MICROPROCESSOR & INTERFACING TECHNIQUES

NOTES

STUDENT ADVISORY

Dear Students,

Please be informed that the notes provided by the institute offer a concise presentation of the syllabus. While these notes are helpful for an overview and quick revision, We would strongly suggest that you refer to the prescribed textbooks / Reference book for a comprehensive understanding and thorough preparation of all exams and writing in the examination.

Best regards,

LJ Polytechnic.

પ્રિય વિદ્યાર્થીઓ,

તમને જાણ કરવામા આવે છે કે સંસ્થા દ્વારા પ્રદાન કરવામાં આવેલી નોંધો અભ્યાસક્રમની સંક્ષિપ્ત પ્રસ્તુતિ આપે છે. આ નોંધો વિહંગાવલોકન અને ઝડપી પુનરાવર્તન માટે મદદરૂપ હોઈ શકે છે તેમ છતા, અમે ભારપૂર્વક સૂચન કરીએ છીએ કે વિદ્યાર્થી તમામ પરીક્ષાઓ અને પરીક્ષામાં લેખનની વ્યાપક સમજણ અને સંપૂર્ણ તૈયારી માટે માત્ર સૂચવેલા પાઠ્યપુસ્તકો/સંદર્ભ પુસ્તકનો સંદર્ભ લો.

એલજે પોલિટેકનિક.

UNIT- 1: ARCHITECHTURE OF 8085

❖ INTRODUCTION TO MICROPROCESSOR AND MICROCOMPUTER-

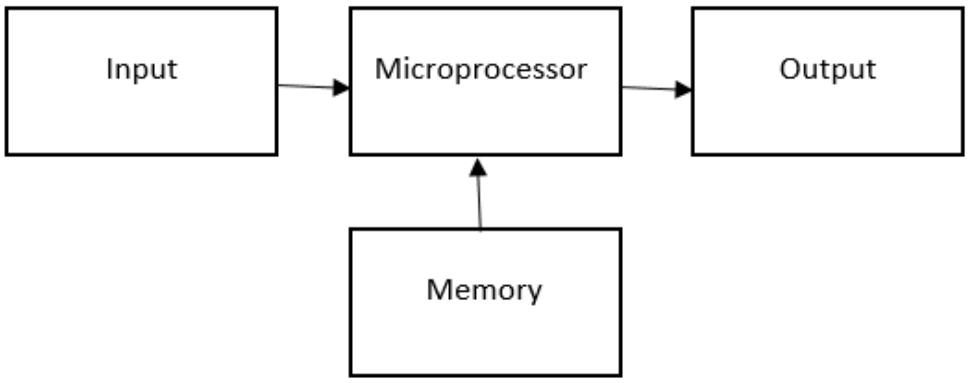
MICROPROCESSOR:

- A Microprocessor is a multipurpose, Programmable clock driven, register based electronic device, that read binary instruction from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as outputs.
- Microprocessor is clock driven semiconductor device which for is manufactured by using LSI and VLSI technique.

MICROCOMPUTER:

- A **microcomputer** is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU). It includes a microprocessor, memory, and input/output (I/O) facilities.
- Microcomputers became popular in the 1970s and 80s with the advent of increasingly powerful microprocessors.
- Examples of Microcomputers are Intel 8051 controller-a single board computer, IBM PC and Apple Macintosh computer.

❖ GENERAL ARCHITECTURE OF MICROCOMPUTER SYSTEM:



The major parts are CPU, Memory and I/O

There are three buses, address bus, data bus and control bus;

MEMORY:

- Memory consist of RAM and ROM, the purpose of memory is to store binary codes for the sequences of instructions you want the computer to carry out.
- The second purpose of the memory is to store the binary-coded data with which the computer is going to be working.

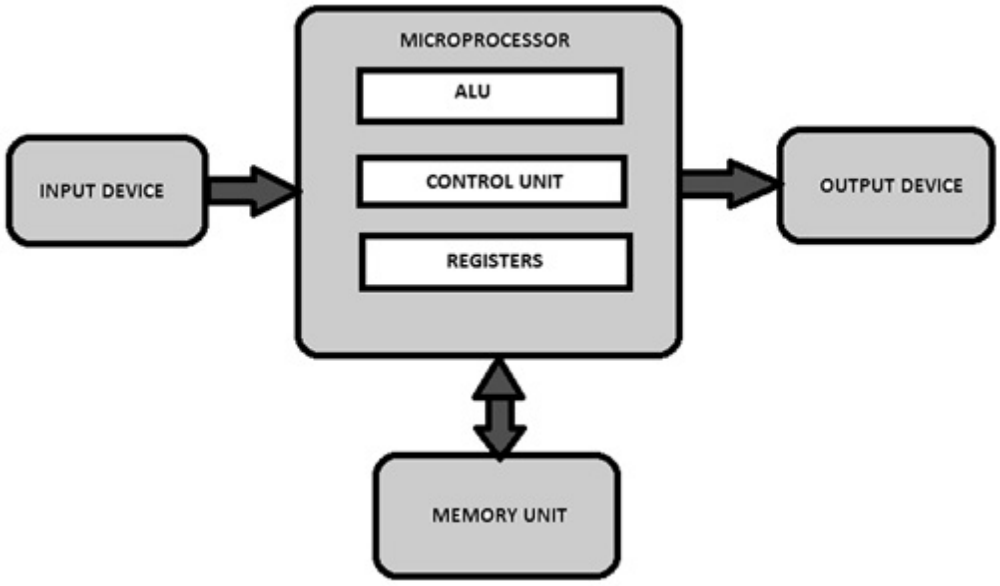
INPUT / OUTPUT:

- The input/output or I/O Section allows the computer to take in data from the outside world or send data to the outside world.
- Peripherals such as keyboards, video display terminals, printers are connected to I/O Port.

CPU (CENTRAL PROCESSING UNIT):

- In a microcomputer CPU is a microprocessor.
- The fetches binary coded instructions from memory, decodes the instructions into a series of simple actions and carries out these actions in a sequence of steps.
- The CPU also contains an address counter or instruction pointer register, which holds the address of the next instruction or data item to be fetched from memory.

❖ ARCHITECTURE OF MICROPROCESSOR-



Microprocessor is divided into three segments-

1. ALU
2. Register
3. Control Unit

Arithmetic Logic Unit:

- This is the area of Microprocessor where various computing functions are performed on data.
- The ALU performs operations such as addition, subtraction and logic operations such as AND, OR and exclusive OR.

Control Unit:

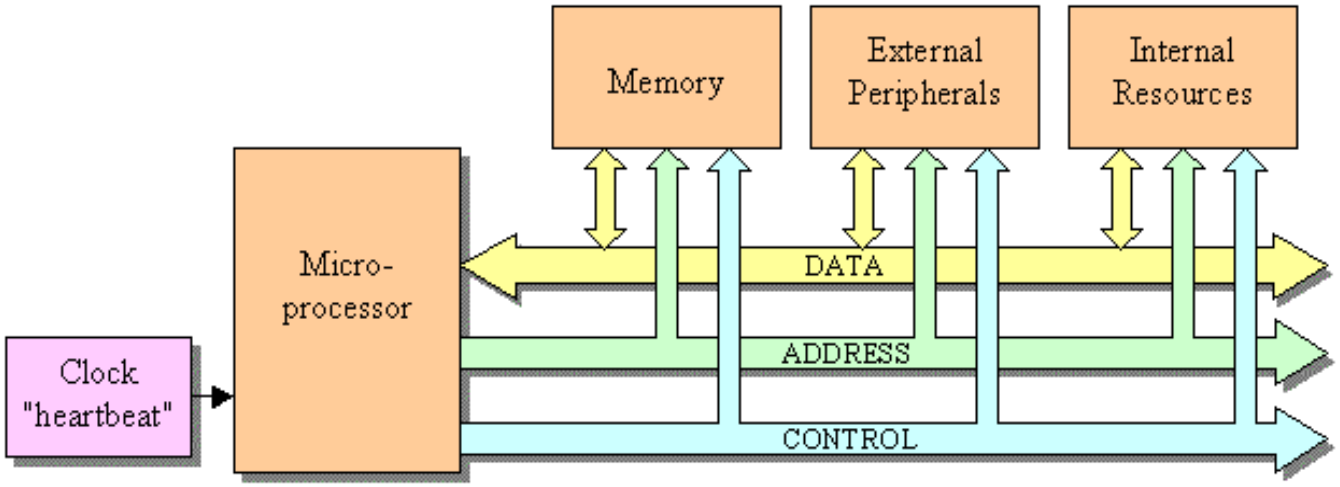
- The Control Unit Provides the necessary timing and control signals to all the operations in the Microcomputer

- It controls the flow of data between the Microprocessor and Memory and Peripherals.
- The Control unit performs 2 basic tasks
  - Sequencing
  - Execution

Register:

- These are storage devices to store data temporarily.
- There are different types of registers depending upon the microprocessor.
- These registers are primarily used to store data temporarily during the execution of a program and are accessible to the user through the instructions.

❖ CONCEPT OF ADDRESS, DATA & CONTROL BUS:



ADDRESS BUS:

- It is a group of conducting wires which carries address only.
- The address bus consists of 16, 20, 24 or 32 parallel signal lines.
- Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices.
- On these lines the CPU sends out the address of the memory location that is to be written to or read from. The no of memory location that the CPU can address is determined by the number of address lines.
- Length of Address Bus of 8085 microprocessor is 16 Bit (i.e. Four Hexadecimal Digits), ranging from 0000 H to FFFF H, (H denotes Hexadecimal).
- The microprocessor 8085 can transfer maximum 16 bit address which means it can address 65,536 different memory location.(MPU carries 16-bit address i.e. 216 = 65,536 or 64KB memory locations.)

- The Length of the address bus determines the amount of memory a system can address. Such as a system with a 32-bit address bus can address 2^32 memory locations.
- If each memory location holds one byte, the addressable memory space is 4 GB.
- However, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

DATA BUS:

- It is a group of conducting wires which carries Data only.
- The data bus consists of 8, 16 or 32 parallel signal lines.
- Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/output devices and from memory or Input/output devices to microprocessor.
- Length of Data Bus of 8085 microprocessor is 8 Bit (That is, two Hexadecimal Digits), ranging from 00 H to FF H. (H denotes Hexadecimal).
- When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.
- The width of the data bus is directly related to the largest number that the bus can carry, such as an 8 bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255.A 16 bit bus can carry 0 to 65535.

CONTROL BUS:

- The control bus consists of 4 to 10 parallel signal lines.
- The CPU sends out signals on the control bus to enable the output of addressed memory devices or port devices.
- It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data i.e. what to do with selected memory location. Some control signals are:
    - Memory read
    - Memory write
    - I/O read
    - I/O Write
    - Opcode fetch

❖ HOW DOES A MICROPROCESSOR WORK?

The microprocessor follows a sequence: Fetch, Decode, and then Execute. Initially, the instructions are stored in the memory in a sequential order. The microprocessor fetches those instructions from the

memory, then decodes it and executes those instructions till STOP instruction is reached. Later, it sends the result in binary to the output port. Between these processes, the register stores the temporarily data and ALU performs the computing functions.

❖ FEATURES OF A MICROPROCESSOR

Cost-effective − The microprocessor chips are available at low prices and results its low cost.

Size − The microprocessor is of small size chip, hence is portable.

Low Power Consumption − Microprocessors are manufactured by using metaloxide semiconductor technology, which has low power consumption.

Versatility − The microprocessors are versatile as we can use the same chip in a number of applications by configuring the software program.

Reliability − The failure rate of an IC in microprocessors is very low, hence it is reliable.

❖ LIST OF TERMS USED IN A MICROPROCESSOR

Instruction Set − It is the set of instructions that the microprocessor can understand.

Bandwidth − It is the number of bits processed in a single instruction.

Clock Speed − It determines the number of operations per second the processor can perform. It is expressed in megahertz (MHz) or gigahertz (GHz).It is also known as Clock Rate.

Word Length − It depends upon the width of internal data bus, registers, ALU, etc. An 8-bit microprocessor can process 8-bit data at a time. The word length ranges from 4 bits to 64 bits depending upon the type of the microcomputer.

Data Types − The microprocessor has multiple data type formats like binary, BCD, ASCII, signed and unsigned numbers

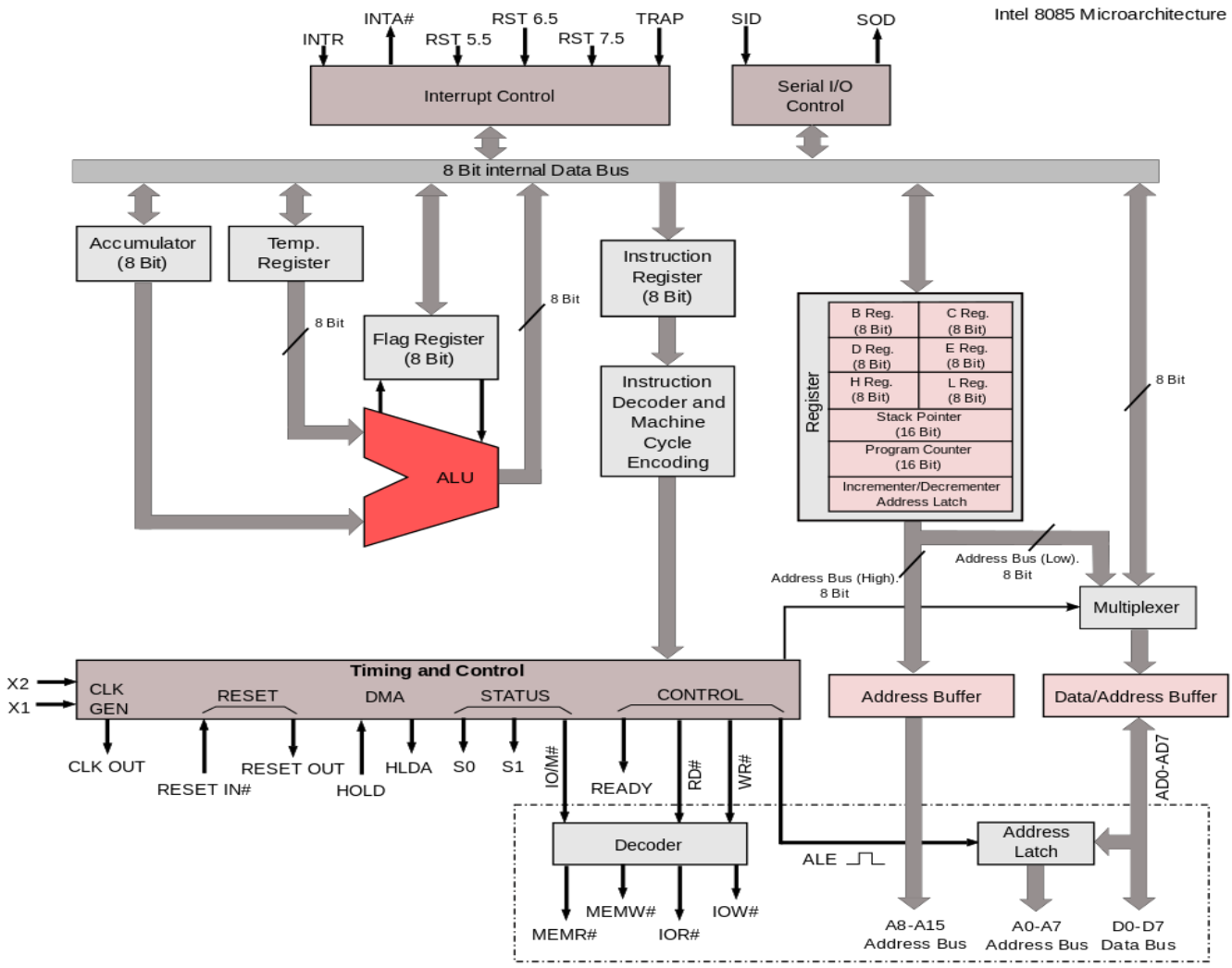
❖ 8085 MICROPROCESSORS:

FEATURES:

1. It is an 8-bit microprocessor i.e. it can accept, process, or provide 8-bit data simultaneously.
2. It operates on a single +5V: power supply connected at V_{cc}; power supply ground is connected to V_{ss}.
3. It operates on clock cycle with 50% duty cycle.
4. It has on chip clock generator. This internal clock generator requires tuned circuit like LC, RC or crystal. The internal clock generator divides oscillator frequency by 2 and generates clock signal, which can be used for synchronizing external devices.
5. It can operate with a 3 MHz clock frequency. The 8085A-2 version can operate at the maximum frequency of 5 MHz.
6. It has 8 bits &16 bits address lines; hence it can access (2^{16}) 64 Kbytes of memory.

7. It provides 8 bit I/O addresses to access (2^8) 256 I/O ports.
8. In 8085, the lower 8-bit address bus ($A_0 – A_7$) and data bus ($D_0 – D_7$) are Multiplexed to reduce number of external pins. But due to this, external hardware (latch) is required to separate address lines and data lines.
9. It supports 74 instructions with the following addressing modes:
   Immediate
   Register
   Direct
   Indirect
   Implied
10. 10.The Arithmetic Logic Unit (ALU) of 8085 performs all arithmetic & logical operation on 8 bits data.
11. It has 8-bit accumulator, flag register, instruction register, six 8-bit general purpose registers (B, C, D, E, H and L) and two 16-bit registers. (SP and PC). Getting the operand from the general-purpose registers is faster than from memory. Hence skilled programmers always prefer general purpose registers to store program variables than memory.
12. It provides five hardware interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.
13. It has serial I/O control which allows serial communication.
14. It provides control signals (IO/M, RD, WR) to control the bus cycles, and hence external bus controller is not required.
15. The external hardware (another microprocessor or equivalent master) can detect which machine cycle microprocessor is executing using status signals (IO/M, S_0, S_1).

❖ 8085 ARCHITECTURE:



- **Accumulator:**

  It is an 8-bit register used to perform arithmetic, logical, I/O & load/store operations. It is connected to internal data bus & ALU.

- **Arithmetic and logic unit:**

  As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

- **General purpose register:**

  There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H& L. Each register can hold 8-bit data.These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

- **Program counter:**

  It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

- **Stack pointer:**

  It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

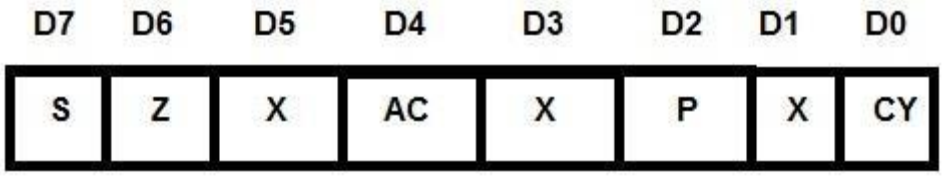
- **Temporary register:**

  It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

- **Flag register:**

  It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

- **These are the set of 5 flip-flops:**
  - Sign (S)
  - Zero (Z)
  - Auxiliary Carry (AC)
  - Parity (P)
  - Carry (C)



Flag Register of 8085

- **Instruction register and decoder:**

  It is an 8-bit register.

  When an instruction is fetched from memory then it is stored in the Instruction register.

  Instruction decoder decodes the information present in the Instruction register.

- **Timing and control unit:**

  It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits:-

  Control Signals: READY, RD', WR', ALE

  Status Signals: S0, S1, IO/M'

  DMA Signals: HOLD, HLDA

  RESET Signals: RESET IN, RESET OUT

- **Interrupt control:**

  As the name suggests it controls the interrupts during a process.

  When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request.

After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, and TRAP.

- **Serial Input/output control:**

  It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

- **Address buffer and address-data buffer:**

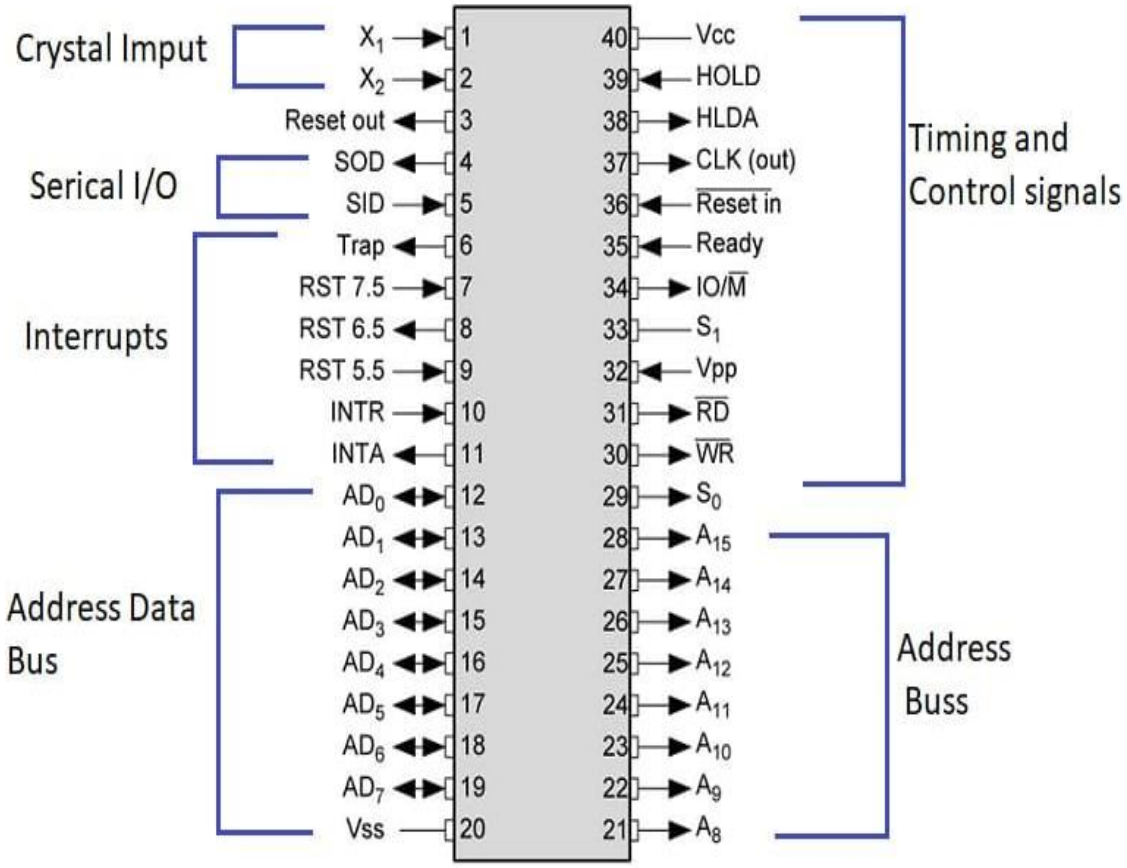
  The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU.

  The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

- **Address bus and data bus:**

  Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

❖ PIN DESCRIPTION OF 8085:



- **Address bus:**

  A15-A8, it carries the most significant 8-bits of memory/IO address.

- **Data bus:**

  AD7-AD0, it carries the least significant 8-bit address and data bus.

- **Control and status signals:**

  These signals are used to identify the nature of operation. There are 3 control signal and 3 status signals.

  Three control signals are RD', WR' & IO/M'.

- **RD'**:

  This signal indicates that the selected IO or memory device is to be read and is ready for accepting data available on the data bus.

- **WR':**

  This signal indicates that the data on the data bus is to be written into a selected memory or IO location.

- **IO/M':**

  This signal is used to differentiate between IO and Memory operations, i.e. when it is high indicates IO operation and when it is low then it indicates memory operation.

- **ALE**:

  It is a positive going pulse generated when a new operation is started by the microprocessor. When the pulse goes high, it indicates address. When the pulse goes down it indicates data.

- **S1 & S0:**

  These signals are used to identify the type of current operation.

Machine cycle	IO/M'	Status		Control Signals
		S1	So	
Opcode Fetch	0	1	1	RD=0
Memory Read	0	1	0	RD=0
Memory Write	0	0	1	WR=0
I/O read	1	1	0	RD=0
I/O write	1	0	1	WR=0
Interrput Ack.	1	1	1	INTA=0

- **Power supply:**

  There are 2 power supply signals V_{CC} & V_{SS}. VCC indicates +5v power supply and VSS indicates ground signal.

- **Clock signals:**

  There are 3 clock signals, i.e. X1, X2, CLK OUT.

- **X1 & X2**:

  A crystal (RC, LC N/W) is connected at these two pins and is used to set frequency of the internal clock generator. This frequency is internally divided by 2.

- **CLK OUT**:

  This signal is used as the system clock for devices connected with the microprocessor.

- **Interrupts & externally initiated signals:**

  - Interrupts are the signals generated by external devices to request the microprocessor to perform a task.There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. We will discuss interrupts in detail in interrupts section.

  TRAP:

It is a non-maskable interrupt, having the highest priority among all interrupts. By default, it is enabled until it gets acknowledged. In case of failure, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H.

RST 7.5:

It is a maskable interrupt, having the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

RST 6.5:

It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address.

RST 5.5:

It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.

INTR:

It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor. When **INTR signal goes high**, the following events can occur: The microprocessor checks the status of INTR signal during the execution of each instruction.

When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.

When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

INTA':

It is an interrupt acknowledgment sent by the microprocessor after INTR is received.

- **RESET IN**:

This signal is used to reset the microprocessor by setting the program counter to zero.

- **RESET OUT:**

This signal is used to reset all the connected devices when the microprocessor is reset.

- **READY:**

This signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.

- **HOLD:**

  This signal indicates that another master is requesting the use of the address and data buses.

- **HLDA (HOLD Acknowledge):**

  It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed.

- **Serial I/O signals:**

  There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.

- **SOD (Serial output data line):**

  The output SOD is set/reset as specified by the SIM instruction.

- **SID (Serial input data line)**:

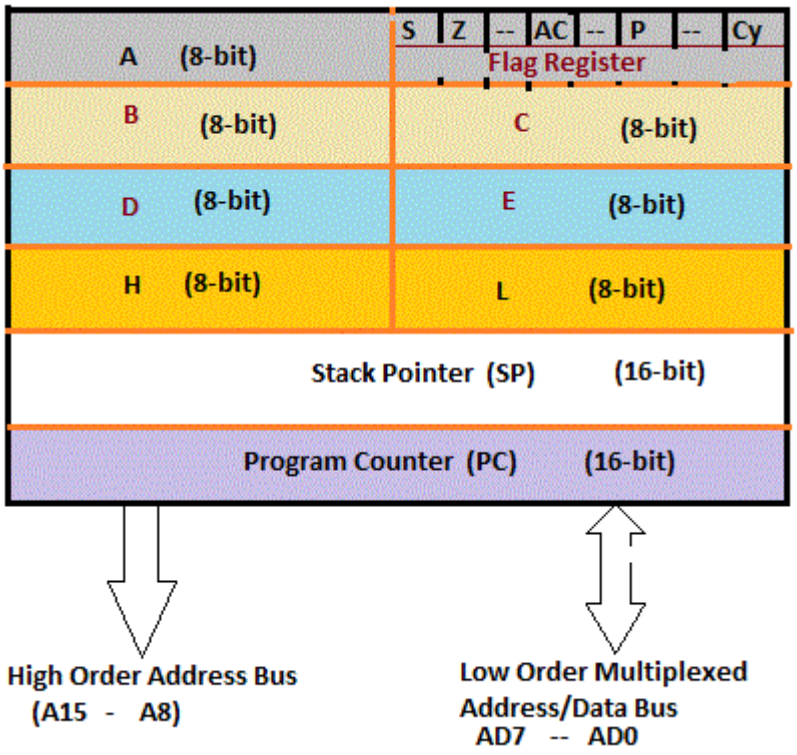
  The data on this line is loaded into accumulator whenever a RIM instruction is executed.

  - When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
  - When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

❖ REGISTER ORGANIZATION:

It has six addressable 8-bit registers: A, B, C, D, E, H, L and two 16-bit registers PC and SP. These registers can be classified as:

- **General Purpose Registers- B, C, D, E, H, L**
- **Temporary Registers:** Temporary data register, W and Z registers
- **Special Purpose Registers:** Accumulator, Flag registers, Instruction register
- **Sixteen-bit Registers:** Program Counter (PC), Stack Pointer (SP)



- **General Purpose Registers:**
  - Registers B, C, D, E, H, and L are general purpose registers in 8085 Microprocessor. All these GPRS are 8-bits wide. They are less important than the accumulator.
  - They are used to store data temporarily during the execution of the program.
  - It is possible to use these registers as pairs to store 16-bit information. Only B-C, D-E, and H-L can form register pairs.
  - When they are used as register pairs in an instruction, the left register is understood to have the MSB byte and the right registers the LSB byte.
  - For example, in D-E register pair, the content of the D register is treated as the MSB byte, and the content of E register is treated as the LSB byte.
- **Temporary Registers:**
  - **Temporary Data Register: -**
  - The ALU has two inputs. One input is supplied by the accumulator and other from the temporary data register.
  - The programmer cannot access this temporary data register. However, it is internally used for execution of most of the arithmetic and logical instructions.

- **W and Z register:-** Wand Z registers are temporary registers. These registers are used to hold 8-bit data during the execution of some instructions. These registers are not available for the programmer since 8085Microprocessor Architecture uses them internally.

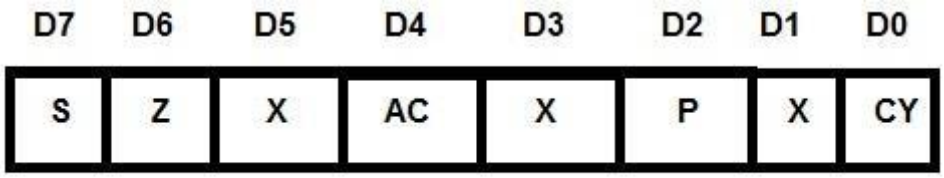
➢ **Special Purpose Registers:**

Accumulator (A):

- Register A is an 8-bit register used in 8085 to perform arithmetic, logical, I/O & load/store operations.
- Register A is quite often called as an Accumulator. An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (Central Processing Unit).
- In an arithmetic operation involving two operands, one operand has to be in this register. And the result of the arithmetic operation will be stored or accumulated in this register.
- Similarly, in a logical operation involving two operands, one operand has to be in the accumulator. Also, some other operations, like complementing and decimal adjustment, can be performed only on the accumulator.

➢ **Flag Register:**

It is a 3-bit register, in which five of the bits carry significant information in the form of flags: S (Sign flag), Z (Zero flag), AC (Auxiliary carry flag), P (Parity flag), and CY (carry flag).



Flag Register of 8085

- **S-Sign flag: -** After the execution of arithmetic or logical operations, if bit D7 of the result is 1, the sign flag is set. In a given byte if D7 is1, the number will be viewed as a negative number. If D7 is U, the number will be considered as a positive number.
- **Z-Zero flag**:-The zero flag sets if the result of the operation in ALU is zero and flag resets if the result is non-zero. The zero flags are also set if a certain register content becomes zero following an increment or decrement operation of that register.
- **AC-auxiliary Carry flag: -** This flag is set if there is an overflow out of bit 3 i.e. carry

from lower nibble to higher nibble (D3 bit to D4 bit). This flag is used for BCD operations and it is not available for the programmer.

- **P-Parity flag: -** Parity is defined by the number of one's present in the accumulator.

After arithmetic or logical operation, if the result has an even number of ones, i.e. even parity, the flag is set. If the parity is odd, the flag is reset.

- **CY-Carry flag:** - This flag is set if there is an overflow out of bit 7. The carry flag also serves as a borrow flag for subtraction. In both the examples shown below, the carry flag is set.

➢ **Instruction Register:-**

- In a typical processor operation, the processor first fetches the opcode of instruction from memory (i.e. it places an address on the address bus and memory responds by placing the data stored at the specified address on the data bus).
- The CPU stores this opcode in a register called the instruction register. This opcode is further sent to the instruction decoder to select one of the 256 alternatives.

➢ **SIXTEEN BIT REGISTERS:**

Program counter (PC):-

- Program is a sequence of instructions. Microprocessor fetches these instructions from the memory and executes them.
- The program counter is a special purpose register which, at a given time, stores the address of the next instruction to be fetched.
- Program Counter acts as a pointer to the next instruction.
- How processor increments program counter depends on the nature of the instruction; for one-byte

  instruction it increments program counter by one, for two-byte instruction it increments program counter by two and for three-byte instruction it increments program counter by three such that program counter always points to the address of the next instruction.

Stack Pointer (SP):-

- The stack is a reserved area of the memory in the RAM where temporary information may be stored. A 16-bit stack pointer is used to hold the address of the most recent stack entry.

❖ DISTINGUISH BETWEEN GPR AND SPR:

GPR-

- It stands for General purpose registers.
- In these registers data can be accessed directly without requiring any intermediate.
- Examples of GPR are B, C, D, E, H, and L.
- These registers are of 8-bit.
- In order to hold 16 bit data, two 8 bit register can be combined or they can work in pairs such as B-C, D-E and H-L. These pairs are known as register pairs.
- The H-L pair works as a memory pointer.
- A memory pointer holds the address of a particular memory location.

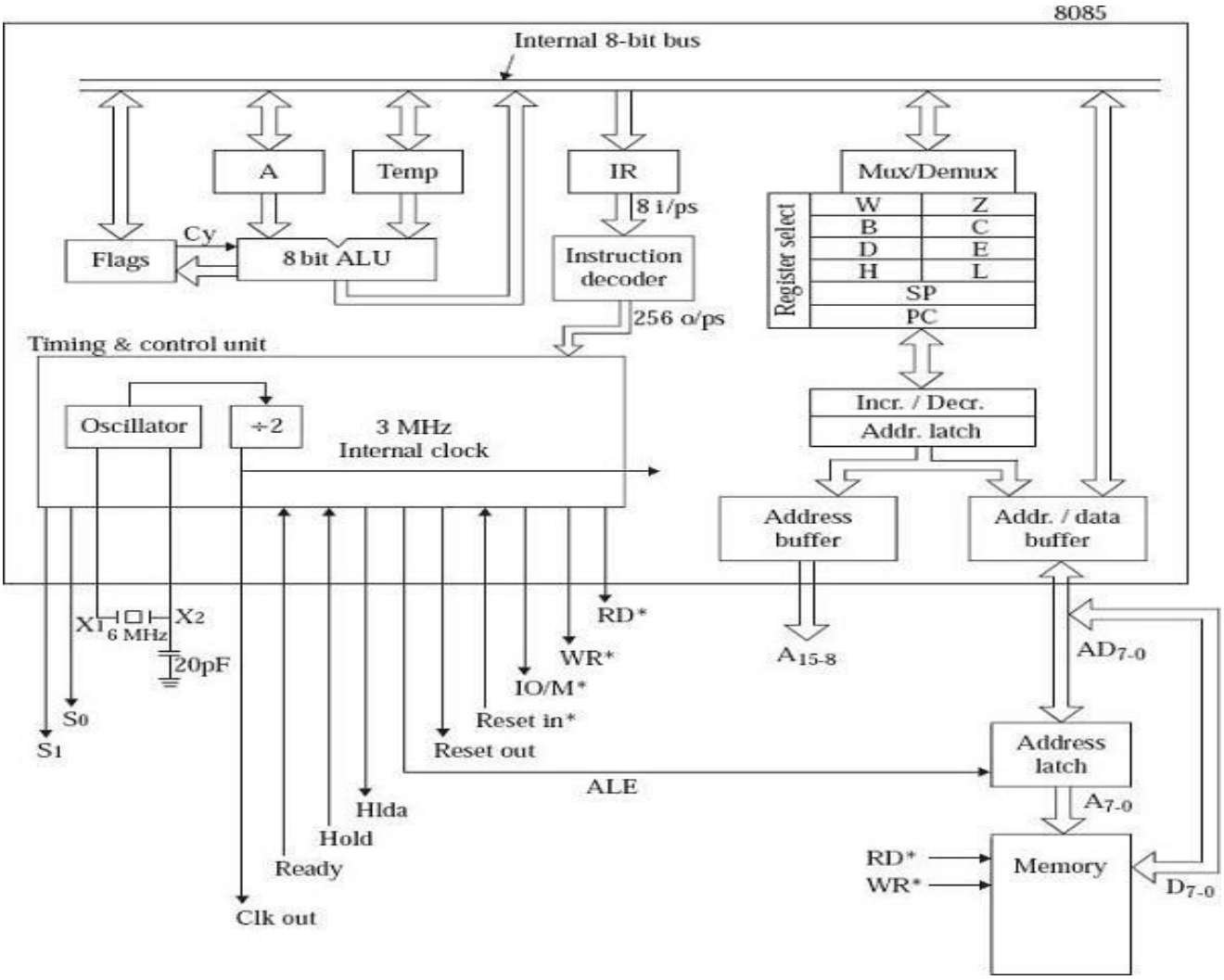
SPR-

- SPR stands for special purpose register.
- In special purpose register data cannot accessed directly and requires an intermediate.
- Examples of SPR are Accumulator, program counter, stack pointer.
- These registers are used only by microprocessor not by users.

❖ TIMING AND CONTROL UNIT:

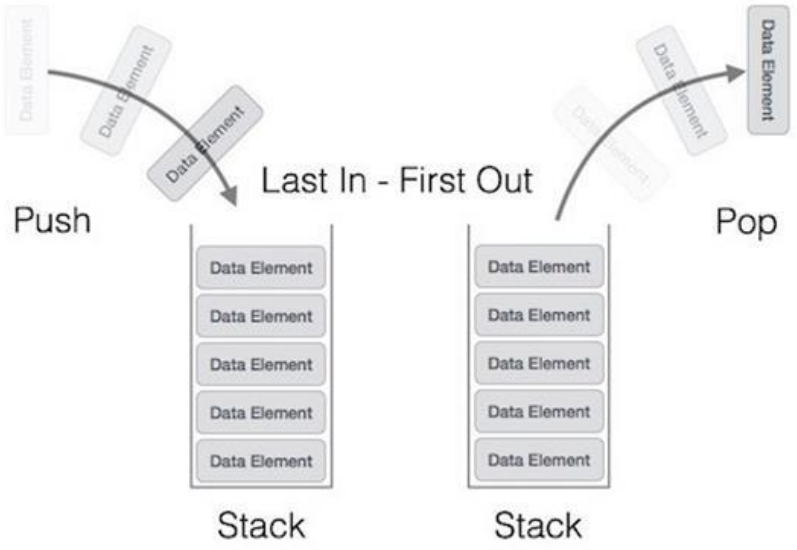
- We use Timing and controlling unit in 8085 for the generation of timing signals and the signals to control. All the operations and functions both interior and exterior of a microprocessor are controlled by this unit.
- X2 and CLK output pins: To do or rather perform the operations of timing in the microcomputer system, we have a generator called clock generator in the CU of 8085.
- Other than the quartz crystal the complete circuit of the oscillator is within the chip.
- The two pins namely X1 and X2 are taken out from the chip to give the connection to the crystal externally.
- We connect a capacitor of 20pF between the terminal X2 and ground just to analyze if the crystal is getting started.
- The frequency of the crystal is divided by 2 which divide the counter of the unit of control by 2.
- Internally 8085A works with a frequency of 3 MHz internally with clock frequency.
- Hence a crystal of frequency of 6-MHz crystal gets connected between X1 and X2.

- Every operation in the entire 8085 system occurs with the given synchronization process with the clock. There are Peripheral chips like 8251 USART, which does not operate until a small clock signal is in need.



❖ STACK, STACK POINTER AND STACK TOP: STACK:

- The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data when the microprocessor branches to a subroutine.
- Then the return address used to get pushed on this stack. Also to swap values of two registers and register pairs we use the stack as well.



STACK POINTER:

- It is a special purpose 16-bit register that stores the address of the "**top of stack**"**.**
- "8085" provides the **"stack pointer"** which gives the address of the "top of stack". So, whenever you want to store an item it stacks, you just store it at the address provided by the stack pointer.

STACK operation in 8085 microprocessor.

- The stack is a reserved area of the memory in RAM where temporary information may be stored. An 8-bit stack pointer is used to hold the address of the most recent stack entry. This location which has the most recent entry is called as the top of the stack.
- When the information is written on the stack, the operation is called PUSH. When the information is read from the stack, the operation is called POP. The stack works on the principle of Last in First Out.

❖ TIMING DIAGRAM AND MACHINE CYCLES OF 8085 MICROPROCESSOR:

- **Timing Diagram:**
- Timing Diagram is a graphical representation. Timing diagram is the display of initiation of read/write and transfer of data operations under the control of 3-status signals IO/M', S1 and S0.
- Each machine cycle is composed of many clock cycles. Since, the data and instructions, both are stored in the memory, the µP performs fetch operation to read the instruction or data and then execute the instruction.
- The 3-status signals: IO / M', S1 and S0 are generated at the beginning of each machine cycle. The unique combination of these 3-status signals identifies read or write operation and remain valid for the duration of the cycle.
- Thus, time taken by any µP to execute one instruction is calculated in terms of the **clock period**.
- The execution of instruction always requires read and writes operations to transfer data to or from the µP and memory or I/O devices.
- Each read/ write operation constitutes one **machine cycle**.
- Each machine cycle consists of many clock periods/ cycles, called **T-states**.

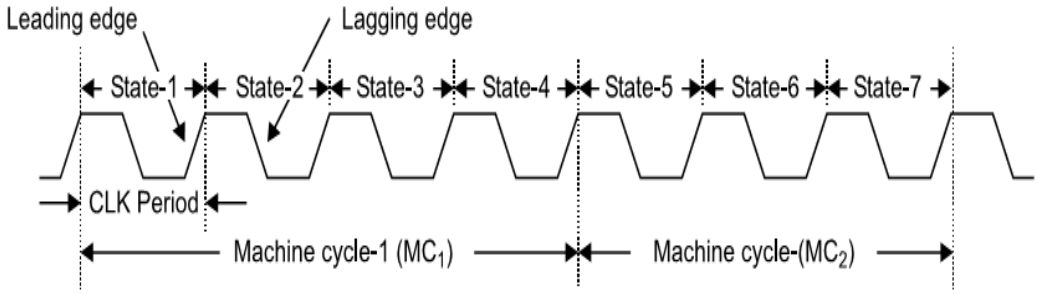


Fig. Machine cycle showing clock periods

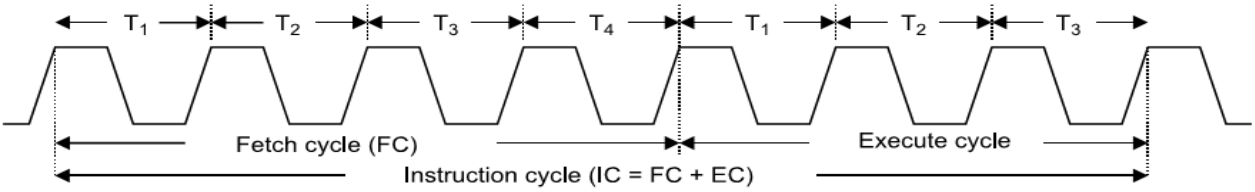
- Each and every operation inside the microprocessor is under the control of the clock cycle. The clock signal determines the time taken by the microprocessor to execute any instruction.
- State is defined as the time interval between 2-trailing or leading edges of the clock.
- **Machine cycle** is the time required to transfer data to or from memory or I/O devices.

The 8085 microprocessor has 5 basic machine cycles. They are

- Opcode fetch cycle (4T)
- Memory read cycle (3 T)
- Memory write cycle (3 T)
- I/O read cycle (3 T)
- I/O write cycle (3 T)

➢ PROCESSOR CYCLE

- The function of the microprocessor is divided into fetch and execute cycle of any instruction of a program.
- In the normal process of operation, the microprocessor fetches (receives or reads) and executes one instruction at a time in the sequence until it executes the halt (HLT) instruction.
- Thus, an instruction cycle is defined as the time required to fetch and execute an instruction.
- For executing any program, basically 2-steps are followed sequentially with the help of clocks
    - Fetch, and
    - Execute.
- The time taken by the µP in performing the fetch and execute operations are called fetch and execute cycle.
- Thus, sum of the fetch and execute cycle is called the instruction cycle as indicated in Fig.
- Instruction Cycle (IC) = Fetch cycle (FC) + Execute Cycle (EC)



➢ The 1st machine cycle of any instruction is always an Opcode fetch cycle in which the processor decides the nature of instruction. It is of at least 4-states. It may go up

to 6-states.

- In the opcode fetch cycle, the processor comes to know the nature of the instruction to be executed.
- The processor during (M1 cycle) puts the program counter contents on the address bus and reads the opcode of the instruction through read process.
- The T1, T2, and T3 clock cycles are used for the basic memory read operation and the T4 clock and beyond are used for its interpretation of the opcode.
- Based on these interpretations, the μP comes to know the type of additional information/data needed for the execution of the instruction and accordingly proceeds further for 1 or 2-machine cycle of memory read and writes.
- **Instruction Fetch (FC)**⇒An instruction of 1 or 2 or 3-bytes is extracted from the memory locations during the fetch and stored in the μP's instruction register.
- **Instruction Execute (EC)**⇒The instruction is decoded and translated into specific activities during the execution phase.

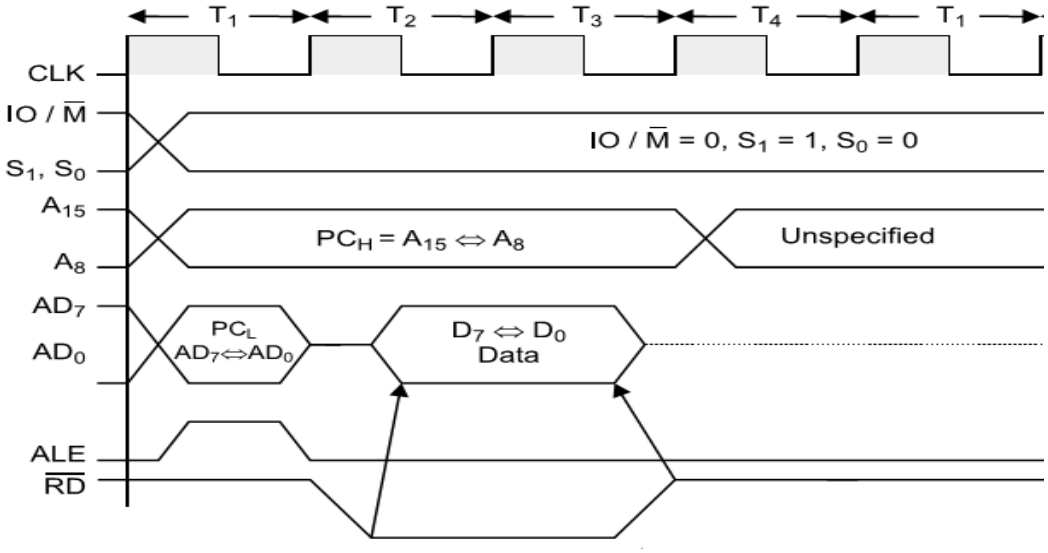
- **OPCODE FETCH**

The 1st step in communicating between the microprocessor and memory is reading from the memory. This reading process is called opcode fetch.

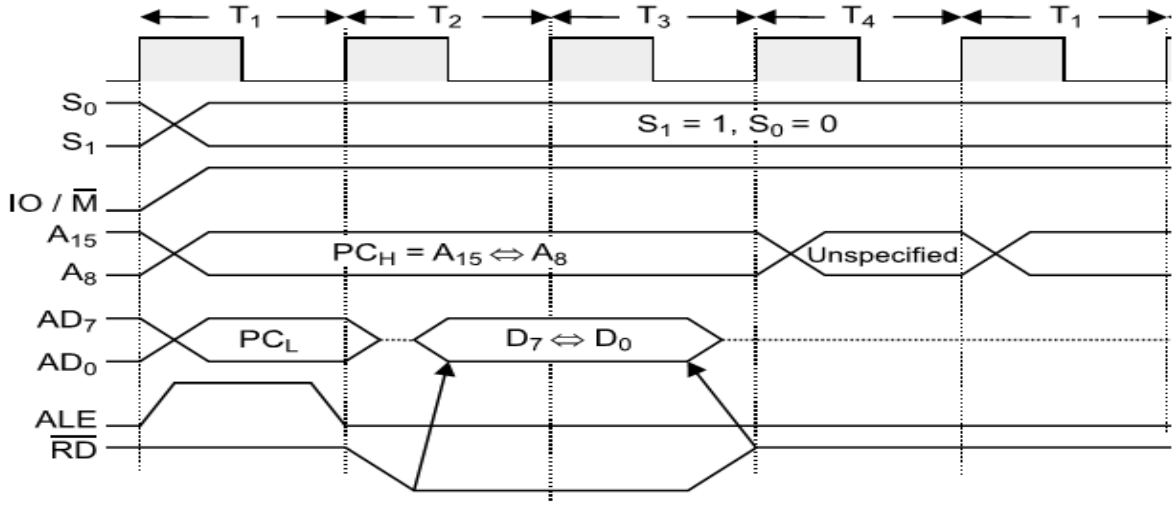
- The process of opcode fetch operation requires minimum 4-clock cycles T1, T2, T3, and T4and is the 1st machine cycle (M1) of every instruction.
- In order to differentiate between the data byte pertaining to an opcode or an address, the machine cycle takes help of the status signal IO/ M, S1, and S0. The IO/ M= 0 indicates memory operation and S1 = S0 = 1 indicates Opcode fetch operation.
- The opcode fetch machine cycle M1 consists of 4-states (T1, T2, T3, and T4). The 1st 3-states are used for fetching (transferring) the byte from the memory and the 4th-state is used to decode it.

➢ MEMORY AND I/O READ CYCLE

- ➢ The memory read machine cycle is executed by the processor to read a data byte from memory.
- ➢ The processor takes 3T states to execute this cycle. The instructions which have more than one-byte word size will use the machine cycle after the opcode fetch machine cycle.



Memory Read Cycle

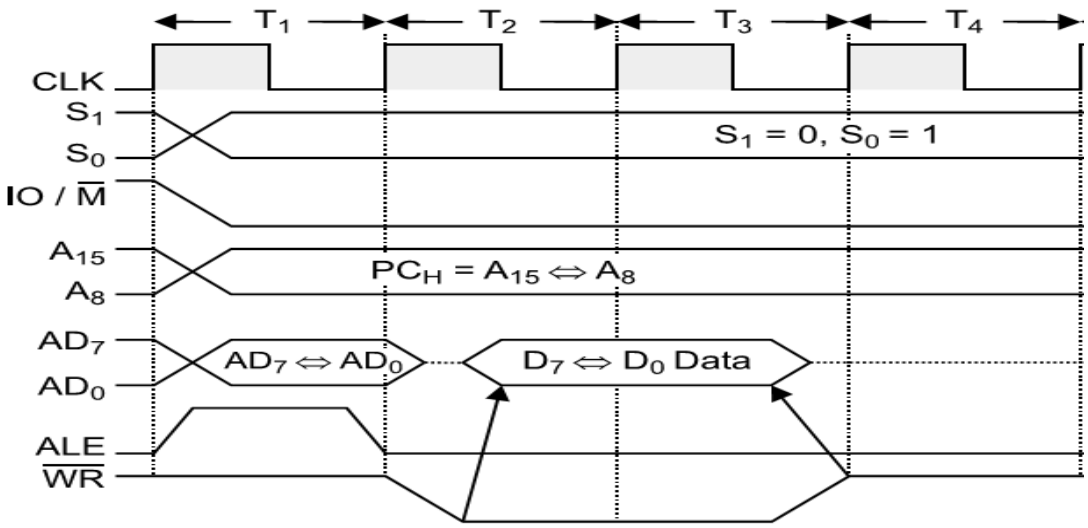


I/O Read Cycle

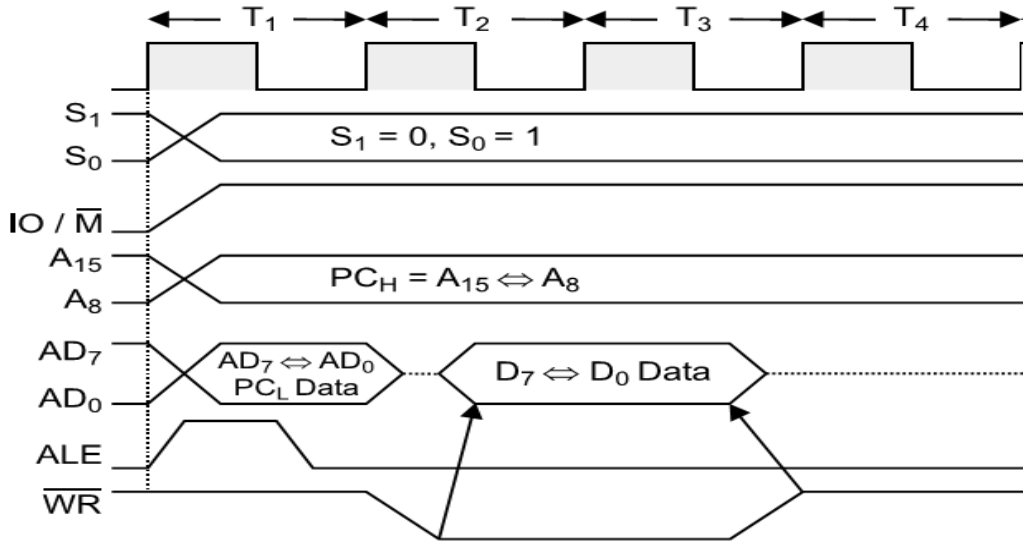
- The high order address (A15 ⇔A8) and low order address (AD7 ⇔AD0) are asserted on 1st low going transition of the clock pulse.
- The timing diagram for IO/M read are shown in Fig.
- The A15 ⇔A8 remains valid in T1, T2, and T3 i.e. duration of the bus cycle, but AD7⇔AD0 remains valid only in T1.
- Since it has to remain valid for the whole bus cycle, it must be saved for its use in the T2 and T3.
- ALE is asserted at the beginning of T1 of each bus cycle and is negated towards the end of T1. ALE is active during T1 only and is used as the clock pulse to latch the address (AD7⇔AD0) during T1.
- The RD' is asserted near the beginning of T2. It ends at the end of T3. As soon as the RD' becomes active, it forces the memory or I/O port to assert data. RD' becomes inactive towards the end of T3, causing the port or memory to terminate the data.

➢ MEMORY AND I/O WRITE CYCLE

- Immediately after the termination of the low order address, at the beginning of the T2, data is asserted on the address/data bus by the processor.
- WR' control is activated near the start of T2 and becomes inactive at the end of T3.
- The processor maintains valid data until after WR' is terminated. This ensures that the memory or port has valid data while WR' is active.
- It is clear from figures that for READ bus cycle, the data appears on the bus as a result of activating RD' and for the WR' bus cycle, the time the valid data is on the bus overlaps the time that the WR' is active.



Memory Write Cycle



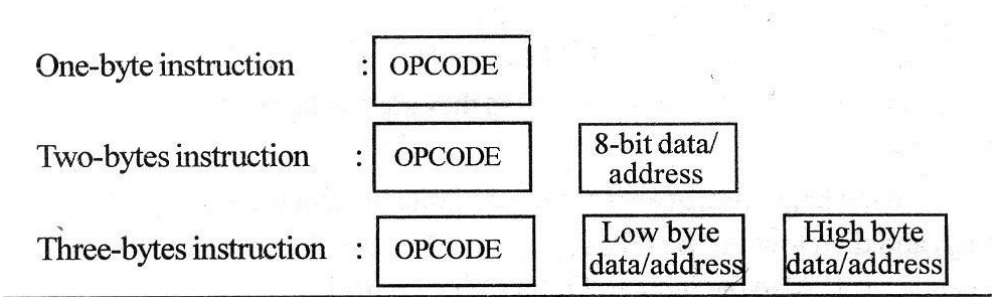
I/O Write Cycle

UNIT-2 8085 INSTRUCTION SET

❖ DEFINE OPCODE AND OPERAND:

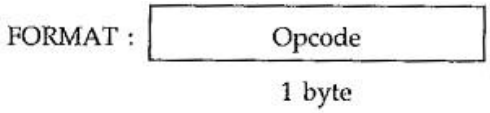
- Opcode (Operation code) is the part of an instruction / directive that identifies a specific operation.
- Operand is a part of an instruction / directive that represents a value on which the instruction acts.
- Each instruction of 8085 has 1-byte opcode. With 8-bit binary code, we can generate 256 different binary codes. In 8085,246 codes have been used for opcodes. The size of 8085 instructions can be 1 byte, 2 bytes or 3 bytes.

❖ INSTRUCTION FORMATS:



The Instruction Format of 8085 set consists of one, two- and three-byte instructions. The first byte is always the opcode; in two-byte instructions the second byte is usually data; in three-byte instructions the last two bytes present address or 16-bit data.

1. **ONE-BYTE INSTRUCTION**:



- They include opcode and operands in the same byte.
- Operands are internal registers and coded into the instruction.
- Instructions require one memory location to store the single byte in the memory.

Note:

Instructions having the only register or register pair as the operand is 1 – Byte Instructions. Instructions in the absence of operand are also 1 – Byte Instructions.

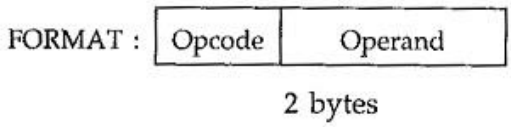
Examples: MOV B, C

LDAX B

NOP

HLT

2. TWO-BYTE INSTRUCTION:



- 1st byte specifies opcode and 2nd byte specifies operand.
- Instructions require two memory locations to store in the memory.

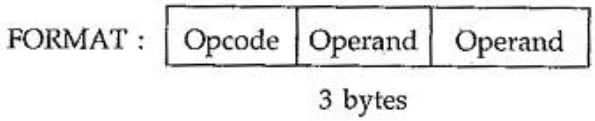
Note:

Instructions having the 8-bit number either as an address or data as the operand is 2 – Byte Instructions.

Examples: MVI B, 26 H

IN 56 H

3. THREE-BYTE INSTRUCTION:



In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address.

- The 2nd byte holds the low order address.
- The 3rd-byte holds the high order address.
- Instructions require three memory locations to store the single byte in the memory.

Note:

Instructions having the 16-bit number either as an address or data as the operand is 3 – Byte Instructions.

Examples: LDA 2050 H

JMP 2085 H

❖ ADDRESSING MODES:

- The various ways of specifying data (or operands) for instructions are called as **addressing modes.**
- The 8085 addressing modes are classified into following types:
  1. Immediate addressing mode
  2. Direct addressing mode
  3. Register addressing mode
  4. Register indirect addressing mode
  5. Implicit addressing mode

1. Direct Addressing mode:

- In this addressing mode the address of the operand is specified in the instruction itself.

OR

- The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called Direct Addressing mode. i.e., the address of the operand is available in the instruction itself. This is a 3-byte instruction.

Example:

LDA 9525H	Load the contents of memory location into Accumulator.
STA 8000	Store the contents of the Accumulator in the location 8000H IN 01H
	Read the data from port whose address is 01H

2. Register addressing modes:

- In this addressing mode the address of the operand is one of the general purpose register.

OR

- In this mode the operands are microprocessor registers only i.e. The operation is performed within various registers of the microprocessor.

Example:

MOV A, B	Move the contents of B register to A register.
SUB D	Subtract the contents of D register from Accumulator.
ADD B, C	Add the contents of C register to the contents of B register.

3. Register indirect addressing modes:

- In this addressing mode the address of the operand is specified by a register pair.

OR

- The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. So it is called

Register indirect addressing mode.

Example:

LXIH 9570H	Load immediate the H-L pair with the address of the location 9570H
MOV A, M	Move the contents of the memory location pointed by the H-L pair to accumulator

4. Immediate Addressing mode:

- In this addressing mode the operand is specified in the instruction itself.

OR

- In this mode operand is a part of the instruction itself is known as Immediate Addressing mode. If the immediate data is 8-bit, the instruction will be of two bytes. If the immediate data is 16 bit, the instruction is of 3 bytes.

Example:

ADI DATA	Add immediate the data to the contents of the accumulator.
LXIH 8500H	Load immediate the H-L pair with the operand 8500H
MVI 08H	Move the data 08 H immediately to the accumulator
SUI 05H	Subtract immediately the data 05H from the accumulator

5. Implicit Addressing mode:

- In this addressing mode the instruction doesn't require the address of the operand.

OR

- The mode of instruction which do not specify the operand in the instruction but it is implicated, is known as implicit addressing mode. i.e., the operand is supposed to be present generally in accumulator.

Example:

CMA	Complement the contents of Accumulator
CMC	Complement carry
RLC	Rotate Accumulator left by one bit
RRC	Rotate Accumulator right by one bit
STC	Set carry.

❖ INSTRUCTION SET OF 8085:

- An instruction is a binary bit pattern which performs a specific function in a system. The entire group of instructions of a system is called the instruction set.
- Instruction set determines what functions the microprocessor can perform with a single instruction.
- The instruction set in microprocessor 8085 can be classified into five functional categories:

OR

- An instruction is a command to the microprocessor to perform a given task on a specified data.
- Each instruction has two parts: one is task to be performed, called the operation code (opcode), and the second is the data to be operated on, called the operand.
- The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

1. Data transfer (copy) operations
2. Arithmetic operations
3. Logical operations
4. Branching operations and
5. Machine-control operations.

1. DATA TRANSFER INSTRUCTION:

- These instructions move data between registers, or between memory and registers.
- This group of instructions copies data from a location called as source to another location called as destination, without modifying the contents of the source
- These instructions are not the data transfer instructions but data copy instruction because the source is not modified.

Opcode Operand Copy from source to destination	**Description**
MOV Rd, Rs	This instruction copies the contents of the source Register into the destination register; the contents of The source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. **Example:** MOV B, C or MOV B, M
M, Rs	

Rd, M Move immediate 8-bit

MVI Rd, data	The 8-bit data is stored in the destination register or Memory. If the operand is a memory location, its location is specified by the contents of the HL registers. **Example:** MVI B, 57H or MVI M, 57H

Load accumulator

LDA 16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents o f the source are not altered. **Example:** LDA 2034H

Load accumulator indirect

LDAX B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. **Example:** LDAX B

Load register pair immediate

LXI Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair **Example:** LXI H, 2034H

Load H and L registers direct

LHLD 16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the Contents of the next memory location into register H. The Contents of source memory locations are not altered. **Example:** LHLD 2040H

Store Accumulator Direct

STA 16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example:** STA 4350H

Store Accumulator Indirect

STAX Reg Pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. **Example:** STAX B

Store H and L Registers Direct

SHLD 16 bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example: SHLD 2470H.**

Exchange H and L with D and E

XCHG none	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. **Example:** XCHG

2. ARITHMETIC INSTRUCTION:

- They perform arithmetic operations, such as, addition, subtraction, increment, and decrement.

Addition:

- Addition of any 8-bit number, or the contents of a register or the contents of a memory location is added to the contents of the accumulator and the sum is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- For example, the contents of register B cannot be added directly to the contents of the register C. 8085 can also perform 16-bit. It can also perform BCD addition.

Subtraction:

- Subtraction of any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator.
- The subtraction is performed in 2's compliment, and if the results is negative. Then they are expressed in 2's complement.
- No two other registers can be subtracted directly. 8085 do not perform 16-bit subtraction.

Increment or Decrement:

- The 8-bit contents of any register or a memory location can be incremented or decrement by 1.
- Similarly, the 16-bit contents of a register pair can be incremented or decrement by 1.
- These increment and decrement operations can be performed directly in the source itself. It means without using accumulator.

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADD R,ADDM
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADC R,ADDM

ADI	8-bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADI 55
ACI	8-bit data	Add the immediate to the accumulator with carry	The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.**Example** − ACI 55
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16- bit data into the register pair designated in the operand. **Example** − LXI H, 3025H
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. **Example** − DAD
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. **Example** − SUB R,SUB M
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. **Example** − SBB R,SBBM
SUI	8-bit data	Subtract the immediate from the accumulator	The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator. **Example** − SUI 55
SBI	8-bit data	Subtract the immediate from the accumulator with borrow	The 8-bit data and borrow is subtracted from the contents of the accumulator & the result is stored in the accumulator

INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. **Example** − INR R, INR M
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. **Example** − INX R
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. **Example** − DCR R,DCR M
DCX	R	Decrement the register pair by 1	The contents of the designated register pair are decremented by 1 and their result is stored at the same place. **Example** − DCX R
DAA	None	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit BCD digits.If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits.If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. **Example** − DAA

3. LOGICAL INSTRUCTION:

- These type instructions performs various logical operations with the contents of the accumulator. 8085 can perform six logical operation which are:
  - AND

- OR
- Exclusive-OR
- NOT
- Compare
- Rotate

- An 8-bit number can be logically ANDed with the contents of the accumulator. It can also be a content of register or of a memory location. The results are stored in the accumulator. The content of the accumulator can be complimented.

Rotate:

- Each bit of the accumulator can be shifted either left or right to the next position.

Compare:

- Any 8-bit number or the content of a register, or content of a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.
- The result is reflected by zero and carry flags.

Opcode	Operand	Meaning	Explanation
CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	8-bit data	Compare immediate with the accumulator	The second byte data is compared with the contents of the accumulator.
ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and
			the result is placed in the accumulator.

XRA	R M	Exclusive OR register or memory with the accumulator	The contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
XRI	8-bit data	Exclusive OR immediate with the accumulator	The contents of the accumulator are Exclusive OR with the 8-bit data and the result is placed in the accumulator.
ORA	R M	Logical OR register or memory with the accumulator	The contents of the accumulator are logically OR with M the contents of the register or memory, and result is placed in the accumulator.
ORI	8-bit data	Logical OR immediate with the accumulator	The contents of the accumulator are logically OR with the 8-bit data and the result is placed in the accumulator.
RLC	None	Rotate the accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
RRC	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
RAL	None	Rotate the accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.
RAR	None	Rotate the accumulator	Each binary bit of the accumulator is rotated right by one position

		right through carry	through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
CMA	None	Complement accumulator	The contents of the accumulator are complemented. No flags are affected.
CMC	None	Complement carry	The Carry flag is complemented. No other flags are affected.
STC	None	Set Carry	Set Carry

4. BRANCH INSTRUCTION:

- This group of instruction transfers the control of microprocessor from one location to another location. 8085 can perform four types of branching operations. These are:
  - JMP-Jump within a program.
  - CALL-Jump from main program to sub-routine.
  - RET-Jump from sub-routine to main program.
  - RST-Jump from main program to instruction sub routine.

Jump:

- Conditional jumps are the important aspect of the decision-making process in the programming of a microprocessor.
- These instructions tests for a certain condition and alter the program sequence when the condition is met.
- For example, zero or carry flag, in addition, the instruction set also includes an instruction called unconditional jump.

Call, return, and restart:

- These type of instructions changes the sequence of a program either by calling a sub-routine or returning from a sub-routine.
- The conditional call and return instructions can also test the condition flags.

1. Jump Instructions: –

- The jump instruction transfers the program sequence to the memory address given in the operand based on the specified flag. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Jump Instructions:

- Transfers the program sequence to the described memory address.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JMP	address	Jumps to the address	JMP 2050

(b) Conditional Jump Instructions:

- Transfers the program sequences to the described memory address only if the condition in satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JC	address	Jumps to the address if carry flag is 1	JC 2050
JNC	address	Jumps to the address if carry flag is 0	JNC 2050
JZ	address	Jumps to the address if zero flag is 1	JZ 2050
JNZ	address	Jumps to the address if zero flag is 0	JNZ 2050
JPE	address	Jumps to the address if parity flag is 1	JPE 2050
JPO	address	Jumps to the address if parity flag is 0	JPO 2050
JM	address	Jumps to the address if sign flag is 1	JM 2050
JP	address	Jumps to the address if sign flag 0	JP 2050

2. Call Instructions: –

- The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

(a) Unconditional Call Instructions:

- It transfers the program sequence to the memory address given in the operand.

OPCODE	OPERAND	EXPLANATION	EXAMPL
CALL	address	Unconditionally calls	CALL 2050

(b) Conditional Call Instructions:

- Only if the condition is satisfied, the instructions execute.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
C	address	Call if carry flag is 1	CC 2050
CN	address	Call if carry flag is 0	CNC 2050
C	address	Calls if zero flag is 1	CZ 2050
CN	address	Calls if zero flag is 0	CNZ 2050
CP	address	Calls if parity flag is 1	CPE 2050

CP	address	Calls if parity flag is 0	CPO 2050
C	address	Calls if sign flag is 1	CM 2050
C	address	Calls if sign flag is 0	CP 2050

3. Return Instructions: –

- The return instruction transfers the program sequence from the subroutine to the calling program. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Return Instruction:

- The program sequence is transferred unconditionally from the subroutine to the calling program.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RET	none	Return from the subroutine unconditionally	RET

(b) Conditional Return Instruction:

- The program sequence is transferred unconditionally from the subroutine to the calling program only is the condition is satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RC	none	Return from the subroutine if carry flag is 1	RC
RNC	none	Return from the subroutine if carry flag is 0	RNC
RZ	none	Return from the subroutine if zero flag is 1	RZ
RNZ	none	Return from the subroutine if zero flag is 0	RNZ
RPE	none	Return from the subroutine if parity flag is 1	RPE
RPO	none	Return from the subroutine if parity flag is 0	RPO
RM	none	Returns from the subroutine if sign flag is 1	RM
RP	none	Returns from the subroutine if sign flag is 0	RP

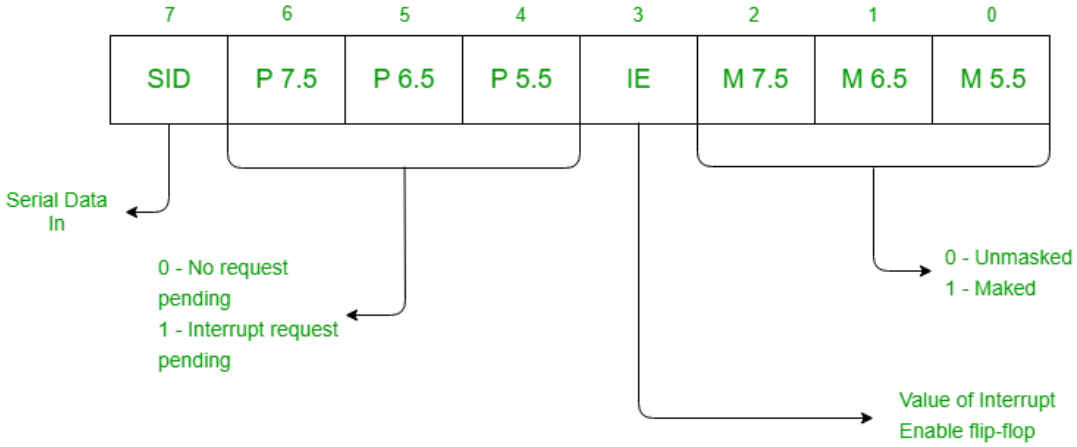
5. STACK, I/O &MACHINE-CONTROL INSTRUCTION:

- These type of instructions controls the machine functions, such as halt, interrupt, or do nothing.

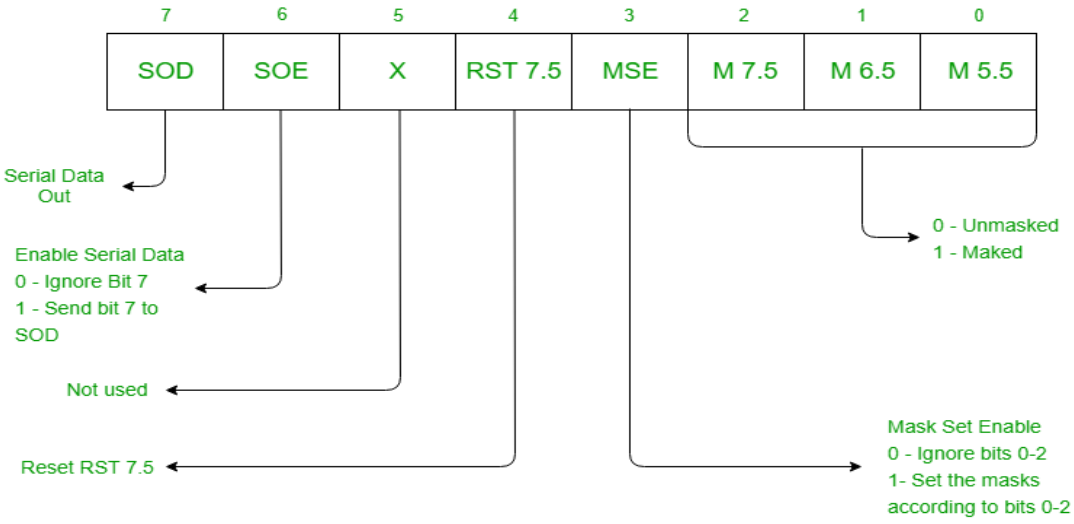
Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.

RIM Instruction

HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.



SIM Instruction



Stack instructions are as follows:

PUSH - Push Two bytes of Data onto the Stack

POP - Pop Two Bytes of Data off the Stack

XTHL - Exchange Top of Stack with H & L

SPHL - Move content of H & L to Stack Pointer

I/O instructions are as follows:

IN - Initiate Input Operation

OUT - Initiate Output Operation

❖ ASSEMBLY LANGUAGE PROGRAMMING OF 8085:

What is Assembly Language Program?

- Machine language and Hex code instructions are very difficult for the programmer.
- Hence for programmer, the instructions of microprocessor are made in the form of English abbreviation (short form). These instructions are name as Assembly Language instructions or mnemonics.
- The combinations of different mnemonics are known as Assembly Language Program and it is a low-level language.

Examples of assembly language program

Loading Register or Memory with Data

Example 1: Write a program to transfer 07 H in register L.

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	2E, 07	MVI	L, 07	Move immediate 07 in register L
2002 H	76	HLT		Stop or terminate the program

Explanation:

- The instruction MVI L, 07 will move the data 07 to the register L.
- The instruction will stop the program.
- The machine code for the instruction MVI L, 07 is 2E, 07.
- The 1st byte of the machine code is 2E which is the Hex code for the instruction MVI L.
- The second byte is the data 07. The machine code for HLT is 76.
- The machine codes are fetched in the memory locations, starting from the memory locations 2000 H.
- Memory location 2000 H contains 2E, 2001 H contains 07 and memory location 2002 H contain 76, After the execution of a program, the contents of Register L can be examined which are 07.

Example 2 Write a program to load register A with 08 H and then move it to register C.

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	3E, 08	MVI	A,08	Get 08 in register A
2002 H	4F	MOV	C, A	Move the contents of register A to register C
2003 H	76	HLT		Halt

Explanation:

- In this program the instruction MVI A, ON H will place the given data 08 1H in the register A.
- The Hex code for MVI A, 08 H is 3E, 08 IH where 3E is the Hex code for MVI A.
- The instruction MOV C, A will move the contents of register A to the register C. Its machine code is 4F.
- With this instruction the data of register A is copies into the register C. It means the given data, is 08 H which was previously placed in register A is now copied into the register C.
- The instruction HLT whose machine code is 76 stops the program.
- The memory locations required for this program are 2000 H to 2003 H. Any other memory locations can be selected. After the execution of a program, the contents of register C can be examined.

Example 3. Write a program to load the contents of memory location 2050 H into accumulator and then move this data into register B

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	3A, 50, 20	LDA	2050 H	Load the contents of memory location 2050 H into the accumulator
2002 H	47	MOV	B,A	Move the contents of register A to register B
2004 H	76	HLT		Stop

Explanation:

- The instruction LDA 2050 H will load the contents of memory location 2050 H into the accumulator.
- The machine code for the instruction LDA is 3A.
- The instruction MOV B. A (Machine code 47) will move the contents of Accumulator to the register B.
- First of all data 07 is fetch in the memory location 2050.
- Then memory locations 2000 H contain 3A, 2001 H contain 50 H, 2002 H contains 20 H, 2003 H contains 47 H and 2004 H contains 76 H.
- After execution of a program, the contents of register B can be examined.

Example 4. Write a program to add two 8-bit numbers.

MEMORY	MACHINE CODE	MNEMONICS	OPERANDS	COMMENTS
2000	21,01,25	LXI	H,2501H	Get address of first number in H-L pair.
2003	7E	MOV	A, M	1st number in accumulator.
2004	23	INX	H	Increment content of H-L
2005	86	ADD	M	Add 1st and 2nd numbers.
2006	32,03,25	STA	2503H	Store sum in 2503H.
2009	76	HLT		Stop the program.

Explanation:

- The 1stnumber was stored in the memory location 2501H.
- 2501 was placed in H-L pair by the execution of the instruction LXI H, 2501H.
- The instruction MOV A, M moved the content of the memory location addressed by H-L pair to the accumulator.
- Thus, the 1stnumber 49H which was in the 2501H was placed in the accumulator.
- The INX H increased the content of H-L pair from 2501 to 2502H.
- The instruction ADD M added the content of the memory location addressed by H-L pair with the accumulator.
- The result got stored in the accumulator.
- The instruction STA 2503H stored the sum in the memory location 2503H.
- The instruction HLT ended the program.

UNIT-3 INTERRUPTS IN 8085

❖ OVERVIEW OF INTERRUPT:

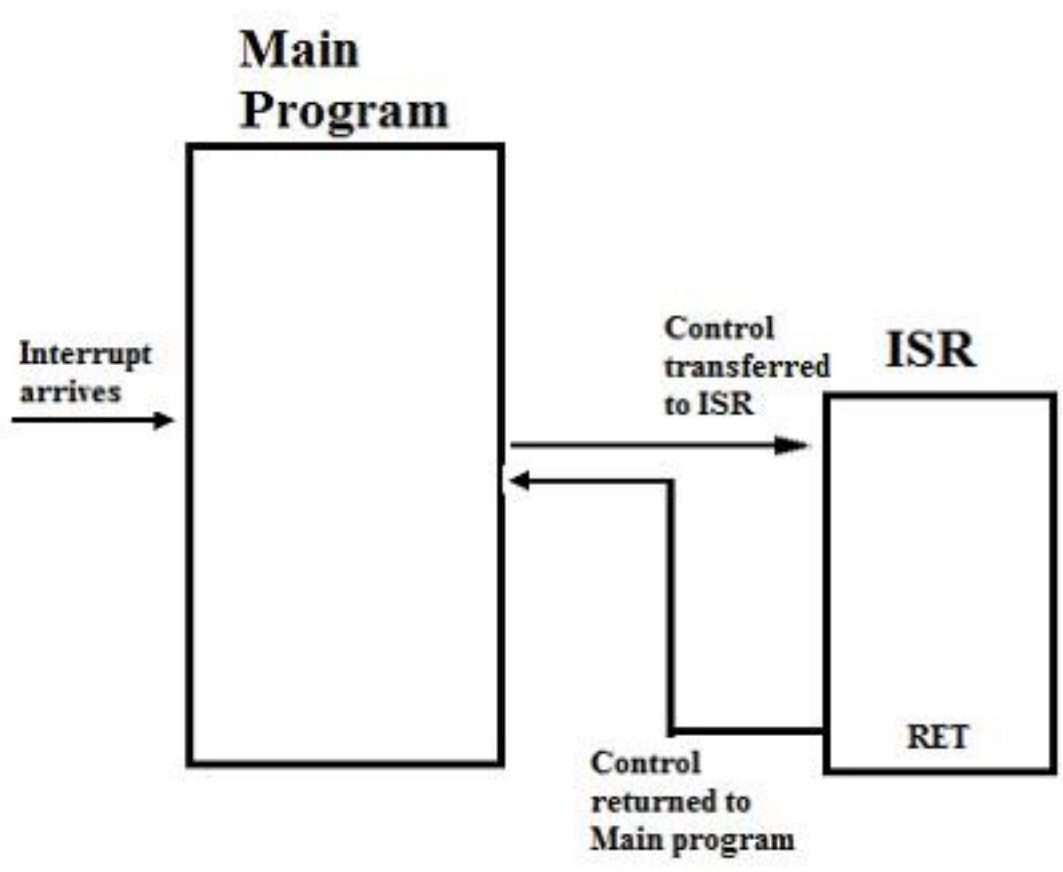
Interrupts are the signals generated by the external devices to request the microprocessor to perform a task. For transferring data between the peripheral and the microprocessor, interrupts are mainly used.

An interrupt is considered to be an emergency signal that may be serviced. The Microprocessor may respond to it as soon as possible.

- **What happens when MP is interrupted?**

When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an Interrupt Service Routine (ISR) to respond to the incoming interrupt. Each interrupt will most probably have its own ISR.

If an interrupt occurs, it accepts the interrupt and sends the INTR' signal to the peripheral. The microprocessor executes an interrupt service routine (ISR) stored in memory. It returns to the main program by RET instruction, after the ISR is executed. The interrupt process is shown in figure.



Interrupt process

❖ TYPES OF INTERRUPTS:

There are six types of Interrupts in 8085.

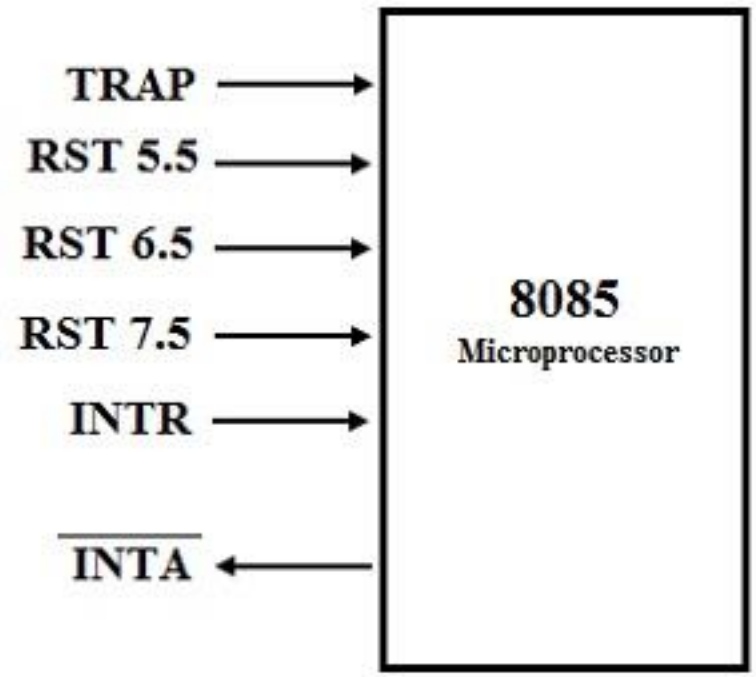
1. Hardware interrupts
2. Software interrupts
3. Maskable interrupts
4. Non-Maskable interrupts
5. Vectored interrupts
6. Non-vectored interrupts

1. **Hardware interrupts:** These interrupts are given by the peripheral devices to the interrupt pin (hardware) of the microprocessor. Hardware interrupts are also called external interrupts.
2. **Software interrupts:** These interrupts are internally generated within the microprocessor using software instructions. Software interrupts are also called internal interrupts.
3. **Maskable interrupts (Can be delayed or rejected):** These external interrupts can be delayed or rejected by the microprocessor.
4. **Non-maskable interrupts (Cannot be delayed or rejected):** These external interrupts cannot be delayed or rejected by the microprocessor. Non-maskable interrupts are used for handling emergency situations.
5. **Vectored interrupts:** When the address of the Interrupt Service Routine (ISR) is fixed within the microprocessor itself, then the interrupt is called Vectored interrupt.
6. **Non-vectored interrupts:** When the address of the Interrupt Service Routine (ISR) is supplied by the peripheral device, then the interrupt is called Non-vectored interrupt.

❖ 8085 HARDWARE INTERRUPTS:

In 8085 microprocessors, there are 5 hardware interrupts as shown in figure.

1. TRAP
2. RST 5.5
3. RST 6.5
4. RST 7.5
5. INTR



In additional to these hardware interrupts, 8085 microprocessors have eight software interrupts.

❖ **8085 SOFTWARE INTERRUPT:**

- The software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if software interrupt instruction is encountered, then the processor executes an interrupt service routine (ISR).
- When the instruction is executed, the processor executes an interrupt service routine stored in the vector address of the software interrupt instruction. The software interrupts of 8085 are RST 0, RST1, RST 2, RST 3, RST 4, RST 5, RST6 and RST 7.
- All software interrupts of 8085 is vectored interrupts. The software interrupts cannot be masked and they cannot be disabled.

❖ **MASKABLE INTERRUPTS:**

- Maskable interrupts are those interrupts which can be enabled or disabled. Enabling and disabling is done by software instructions. The interrupts can be masked by moving an appropriate data to accumulator and then executing SIM instruction. (SIM - Set Interrupt Mask). The status of maskable interrupts can be read into accumulator by executing RIM instruction (RIM - Read Interrupt Mask). List of Maskable Interrupts: RST 7.5, RST 6.5 ,RST 5.5 ,INTR

❖ **NON-MASKABLE INTERRUPTS:**

- The interrupts which are always in enabled mode are called non maskable interrupts. These interrupts can never be disabled by any software instruction. TRAP is a non-maskable interrupt.

❖ VECTORED INTERRUPTS:

- The interrupts which have fixed memory location for transfer of control from normal execution.

List of vectored interrupts:

1. RST 7.5
2. RST 6.5
3. RST 5.5
4. TRAP

The addresses to which program control goes for vectored interrupts:

Name	Vectored Addres
RST 7.5	003C H (7.5 x 0008 H)
RST 6.5	0034 H (6.5 x 0008 H)
RST 5.5	002C H (5.5 x 0008 H)
TRAP	0024 H (4.5 x 0008 H)

❖ NON-VECTORED INTERRUPTS:

- The interrupts which don't have fixed memory location for transfer of control from normal execution is called Non-Vectored Interrupts. The address of the memory location is sent along with the interrupt. INTR is a non-vectored interrupt.

❖ Interrupt priority:

- The microprocessor can respond to only one interrupt at one time. When multiple (more than one) interrupts occur simultaneously, the microprocessor will service the interrupts in their fixed priority order. Interrupt having the highest priority level will be serviced first.

In 8085, TRAP interrupt has the highest priority and INTR has the lowest priority.

➢ TRAP

- This interrupt can be considered as a non-maskable interrupt. Any mask or interrupt enable cannot affect this.
- It is a vectored interrupt. The interrupt vector address is 0024H.
- TRAP has the highest priority level.
- We can say that TRAP interrupt is level and edge triggered. This means that till the acknowledgement, the TRAP must go high and remain high.

- In emergency situations like sudden power failure, it executes an ISR and sends the data from main memory to backup memory.

➢ **RST 7.5**

- The RST 7.5 can be considered as a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 003CH.
- It has second highest priority.
- It is edge triggered. i.e. Input attains at high and no need to retain the high state until it is recognized and acknowledged.

➢ **RST 6.5**

- The RST 6.5 interrupt is a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 0034H.
- It has the third highest priority.
- It is level triggered. i.e. Input goes to high and stays high until it is recognized and acknowledged.

➢ **RST 5.5**

- The RST 5.5 interrupt is a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 002CH.
- It has the fourth highest priority.
- It is level triggered. i.e. Input goes to high and stays high until it is recognized and acknowledged.

➢ **INTR**

- INTR is a maskable interrupt.
- It is a non- vectored interrupt.
- It has the lowest priority.
- It is a level triggered. i.e. Input goes to high and it is necessary to maintain high state until it is recognized and acknowledged.

❖ INSTRUCTIONS FOR INTERRUPTS HANDLING IN 8085 MICROPROCESSORS:

There are four instructions available for interrupts handling. They are,

1. DI (Disable Interrupt)
2. EI (Enable Interrupt)
3. SIM (Set Interrupt Mask)
4. RIM (Read Interrupt Mask)

➢ DI (Disable Interrupt)

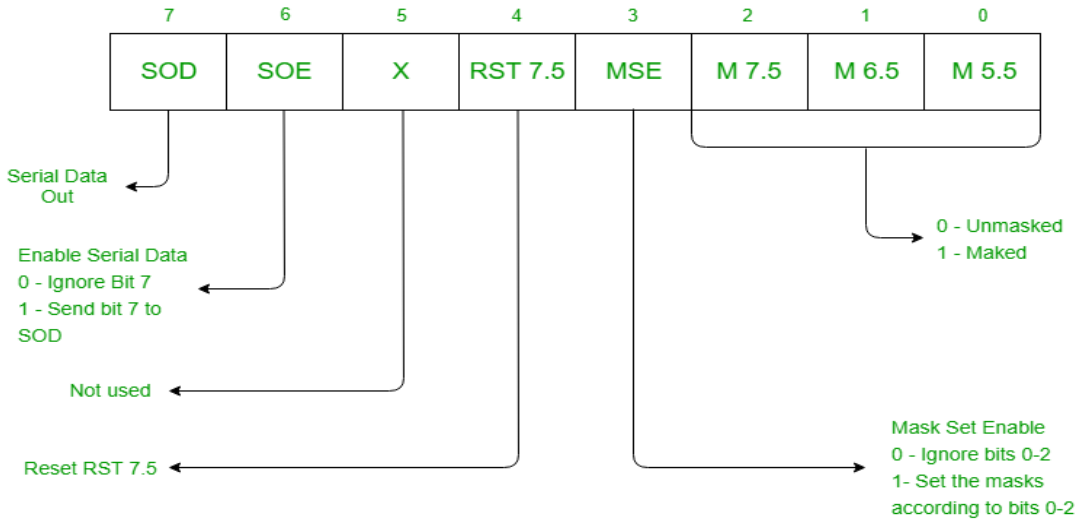
This instruction resets the Interrupt Enable Flip-flop inside the microprocessor. All the interrupts except the TRAP are disabled.

➢ EI (Enable Interrupt)

Inside the microprocessor, this instruction sets the Interrupt Enable Flip-flop also all the interrupts are enabled.

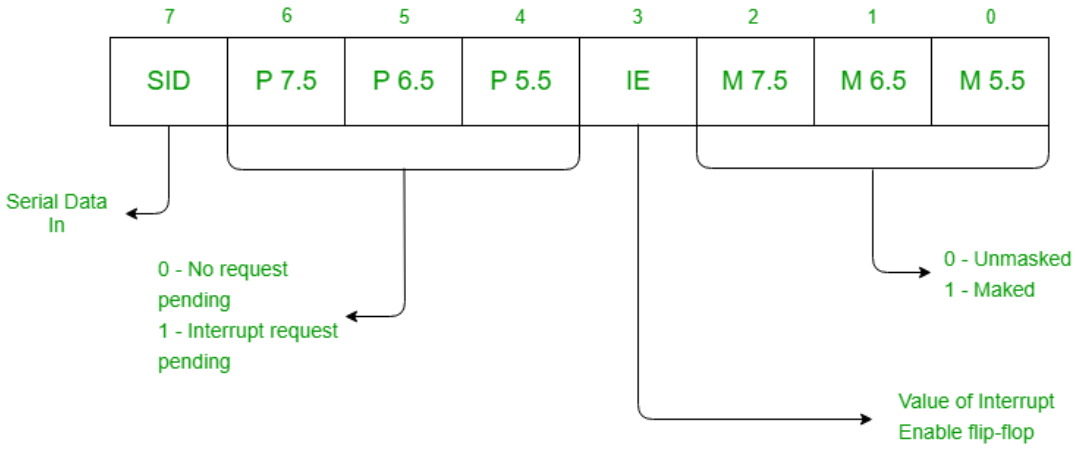
➢ SIM (Set Interrupt Mask)

This instruction is used to selectively mask (disable) and unmask (enable) RST 7.5, RST 6.5 and RST 5.5 interrupts. For serial data output, we can use this instruction. The SIM the instruction uses the accumulator contents for masking and unmasking the interrupts.



➢ RIM (Read Interrupt Mask)

This instruction is used to read the status of RST 7.5, RST 6.5 and RST 5.5 interrupts like pending and enable / disable details. This instruction is also used for reading the serial data. When the RIM instruction is given, the microprocessor loads the details into the accumulator.



❖ PROCESS OF INTR INTERRUPT:

1. With the use of the EI instruction, the interrupt process should be enabled.
2. Whenever an instruction is executed, the 8085 checks for an interrupt signal.
3. If INTR is high, the microprocessor completes current instruction, disables the interrupt and sends INTR' signal to the peripheral device.
4. INTR' allows the peripheral device to send an RST instruction through data bus.
5. Upon receiving the INTR' signal, the microprocessor saves the memory location of the next instruction on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction.
6. Microprocessor executes the ISR.
7. Within the program, in order to enable the further interrupt, ISR must include the 'EI' instruction.
8. The RET instruction at the end of the ISR retrieves the return address from the stack and the program is transferred back to main program which was interrupted.

❖ INTERRUPT VECTOR AND THE VECTOR TABLE:

An interrupt vector is a pointer to where the ISR is stored in memory. All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT). The IVT is usually located in memory page 00 (0000H - 00FFH). The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.

Example: Let, a device interrupts the Microprocessor using the RST 7.5 interrupt line.

Because the RST 7.5 interrupt is vectored, Microprocessor knows, in which memory location it has to go using a call instruction to get the ISR address.

RST7.5 is known as Call 003Ch to Microprocessor. Microprocessor goes to 003C location and will get a JMP instruction to the actual ISR address. The Microprocessor will then, jump to the ISR location.

❖ THE 8085 NON-VECTORED INTERRUPT PROCESS:

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If INTR is high, MP completes current instruction, disables the interrupt and sends INTA (Interrupt ~~ackno~~wledge)signal to the device that interrupted
4. INTA allows the I/O device to send a RST instruction through data bus.
5. Upon receiving the INTA signal, MP saves the memory location of the next instruction on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction
6. Microprocessor Performs the ISR.
7. ISR must include the 'EI' instruction to enable the further interrupt within the program.
8. RET instruction at the end of the ISR allows the MP to retrieve the return address from the stack and the program is transferred back to where the program was interrupted.

❖ SUMMARY OF 8085 INTERRUPTS:

INTERRUPT	VECTOR ADDRESS	PRIORITY	TYPE
TRAP	0024H	1(highest priority)	Hardware interrupt Vectored interrupt Non-maskable interrupt
RST 7.5	003CH	2	Hardware interrupt Vectored interrupt Maskable interrupt
RST 6.5	0034H	3	Hardware interrupt Vectored interrupt Maskable interrupt
RST 5.5	002CH	4	Hardware interrupt Vectored interrupt Maskable interrupt
INTR	-	5(lowest priority)	Hardware interrupt Non - Vectored interrupt Maskable interrupt

RST instruction	RST 0 – 0000H RST 1 – 0008H RST 2 – 0010H RST 3 – 0018H RST 4 – 0020H RST 5 - 0028H RST 6 – 0030H RST 7 – 0038H	-	Software interrupt Vectored interrupt Maskable interrupt

UNIT 4 INTERFACING USING 8085 MICRO PROCESSOR

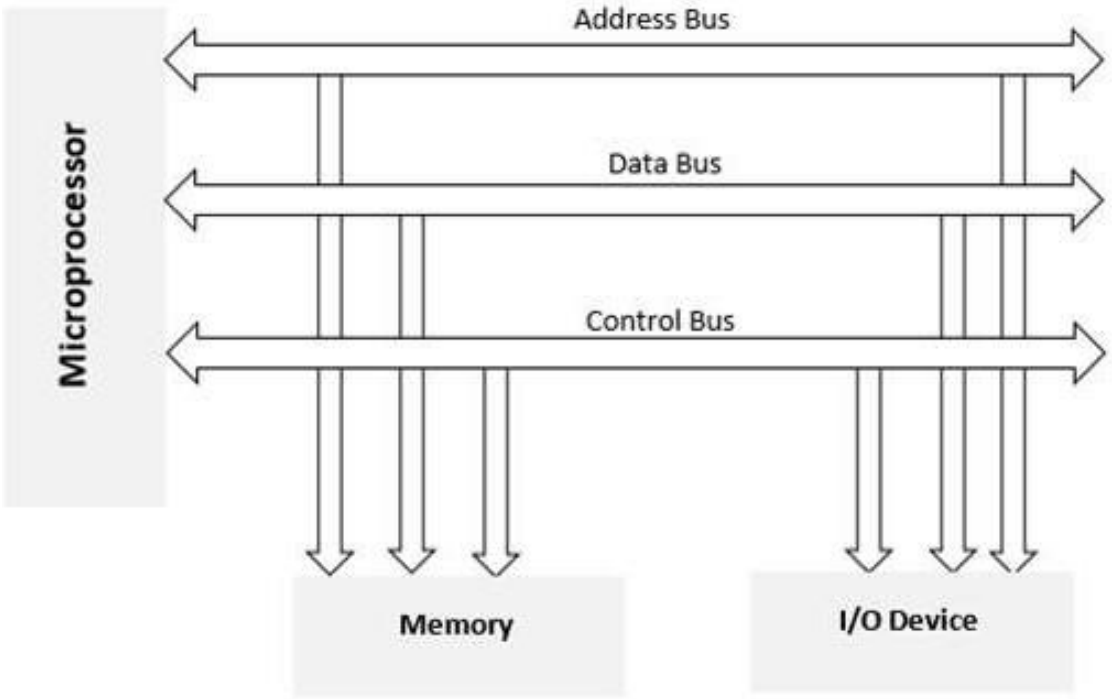
- **INTERFACE:** Interface is the path for communication between two components. Interfacing is of two types, memory interfacing and I/O interfacing.

- **MEMORY INTERFACING:**

  When we are executing any instruction, we need the microprocessor to access the memory for reading instruction codes and the data stored in the memory. For this, both the memory and the microprocessor require some signals to read from and write to registers. The interfacing process includes some key factors to match with the memory requirements and microprocessor signals. The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.

- **I/O INTERFACING:**

  There are various communication devices like the keyboard, mouse, printer, etc. So, we need to interface the keyboard and other devices with the microprocessor by using latches and buffers. This type of interfacing is known as I/O interfacing.



Block Diagram of Memory & I/O Interfacing

❖ 8085 INTERFACING PINS

Following is the list of 8085 pins used for interfacing with other devices –

- A_{15} - A_8 (Higher Address Bus)
- AD_7 - AD_0(Lower Address/Data Bus)
- ALE
- RD
- WR
- READY

(Note: Explanation of all the above pins is available in Unit-1 notes)

❖ WAYS OF COMMUNICATION- MICRPROCESSOR WITH THE OUTSIDE WORLD:

There are two ways of communication in which the microprocessor can connect with the outside world.

- Serial Communication Interface
- Parallel Communication interface
  - **Serial Communication Interface** − In this type of communication, the interface gets a single byte of data from the microprocessor and sends it bit by bit to the other system serially and vice-a-versa.
  - **Parallel Communication Interface** − In this type of communication, the interface gets a byte of data from the microprocessor and sends it bit by bit to the other systems in simultaneous (or) parallel fashion and vice-a-versa.

❖ 8257 DMA CONTROLLER:

DMA stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference of the CPU. Using a DMA controller, the device requests the CPU to hold its data, address and control bus, so the device is free to transfer data directly to/from the memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU.

➢ How DMA Operations are performed?

Following is the sequence of operations performed by a DMA −

- Initially, when any device has to send data between the device and the memory, thedevice has to send DMA request (DRQ) to DMA controller.
- The DMA controller sends Hold request (HRQ) to the CPU and waits for the CPU toassert the HLDA.

- Then the microprocessor tri-states all the data bus, address bus, and control bus. The CPU leaves the control over bus and acknowledges the HOLD request through HLDA signal.
- Now the CPU is in HOLD state and the DMA controller has to manage the operations over buses between the CPU, memory, and I/O device

➢ **Features of 8257**

Here is a list of some of the features of 8257 −

- It has four channels which can be used over four I/O devices.
- Each channel has 16-bit address and 14-bit counter.
- Each channel can transfer data up to 64kb.
- Each channel can be programmed independently.
- Each channel can perform read transfer, write transfer and verify transfer operations.
- It generates MARK signal to the peripheral device that 128 bytes have been transferred.
- It requires a single-phase clock.
- Its frequency ranges from 250Hz to 3MHz.
- It operates in 2 modes, i.e., **Master mode** and **Slave mode**.

❖ **8255A - PROGRAMMABLE PERIPHERAL INTERFACE:**

The 8255A is a general purpose programmable I/O device designed to transfer the data fromI/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor. It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

➢ **Ports of 8255A**

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

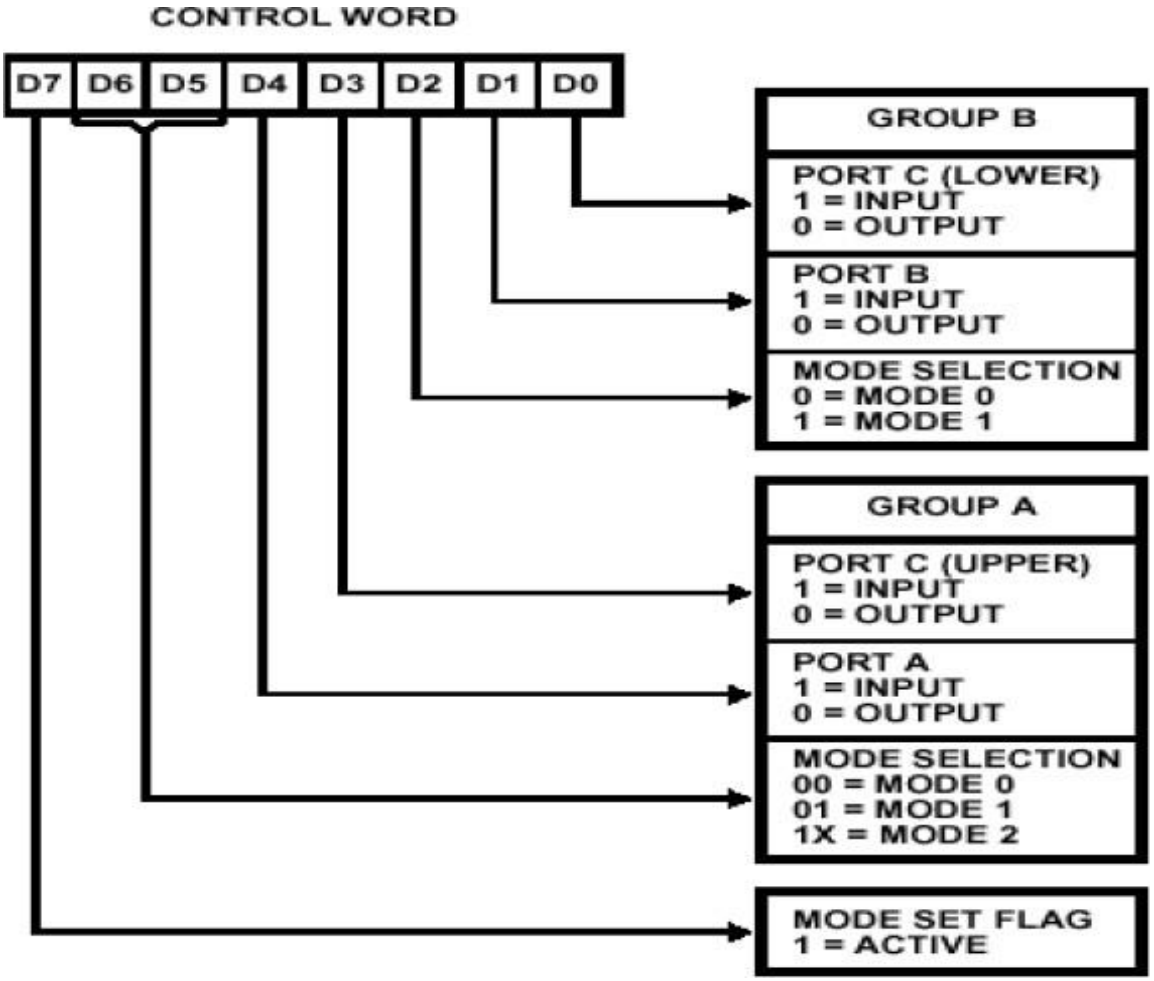
- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

➢ **Operating Modes:**

8255A has three different operating modes −

- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4- bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B. Fig shows control word of 8255A.



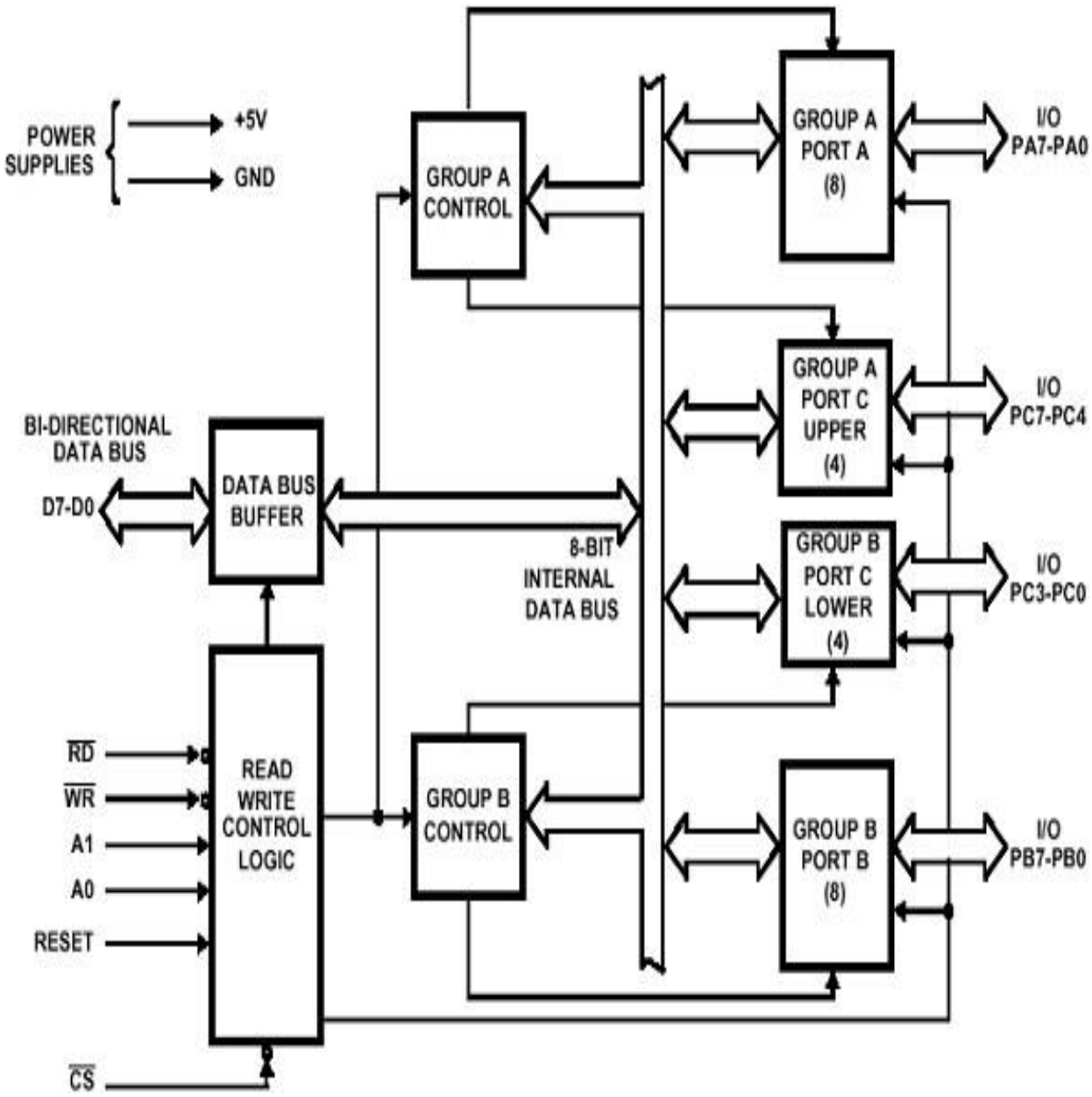
- **Features of 8255A**

The prominent features of 8255A are as follows −

- It consists of 3 8-bit IO ports i.e. PA, PB, and PC.
- Address/data bus must be externally demultiplexed.
- It is TTL compatible.
- It has improved DC driving capability.

- **8255 Architecture:**

The following figure shows the architecture of 8255A −



Now let us discuss the functional description of the pins in 8255A.

- **Data Bus Buffer:**

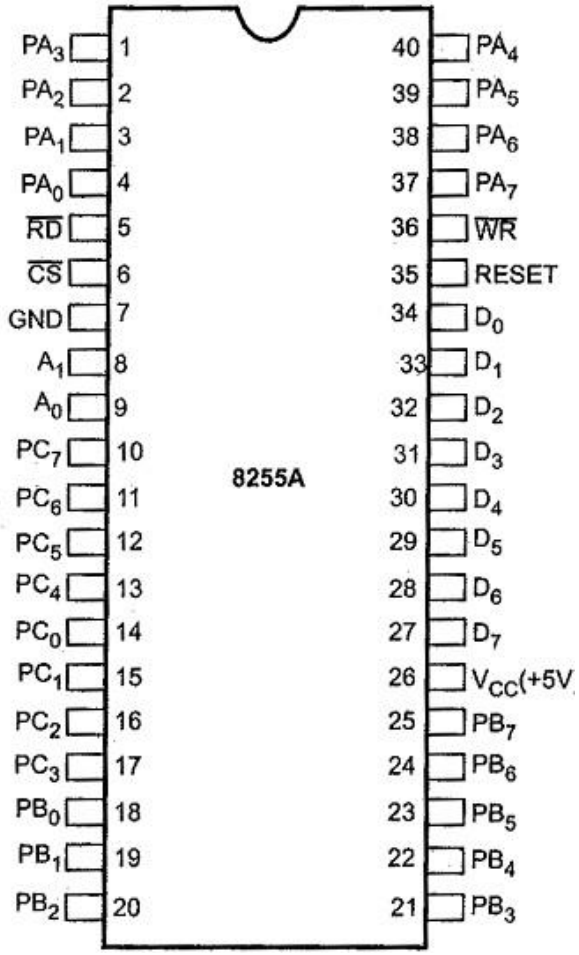
  It is a tri-state 8-bit buffer, which is used to interface the microprocessor to the system data bus. Data is transmitted or received by the buffer as per the instructions by the CPU. Controlwords and status information is also transferred using this bus.

- **Read/Write Control Logic:**

  This block is responsible for controlling the internal/external transfer of data/control/status word. It accepts the input from the CPU address and control buses, and in turn issues command to both the control groups.

- **CS:**

  It stands for Chip Select. A LOW on this input selects the chip and enables the communication between the 8255A and the CPU. It is connected to the decoded address, and A_0 & A_1 are connected to the microprocessor address lines.



Their result depends on the following conditions –

CS	A_1	A_0	Result
0	0	0	**PORT A**
0	0	1	**PORT B**
0	1	0	**PORT C**
0	1	1	**Control Register**
1	**X**	**X**	**No Selection**

- **WR:**

  It stands for write. This control signal enables the write operation. When this signal goeslow, the microprocessor writes into a selected I/O port or control register.

- **RESET**

  This is an active high signal. It clears the control register and sets all ports in the input mode.

- **RD**

  It stands for Read. This control signal enables the Read operation. When the signal is low,the microprocessor reads the data from the selected I/O port of the 8255.

- **A_0 and A_1**

A_1	A_0	RD	WR	CS	Result
0	0	0	1	0	**<u>Input Operation</u>** **PORT A → Data Bus**
0	1	0	1	0	**PORT B → Data Bus**
1	0	0	1	0	**PORT C → Data Bus**
0	0	1	0	0	**<u>Output Operation</u>** **Data Bus → PORT A**
0	1	1	0	0	**Data Bus → PORT A**
1	0	1	0	0	**Data Bus → PORT B**
1	**1**	**1**	**0**	**0**	**Data Bus → PORT D**

These input signals work with RD, WR, and one of the control signals. Following is the table showing their various signals with their result.

❖ INTEL 8253/54 - PROGRAMMABLE INTERVAL TIMER:

The Intel 8253 and 8254 are Programmable Interval Timers (PTIs) designed for microprocessors to perform timing and counting functions using three 16-bit registers. Each counter has 2 input pins, i.e. Clock & Gate, and 1 pin for "OUT" output. To operate a counter, a 16-bit count is loaded in its register. On command, it begins to decrement the count until it reaches 0, then it generates a pulse that can be used to interrupt the CPU.

Difference between 8253 and 8254

8253	8254
Its operating frequency is 0 - 2.6 MHz	**Its operating frequency is 0 - 10 MHz**
It uses N-MOS technology	**It uses H-MOS technology**
Read-Back command is not available	**Read-Back command is available**
Reads and writes of the same counter cannotbe interleaved.	**Reads and writes of the same counter canbe interleaved.**

Features of 8253/54

The most prominent features of 8253/54 are as follows −

- It has three independent 16-bit down counters.
- It can handle inputs from DC to 10MHz.
- These three counters can be programmed for either binary or BCD count.
- It is compatible with almost all microprocessors.

8254 has a powerful command called READ BACK command, which allows the user to check the count value, the programmed mode, the current mode, and the status of the count

❖ INTEL 8259A PROGRAMMABLE INTERRUPT CONTROLLER:

Intel 8259 is a Programmable Interrupt Controller (PIC). There are 5 hardware interrupts and 2 hardware interrupts in Intel 8085 and Intel 8086 microprocessors respectively. But by connecting Intel 8259 with these microprocessors, we can increase their interrupt handling capability. Intel 8259 combines the multi-interrupt input sources into a single interrupt output. Interfacing of single PIC provides 8 interrupts inputs from IR0-IR7. For example, Interfacing of 8085 and 8259 increases the interrupt handling capability of 8085 microprocessor from 5 to 8 interrupt levels.

Features of Intel 8259 PIC are as follows:

- Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
- It can be programmed either in level triggered or in edge triggered interrupt level.
- We can mask individual bits of interrupt request register.
- We can increase interrupt handling capability up to 64 interrupt level by cascading further 8259 PICs.
- Clock cycle is not required.

UNIT-5 ADVANCED MICROPROCESSOR

❖ OVERVIEW OF 8086 MICROPROCESSOR:

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor

❖ FEATURES OF 8086 MICROPROCESSOR:

1. Single +5V power supply
2. Clock speed range of 5-10MHz
3. Capable of executing about 0.33 MIPS (Millions instructions per second)
4. It is 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
5. It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
6. Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
7. It has 256 interrupts.

❖ 8086 ARCHITECTURES:

- The above diagram depicts the architecture of an 8086 Microprocessor.
- 8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

EU (Execution Unit)

- Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

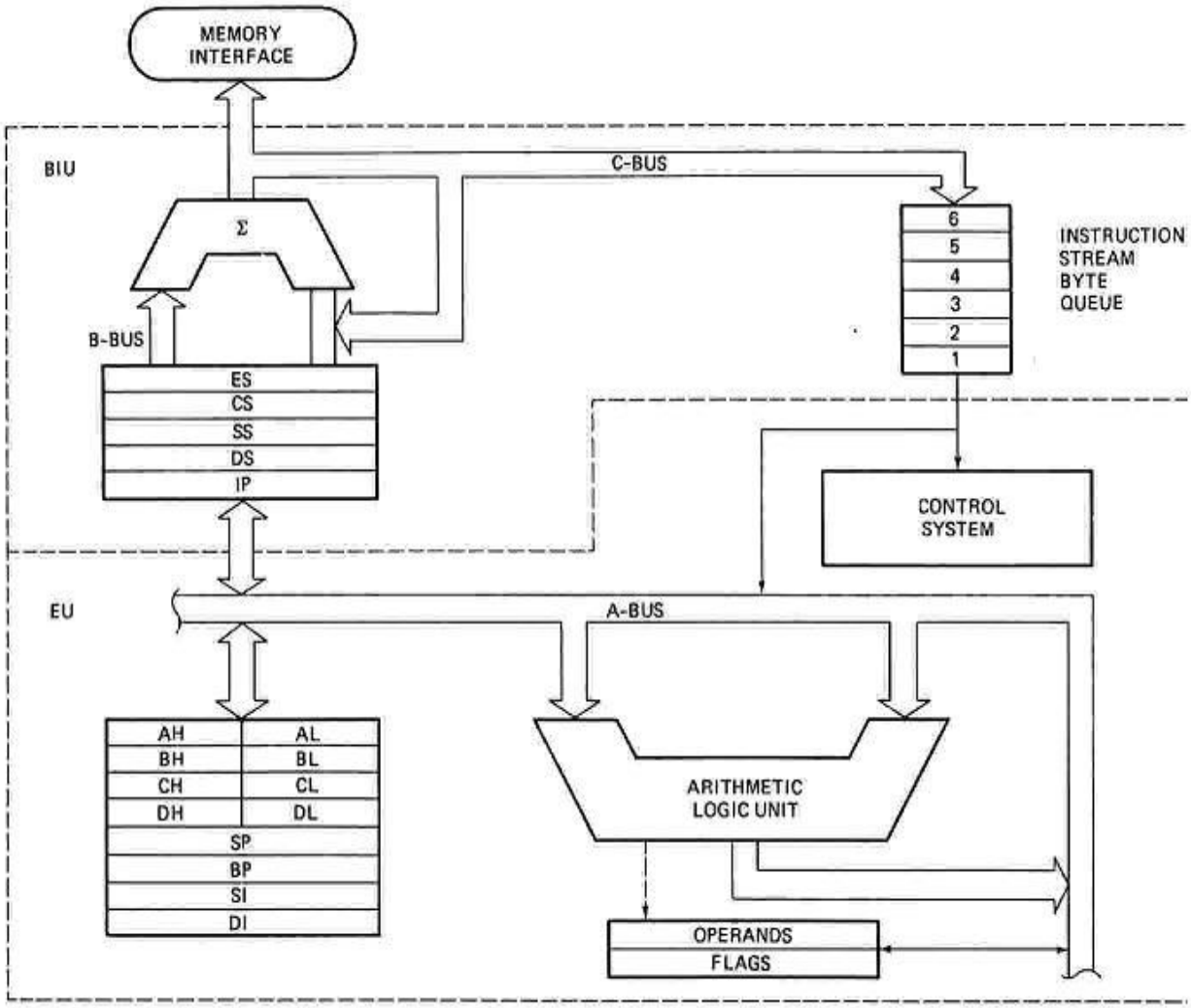
  Let us now discuss the functional parts of 8086 microprocessors.

- **ALU**

  It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

- **Flag Register**

  It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − Conditional Flags and Control Flags.



- **Conditional Flags**

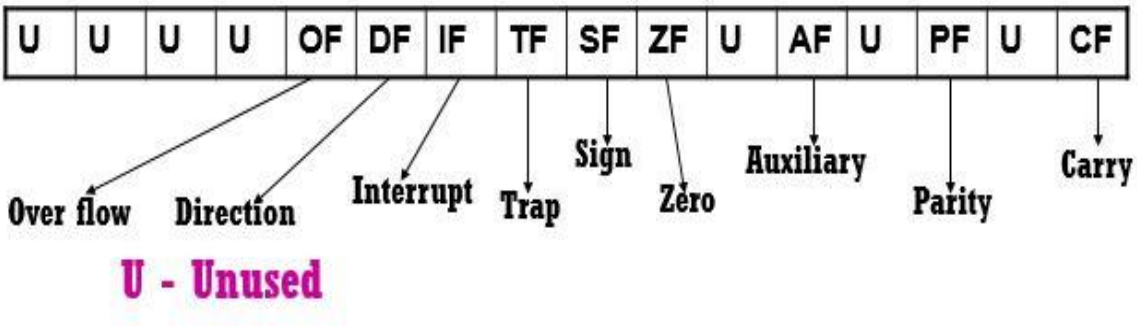
It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags −

- **Carry flag** − This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag** − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** − This flag represents the result when the system capacity is exceeded.

- **Control Flags**

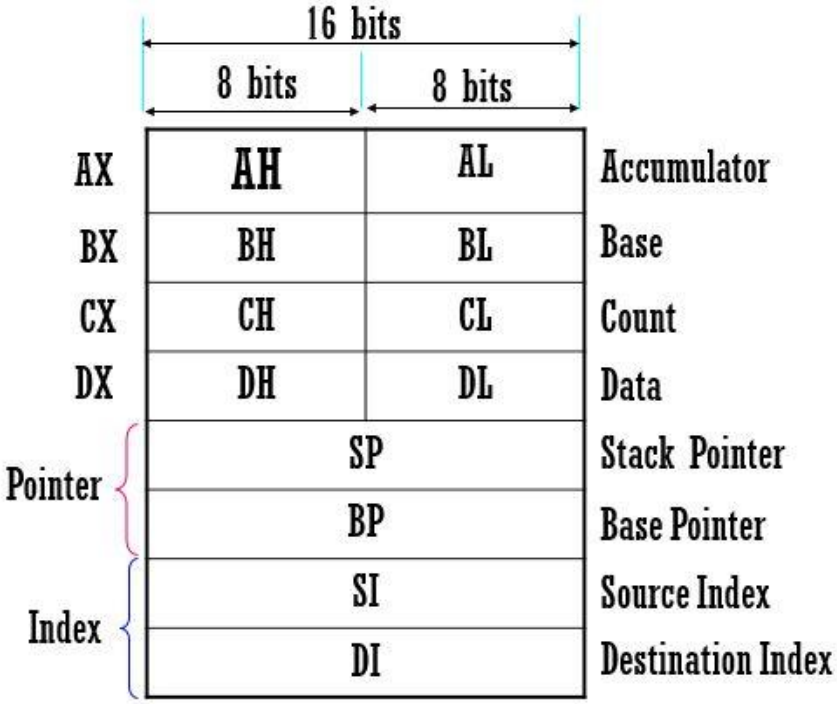
Control flags controls the operations of the execution unit. Following is the list of control flags

- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.



- **General purpose register**

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.



- **AX register −** It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** − It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** − This register is used to hold I/O port address for I/O instruction.

➢ **Stack pointer register**

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

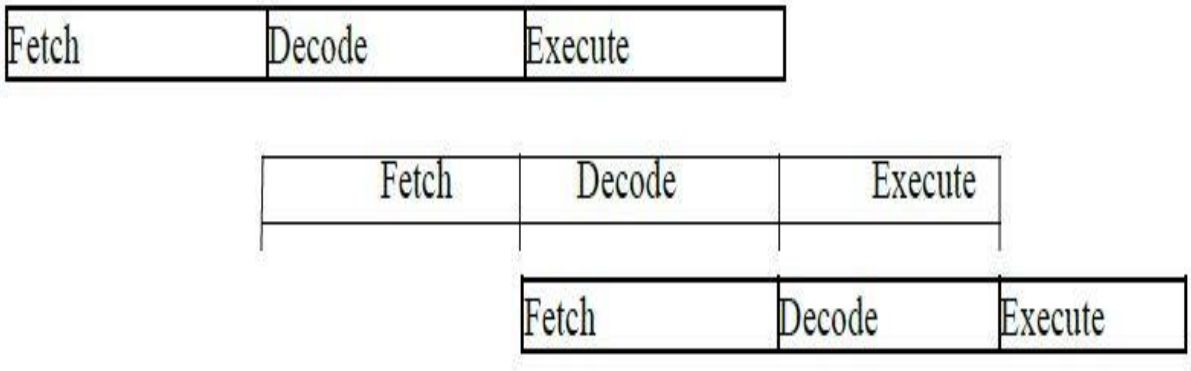
BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts −

➢ **Instruction queue** − BIU contains the instruction queue. BIU gets up to 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.

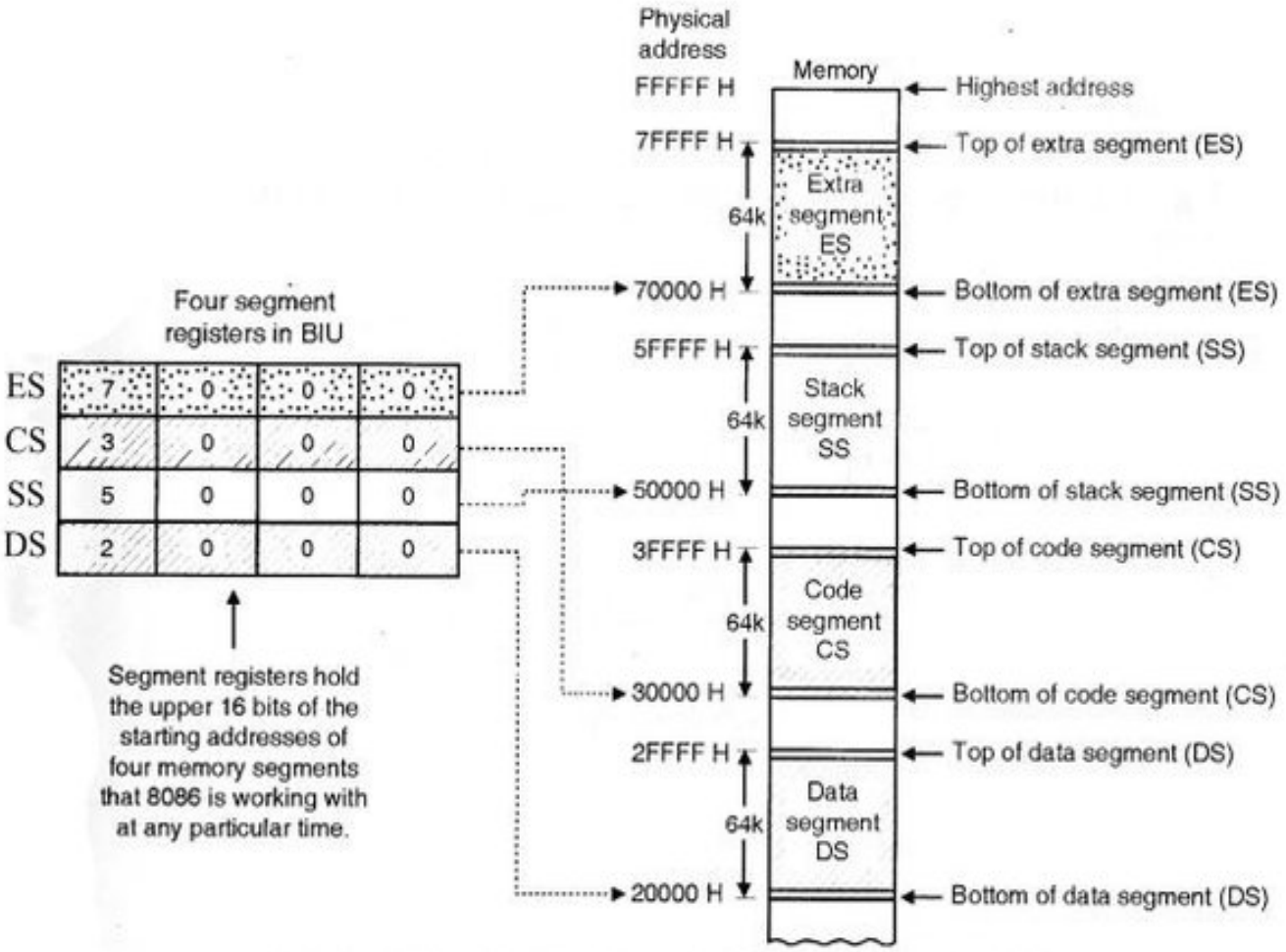
➢ Fetching the next instruction while the current instruction executes is called **pipelining**.



➢ **Segment register** − The total memory size is divided into segments of various sizes. The process of dividing memory this way is called Segmentation. In memory, data is stored as bytes. Each byte has a specific address. Intel 8086 has 20 lines address bus. With 20 address lines, the memory that can be addressed is 220 = 1,048,576 bytes (1 MB).

➢ 8086 can access memory with address ranging from 00000 H to FFFFFH. In 8086, memory has four different types of segments. These are:

- **CS** − It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
- **DS** − It stands for Data Segment. It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address.



One way of positioning four 64k byte segments within the 1M byte memory space of an 8086

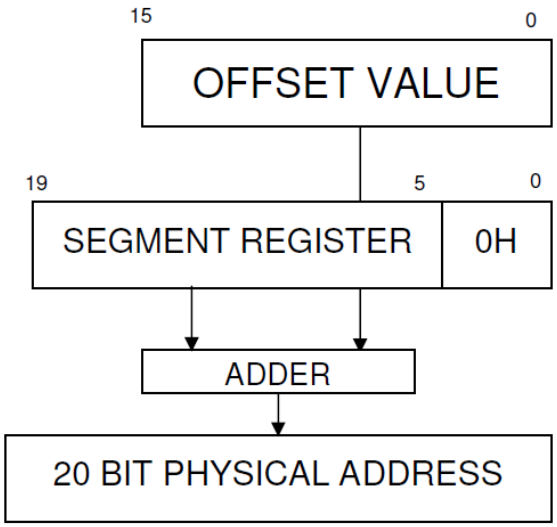
- **SS** − It stands for Stack Segment. It handles memory to store data and addresses during execution.
- **ES** − It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.

➢ **Instruction pointer** − It is a 16-bit register used to hold the address of the next instruction to be executed.

➢ **Pointers and Index Registers**

- The pointers contain offset within the particular segments.
- The pointer register IP (instruction Pointer) contains offset within the code segment. The pointer register BP (base pointer) contains offset within the data segment.
- The pointer register SP (stack pointer) contains offset within the stack segment.
- The register SI (source index) is used to store the offset of source data in data segment.
- The register DI (destination index) is used to store the offset of destination in data or extra segment.
- The index registers are particularly useful for string manipulation.

➢ **Physical Address**



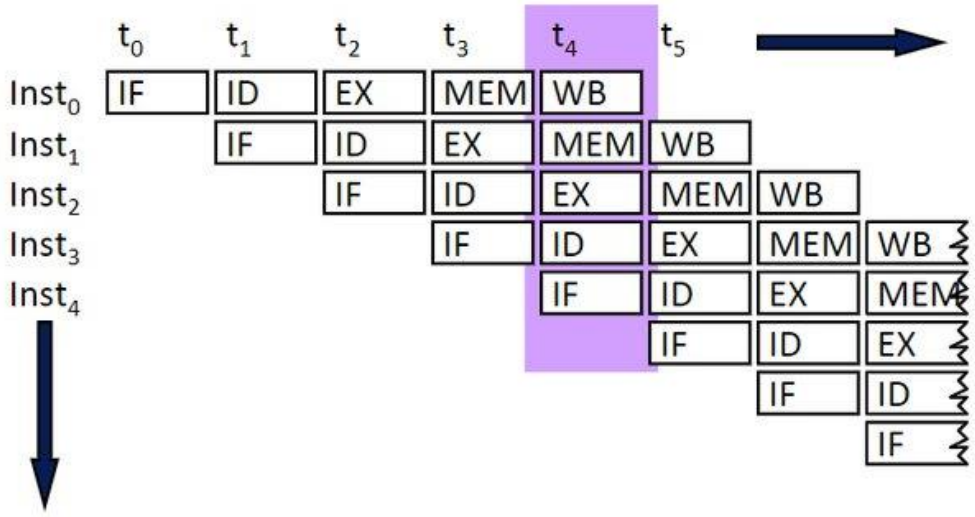
- The physical address is expressed in terms of the logical address.
- In the form of Base Address: OFFSET address
  Ex: 3000H:2000H
- Base address is the address of segment (data, extra, stack, or code).
- **Physical address=Segment address*10H + offset address**
- The equivalent physical address will be 3000*10+2000= 32000H. The address summer of 8086 does all the work to generate this 20-bit physical address.
- As we have only 16 bit registers two registers will be used to express the logical address. The segment registers are used for the base address.

❖ **INSTRUCTION PIPELINING:**

➢ An instruction pipeline is a technique used in the design of computers to increase their instruction throughput (the number of instructions that can be executed in a unit of time). The basic instruction cycle is broken up into a series called a pipeline. Rather than processing each instruction sequentially (one at a time, finishing one instruction before starting the next), each instruction is split up into a sequence of steps so different steps can be executed concurrently (at

the same time) and in parallel (by different circuitry). Each instruction is split into a sequence of dependent steps. The first step is always to fetch the instruction from memory; the final step is usually writing the results of the instruction to processor registers or to memory.

- There are five stages of pipelining:
  1. Instruction fetch
  2. Instruction decode and register fetch
  3. Execute
  4. Memory access
  5. Register write-back



- Above diagram contains following points:
  - All stages will be of equal duration.
  - Each instruction goes through all five stages of pipeline.
  - All the stages will be performed parallel.
  - No memory conflicts.
  - All the accesses occur simultaneously.

❖ FACTORS AFFECTING PIPELINE PERFORMANCE:

- If five stages are not of equal duration, then there will be some waiting time at various stages.
- Conditional branch instruction which can invalidate several instruction fetches.
- Interrupt which is unpredictable event.
- Register and memory conflicts.

****