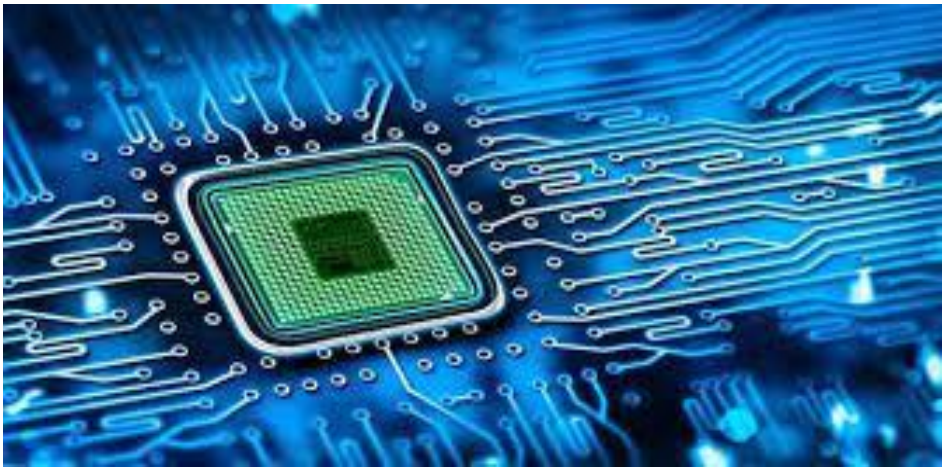
BHADRAK ENGINEERING SCHOOL & TECHNOLOGY
(BEST), ASURALI, BHADRAK

Microprocessor & Micro controller

(Th- 03)

(As per the 2020-21 syllabus of the SCTE&VT, Bhubaneswar, Odisha)



Fourth Semester

Electronics & Tele-comm. Engg.

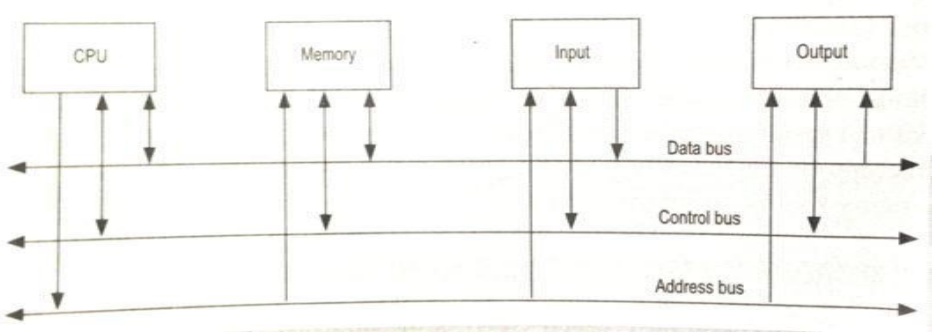
Prepared By: *Er B.R.Nayak*

CHAPTER-1.0

Microprococessor(Architecture and programming-8 bit-8085)

1.1-INTRODUCTION TO MICROPROCESSOR AND MICROCOMPUTER.

A *microprocessor* is a programmable electronics chip that has computing and decision making capabilities similar to central processing unit of a computer. Any microprocessor-based systems having limited number of resources are called *microcomputers*. Nowadays, microprocessor can be seen in almost all types of electronics devices like mobile phones, printers, washing machines etc. Microprocessors are also used in advanced applications like radars, satellites and flights. Due to the rapid advancements in electronic industry and large scale integration of devices results in a significant cost reduction and increase application of microprocessors and their derivatives.



- **Bit**: A bit is a single binary digit.
- **Word**: A word refers to the basic data size or bit size that can be processed by the arithmetic and logic unit of the processor. A 16-bit binary number is called a word in a 16-bit processor.
- **Bus**: A bus is a group of wires/lines that carry similar information.
- **Memory Word**: The number of bits that can be stored in a register or memory element is called a memory word.
- **Diffrence between microprocessor and micro computer**.

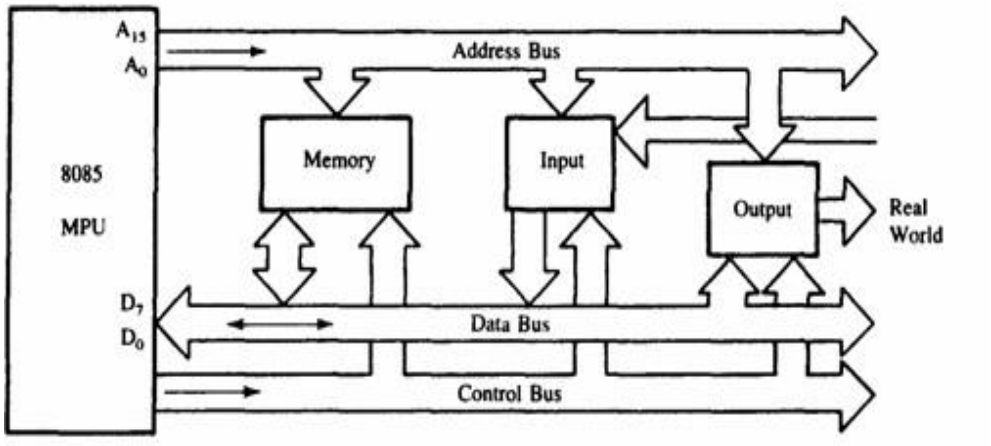
1-Microprocessor is one component of a microcomputer. **2**-Microprocessor is a programmable integrated cercit which has its own decission making capability. Example-8085,INTEL8086,8088,8008,8080 etc	**1**-A digital computer in which one microprocessor is used as a CPU known as microcomputer. **2**-Microcomputer usess a microprocessor for its processing operation . Example-Desktop,Laptop,Note Book etc.

- **1.2-Concept of Address Bus,Data Bus,Control Bus and System Bus**
- **Address Bus**: It carries the address, which is a unique binary pattern used to identify a memory location or an I/O port. For example, an eight bit address bus has eight lines and thus it can address 2^8 = 256 different locations. The locations in

hexadecimal format can be written as 00H – FFH.

- **Data Bus**: The data bus is used to transfer data between memory and processor or between I/O device and processor. For example, an 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have 16-bit data bus.
- **Control Bus**: The control bus carry control signals, which consists of signals for selection of memory or I/O device from the given address, direction of data transfer and synchronization of data transfer in case of slow devices.
- **System Bus**: The system bus is a group of wires/lines used for communication between the microprocessor and peripherals.

1.3-Block diagram of General bus structure.



1.4 8085(8 bit) MICROPROCESSOR ARCHITECTURE

The 8085 microprocessor is an 8-bit processor available as a 40-pin IC package and uses +5 V for power. It can run at a maximum frequency of 3 MHz. Its data bus width is 8-bit and address bus width is 16-bit, thus it can address 2^{16} = 64 KB of memory. The internal architecture of 8085 is shown is Fig. 2.

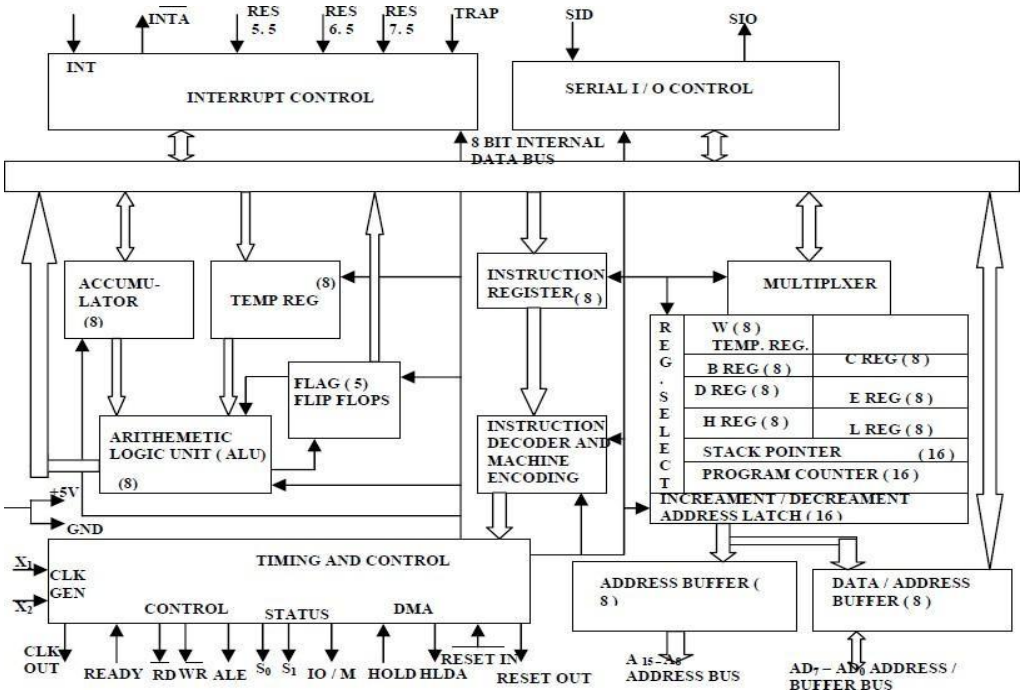


Fig. 2 Internal Architecture of 8085

Arithmetic and Logic Unit

The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Registers

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

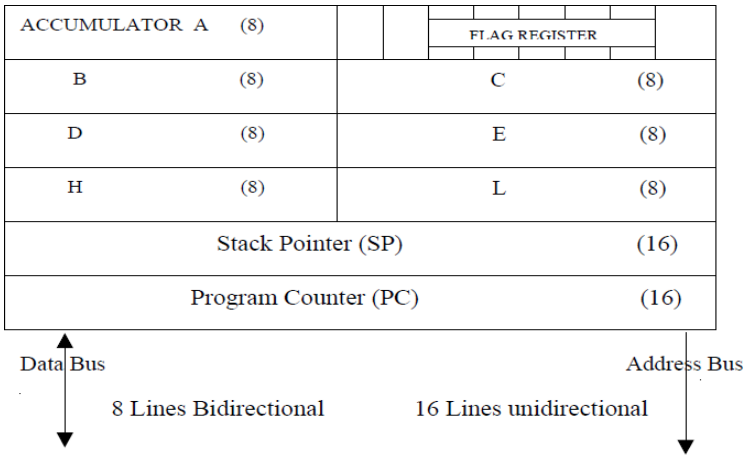


Fig. Register organisation

Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions.

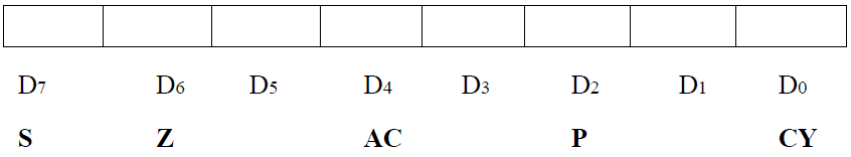


Fig.Flag register

For example, after an addition of two numbers, if the result in the accumulator is larger than 8-bit, the flip-flop uses to indicate a carry by setting CY flag to 1. When an arithmetic operation results in zero, Z flag is set to 1. The S flag is just a copy of the bit D7 of the accumulator. A negative number has a 1 in bit D7 and a positive number has a 0 in 2's complement representation. The AC flag is set to 1, when a carry result from bit D3 and passes to bit D4. The P flag is set to 1, when the result in accumulator contains even number of 1s.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register, used as a memory pointer. It points to a memory location in R/W memory, called stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register/Decoder

It is an 8-bit register that temporarily stores the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and decodes or interprets the instruction. Decoded instruction then passed to next stage.

Control Unit

Generates signals on data bus, address bus and control bus within microprocessor to carry out the instruction, which has been decoded. Typical buses and their timing are described as follows:

- *Data Bus*: Data bus carries data in binary form between microprocessor and other external units such as memory. It is used to transmit data i.e. information, results of arithmetic etc between memory and the microprocessor. Data bus is bidirectional in nature. The data bus width of 8085 microprocessor is 8-bit i.e. 2^8 combination of binary digits and are typically identified as D0 – D7. Thus size of the data bus determines what arithmetic can be done. If only 8-bit wide then largest number is 11111111 (255 in decimal). Therefore, larger numbers have to be broken down into chunks of 255. This slows microprocessor.
- *Address Bus*: The address bus carries addresses and is one way bus from microprocessor to the memory or other devices. 8085 microprocessor contain 16-bit address bus and are generally identified as A0 - A15. The higher order address lines (A8 – A15) are unidirectional and the lower order lines (A0 – A7) are multiplexed (time-shared) with the eight data bits (D0 – D7) and hence, they are bidirectional.
- *Control Bus*: Control bus are various lines which have specific functions for coordinating and controlling microprocessor operations. The control bus carries control signals partly unidirectional and partly bidirectional. The following control and status signals are used by 8085 processor:
    I. ALE (output): Address Latch Enable is a pulse that is provided when an address appears on the AD0 – AD7 lines, after which it becomes 0.

II. $\overline{RD}$ (active low output): The Read signal indicates that data are being read from the selected I/O or memory device and that they are available on the data bus.

III. $\overline{WR}$ (active low output): The Write signal indicates that data on the data bus are to be written into a selected memory or I/O location.

IV. IO/$\overline{M}$ (output): It is a signal that distinguished between a memory operation and an I/O operation. When IO/$\overline{M}$ = 0 it is a memory operation and IO/$\overline{M}$ = 1 it is an I/O operation.

V. S1 and S0 (output): These are status signals used to specify the type of operation being performed; they are listed in Table .

Table Status signals and associated operations

S1	S0	States
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

The schematic representation of the 8085 bus structure is as shown in Fig. The microprocessor performs primarily four operations:

I. Memory Read: Reads data (or instruction) from memory.
II. Memory Write: Writes data (or instruction) into memory.
III. I/O Read: Accepts data from input device.
IV. I/O Write: Sends data to output device.

1.5 PIN DESCRIPTION OF 8085 MICROPROCESSOR

Properties:

- It is a 8-bit microprocessor
- Manufactured with N-MOS technology
- 40 pin IC package
- It has 16-bit address bus and thus has 2^{16} = 64 KB addressing capability.
- Operate with 3 MHz single-phase clock
- +5 V single power supply

The logic pin layout and signal groups of the 8085nmicroprocessor are shown in Fig. 6. All the signals are classified into six groups:

- Address bus
- Data bus
- Control & status signals
- Power supply and frequency signals
- Externally initiated signals
- Serial I/O signals

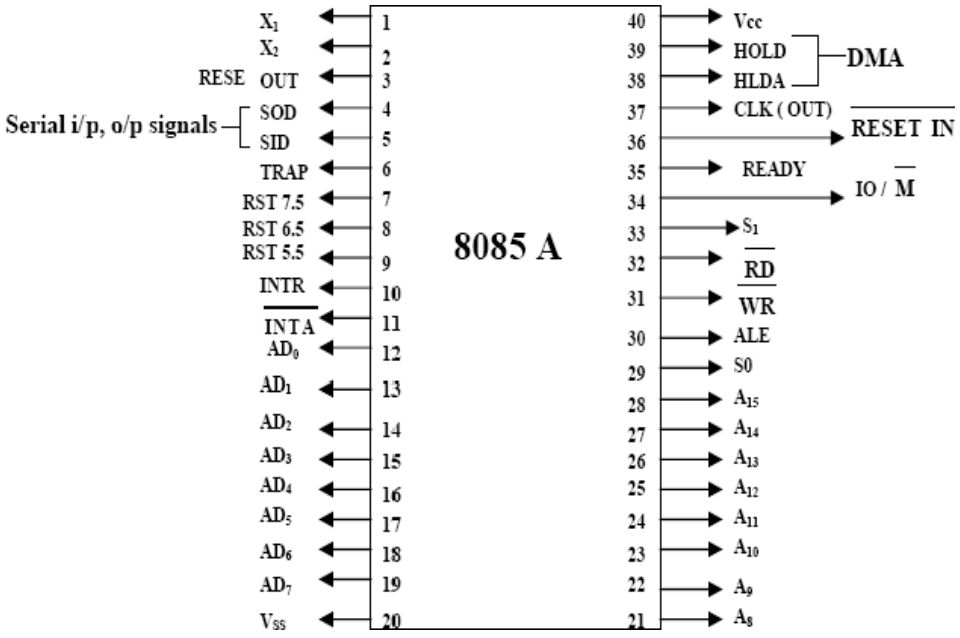


Fig. 6 8085 microprocessor pin layout and signal groups

Address and Data Buses:

- A8 – A15 (output, 3-state): Most significant eight bits of memory addresses and the eight bits of the I/O addresses. These lines enter into tri-state high impedance state during HOLD and HALT modes.
- AD0 – AD7 (input/output, 3-state): Lower significant bits of memory addresses and the eight bits of the I/O addresses during first clock cycle. Behaves as data bus

during third and fourth clock cycle. These lines enter into tri-state high impedance state during HOLD and HALT modes.

Control & Status Signals:

- ALE: Address latch enable
- $\overline{RD}$: Read control signal.
- $\overline{WR}$: Write control signal.
- IO/$\overline{M}$, S1 and S0 : Status signals.

Power Supply & Clock Frequency:

- Vcc: +5 V power supply
- Vss: Ground reference
- X1, X2: A crystal having frequency of 6 MHz is connected at these two pins
- CLK: Clock output

Externally Initiated and Interrupt Signals:

- $\overline{RESET\ IN}$: When the signal on this pin is low, the PC is set to 0, the buses are tri-stated and the processor is reset.
- RESET OUT: This signal indicates that the processor is being reset. The signal can be used to reset other devices.
- READY: When this signal is low, the processor waits for an integral number of clock cycles until it goes high.
- HOLD: This signal indicates that a peripheral like DMA (direct memory access) controller is requesting the use of address and data bus.
- HLDA: This signal acknowledges the HOLD request.
- INTR: Interrupt request is a general-purpose interrupt.
- $\overline{INTA}$: This is used to acknowledge an interrupt.
- RST 7.5, RST 6.5, RST 5,5 – restart interrupt: These are vectored interrupts and have highest priority than INTR interrupt.
- TRAP: This is a non-maskable interrupt and has the highest priority.

Serial I/O Signals:

- SID: Serial input signal. Bit on this line is loaded to D7 bit of register A using RIM instruction.
- SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

1.6 REGISTER ORGANISATION OF 8085 MICROPROCESSORRegisters

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions.

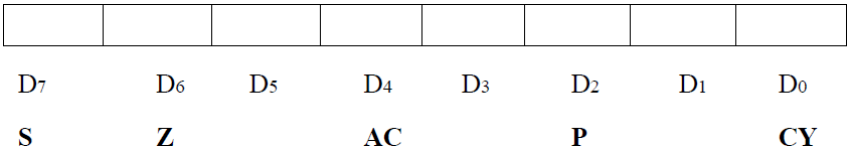
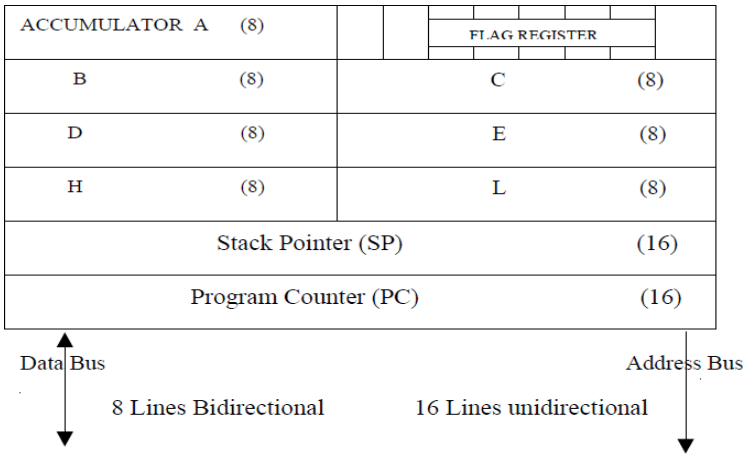


Fig. 4 Flag register

For example, after an addition of two numbers, if the result in the accumulator is larger than 8-bit, the flip-flop uses to indicate a carry by setting CY flag to 1. When an arithmetic operation results in zero, Z flag is set to 1. The S flag is just a copy of the bit D7 of the accumulator. A negative number has a 1 in bit D7 and a positive number has a 0 in 2's



complement representation. The AC flag is set to 1, when a carry result from bit D3 and passes to bit D4. The P flag is set to 1, when the result in accumulator contains even number of 1s.

Difference between SPR and GPR

SPR	GPR
1- SPR holds programme state , they usualy include programme counter , stack counter and status register . 2- There are 3 types of SPR are present in 8085 microprocessor. That is Instruction pointer , Instruction register and programme status bar. 3- When the control unit fetches an instruction from memory, it stores it in the instruction register.	1- GPR can store any temporary data during programme operation. 2- There are 6 8 bit GPR ,that are B,C,D,E,H and L. 3- They can also used as register pair like BC,DE,HL.It is also used as memory pointer.

1.7 Stack, Stack pointer and stack top.

Stack-Stack is a sequence of memory location set by a programmer to store different elements so it is called storage device.
Stack works by LIFO(Last in first out operation)
Stack Pointer(SP)- It is a 16 bit SPR.
Stack is a memory unit with an adress register called SP.
The SP always points at the top element in the stack is known as STACK TOP.

1.8 Interrupt of 8085 microprocessor.

Interrupt Structure:

Interrupt is the mechanism by which the processor is made to transfer control from its current program execution to another program having higher priority. The interrupt signal may be given to the processor by any external peripheral device.

The program or the routine that is executed upon interrupt is called interrupt service routine (ISR). After execution of ISR, the processor must return to the interrupted program. Key features in the interrupt structure of any microprocessor are as follows:

i. Number and types of interrupt signals available.
ii. The address of the memory where the ISR is located for a particular interrupt signal. This address is called interrupt vector address (IVA).
iii. Masking and unmasking feature of the interrupt signals.
iv. Priority among the interrupts.
v. Timing of the interrupt signals.
vi. Handling and storing of information about the interrupt program (status information).

Types of Interrupts:

Interrupts are classified based on their maskability, IVA and source. They are classified as:

i. Vectored and Non-Vectored Interrupts
   - Vectored interrupts require the IVA to be supplied by the external device that gives the interrupt signal. This technique is vectoring, is implemented in number of ways.
   - Non-vectored interrupts have fixed IVA for ISRs of different interrupt signals.

ii. Maskable and Non-Maskable Interrupts
   - Maskable interrupts are interrupts that can be blocked. Masking can be done by software or hardware means.
   - Non-maskable interrupts are interrupts that are always recognized; the corresponding ISRs are executed.

iii. Software and Hardware Interrupts
   - Software interrupts are special instructions, after execution transfer the control to predefined ISR.

Hardware interrupts are signals given to the processor, for recognition as an interrupt and execution of the corresponding ISR

Interrupt Handling Procedure:

The following sequence of operations takes place when an interrupt signal is recognized:

i. Save the PC content and information about current state (flags, registers etc) in the stack.
ii. Load PC with the beginning address of an ISR and start to execute it.
iii. Finish ISR when the return instruction is executed.
iv. Return to the point in the interrupted program where execution was interrupted.

Interrupt Sources and Vector Addresses in 8085:

Software Interrupts:

8085 instruction set includes eight software interrupt instructions called Restart (RST) instructions. These are one byte instructions that make the processor execute a subroutine at predefined locations. Instructions and their vector addresses are given in Table

Table Software interrupts and their vector addresses

Instruction	Machine hex code	Interrupt Vector Address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0032H

The software interrupts can be treated as CALL instructions with default call locations. The concept of priority does not apply to software interrupts as they are inserted into the program as instructions by the programmer and executed by the processor when the respective program lines are read.

Hardware Interrupts and Priorities:

8085 have five hardware interrupts – INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP. Their IVA and priorities are given in Table

Table Hardware interrupts of 8085

Interrupt	Interrupt vector address	Maskable or non-maskable	Edge or level triggered	priority
TRAP	0024H	Non-makable	Level	1
RST 7.5	003CH	Maskable	Rising edge	2
RST 6.5	0034H	Maskable	Level	3
RST 5.5	002CH	Maskable	Level	4
INTR	Decided by hardware	Maskable	Level	5

Masking of Interrupts:

Masking can be done for four hardware interrupts INTR, RST 5.5, RST 6.5, and RST 7.5. The masking of 8085 interrupts is done at different levels. Fig. 13 shows the organization of hardware interrupts in the 8085.

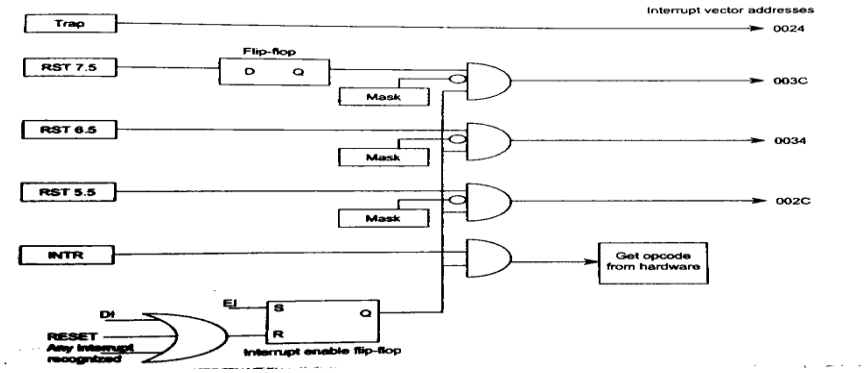


Fig. Interrupt structure of 8085

The Fig is explained by the following five points:

i. The maskable interrupts are by default masked by the Reset signal. So no interrupt is recognized by the hardware reset.
ii. The interrupts can be enabled by the EI instruction.
iii. The three RST interrupts can be selectively masked by loading the appropriate wordin the accumulator and executing SIM instruction. This is called software masking.
iv. All maskable interrupts are disabled whenever an interrupt is recognized.
v. All maskable interrupts can be disabled by executing the DI instruction.

RST 7.5 alone has a flip-flop to recognize edge transition. The DI instruction reset interrupt enable flip-flop in the processor and the interrupts are disabled. To enable interrupts, EI instruction has to be executed.

SIM Instruction:

The SIM instruction is used to mask or unmask RST hardware interrupts. When executed, the SIM instruction reads the content of accumulator and accordingly mask or unmask the interrupts. The format of control word to be stored in the accumulator before executing SIM instruction is as shown in Fig. .

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
Explanation	Serial data to be sent	Serial data enable—set to 1 for sending	Not used	Reset RST 7.5 flip-flop	Mask set enable—Set to 1 to mask interrupts	Set to 1 to mask RST 7.5	Set to 1 to mask RST 6.5	Set to 1 to mask RST 5.5

Fig. Accumulator bit pattern for SIM instruction

In addition to masking interrupts, SIM instruction can be used to send serial data on the SOD line of the processor. The data to be send is placed in the MSB bit of the accumulator and the serial data output is enabled by making D6 bit to 1.

RIM Instruction:

RIM instruction is used to read the status of the interrupt mask bits. When RIM instruction is executed, the accumulator is loaded with the current status of the interrupt masks and the pending interrupts. The format and the meaning of the data stored in the accumulator after execution of RIM instruction is shown in Fig..

In addition RIM instruction is also used to read the serial data on the SID pin of the processor. The data on the SID pin is stored in the MSB of the accumulator after the execution of the RIM instruction.

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
Explanation	Serial input data in the SID pin	Set to 1 if RST 7.5 is pending	Set to 1 if RST 6.5 is pending	Set to 1 if RST 5.5 is pending	Set to 1 if interrupts are enabled	Set to 1 if RST 7.5 is masked	Set to 1 if RST 6.5 is masked	Set to 1 if RST 5.5 is masked

Fig. Accumulator bit pattern after execution of RIM instruction

Ex: Write an assembly language program to enables all the interrupts in 8085 after reset.

EI : Enable interrupts

MVI A, 08H : Unmask the interrupts

SIM : Set the mask and unmask using SIM instruction

Timing of Interrupts:

The interrupts are sensed by the processor one cycle before the end of execution of each instruction. An interrupts signal must be applied long enough for it to be recognized. The longest instruction of the 8085 takes 18 clock periods. So, the interrupt signal must be applied for at least 17.5 clock periods. This decides the minimum pulse width for the interrupt signal.

The maximum pulse width for the interrupt signal is decided by the condition that the interrupt signal must not be recognized once again. This is under the control of the programmer.

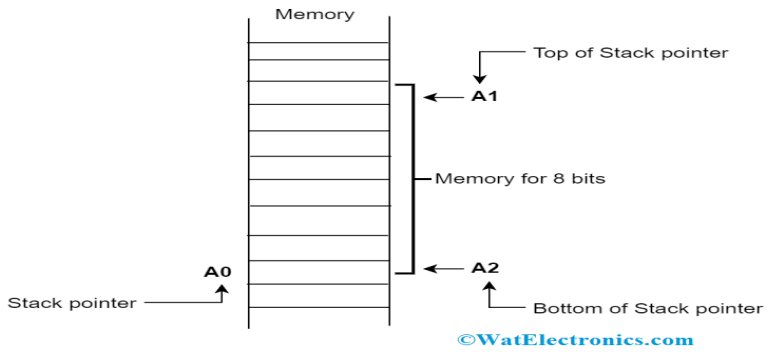
Important short question with answer

1-Define programme counter.

A program counter is a 16 bit special purpose register in a micro processor that contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its stored value by 1.

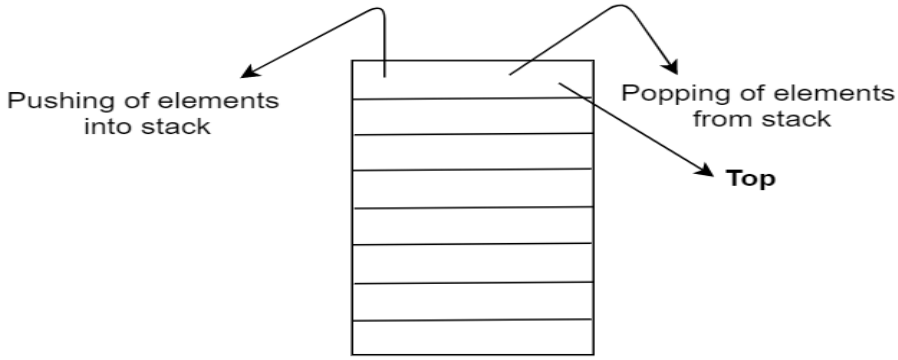
2-What is the function of Stack pointer ?

It is a 16 bit special purpose register . The stack pointer always point at the top element in the stack.



3-Define Stack .

A stack is stated as the container of elements where insertion and removal of the elements follow with the last-in-first-out (LIFO) theory. Here, the insertion of elements is done through push operation and removal of elements is done through a pop operation



4-Write the function of ALU ?

The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data. The control unit fetches data and instructions from memory and uses operations of the ALU to carry out those instructions using that data.

5-Write the different register name of 8085 MP ?

In 8085 two types of register are used .

1-General purpose register

2-Special purpose register

General purpose are B,C,D,E,H and L , Accumalotor,Instruction register ,Temporary Register are in 8 bit.

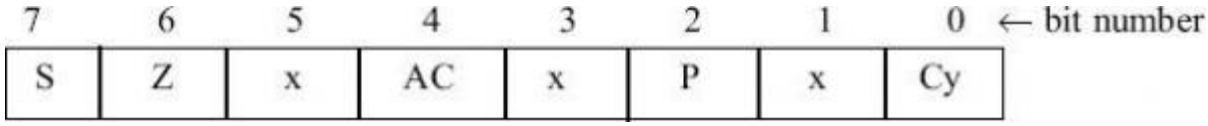
Special purpose register are Stack pointer and programme counter are in 16 bit.

6- Show the bit position of Flag register

In 8085 microprocessor, the flags register can have a total of eight flags. Thus a flag can be represented by 1 bit of information. But only five flags are implemented in 8085. And they are:

- Carry flag (Cy),
- Auxiliary carry flag (AC),
- Sign flag (S),
- Parity flag (P), and
- Zero flag (Z).

The respective position of these flag bits in flag register has been show the below figure. The positions marked by “x” are to be considered as don't care bits in the flags register. The user is not required to memorize the positions of these flags in the flags register.



7-What is carry flag and auxiliary cary flag ?

Auxiliary carry flag (Ac): Now let us consider the addition of any two 8-bit (2-hex digit) numbers, a **carry** may be generated when we add the LS hex digits of the two numbers. Such a **carry** is called intermediate **carry** also known as half **carry**, or **auxiliary carry.**
Carry Flag (CY) – **Carry** is generated when performing n bit operations and the result is more than n bits, then this **flag** becomes set i.e. 1, otherwise it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset and if (A<B) it becomes set. **Carry flag** is also called borrow **flag.**

8-What is the function of ALE.

ALE (Address Enable Latch) is the control signal which is nothing but a positive going pulse generated when a new operation is started by microprocessor. So when pulse goes high means **ALE**=1, it makes address bus enable and when **ALE**=0, means low pulse makes data bus enable.

9-Write different interrept are present accordind to the priroty order.

They are TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. These **interrupts** have a fixed **priority** of **interrupt** service. If two or more **interrupts** go high at the same time, the **8085** will service them on **priority** basis. The TRAP has the highest **priority** followed by RST 7.5, RST 6.5, RST 5.5.

10-What is the function of SO and S1 .

This signal separates memory and I/O devices. Status signals(**S0 and S1**): These are output status signals **used** to give information of operation performed by microprocessor. The **S0 and S1** lines specify 4 different conditions of 8085 machine cycles.

11- Define adress Bus

The **address bus** is uni-directional. It is concerned with passing an **address** one way, from the CPU to RAM. The sole purpose of an **address bus** is to identify the **address** of the location in cache or main memory that is to be read from or written to the processor.

12-Define data Bus.

Data bus - carries the **data** between the processor and other components. The **data bus** is bidirectional . Control **bus** - carries control signals from the processor to other components. The control **bus** also carries the clock's pulses.
Lond Questions .

1-Explain the Architechture of 8085 MP with neat block diagram ?
2-Explain the pin configuration of 8085 MP ?
3-Write soft notes on
Stack, Flag register,8085 Busses

CHAPTER-2.0

Instruction set and Assembley language programming of 8085 microprocessor.

2.1-Adressing data and differenciating between one byte , two byte and three byte instruction with examples.

- According to the length,the instruction of 8085 microprocessor is classified into 3 types.

1-Single byte instruction
2-Two byte instruction
3-Three byte instruction

1-Single byte instruction

In single byte instruction only opcode is present, there is no operand.Examples-MOV A,B, ADD B, CMA

In this instructions only opcode is present so these are the single byte instruction.
The length of this instruction is 8 bit.Each instruction requirdes one memory location.

2-Two byte instruction

In two byte instruction the first 8 bit indicates the opcode and next 8 bit indicatesa the oprand.
Example MVI A,32H, ADI A,08H

The length of this instruction is 16 bit that is the opcode is 8 bit and oprand is 8 bit.
Each instruction requirdes two memory location.

3-Three byte instruction

Three instruction is the type of instruction in which the first 8 bit indicates the opcode and next 2 bytes specifieds the oprand which is 16 b it address.The low order address is represented in 2nd byte and the higher orders adress represented in the 3rd byte.

Example-LBA 2050H, JMP 2085H

These instruction would requirdes 3 memory location to store the binary code.

2.2-Adressing modes in instructions with suitable examples.

Addressing Modes in Instructions:

The process of specifying the data to be operated on by the instruction is called addressing. The various formats for specifying operands are called addressing modes. The 8085 has the following five types of addressing:

I. Immediate addressing
II. Memory direct addressing
III. Register direct addressing
IV. Indirect addressing
V. Implicit addressing

Immediate Addressing:

In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location.

Ex: MVI A, 9AH

- The operand is a part of the instruction.
- The operand is stored in the register mentioned in the instruction.

Memory Direct Addressing:

Memory direct addressing moves a byte or word between a memory location and register. The memory location address is given in the instruction.

Ex: LDA 850FH

This instruction is used to load the content of memory address 850FH in the accumulator.

Register Direct Addressing:

Register direct addressing transfer a copy of a byte or word from source register to destination register.

Ex: MOV B, C

It copies the content of register C to register B.

Indirect Addressing:

Indirect addressing transfers a byte or word between a register and a memory location.

Ex: MOV A, M

Here the data is in the memory location pointed to by the contents of HL pair. The data is moved to the accumulator.

Implicit Addressing

In this addressing mode the data itself specifies the data to be operated upon.

Ex: CMA

The instruction complements the content of the accumulator. No specific data or operand ismentioned in the instruction

2.3-Instruction set of 8085(Data transfer, arithimatic, logical, banching, Stack and I/O machine control.

2. INSTRUCTION SET OF 8085

Data Transfer Instructions:

Opcode	Operand	Description
Copy from source to destination MOV	Rd, Rs M, Rs Rd, M	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. Example: MOV B, C or MOV B, M
Move immediate 8-bit MVI	Rd, data M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: MVI B, 57 or MVI M, 57
Load accumulator LDA	16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. Example: LDA 2034 or LDA XYZ
Load accumulator indirect LDAX	B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B
Load register pair immediate LXI	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034
Load H and L registers direct LHLD	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. Example: LHLD 2040

Store accumulator direct

STA 16-bit address

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

Store accumulator indirect

STAX Reg. pair

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

Store H and L registers direct

SHLD 16-bit address

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

Exchange H and L with D and E

XCHG none

The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.
Example: XCHG

Copy H and L registers to the stack pointer

SPHL none

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

Exchange H and L with top of stack

XTHL none

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

Push register pair onto stack

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

Pop off stack to register pair

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

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

OUT	8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand. Example: OUT 87

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

IN	8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example: IN 82

Arithmetic Instructions:

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

Subtract register or memory from accumulator

SUB	R M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SUB B or SUB M

Subtract source and borrow from accumulator

SBB	R M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SBB B or SBB M

Subtract immediate from accumulator

SUI	8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction. Example: SUI 45

Subtract immediate from accumulator with borrow

SBI	8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtracion. Example: SBI 45

Increment register or memory by 1

INR	R M	The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: INR B or INR M

Increment register pair by 1

INX	R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place. Example: INX H

Decrement register or memory by 1

DCR R
M

The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
Example: DCR B or DCR M

Decrement register pair by 1

DCX R

The contents of the designated register pair are decremented by 1 and the result is stored in the same place.
Example: DCX H

Decimal adjust accumulator

DAA none

The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation.

If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits.

If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

Example: DAA

BRANCHING INSTRUCTIONS

Opcode Operand Description

Jump unconditionally

JMP 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
Example: JMP 2034 or JMP XYZ

Jump conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.
Example: JZ 2034 or JZ XYZ

Opcode	Description	Flag Status
JC	Jump on Carry	CY = 1
JNC	Jump on no Carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JZ	Jump on zero	Z = 1
JNZ	Jump on no zero	Z = 0
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0

Unconditional subroutine call

CALL 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
Example: CALL 2034 or CALL XYZ

Call conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
Example: CZ 2034 or CZ XYZ

Opcode	Description	Flag Status
CC	Call on Carry	CY = 1
CNC	Call on no Carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CZ	Call on zero	Z = 1
CNZ	Call on no zero	Z = 0
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0

Return from subroutine unconditionally

RET none

The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RET

Return from subroutine conditionally

Operand: none

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RZ

Opcode	Description	Flag Status
RC	Return on Carry	CY = 1
RNC	Return on no Carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RZ	Return on zero	Z = 1
RNZ	Return on no zero	Z = 0
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0

Load program counter with HL contents

PCHL none — The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
Example: PCHL

Restart

RST 0-7 — The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

LOGICAL INSTRUCTIONS

Opcode	Operand	Description

Compare register or memory with accumulator

CMP R / M — The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < (reg/mem): carry flag is set, s=1
if (A) = (reg/mem): zero flag is set, s=0
if (A) > (reg/mem): carry and zero flags are reset, s=0
Example: CMP B or CMP M

Compare immediate with accumulator

CPI 8-bit data — The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < data: carry flag is set, s=1
if (A) = data: zero flag is set, s=0
if (A) > data: carry and zero flags are reset, s=0
Example: CPI 89

Logical AND register or memory with accumulator

ANA R / M — The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANA B or ANA M

Logical AND immediate with accumulator

ANI 8-bit data — The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANI 86

Exclusive OR register or memory with accumulator

XRA	R M	The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M

Exclusive OR immediate with accumulator

XRI	8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRI 86

Logical OR register or memory with accumulaotr

ORA	R M	The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORA B or ORA M

Logical OR immediate with accumulator

ORI	8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORI 86

Rotate accumulator left

RLC	none	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D_0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RLC

Rotate accumulator right

RRC	none	Each binary bit of the accumulator is rotated right by one position. Bit D_0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D_0. S, Z, P, AC are not affected. Example: RRC

Rotate accumulator left through carry

RAL	none	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RAL

Rotate accumulator right through carry

RAR	none	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RAR

Complement accumulator

CMA	none	The contents of the accumulator are complemented. No flags are affected. Example: CMA

Complement carry

CMC	none	The Carry flag is complemented. No other flags are affected. Example: CMC

Set Carry

STC	none	The Carry flag is set to 1. No other flags are affected. Example: STC

CONTROL INSTRUCTIONS

Opcode	Operand	Description

No operation

NOP	none	No operation is performed. The instruction is fetched and decoded. However no operation is executed. Example: NOP

Halt and enter wait state

HLT	none	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. Example: HLT

Disable interrupts

DI	none	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected. Example: DI

Enable interrupts

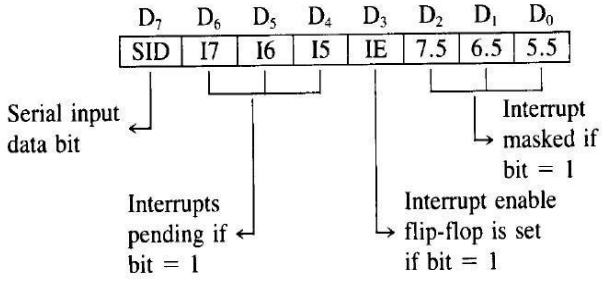
EI	none	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP). Example: EI

Read interrupt mask

RIM none

This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

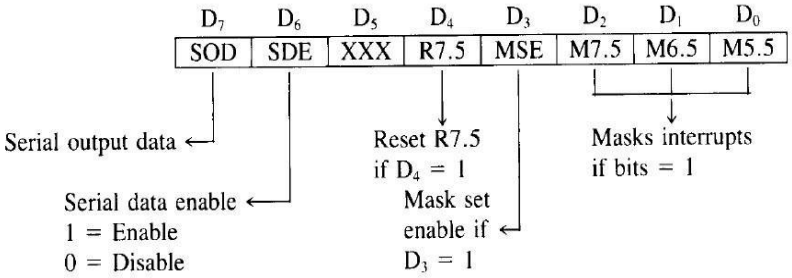


Set interrupt mask

SIM none

This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.

Example: SIM



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

2.4 Simple Assembly language programming of 8085 microprocessor

ADDITION OF TWO 8 BIT NUMBERS

AIM:

To perform addition of two 8 bit numbers using 8085.

ALGORITHM:

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Store the value of sum and carry in memory location.
7) Terminate the program.

PROGRAM:

```
        MVI   C, 00    Initialize C register to 00
        LDA   4150     Load the value to Accumulator.
        MOV   B, A     Move the content of Accumulator to B register.
        LDA   4151     Load the value to Accumulator.
        ADD   B        Add the value of register B to A
        JNC   LOOP     Jump on no carry.
        INR   C        Increment value of register C
LOOP:   STA   4152     Store the value of Accumulator (SUM).
        MOV   A, C     Move content of register C to Acc.
        STA   4153     Store the value of Accumulator (CARRY)
        HLT            Halt the program.
```

OBSERVATION:

Input:	80 (4150)
	80 (4251)
Output:	00 (4152)
	01 (4153)

RESULT:

Thus the program to add two 8-bit numbers was executed.

SUBTRACTION OF TWO 8 BIT NUMBERS

AIM:

To perform the subtraction of two 8 bit numbers using 8085.

ALGORITHM:

1. Start the program by loading the first data into Accumulator.
2. Move the data to a register (B register).
3. Get the second data and load into Accumulator.
4. Subtract the two register contents.
5. Check for carry.
6. If carry is present take 2's complement of Accumulator.
7. Store the value of borrow in memory location.
8. Store the difference value (present in Accumulator) to a memory
9. location and terminate the program.

PROGRAM:

```
        MVI     C, 00     Initialize C to 00
        LDA     4150      Load the value to Acc.
        MOV     B, A      Move the content of Acc to B  register.
        LDA     4151      Load the value to Acc.
        SUB     B
        JNC     LOOP      Jump on no carry.
        CMA               Complement Accumulator contents.
        INR     A         Increment value in Accumulator.
        INR     C         Increment value in register C
LOOP:   STA     4152      Store the value of A-reg to memory address.
        MOV     A, C      Move contents of register C to Accumulator.
        STA     4153      Store the value of Accumulator memory address.
        HLT               Terminate the program.
```

OBSERVATION:

Input: 05 (4200) ----- Array Size
0A (4201)
F1 (4202)
1F (4203)
26 (4204)
FE (4205)

Output: FE (4300)

RESULT:

Thus the program to find the largest number in an array of data was executed

LARGEST NUMBER IN AN ARRAY OF DATA

AIM:

To find the largest number in an array of data using 8085 instruction set.

ALGORITHM:

1) Load the address of the first element of the array in HL pair
2) Move the count to B – reg.
3) Increment the pointer
4) Get the first data in A – reg.
5) Decrement the count.
6) Increment the pointer
7) Compare the content of memory addressed by HL pair with that of A - reg.
8) If Carry = 0, go to step 10 or if Carry = 1 go to step 9
9) Move the content of memory addressed by HL to A – reg.
10) Decrement the count
11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
12) Store the largest data in memory.
13) Terminate the program.

PROGRAM:

	LXI	H,4200	Set pointer for array
	MOV	B,M	Load the Count
	INX	H	
	MOV	A,M	Set 1st element as largest data
	DCR	B	Decrement the count
LOOP:	INX	H	
	CMP	M	If A- reg > M go to AHEAD
	JNC	AHEAD	
	MOV	A,M	Set the new value as largest
AHEAD:	DCR	B	
	JNZ	LOOP	Repeat comparisons till count = 0
	STA	4300	Store the largest value at 4300
	HLT		

OBSERVATION:

Input:
05 (4200) ----- Array Size
0A (4201)
F1 (4202)
1F (4203)
26 (4204)
FE (4205)

Output: FE (4300)

RESULT:

Thus the program to find the largest number in an array of data was executed

SMALLEST NUMBER IN AN ARRAY OF DATA

AIM:

To find the smallest number in an array of data using 8085 instruction set.

ALGORITHM:

1) Load the address of the first element of the array in HL pair
2) Move the count to B – reg.
3) Increment the pointer
4) Get the first data in A – reg.
5) Decrement the count.
6) Increment the pointer
7) Compare the content of memory addressed by HL pair with that of A - reg.
8) If carry = 1, go to step 10 or if Carry = 0 go to step 9
9) Move the content of memory addressed by HL to A – reg.
10) Decrement the count
11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
12) Store the smallest data in memory.
13) Terminate the program.

PROGRAM:

	LXI	H,4200	Set pointer for array
	MOV	B,M	Load the Count
	INX	H	
	MOV	A,M	Set 1st element as largest data
	DCR	B	Decrement the count
LOOP:	INX	H	
	CMP	M	If A- reg < M go to AHEAD
	JC	AHEAD	
	MOV	A,M	Set the new value as smallest
AHEAD:	DCR	B	
	JNZ	LOOP	Repeat comparisons till count = 0
	STA	4300	Store the largest value at 4300
	HLT		

OBSERVATION:

Input: 05 (4200) ----- Array Size
0A (4201)
F1 (4202)
1F (4203)
26 (4204)
FE (4205)

Output: 0A (4300)

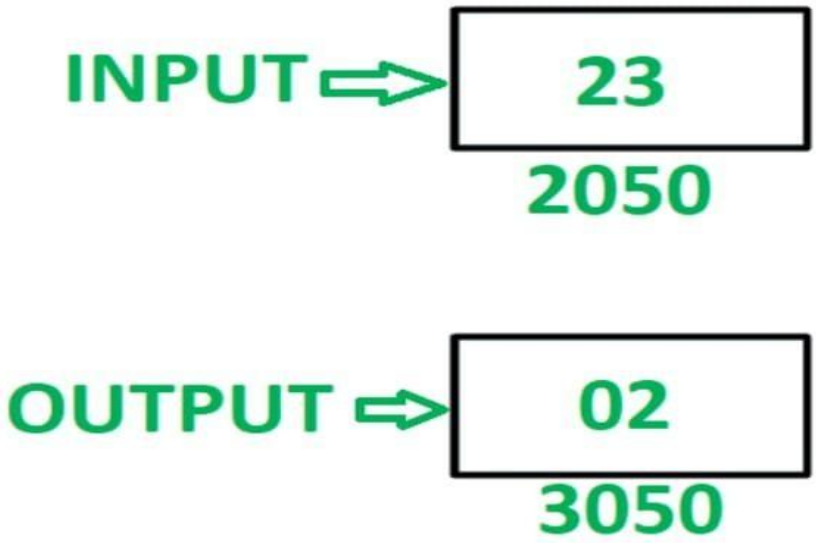
RESULT:

Thus the program to find the smallest number in an array of data was executed

8085 program to perform AND operation in nibbles of 8 bit number

Problem – Write an assembly language program in 8085 microprocessor to perform **AND** operation between lower and higher order nibble of 8 bit number.

Example –



Assumption – 8 bit number is stored at memory location 2050. Final result is stored at memory location 3050.

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LDA 2050	A <- M[2050]
2003	ANI 0F	A <- A (AND) 0F
2005	MOV B, A	B <- A
2006	LDA 2050	A <- M[2050]
2009	ANI F0	A <- A (AND) F0
200B	RLC	Rotate accumulator left by one bit without carry
200C	RLC	Rotate accumulator left by one bit without

200D	RLC	Rotate accumulator left by one bit without carry
200E	RLC	Rotate accumulator left by one bit without carry
200F	ANA B	A <- A (AND) B
2010	STA 3050	M[3050] <- A
2013	HLT	END

Explanation – Registers A, B are used for general purpose.

1. **LDA 2050:** load the content of memory location 2050 in accumulator A.
2. **ANI 0F:** perform AND operation in A and 0F. Store the result in A.
3. **MOV B, A:** moves the content of A in register B.
4. **LDA 2050:** load the content of memory location 2050 in accumulator A.
5. **ANI F0:** perform AND operation in A and F0. Store the result in A.
6. **RLC:** rotate the content of A left by one bit without carry. Use this instruction 4 times to reverse the content of A.
7. **ANA B:** perform AND operation in A and B. Store the result in A.
8. **STA 3050:** store the content of A in memory location 3050.
9. **HLT:** stops executing the program and halts any further execution.

Questions

Short questions with answer.

1-How many types of instruction sets are there.

According to the length the instruction of 8085 MP is classified into three types.
1-Single byte instruction
2-Two byte in struction
3-Three byte instruction

Q.2-Define Single byte instruction.

In single byte instruction only Opcode is present,there is no oprand.Example-MOV A,B,ADD B,CMA.
The length of this instruction is 8 bit. Each instruction requires one memory location.

Q 3- Define adressing mode of 8085 MP

The way of specifying data to be operated by an instruction is called **addressing mode**. In immediate **addressing mode** the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

Q- 4 – How many types of adressing modes are present in 8085 MP

i-Immediate Addressing Mode
ii-Register Addressing Mode
iii-Direct Addressing Mode
iv-Register Indirect Addressing Mode
v-mplied/Implicit Addressing Mode

Q 5-Define Implicit adressing mode of 8085 MP

In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.
Examples:
CMA (finds and stores the 1's complement of the contains of accumulator A in A)
RRC (rotate accumulator A right by one bit)
RLC (rotate accumulator A left by one bit)

Q 6- Define Immediate Addressing Mode

In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.
Examples:
MVI B 45 (move the data 45H immediately to register B)
LXI H 3050 (load the H-L pair with the operand 3050H immediately)
JMP address (jump to the operand address immediately)

Q 7- Define STA 16 bit adress

In 8085 Instruction set, **STA** is a mnemonic that stands for STore Accumulator contents in memory. In this instruction,Accumulator8-**bit** content will be stored to a memory location whose **16-bit address** is indicated in the instruction as a16. ... This instruction occupies 3-Bytes of memory.

Q 8- Define XCHG in 8085 MP

In **8085** Instruction set, there is one mnemonic **XCHG**, which stands for eXCHanGe. This is an instruction to exchange contents of HL register pair with DE register pair. ... After execution of this instruction, the content between H and D registers and L and E registers will get swapped respectively.

Q 9-.Difference between RIM ans SIM instruction.

1	SIM stands for Set Interrupt Mask.	RIM stands for Read Interrupt Mask.
2	It is responsible for masking/unmasking of RST 7.5, RST 6.5 and RST 5.5.	It checks whether RST 7.5, RST 6.5, RST 5.5 are masked or not.
3	It resets to 0 RST 7.5 flip flop.	It checks whether interrupts are enabled or not and to check whether RST 7.5, RST 6.5 or RST 5.5 interrupts are pending or not.

Q 10- Define branching instruction with example.

Branching instructions refer to the act of switching execution to a different **instruction** sequence as a result of executing a **branch instruction**. The three types of **branching instructions** are: Jump (unconditional and conditional) Call (unconditional and conditional) Return (unconditional and conditional)
Example-JC,JNC,JP,JM etc

Long Questions

1-Differenciate between 1 byte ,2 byte ,3 byte instruction with examples.
2-Explain different types of adressing mode in 8085 MP with examples
3-Explain RIM and Sim instruction.
4-Write an assembly language programme of addition of two 8 bit no result 16 bit.
5-Write an assembly language programme to find smallest number in an arrey of data.

CHAPTER-3.0

TIMEING DIAGRAMS

3.1 DEFINE OPCODE, OPRAMB,T-STATE,FETCH CYCLE,MACHINE CYCLE, INSTRUCTION CYCLE AND DISCUSS THE CONCEPT OF TIMEING DIAGRAM.

Each instruction in 8085 microprocessor consists of two part- operation code (opcode) and operand. The opcode is a command such as ADD and the operand is an object to be operated on, such as a byte or the content of a register.

Instruction Cycle: The time taken by the processor to complete the execution of an instruction. An instruction cycle consists of one to six machine cycles.

Machine Cycle: The time required to complete one operation; accessing either the memory or I/O device. A machine cycle consists of three to six T-states.

T-State: Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.

To execute a program, 8085 performs various operations as:

- Opcode fetch
- Operand fetch
- Memory read/write
- I/O read/write

External communication functions are:

- Memory read/write
- I/O read/write
- Interrupt request acknowledge

Opcode Fetch Machine Cycle:

It is the first step in the execution of any instruction. The timing diagram of this cycle is given in Fig. .

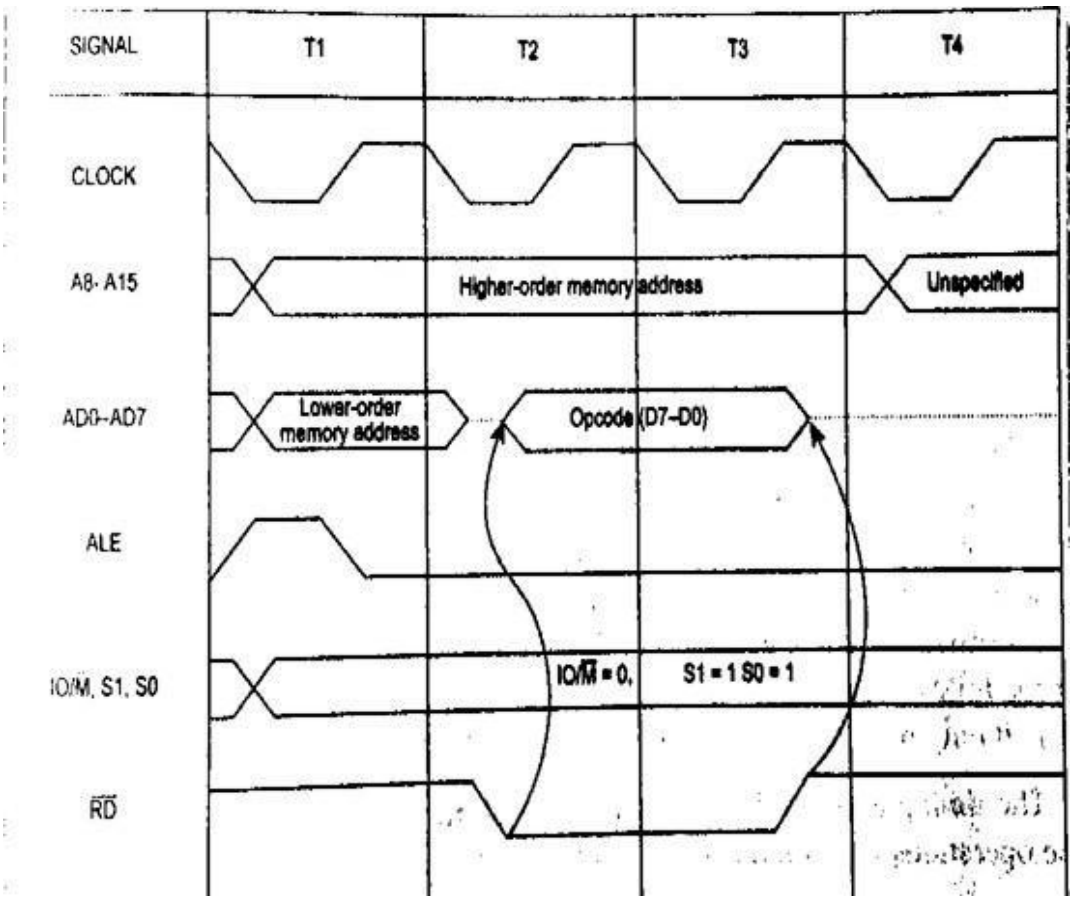
The following points explain the various operations that take place and the signals that are changed during the execution of opcode fetch machine cycle:

T1 clock cycle

i. The content of PC is placed in the address bus; AD0 - AD7 lines contains lower bit address and A8 – A15 contains higher bit address.
ii. IO/$\overline{M}$ signal is low indicating that a memory location is being accessed. S1 and S0 also changed to the levels as indicated in Table 1.
iii. ALE is high, indicates that multiplexed AD0 – AD7 act as lower order bus.

T2 clock cycle

i. Multiplexed address bus is now changed to data bus.



ii. The RD signal is made low by the processor. This signal makes the memory device load the data bus with the contents of the location addressed by the processor.

T3 clock cycle

i. The opcode available on the data bus is read by the processor and moved to the instruction register.

ii. The RD signal is deactivated by making it logic 1.

T4 clock cycle

i. The processor decode the instruction in the instruction register and generate the necessary control signals to execute the instruction. Based on the instruction further operations such as fetching, writing into memory etc takes place.

Memory Read Machine Cycle:

The memory read cycle is executed by the processor to read a data byte from memory. The machine cycle is exactly same to opcode fetch except: a) It has three T-states b) The S0 signal is set to 0. The timing diagram of this cycle is given in Fig.

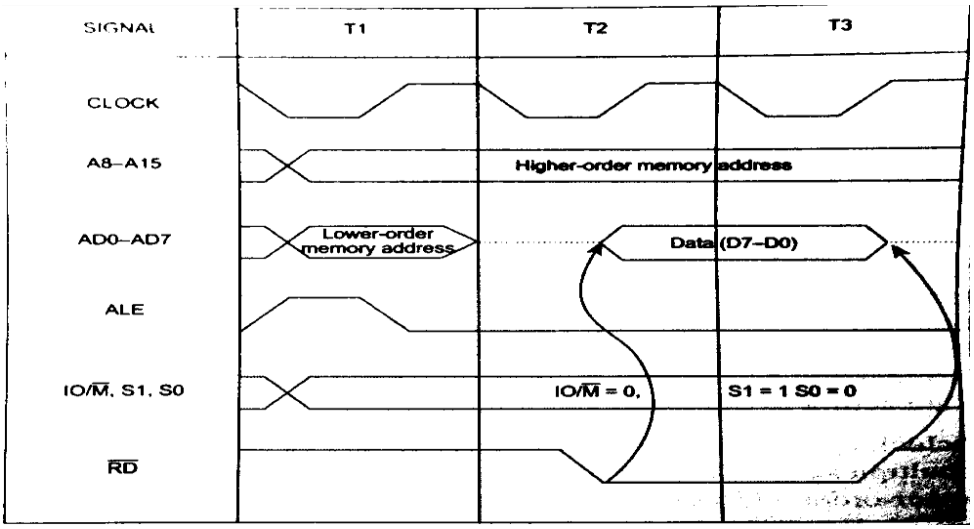


Fig. Timing diagram for memory read machine cycle

Memory Write Machine Cycle:

The memory write cycle is executed by the processor to write a data byte in a memory location. The processor takes three T-states and $\overline{WR}$ signal is made low. The timing diagram of this cycle is given in Fig. 9.

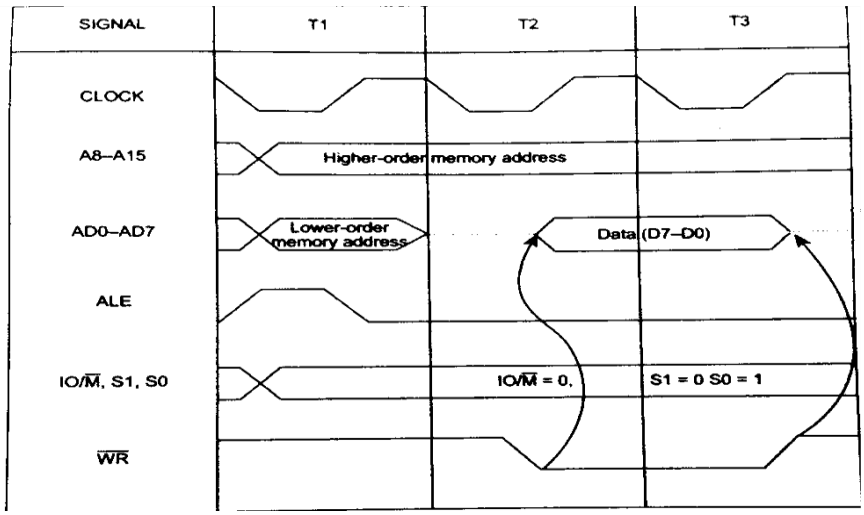


Fig.Timing diagram for memory write machine cycle

I/O Read Cycle:

The I/O read cycle is executed by the processor to read a data byte from I/O port or from peripheral, which is I/O mapped in the system. The 8-bit port address is placed both in the lower and higher order address bus. The processor takes three T-states to execute this machine cycle. The timing diagram of this cycle is given in Fig.

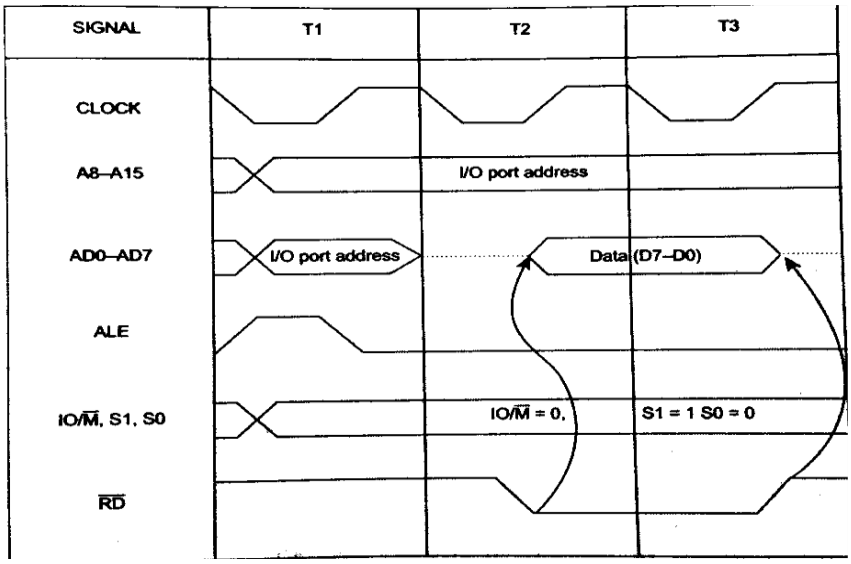
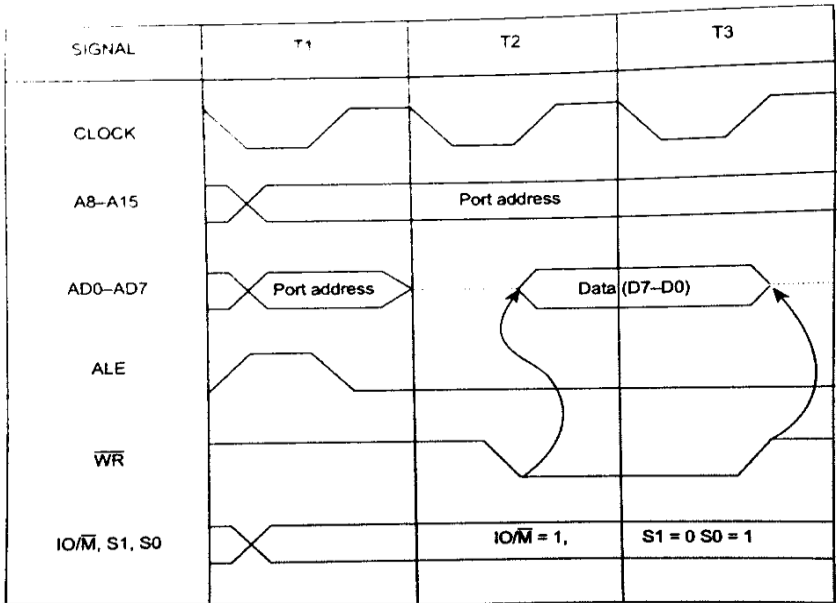


Fig.Timing diagram I/O read machine cycle

I/O Write Cycle:

The I/O write cycle is executed by the processor to write a data byte to I/O port or to a peripheral, which is I/O mapped in the system. The processor takes three T-states to execute this machine cycle. The timing diagram of this cycle is given in Fig.



Timing diagram I/O write machine cycle

Short question with answwer.

Q 1-Define Upcode and Operand.

Each assembly language statement is split into an **opcode** and an **operand** . The **opcode** is the instruction that is executed by the CPU and the **operand** is the data or memory location used to execute that instruction.

Q 2- Define instruction cycle,machine cycle, T-State.

Instruction cycle

1. The time a microprocessor needs to fetch and execute one entire instruction is known as an instruction cycle.
2. There are typically four stages of an instruction cycle that the CPU carries out-

Machine cycle:

1. The basic microprocessor operation such as reading a byte from I/O port or writing a byte to memory is called as machine cycle.
2. The time TCY in the above figure is called as the machine cycle. Thus a machine cycle consists of several T-states.

T-state:

1. One complete cycle of clock is called as T-state as shown in the above figure. The time intervals T_1T1 orT_2T2 are the examples of T-state.
2. A T-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.
3. Various versions of 8086 have maximum clock frequency from 5MHz to 10MHz. Hence the minimum time for one T-state is between 100 to 200 n sec.

Q 3- In timeing diagram how many ,machine cycles are present in 8085 MP.

1-Opcode fetch machine cycle
2-Memery read machine cycle
3-Memory write machine cycle
4-I/o read machine cycle
5-I/O write machine cycle

Long Questions

1-Draw the timeing diagram of the instruction LDAX B.
2-Draw the timeing diagram of Opcode fetch machine cycle.
3-Draw and explain the timeinf diagram of the instruction IN 82H.

QUESTIONS:

1. What is the function of a microprocessor in a system?
2. Why is the data bus in 8085 bidirectional?
3. How does microprocessor differentiate between data and instruction?
4. How long would the processor take to execute the instruction LDA 1753H if the T-state duration is 2μs?
5. Draw the timing diagram of the instruction LDAX B.
6. Sketch and explain the various pins of the 8085.
7. Explain direct addressing mode of 8085 with an example?
8. Draw and explain the timing diagram of the instruction IN 82H.
9. What is meant by 'priority of the interrupts'? Explain the operation of the interrupts structure of the 8085, with the help of a circuit diagram.
10. Explain the bit pattern for SIM instruction. Write the assembly language program lines to enable all the interrupts in the 8085 after reset.