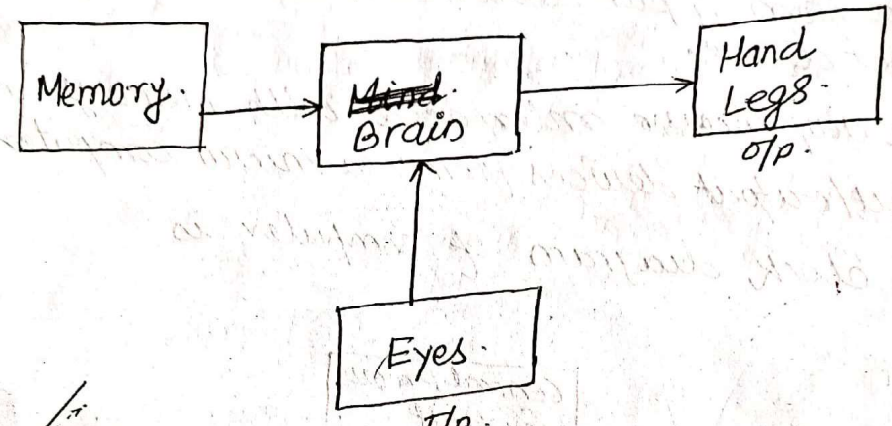
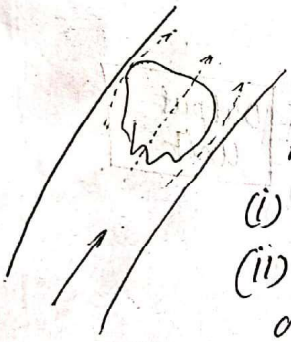




Assembly Language.

(i) Low level language. — Hindi

(ii) High level language.

or English → translate to Hindi → Hindi.

↓

1 paragraph at at time (Complier)

line by line (Interpreter)

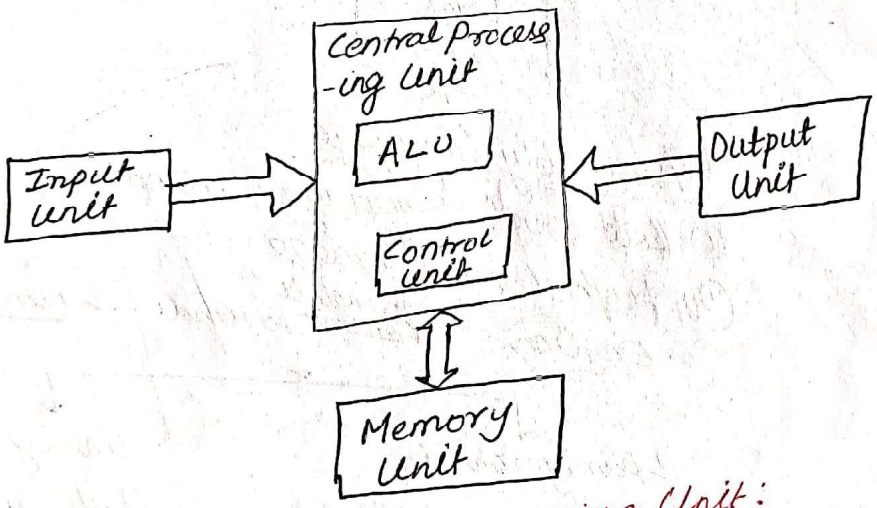
1. Microprocessor Architecture, Programming and applications with 8085. Ramesh Gaonkar.

2. An introduction to microprocessor 8085. Dr. D.K. Kaushik.

Microprocessor: Microprocessor is a digital device on a chip which can fetch instruction from a memory, decode and execute them i.e. perform certain arthematic and logical operation accept data from input device, and send result to output device.

A microprocessor interfaced with memory and Input/output devices forms a micro computer

The block diagram of computer is



~~Input Unit~~:- Central processing Unit:

* CPU consists ALU, register unit and control unit.
* It fetches the ~~the~~ stored instruction from the prog. memory, data word from data memory or from an input device and after processing the data stores the result in data memory or sends it to the output device.

a) ALU:- This unit performs computing functions on m-bit data where m is bit size of processor

* These functions are arthematic operation such as addition & subtraction and logical operation as AND, OR, XOR. etc.
* Results are either stored in register or memory or sent to output device.

(b) Register:- It contains various 8 bit & 16 bit registers. These register are used primarily to store, data temporarily during execution of program.

8085 (A) microprocessor contains 8 bit register such as Accumulator (reg A), B, C, D, E, H, L etc. and 16 bit register such as Program Counter & Stack pointer.

(c) Control unit:- It provides necessary timing and control signal required for the operation of microcomputer. It controls the flow of data between microprocessor and its peripherals. The control unit gets a clock signal which determine the speed of microprocessor.

Basic function of CPU:-

1. It fetches an instructions word stored in memory
2. It decodes the instruction to determine what instruction is telling it to do.

(3)

3- Executes the instruction. Executing instruction may include following major task

(a) transfer data from one register to another register in CPU itself

(b) transfer data ~~from~~ between a CPU register ~~to another register itself~~ and specified memory location or I/o device

(c) Perform arthematic and logical data from specific memory location or a designated CPU registster.

4- It looks for control signal such as interrupts and provides appropriate responses.

5- It provides status, control, and timing signal that memory and I/o section can use

Memory:- It ~~contains~~ stores both the instruction to be executed (i.e. program) and the data involved.
It contains (i) ROM (Read only memory).
(ii) RWM (Read write memory).

ROM:- The ROM can only read and can't be written into and is non volatile that is, it retains its content when power is turned off. It is typically used to stores the ~~and~~ instruction and data that donot change.
eg: monitor prog. of microcomputer.

RWM:- One can read and write in RWM. The RWM is volatile i.e. it doesnot retains its content when power is turned off. It is used to store user prog and data which are temporary.

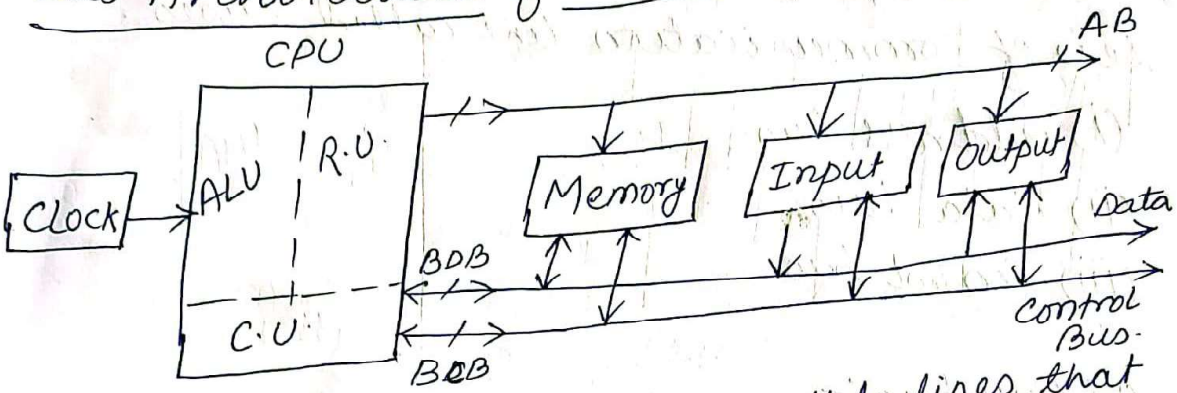
Input/Output Port:- The input and output port provides the micro computer the capabilites to communicate with outside world. The input port allows ~~the~~ data to pass from outside world to µc data which will be used in data manipulating being done by micro computer to send data to output device.

eg: Input device : keyboard, pendrive, trausducers etc
O/p device: LED, CRT, Printers, etc.

I/o device are called peripherals.

Clock generator:- Operation inside the microprocesser as well as other part of microcomputer are usually synchronous by nature. This is done so that event ~~taking~~ in different parts of system can proceed in systematic fashion. The clock needed to perform this sync. operation is provided by clock generator. The clock generator generates the appropriate clock period during which instruction executions are carried out by µP.

Bus Architecture of 8085:-



Bus: Bus is a group of parallel lines that connect two or more devices. It carries information in bits.

Microprocessor basicly perform 4 operation
(i) Memory – Read / Write
(ii) I/O device – " "

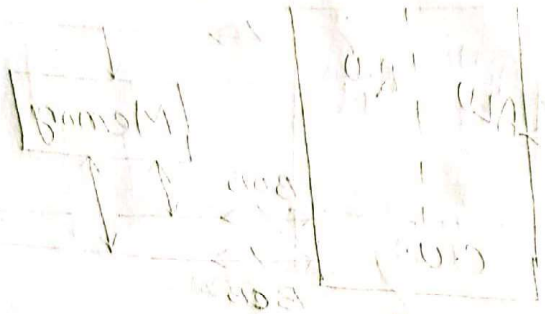
All these 4 operation requires communication between µp and peripheral device (including memory).

To communicate with peripherals MPU performs following steps/operations.

1. Identify the peripheral or the memory location (with its address).
2. Transfer binary information (data/instruction
3. Provide timing and synchronizing sig.

8085 MPU performs these function using three sets of communication line called 'buses':

(i) address bus
(ii) data bus.
(iii) Control bus.



Address bus:-

(i) Processor uses address bus to identify a peripheral or a memory location
(ii) Unidirectional i.e. information flow takes place only from µP to memory / IO device.
(iii) In 8085 Address bus is of 16 bit therefore µP can generate 2^{16} or 65536 different address.

Data bus:-

(i) This bus is used for transfer of data between µP and peripheral.
(ii) Bidirectional i.e. data transfer can take place in both dirn.
(iii) length of this bus is 8 bit

control Bus:-

(i) comparise of various signal lines that carry control signals
(ii) Bidirectional in nature

8085 Microprocessor Architecture :-

1. 8085 µP features :-

1. It is an 8 bit microprocessor designed using NMOS technology.
2. It is a 40 pin IC package ~~designed using~~ fabricated on a single LSI chip.
3. It uses +5V supply for its operation.
4. Its clock speed is about 3MHz and clock cycle is 320 nsec.
5. It consist of 4 main part.
   a.) Arthematic and logic unit (ALU).
   b.) Timing and Control unit
   c.) Register unit
   d.) Interface section

2. Historical developement :-

1. Microprocessor and microcomputer are catagorised in terms of no of data bits they process i.e. wordlength.
2. The standered data bits for microprocessor are 4 bit, 8 bit, 16 bit, and 32 bits.
3. Intel corporation introduced world's 1st µP the intel 4004, a 4 bit µP in 1971. It has 4bit wide memory location. It is designed to process 4 bit or a nebble of data. It has 45 instruction & 50 KIPs (Kilo instruction per second)
   Data - 4bit, Address - 12bit, Clock speed - 740KHz
   Application (i) Early video game system
   (ii) Calculators

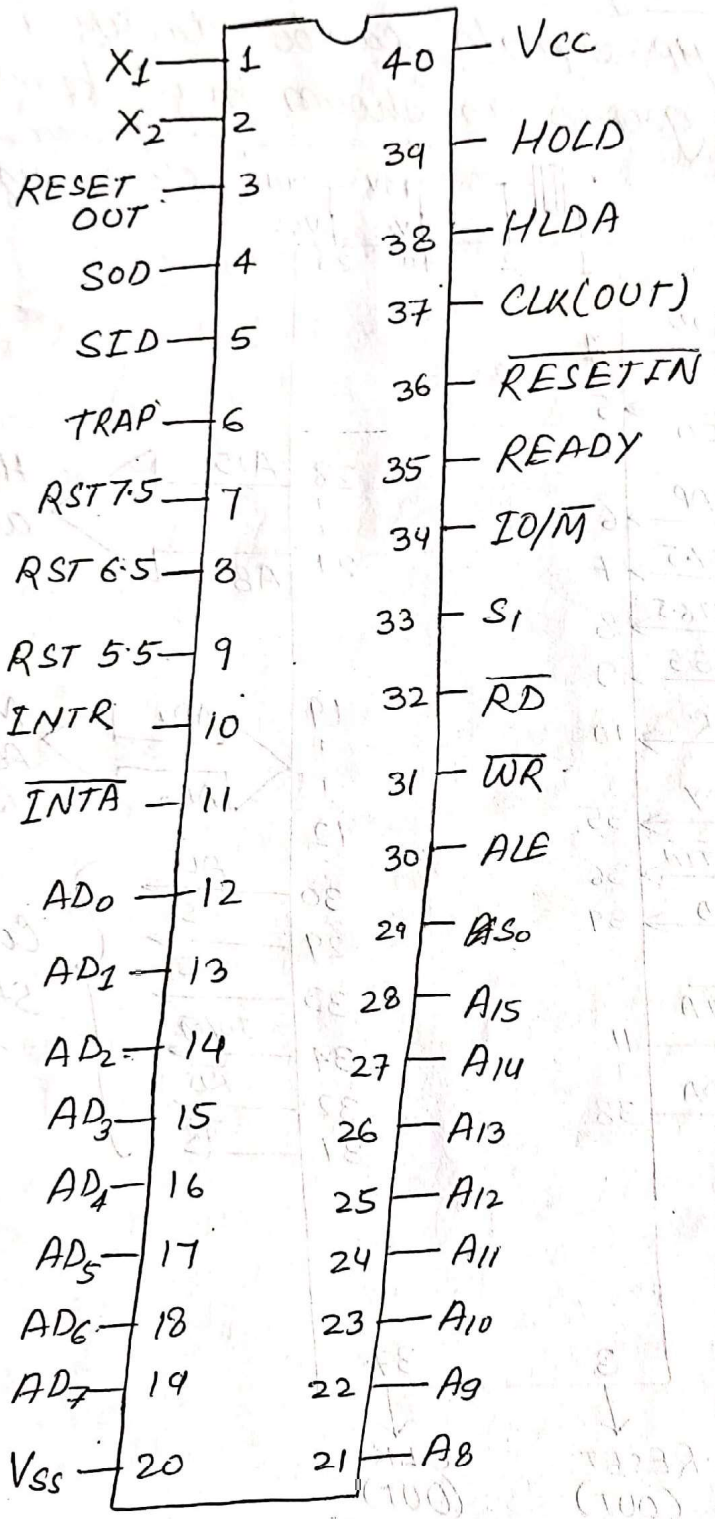
4. In 1972, INTEL released 8008 (extended version of 4004 µP). It can address 16 KB of extended memory and contain some additional instructions. 8 bit data bus & 14 bit address bit clk speed — 200 KHz 2.500 MHz

5. In 1974 INTEL launched, the first general purpose 8 bit µP 8080. 8bit, 16bit, 2 MHz to 3.125 MHz
* INTEL 8085 was developed in 1977 with a few more feature added to its architecture and instruction. It has 8 bit of data ~~lines~~, 16 bit of address and 64 KB of memory. It is 2nd generation µP.

6. In 3rd generation, 16 bit µP are developed like, 1978 – 8086 (16 bit)
(16 bit data, 20 bit address).

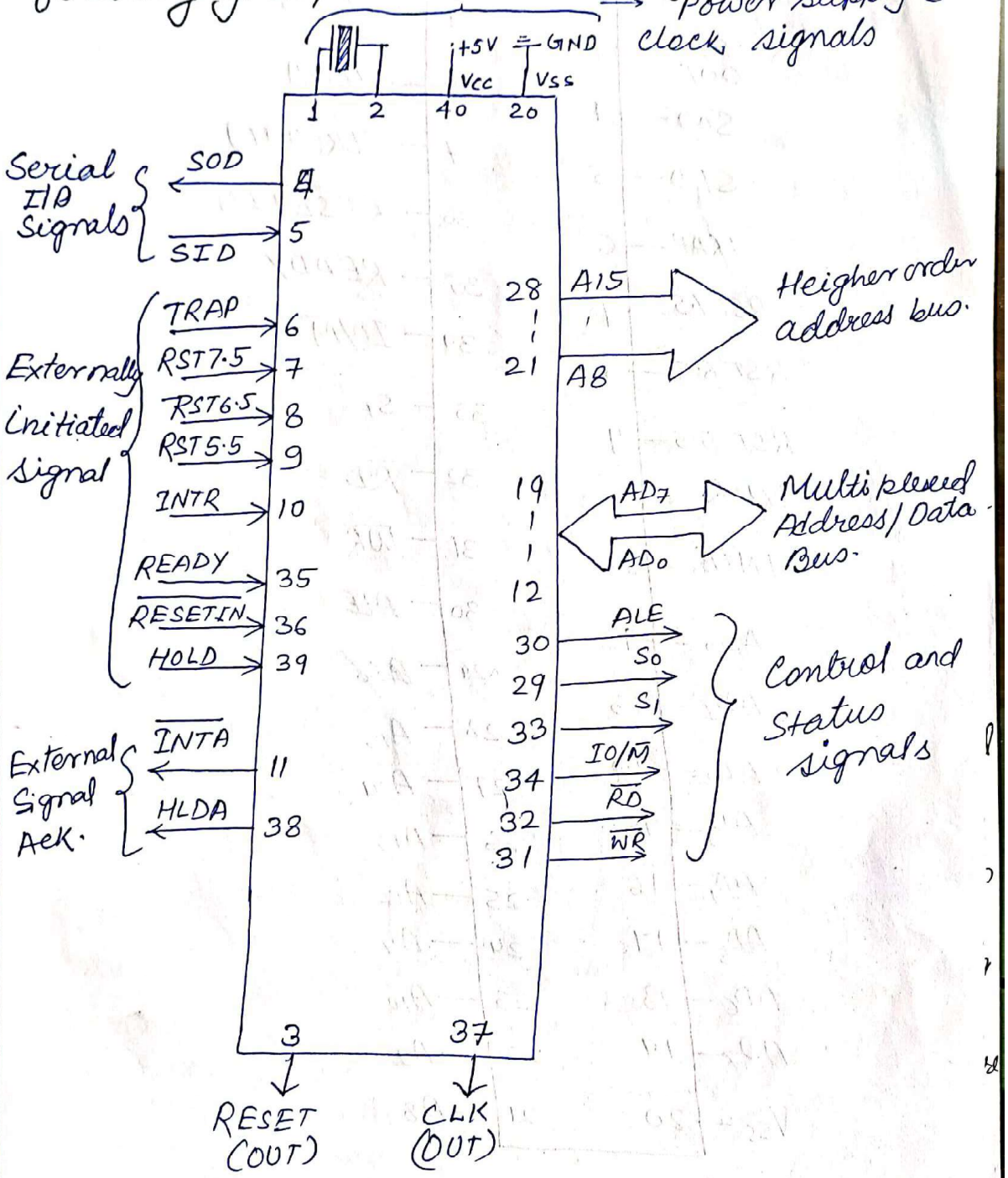
1983 – 80286
1986 – 80386
1989 – 80486

3. PIN Diagram :-



4. Pin Configuration:-

8085 µP signals can be classified into following groups as shown in fig. below:-



1. ADDRESS BUS:-

8085 has 16 pins that are used as the address bus, however these lines are split into two segments: A15 - A8 and AD7 - AD0. The 8 signal lines, A15 - A8 are unidirectional and used for most significant bits, called the heigher order address of 16 bit address. The signal lines AD7 - AD0 are used for dual purpose ~~as explained~~ i.e. as address and data lines.

2. Multiplexed Address/data lines:-

The signal lines AD7 - AD0 are bidirectional. they serve dual purpose. They are used as low order address bus as well as data bus. In executing as instruction, during the earlier part of cycle, these lines are used as low order address bus. During later part of cycle these lines are used as data bus. (This is known as multiplexing the bus.

If we had 16 pins dedicated to address bus & 8 pins dedicated to data bus then 24 pins are used only for addressing & data transfer. Since we do not require address and data at same time in that address we perform data transfer). we can share (multiplex) some of the pins for handelling both address and data and use remaining pins for other functions. Thus by multiplexing the addrss ss and data bus we increase the fun ctionality of µp.

• and reduce pin count, thereby reducing the size of µP.

3. Control and Status signal:-

This group of signals include two control signal ($\overline{RD}$ & $\overline{WR}$), three status signals ($IO/\overline{M}$, S_1, S_0) to identify the nature of the operation and one special signal (ALE) to indicate begning of the operation. These signals are as follows:-

(i) ALE (output):- (Address Latch Enable). This signal latches lower order address from the multiplexed bus and generate separate set of eight address lines. (A_7-A_0).
During 1st clock cycle as ALE pin goes high (i.e logic 1) at this time the multiplexed bus will ~~have~~ cook as address lines, in sub sequent cycle when ALE signal goes low it will act as data lines.

(ii) $\overline{RD}$ (output) Read This is read control signal (active low). This signal indicate that the data is to be read from selected I/O or memory device i.e. take the data from I/O or memory and put it is data bus.

(iii) $\overline{WR}$ (output, Write, active low): This is write control signal. This signal indicates that the data from the data bus is to be written into selected I/O or memory device i.e give the data from µP to I/O or memory.

(3) $IO/\overline{M}$ → This is a status signal used to differentiate between I/O and memory operation

$IO/\overline{M} \rightarrow 1 \rightarrow$ I/O operation

$IO/\overline{M} \rightarrow 0 \rightarrow$ Memory operation

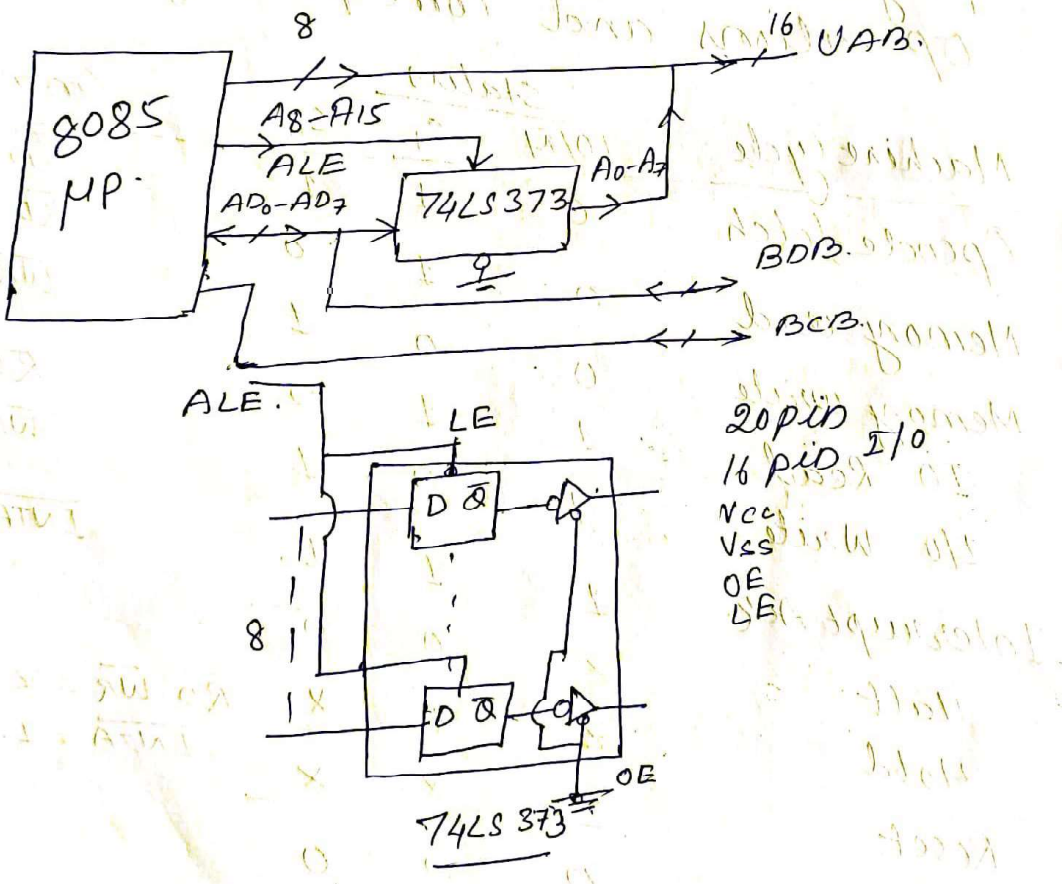
(4) S_0, S_1 – These are status signal similar to $IO/\overline{M}$. These status signal along with $IO/\overline{M}$ indicate type of machine cycle in progress.

Operations and Corresponding status s/g.

Machine Cycle	Status $IO/\overline{M}$	S_1	S_0	Control s/g
Opcode fetch	0	1	1	$\overline{RD} = 0$.
Memory read	0	1	0	$\overline{RD} = 0$.
Memory write	0	0	1	$\overline{WR} = 0$.
IO Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$.
Interrupt Ack.	1	1	1.	$\overline{INTA} = 0$.
Halt	Z	0	0	
Hold	Z	X	X	$\overline{RD}\ \overline{WR} = Z$
Reset	Z	X	X	$\overline{INTA} = 1$.
Bus Idle	0	0	0	

ALE :- This is an active high signal generated at the begning of operation. During 1st clock cycle as ALE pin goes high at this time the multiplexed bus will work as address lines in subsequent cycle when ALE signal goes low it acts as data lines.

ALE stands for Address Latch Enable. It is used to distinguish wether $AD_0 - AD_7$ bus contain address bits $A_0 - A_7$ or data bits $D_0 - D_7$



Power Supply and clock frequency:

1. $+V_{CC}$ → +5 V power supply.
2. V_{SS} → Ground Reference.
3. X_1, X_2 → A crystal is connected at these two pins. The frequency is internally divided by two, therefore to operate a system at 3 MHz, the crystal freq. should be 6 MHz.
4. CLK(out): This signal can be used as system clock for other devices.

Externally initiated signals:

External device can initiate following operations for which individual pins on the microprocessor chip are assigned: Reset, Interrupt,, Ready, Hold

Following signals comes under this group:

HOLD, RESET IN, READY, INTR, RST5.5, RST 6.5, RST 7.5, TRAP(RST 4.5)

1. HOLD: When HOLD pin is activated by an external device, the microprocessor relinquish control of buses and allow the external peripheral to use them.

2. RESET IN: When RESET pin is activated by external device all internal operation are suspended and the program counter is cleared (loaded with 00H). Now the program execution can again begin at the zero memory address .

3. READY: This signal is used to delay the microprocessor Read or Write cycle until slow responding peripheral is ready to send or accept data. When the signal goes low, the microprocessor waits for an integral no of clock cycles until it goes high again.

4. INTERRUPT: The microprocessor can be interrupted from normal execution of instruction and asked to execute some other instruction called service routine. The microprocessor resumes its operation after completing the service routine.

Microprocessor working when receives Interrupt signal

1. Microprocessor receives interrupt signal.

2. Stops execution of current program (after completing execution of current instruction), control transfers to subroutine by generating CALL signal, send acknowledgement signal to peripheral signal

3. After executing subroutine, by generating RET signal program control is again transferred to main Program

Classification of Interrupts:

Interrupts can be classified into various categories based on different parameters:

1. **Hardware and Software Interrupts –** When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as *Hardware Interrupts*. There are 5 Hardware Interrupts in 8085 microprocessor. They are – *INTR, RST 7.5, RST 6.5, RST 5.5, TRAP*

Software Interrupts are those which are inserted in between the program which means these are mnemonics of microprocessor. There are 8 software interrupts in 8085 microprocessor. They are – *RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7*.

2.**Vectored and Non-Vectored Interrupts –** *Vectored Interrupts* are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address.
Vector Addresses are calculated by the formula 8 * TYPE

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	24 H
RST 5.5	2C H
RST 6.5	34 H
RST 7.5	3C H

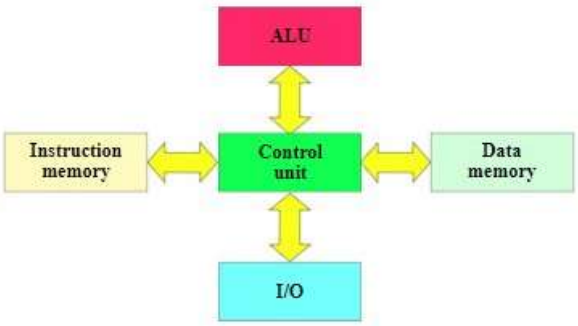
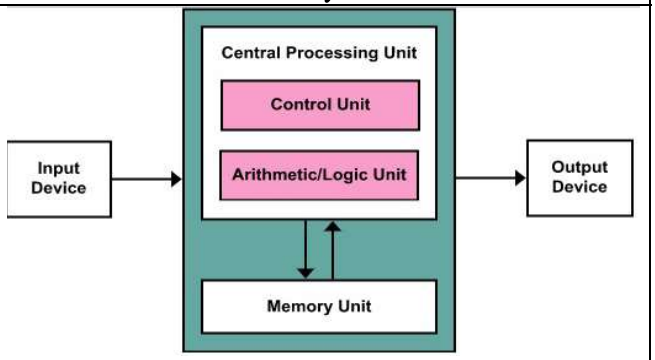
Non-Vectored Interrupts are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts. INTR is the only non-vectored interrupt in 8085 microprocessor.

Maskable and Non-Maskable Interrupts – Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled. INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupts in 8085 microprocessor.
Non-Maskable Interrupts are those which cannot be disabled or ignored by microprocessor. TRAP is a non-maskable interrupt. It consists of both level as well as edge triggering and is used in critical power failure conditions.

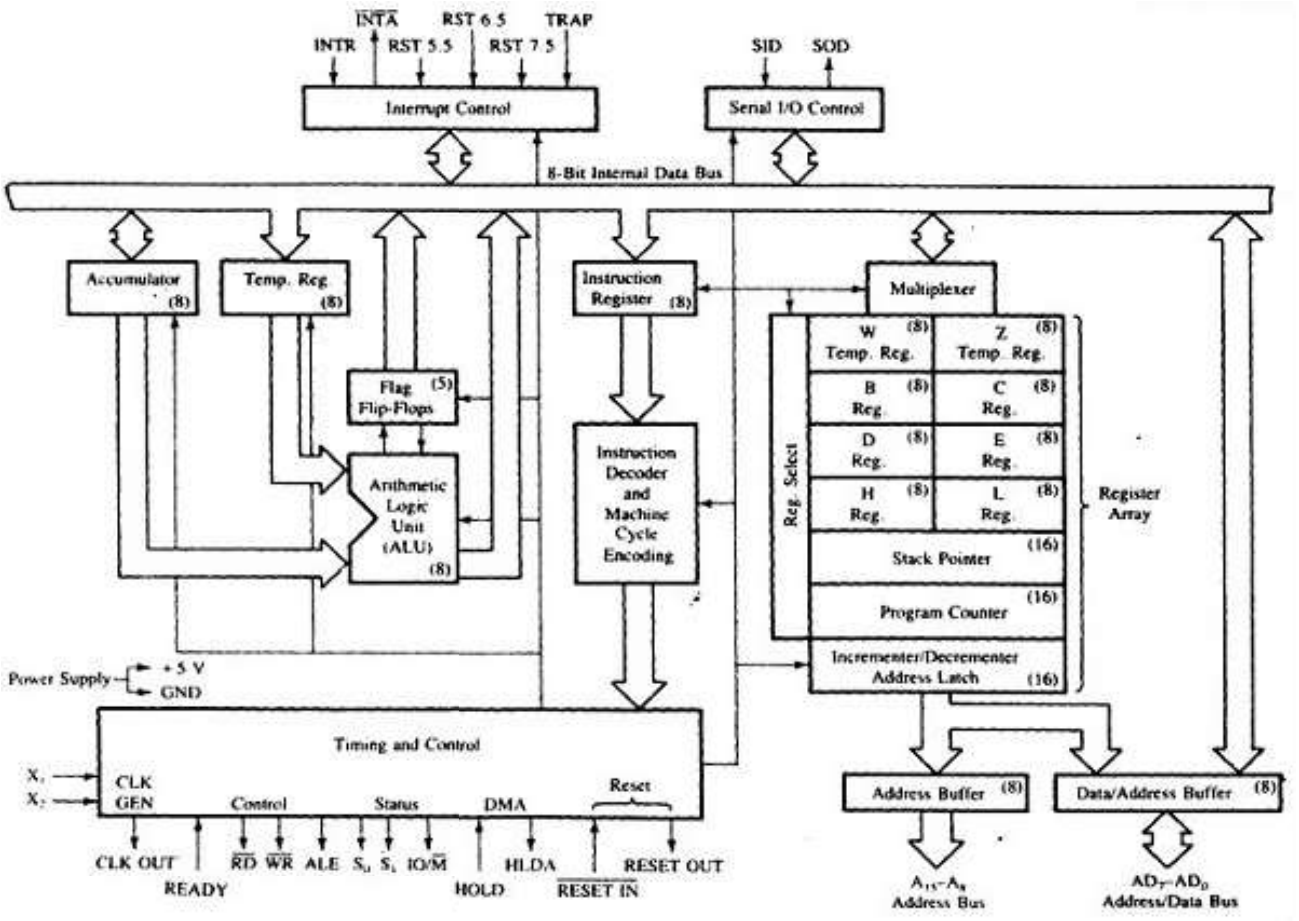
Priority of Interrupts –When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.

Architecture of 8085 microprocessor

The architecture of 8085 microprocessor chip is a description of physical layout of the various element that forms it. Two common architecture model used in computer organisation is Von Neumann architecture and Harvard architecture.

Harvard Architecture	Van Neumann Architecture
Harvard architecture has physically separate pathways for instruction and data.	Von Neumann architecture uses same physical pathway for instructions and data
It has one dedicated set of address and data bus for reading data from and writing data to memory and another set of address and data buses for fetching instructions.	It has same set of data and address buses for memory read/write and fetching instructions.
CPU can both read instruction and perform data memory access at the same time	CPU can be either reading an instruction or reading/writing data from/to the memory. Both can't occur at the same time since instruction and data uses same bus system
	

The 8085 microprocessor uses the Von Neumann architecture. The various units of 8085 microprocessor are discussed as follows.


INTA
RST 6.5
TRAP
INTR
RST 5.5
RST 7.5
SID
SOD
Interrupt Control
Serial I/O Control
8-Bit Internal Data Bus
Accumulator (8)
Temp. Reg (8)
Flag Flip-Flops (5)
Arithmetic Logic Unit (ALU) (8)
Instruction Register (8)
Instruction Decoder and Machine Cycle Encoding
Multiplexer
W Temp. Reg. (8)
Z Temp. Reg. (8)
B Reg. (8)
C Reg. (8)
D Reg. (8)
E Reg. (8)
H Reg. (8)
L Reg. (8)
Stack Pointer (16)
Program Counter (16)
Incrementer/Decrementer Address Latch (16)
Reg. Select
Register Array
Power Supply
+5 V
GND
Timing and Control
X1
X2
CLK GEN
Control
Status
DMA
Reset
CLK OUT
READY
RD WR ALE S0 S1 IO/M
HOLD
HLDA
RESET IN
RESET OUT
Address Buffer (8)
Data/Address Buffer (8)
A15-A8
Address Bus
AD7-AD0
Address/Data Bus

Registers

The data and instructions which are stored in the memory are fetched to the microprocessor and are stored temporarily in registers for processing

A register is nothing but set of flip flops. The number of flip flops used depends on the size of the data that the microprocessor needs to process.

Registers can be broadly classified into two categories, namely

1. **General Purpose Registers:** These registers have not been explicitly defined for any purpose or to store any particular type of information.

2. **Special Purpose Registers:** These registers are used for some specific function.

General Purpose Registers:

In 8085 microprocessor, there are 8 general purpose registers identified as B, C, D, E, H & L. They can be combined as register pair- BC, DE, and HL to perform some 16 Bit Operations. The HL register pair also work as a memory pointer (M), i.e. the 16-bit data is the address of a particular memory location. E.g. MOV r, M

Special Purpose Registers:

Program Counter:

This is a 16-bit register accessible to the user. It is a special purpose register and it always contains the address of the next instruction to be fetched from the program memory and executed by the CPU in a program sequence. Thus the program counter keeps the track of the program execution in which instructions are to be executed next.
Whenever necessary in the program execution, the address information available in PC is sent out to the address lines during T1 timing slot of a machine cycle. The higher order 8-bits of program counter (PCH) are sent out through A15–A8 address lines & the lower order 8-bits of program counter (PCL) are sent out through AD7–AD0 lines during T1 states.

Whenever the address information sent from the program counter to the address bus (external world) during T1 state, then the (PC) shall be incremented by 1 during the subsequent T2 state so that program counter points to the next sequential byte.

Stack Pointer

The stack pointer is a 16-bit register accessible to the user. It is required to refer any memory location of the stack. It contains the address of the top of stack into which last data is put or written. Writing data into a stack is called a PUSH operation and reading data from a stack is called a POP operation.

W-Z

(W) and (Z) are two 8-bit temporary registers not accessible to the user. They are exclusively used for the internal operation by the microprocessor. These registers are used either to store 8-bit of information in each (W) and (Z) registers or a 16-bit data in (W,Z) register pair with lower order 8-bits in (Z) and higher-order 8-bits in (W) register.

Address/Data Buffer and Address Buffer

The content of stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external data bus and address data bus. Any exchange of data between memory or I/O device is done through these buffers.

MUX/DEMUX

MUX/DEMUX unit is used to select a register out of all the available registers. This unit behave as MUX when data is going from the register to internal data bus. It behaves as DEMUX when data is coming to register from internal data bus of the microprocessor.

Register Select

The register select behaves as the select lines of MUX/DEMUX unit.

Increment-Decrement Address Latch:

It is another 16-bit internal register latch available in the register section for internal operations and is not accessible to the user. The address latch serves two functions.

First, it selects an address to be sent out from the program counter, from the stack pointer, or from one of the 16-bit register pairs.

Second, it latches this address onto the address lines for the required time. The 16-bit addresses from 8085A allow the microprocessor up to 216 memory locations through A15-A8 and AD7-AD0 lines.

An increment/decrement register allows the contents of any of the 16-bit registers to be incremented or decremented.

Instruction Register & Instruction Decoder:

The first word of an instruction is the operation code, i.e., binary code for that instruction. Therefore, in the first machine cycle of any instruction μp fetches the instruction from the memory. The op-code representing the instruction to be executed is fetched from the (program) memory location pointed to by (PC) and loaded into the instruction register (IR).

The IR passes this op-code to the instruction decoder which interprets this op-code appropriately in order to decide what operation needs to be done for executing this instruction.

The instruction decoder tells the control unit the type of instruction to be executed; the number of machine cycles necessary to execute the instruction etc. In response, the control unit generates all the necessary control signals which go into the different internal block of the microprocessor.

Arithmetic & Logic Section: This section consists of:

(a) Accumulator (A)

(b) Temporary Register (TR)

(c) Flag Register (FR)

(d) Arithmetic Logic Unit (ALU)

Accumulator:

Arithmetic and/or logic operations on one or two operations are the basic data transformations implemented in a μρ one of these two operands is always in the accumulator. Accumulator is an

8-bit register accessible to the user. The result of arithmetic/ logical operations carried out by ALU is also stored back in the accumulator.

Temporary registrar (TR):

This is an 8-bit register not accessible to the user. It is used by the processor for internal operations. The second operand as and when necessary is loaded in to this register by the microprocessor before the desisted operation takes placed in the ALU.

Arithmetic Logical Unit (ALU):

ALU is a combination logic block which performs the desired operation on the two operands. The contents of the accumulator and the temporary register are the inputs to the ALU. The various arithmetic and logical operations that can be performed by ALU are:

- Binary addition, subtraction, increment and decrement,
- Logical AND, OR and EX-OR,
- Complement,
- Rotate left of right.

The result of the operation is, in general, stored back in accumulator.

Flags register:

The ALU influences a number of flip flops called flags which store information related to the results of arithmetic and logical operations. Taken together this flags constitute a flag register. Flag register is an 8-bit register accessible to the user through instruction. Each bit in the flag register has a specific function. Only 5 bits out of 8 bits are used as shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

The three crossed bit are redundant bits and not used. They can be either ‘0’ or ‘1’ but normally they are forced to be zero. The other five bits are affected as a result of execution of an instruction.

CY (Carry) Flag bit:

This particular bit is SET (=1) if there is a carry from the MSB position during an addition operation or if there is a borrow during the subtraction operation otherwise the flag is reset (=0). The processor, by design, does the subtraction operation also by taking 2's complement of one operand and adding it to another operand.

P (Parity) Flag bit:

The parity flag test for the number of '1's in the accumulator. If the accumulator holds on an even number of 1's, it is said that even parity exists and the parity flag is set to '1'. However, if the accumulator holds an odd number of '1' it is called odd parity and the parity flag is reset to '0'.

AC (Auxiliary Carry) Flag bit:

This bit is set if there is a carry from b_3 bit to b_4 bit of accumulator during addition operation otherwise it is reset.

Z (Zero) Flag bit:

Zero flag bit is SET if the result of an operation is zero, otherwise it is RESET.

Sign Flag bit:

The sign flag is set to the condition of the most significant bit of the accumulator following the execution of arithmetic or logical operation. These instructions use the MSB of the data (result) to represent the sign of the number contained in the accumulator. A set sign flag represents a negative number, where as a reset flag means a positive number.

Serial I/O Control:

I/O devices work with serial data rather than parallel. In this case, the serial data stream from an input device must be converted to 8-bit parallel data before the computer can use it. Likewise the 8-bit data out of a processor must be converted to serial form before a serial output device can use it.

The SID (Serial Input Data) input is where serial data enters the 8085A. The SOD (Serial Output Data) output is where the serial data leaves the 8085A.

Interrupt Control Section:

To enable the processor to service the device requesting service through interrupt, processor accepts and issues control signals through interrupt control section.

Timing Diagram for 8085 Instructions

The working of 8085 microprocessor can best be understood by considering the timing diagrams. Timing diagram is graphical representation of various control signal generated during execution of an instruction. In timing diagram buses and various control signals are shown which include the following:

- Higher order address bus: $A_{15} - A_8$
- Multiplexed data and lower order address bus: $AD_7 - AD_0$
- ALE
- IO/$\overline{M}$
- S0, S1
- $\overline{RD}$
- $\overline{WR}$

Before we discuss timing diagram, we need to discuss few terms associated with timing diagram:

Machine Cycle

Machine cycle is defined as time required completing one operation of accessing memory I/O or acknowledging an external request. In other words every time a byte of data is move from CPU to I/O or memory or from memory or I/O to CPU, a machine cycle is required.

There are seven different kinds of machine cycles in the 8085 A:

1. Opcode Fetch Machine Cycle (OFMC)
2. Memory Read Machine Cycle (MRMC)
3. Memory Write Machine Cycle (MWRMC)
4. I/O Read Machine Cycle (IORDMC)
5. I/O Write Machine Cycle (IOWRMC)
6. Interrupt Acknowledge Machine Cycle (INTAMC)
7. Bus Idle Machine Cycle (BIMC)

Flow Chart For Machine Cycle:

1. Three status signals IO/M, S1 and S0 generated at the beginning of each machine cycle.
2. $\overline{RD}$, $\overline{WR}$ and INTA generated during T2 state of the machine cycle identify each type of the machine cycle
3. The status signals remain valid for the entire duration of the cycle.
4. The instruction fetch portion of an instruction cycle requires a machine cycle for each byte of the instruction to be fetched.

T-state

T state is defined as one subdivision of the operation performed in one clock period. Each T state is precisely equal to one clock period.

So, if the clock frequency is 3 MHz then 1 T state will be equal to

$$T_{clk} = \frac{1}{f_{clk}}$$

T_{clk} is of 320ns.

Instruction cycle

Instruction cycle is defined as time required to complete total execution of an instruction. The number of machine cycles required to execute the instruction depends on the particular instruction. The first machine cycle of an instruction cycle is always an OPCODE FETCH machine cycle which is always single byte long and the 8-bits obtained during an OPCODE FETCH are always interpreted as an OPCODE of an instruction. Some of the instructions require no addition machine cycles after the instruction fetch is complete, other requires additional machine cycles to write or read data to or from memory or I/O devices.

For Example: EI, DI, RLC, HLT

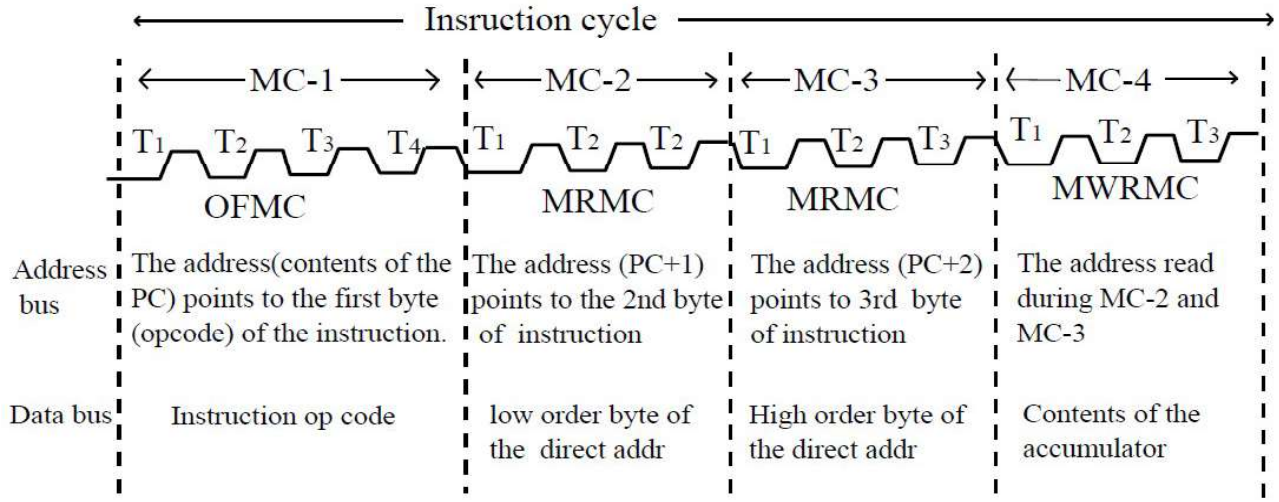
STA ADDR: STA stands for store accumulator direct. The meaning of the instruction is transfer the content of the accumulator to an external memory location whose address is specified in the instruction i.e. ADDR. Since this location can be anywhere in the 64k memory space that the 8085A can directly address, 16-bits are required for the address. Thus the instruction contains 3 bytes- a 1 byte op-code and 2-byte address. The instruction is stored in the memory as follows:

OP CODE	Byte -1
LOWER ADDR	Byte - 2
HIGHER ADDR	Byte - 3

Three machine cycles (MC) are required to fetch this instruction. In MC-1, i.e., Op-code fetch machine cycle, the op-code is transferred from memory to the instruction register during T1-T3 states and then during T4 state it is interpreted. At this point, the CPU knows that it must do more machine cycles - two MRMCs to fetch the complete instruction. In MC-2 the lower address is transferred from the memory to the temporary register (Z). In MC-3 the third byte, i.e. the higher byte address is transferred from the memory to the temporary register (W). When the entire instruction is in the μp, it is executed. Execution means a data transfer from the μp to memory. The content of the accumulator is transferred to the memory location, whose address was previously transferred to the μp by the proceeding two memory read machine cycles.
The address of the memory location to be written is generated as follows:
The high order address byte in temp register (W) is transferred to the address latch and the low order address byte in temp register (Z) is transferred to address/data latch. The content of the (A) is then placed on the data bus. This data transfer is affected by a MWRMC.
Thus 3-byte STA instruction has four machine cycles in its instruction cycles.

Mnemonics	Instruction Byte	Machine cycle
STA	Op-Code	OFMC
	Lower Address	MRMC
	Higher Address	MRMC
		MWRMC

The action taken in different machine cycles are:

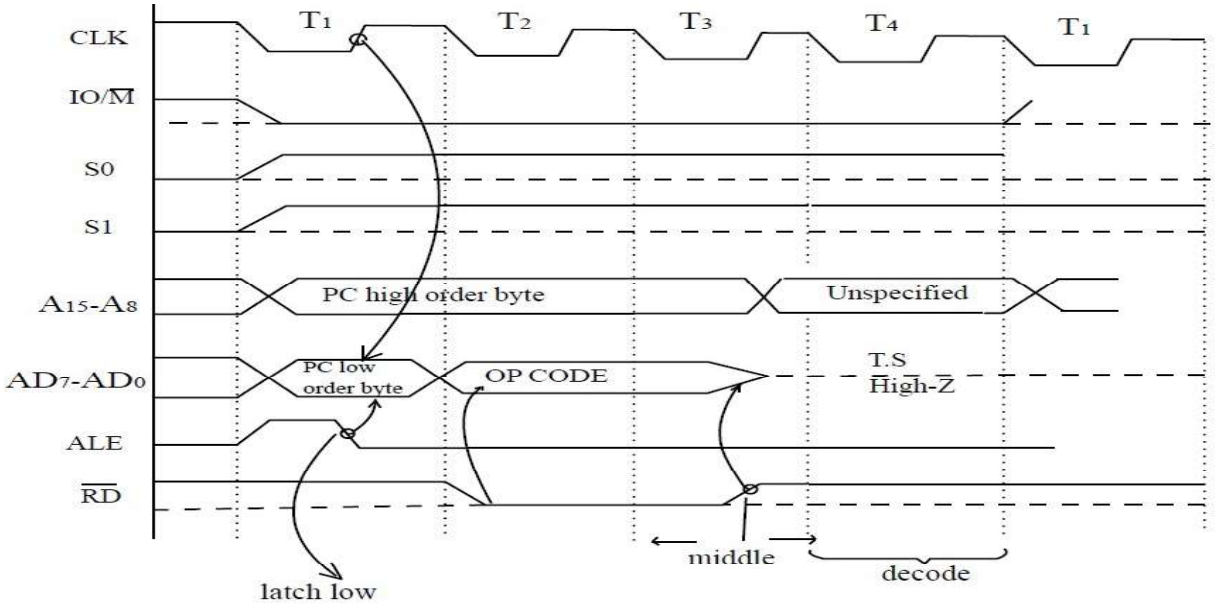


Thus STA ADDR instruction has a total of 13 states. If the 8085A is operating at 325.5ns time, the STA instruction cycle is executed in 4.23 μsec.

OPCODE FETCH Machine Cycle

The instruction fetch cycle requires either four or six clock periods (Tstates). The purpose of an OFMC is to read the contents of a memory location containing the opcode addressed by the program counter and to place it in the instruction register (IR).

The following Figure shows the 8085A instruction fetch timing diagram.



Steps involved in Op Code fetch machine cycle:

T1 State:

1. The 8085A puts a low on the IO/$\overline{M}$ line of the system bus indicating a memory operation.
2. The 8085A sets S1=1 and S0=1 on the system bus, indicating the memory fetch operation. This status information remains available for the duration of the machine cycle.
3. Address latch enable (ALE) signal issued by the mp during T1 is used to latch this lower order address in some external latch 74LS373 on its falling edge.
4. The 16-bit address A15-A0 of the memory location containing the opcode is obtained from the program counter (PC) and placed on the address and address/data latches. The higher order 8-bits of the address appear on the address bus A8- A15 remains constants until the end of the state T3. During T4 state the data on the address bus is unspecified. The low order 8-bits of the address are placed on the address/data bus, AD7-AD0 at the beginning of T1. This data however remains valid only until the beginning of state T2 at which time the address/data bus is floated (tri-stated) because this is time multiplexed bus and used as the data bus during T2 and T3 states.

T2 State:

1. The $\overline{RD}$ signal goes low indicating read operation and the opcode to be fetched is placed on the data bus, AD7-AD0 by the addressed memory location.
2. The contents of (PC) is incremented be 1 during this state as during T1 state the (PC) has sent the address to address bus.

T3 State:

1. On the rising edge of the $\overline{RD}$ control signal in T3 state, the opcode obtained from the memory is transferred to the microprocessor instruction register.

T4 State:

1. The 8085A decodes the instruction and determines whether to enter state T5 or to enter T1 state of the next machine cycle. From the operation code, the mp determines what other machine cycles, if any, must be executed to complete the instruction cycle.

Summary

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← BDB

T_4: μp decodes the opcode and decides whether T_5 and T_6 states are required or next machine cycle executed is T_1

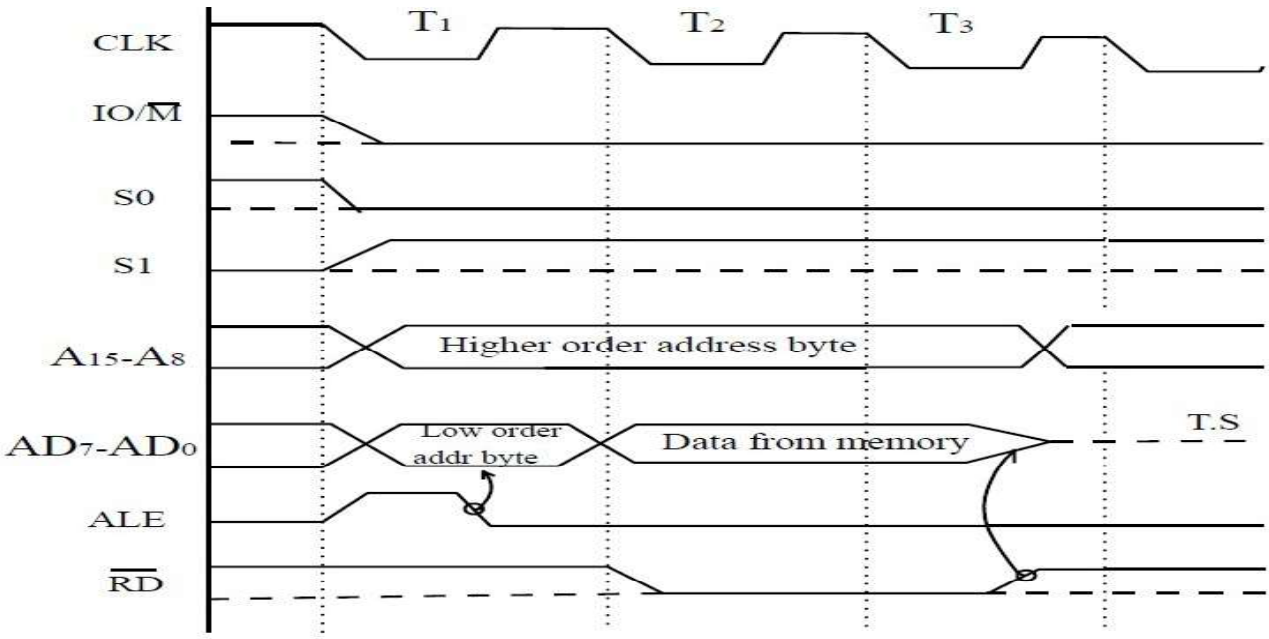
Memory READ Machine Cycle

The purpose of the memory READ operation is to read the contents of a memory location addressed by a register pair and place the data in one of internal registers of the mp.

It requires 3 states T1 to T3

The source of address issued during T1 is not always the program counter but may be any one of the several other register pairs in the μP depending on the particular instruction of which the machine cycle is a part.

The timing diagram during memory ready machine cycle is shown in fig



Steps involved in memory READ machine cycle:

T1 State

1. The IO/M signal is made LOW to indicate the external world that a memory reference is required.
2. μP made S0=0 and S1=1 indicating that memory READ operation is to be performed.
3. The μP places the contents of higher byte of the memory address register, such as that contents of the (PCH) or (H) register on A15-A8 and the contents of the lower byte of the memory address register such as contents of the (PCL) or (L) register on AD7-AD0.
4. The μP sets ALE signal HIGH, as soon as ALE goes to LOW in the middle of T1, the lower byte of the address is latched in an external latch.

T2 State

1. The $\overline{RD}$ signal goes LOW indicating a READ operation. The external logic gets the data from the memory location addressed by the memory address register such as (H,L) pair and places the data on to bi-directional data bus AD7-AD0.

2. The contents of (PC) is incremented be 1 during this state as during T1 state the (PC) has sent the address to address bus.

T3 State

1. $\overline{RD}$ signal goes HIGH. This LOW to HIGH transition of signal transfers the data from the data bus to internal register

Summary

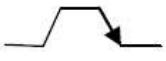
Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Internal Reg.) ← AD_7-AD_0 or BDB

OR

T_1: A_{15}-A_8 ← (H), AD_7-AD_0 ← (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ← M(AB)

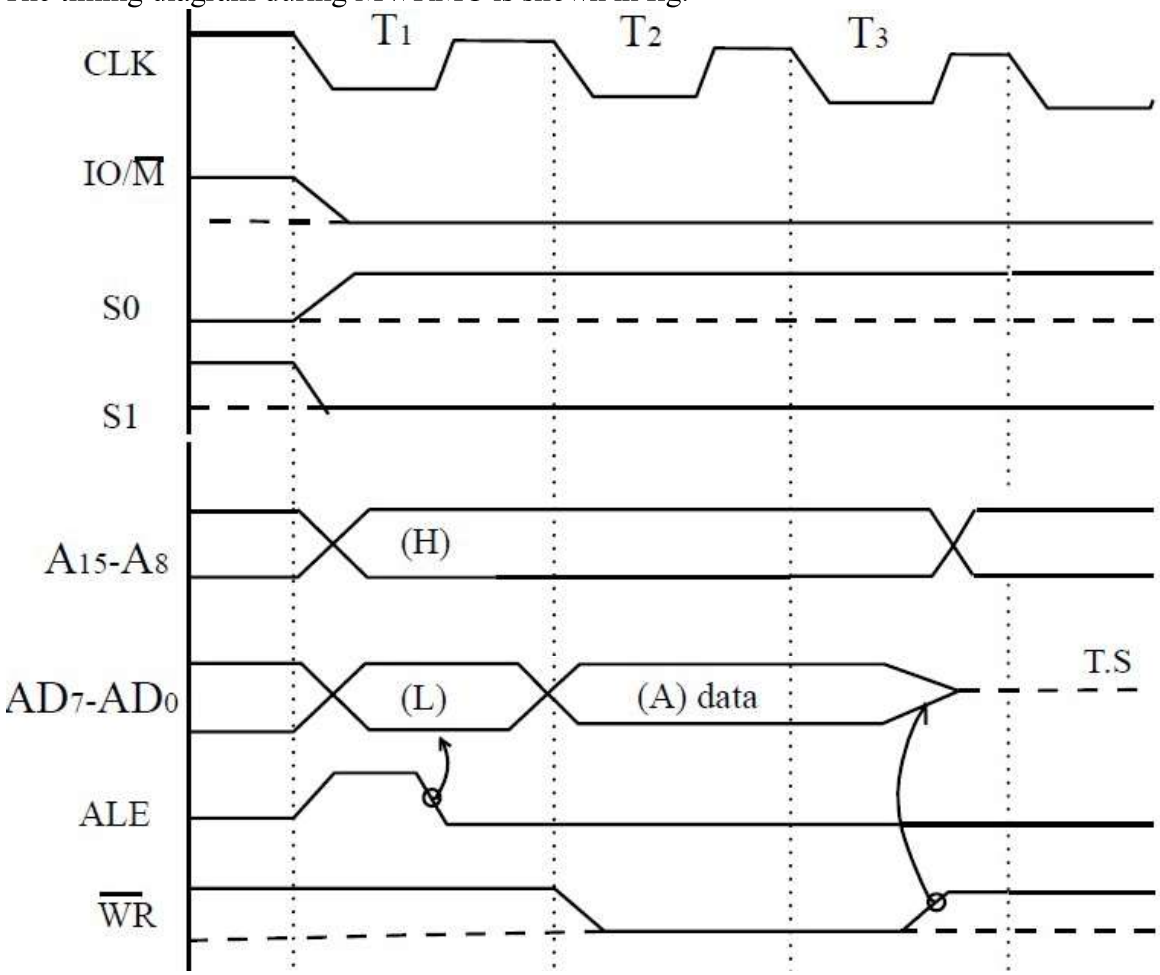
T_3: $\overline{RD}$ = 1, ↑ , (Internal Reg.) ← AD_7-AD_0 or BDB

Memory WRITE Machine Cycle

The purpose of memory write machine cycle is to store the contents of any of the 8085A register such as the accumulator into a memory location addressed by a register pair such as (H,L).

It also requires 3 T states.

The timing diagram during MWRMC is shown in fig.



Steps involved in memory WRITE machine cycle:

T1 State

1. The 8085A μP made IO/M = 0 in the beginning of T1 state to indicate memory reference operation. Then it puts S0 = 1 and S1 = 0 indicating a memory write operation.
2. 8085A places the memory address register (MAR) higher byte such as the contents of the (H) register on lines A15-A8 and also places the MAR lower byte such as the contents of the (L) register on lines AD7-AD0.
3. The μp sets ALE signal HIGH indicating the beginning of MWRMC. As soon as ALE goes to low, the lower byte of the address is latched in an external latch.

T2 State

1. $\overline{WR}$ goes LOW indicating memory write operation. It also places the contents of the internal register, say accumulator, on data lines AD7-AD0.

T3 State

1. $\overline{WR}$ goes HIGH. This LOW to HIGH transition is used to transfer the data from the data lines to the memory location address by MAR such as (H,L) register pair.

Summary

Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

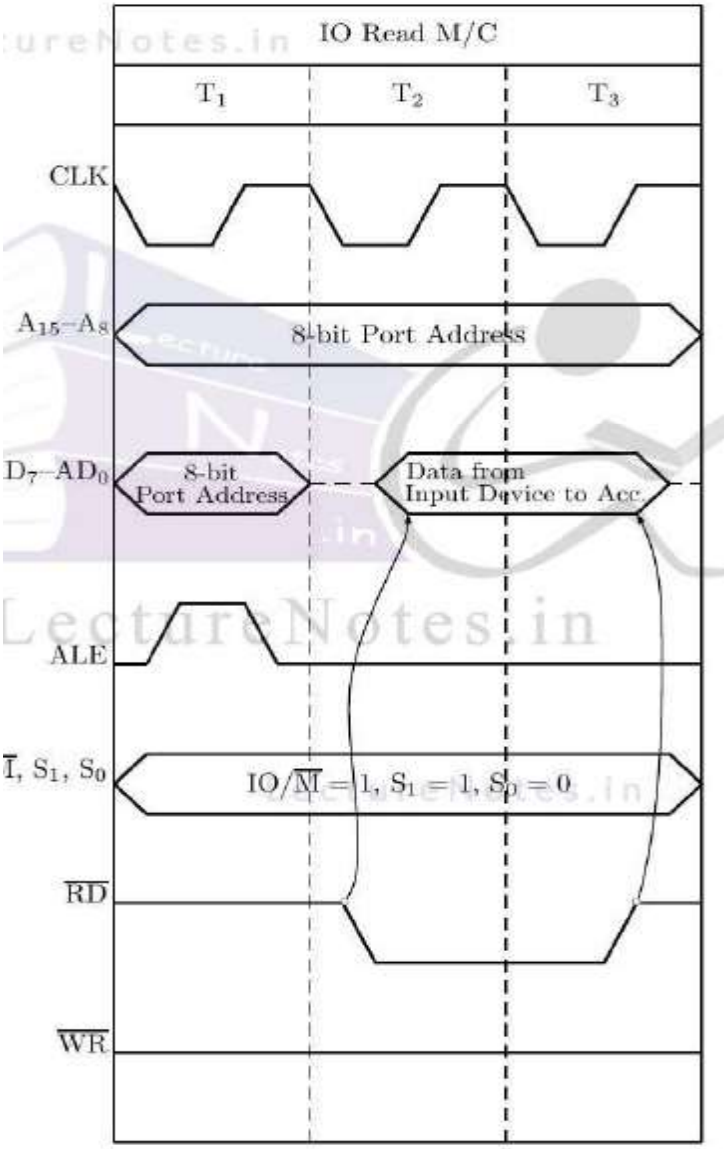
T_1: A_{15}-A_8 ← (H), AD_7-AD_0 ← (L), ALE = 

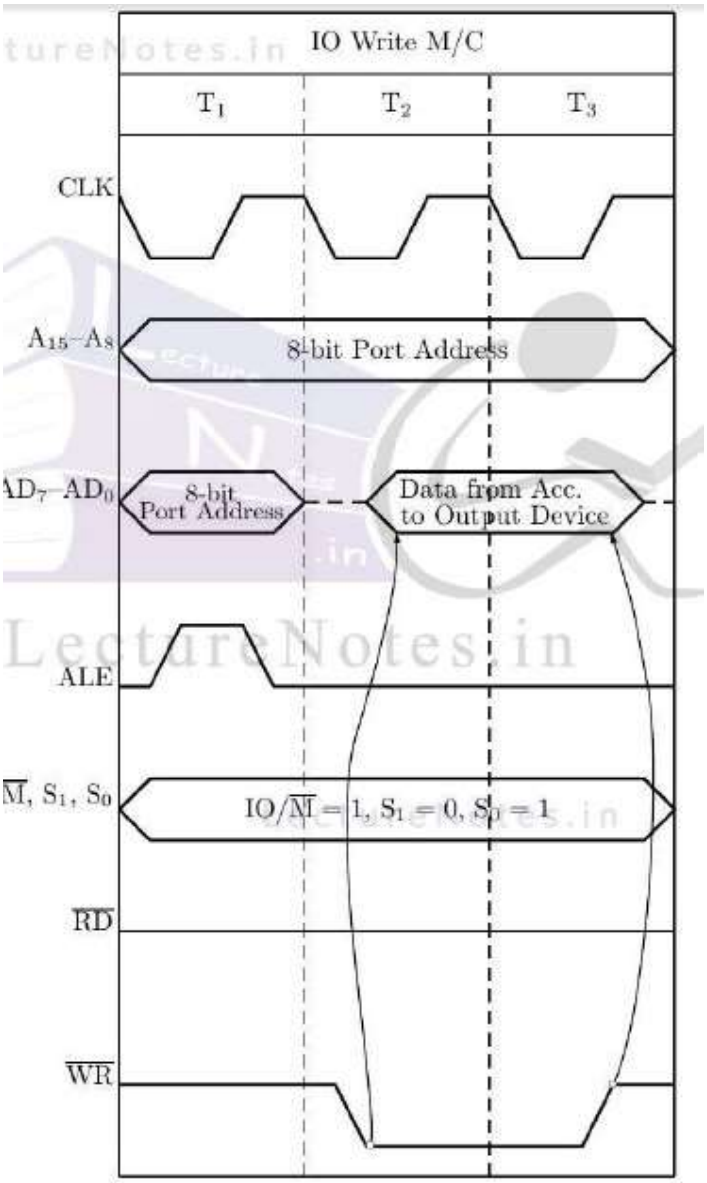
T_2: $\overline{WR}$ = 0, AD_7-AD_0 ← (μp Internal Reg.)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ← AD_7-AD_0 or BDB

I/O READ and I/O WRITE M/C cycle

1. The IORDMC and IOWRMC are identical to MRMC & MWRMC respectively except that appropriate status signals are issued at the beginning of T1 state.
2. IO/M signal goes HIGH at the beginning to indicate I/O device reference is needed in case of I/O mapped input/output device.
3. In these machine cycles higher & lower address bytes are identical and equal to the 8-bit address of the I/O port while in case of MRMC or MWRMC, the address bus output is the true 16-bits address.





HALT State (T_{HALT})

Whenever the HLT instruction is executed μp enters in to the HALT state. The opcode for HLT instruction is 76H. Assume that an opcode fetch machine cycle is initiated, and the opcode transferred to the instruction register during T3 state is 76H i.e. the opcode of HLT instruction. During state T4, the control unit decodes the instruction opcode and sets an internal HALT flip-flop of the processor. Upon exiting state T4, the μp enters state T1 of the next machine cycle. The HALT flip-flop is checked in T1 state of the next machine cycle. If it is found set, instead of entering T2 state, the μp enters in to the HALT state. Thus five states are required to reach the HALT state.

In the HALT state, the address and address/data buses along with $\overline{\text{RD}}$, $\overline{WR}$ and IO/$\overline{\text{M}}$ are placed in their high independence states (floated).

There are only three ways to exit from a HALT state

1. A LOW on $\overline{\text{RESETIN}}$ input of the 8085A resets the entire system and loads the program counter with all 0's. When $\overline{\text{RESETIN}}$ signal is active, μp comes out of HALT state and enters into RESET state and remains their as long as $\overline{\text{RESETIN}}$ is active. After reset, 8085A immediately starts program execution from 0000H.

2. The second way to get out of the HALT state is to make the HOLD signal input high. The processor then enters the HOLD state, but when the HOLD input goes LOW again, the CPU returns to the HALT state.

3. The third method of coming out of a HALT state is when and interrupt signal is active. This method works only if interrupts were enabled with an enable interrupt (EI) instruction in the program before HALT instruction is executed. Whenever interrupt comes μp leaves the HALT state and start executing the interrupt service subroutine (ISR).

WAIT StateT_{WAIT}:

According to timing specification for the 80854A, during a read operation (OFMC/ MRMC/ IORMC), the device providing data to the μp must have valid data on the dater bus within [(5/2) T-225] ns after the μp provides a valid address at its address pins.
For T=320ns, the memory or input device must have an access time of 575ns or less.
Sometimes microprocessors are used with memories or I/O devices which have longer access time. To accommodate long access time, the 8085A has a state called the WAIT state

Instruction Set

The 8085 microprocessor instruction set has 74 operation code that result in 246 individual instructions. An instruction is a binary pattern designed inside a microprocessor to perform specific function.

Each instruction has two part first is the task to be performed that is operation code and second is the data to be operated on that is operand.

The 8085 instruction has been classified based on three criterions:

1. Instruction size
2. Function
3. Addressing Mode

1. Instruction size

Instruction size refers to the number of bytes an instruction requires for its complete operations. Based on instruction size, instruction can be classified into the following three groups:

- **1 byte instruction:** An one byte instruction include opcode and the operand in same byte

Eg.: MOV C,A , ADD B etc

- **2 byte instruction:** In two byte instruction first byte specifies the opcode and second byte specifies the operand.

Eg.: MVI A 32H; ADI 54H (Opcode and 8 bit data)

- **3 byte instruction:** In three byte instruction first byte specifies the opcode and following two byte specifies the operand.

Eg: LDA 8000H, LXI H, F150H etc

2. Function

The 8085 instruction set can be classified in following five functional categories:

Data Transfer: This group of instruction copies data from source location and to destination location. The various types of data transfer takes place are:

a) Between registers – MOV D , B
b) Between memory location and register - LDA F200H
c) Specific data byte to a register or a memory location – MVI C 43H, and
d) Between I/O Device and memory – IN 85H

Arithmetic- These instructions perform arithmetic operations such as

a) Addition ADD B
b) Subtraction SUB C
c) Increment INR M and
d) Decrement DCX H

Logical- These instructions perform various logical operations with the content of accumulator such as

a) AND, OR, XOR- ANA B, ORA M, XRI 43H
b) ROTATE RLC RAR
c) Compare CMP B and
d) Complement CMA

Branching-This group of instruction alters the sequence of program execution either conditionally or unconditionally. Conditions here specify the status of flag:
JNZ F170H, CALL F070H etc

Machine Control- These instructions control machine function such as stop interrupt or do nothing HLT, NOP

Note: In arithmetic and logical operations, one of the operand is always in accumulator. The other operand can be a register, an 8-bit data, or a memory location. The increment and decrement operations are little bit different as they can be performed on any one of the register or in memory location.

3. Addressing Mode

Most of the instruction execution requires two operands e.g. transfer of data between two registers of a microprocessor system. How the μp knows the positions of these operands? The method of identifying the operands position by the instruction format is known as the addressing mode.
In Intel 8085A μp, the following addressing modes are used:

Register Addressing Mode:

When the operands for any instruction are available in internal general purpose registers, only the registers need be specified as the address of the operands. Such instruction are said to use the register addressing mode. These instructions are one byte instructions. Within that byte i.e. op-code, the registers are specified.
Eg: **MOV r1, r2; ADD r; XCHG; DAD rp**

MOV r1, r2 (47): This is an ALP statement and meaning of the instruction is move the content of register r2 to register r1.

ADD r (84): This is an ALP statement. ADD is the mnemonic for addition and meaning of the instruction is add the content of the register to the content of the accumulator and store the result back in accumulator.

Direct Addressing Mode:

In this addressing mode, the instruction contains the address of the operand (external register) involved in the transfer. The 8085A provides16-bit memory address requiring that the address contained in the instruction be 16-bit long as a second and third bytes of the instruction. Thus it is invariably a 3-byte instruction

LDA addr. This is an ALP statement 'addr' in operand field is a symbolic name given to 16-bit address. LDA is the mnemonic for Load Accumulator Direct. The meaning of instruction is load the accumulator from the memory location whose 16-bit address is available directly, in the instruction itself.

Register Indirect Addressing Mode:

In this case, the instruction specifies a register pair which contains the address of the memory where the data is located or into which the data in to be placed. Thus the address of the operand is given indirectly through a register pair. In other words, the operand is in memory location or external register whose address is available in an internal general purpose register pair. The (H, L) register pair is used as a pointer in many register indirect instructions of Intel 8085A. The register (H) holds the higher and register (L) holds the lower bytes of the effective address.

MOV r, M transfers single byte from an external register (M) to any of the several internal working registers. External register (M) means the external register pointed by (H,L) register pair.

The register pair (B, C) & (D, E) are also used as memory pointer register in two 8085A instructions i.e. **LDAX rp** and **STAX rp**. The internal register involved for data transfer is always accumulator.

The meaning of LDAX rp is load the accumulator from the memory location whose address is available in rp.

Immediate Addressing Mode

In this type of addressing mode, the operand is available directly in the instruction itself. If the operand data involved is of 8-bits then the instruction is of two bytes. The first byte is the opcode followed by 8-bit data byte. If 16-bit data is involved in the instruction then the first byte is opcode at memory location N followed by the lower order data byte at memory location N+1 and higher order data byte at memory location N+2.

MVI r, data there is two byte instruction LXI rp data.

Implied Addressing Mode:

There are certain instructions that operate on one operand. Such instructions assume that the operand is in the ACC and, therefore, need not specify any address.

Eg: RLC, RRC, RAR, RAL and CMA

All these are one byte instruction

DATA TRANSFER GROUP

It consists of 15 basic instructions and 86 variations. The basic operation involved is DATA transfer between two register of a microprocessor system. One of the register is always located in the μp itself; the other may be located in one of the following

1) An I/o device
2) Memory
3) The microprocessor

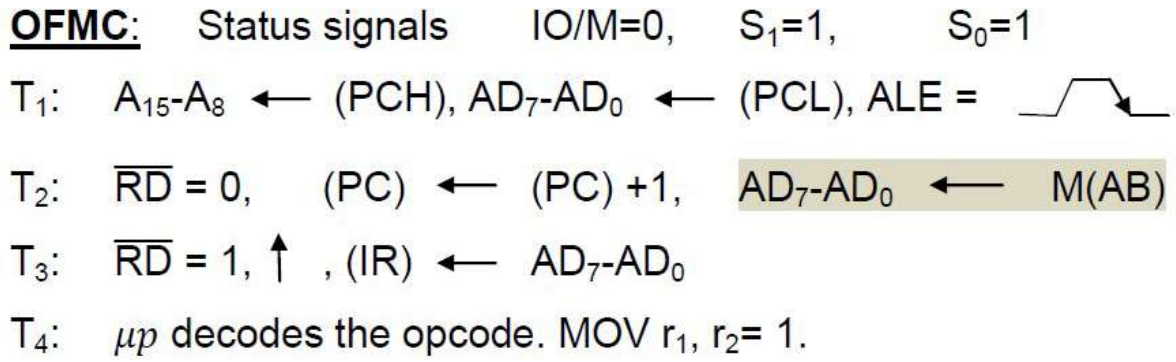
This group includes transfer of data from internal register to another internal register, internal register to memory, memory to internal register, accumulator (A) to output device, or from input device to accumulator.

The register from which data is transferred is the source register and the register to which data is transferred in the destination register. A transfer involves copying the contents of the source register into the destination register; the contents of the source register are not altered.

1. MOV r1, r2: This is an ALP statement. MOV is the mnemonic for move operation. r2 is the source register and r1 is the destination register in the operand fields. The meaning of the instruction is "Move the contents of the register r2 into r1". The content of register r2 is not destroyed. Content of r1 is destroyed and new value form r2 takes it place.

This is a **single byte instruction** at memory location N. It has **register addressing mode**.

The instruction cycle for MOV r1,r2 takes only **4 T states**.



This instruction has 49 variations.

2. **MOV r, M:** This is an ALP statement. The meaning of this instruction is "Move the content of the memory location whose address is available into (H,L) pair into the internal general purpose register r". M(H,L) is the source register, (r) is the destination register.

It is a **single byte instruction** at memory location N. This Operation requires only **two machine cycles**, OFMC & MRMC, and total **7 states.** The addressing mode is **register indirect addressing mode.**

This instruction has **7 variations.**

3. **MOV M, r**: This is an ALP statement. The meaning of the instruction is "Move the content of the internal general purpose register r into the memory location whose address is available in (H,L) register pair".

It is a **single byte instruction**. The operation requires only **two machine cycles- OFMC & MWRMC and total 7 states**. The addressing mode is **register indirect addressing mode.**

This instruction has **7 variations.**

4. **MVI r, DATA:** This is an ALP statement. DATA is the symbolic name given to 8-bit data which is immediately available as second byte of the instruction. Therefore, the source of data is the 2nd byte of the instruction itself. The meaning of the instruction is "Move 8-bit data immediately available as a 2nd byte of the instruction itself into the destination register r".

It is a **two byte instruction**. The operation requires only **two machine cycles- OFMC & MRMC and total 7 states**. The addressing mode is immediate addressing mode

This instruction has **7 variations.**

5. **MVI M, DATA:** This is an ALP statement. DATA is the symbolic name given to the 2nd byte of the instruction. MVI is the mnemonic for move immediate. M in the operand field stands for memory pointer. The meaning of the instruction is "Move 8-bit data available immediately as a 2nd byte of the instruction to the memory location whose address is available in memory pointed by (H, L) register pair".

It is a **two byte instruction**. The operation requires only **three machine cycles- OFMC MRMC & MWRMC and total 10 states**. The addressing mode is immediate addressing mode.

6. **LXI rp, DDATA:** This is an ALP statement. DDATA stands for double-data, a symbolic name given to 16-bit data available immediately as the 2nd & 3rd bytes of the instruction. „rp" stands for LOAD IMMEDIATE register pair. The alphabet „X" in the mnemonic tells that a register pair is involved in the instruction. It is true for all instructions. The meaning of the instruction is "Load the 16 bit data immediate available as the 2nd & 3rd bytes of the instruction into register pair rp".

It is a **three byte instruction**. The operation requires only **three machine cycles- OFMC & two MRMC and total 10 states**. The addressing mode is immediate addressing mode.

7. **LDA ADDR:** This is an ALP statement. ADDR is the symbolic name given to the 16 bit address directly available in the instruction. LDA is the mnemonic for LOAD ACCUMULATOR DIRECT. The meaning of the instruction is "Load the content of the memory location whose address is directly available in the instruction as 2^{nd} and 3rd bytes into the accumulator".

It is a **three byte instruction**. The operation requires only **four machine cycles- OFMC & three MRMC and total 13 states**. The addressing mode is direct addressing mode.

8. **STA ADDR:** This is an ALP statement. STA is the mnemonic for the STORE ACCUMULATOR DIRECT. The meaning of the instruction is “Store the content of the accumulator into the memory location whose address is directly available in the instruction itself as a 2nd & 3rd byte of instruction”.

It is a **three byte instruction**. The operation requires only **four machine cycles- OFMC, two MRMC and one MWRMC and total 13 states**. The addressing mode is direct addressing mode.

9. **LHLD ADDR**: This is an ALP statement. LHLD is the mnemonic for LOAD (H,L) REGISTER PAIR DIRECT. The meaning of the instruction is “Load the content of the memory location whose address is directly available as 2nd & 3rd byte of the instruction to register L and the content of the memory location at next higher address to the register H”.

It is a **three byte instruction**. The operation requires only **five machine cycles- OFMC, four MRMC and total 16 T states**. The addressing mode is direct addressing mode.

10. **SHLD ADDR:** This is an ALP statement. SHLD is the mnemonic for STORE (H, L) REGISTER PAIR DIRECT. The meaning of the instruction is the “Store the content of (L) register into the memory location whose address is available as 2nd & 3rd byte of the instruction itself and store the content of (H) register into the memory location whose address is next higher address”.

It is a **three byte instruction**. The operation requires only **five machine cycles- OFMC, four MRMC and total 16 T states**. The addressing mode is direct addressing mode.

11. **LDAX rp**: This is an ALP statement. LDAX is the mnemonic for LOAD ACCUMULATOR INDIRECTLY. As discussed earlier, the alphabet „X" in the mnemonic tells that a register pair is involved in the instruction. “rp” stands for register pair. The meaning of the instruction is “Load the accumulator from the memory location whose address is available in a register pair specified in the instruction”. Only one operand in involved in this instruction & the operand is available in the memory location whose address is in a register pair.

It is a **one byte instruction**. The operation requires only **two machine cycles- OFMC, MRMC and totals 7 T states**. The addressing mode is direct addressing mode.

12. **STAX rp:** This is an ALP statement. STAX is the mnemonic for store accumulator indirectly using register indirect addressing mode. The meaning of the instruction is “Store the content of accumulator in the memory location whose address is available in register pair specified in the instruction”.

It is a **one byte instruction**. The operation requires only **two machine cycles- OFMC, MWRMC and totals 7 T states**. The addressing mode is direct addressing mode.

13. **IN PORT:** This is an ALP statement. PORT is the symbolic name given to 8-bit address of the input device available as a 2nd byte of the instruction. IN is the mnemonic for INPUT. The meaning of the instruction is "Input an 8-bit data from the input device whose address is available as a 2nd byte of the instruction and load it into accumulator".

It is a **two byte instruction**. The operation requires only **three machine cycles- OFMC, MRMC IORMC and totals 10 T states**. The addressing mode is direct addressing mode.

14. **OUT PORT:** This is an ALP statement. PORT is the symbolic name given to the 8-bit address of the output device available as the 2nd byte of instruction. The meaning of the instruction is "Output the content of accumulator to an output device whose address is available in the instruction as the 2nd byte of the instruction".

It is a **two byte instruction**. The operation requires only **three machine cycles- OFMC, MRMC IOWRMC and totals 10 T states**. The addressing mode is direct addressing mode.

15. **XCHG:** This is an ALP statement. This is a single byte instruction. The meaning of the instruction is "Exchanged the contents of (H, L) register pair with the contents of (D,E) pair".

It is a **one byte instruction**. The operation requires only **one machine cycles- OFMC and totals 4 T states**. The addressing mode is register addressing mode.

ARITHMETIC GROUP

The instructions of this group perform the arithmetic operations on the operands. Normally two operands are necessary for any arithmetic operation. One of the operand is always assumed to be available in accumulator. The other operand can be made available in one of the three locations:

A. In an internal general purpose register (r).
B. In a memory location pointed by M-pointer i.e., (H, L) pair.
C. Immediately in the instruction itself as a 2nd byte.

All the flags of Flag register are affected as per standard rule. There are 20 basic instructions in this group.

1. **ADD r:** This is a single byte instruction. The meaning of the instruction is "Add the content of register (r) to the content of accumulator and store the result back into the accumulator".

It requires **single machine cycle OFMC and 4 states**. The addressing mode is **register addressing mode.**

2. **ADD M:** This is a single byte instruction. The meaning of the instruction is "Add the content of memory location whose address is in (H,L) pair to the content of accumulator and store the result back into the accumulator".

It requires **two machine cycles OFMC & MRMC and 7 states**. **Register indirect addressing** mode is used in this instruction.

3. **ADI DATA:** It is a two byte instruction. The meaning of the instruction is "Add the content available as the second byte of the instruction to the content of accumulation and store the result back into the accumulator".

It requires **two machine cycles OFMC & MRMC and 7 states**. It has **immediate addressing** mode.

4. **ADC r**: It is a single byte instruction. The meaning of the instruction is "Add the content of register (r) to the content of accumulator with carry and store the result back into accumulator".

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

5. **ADC M:** It is a single byte instruction. The meaning of the instruction is "Add the contents of memory location whose address is available in (H, L) pair to the contents of accumulator with carry and store the result back into accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. The addressing mode is register indirect addressing mode.

6. **ACI DATA:** It is a two byte instruction. The meaning of the instruction is "Add the content available as the second byte of instruction itself to the content to accumulator with carry and store the result back into the accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. It has immediate addressing mode.

7. **SUB r:** This is a single byte instruction. The meaning of the instruction is "Subtract the content of register (r) from the content of accumulator and store the result back into the accumulator".

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

8. **SUB M:** This is a single byte instruction. The meaning of the instruction is "Subtract the content of memory location whose address is in (H,L) pair from the content of accumulator and store the result back into the accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. Register indirect addressing mode is used in this instruction.

9. **SUI DATA:** It is a two byte instruction. The meaning of the instruction is "Subtract the content available as the second byte of the instruction from the content of accumulation and store the result back into the accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. It has immediate addressing mode.

10. **SBB r**: It is a single byte instruction. The meaning of the instruction is "Subtract the content of register (r) from the content of accumulator with borrow and store the result back into accumulator".

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

11. **SBB M:** It is a single byte instruction. The meaning of the instruction is "Subtract the content of memory location whose address is available in (H, L) pair from the content of accumulator with borrow and store the result back into accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. The addressing mode is register indirect addressing mode.

12. **SBI DATA:** It is a two byte instruction. The meaning of the instruction is "Subtract the content available as the second byte of instruction itself from the content to accumulator with borrow and store the result back into the accumulator".

It requires two machine cycles OFMC & MRMC and 7 states. It has immediate addressing mode.

13. **INR r:** It is a single byte instruction. The meaning of the instruction is "Increment the content of register (r) by 1 and store the result back to the register (r)".

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode. All flags are affected except CY flag.

14. **INR M:** It is a single byte instruction. The meaning of the instruction is "Increment the content of memory location by 1 whose address is available in (H, L) pair and stores the result back in the same location".

It requires three machine cycles OFMC, MRMC & MWRMC, and 10 states. The addressing mode is register addressing mode. All flags are affected except CY flag.

15. **DCR r:** It is a single byte instruction. The meaning of the instruction is “Decrement the content of register (r) by 1 and store the result back to the register (r)”.

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode. All flags are affected except the CY flag.

16. **DCR M:** It is a single byte instruction. The meaning of the instruction is “Decrement the content of memory location by 1 whose address is available in (H, L) pair and stores the result back in the same location”.

It requires three machine cycles OFMC, MRMC & MWRMC, and 10 states. The addressing mode is register addressing mode. All flags are affected except CY.

17. **INX rp:** It is a single byte instruction. The meaning of the instruction is “Increment the content of register pair (rp) by 1 and store it in same register pair.

The instruction requires only one machine cycle OFMC and of 6 states. No flag is affected in this instruction. It is register addressing mode.

18. **DCX rp:** It is a single byte instruction. The meaning of the instruction is “Decrement the content of register pair (rp) by 1 and store it in same register pair.

The instruction requires only one machine cycle OFMC and of 6 states. No flag is affected in this instruction. It is register addressing mode.

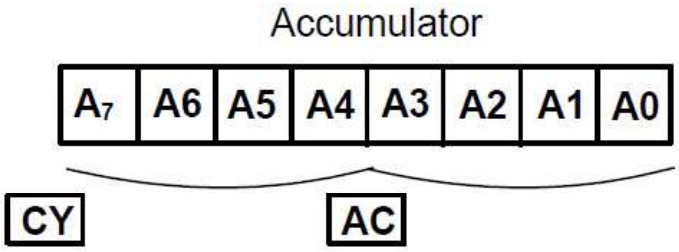
19. **DAD rp:** It is a single byte instruction. The meaning of the instruction is double precision addition, i.e., “Add the contents of (H,L) register pair with the contents of register pair (rp) and store the result back into (H,L) register pair”.

Only CY flag is affected in this instruction. It is register addressing mode. Since the instruction is a single byte instruction, therefore, only one memory reference operation is required. To add two 16-bit numbers, two 8-bit operations are required one by one which need two machine cycles. These operations cannot be completed during FEO (where only two extra states are available) and, therefore, two Bus Idle Machine Cycles (BIMC) are required.

The instruction requires three machine cycles OFMC & two BIMC and a total of 10 states.

20. **DAA:** It is a single byte instruction. The meaning of the instruction is decimal adjust accumulator. It is specifically used for BCD addition operation. It adjusts the content of

accumulator to two digit BCD. It performs the following operations on 8-bit data in the accumulator.



CY Flag, AC Flag and Both Nibbles checked for DAA Instruction

This instruction is used just after the addition operation. Whenever DAA is executed the following checks and corrections are made:

A. If the lower order 4-bits *A3A2A1A0* is an illegal BCD code or if AC is SET then 06H is added to accumulator.

B. Thereafter, if the higher order 4-bits *A7A6A5A4* is an illegal BCD code or if CY is SET then 60 H is added to accumulator else no action.

The instruction requires only one machine cycle OFMC and a total of 4 states.

LOGICAL GROUP

It consists of 19 basic instructions. There are three basic logic operation AND, OR & XOR. Logical operation takes place bit by bit. If two operands are involved in the instruction, the first operand is assumed to be in the accumulator. The second 8-bit operand is available either in the internal general purpose register or in the memory location pointed to by the M-pointer or as the 2nd byte of the instruction itself. The result of the logical operation is stored back in the accumulator.

1. **ANA r:** This is a single byte instruction. The meaning of the instruction is "AND bit by bit the content of register (r) to the content of accumulator and store the result back in the accumulator".

In this AND operation all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set.

It requires single machine cycle OFMC of 4 states. It has register addressing mode.

2. **ANA M:** This is a single byte instruction. The meaning of the instruction is "AND bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator".

In this instruction also, all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set.

It requires two machine cycles OFMC & MRMC and a total of 4 states. It makes use of register indirect addressing mode

3. **ANI DATA:** It is a two byte instruction. The meaning of the instruction is "AND bit by bit the content available as a second byte of instruction to the content of the accumulator and store the result back in the accumulator".

In this AND immediate instruction all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set.

It requires two machine cycles OFMC & MRMC and a total of 7 states.

4. **XRA r:** It is a single byte instruction. The meaning of the instruction is "Exclusively ORed bit by bit the content of register (r) with the content of accumulator and store the result back into accumulator".

It has seven variations and the addressing mode is register addressing mode. Only one Machine cycle is required i.e. OFMC of 4 states.

5. **XRA M:** (Exclusive OR memory) This is a single byte instruction. The meaning of the instruction is “Exclusively ORed bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator”.

The addressing mode is register indirect addressing mode. It requires two machine cycles OFMC & MRMC and a total of 7 states.

6. **XRI data:**(Exclusive OR immediate) It is a two byte instruction. The meaning of the instruction is “Exclusively ORed bit by bit the content of second byte of the instruction with the content of accumulator and store the result back into accumulator”.

The addressing mode is immediate addressing mode. It needs two machine cycles OFMC & MRMC and a total of 7 states.

Note:

A. In all exclusive–OR instructions, all the flags are affected as per standard rule expect AC & CY flags are cleared.

B. There is no explicit instruction to clear the accumulator to „zero". However, it is implicit is the statement XRA A.

7. **ORA r:** It is a single byte instruction. The meaning of the instruction is “ORed bit by bit the content of register (r) with the content of accumulator and store the result back into accumulator”.

It has seven variations and the addressing mode is register addressing mode. Only one Machine cycle is required i.e. OFMC of 4 states. All the flags are affected except AC& CY flags are cleared.

8. **ORA M:** (OR memory) This is a single byte instruction. The meaning of the instruction is “ORed bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator”.

The addressing mode is register indirect addressing mode. It requires two machine cycles OFMC & MRMC and a total of 7 states. All the flags are affected except AC& CY flags are cleared.

9. **ORI data:**(OR immediate) It is a two byte instruction. The meaning of the instruction is “ORed bit by bit the content of second byte of the instruction with the content of accumulator and store the result back into accumulator”.

The addressing mode is immediate addressing mode. It needs two machine cycles OFMC & MRMC and a total of 7 states. All the flags are affected except AC& CY flags are cleared.

10. **CMP r:** (Compare register with ACC) It is a single byte instruction. The meaning of the instruction "Compare the content of accumulator with the content of register (r)."

In this instruction, the content of register (r) is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc & register (r) are not destroyed but as result of this operation the flags are affected as per standard rule as given below:

(A) > (r)	CY = 0, Z = 0
(A) = (r)	CY = 0, Z = 1
(A) < (r)	CY = 1, Z = 0

The instruction needs one machine cycle OFMC of 4 states. The addressing mode is register addressing mode.

11. **CMP M:** (Compare memory with accumulator) It is a single byte instruction. The meaning of the instruction is "Compare the content of accumulator with the content of memory location whose address is available in (H, L) register pair".

In this instruction, the content of memory location pointed to by (H, L) register pair is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc & memory are not destroyed but as result of this operation the flags are affected as per standard rule as given below:

(A) > (r)	CY = 0, Z = 0
(A) = (r)	CY = 0, Z = 1
(A) < (r)	CY = 1, Z = 0

The instruction requires two machine cycles OFMC & MRMC and a total of 7 states. The addressing mode is register indirect addressing mode.

12. **CPI data:** (Compare accumulator with immediate data) This is an ALP statement. The meaning of the instruction is "Compare the content of accumulator with the 8-bit data available in the instructions immediate data byte".

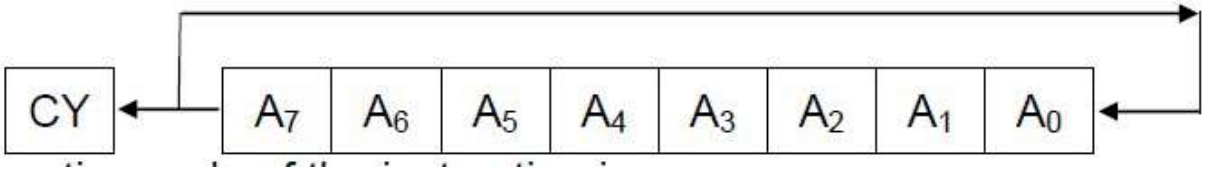
In this instruction, the 8-bit data available in the instruction as 2nd byte is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc is not destroyed but as result of this operation the flags are affected as per standard rule as given below:

(A) > (r)	CY = 0, Z = 0
(A) = (r)	CY = 0, Z = 1
(A) < (r)	CY = 1, Z = 0

The instruction requires two machine cycles OFMC & MRMC and a total of 7 states. The addressing mode is immediate addressing mode.

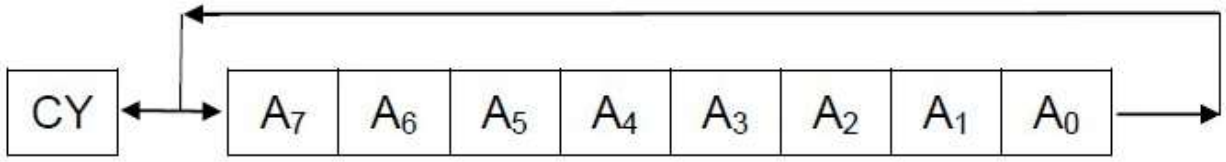
13. RLC: This is an ALP statement. The meaning of the instruction is "Rotate left the content of the accumulator by one bit without carry flag. The lower order bit & the CY flag are both set to the value shifted out the high order bit position".

In this instruction it is assumed that the operand is available in accumulator.



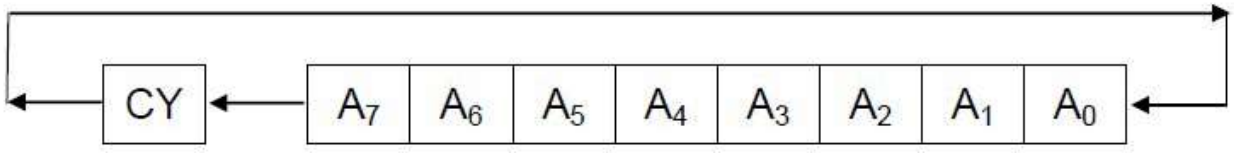
The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

14. RRC: This is an ALP statement. The meaning of the instruction is "Rotate right the content of the accumulator by one bit without carry flag. The higher order bit & the CY flag are both set to the value shifted out the lower order bit position". In this instruction also it is assumed that the operand is available in accumulator.



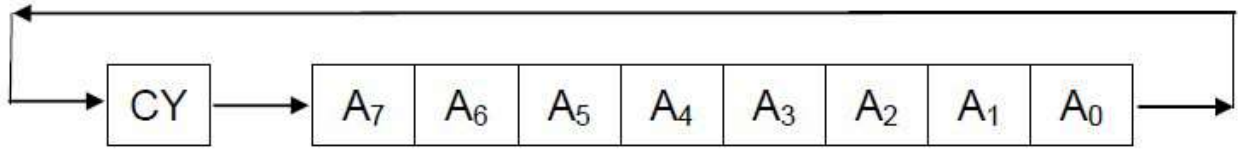
The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

15. RAL: (Rotate left through carry) This is an ALP statement. The meaning of the instruction is "Rotate left the content of the accumulator by one bit through the carry flag. The lower order bit set equal to the CY flag and CY flag is set to the value shifted out of the A_7 bit.". In this instruction it is assumed that the operand is available in accumulator.



The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

16. **RAR:** (Rotate accumulator right through carry) This is an ALP statement. The meaning of the instruction is “Rotate right the content of the accumulator by one bit through carry flag. The higher order bit is set equal to the CY flag and the CY flag is set to the value shifted out the lower order bit position”. The operand is assumed to be available in accumulator.



The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

17. **CMA:** (Complement accumulator) This is an ALP statement. The meaning of the instruction is “Complement the content of accumulator bit by bit and store the result back into the accumulator”.

No flag is affected in this instruction. It is a single byte instruction. The addressing mode is implied addressing mode.

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

18. **CMC:** (Complement carry flag) This is an ALP statement. The meaning of the instruction is “Complement CY flag and store it back into the same CY flag position”.

The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. This instruction may be used after STC to clear carry without affecting any other flag and register. Only the CY flag is affected in this operation.

19. **STC:** (Set carry flag) This is an ALP statement. The meaning of the instruction is “Set the CY flag to „1".

The addressing mode is implied addressing mode. The instruction requires one machine cycle OFMC and a total of 4 states. Only the CY flag is affected in this operation.

BRANCH GROUP

This group of instructions is used to alter the normal sequential program flow and force the program to proceed from a different point.
Branch instructions can be of two types: conditional & unconditional.
Unconditional branch instructions simply cause the program to branch to the indicated instruction whenever these instructions are encountered, i.e., (PC) is loaded with a new address. Conditional branch instructions examine the status of one of the four processor flags (Z, CY, P, S) to determine if the specified branch instruction is to be executed. If the condition tested is TRUE, it causes a branching to occur otherwise not. AC is not used for specifying condition. The 8 conditions that are tested are given below:

Condition		Flag Tested
NZ	Not zero	Z=0
Z	Zero	Z=1
NC	Not carry	CY=0
C	Carry	Cy=1
PO	Parity odd	P=0
PE	Parity even	P=1
P	Plus	S=0
M	Minus	S=1

If Z = 0 is not true, then NZ is true. If Z = 0 is true, and then NZ is not true. There are 8 basic operations in this group:

1. **JMP ADDR:** The meaning of the instruction is "Load the PC with the 16- bit data available in the instruction itself as the 2nd and 3rd bytes of instruction so that the next instruction is fetched from this address in the succeeding instruction cycle.

This is a **3 –byte instruction.** It requires **three machine cycle OFMC, MRMC & MRMC** and total **10 states**. The addressing mode used in this instruction is **immediate addressing mode** because the 16-bit address data is immediately available in the instruction itself to be loaded into the PC.

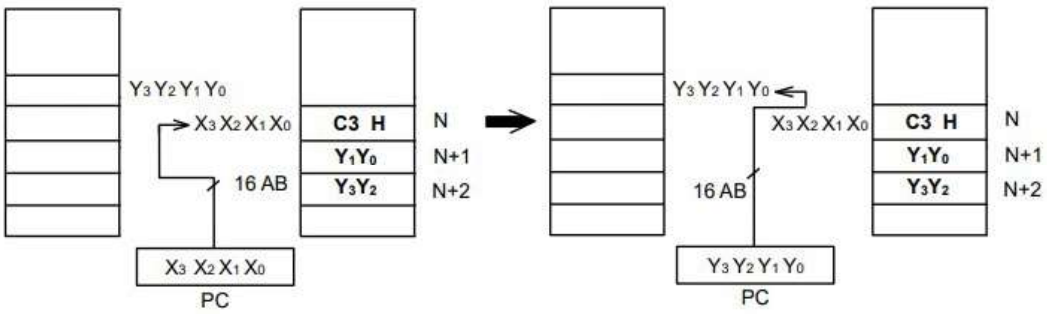


Fig.5.19 Memory Conditions Before and After JMP ADDR Instruction

2. **Jcond ADDR:** This is a 3-byte conditional jump instruction. There are 8 conditions that can be checked. Accordingly, there are 8-variations for this instruction depending upon the conditions to be tested. They are JNZ, JZ, JNC, JC, JPO, JPE, JP, and JM. The meaning of the instruction is illustrated in the flow chart

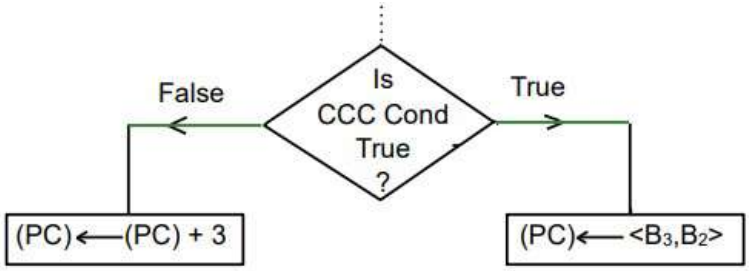


Fig.5.20 Flow Chart for Conditional Jump Instruction

where (PC) = (X3X2X1X0)H just at the start of instruction exist.

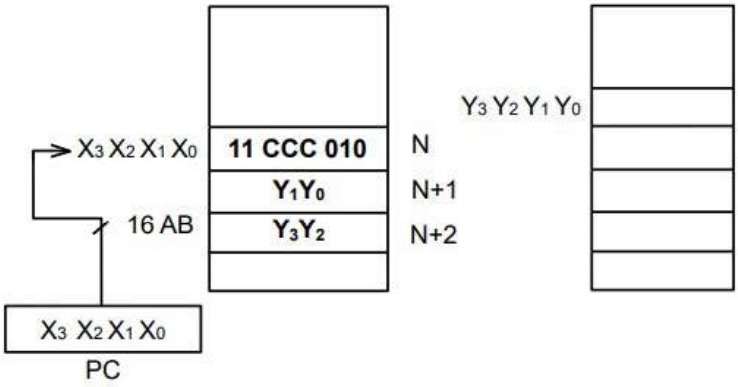


Fig.5.21 Memory Conditions Before Execution of Jcond Instruction

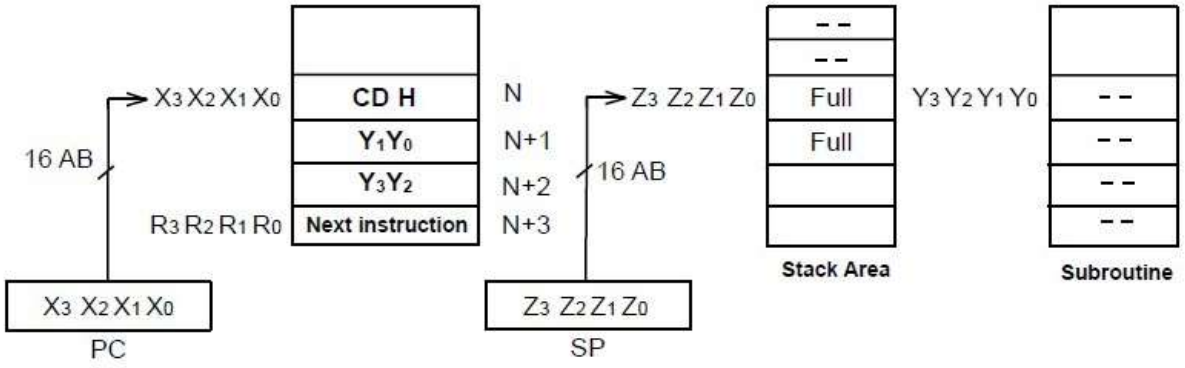
The I6-bit address in the (PC) just at the end of the instruction depends upon the condition to be tested. (PC) will be loaded with B3B2 if the given condition is TRUE, otherwise (PC) will go to (PC)i+3 where (PC)i shall be the address X3X2X1X0

If the condition is TRUE, it takes three machine cycles OFMC, MRMC & MRMC to get the instruction executed and a total of 10 states. If the condition is not TRUE then it takes two machine cycles OFMC, MRMC and a total of 7 states. The addressing mode used in this

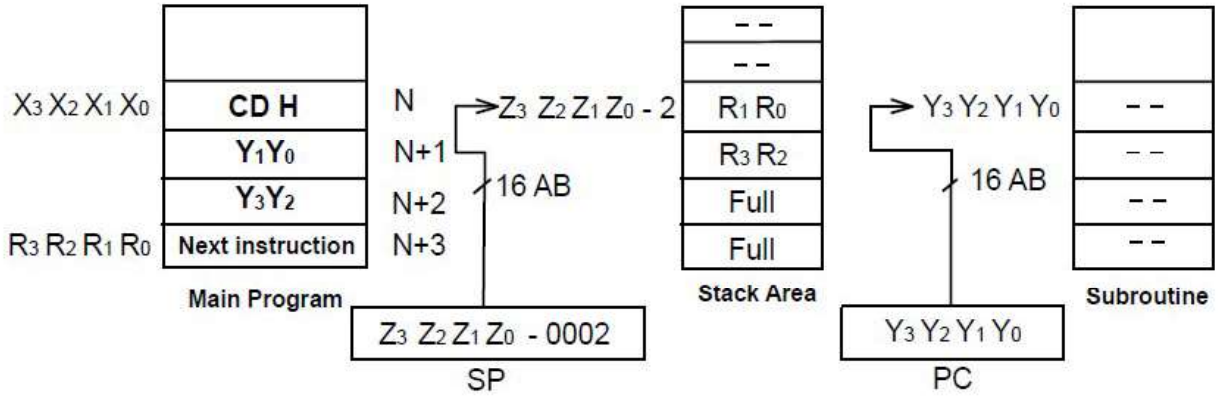
instruction is immediate addressing mode because the 16-bit address data is immediately available in the instruction itself to be loaded into the PC. No flag is affected.

CALL ADDR: (Unconditional subroutine call) This is an ALP statement. ADDR is the symbolic name given to the 16-bit address available as the 2nd and 3rd byte of the instruction. This is a 3-byte instruction.

"Save the return address on the top of the stack and thereafter, load the (PC) with the 16-bit address immediately available in the instruction as 2nd and 3rd bytes".



After Execution



it requires five machine cycles OFMC, two MRMC & two MWRMC and total 18 states, the longest instruction cycle in 8085A processor.

Ccond ADDR (Unconditional subroutine call): This is an ALP statement. This is also 3-byte instruction. 2nd and 3rd bytes give the address of the subroutine. When this instruction is executed, the μp jump to the subroutine if the condition tested is TRUE. If the condition is not TRUE then μp goes to execute the next instruction.

If condition is TRUE, it requires five machine cycles OFMC, two MRMC & two MWRMC and total 18 states otherwise only two machine cycles OFMC & one MRMC and total of 9 states are required.

RET (Unconditional Return): This is an ALP statement and stands for RETURN. The meaning of the instruction is "Return to the main program from the subroutine unconditionally".
This is a single byte instruction.

When this instruction is executed the 16-bit address data available at the top of the stack is loaded into the (PC) and stack pointer is readjusted so that it again points to the top of the stack.
The content of the memory location whose address is specified in register (SP) is moved to the lower order 8-bits of program counter (PCL).
The content of the memory location whose address is one more than the content of register (SP) is moved to the higher order 8-bits of program counter (PCH).
In this process the stack goes down and the content of the register (SP) is incremented by 2. It is for the user to ensure that the proper RETURN ADDR is available correctly on the top of the stack before asking the processor to execute the RET instruction.

The macro RTL implemented is,

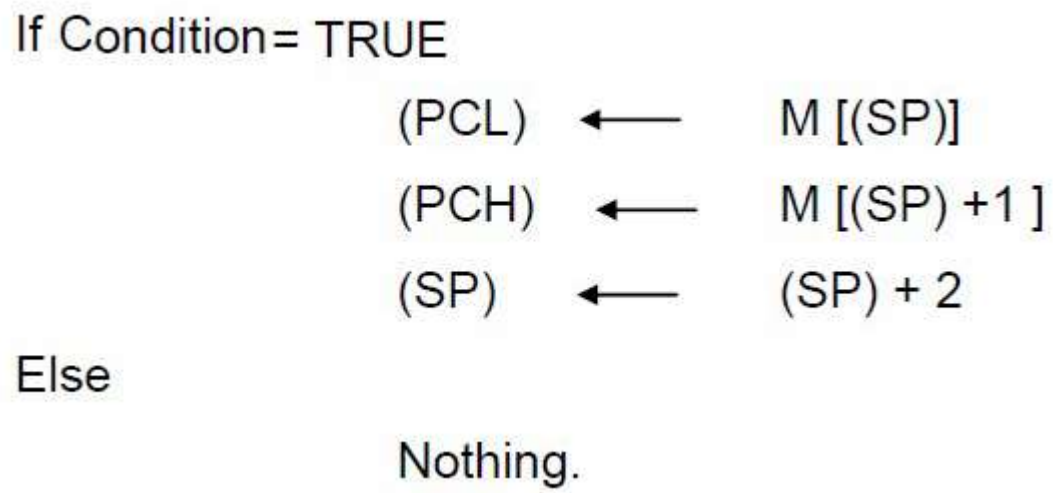
$$(PCL) \longleftarrow M[(SP)]$$
$$(PCH) \longleftarrow M[(SP) + 1]$$
$$(SP) \longleftarrow (SP)+2$$

It requires 3 machine cycles and 10 states. For 2MH3 internal clock the time required is 5μsec. The addressing mode is register indirect addressing mode as the return address is to be read from top of the stack pointed to by (SP).

Rcond: This is a conditional return statement. It is also a part of the subroutine. Whenever this instruction is executed, *μp* checks the conditional flags. If the condition is found true then the *μp* returns to the main program by loading the (PC) with the return address stored at the top of the stack. If the condition is found not true then the next instruction of the subroutine will be executed.

This is a single byte instruction.

There are 8 variations in this statement. They are RNZ, RZ, RNC, RC, RPO, RPE, RP and RM. The addressing mode is register indirect because the return address is available in the memory location pointed by SP.



If the condition tested is TRUE, then the 8-bit data (lower order 8-bit return address) from the top of the stack pointed by (SP) is moved to (PCL). The 8-bit data (higher order 8-bit return address) from next higher memory location whose address is one more than the content of (SP) is moved to
(PCH). The content of the register (SP) is incremented by 2 in this process so that it always point to the top of the stack. Control is transferred back to the main program. If the condition is not found TRUE, the control goes to the next instruction of subroutine in sequence.

If condition is TRUE it requires 3 machine cycles and 12 states otherwise one machine cycle OFMC and 6 states.

Stack Group: This group of instructions manipulates the stack. Unless otherwise specified, condition flags are not affected by any instruction of this group. It is the user responsibility to define stack area and to initialize the stack pointer (SP) with the bottom address before these instructions are used.

1. **PUSH rp:** This is a single byte instruction. The meaning of the instruction is "Push or save the contents of register pair (rpH, rpL) on top of the stack.

The content of the higher order register of register pair (rp) is moved to the memory location whose address is one less than the content of register (SP). The content of the lower order register of register pair (rp) is moved to the memory location whose address is two less than the content of register SP. In this process, the content of the (SP) register is decremented by 2.

This instruction has 3 variations for register pair (B, C), (D, E) & (H, L) for three combinations of RP. The addressing mode is register indirect addressing mode. Thus, it requires three machine cycles OFMC, two MWRMC and a total 12 states

2. **PUSH PSW (Push Processor Status Word):** This is also a single byte instruction. The meaning of the instruction is "Save or push the processor status word, comprising of (A) and (F) register, on the top of the stack". The accumulator & flag register together form a 16- bit word known as the processor status word (PSW), ACC occupies the higher order 8-bits and flag register occupies the lower order 8-bits in PSW.

The content of the accumulator (A) is moved to the memory location whose address is one less than the content of register (SP). The content of flag register (F) is moved to the memory location whose address is two less than the content of register SP. In this process, the content of the (SP) register is decremented by 2. The addressing mode is register indirect addressing mode. It requires three machine cycles OFMC, two MWRMC and a total 12 states.

3. **POP rp:** It is a single byte instruction. The meaning of the instruction is "Pop or load the content form the top of the stack into register pair (rp)".

The content of the memory location pointed by the content of stack pointer (SP) is moved to higher order register of register pair (rp) and the content of the next memory location whose address is one more than the content of (SP) is moved to lower order register of register pair (rp). In this process, the content of the (SP) register is incremented by 2. This instruction has 3 variations for register pair (B, C), (D, E) & (H, L) for three combinations of RP. It requires three machine cycles OFMC, two MRMC and a total 10 states

4. **POP PSW: (Pop Processor Status Word)** It is a single byte instruction. The meaning of the instruction is "Pop or load the content form the top of the stack into processor status word comprising of (A) and (F) register". The accumulator & flag register together form a 16- bit word known as the processor status word (PSW), ACC occupies the higher order 8-bits and flag register occupies the lower order 8- bits in PSW.

The content of the memory location pointed by the content of stack pointer (SP) is moved to accumulator (A) and the content of the next memory location whose address is one more than the content of (SP) is moved to flag register (F).
The addressing mode is register indirect addressing mode. It requires three machine cycles OFMC, two MRMC and a total 10 states. Flags affected are Z, S, AC, P, and CY.

5. **XTHL:** It is a single byte instruction. The meaning of the instruction is "Exchange the contents from the top of the stack with the contents of (H, L) register pair". The content of register (L) is exchanged with the content of the memory location whose address is specified

by the content of (SP). The content of register (H) is exchanged with content the memory location whose address is one more than the content of (SP)

The addressing mode is register indirect addressing mode none flags affected. It requires five machine cycles OFMC, two MRMC & two MWRMC and a total 16 states.

6. **SPHL:** It is a single byte instruction. The meaning of the instruction is "Move the contents of register pair (H, L) to 16-bit register stack pointer (SP). The addressing mode is register addressing mode. None of the flags are affected. It requires one machine cycle OFMC and 6 states.

7. PCHL: This is a single byte instruction. The meaning of the instruction is "Move the content of register (H) to the higher byte of program counter (PCH) and move the content of register (L) to lower byte of program counter (PCL)".

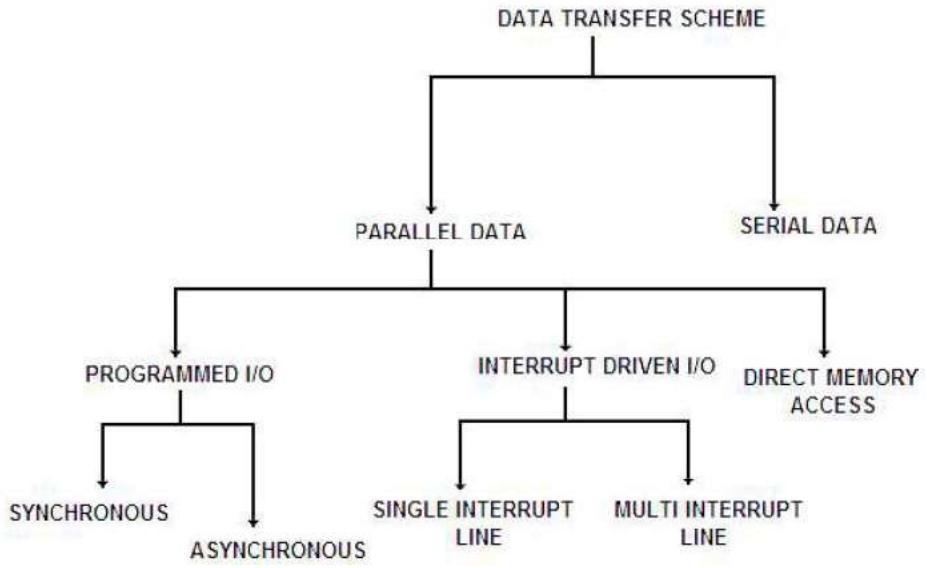
This instruction uses register addressing mode.

Data Transfer schemes

- In microprocessor based systems several input / output devices are connected and **data transfer may take place between microprocessor and memory, microprocessor and I/O devices and memory & I/O devices.**

- For the data communication between microprocessor and memory not much problems arise as **same technology is used in the manufacturing of memory and microprocessor.** The speed of the memory is almost compatible with the speed of microprocessor.

- For the data transfer between the microprocessor and I/O devices, the problems arise mainly **due to mismatch of the speed of the I/O devices and the speed of microprocessor or memory.** To overcome the problem of speed mismatch between the microprocessor and I/O devices following data transfer schemes may be considered.



- These data transfer schemes are categorized depending upon the **capabilities of I/O devices for accepting serial or parallel data.**

- **The 8085 microprocessor is a parallel device** i.e. it transfers eight bits of data simultaneously over eight data lines (parallel I/O mode). However in many situations, the parallel I/O mode is either impractical or impossible. For example, **parallel data communication over a long distance becomes very expensive.**

- For device which accept serial data, serial I/O mode is used which transfer a single bit on a single line at a time**. For serial data transmission, 8-bit parallel word is converted to a stream of eight serial bit using parallel-to-serial conversion.**

- In serial reception of data, the microprocessor receives a stream of 8-bit one by one which are then converted to 8-bit parallel word using serial-to-parallel conversion

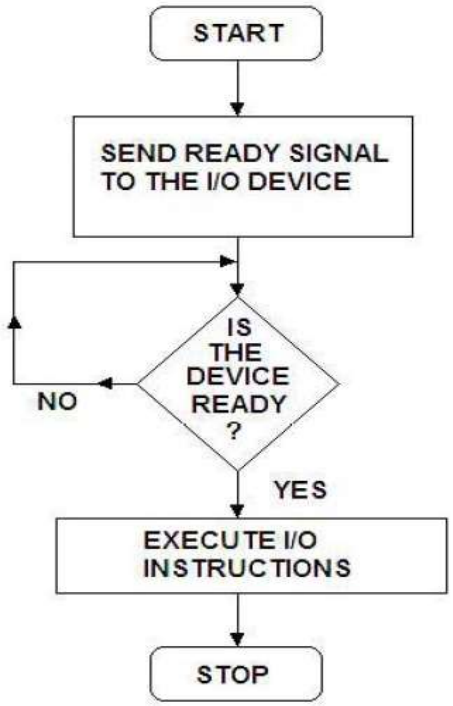
Programmed I/O Data Transfer

This method of data transfer is generally used in the **simple microprocessor systems**. This method **uses instructions to get the data into or out of the microprocessor**. The data transfer can be synchronous or asynchronous depending upon the type and the speed of the I/O devices.

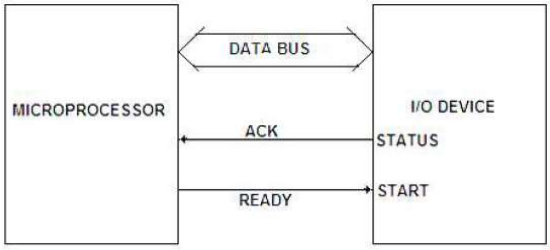
- **Synchronous data transfer:** It can be used when the **speed of the I/O devices matches with the speed of the microprocessor.** Common clock pulse is used to synchronize the microprocessor and the I/O devices. In such data transfer scheme because of the matching of the speed, the microprocessor does not have to wait for the availability of the data; the microprocessor immediately sends data for the transfer as soon as the microprocessor issues a signal.

- **Asynchronous data transfer** This method is used when the **speed of the I/O devices is slower than the speed of the microprocessor. The asynchronous data transfer is normally implemented using 'handshaking' mode.** In the handshaking mode some signals are exchanged between the I/O device and microprocessor before the data transfer takes place.

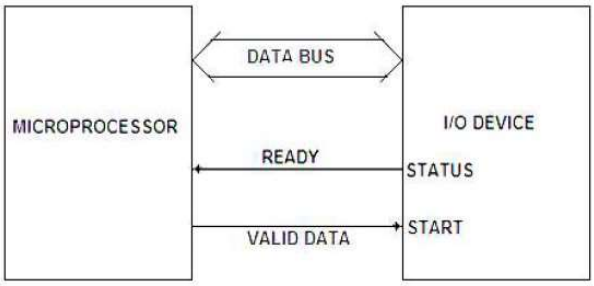
- The microprocessor checks the status to the input/output device, if the device is ready for the data transfer. It sends instructions to transfer the data. Flow chart for this mode of data transfer is shown in figure.



Asynchronous Handshaking Process



To transfer the data from the microprocessor to I/O device. Microprocessor sends ready signal to I/O device. When the device is ready to accept the data, the I/O device sends an 'ACK' (Acknowledge) signal to microprocessor indicating that the I/O device has acknowledged the 'Ready' signal and is ready for the transfer of data.



To transfer the data from the I/O device to microprocessor. I/O device issues the ready signal to microprocessor indicating that I/O device is ready to send the data to microprocessor. In response to this signal, valid data signal is sent by the microprocessor to I/O device and then the valid data is put on the data bus for the transfer.

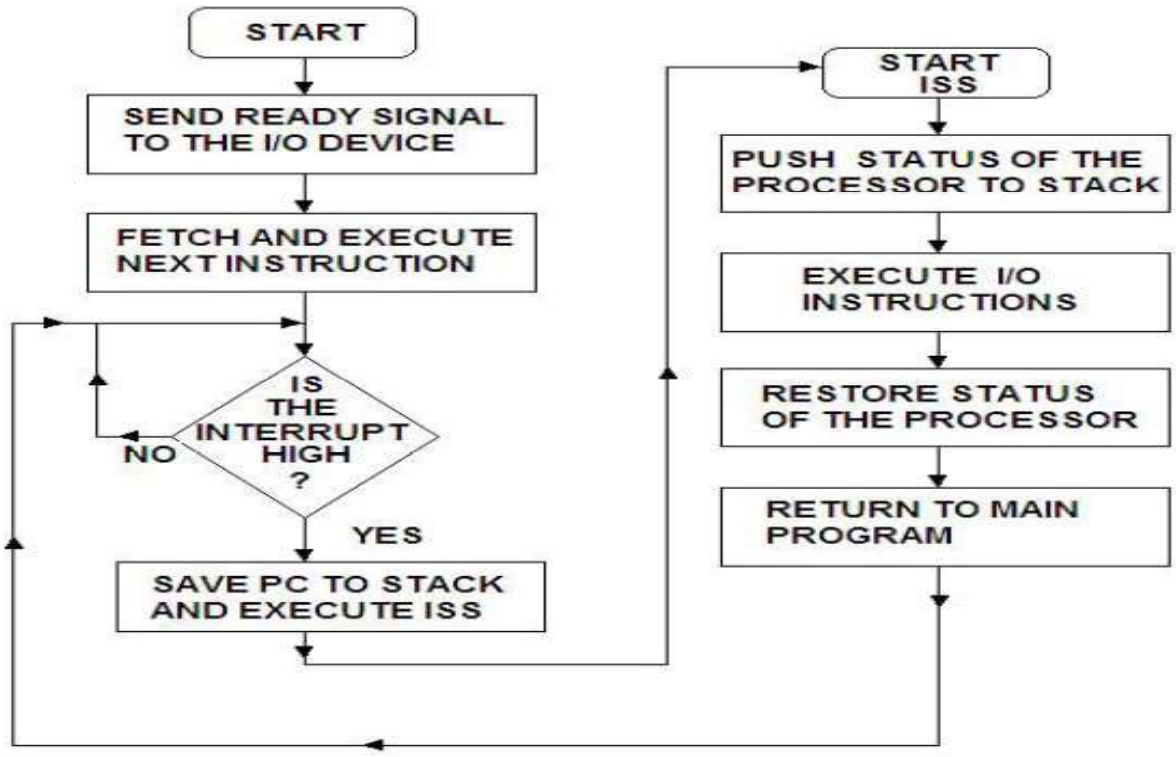
Interrupt Driven I/O Data Transfer

- In the programmed I/O data transfer method, the microprocessor is busy all the time in checking for the availability of data from the slower I/O devices and also in checking if I/O device is ready for the data transfer.
- The interrupt driven I/O data transfer method is efficient as no time is wasted in waiting for an I/O device to be ready. The I/O device informs the microprocessor for the data transfer whenever the I/O device is ready. This is achieved by interrupting the microprocessor.

Steps for data transfer:

1. The microprocessor initiates data transfer by requesting the I/O device 'to get ready' and then continue executing its original program rather wasting its time by checking the status of I/O device.
2. Whenever the device is ready to accept or supply data, it informs the processor through a control signal known as interrupt signal. In response to this interrupt signal, the microprocessor sends back an interrupt acknowledge signal to the I/O device indicating that it received the request. It then suspends its job after executing the current instruction and jumps to the subroutine program. This subroutine program is called Interrupt Service Subroutine (ISS) program. The ISS saves the processor status into stack; and after executing the instruction for the data transfer, it restores the processor status and then returns to main program.

The flow chart for this method of data transfer is shown in figure.



Since several input/output devices may be connected to microprocessor using Interrupt Driven Data Transfer Scheme. Following interrupt request configuration may arise while interfacing the I/O devices to microprocessor.

1. Single Interrupt system
2. Multi Interrupt System

1. Single Interrupt System

When only one interrupt line is available with the microprocessor and several I/O devices are to be connected, then the method is known as Single Interrupt System.

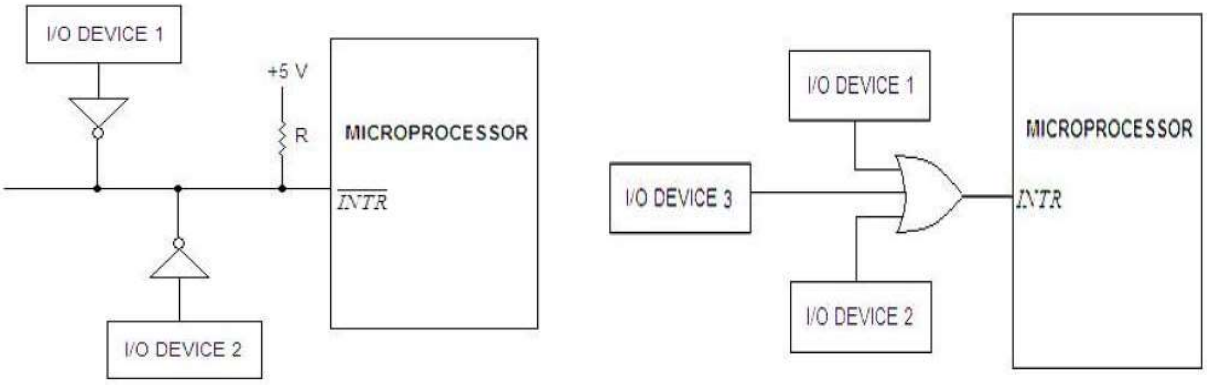


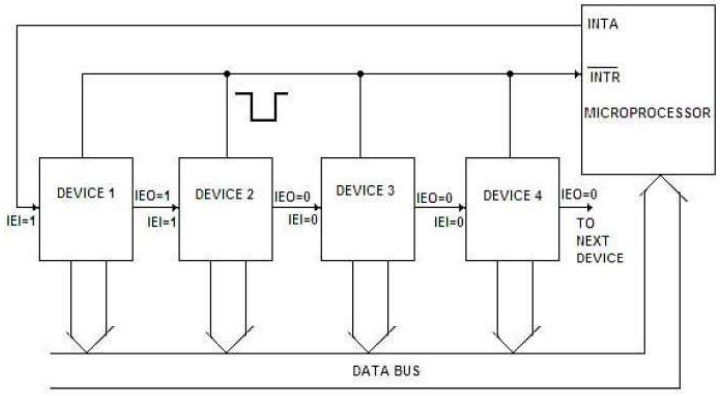
Figure1 shows the way to connect several devices to one active low input interrupt terminal ($\overline{INTR}$) of the microprocessor. In the active low interrupt line of the microprocessor the devices are connected to $\overline{INTR}$ terminal through different open collector NOT gates; when any of the devices is active it provides a low signal to $\overline{INTR}$ enabling the interrupt line.

To connect the several I/O devices to active high interrupt terminal (INTR) is shown in figure2. In the active high interrupt line the I/O devices are connected through an OR gate. When any of the device is high the output of OR gate sends a high signal to interrupt line (INTR).

When the interrupt line is active in either of the two methods discussed above, then the microprocessor will not know which device has sent the interrupt signal. Three techniques are commonly used to solve the problem for the microprocessor that is requesting the interrupt and resolving any simultaneous requests by two or more devices. These are:

1. Polling: The interrupt signal from each device can be used to set one bit of a register wired as an input port. When an interrupt occurs, the ISS polls this port to see who requested service. A priority is automatically established by the order of polling. This technique is very simple but has the negative of degrading response time.

2. Daisy Chain: This technique has been shown in figure 3. In this method each I/O device has an Interrupt Enable Input (IEI) and an Interrupt Enable Output (IEO). An interrupt request can be made only if IEI is high. A serial connection like a chain of all the I/O devices is made. The highest priority I/O device is placed at the first position followed by the lower priority devices in sequence. If any device sends an interrupt signal i.e. low to the interrupt line, then the INTR line of the processor is enabled. The interrupt acknowledge signal (INTA) is enabled in response to low INTR line. In figure, device 2 is requesting an interrupt causing its IEO to be low. This in turn disables devices 3 and 4. It may be noted that device 1 is still able to request an interrupt because it has a higher priority. The interrupt acknowledge signal can be used to reset IEO of the interrupting device.



3. Priority Interrupt Controller (PIC): In this method several I/O devices may be connected to a single interrupt line through programmable interrupt controller (IC 8259). Up to 8 input/output devices may be connected to the microprocessor. If more than 8 I/O devices to be connected, more PICs (programmable interrupt controllers) are used in cascade. The details of PIC will be discussed later.

Multi Interrupt System

When the microprocessor has several interrupt terminals and one I/O device is to be connected to each interrupt terminal, then it is known as multi interrupt system. In this scheme, the number of I/O devices to be connected to the interrupt lines should be equal to or less than the number of interrupt terminals. In this way one device is connected to each level of interrupt. So when a device interrupts the microprocessor, it immediately knows which device has interrupted. Such an interrupt scheme is known as vectored interrupt.

Direct Memory Access (DMA) Data Transfer

In programmed I/O or interrupt driven I/O methods of data transfer between the I/O devices and external memory is via the microprocessor. For bulk data transfer from I/O devices to memory or vice-versa, these two methods discussed above are time consuming and quite uneconomical even though the speed of I/O devices matches with the speed of microprocessor since the data is first transferred to microprocessor and then to concerned device.

The Direct Memory Access (DMA) data transfer method is **used for bulk data transfer** from I/O devices to microprocessor or vice-versa. In this method I/O devices are allowed to transfer the data directly to the external memory without being routed through accumulator.

In this process the microprocessor relinquishes the control over the data bus and address bus, so that these can be used for transfer of data between the devices.

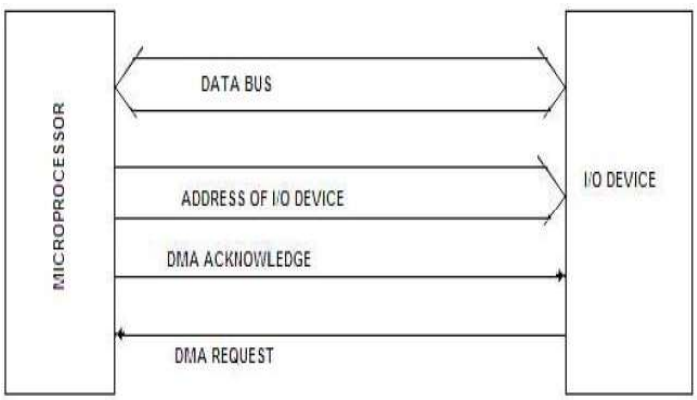
For transferring the data through DMA, an interfacing chip known as DMA Controller is used with the microprocessor that helps to generate the addresses for the data to be transferred from the I/O devices.

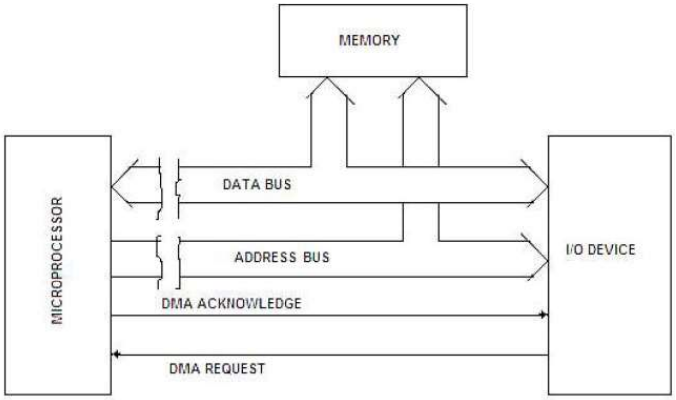
The peripheral device sends the request signal (DMARQ) to the DMA controller and the DMA controller in turn passes it to the microprocessor (HOLD signal).

On receipt of the DMA request the microprocessor sends an acknowledge signal (HLDA) to the DMA controller. On receipt of this signal (HLDA) the DMA controller sends a DMA acknowledge signal (DMACK) to the I/O device.

The DMA controller then takes over the control of the buses of microprocessor and controls the data transfer between RAM and I/O device.

When the data transfer is complete, DMA controller returns the control over the buses to the microprocessor by disabling the HOLD and DMACK signals.





Following types of Data transferred using DMA operation:

1. Memory to I/O device
2. I/O device to memory
3. Memory to memory
4. I/O device to I/O device