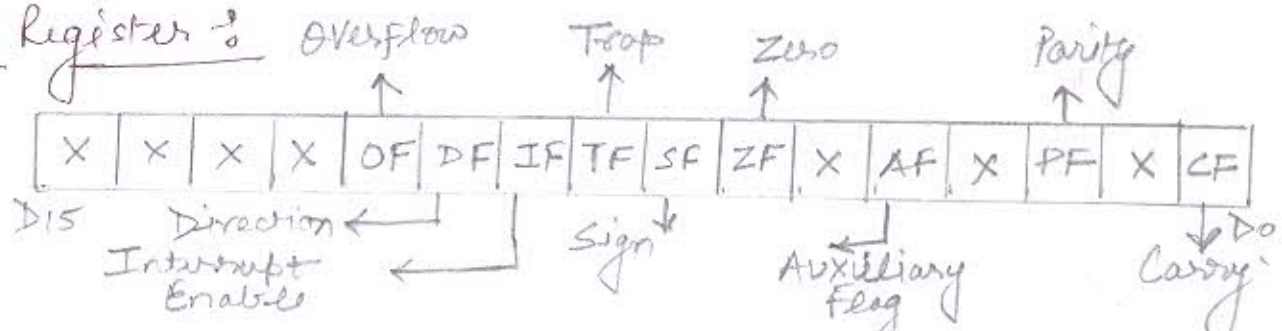
(3) Instruction Pointer: where segments Registers Are used for storing the upper 16 Bits of starting address of the segment from which the BIU is fetching the Currunt Code.

- Here the IP register Holds the 16 Bit address or offset of the next Code Byte within the Code Segment.
- CS Register points to Base or start address of Current Code & IP refers to the distance from the Base Address to the Instruction Byte to be fetched.

Stack Pointer :- Stack is a Section of memory set Aside to store Address & data while a subprogram is executing

- Stack Segment register is used to store the upper 16 Bits of Starting Address of Program stack.
- The SP Register in Execution Unit Holds the 16 Bit offset from start of the segment to the memory Location where recent Entry has Been done in stack Space.
- the memory Location where a word was recently stored is called Top of stack.

flag Register :-



Out of these 9 Bits 6 Are status Bits/flags. while 3 Are Control flag ie Direction flag, Trap flag, Interrupt enable.

⇒ Remaining three Are Pointer & Index Registers in Execution unit

→ 16 Bit source Index Registers
→ 16 Bit Destination Index Registers
→ Base Pointer Registers (16 Bit).

- these Registers Are used for temporary storage of data Just as general purpose Registers.
- their main Use is to store the 16 Bit offset
- SI is Used for storing offset of Data segment (DS)
- DI is used for storing offset of Extra segment (ES)
- The Index registerss Are Specifically used for string manipulations.

Block Diagram for 8086