# UNIT- 2

## Pin diagram and internal architecture of 8085 microprocessor:-

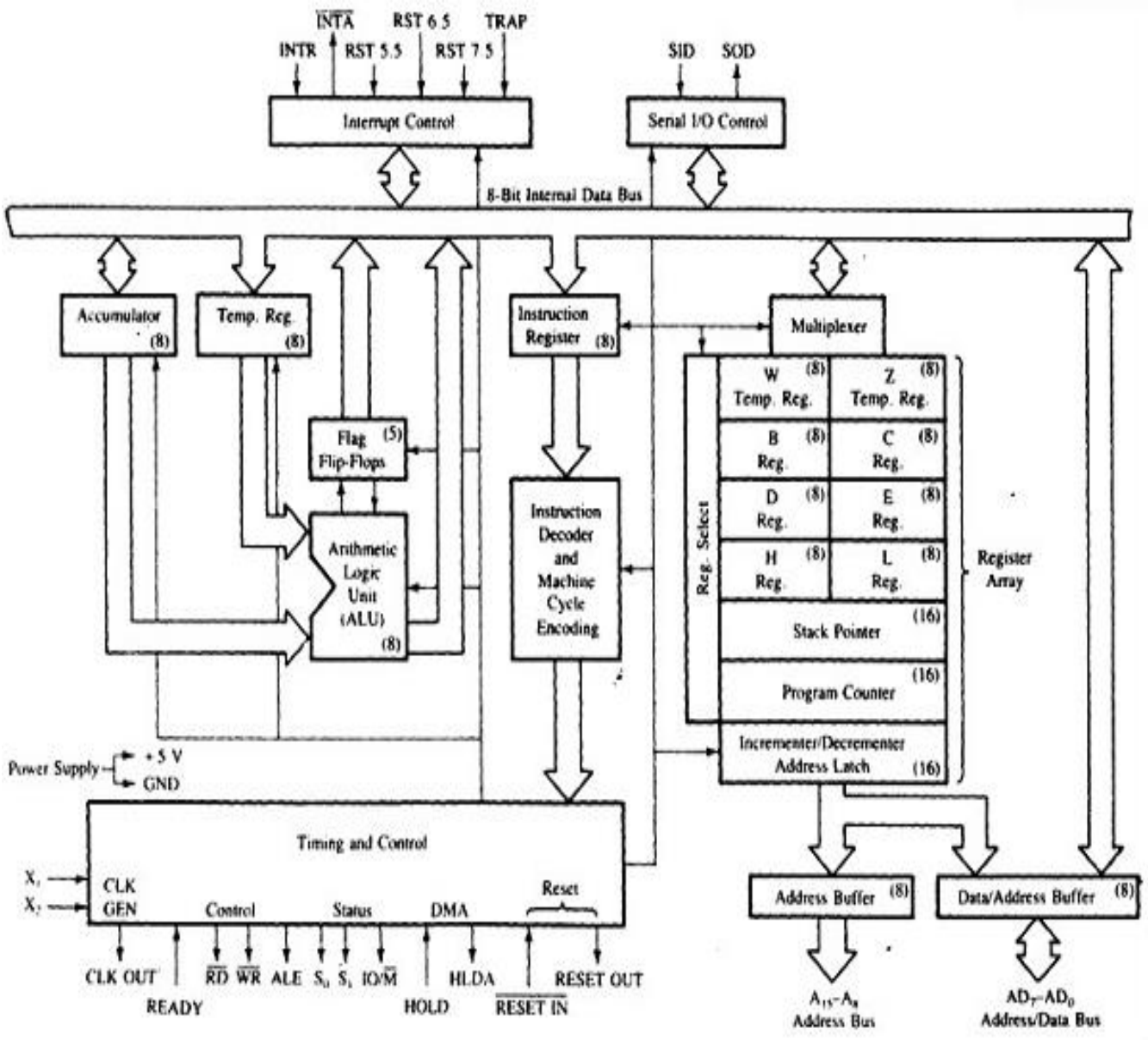



**Fig 1- 8085 Internal Architecture**

8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration −

1. 8-bit data bus
2. 16-bit address bus, which can address upto 64KB
3. A 16-bit program counter
4. A 16-bit stack pointer
5. Six 8-bit registers arranged in pairs: BC, DE, HL
6. Requires +5V supply to operate at 3.2 MHZ single phase clock

It is used in washing machines, microwave ovens, mobile phones, etc.

## 8085 Microprocessor – Functional Units

**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, and 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)

Its bit position is shown in the following table −

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| S | Z | | AC | | P | | CY |

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, 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.

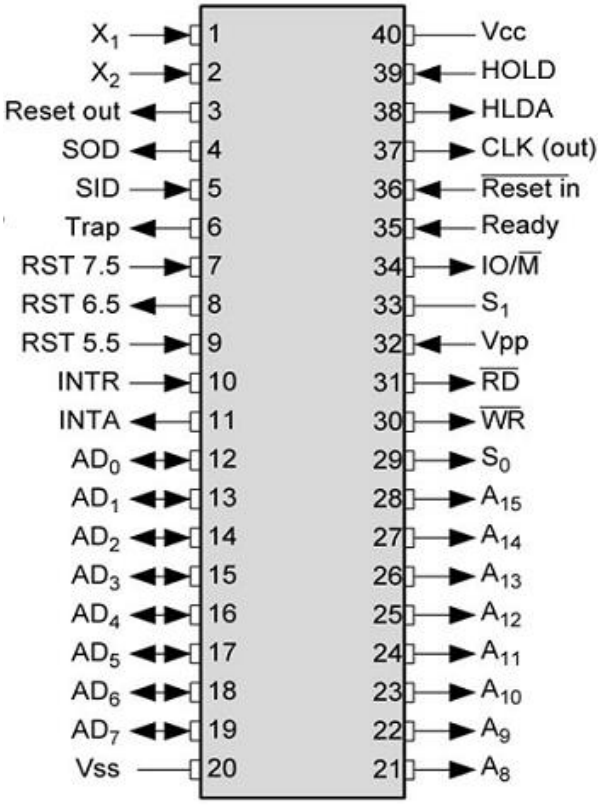



**Fig 2-PIN diagram 8085**

The pins of a 8085 microprocessor can be classified into seven groups −

## 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 & ALE.

- **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.

- **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.

Three status signals are IO/M, S0 & S1.

## 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.

## S1 & S0

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

## Power supply

There are 2 power supply signals − VCC & VSS. 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.

- **INTA** − It is an interrupt acknowledgment signal.
- **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.

# Registers

A **microprocessor** is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provide results as output. A 8085 microprocessor, is a second generation 8-bit microprocessor and is the base for studying and using all the microprocessor available in the market.

## Registers in 8085:

**(a) General Purpose Registers –** The 8085 has six general-purpose registers to store 8-bit data; these are identified as- B, C, D, E, H, and L. These can be combined as register pairs – BC, DE, and HL, to perform some 16-bit operation. These registers are used to store or copy temporary data, by using instructions, during the execution of the program.

### (b) Specific Purpose Registers –

- **Accumulator:**
  The accumulator is an 8-bit register (can store 8-bit data) that is the part of the arithmetic and logical unit (ALU). After performing arithmetical or logical operations, the result is stored in accumulator. Accumulator is also defined as register A.

- **Flag registers:**
  The flag register is a special purpose register and it is completely different from other registers in microprocessor. It consists of 8 bits and only 5 of them are useful. The other three are left vacant and are used in the future Intel versions.These 5 flags are set or reset (when value of flag is 1, then it is said to be set and when value is 0, then it is said to be reset) after an operation according to data condition of the result in the accumulator and other registers. The 5 flag registers are:

1. **Sign Flag:** It occupies the seventh bit of the flag register, which is also known as the most significant bit. It helps the programmer to know whether the number stored in the accumulator is positive or negative. If the sign flag is set, it means that number stored in the accumulator is negative, and if reset, then the number is positive.

2. **Zero Flag:**: It occupies the sixth bit of the flag register. It is set, when the operation performed in the ALU results in zero(all 8 bits are zero), otherwise it is reset. It helps in determining if two numbers are equal or not.

3. **Auxiliary Carry Flag:** It occupies the fourth bit of the flag register. In an arithmetic operation, when a carry flag is generated by the third bit and passed on to the fourth bit, then Auxiliary Carry flag is set. If not flag is reset. This flag is used internally for BCD(Binary-Coded decimal Number) operations.
   **Note –** This is the only flag register in 8085 which is not accessible by user.

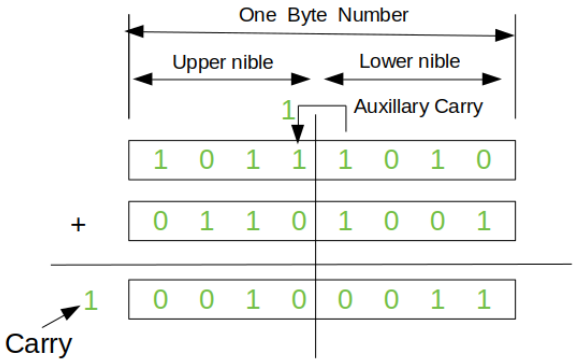
4. **Parity Flag:** It occupies the second bit of the flag register. This flag tests for number of 1's in the accumulator. If the accumulator holds even number of 1's, then this flag is set and it is said to even parity. On the other hand if the number of 1's is odd, then it is reset and it is said to be odd parity.

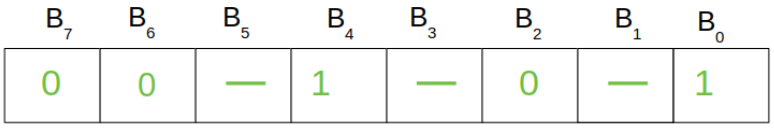
5. **Carry Flag:** It occupies the zeroth bit of the flag register. If the arithmetic operation results in a carry(if result is more than 8 bit), then Carry Flag is set; otherwise it is reset.

## (c) Memory Registers –

There are two 16-bit registers used to hold memory addresses. The size of these registers is 16 bits because the memory addresses are 16 bits. They are :-

- **Program Counter:** This register is used to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.
- **Stack Pointer:** It is used as a memory pointer. It points to a memory location in read/write memory, called the stack. It is always incremented/decremented by 2 during push and pop operation.
  **Example –**
  Here two binary numbers are added. The result produced is stored in the accumulator. Now lets check what each bit means. Refer to the below explanation simultaneously to connect them with the example.




Value of flags

- **Sign Flag (7th bit):** It is reset(0), which means number stored in the accumulator is positive.
- **Zero Flag (6th bit):** It is reset(0), thus result of the operations performed in the ALU is non-zero.
- **Auxiliary Carry Flag (4th bit):** We can see that b3 generates a carry which is taken by b4, thus auxiliary carry flag gets set (1).
- **Parity Flag (2nd bit):** It is reset(0), it means that parity is odd. The accumulator holds odd number of 1's.
- **Carry Flag (0th bit):** It is set(1), output results in more than 8

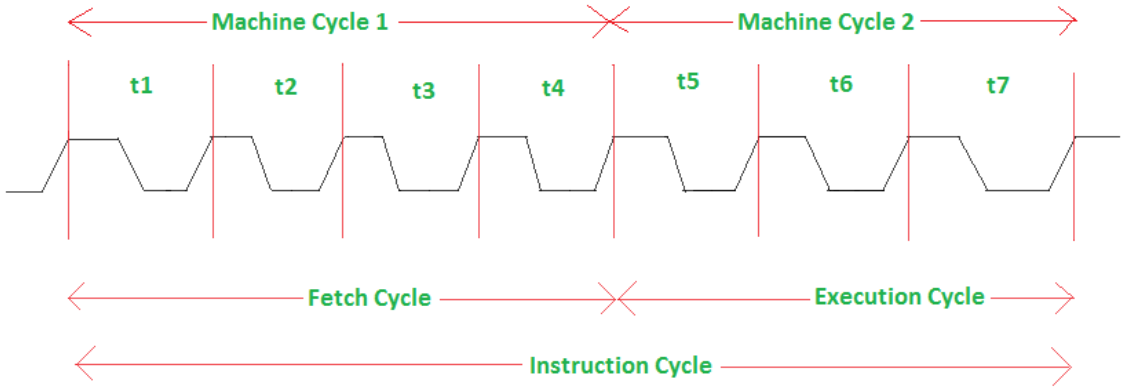
## Interrupt and machine cycle

Time required to execute and fetch an entire instruction is called *instruction cycle*. It consists:

- **Fetch cycle –** The next instruction is fetched by the address stored in program counter (PC) and then stored in the instruction register.
- **Decode instruction –** Decoder interprets the encoded instruction from instruction register.
- **Reading effective address –** The address given in instruction is read from main memory and required data is fetched. The effective address depends on direct addressing mode or indirect addressing mode.
- **Execution cycle –** consists memory read (MR), memory write (MW), input output read (IOR) and input output write (IOW)

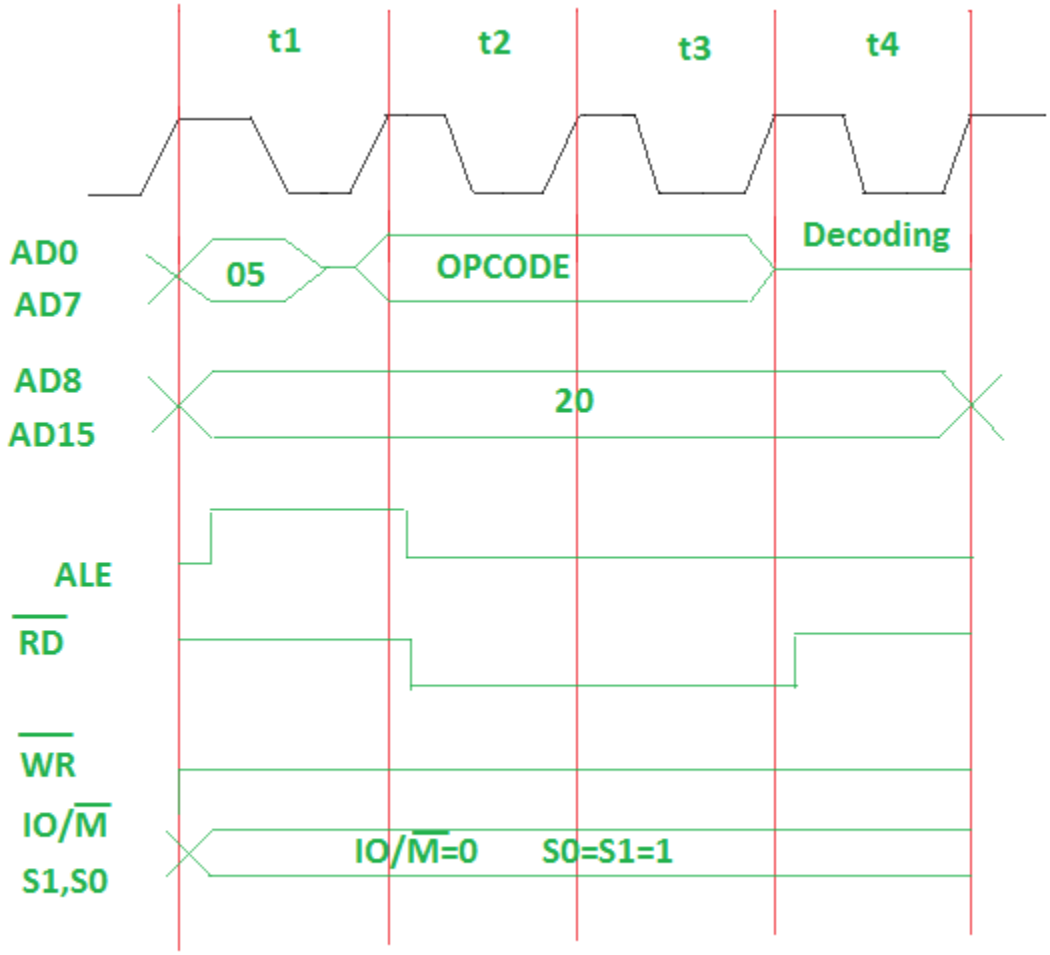
The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called *machine cycle*. One time period of frequency of microprocessor is called *t-state*. A t-state is measured from the falling edge of one

clock pulse to the falling edge of the next clock pulse.
Fetch cycle takes four t-states and execution cycle takes three t-states.




**Instruction cycle in 8085 microprocessor**

Timing diagram for fetch cycle or opcode fetch:




**Timing diagram for opcode fetch**

Above diagram represents:

- **05 –** lower bit of address where opcode is stored. Multiplexed address and data bus AD0-AD7 are used.
- **20 –** higher bit of address where opcode is stored. Multiplexed address and data bus AD8-AD15 are used.
- **ALE –** Provides signal for multiplexed address and data bus. If signal is high or 1, multiplexed address and data bus will be used as address bus. To fetch lower bit of address, signal is 1 so that multiplexed bus can act as address bus. If signal is low or 0, multiplexed bus will be used as data bus. When lower bit of address is fetched then it will act as data bus as the signal is low.
- **RD (low active) –** If signal is high or 1, no data is read by microprocessor. If signal is low or 0, data is read by microprocessor.
- **WR (low active) –** If signal is high or 1, no data is written by microprocessor. If signal is low or 0, data is written by microprocessor.
- **IO/M (low active) and S1, S0 –** If signal is high or 1, operation is performing on input output. If signal is low or 0, operation is performing on memory.

| Machine Cycle | Status | | | Control Signals | | |
|---|---|---|---|---|---|---|
| | $\overline{IO/M}$ | S1 | S0 | $\overline{RD}$ | $\overline{WR}$ | $\overline{INTA}$ |
| Opcode Fetch | 0 | 1 | 1 | 0 | 1 | 1 |
| Memory Read | 0 | 1 | 0 | 0 | 1 | 1 |
| Memory Write | 0 | 0 | 1 | 1 | 0 | 1 |
| I/0 Read | 1 | 1 | 0 | 0 | 1 | 1 |
| I/O Write | 1 | 0 | 1 | 1 | 0 | 1 |
| Interrupt Acknowledge | 1 | 1 | 1 | 1 | 1 | 0 |
| HALT | Z | 0 | 0 | Z | Z | 1 |
| HOLD | Z | X | X | Z | Z | 1 |
| RESET | Z | X | X | Z | Z | 1 |

Where Z is tri state (pin neither connected to supply nor ground. High impedance) and X represents do not care.

**8085 machine cycle status and control signals**

# Addressing Modes in 8085

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups −

## Immediate addressing mode

In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. **For example:** MVI K, 20F: means 20F is copied into register K.

## Register addressing mode

In this mode, the data is copied from one register to another. **For example:** MOV K, B: means data in register B is copied to register K.

## Direct addressing mode

In this mode, the data is directly copied from the given address to the register. **For example:** LDB 5000K: means the data at address 5000K is copied to register B.

### Indirect addressing mode

In this mode, the data is transferred from one register to another by using the address pointed by the register. **For example:** MOV K, B: means data is transferred from the memory address pointed by the register to the register K.

### Implied addressing mode

This mode doesn't require any operand; the data is specified by the opcode itself. **For example:** CMP.

## Instruction Set Classification of 8085 Processor

These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

### Instruction Set Classification

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

## 1 Data Transfer Croup

The data transfer instructions move data between registers or between memory and registers.

| | |
|---|---|
| MOV | Move |
| MVI | Move Immediate |
| LDA | Load Accumulator Directly from Memory |

| | |
|---|---|
| STA | Store Accumulator Directly in Memory |
| LHLD | Load H & L Registers Directly from Memory |
| SHLD | Store H & L Registers Directly in Memory |

An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

| | |
|---|---|
| LXI | Load Register Pair with Immediate data |
| LDAX | Load Accumulator from Address in Register Pair |
| STAX | Store Accumulator in Address in Register Pair |
| XCHG | Exchange H & L with D & E |
| XTHL | Exchange Top of Stack with H & L |

## 2 Arithmetic Group

The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

| | |
|---|---|
| ADD | Add to Accumulator |
| ADI | Add Immediate Data to Accumulator |
| ADC | Add to Accumulator Using Carry Flag |
| ACI | Add immediate data to Accumulator Using Carry |
| SUB | Subtract from Accumulator |
| SUI | Subtract Immediate Data from Accumulator |
| SBB | Subtract from Accumulator Using Borrow (Carry) Flag |
| SBI | Subtract Immediate from Accumulator Using Borrow (Carry) Flag |
| INR | Increment Specified Byte by One |
| DCR | Decrement Specified Byte by One |
| INX | Increment Register Pair by One |
| DCX | Decrement Register Pair by One |
| DAD | Double Register Add; Add Content of Register |

Pair to H & L Register Pair

## 3 Logical Group

This group performs logical (Boolean) operations on data in registers and memory and on condition flags. The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in

the accumulator ON or OFF.

| | |
|---|---|
| ANA | Logical AND with Accumulator |
| ANI | Logical AND with Accumulator Using Immediate Data |
| ORA | Logical OR with Accumulator |
| OR | Logical OR with Accumulator Using Immediate Data |
| XRA | Exclusive Logical OR with Accumulator |
| XRI | Exclusive OR Using Immediate Data |

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

| | |
|---|---|
| CMP | Compare |

CPI Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC Rotate Accumulator Left

RRC Rotate Accumulator Right

RAL Rotate Left Through Carry

RAR Rotate Right Through Carry

Complement and carry flag instructions:

CMA Complement Accumulator

CMC Complement Carry Flag

STC Set Carry Flag

## 4 Branch Group

The branching instructions alter normal sequential program flow, either unconditionally or

conditionally. The unconditional branching instructions are as follows:

| | |
|---|---|
| JMP | Jump |
| CALL | Call |
| RET | Return |

Conditional branching instructions examine the status of one of four condition flags to determine

whether the specified branch is to be executed. The conditions that may be specified are as follows:

| | |
|---|---|
| NZ | Not Zero (Z = 0) |
| Z | Zero (Z = 1) |
| NC | No Carry (C = 0) |
| C | Carry (C = 1) |
| PO | Parity Odd (P = 0) |
| PE | Parity Even (P = 1) |
| P | Plus (S = 0) |

| | |
|---|---|
| M | Minus (S = 1) |

Thus, the conditional branching instructions are specified as follows:

| Jumps | Calls | Returns | |
|---|---|---|---|
| INC | CNC | RNC | (No Carry) |
| JNZ | CNZ | RNZ | (Not Zero) |
| JM | CM | RM | (Minus) |
| JP0 | CPO | RPO | (Parity Odd) |
| JM | CM | RM | (Minus) |
| JPE | CPE | RPE | (Parity Even) |
| JP0 | CPO | RPO | (Parity Odd) |

Two other instructions can affect a branch by replacing the contents or the program counter:

| | |
|---|---|
| PCHL | Move H & L to Program Counter |
| RST | Special Restart Instruction Used with Interrupts |

## 5 Stack Instructions

The following instructions affect the Stack and/or Stack Pointer

| | |
|---|---|
| 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 |

## 6 I/0 instructions

| | |
|---|---|
| IN | Initiate Input Operation |
| OUT | Initiate Output Operation |

## 7 Machine Control instructions

| | |
|---|---|
| EI | Enable Interrupt System |
| DI | Disable Interrupt System |
| HLT | Halt |
| NOP | No Operation |

### Assembler Directives of 8085

The assembler directives given below are used by 8085 and 8086 assemblers:

**DB:** *Define Byte*
This directive is used for the purpose of allocating and initializing single or multiple data bytes.

```
AREA DB 30H, 52H, 35H
```

Memory name AREA has three consecutive locations where 30H, 52H and 35H are to be stored.


**DW:** *Define Word*
It is used for initialising single or multiple data words (16-bit).

```
MARK DW 1020H, 4216H
```

These two 16-bit data 1020H and 4216H are stored at 4 consecutive locations in the memory MARK.


**END:** *End of program*
This directive is used at the time of program termination.

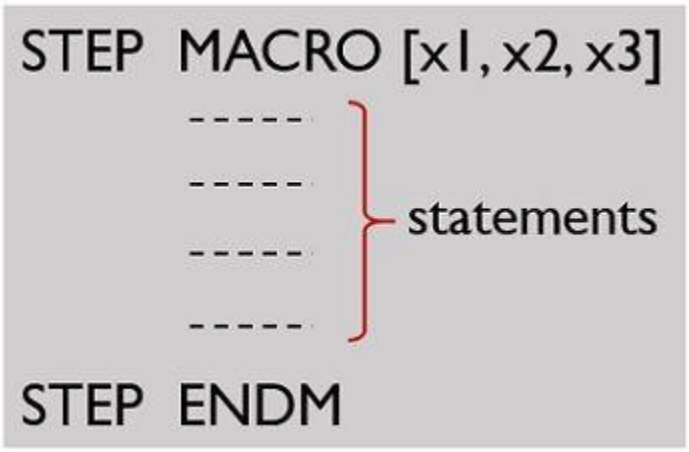
**EQU:** *Equate*
It is used to assign any numerical value or constant to the variable.

```
DONE EQU 10H
```

Variable name ‘DONE’ has value 10H

**MACRO:** *Represents beginning*
Shows the beginning of macro along with defining name and parameters.

**ENDM:** *End of macro*
ENDM indicates the termination of macro.




where macroname (STEP) is specified by the user.

**ORG:** *Origin*
This directive is used at the time of assigning starting address for a module or segment.

```
ORG 1050H
```

By this instruction, the assembler gets to know that the statements following this instruction, must be stored in the memory location beginning with address 1050H.

The 8085 instruction set is classified into 3 categories by considering the length of the instructions. In 8085, the length is measured in terms of “byte” rather then “word” because 8085 microprocessor has 8-bit data bus. Three types of instruction are: 1-byte instruction, 2-byte instruction, and 3-byte instruction.

## 1. One-byte instructions –

In 1-byte instruction, the opcode and the operand of an instruction are represented in one byte.

- **Example-1:**
  Task- Copy the contents of accumulator in register B.
- `Mnemonic- MOV B, A`

- Opcode- MOV
- Operand- B, A
- Hex Code- 47H
  Binary code- 0100 0111
- **Example-2:**
  Task- Add the contents of accumulator to the contents of register B.
- **Mnemonic- ADD B**
- Opcode- ADD
- Operand- B
- Hex Code- 80H
  Binary code- 1000 0000
- **Example-3:**
  Task- Invert (complement) each bit in the accumulator.

**Mnemonic- CMA**
Opcode- CMA
Operand- NA
Hex Code- 2FH
Binary code- 0010 1111

**Note –** The length of these instructions is 8-bit; each requires one memory location. The mnemonic is always followed by a letter (or two letters) representing the registers (such as A, B, C, D, E, H, L and SP).

## 2. Two-byte instructions –

Two-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next 8 bits indicates the operand.

- **Example-1:**
  Task- Load the hexadecimal data 32H in the accumulator.
- **Mnemonic- MVI A, 32H**
- Opcode- MVI
- Operand- A, 32H
- Hex Code- 3E
- 32
- Binary code- 0011 1110
  0011 0010
- **Example-2:**
  Task- Load the hexadecimal data F2H in the register B.
- **Mnemonic- MVI B, F2H**
- Opcode- MVI
- Operand- B, F2H
- Hex Code- 06
- F2
- Binary code- 0000 0110

1111 0010

**Note –** This type of instructions need two bytes to store the binary codes. The mnemonic is always followed by 8-bit (byte) data.

## 3. Three-byte instructions –

Three-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next two bytes specify the 16-bit address. The low-order address is represented in second byte and the high-order address is represented in the third byte.

- **Example-1:**
  Task- Load contents of memory 2050H in the accumulator.
- **Mnemonic- LDA 2050H**
- Opcode- LDA
- Operand- 2050H
- Hex Code- 3A
- 50
- 20
- Binary code- 0011 1010
- 0101 0000
  0010 0000
- **Example-2:**
  Task- Transfer the program sequence to the memory location 2050H.
- **Mnemonic- JMP 2085H**
- Opcode- JMP
- Operand- 2085H
- Hex Code- C3
- 85
- 20
- Binary code- 1100 0011
- 1000 0101
  0010 0000

**Note –** These instructions would require three memory locations to store the binary codes. The mnemonic is always followed by 16-bit (or adr).