Introduction to 8085 Microprocessors:

The microprocessor is a semiconductor device (Integrated Circuit) manufactured by the VLSI (Very Large Scale Integration) technique.

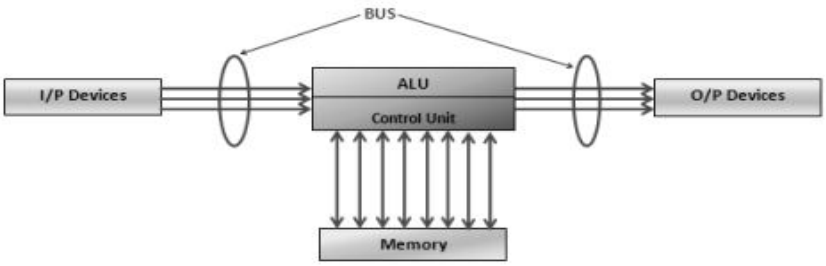
It includes the Arithmetic and Logic Unit, registers and control circuit on a single chip, to perform a function. The memory and other external devices can be connected to the microprocessor. The microprocessor then commands these devices to act according to the program or code written in it.

The microprocessor has a Central Processing Unit, semiconductor memories like EPROM and RAM, input device, output device and interfacing devices. The memories, input device, output device and interfacing devices are called peripherals. The popular input devices are keyboard and floppy disk and the output devices are printer, LED/LCD displays, CRT monitor, etc.

- Advantages of Microprocessor based system
    1. Computational/Processing speed is high.
    2. Systems have become intelligent.
    3. Automation of industrial processes and office administration.
    4. Since the devices are programmable, the system can be altered or changed by changing the software.
    5. Less number of components, compact in size and cost less. Also it is more reliable.
    6. Operation and maintenance are easier.
- Disadvantages of Microprocessor based System
    1. It has limitations on the size of data.
    2. The applications are limited by the physical address space.
    3. The analog signals cannot be processed directly and digitizing the analog signals introduces errors.
    4. The speed of execution is slow and so real time applications are not possible.
    5. Most of the microprocessors do not contain floating point operations.

Features of 8085:

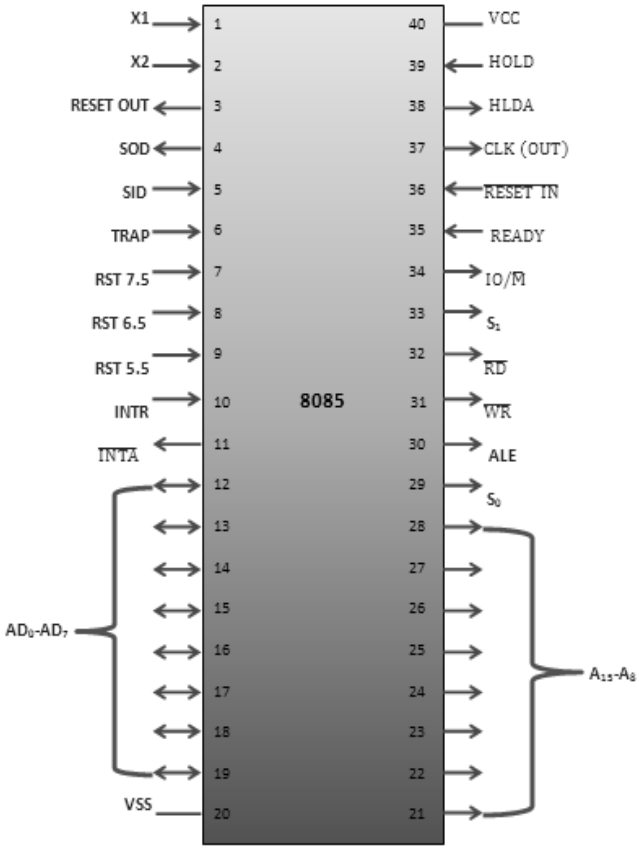
- It is an 8 bit microprocessor that is the the size of data bus is of 8- bits or each character is represented by 8 bits.

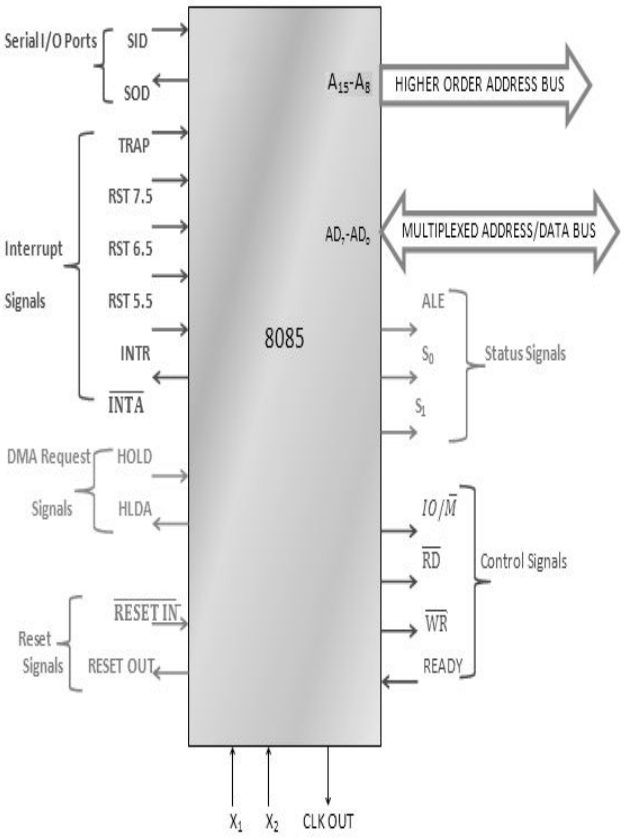


- It has 16-bit address lines ➔ A0-A15 which point to the memory locations so that it can access microprocessor to 2^{16} = 65535 bytes (64KB) memory locations.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD0-AD7. Data bus is a microprocessor is of 8 lines D0-D7.
- It provides 5 level interrupt signals.
- It has 16 bit program counters (PC) register.
- It has 16 bit stack pointer (SP) register.
- It has a accumulator, flag register, six 8-bit general purpose register arranged in pairs: BC, DE, HL and 2special purpose registers.
- It consists of 74 instruction sets & 246 instructions.

- It performs arithmetic and logical operations.
- It provides status for advanced control signals ON-chip clock generator.
- It requires a signal +5V power smicroprocessorply and operates at 3 MHZ single phase clock with maximum clock frequency 6 MHz and minimum clock frequency 500 kHz.
- It has Serial input/output port.
- It has an instruction cycle of 1.3 micro sec.
- It is enclosed with 40 pins DIP (Dual in line package).
- It can be used to implement (interface) 3 chip micro-computers (8085, 8155,8255 and 8355:Peripheral IC's).

Pin Description of 8085:





Functions of Pins of 8085:

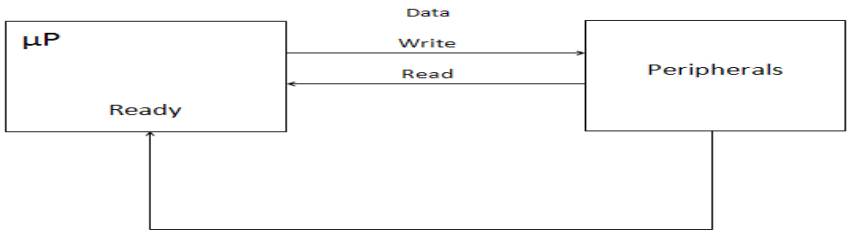
1. A8-A15 Higher Order Address bus:
   - These are signals used as higher order 8 bits of 16 bit address.
   - These signals are unidirectional and are given from 8085 to select memory or I/O devices.
2. AD0-AD7 Multiplexed Address/Data bus:
   - These are signals, having 2 sets of signals. They are address and data.
   - The lower 8 bit of 16 bit address is multiplexed/time shared with data bus.
3. Address latch Enable(ALE):
   - It is an output signal used to give information of AD0-AD7 contents.

- It is a positive going pulse generated when a new operation is started by processor.
- When pulse goes high it indicates that AD0-AD7 are address.
- When it is low it indicates that the contents are data.

4. IO/$\overline{M}$:
   - This is an output status signal used to give information of operation to be performed with memory or I/O devices.
   - When IO/$\overline{M}$=0, the processor is performing memory related operation.
   - When IO/$\overline{M}$=1,the microprocessor is performing I/O device related operation.
   - This signal separates memory and I/O devices.
5. Status signals(S0 and S1):
   - These are output status signals used to give information of operation performed by microprocessor.
   - The S0 and S1 line specifies 4 different conditions of 8085 machine cycles.

Operation	S0	S1
Opcode fetch(instruction read from memory)	1	1
Read(data read from memory)	0	1
Write	1	0
Halt	0	0

6. ($\overline{RD}$)Read:
   - This is an active low output control signal used to read data from memory or an I/O device.
7. ($\overline{WR}$)Write:
   - This is an active low output signal used to write data to memory or an I/O device.
8. Ready:
   - This is an active high input control signal.
   - It is used by microprocessor to detect whether a peripheral has completed or is Ready for the data transfer or not.

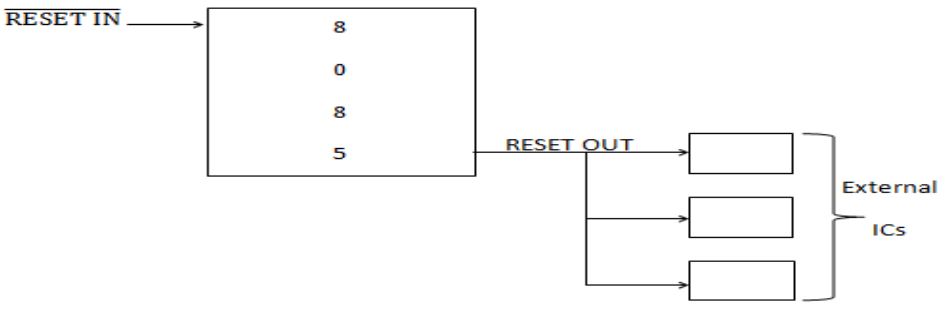


   - The main function of this pin is to synchronize slower peripheral to faster microprocessor.
   - If ready pin is high the microprocessor will complete the operation and proceeds for the next operation.
   - If ready pin is low the microprocessor will wait until it goes high.
9. Trap:
   - This is an active high, level and edge triggered, non-maskable higher priority interrupt in the microprocessor.

- When TRAP is active, the program counter of microprocessor jumps automatically at address 0024.

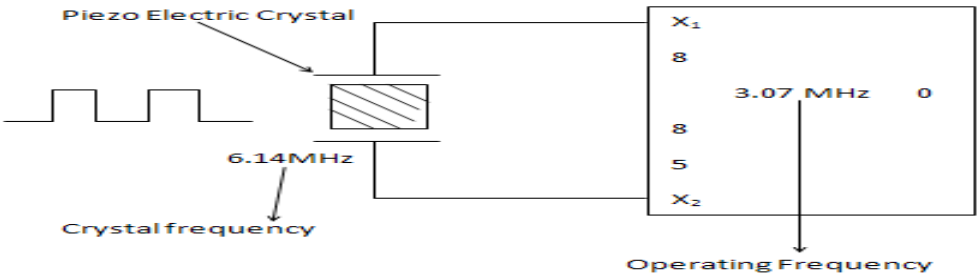
10. RST 7.5,RST 6.5 and RST 5.5:
    - These are active high, edge (RST 7.5) or level (RST 6.5 and RST 5.5) triggered maskable interrupt of the microprocessor.
    - The priorities of these are TRAP, RST 7.5, RST 6.5, and RST 5.5.
    - When RST 7.5, RST 6.5 and RST 5.5 are active, the program counter jumps automatically at address 003C, 0034, 002C respectively.
11. INTR and $\overline{\text{INTA}}$:
    - INTR is an active high, level triggered general purpose interrupt in microprocessor.
    - When INTR is active the µp generates an interrupt. The microprocessor acknowledges by signal $\overline{\text{INTA}}$.
    - If INTR is active, the Program Counter (PC) will be restricted from incrementing and an INTA will be issued.
    - During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt Service routine.
    - The INTR is enabled and disabled by software. It is disabled by Reset And immediately after an interrupt is accepted.
12. HOLD:
    - HOLD indicates that another Master is requesting the use of the Address and Data Buses.
    - The CPU, microprocessor on receiving the Hold request, will withdraw the use of buses as soon as the completion of the current machine cycle. Internal processing can continue.
    - The processor can regain the buses only after the Hold is removed.
13. HLDA:
    - HOLD ACKNOWLEDGE indicates that the CPU has received the Hold request and that it will withdraw the buses in the next clock cycle.
    - HLDA goes low after the Hold Request is removed.
    - The CPU takes the buses one half clock cycles after HLDA goes Low.
14. $\overline{\text{RESET IN}}$:
    - Reset sets the Program Counter to zero and resets the Interrupt
    - The CPU is held in the reset condition as long as Reset is applied.
    - After reset status internal register and flag are unpredictable.
    - After reset microprocessor starts executing from instruction from 0000H onwards.

15. RESET OUT:
    - This is an active high output signal used to indicate CPU is being reset

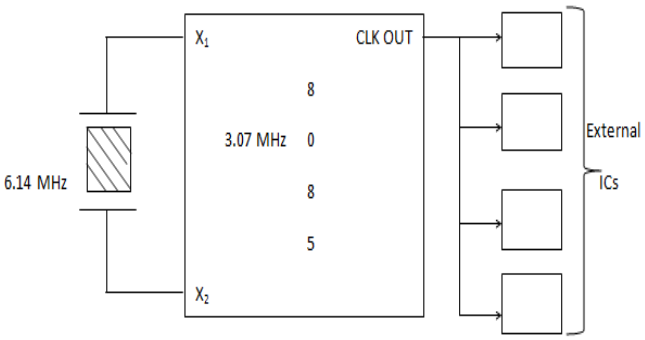


and can be used as a system RESET.
- The signal is synchronized to the processor clock.
- This signal is also used to reset the peripherals once the microprocessor is reset
- It is an acknowledgement signal to RESET IN (bar).

16. Serial input data(SID):
    - This is an active high Serial input data line & the data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
17. Serial output data(SOD):
    - This is an active high Serial output data line.
    - The output SOD is set or reset as specified by the SIM instruction.
18. X1,X2:
    - Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal.
    - The input frequency is divided by 2 to give the internal operating frequency



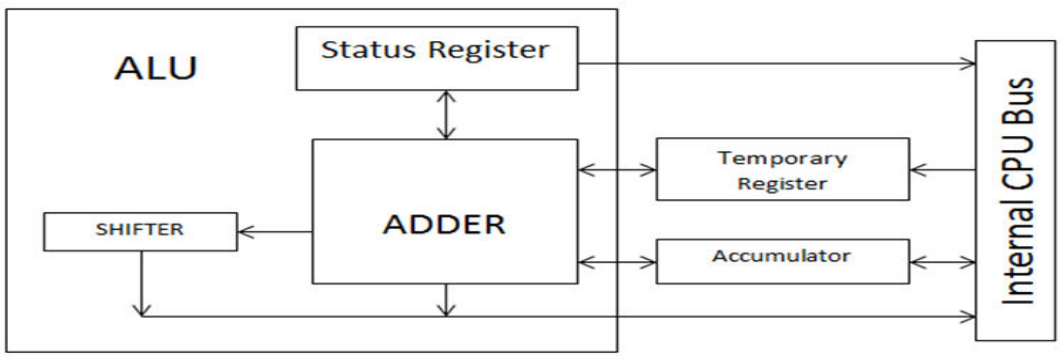
19. CLK OUT:
    - Clock Output for use as a system clock when a crystal or R/ C network is used as an Input to the CPU.
    - Clock input to all other peripherals is provided through CLK OUT pin.
    - The period of CLK is twice the X1, X2 input period.



20. VCC and VSS:
    - +5 volts microprocessor supply and Ground Reference

Architecture of 8085:

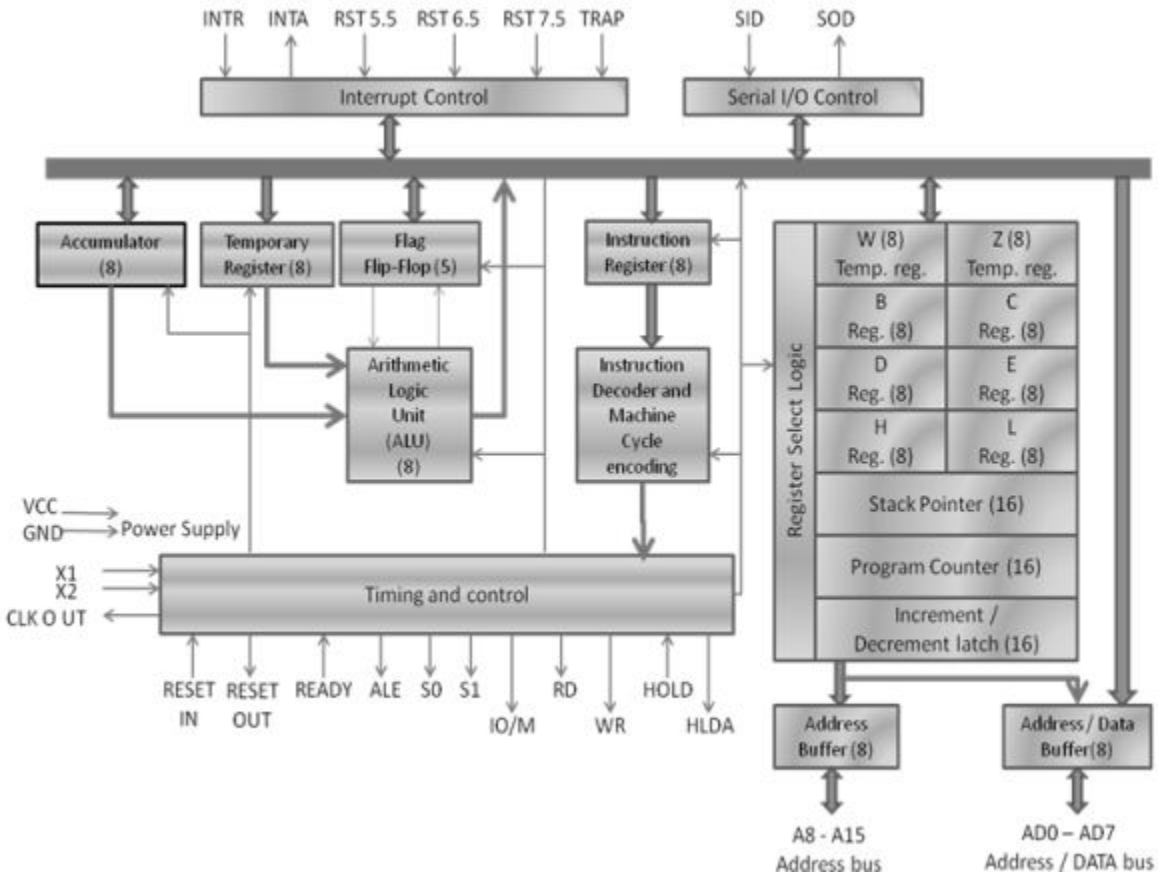
The internal architecture of 8085 includes the ALU, timing and control unit, instruction register and decoder, register array, interrupt control and



serial I/O control. The main heart of microprocessor is CPU. CPU comprises of ALU, timing and control unit, instruction register and decoder, register array, interrupt control and serial I/O control.

I). Arithmetic and logical section (ALU):

It performs arithmetic and logical operations like ANDing, ORing, EX-ORing, ADDITON, SUBTRACTION etc. It is not accessible by user. The word length of ALU depends upon of an internal data bus. It is of 8 bit. It is always controlled by timing and control circuits. It provides status or result of flag register.



- The ALU contains following blocks:
  - i. Adder: It performs arithmetic operations like addition, subtraction, increment, decrement, etc. The result of operation is stored into accumulator.
  - ii. Shifter: It performs logical operations like rotate left, rotate right, etc.The result of operation is again stored into accumulator.
  - iii. Status Register: Also known as flag register. It contains a no. of flags either to indicate conditions arising after last ALU operation or to control certain operations.

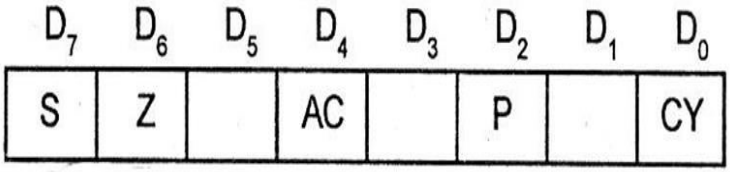
2). Accumulator: It is one of the general purpose register of microprocessor also called as A register. The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The user can access this register by giving instructions.

3). Temporary Register:

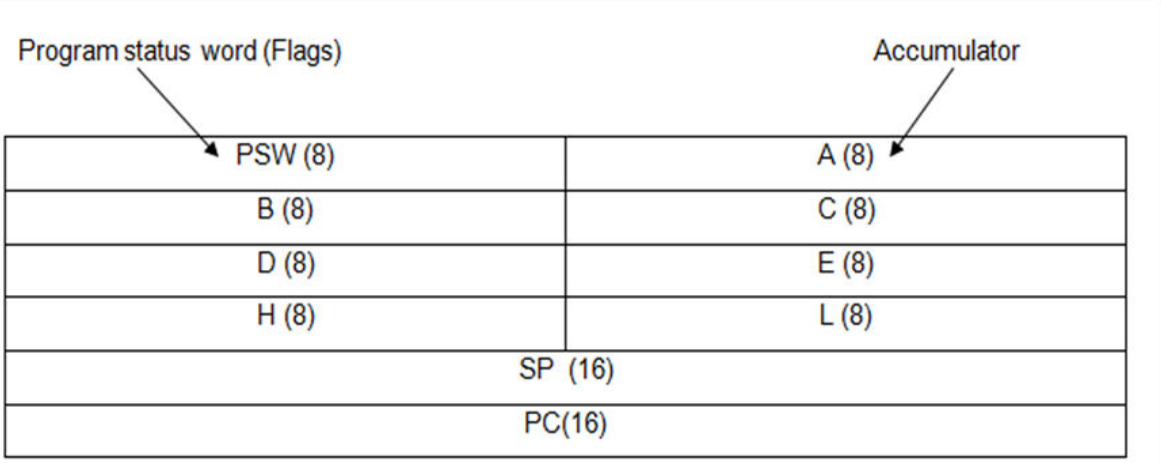
- It is also called as operand register (8 bit).
- It provides operands to ALU.ALU can store immediate result in temporary register.
- It is not accessible by user.

4). Status or flag register:

- Flag register is a group of flip flops used to give status of different operations result.
- The flag register is connected to ALU.
- Once an operation is performed by ALU the result is transferred on internal data bus and status of result will be stored in flip flops.
- They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags.



Carry flag(CY): If an operation performed in ALU generates the carry from D7 to next stage then CY flag is set, else it is reset.
Auxillary carry(AC): If an operation performed in ALU generates the carry from lower nibble (D0 to D3) to upper nibble (D4 to D7) AC flag is set, else it resets.
Zero flag(z): If an operation performed in ALU results 0 value of entire 8-bits then zero flag is set, else it resets.
Sign flag(s): If MSB bit =0 then the number is positive, else it is negative.
Parity flag(p): If the result contains even no. of ones this flag is set and for odd no. of ones this flag is reset.
II). Register section: CPU registers of 8085 are as follows:



a). Temporary register(W and Z): This temporary register can only be accessed by the microprocessor and it is completely not available or inaccessible to programmers. Temporary register is an 8-bit register. This register is used by control systems to hold operand, intermediate operand, and address of memory and I/O devices temporarily.

b). **General purpose register: Other than** accumulator 8085 consists of six special types of registers called General Purpose Registers. These general purpose registers are used to hold data like any other registers. The general purpose registers in 8085 processors are B, C, D, E, H and L. Each register can hold 8-bit data. These registers can also be used to work in pairs to hold 16-bit data. They can work in pairs such as B-C, D-E and H-L to store 16-bit data. The H-L pair works as a memory pointer. A memory pointer holds the address of a particular memory location. They can store 16-bit address as they work in pair.

c). Special purpose register:

1). Program counter: Program counter is a special purpose register. For ex: An instruction is being executed by processor. As soon as the ALU finished executing the instruction, the processor looks for the next instruction to be executed. So, there is a needed for holding the address of the next instruction to be executed in order to save time. This is done by the program counter. A program counter stores the address 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. Program counter is a 16-bit register.

2). **Stack pointer:** Stack pointer is also a 16-bit register which is used as a memory pointer. A stack is nothing but the portion of RAM (Random access memory). Stack pointer maintains the address of the last byte that is entered into stack. Each time when the data is loaded into stack, Stack pointer gets decremented. Oppositely it is incremented when data is retrieved from stack.

3). **Increment/decrement register:** The 8-bit contents of a register or a memory location can be incremented or decremented by 1. This 16-bit register is used to increment or decrement the content of program counter and stack pointer register by 1. Increment or decrement can be performed on any register or a memory location.

4). **Address /data buffer and address buffer:** The contents of the stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external address bus and address-data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to the memory and I/O chips. The address-data buffer is not only connected to the external data bus but also to the internal data bus which consists of 8-bits. The address data buffer can both send and receive data from internal data bus.

III). Interrupt control section: Used to interrupt a process. For ex: A a microprocessor is executing the main program. Now whenever the interrupt signal is enabled or requested the microprocessor shifts the control from main program to process the incoming request and after the completion of request, the control goes back to the main program. For example an Input/output device may send an interrupt signal to notify that the data is ready for input. The microprocessor temporarily stops the execution of main program and transfers control to specific special routine known as "Interrupt Service Routine"(ISR). After ISR control is transferred back to main program.

Interrupt signals present in 8085 are:

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

Of the above four interrupts TRAP is a NON-MASKABLE interrupt control and other three are maskable interrupts.

A non-maskable interrupt is an interrupt which is given the highest priority in the order of interrupts.

- Suppose you want an instruction to be processed immediately, then you can give the instruction as a non-maskable interrupt.
- Further the non-maskable interrupt cannot be disabled by programmer at any point of time.
- Whereas the maskable interrupts can be disabled and enabled using EI and DI instructions.
- Among the maskable interrupts RST 7.5 is given the highest priority above RST 6.5 and least priority is given to INTR.

IV). Serial I/O control section:

- The input and output of serial data can be carried out using 2 instructions in 8085.
- SID-Serial Input Data
- SOD-Serial Output Data
- Two more instructions are used to perform serial-parallel conversion needed for serial I/O devices. They are: SIM(Set Interrupt Mask), RIM(Read Interrupt mask.)

V). Instruction register and decoder:

- Instruction register is 8-bit register just like every other register of microprocessor.
- The instruction may be anything like adding two data's, moving a data, copying a data etc.
- When such an instruction is fetched from memory, it is directed to Instruction register. So the instruction registers are specifically to store the instructions that are fetched from memory.
- There is an Instruction decoder which decodes the information present in the Instruction register for further processing.

VI). Timing and control unit:

- Timing and control unit is a very important unit as it synchronizes the registers and flow of data through various registers and other units.
- This unit consists of an oscillator and controller sequencer which sends control signals needed for internal and external control of data and other units.
- The oscillator generates two-phase clock signals which aids in synchronizing all the registers of 8085 microprocessor.
- Signals that are associated with Timing and control unit are:

Control Signals: READY, $\overline{RD}$, $\overline{WR}$, ALE
Status Signals: S0, S1, IO/$\overline{M}$
DMA Signals: HOLD, HLDA
RESET Signals:$\overline{RESET\ IN}$, RESET OUT

Applications of 8085

The microprocessor has created a significant impact in its various fields. The availability of low cost, low power and small weight, computing capability allows user to use it in different applications. Nowadays, a microprocessor based systems are used in automatic testing product, speed control of motors, traffic light control, light control of furnaces, moving message display etc. Some of the important areas of applications are mentioned below:

1. **Instrumentation:** It has very wide applications in the field of instrumentation. Frequency counters, function generators, frequency synthesizers, frequency divider, digital meter and many other instruments are available, where microprocessors are used as controller. It is also used in medical instrumentation like ECG (electronic cardiogram) etc.

2. **Control Systems:** Microprocessor based controllers are available in home based appliances, such as microwave oven, washing machine etc., where microprocessors are being used to control various parameters like speed, pressure, temperature, mode of operation etc.
3. **Communication :** Microprocessors are being used in a wide range of communication equipments. In telephone industry, these are used in digital telephone sets, telephone exchanges and modem, etc. The uses of microprocessor in television, satellite communication have made teleconferencing possible. Railway reservation and air reservation system also uses this technology.
4. **Office and Home Based Automation:** Microprocessor based microcomputer with software packages has changed the office environment. Microprocessors based systems are being used for word processing, spread sheet operations, storage, attendance counting using fingerprint verification of employee, home automation system etc. The microprocessor has revolutionized the publication technology.
5. **Entertainment:** The use of microprocessor in toys, entertainment equipment and home applications is making them more entertaining and full of features. The use of microprocessors is more widespread and popular.

ADDRESSING MODES OF 8085:

- Every instruction of a program has to operate on a data.
- The method of specifying the data to be operated by the instruction is called Addressing.
- The 8085 has the following 5 different types of addressing.
  1. Immediate Addressing
  2. Direct Addressing
  3. Register Addressing
  4. Register Indirect Addressing
  5. Implied Addressing

1. Immediate Addressing:

- In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.
- EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

2. Direct Addressing:

- In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.
- EX. LDA 1050H - Load the data available in memory location 1050H in to accumulator; SHLD 3000H

3. Register Addressing:

- In register addressing mode, the instruction specifies the name of the register in which the data is available.
- EX. MOV A, B - Move the content of B register to A register; SPHL; ADD C.

4. Register Indirect Addressing:

- In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.
- EX. MOV A, M - The memory data addressed by H L pair is moved to A register. LDAX B.

5. Implied Addressing:

- In implied addressing mode, the instruction itself specifies the data to be operated.
- EX. CMA - Complement the content of accumulator; RAL

Assembly language Programming:

A microprocessor executes instructions given by the user. Instructions should be in a language known to the microprocessor. Microprocessor understands the language of 0's and 1's only. This language is called **Machine Language.**

- For e.g. 01001111. This is a valid machine language instruction of 8085. It copies the contents of one of the internal registers of 8085 to another.

Assembly Language of 8085:

It uses English like words to perform any task needed by the users. These English like words are called Mnemonics. For e.g.

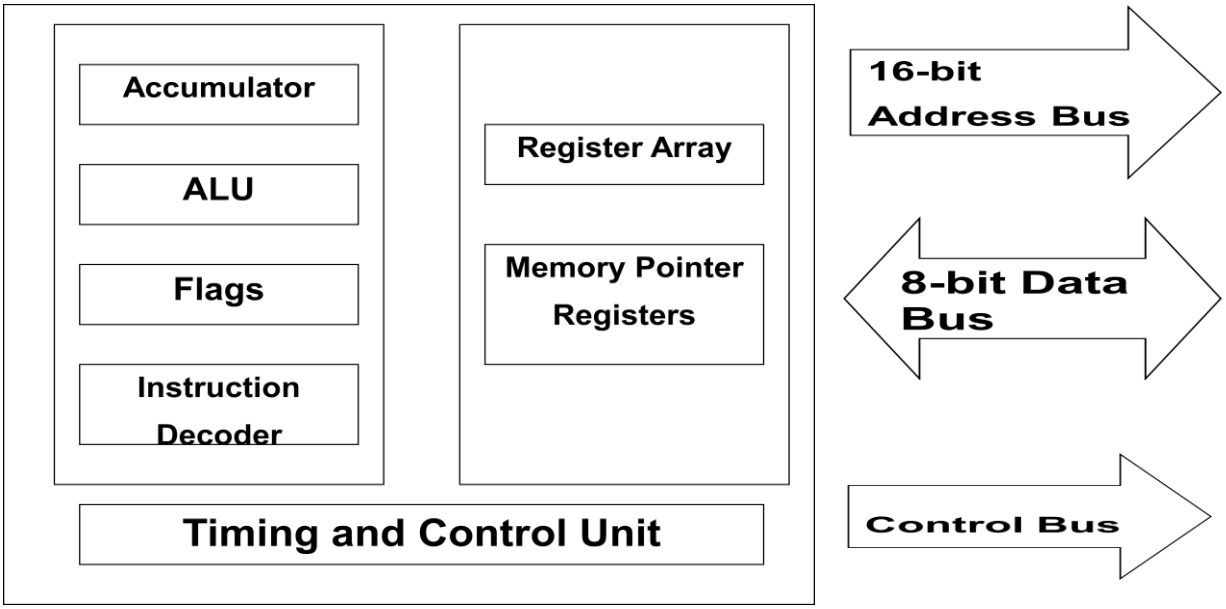
- MOV to indicate data transfer
- ADD to add two values
- SUB to subtract two values

Assembly language program to add two numbers:

MVI A, 2H Copy value 2H in register A
MVI B, 4H Copy value 4H in register B
ADD B A = A + B

Assembly language is specific to a given processor. For e.g. assembly language of intel 8085 is different than that of Motorola 6800 microprocessor. Microprocessor cannot understand a program written in Assembly language. A program known as **Assembler** is used to convert a Assembly language program to machine language. Machine language and Assembly language are both Microprocessor specific (**Machine dependent**), so they are called Low-level languages. **Machine independent** languages are called High-level languages. For e.g. BASIC, PASCAL,C++,C,JAVA, etc.

A software called **Compiler** is required to convert a high-level language program to machine code.



8085 Programming model:

1. Six general-purpose Registers

2. Accumulator Register
3. Flag Register
4. Program Counter Register6c
5. Stack Pointer Register

1.Six general-purpose registers

6. B, C, D, E, H, L
7. Can be combined as register pairs to perform 16-bit operations (BC, DE, HL)

2. **Accumulator – identified by name A**
    1. This register is a part of ALU
    2. 8-bit data storage
    3. Performs arithmetic and logical operations
    4. Result of an operation is stored in accumulator
3. **Flag Register**
    1. This is also a part of ALU
    2. 8085 has five flags named
        1. **Zero** flag (Z)
        2. **Carry** flag (CY)
        3. **Sign** flag (S)
        4. **Parity** flag (P)
        5. **Auxiliary Carry** flag (AC)

    These flags are five flip-flops in flag register. Execution of an arithmetic/logic operation can **set** or **reset** these flags. Condition of flags (set or reset) can be tested through software instructions. 8085 uses these flags in decision-making process.
4. **Program Counter (PC)**
    1. A 16-bit memory pointer register
    2. Used to sequence execution of program instructions
    3. Stores address of a memory location
        1. where next instruction byte is to be fetched by the 8085
    4. when 8085 gets busy to fetch current instruction from memory
        1. PC is incremented by one
        2. PC is now pointing to the address of next instruction
5. **Stack Pointer Register**
    1. a 16-bit memory pointer register
    2. Points to a location in **Stack** memory
    3. Beginning of the stack is defined by loading a 16-bit address in stack pointer register

Basic Operation of 8085:

The basic 5 operations of 8085 are:

1). Fetch: Fetching (bringing) of an instruction.

2). Memory read: Taking a data from the memory i.e write

3). Memory write: Write data into memory.

4). I/O read: Reading data from I/P devices.

5). I/O write: Write data into O/P device.

Instructions of 8085:

Instruction is a command to a processor to perform some task or operations. Each instruction consists of 2 parts.

a). Opcode: It is the first part of an instructin which specifies the task performed by the processor.

b). Operand: It is the second part of an instruction. The operand can be 8-bit data, 8-bit address, 16-bit data, 16-bit address.

The I/O devices have 8-bit address & the memory address is of 16-bits.

Types of Instructions according to word size:

According to word size the 8085 instructions are categorized in 3 groups:

a). 1 byte

b). 2 byte

c). 3 byte

a). 1-byte Instructions: If the opcode & operand take 1-byte of memory space, then it is called as 1-byte instruction.

Ex: MOV A, B ➔ opcode (MOV A) operand (B)

ADD B ➔ opcode (ADD) operand (B)

ADD M ➔ opcode (ADD) operand (M)

b). 2- byte Instructions: If the opcode & operand take 2-byte of memory space, then it is called as 2-byte instruction. First byte specifies the opcode & second byte specifies a operand that can be a data or address.

Ex: MVI B, 78H

Here, MVI B is the opcode. Fetch operation does the reading of the opcode (MVI B) & Memory Read operation does the reading of the operand (78h).

c). 3- byte Instructions: If the opcode & operand take 2-byte of memory space, then it is called as 2-byte instruction. First byte specifies the opcode & the next two (2) bytes specifies a operand that can be a 16-bit data/address.

Based on Operation:

Based on operation there are 5 types of instructions category in 8085:

a). Data Transfer Instruction group

b). Arithmetic group

c). Logical group

d). Branch control group

e). I/O machine control group

DATA TRANSFER INSTRUCTIONS

1). Copy from source to destination.

a). MOV Rd, Rs

b). MOV Rd, M

c). MOV M, Rs

This instruction copies the contents of the source register into the destination register. The contents of the source register are not altered or changed. If one of the operands is a memory location, its location is specified by the contents of the HL registers.

Ex: MOV B, C or MOV B, M

2). Move immediate 8-bit

a). MVI Rd, data

b). MVI M, 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.
Ex: MVI B, 57H or MVI M, 57H
3). Load accumulator
a). 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 of the source are not altered.
Example: LDA 2034H
4). Load accumulator indirect
a). LDAX B/D Reg. pair
The contents of the specified 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
5). Load register pair immediate
a). LXI Reg. pair, 16-bit data
The instruction loads 16-bit data in the register pair specified in the operand.
Example: LXI H, 2034H or LXI H, XYZ
6). Load H and L registers direct
a). LHLD 16-bit address
The instruction copies the contents of the memory location pointed 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
7). Store accumulator direct
a). 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
8). Store accumulator indirect
a). 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
9). Store H and L registers direct
a). 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
10). Exchange H and L with D and E
a). XCHG

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

11). Copy H and L registers to the stack pointer

a). SPHL

The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.
Example: SPHL

12). Exchange H and L with top of stack

a).XTHL

The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1) but the contents of the stack pointer register are not altered.
Example: XTHL

13). Push register pair onto stack

a). PUSH Reg. pair

The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.
Example: PUSH B or PUSH A

14). Pop off stack to register pair

a).POP Reg. pair

The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.
Example: POP H or POP A

15). Output data from accumulator to a port with 8-bit address

a). OUT 8-bit port address

The contents of the accumulator are copied into the I/O port specified by the operand.
Example: OUT F8H

16). Input data to accumulator from a port with 8-bit address

a). IN 8-bit port address

The contents of the input port designated in the operand are read and loaded into the accumulator.
Example: IN 8CH

2). ARITHMETIC INSTRUCTIONS

a). Add register or memory to accumulator

1). ADD R

2). ADD M

The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.
Example: ADD B or ADD M
b). Add register to accumulator with carry
1). ADC R
2). ADC M
The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.
Example: ADC B or ADC M
c). Add immediate to accumulator
1). ADI 8-bit data
The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.
Example: ADI 45H
d). Add immediate to accumulator with carry
1). ACI 8-bit data
The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.
Example: ACI 45H
e). Add register pair to H and L registers
1). DAD Reg. pair
The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.
Example: DAD H
f). Subtract register or memory from accumulator
1). SUB R
2). SUB M
The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
Example: SUB B or SUB M

g). Subtract source and borrow from accumulator
1). SBB R
2). SUB M
The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
Example: SBB B or SBB M
h). Subtract immediate from accumulator
1). SUI 8-bit data

The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.
Example: SUI 45H
i). Subtract immediate from accumulator with borrow
SBI 8-bit data
The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.
Example: SBI 45H
j). Increment register or memory by 1
1). INR R
2). INR M
The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
Example: INR B or INR M
k). Increment register pair by 1
INX R
The contents of the designated register pair are incremented by 1 and the result is stored in the same place.
Example: INX H
l). Decrement register or memory by 1
1). DCR R
2). DCR M
The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
Example: DCR B or DCR M
m). Decrement register pair by 1
DCX R
The contents of the designated register pair are decremented by 1 and the result is stored in the same place.
Example: DCX H
n). Decimal adjust accumulator
DAA
The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. 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

BRANCHING INSTRUCTIONS
1). Jump unconditionally
JMP 16-bit address
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
Example: JMP 2034H or JMP XYZ

2). Jump conditionally Operand: 16-bit address
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described
Example: JZ 2034H or JZ XYZ
JC Jump on Carry CY = 1
JNC Jump on no Carry CY = 0
JP Jump on positive S = 0
JM Jump on minus S = 1
JZ Jump on zero Z = 1
JNZ Jump on no zero Z = 0
JPE Jump on parity even P = 1
JPO Jump on parity odd P = 0
3). Unconditional subroutine call
CALL 16-bit address
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
Example: CALL 2034H or CALL XYZ
4). Call conditionally
Operand: 16-bit address
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
Example: CZ 2034H or CZ XYZ
CC Call on Carry CY = 1
CNC Call on no Carry CY = 0
CP Call on positive S = 0
CM Call on minus S = 1
CZ Call on zero Z = 1
CNZ Call on no zero Z = 0
CPE Call on parity even P = 1
CPO Call on parity odd P = 0
5). Return from subroutine unconditionally
RET
The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RET
6).Return from subroutine conditionally
Operand: none
The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RZ
RC Return on Carry CY = 1

RNC Return on no Carry CY = 0
RP Return on positive S = 0
RM Return on minus S = 1
RZ Return on zero Z = 1
RNZ Return on no zero Z = 0
RPE Return on parity even P = 1
RPO Return on parity odd P = 0
7). Load program counter with HL contents
PCHL none
The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
Example: PCHL
8). Restart
RST 0-7
The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. But these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:
RST 0 0000H
RST 1 0008H
RST 2 0010H
RST 3 0018H
RST 4 0020H
RST 5 0028H
RST 6 0030H
RST 7 0038H
The 8085 has four additional interrupts and these interrupts generate RST instructions internally and so do not require any external hardware. These instructions and their Restart addresses are:
TRAP 0024H
RST 5.5 002CH
RST 6.5 0034H
RST 7.5 003CH

4). LOGICAL INSTRUCTIONS

1). Compare register or memory with accumulator
a). CMP R
b). CMP M
The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < (reg/mem): carry flag is set
if (A) = (reg/mem): zero flag is set
if (A) > (reg/mem): carry and zero flags are reset
Example: CMP B or CMP M
2). Compare immediate with accumulator
CPI 8-bit data

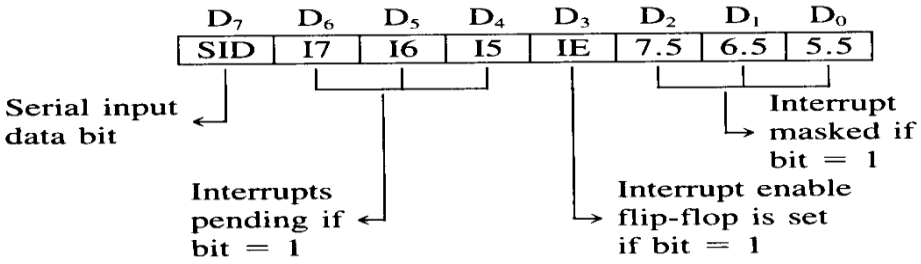
The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < data: carry flag is set
if (A) = data: zero flag is set
if (A) > data: carry and zero flags are reset
Example: CPI 89H
3). Logical AND register or memory with accumulator
a). ANA R
b). ANA M
The contents of the accumulator are logically ANDed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANA B or ANA M
4). Logical AND immediate with accumulator
ANI 8-bit data
The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANI 86H
5). Exclusive OR register or memory with accumulator
a). XRA R
b). XRA M
The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
Example: XRA B or XRA M
6). Exclusive OR immediate with accumulator
XRI 8-bit data
The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
Example: XRI 86H
7). Logical OR register or memory with accumulator
a). ORA R
b). ORA M
The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
Example: ORA B or ORA M
8). Logical OR immediate with accumulator
ORI 8-bit data
The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
Example: ORI 86H
9). Rotate accumulator left

RLC
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. S, Z, P, AC are not affected.
Example: RLC
10). Rotate accumulator right
RRC
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. S, Z, P, AC are not affected.
Example: RRC
11). Rotate accumulator left through carry
RAL
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. S, Z, P, AC are not affected.
Example: RAL
12). Rotate accumulator right through carry
RAR
Each binary bit of the accumulator is rotated right by one position 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. S, Z, P, AC are not affected.
Example: RAR
13). Complement accumulator
CMA
The contents of the accumulator are complemented. No flags are affected.
Example: CMA
14). Complement carry
CMC
The Carry flag is complemented. No other flags are affected.
Example: CMC
15). Set Carry
STC
The Carry flag is set to 1. No other flags are affected.
Example: STC

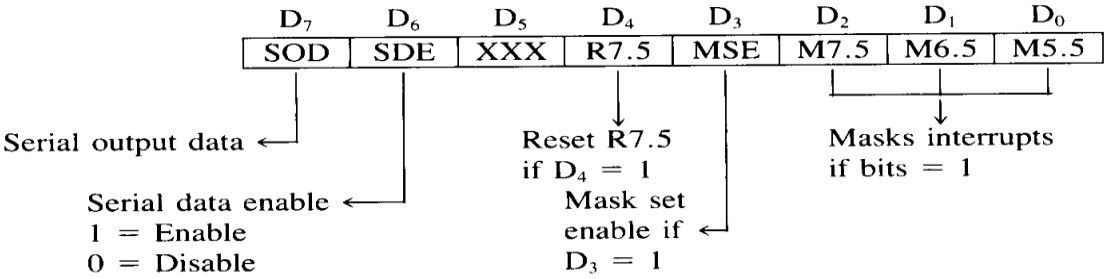
5). CONTROL INSTRUCTIONS

1). No operation
NOP
No operation is performed. The instruction is fetched and decoded. However no operation is executed.
Example: NOP
2). Halt and enter wait state
HLT
The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.
Example: HLT
3). Disable interrupts

DI
The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
Example: DI
4). Enable interrupts
EI
The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip flop is reset, so disabling the interrupts. This instruction is necessary to re-enable the interrupts (except TRAP).
Example: EI
5). Read interrupt mask
RIM
This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator as follows:
Example: RIM



6). Set interrupt mask
SIM
This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.
Example: SIM



- SOD — Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE — Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX — Don't Care
- R7.5 — Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE — Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5 — D_2 = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5 — D_1 = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5 — D_0 = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.