DON BOSCO COLLEGE

DEPARTMENT OF PHYSICS

STUDY MATERIAL

SUBJECT NAME : MICROPROCESSOR AND ITS APPLICATIONS

PAPER CODE : 19UPHS06

CLASS : III -B.Sc., PHYSICS

SEMESTER : II

UNIT – I

1-1.Evolution of Microprocessor 8085

(i) 4-bit Microprocessors: The first microprocessor was introduced in 1971 by Intel Corp. It was named Intel 4004 as it was a 4 bit processor. It was a processor on a single chip. It could perform simple arithmetic and logic operations such as addition, subtraction, boolean AND and boolean OR. It had a control unit capable of performing control functions like fetching an instruction from memory, decoding it, and generating control pulses to execute it. It was able to operate on 4 bits of data at a time.This first microprocessor was quite a success in industry. Soon other microprocessors were also introduced. Intel introduced the enhanced version of 4004, the 4040. Some other 4 bit processors are International‟s PPS4 and Thoshiba‟s T3472.

(ii) 8-bit Microprocessors: The first 8 bit microprocessor which could perform arithmetic and logic operations on 8 bit words was introduced in 1973 again by Intel. This was Intel 8008 and was later followed by an improved version, Intel 8088. Some other 8 bit processors are Zilog-80 and Motorola M6800.

(iii) 16-bit Microprocessors: The 8-bit processors were followed by 16 bit processors. They are Intel 8086 and 80286.

(iv) 32-bit Microprocessors: The 32 bit microprocessors were introduced by several companies but the most popular one is Intel 80386.

(v) Pentium Series: Instead of 80586, Intel came out with a new processor namely Pentium processor. Its performance is closer to RISC performance. Pentium was followed by Pentium Pro CPU. Pentium Pro allows allow multiple CPUs in a single system in order to achive multiprocessing. The MMX extension was added to Pentium Pro and the result was Pentiuum II. The low cost version of Pentium II is Celeron. The Pentium III provided high performance floating point operations for certain types of computations by using the SIMD extensions to the instruction set. These new instructions makes the Pentium III faster than high-end RISC CPUs. Interestingly Pentium IV could not execute code faster than the Pentium III when running at the

same clock frequency. So Pentium IV had to speed up by executing at a much higher clock frequency.

1.2. Computer & Classification of Computers:

Computer is an electronic device which has many units like Input unit, Control unit and Output unit. Input unit consists of input devices like keyboard, mouse, scanner, light pen, etc., Output unit consists of output devices like printer, monitor, etc., Control unit controls all the actions of computer which consists of memory unit, Arithmetic and logic unit. A computer is one of the most brilliant inventions of mankind. Depending on the processing power and size of computers, they have been classified under various types.

(a) Classification of Computers on the basis of operational principle

Based on the operational principle of computers, they are categorized as analog, digital and hybrid computers.

(i). Analog Computers: These are almost extinct today. These are different from a digital computer because an analog computer can perform several mathematical operations simultaneously. It uses continuous variables for mathematical operations and utilizes mechanical or electrical energy.

(ii). Digital Computers: They use digital circuits and are designed to operate on two states, namely bits 0 and 1. They are analogous to states ON and OFF. Data on these computers is represented as a series of 0s and 1s. Digital computers are suitable for complex computation and have higher processing speeds. They are programmable. Digital computers are either general purpose computers or special purpose ones. General purpose computers, as their name suggests, are designed for specific types of data processing while general purpose computers are meant for general use.

(iii). Hybrid Computers: These computers are a combination of both digital and analog computers. In this type of computers, the digital segments perform process control by conversion of analog signals to digital ones.

(b) Classification on the basis of types:

(i). Mainframe Computers: Large organizations use mainframes for highly critical applications such as bulk data processing and ERP. Most of the mainframe computers have capacities to host multiple operating systems and operate as a number of virtual machines. They can substitute for several small servers.

(ii). Microcomputers: A computer with a microprocessor and its central processing unit is known as a microcomputer. They do not occupy space as much as mainframes do. When supplemented with a keyboard and a mouse, microcomputers can be called personal computers. A monitor, a keyboard and other similar input-output devices, computer memory

in the form of RAM and a power supply unit come packaged in a microcomputer. These computers can fit on desks or tables and prove to be the best choice for single-user tasks.

(iii). Personal Computers: Personal computers come in different forms such as desktops, laptops and personal digital assistants. Let us look at each of these types of computers.

(iv) Desktops: A desktop is intended to be used on a single location. The spare parts of a desktop computer are readily available at relatively lower costs. Power consumption is not as critical as that in laptops. Desktops are widely popular for daily use in the workplace and households.

(v)Laptops: Similar in operation to desktops, laptop computers are miniaturized and optimized for mobile use. Laptops run on a single battery or an external adapter that charges the computer batteries. They are enabled with an inbuilt keyboard, touch pad acting as a mouse and a liquid crystal display. Their portability and capacity to operate on battery power have proven to be of great help to mobile users.

(vi) Notebooks: They fall in the category of laptops, but are inexpensive and relatively smaller in size. They had a smaller feature set and lesser capacities in comparison to regular laptops,

(v) Personal Digital Assistants (PDAs): It is a handheld computer and popularly known as a palmtop. It has a touch screen and a memory card for storage of data. PDAs can also be used as portable audio players, web browsers and smart phones. Most of them can access the Internet by means of Bluetooth or Wi-Fi communication.

(vi) Minicomputers: In terms of size and processing capacity, minicomputers lie in between mainframes and microcomputers. Minicomputers are also called mid-range systems or workstations. The term began to be popularly used in the 1960s to refer to relatively smaller third generation computers. They took up the space that would be needed for a refrigerator or two and used transistor and core memory technologies. The 12-bit PDP-8 minicomputer of the Digital Equipment Corporation was the first successful minicomputer.

(vii) Servers: They are computers designed to provide services to client machines in a computer network. They have larger storage capacities and powerful processors. Running on them are programs that serve client requests and allocate resources like memory and time to client machines. Usually they are very large in size, as they have large processors and many hard drives. They are designed to be fail-safe and resistant to crash.

(viii) Supercomputers: The highly calculation-intensive tasks can be effectively performed by means of supercomputers. Quantum physics, mechanics, weather forecasting, molecular theory are best studied by means of supercomputers. Their ability of parallel processing and their well-designed memory hierarchy give the supercomputers, large transaction processing powers.

(ix) Wearable Computers: A record-setting step in the evolution of computers was the creation of wearable computers. These computers can be worn on the body and are often used in the study of behavior modeling and human health. Military and health professionals have incorporated wearable computers into their daily routine, as a part of such studies. When the users' hands and sensory organs are engaged in other activities, wearable computers are of great help in tracking human actions. Wearable computers do not have to be turned on and off and remain in operation without user intervention.

(x) Tablet Computers: Tablets are mobile computers that are very handy to use. They use the

touch screen technology. Tablets come with an onscreen keyboard or use a stylus or a digital pen. Apple's iPad redefined the class of tablet computers.

1.3. Pin diagram and Pin description of 8085

8085 is a 40 pin IC, DIP package. The signals from the pins can be grouped as follows

(i) Power supply and clock signals
(ii) Address bus
(iii) Data bus
(iv) Control and status signals
(v) Interrupts and externally initiated signals
(vi) Serial I/O ports

(i) Power supply and Clock frequency signals:

- Vcc + 5 volt power supply
- Vss Ground
- X1, X2 : Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
- The frequency is internally divided by two. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected externally.
- CLK (output)-Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

(ii) Address Bus:

- A8 - A15 (output; 3-state). It carries the most significant 8 bits of the memory address or the 8 bits of the I/O address;

(iii). Multiplexed Address / Data Bus:

- AD0 - AD7 (input/output; 3-state). These multiplexed set of lines used to carry the lower order 8 bit address as well as data bus. During the opcode fetch operation, in the first clock cycle, the lines deliver the lower order address A0 - A7.
- In the subsequent IO / memory, read / write clock cycle the lines are used as data bus.
- The CPU may read or write out data through these lines.

(iv). Control and Status signals:

- ALE (output) - Address Latch Enable. This signal helps to capture the lower order address presented on the multiplexed address / data bus.
- RD (output 3-state, active low) - Read memory or IO device. This indicates that the selected memory location or I/O device is to be read and that the data bus is ready for accepting data from the memory or I/O device.

- WR (output 3-state, active low) - Write memory or IO device. This indicates that the data on the data bus is to be written into the selected memory location or I/O device.
- IO/M (output) - Select memory or an IO device. This status signal indicates that the read / write operation relates to whether the memory or I/O device. It goes high to indicate an I/O operation.It goes low for memory operations.

(v). Status Signals:

- It is used to know the type of current operation of the microprocessor.

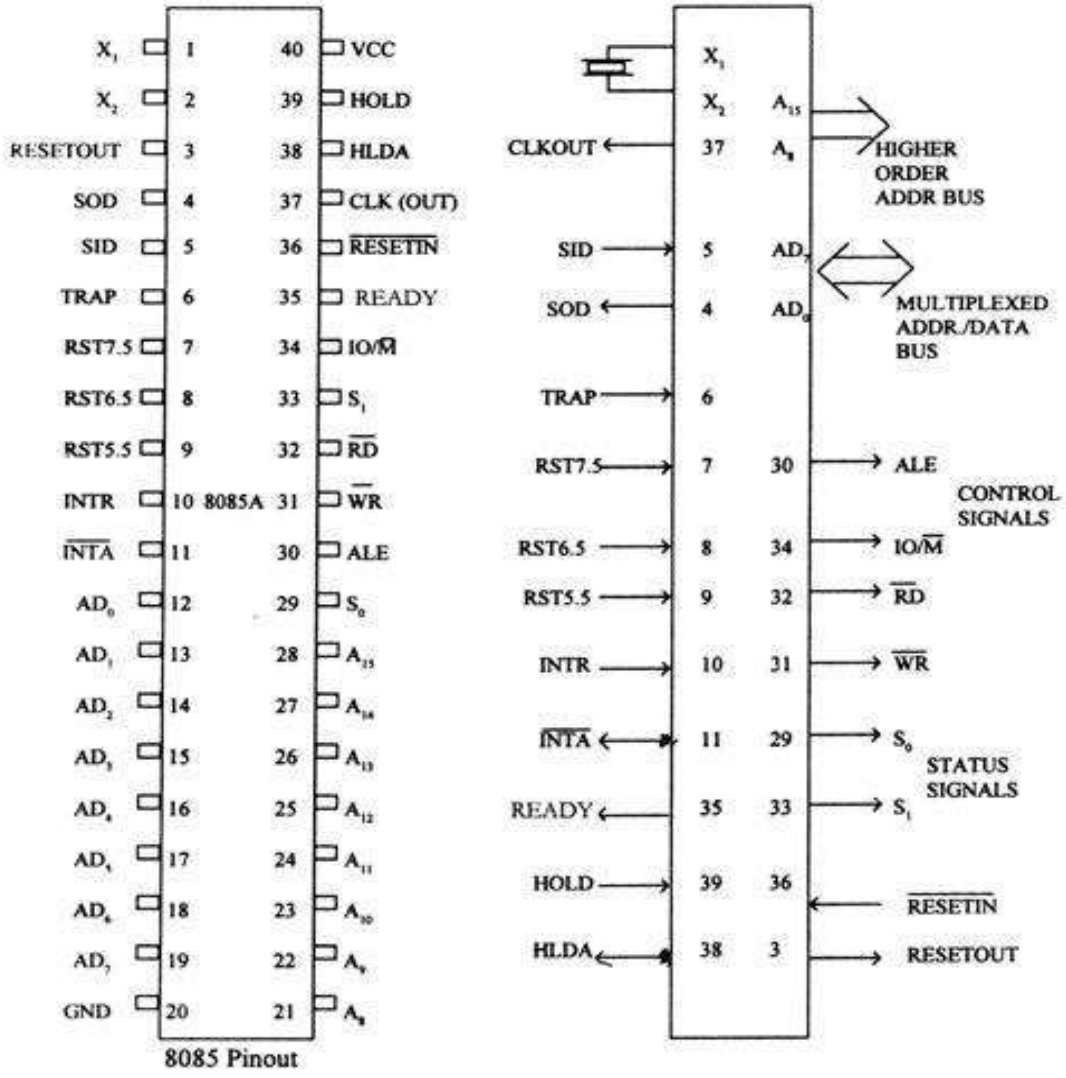


Fig 1.1(a) - Pin Diagram of 8085 & Fig.1.1(b) - logical schematic of Pin diagram.

IO/M(Active Low)	S1	S2	Data Bus Status (Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

(vi). Interrupts and Externally initiated operations:

- They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
- There are five hardware interrupts called,

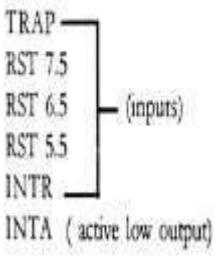


Fig. 1,2

On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.

READY (input)

- Memory and I/O devices will have slower response compared to microprocessors.
- Before completing the present job such a slow peripheral may not be able to handle further data or control signal from CPU.
- The processor sets the READY signal after completing the present job to access the data.
- The microprocessor enters into WAIT state while the READY pin is disabled.

Direct Memory Access (DMA):

Tri state devices:

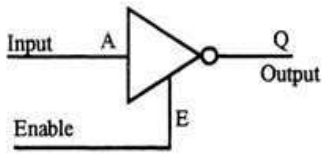


Fig. 1.3

- 3 output states are high & low states and additionally a high impedance state. When enable E is high the gate is enabled and the output Q can be 1 or 0 (ifA is 0, Q is 1, otherwise Q is 0). However, when E is low the gate is disabled and the output Q entersinto a high impedance state.

E	A	Q	State
1(high)	0	1	High
1	1	0	Low
0(low)	0	0	High impedance
0	1	0	High impedance

.

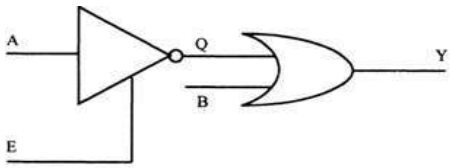


Fig. 1.4

- For both high and low states, the output Q draws a current from the input of the OR gate.
- When E is low, Q enters a high impedance state; high impedance means it is electrically isolated from the OR gate's input, though it is physically connected. Therefore, it does not

draw any current from the OR gate's input.

- When 2 or more devices are connected to a common bus, to prevent the devices from interfering with each other, the tristate gates are used to disconnect all devices except the one that is communicating at a given instant.
- The CPU controls the data transfer operation between memory and I/O device. Direct Memory Access operation is used for large volume data transfer between memory and an I/O device directly.
- The CPU is disabled by tri-stating its buses and the transfer is effected directly by external control circuits.
- HOLD signal is generated by the DMA controller circuit. On receipt of this signal, the microprocessor acknowledges the request by sending out HLDA signal and leaves out the control of the buses. After the HLDA signal the DMA controller starts the direct transfer of data.

(vi). Single Bit Serial I/O ports:

- SID (input) - Serial input data line
- SOD (output) - Serial output data line
- These signals are used for serial communication.

1.4. Bus Structure of 8085 Microprocessor : There are three buses in Microprocessor:

1. Address Bus 2. Data Bus . 3. Control Bus

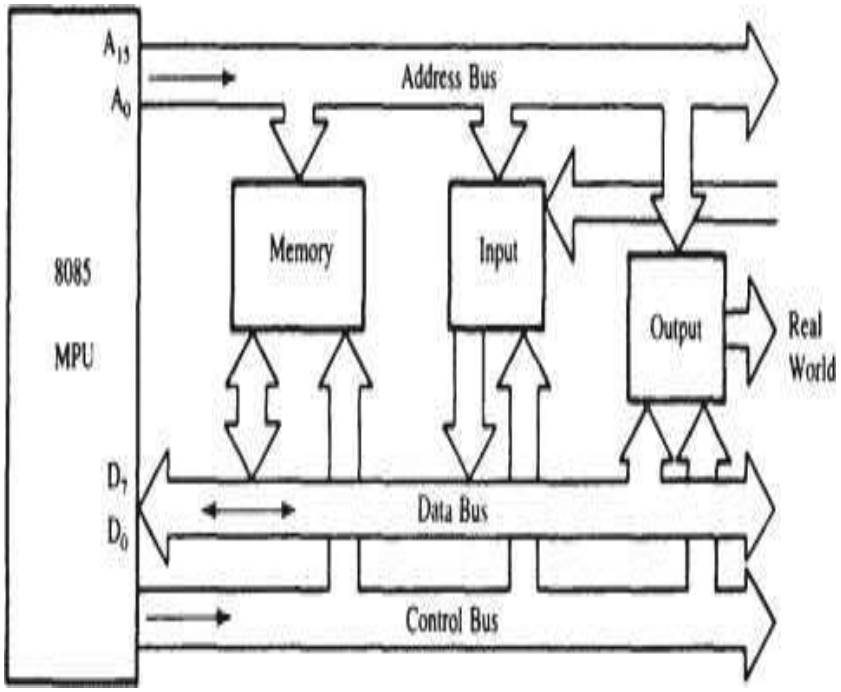


Fig. 1.5: Bus Structure

1.**Address Bus**:- Genearlly, Microprocessor has 16 bit address bus. The bus over which the CPU sends out the address of the memory location is known as Address bus. The address bus carries the address of memory location to be written or to be read from. The address bus is

unidirectional. It means bits flowing occurs only in one direction, only from microprocessor to peripheral devices.

2. **Data Bus**:- 8085 Microprocessor has 8 bit data bus. So it can be used to carry the 8 bit data starting from 00000000H(00H) to 11111111H(FFH). Here 'H' tells the Hexadecimal Number. It is bidirectional. These lines are used for data flowing in both direction means data can be transferred or can be received through these lines. The data bus also connects the I/O ports and CPU. The largest number that can appear on the data bus is 11111111.

3.**Control Bus:-**The control bus is used for sending control signals to the memory and I/O devices. The CPU sends control signal on the control bus to enable the outputs of addressed memory devices or I/O port devices. Some of the control bus signals are as follows:

(i).Memory read (ii) . Memory write (iii). I/O read (iv). I/O write.

1.5. Architecture of 8085 Microprocessor :

The Functional Block Diagram of 8085 Microprocessor is given below:

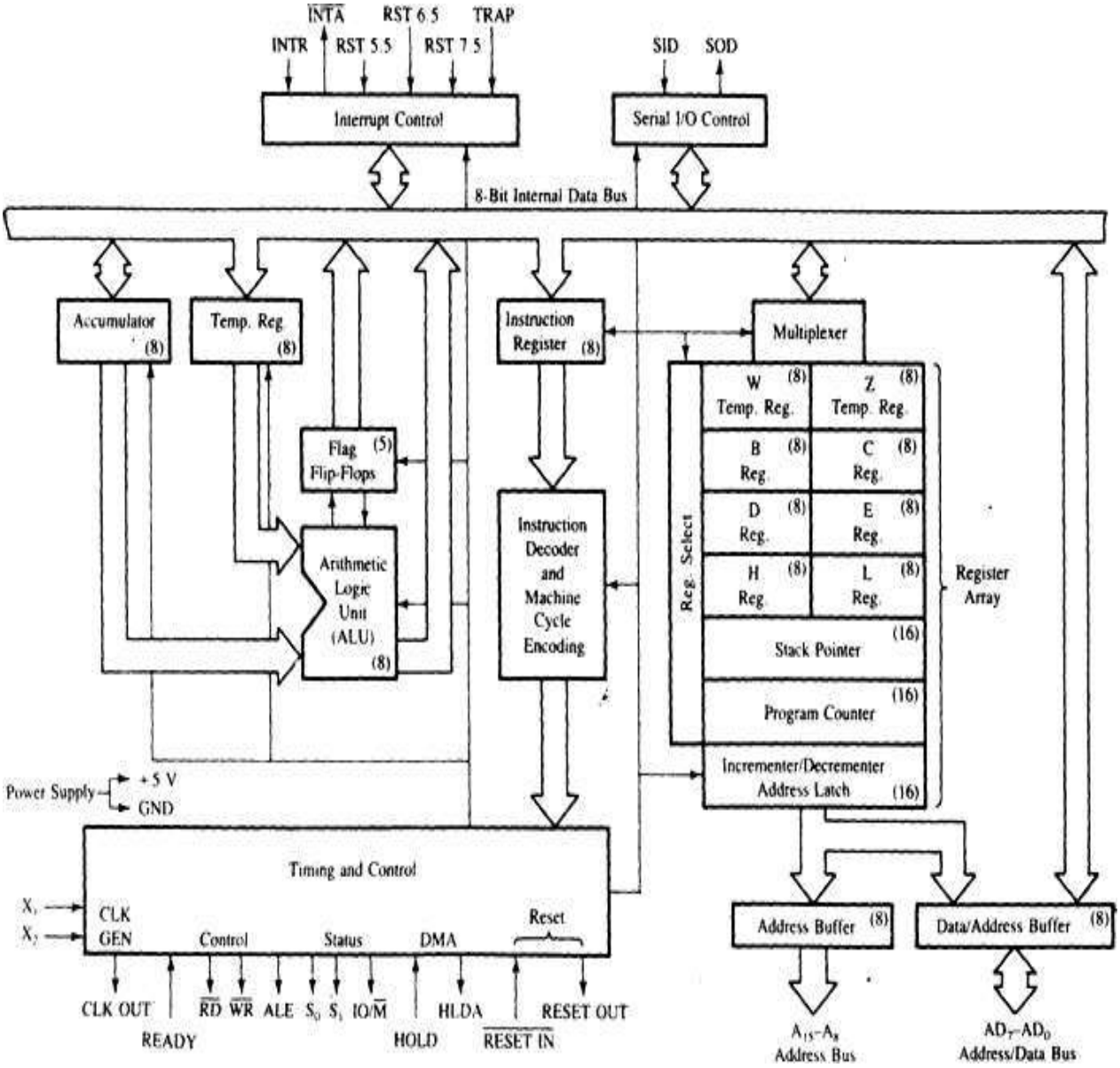


Fig. 1.6 Architecture of 8085

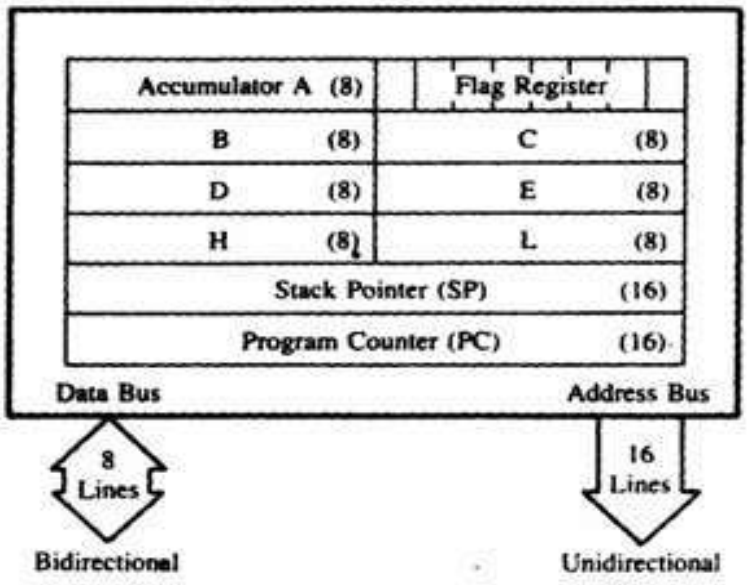


Fig.1.7

Accumulator:- It is a 8-bit register which is used to perform airthmetical and logical operation. It stores the output of any operation. It also works as registers for i/o accesses.

Temporary Register:- It is a 8-bit register which is used to hold the data on which the acumulator is computing operation. It is also called as operand register because it provides operands to ALU.

Registers:- These are general purposes registers. Microprocessor consists 6 general purpose registers of 8-bit each named as B, C, D, E,H and L. Generally theses registers are not used for storing the data permanently. It carries the 8-bits data. These are used only during the execution of the instructions. These registers can also be used to carry the 16 bits data by making the pair of 2 registers. The valid register pairs available are BC, DE HL. We can not use the pairs except BC, DE and HL. These registers are programmed by user.

A**LU:-**ALU performs the airthmetic operations and logical operation.

Flag Registers:-It consists of 5 flip flop which changes its status according to the result stored in

an accumulator. It is also known as status registers. It is connected to the ALU. There are five flip-flops in the flag register are as follows:

1.Sign(S) 2.Zero (Z) 3.Auxiliary carry (AC) 4.Parity (P) 5.Carry (C)

The bit position of the flip flop in flag register is:

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

All of the three flip flop set and reset according to the stored result in the accumulator.

1. Sign- If D7 of the result is 1 then sign flag is set otherwise reset. As we know that a number on the D7 always desides the sign of the number.

if D7 is 1: the number is negative.

if D7 is 0: the number is positive.

2. Zeros (Z)-If the result stored in an accumulator is zero then this flip flop is set otherwise it is reset.

3.Auxiliary carry(AC)-If any carry goes from D3 to D4 in the output then it is set otherwise it is reset.

4.Parity(P)-If the no of 1's is even in the output stored in the accumulator then it is set otherwise it is reset for the odd.

5.Carry(C)-If the result stored in an accumulator generates a carry in its final output then it is set otherwise it is reset.

Instruction registers(IR):-It is a 8-bit register. When an instruction is fetched from memory then it is stored in this register.

Instruction Decoder:- Instruction decoder identifies the instructions. It takes the informations from instruction register and decodes the instruction to be performed.

Program Counter:-It is a 16 bit register used as memory pointer. It stores the memory address of the next instruction to be executed. So we can say that this register is used to sequencing the

program. Generally the memory have 16 bit addresses so that it has 16 bit memory.
The program counter is set to 0000H.

Stack Pointer:-It is also a 16 bit register used as memory pointer. It points to the memory location called stack. Generally stack is a reserved portion of memory where information can be stores or taken back together.

Timing and Control Unit:-It provides timing and control signal to the microprocessor to perform the various operation.It has three control signal. It controls all external and internal circuits. It operates with reference to clock signal.It synchronizes all the data transfers.

There are three control signal:

1. *ALE*-Airthmetic Latch Enable, It provides control signal to synchronize the components of microprocessor.
2. *RD*- This is active low used for reading operation.
3. *WR*-This is active low used for writing operation. There are three status signal used in microprocessor S0, S1 and IO/M. It changes its status according the provided input to these pins.

IO/M(Active Low)	S1	S2	Data Bus Status(Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

Serial Input Output Control-There are two pins in this unit. This unit is used for serial data communication.

Interrupt Unit-There are 6 interrupt pins in this unit. Generally an external hardware is connected to these pins. These pins provide interrupt signal sent by external hardware to microprocessor and microprocessor sends acknowledgement for receiving the interrupt signal. Generally INTA is used for acknowledgement.

8085- Registers

The 8085 has six general purpose registers to store 8 bit data. These are identified as B, C, D, E,

H, L. they can be combined as register pairs BC, DE, and HL, to perform 16 bit operations.

Accumulator

The acc is an 8 bit register that is part of the arithmetic logic unit [ALU]. This register is used to store 8 bit data and to perform arithmetic and logical operations. The result of the operation is stored in the accumulator and identified as A.

Flags

The arithmetic logic unit [ALU] includes 5 flip flops which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called zero (Z), carry (CY), sign(S), parity (P), and auxiliary carry (AC). The microprocessor used these flags to test data conditions.

Program counter

The microprocessor uses the PC register to sequence the execution of the instructions. The function of the PC is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the pc is increased by one to point to the next memory location.

Stack pointer

The SP is also a 16 bit register used as a memory pointer. It points to a memory location in R/W memory, called the *stack*

Interrupts In 8085

What is Interrupt?

Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced. Generally, a particular task is assigned to that interrupt signal. In the microprocessor based system the interrupts are used for data transfer between the peripheral devices and the microprocessor.

Interrupt Service Routine(ISR)

Interrupt means to break the sequence of operation. While the CPU is executing a program an interrupt breaks the normal sequence of execution of instructions & diverts its execution to some other program. This program to which the control is transferred is called the *interrupt service routine.* A small program or a routine that when executed services the corresponding interrupting source is called as an ISR.

Execution of Interrupts

When there is an interrupt requests to the Microprocessor then after accepting the interrupts Microprocessor send the INTA (active low) signal to the peripheral. The vectored address of particular interrupt is stored in program counter. The processor executes an interrupt service routine (ISR) addressed in program counter.

There are two types of interrupts used in 8085 Microprocessor:

Hardware Interrupts and Software Interrupts

Software Interrupts

A software interrupts is a particular instructions that can be inserted into the desired location in the rpogram. There are eight Software interrupts in 8085 Microprocessor. From RST0 to RST7.

RST0, RST1, RST2, RST3, RST4, RST5, RST6, RST7

They allow the microprocessor to transfer program control from the main program to the subroutine program. After completing the subroutine program, the program control returns back to the main program.

Hardware Interrupts

There are 6 interrupt pins in the microprocessor used as Hardware Interrrupts given below:

TRAP, RST7.5, RST6.5, RST5.5, INTR

Note:

INTA is not an interrupt. INTA is used by the Microprocessor for sending the acknowledgement. TRAP has highest priority and RST7.5 has second highest priority and so on.

TRAP

It is non maskable edge and level triggered interrupt. TRAP has the highest priority and vectored interrupt. Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged. In case of sudden power failure, it executes a ISR and send the data from main memory to backup memory.

As we know that TRAP can not be masked but it can be delayed using HOLD signal. This interrupt transfers the microprocessor's control to location 0024H.

TRAP interrupts can only be masked by reseting the microprocessor. There is no other way to mask it.

RST7.5

It has the second highest priority. It is maskable and edge level triggered interrupt. The vector

address of this interrupt is 003CH. Edge sensitive means input goes high and no need to maintain high state until it is recognized. It can also be reset or masked by reseting microprocessor. It can also be resetted by DI instruction.

RST6.5 and RST5.5

These are level triggered and maskable interrupts. When RST6.5 pin is at logic 1, INTE flip-flop is set. RST 6.5 has third highest priority and RST 5.5 has fourth highest priority.

It can be masked by giving DI and SIM instructions or by reseting microprocessor.

INTR

It is level triggered and maskable interrupt. The following sequence of events occurs when INTR signal goes high. The 8085 checks the status of INTR signal during execution of each instruction. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction. It has the lowest priority. It can be disabled by reseting the microprocessor or by DI and SIM instruction.

UNIT-II

1. 8085 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.

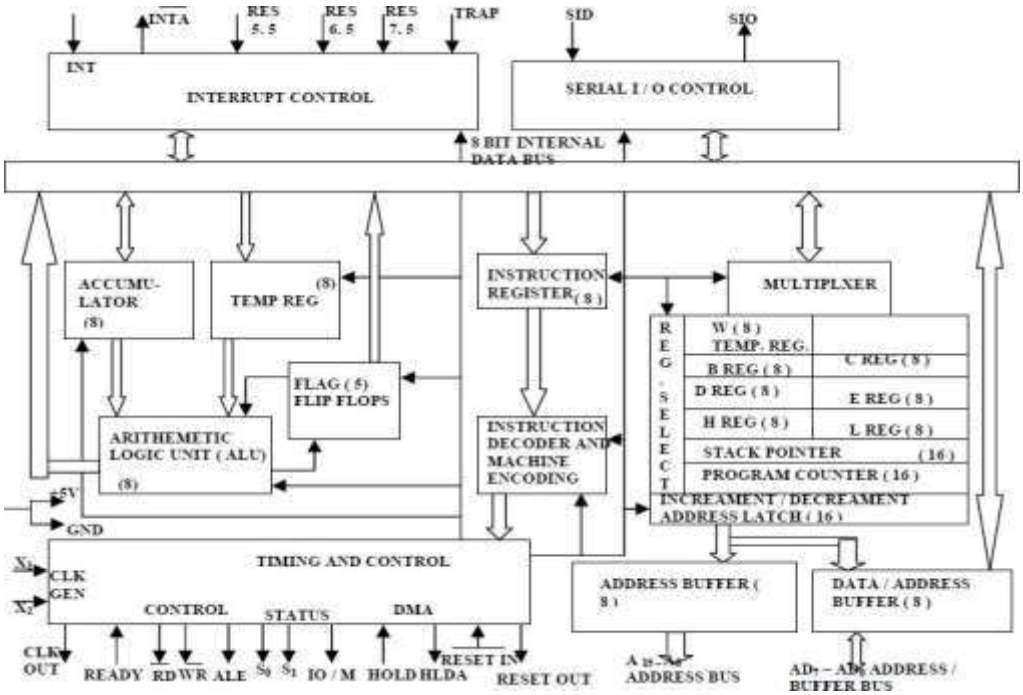


Fig. 2 Internal Architecture of 8085

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

16-bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

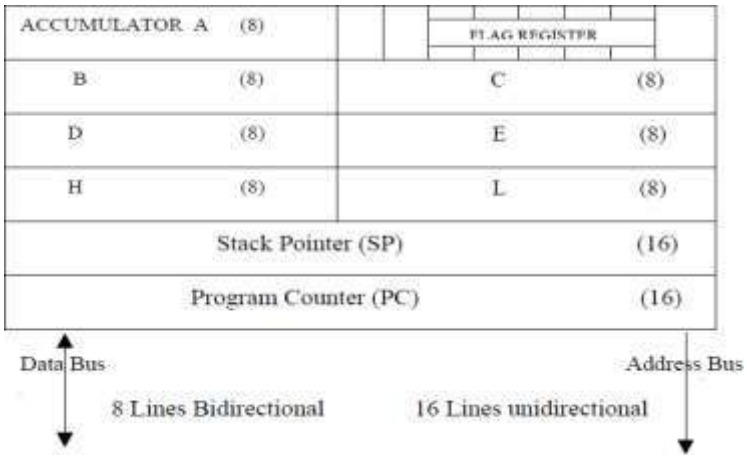


Fig. 3 Register organisation

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.

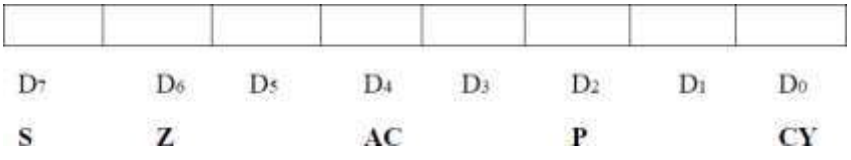


Fig. 4 Flag register

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.

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 $\overline{data}$ bus.

III. 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 1.

Table 1 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. 5. 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.

The 8085 processor performs these functions using address bus, data bus and control bus as shown in Fig. 5.

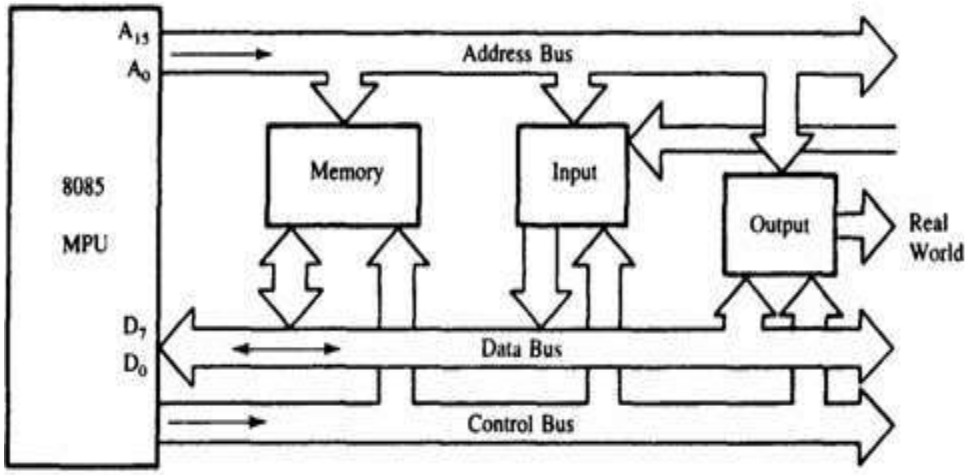


Fig. 5 The 8085 bus structure

2. 8085 PIN DESCRIPTION

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

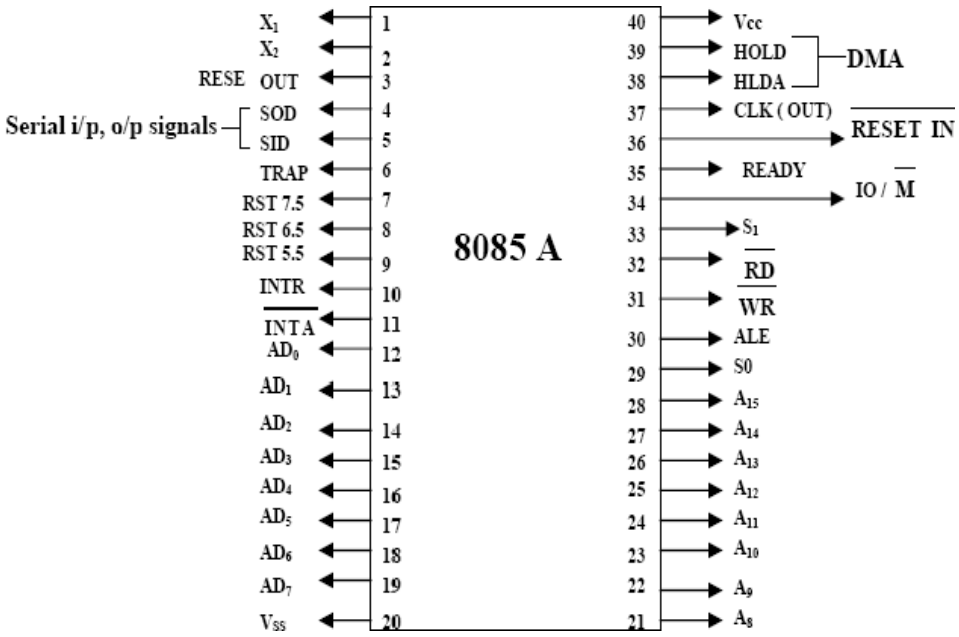


Fig. 6 8085 microprocessor pin layout and signal groups

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:

- $\overline{ALE}$: Address latch enable
- $\overline{RD}$: Read control signal.
- $\overline{WR}$: Write control signal.
- IO/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:

- $\overline{\text{RESET}}$ 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.

- $\overline{\text{INTR}}$: Interrupt request is a general-purpose interrupt.
- $\overline{\text{INTA}}$: 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.
-

3. INSTRUCTION SET AND EXECUTION IN 8085

Based on the design of the ALU and decoding unit, the microprocessor manufacturer provides instruction set for every microprocessor. The instruction set consists of both machine code and mnemonics.

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called instruction set. Microprocessor instructions can be classified based on the parameters such functionality, length and operand addressing.

Classification based on functionality:

I. Data transfer operations: This group of instructions copies data from source to destination. The content of the source is not altered.
II. Arithmetic operations: Instructions of this group perform operations like addition, subtraction, increment & decrement. One of the data used in arithmetic operation is stored in accumulator and the result is also stored in accumulator.
III. Logical operations: Logical operations include AND, OR, EXOR, NOT. The operations like AND, OR and EXOR uses two operands, one is stored in accumulator and other can be any register or memory location. The result is stored in accumulator. NOT operation requires single operand, which is stored in accumulator.
IV. Branching operations: Instructions in this group can be used to transfer program sequence from one memory location to another either conditionally or unconditionally.
V. Machine control operations: Instruction in this group control execution of other instructions and control operations like interrupt, halt etc.

Classification based on length:

I. One-byte instructions: Instruction having one byte in machine code. Examples are depicted in Table 2.
I. Two-byte instructions: Instruction having two byte in machine code. Examples are depicted in Table 3

II. Three-byte instructions: Instruction having three byte in machine code. Examples are depicted in Table 4.

Table 2 Examples of one byte instructions

Opcode	Operand	Machine code/Hex code
MOV	A, B	78
ADD	M	86

Table 3 Examples of two byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
MVI	A, 7FH	3E	First byte
		7F	Second byte
ADI	0FH	C6	First byte
		0F	Second byte

Table 4 Examples of three byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
JMP	9050H	C3	First byte
		50	Second byte
		90	Third byte
LDA	8850H	3A	First byte
		50	Second byte
		88	Third byte

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 is mentioned in the instruction.

4. INSTRUCTION EXECUTION AND TIMING 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. 7.

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/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.

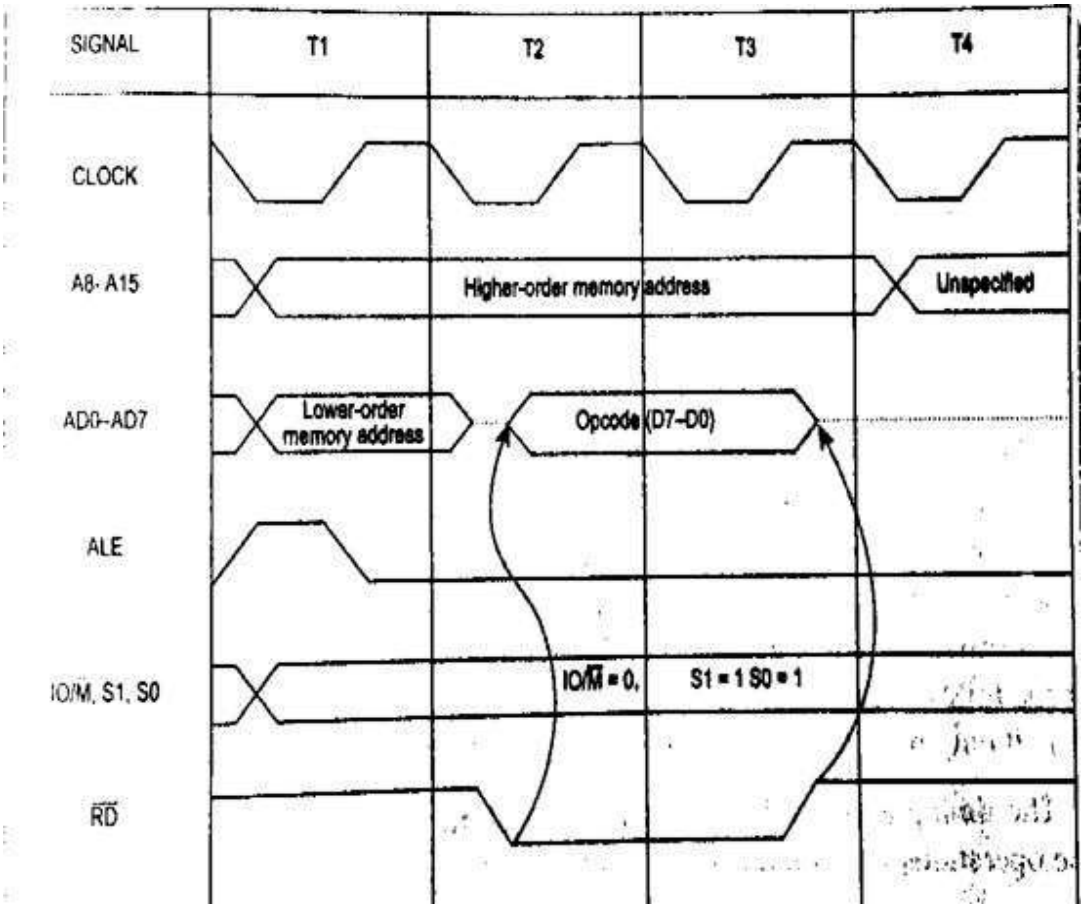


Fig. 7 Timing diagram for opcode fetch cycle

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. 8.

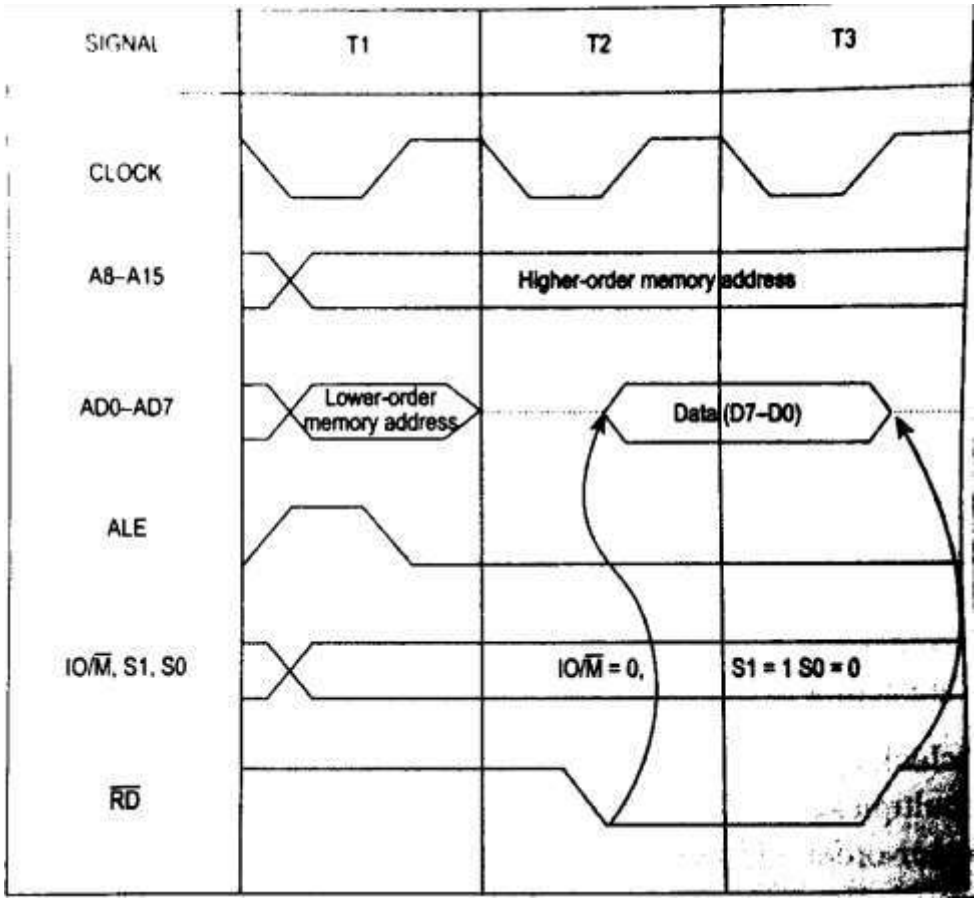


Fig. 8 Timing diagram for memory read machine cycle

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.

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. 10.

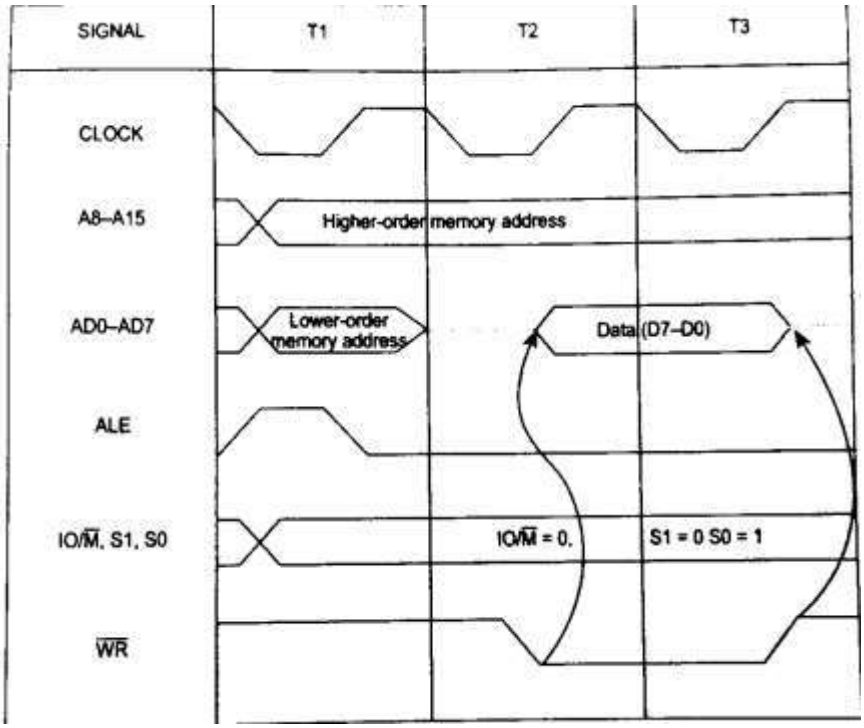


Fig. 9 Timing diagram for memory write machine cycle

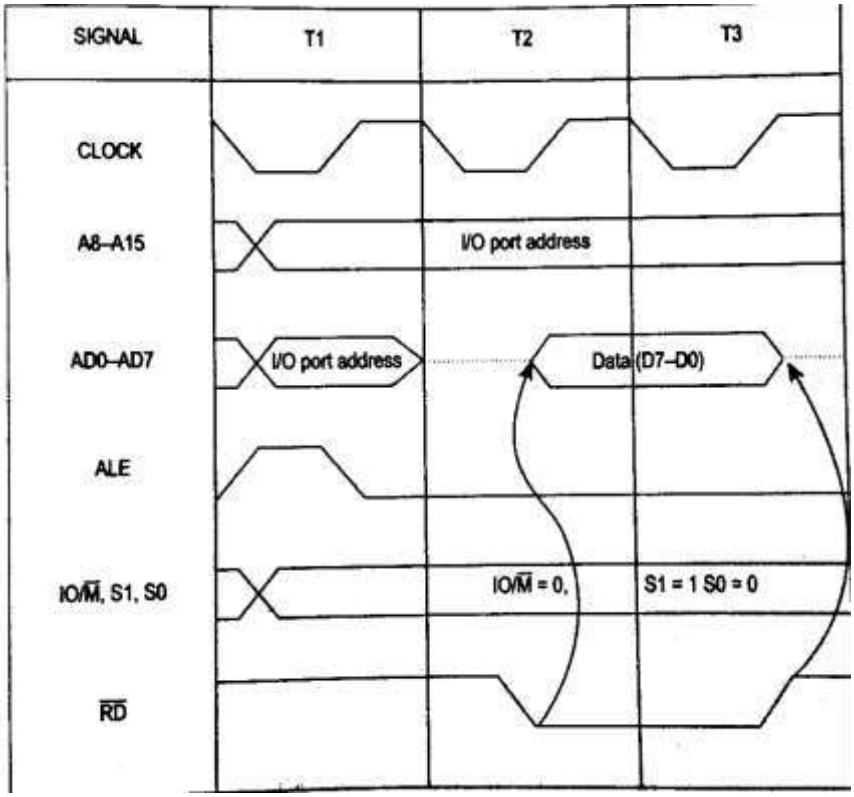


Fig. 10 Timing diagram I/O read machine cycle

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. 11.

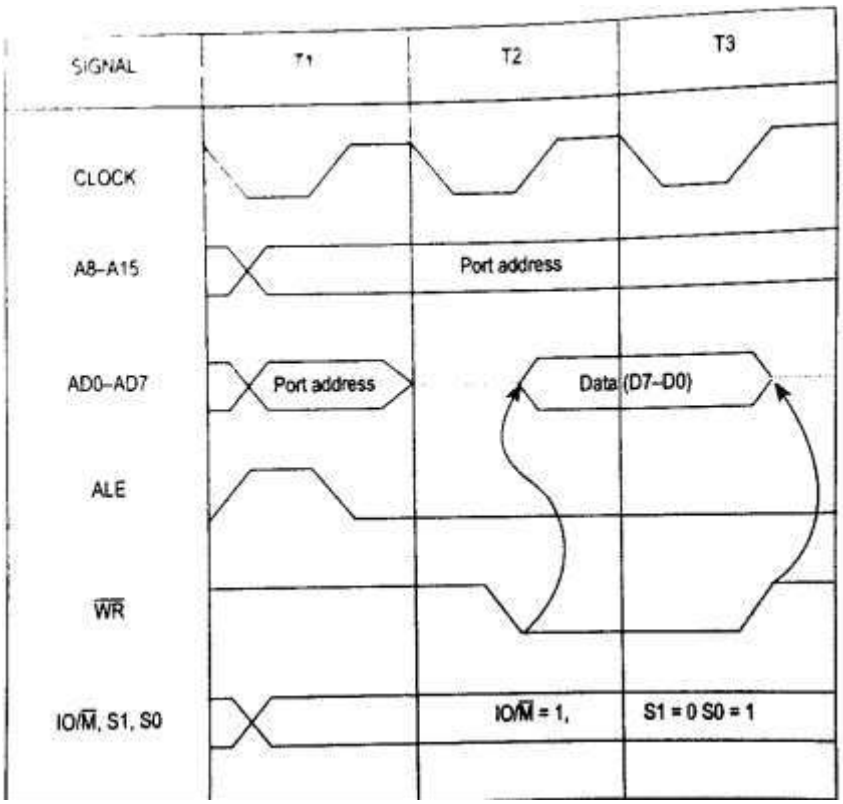


Fig. 11 Timing diagram I/O write machine cycle

Ex: Timing diagram for IN 80H.

The instruction and the corresponding codes and memory locations are given in Table 5.

Table 5 IN instruction

Address	Mnemonics	Opcode
800F	IN 80H	DB
8010		80

i. During the first machine cycle, the opcode DB is fetched from the memory, placed in the instruction register and decoded.

ii. During second machine cycle, the port address 80H is read from the next memory location.

iii. During the third machine cycle, the address 80H is placed in the address bus and the data read from that port address is placed in the accumulator.he timing diagram is shown in Fig. 12.

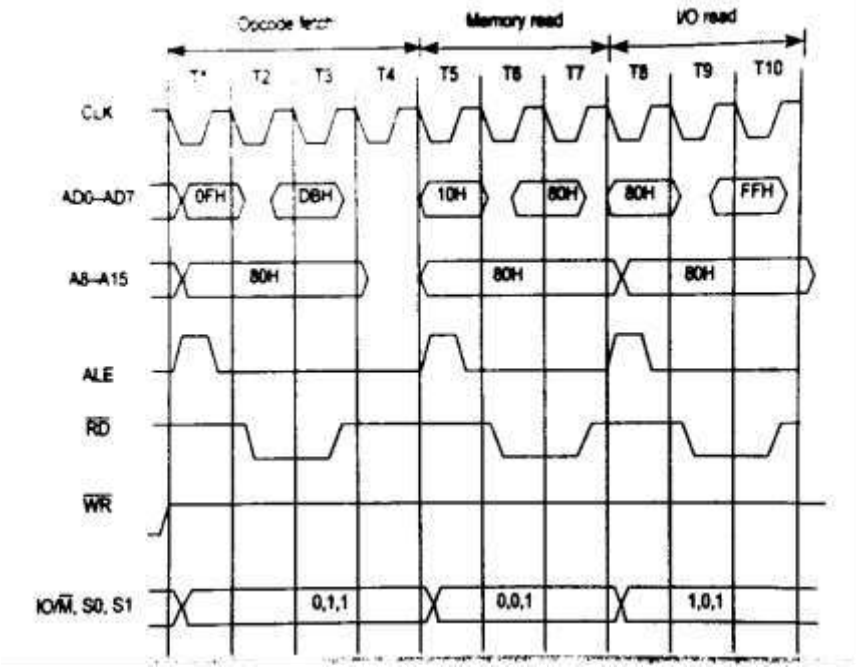


Fig. 12 Timing diagram for the IN instruction

5. *8085 INTERRUPTS*

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 6.

Table 6 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 7.

Table 7 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.

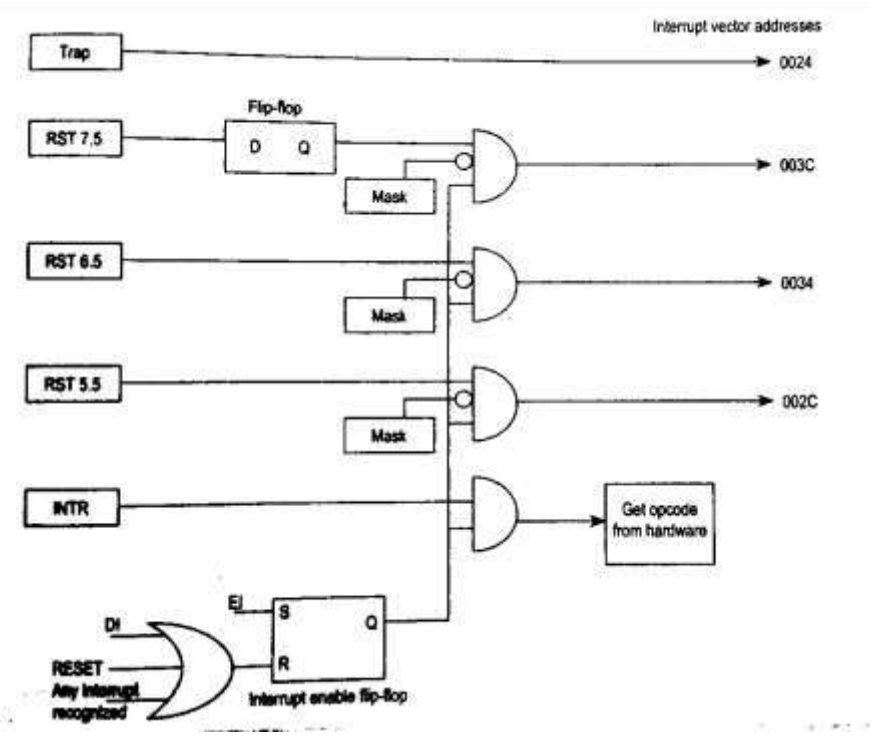


Fig. 13 Interrupt structure of 8085

The Fig. 13 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 word in 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. 14.

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. 14 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. 15.

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. 15 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.

Unit – III

3.1 What is meant by memory address space?

It is the maximum possible memory size which can be used with µp.

(i) Address space partitioning:

The allocation of address to memory chips and I/O devices depends on the µp architecture. Some processors provide only one address space thereby treating I/O devices as memory locations. Intel 8085 uses a 16- bit wide address bus for addressing memories and I/O devices, using 16- bit wide address bus it can access 2^{16} = 64 bytes of memory and I/O devices. The 64k address are to be assigned to memories and I/O devices for their addressing. There are two schemes for the allocation of address to memories and input/output devices.

Arithmetic Operators

The basic **arithmetic operations** are addition, subtraction, multiplication, and division. Arithmetic is performed according to an order of operations.An operator performs an action on one or more operands. The common arithmetic operators are:

Action	Common Symbol

Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus (associated with integers)	%

These arithmetic operators are binary that is they have two operands. The operands may be either constants or variables.This expression consists of one operator (addition) which has two operands. The first is represented by a variable named age and the second is a literal constant. If age had a value of 14 then the expression would evaluate (or be equal to) 15.These operators work as you have learned them throughout your life with the exception of division and modulus. We normally think of division as resulting in an answer that might have a fractional part (a floating-point data type). However, division, when both operands are of the integer data type, may act differently. Please refer to the next section on "Integer Division and Modulus".

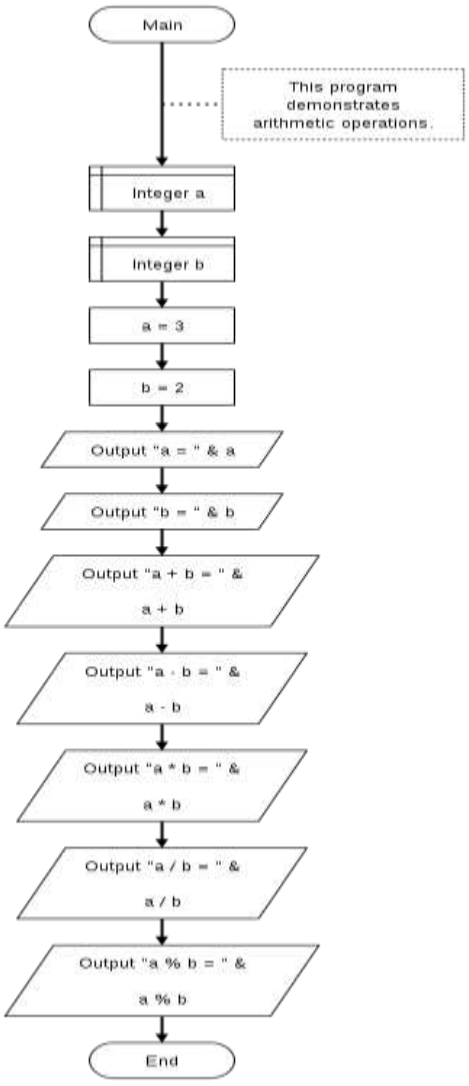
Arithmetic Assignment Operators

Many programming languages support a combination of the assignment (=) and arithmetic operators (+, -, *, /, %). Various textbooks call them "compound assignment operators" or "combined assignment operators". Their usage can be explained in terms of the assignment operator and the arithmetic operators. In the table, we will use the variable age and you can assume that it is of integer data type.

Arithmetic assignment examples:	**Equivalent code:**
`age += 14;`	`age = age + 14;`
`age -= 14;`	`age = age - 14;`

`age *= 14;`	`age = age * 14;`
`age /= 14;`	`age = age / 14;`
`age %= 14;`	`age = age % 14;`

Flowchart


Main
This program demonstrates arithmetic operations.
Integer a
Integer b
a = 3
b = 2
Output "a = " & a
Output "b = " & b
Output "a + b = " & a + b
Output "a - b = " & a - b
Output "a * b = " & a * b
Output "a / b = " & a / b
Output "a % b = " & a % b
End

1. Multiplication of two 8-bit numbers

Example: Multiply two 8-bit numbers stored at memory locations 1050 and 1051 and store the result in memory location 1052.

Memory location	Label	Mnemonics	Opcode
1000		MVI B, 00	06
1001			00
1002		LHLD 1050	2A
1003			50
1004			10
1005		XCHG	EB
1006		LHLD 1052	2A
1007			52
1008			10
1009		CALL SUB1	CD
100A			1C
100B			10
100C		DAD D	19
100 D		JC LOOP1	DA
100E			14
100F			10
1010		CALL SUB1	CD
1011			1C
1012			10
1013		INR B	04
1014	LOOP1	SHLD 2500	22

1015			00
1016			25
1017		MOV A,B	78
1018		SHLD 2502	22
1019			02
101A			25
101B		HLT	76
101C	SUB1	MOV A,L	7D
101D		CMA	2F
101E		MOV L,A	6F
101F		MOVA,H	7C
1020		CMA	2F
1021		MOV H,A	67
1022		INX H	23
1023		RET	C9

2. Division of two 8-bit numbers

Example: Write an ALP to divide two 8-bit numbers

Label	Mnemonics	Comment
	LDA 6501 H	Load the divisor in accumulator
	MOV B, A	Move the divisor to B register
	LDA 6500 H	Load the dividend in accumulator
	MVI C, 00H	Clear C register to store quotient
STEP 1:	CMP B	Compare the content of A and B
	JC STEP 2	If divisor < dividend , go to STEP 2
	SUB B	Subtract divisor from dividend
	INR C	Increment quotient
	JMP STEP 1	continue the subtraction
STEP 2	STA 6503H	Store the accumulator
	MOV A, C	Move the content of C to A
	STA 6502 H	Store the quotient

3. Ascending order – Sorting of Numbers

Example: Write an ALP to arrange the data bytes in acending order (sorting of numbers)

Memory Address	Label	Mnemonics	Comment
4100H		LDA 4300 H	Load the number of passes from memory into acc.
4103 H		MOV B, A	Move the data from the acc. to register B
4104 H	LOC 5:	MOV C, B	Move the data from register B into the register C
4105 H		LXI H 4400 H	Load H-L pair with memory address
4108 H	LOC 3:	MOV A, M	Move the data from memory location to acc.
4109 H		INX H	Increment the content of H-L pair

410A H		CMP M	Compare the content of M with acc.
410B H		JC LOC 1	If carry = 1, Go to LOC 1.
410E H		MOV D, M	The data in memory pointed by H-L pair is transferred to register D
4110 H		DCX H	Decrement the content of H-L pair
4111 H		MOV M, D	Move the data in register D to H-L pair
4112 H		INX H	Increment the content of the H-L pair
4113 H	LOC 1:	DCR C	Decrement the register C
4114 H		JZ LOC 2	If the counter ≠ 0, jump to LOC 3
4117 H		JMP LOC 3	If the counter = 0, jump to LOC 2
411A H	LOC 2:	DCR B	Decrement register B (pass counter)
411B H		JZ LOC 4	If the counter = 0, jump to LOC 4
411E H		JMP LOC 5	If the counter ≠ 0, jump to LOC 5

4121 H	LOC 4:	HLT	Stop

4. To find Two's compliment

Example: Write an ALP to find Two‟s compliment of a 16-bit number

Label	Mnemonics	Comment
	LXI H, 6501 H	Address of 8 LSBs of the given no.
	MOV B, 00 H	Clear B register
	MOV A, M	Move 8 LSBs to accumulator
	CMA	1‟s compliment of 8 LSBs
	ADI 01 H	2‟s compliment of 8 LSBs
	STA 6503 H	Store 8 LSBs of the result
	JNC STEP	Jump on no carry to "STEP"
	INR B	If carry is available, store in B
STEP:	INX H	Address of 8 MSBs of the given no.
	MOV A, M	Move 8 MSBs to accumulator
	CMA	1‟s compliment
	ADD B	Add carry
	STA 6504 H	Store 8 MSBs of the result
	HLT	Halt

UNIT-IV

Execution Unit (EU):

The EU contains (i) ALU (ii) General purpose registers (iii) Index registers (iv) pointers .

Execution unit receives program instruction codes and data from the BIU, executes them and stores the results in the general registers. It can also store the data in a memory location or send them to an I/O device by passing the data back to the BIU. This unit, EU, has no connection with the system Buses. It receives and outputs all its data through BIU.

ALU (Arithmetic and Logic Unit) : The EU unit contains a circuit board called the Arithmetic and Logic Unit. This unit can perform various arithmetic and logical operation, if required, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, convert byte/word and compare etc and logical operations, such as AND, OR, exclusive OR, shift/rotate and test etc.

Registers : A register is like a memory location where the exception is that these are denoted by name rather than numbers. It has 4 data registers, AX, BX, CX, DX and 2 pointer registers SP, BP and 2 index registers SI, DI and 1 temporary register and 1 status register FLAGS .

AX, BX, CX and DX registers has 2 8-bit registers to access the high and low byte data registers. The high byte of AX is called AH and the low byte is AL. Similarly, the high and low bytes of BX, CX, DX are BH and BL, CH and Cl, DH and DL respectively. All the data, pointer, index and status registers are of 16 bits. Else these, the temporary register holds the operands for the ALU and the individual bits of the FLAGS register reflect the result of a computation.

Bus Interface Unit:

The BIU contains (i) Segment registers (ii) Instruction registers (iii) Instruction queue and

(v) Flag register

As the EU has no connection with the system Busses, this job is done by BIU. BIU and EU are connected with an internal bus. BIU connects EU with the memory or I/O circuits. It is responsible for transmitting data, addresses and control signal on the busses.

Registers : BIU has 4 segment busses, CS, DS, SS, ES. These all 4 segment registers holds the addresses of instructions and data in memory. These values are used by the processor to access memory locations. It also contain 1 pointer register IP. IP contains the address of the next instruction to executed by the EU.

Instruction Queue : BIU also contain an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. Also when

the EU needs to be connected with memory or peripherals, BIU suspends instruction prefetch and performs the needed operations

Purpose of using Instruction Queue:

BIU contains an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. A subtle advantage of instruction queue is that, as next several instructions are usually in the queue, the BIU can access memory at a somewhat "leisurely" pace. This means that slow-memory parts can be used without affecting overall system performance.

5.2 Instruction Set of Intel 8085 Microprocessor

Instruction set of 8085 microprocessor can be divided into data copy/transfer instructions, arithmetic and logical instructions, branch/loop instructions, machine control instructions, flag manipulation instructions, string manipulation instructions. Instruction set refers to the instructions which can be used to program a microprocessor etc,. Instruction set can be divided into data copy/transfer instructions, arithmetic and logical instructions, branch/loop instructions, machine control instructions, flag manipulation instructions, string manipulation instructions. The instruction set of 8085 microprocessor is:

(i) Data Copy/Transfer Instructions

These are the type of instructions used to copy, move etc., data from source to destination. Some of the data copy/transfer instructions are:

MOV : Move data from register to register, memory to register, register to memory, memory to accumulator, accumulator to memory etc,.

PUSH : Push data into register, memory etc,.

POP : Pop data from register, memory etc,.

XCHG : Exchange data between register, memory etc,.

IN : Input from fixed port or variable port

OUT : Output to fixed port or variable port

LDS : Load pointer to DS

LES : Load pointer to ES

LAHF : Load AH with flags

SAHF : Store AH into flags

PUSHF : Push flags

POPF : Pop flags

(ii) Arithmetic and Logical Instructions

These are the type of instructions used to perform arithmetic operations like addition, subtraction etc., and logical operations like and, or etc,. Some of the arithmetic and logical instructions are :

(a) Arithmetic Instructions

ADD : Addition
ADC : Addition with Carry
INC : Increment by 1
AAA : ASCII Adjust for Addition
DAA : Decimal Adjust for Addition
SUB : Subtraction
SBB : Subtraction with Borrow
DEC : Decrement by 1
AAS : ASCII Adjust for Subtraction
DAS : Decimal Adjust for Subtraction
MUL : Unsigned Multiplication
IMUL : Signed Multiplication
AAM : ASCII Adjust for Multiplication
DIV : Unsigned Division
IDIV : Signed Division
AAD : ASCII Adjust for Division
NEG : Change Sign
CMP : Compare
CBW : Convert Byte to Word
CWD : Convert Word to Double Word

(b) Logical Instructions

AND : Logical AND
OR : Logical OR
NOT : Logical NOT
XOR : Logical XOR
SHL : Shift Logical Left
SHR : Shift Logical Right

ROL : Rotate Left

ROR : Rotate Right

RCL : Rotate Left through Carry Flag

RCR : Rotate Right through Carry Flag

(iii) Branch/Loop Instructions

These are the type of instructions used to control the transfer to a specified address. Some of the branch/loop instructions are:

(a) Unconditional Branch/Loop Instructions

CALL : Call a subroutine Unconditionally

RET : Return from a procedure

INTN : Interrupt of Type N

INTO : Interrupt on Over flow

LOOP : Loop instructions Unconditionally

(b) Conditional Branch/Loop Instructions

JZ : Jump if zero

JE : Jump if equal

JNZ : Jump if not zero

JNE : Jump if not equal

JL : Jump if lesser

JLE : Jump if lesser or equal

JG : Jump if greater

JGE : Jump if greater or equal

JO : Jump on Over flow

JNO : Jump on not Over flow

JS : Jump on Sign

JNS : Jump on not Sign

LOOPZ : Loop if zero

LOOPE : Loop if equal

LOOPNZ : Loop if not zero

LOOPNE : Loop if not equal

(iv) Machine Control Instructions

These are type of instructions used to control machine status. Some of the machine control instructions are:

WAIT : Wait for the test input to go low

HLT : Halt the processor

NOP : No operation

ESC : Escape to external device

LOCK : Lock instruction prefix

(v) Flag manipulation Instructions

These are the type of instructions used to manipulate different flags present in the flag register of 8086 microprocessor. Some of the flag manipulation instructions are:

CLC : Clear Carry Flag

STC : Set Carry Flag

CLD : Clear Direction Flag

STD : Set Direction Flag

CLI : Clear Interrupt Flag

STI : Set Interrupt Flag

(vi) String Manipulation Instructions

These are the type of instructions used to manipulate strings. Some of the string manipulation operations are:

REP : Repeat Instruction Prefix

REPE : Repeat if equal

REPZ : Repeat if zero

REPNE : Repeat if not equal

REPNZ : Repeat if not zero

MOVS : Move String Byte/Word

CMPS : Compare String Byte/Word

SCAS : Scan String Byte/Word

LODS : Load String Byte/Word

STOS : Store String Byte/Word

5.3 Addressing Modes of 8085

8085 memory addressing modes provide flexible access to memory, allowing you to easily access variables, arrays, records, pointers, and other complex data types. 12 addressing modes classified in 5 groups

(1) Addressing modes for register and immediate data

(i). Register addressing: - the instruction will specify the name of the register which holds the data to be operated by the instruction

Ex: MOV CL, DH : content of 8-bit DH register is moved to another 8-bit register CL

(ii). Immediate addressing: - an 8-bit or 16-bit data is specified as a part of the instruction

Ex: MOV DL, 08H : The 8-bit data (08H) given in the instruction is moved to DL register

1. Addition of two 16-bit data:

Label	Mnemonics	Comments
	MOV AX, DATA 1	Load the first data in AX
	MOV CL, 00H	register Clear the CL register for carry
	ADD AX, DATA 2	Add 2nd data to AX, sum will be in AX
	MOV 2000H, AX	Store sum in memory location 1
	JNC STEP	Check the status of carry flag
	INC CL	If carry is set: increment CL by one
STEP	MOV 2002H CL	Store carry in memory location 2
	HLT	Halt

2. Subtraction of Two 16-bit data

Label	Mnemonics	Comments

	MOV SI, 2000H MOV AX, [SI] MOV BX, [SI+2]	Load the address of data in SI register Get the minuend in AX register Get the subtrahend in BX register
	MOV CL, 00H	Clear the CL register to account for sign
	SUB AX, BX	Get the difference in AX register
	JNC STEP	Check the status of carry flag
	INC CL	If carry is set: increment CL by one
	NOT AX	Then take 2’s compliment of difference
	ADD AX, 0001H MOV	
STEP	[SI+4], AX	Store the difference in memory location 1
	MOV [SI+6], CL	Store sign bit in memory location 2

HLT	Halt

3. Multiplication of Two 16-bit data

Label	Mnemonics	comments
	MOV AX, [2000]	Move the first data to AX register from memory
	MUL [2002]	Multiply the data in AX with the data in memory location 2002 H
	MOV [2100], DX	Save the MSW (higher order) of the result in DX Register
	MOV [2102], AX	Save the LSW (ligher order) of the result in AX register
	HLT	Halt

4. Division of Two 32-bit data by 16-bit data

Label	Mnemonics	comments
	MOV DX, [2000]	Move the high order word dividend to DX register
	MOV AX, [2002	Move the lower order word dividend to AX register
	DIV [2004]	Divide the data in DX: AX by the divisor
	MOV [2100], AX	The quotient is stored in AX
	MOV [2102], DX	The remainder is stored in DX
	HLT	Halt

5. Find the sum of the elements in an array

Label	Mnemonics	comments

```
                MOV  SI,  2000H          Set SI register as pointer for
                array MOV   DI,  3000H   Set DI register as pointer for
                result
                MOV   CL,  [SI]          Set CL as count for number of bytes in
                 array INC  SI           Set Si to point to I- st byte of array
                MOV  AX,  0000H          Set initial sum as zero
STEP 1:         DD   AL, [SI]            Add a byte of array to
```

	JNC STEP2	Check for carry flag
	INC AH	If carry flag is set then increment
AH STEP 2:	INC SI	Increment array pointer
	LOOP STEP 1	Repeat addition until count is zero
	MOV [DI], AX	Store the sum in memory
	HLT	Halt

6. Find the largest data of signed numbers

Label	Mnemonics	comments
	MOV AX, @data	Initialize DS
	register MOV DS, AX	
	MOV SI 2000H	Initialize SI register
	MOV BX, 0000H	Initialize maximum number
	MOV CX, 05H	5 numbers to be
processed STEP 2:	MOV AX, [SI]	Load
number from sequence		
	CMP BX, AX	Compare with current maximum
	number JCE	STEP1
	MOV BX, AX	Save new maximum number
	MOV DX, SI	Save location of maximum
number STEP 2:	INC SI	Update pointer
	LOOP STEP 2:	Repeat until CX
	> 0 MOV AX, 5C00H	
	HLT	Halt

7. Sorting of Data in ascending order

Label	Mnemonics	comments
	MOV AX, @data	Initialize DS
	register MOV DS, AX	
	MOV CX,Bytes	Number of bytes used by elements

STEP 1	MOV DX, CX	Copy in DX register
	DEC DX	Number of comparisons
	ADD SI, DX	Point to the last element
	MOV AX, Array [SI]	Move the number to AX register
STEP 2:	CMP Array [SI-2], AX	Compare it with previous number
	JBE STEP 3	If previous number < AX, then go to step 3
	MOV DL, Array {SI-2}	Exchange the elements
	MOV Array [SI], DL	Point to previous element
	DEC SI	
	DEC DX	Decrement counter
	JNZ STEP 2	If DX ≠ 0 then repeat
STEP 3:	MOV Array [SI], AX	Exchange
	INC CX	CX = CX +1
	CMP CX, Bytes	Compare CX with number of bytes
	JBE STEP 1	If CX < number of bytes, goto
	step 1 MOV AX, 5C00H	
	HLT	Halt

8. Finding Factorial of 8-bit dat

Label	**Mnemonics**	**comments**
	MOV AX, @data	Initialize DS register
	MOV DS, AX	
	MOV AL, Num	Number to AL
STEP 1	MOV CL, AL	Copy in DX register

	DEC DX	Keep a copy in CL
	MOV AH, 00	Clear upper 8 bits of accumulator If number = 1 then factorial = 1
	CMP AL, 1	
	JC LOOP 1	
	MOV BL, 2	b = 2
	MOV AL,	a = 1
LOOP 2:		a = a x b
	C BL	b = b +1
	CMP BL, CL	Is BL < = CL
	JNA LOOP2	If yes, then do another pass
LOOP 1:	MOV Fact , AX	Fact =
	AX MOV AX, 5C00H	
	HLT	Halt

Unit –V

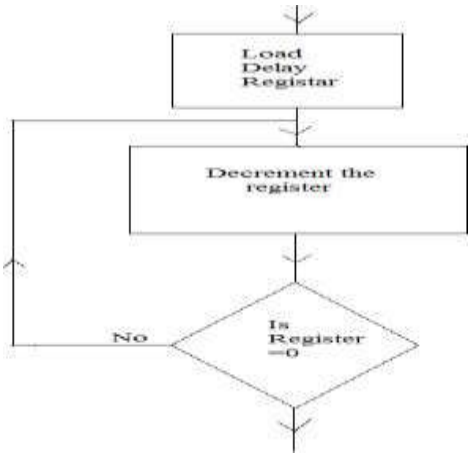
5.1 Delays:

Delay routines are subroutines used for maintaining the timings of various operations in µp. In control applications, certain equipment need to be ON/OFF after a specified time delay. In some applications, a certain operation has to be repeated after a specified time interval. In such cases simple time delay routines can be used to maintain the timings of the operations. A delay routine is generally written as a subroutine (It need not be a subroutine always. It can be even a part of main program). In delay routine a count (number) is loaded in a register of µp. Then it is decremented by one and the zero flags is checked to verify whether the content of register is zero or not. When it is zero the time delay is over and the controls transferred to main program to carry out the desired operation.

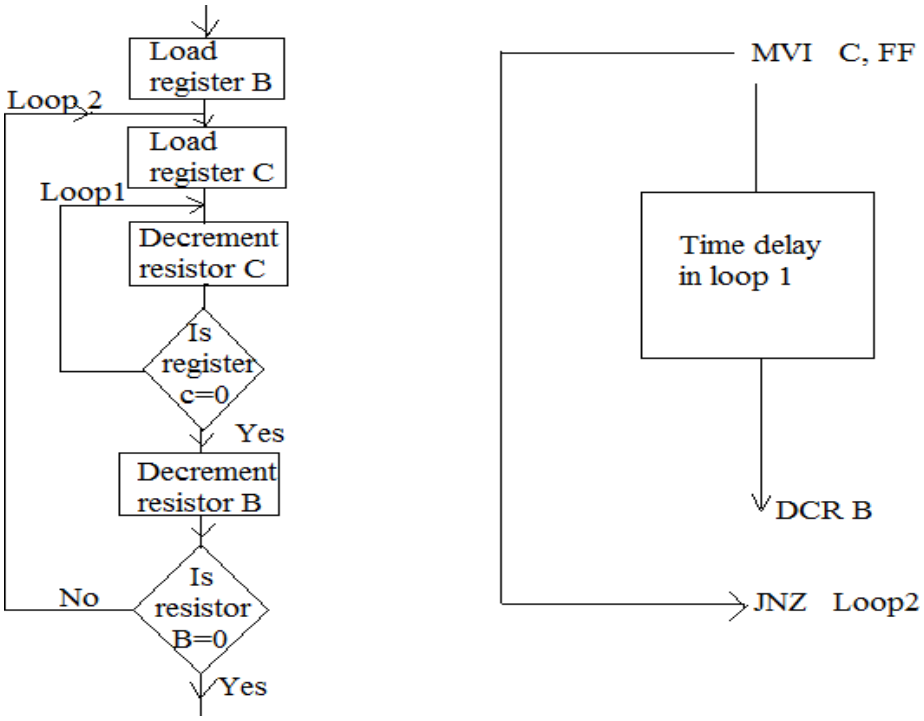
The delay time is given by the total time taken to execute the delay routine. It can computed by multiplying the total number of T required to execute subroutine and the time for T-state of the processor. The total number of time can be computed from the knowledge of T states each instruction. The time for one T state processor is given by the inverse of the internal clock frequency of the processor. For example, of the 8085 µp has 5 MHz quartz crystal then,

5.1.1 Time delay using one register:

The flow chart in Fig. 5.1 shows a time delay loop. A count is loaded in a register, and the loop is executed until the count reaches zero. The set of instructions necessary to set up the loop is also given Fig 5.1



Thus we see that delay obtained is very small. For some purpose this time is sufficient. If more delay is required 3 or more registers can be used.



(a) (b)

Fig.5.2

Similarly , the time delay with in a loop can be increased by using instructions such that will not affect the program except to increase the time delay. For example, the instruction NOP (no operation) can add four T states in the delay loop. The desired time delay can obtained by using any or all available registers.

5.1.2 Delay subroutine using 3 register:

The delay program using 3 registers is given below. To see the indication after certain delay one LED display can be incorporated. The interfacing circuit of a simple LED is shown in Fig.5.3. The LED display is connected to the pin PB_0 of the port B through the buffer. A 560 ohm register is used in series with the LED to limit the circuit drawn by it. A pull up register of

about 1 k is used to boost the output voltage of the buffer. The control wordis 98 H, which makes port B as output port.

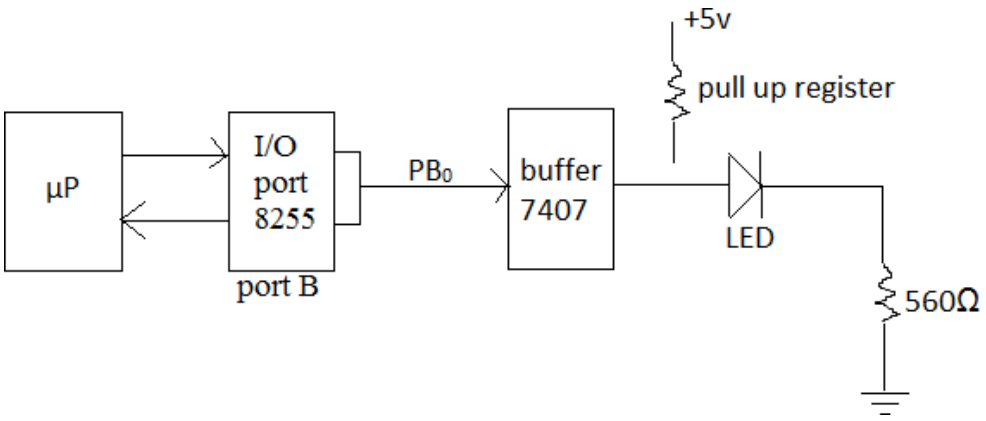


Fig 5.3: Interfacing of LED display

The numbers 50 in register B, FF in C and FF in D are moved to get the desired delay. The delay time with to get the desired delay. The delay time with these numbers is a about 25 sec. This time has been noted by stop watch in the laboratory. The maximum delay with 3 registers can be obtained when all registers are loaded with FF. The maximum delay is about 74 sec.

Example 1:

Write a delay routine to produce a time delay of 0.5 m sec in 8085 µp based system whose clock source is 6 MHz quartz crystal.

Solution:

The delay required in 0.5 m sec, hence an 8- bit register of 8085 can be used to store a count value and then decrement to zero. The delay routine is written as a subroutine as shown below:

Loop	MVI	D,N	Load the count value, N in D register.
	DCR	D	Decrement the count

	JNZ	Loop	If the count is not zero go to the loop
	RET		Return to main program

5.2 Generation of square waves or pulses using µp:

A square wave or pulse can easily be generated by microprocessor. The µp sends high and then low signals to generate square wave or pulse. A pulse or square wave can be generated using I/O port or SOD line or timer/ counter (Intel 8253)

To generate square wave connections are made in Fig. 5.4. The pin PB_o of the port B of 8255-2 is used for taking output. This is connected to a buffer 7407. The final output is taken from the buffer terminal. The control word used in the program is 98H to make port B an output port.

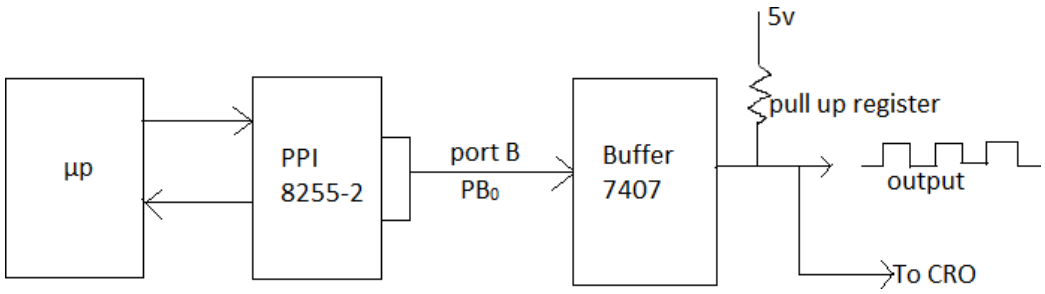


Fig 5.4: To generate square wave using µp

In this program Delay 1 controls the time period for which the square wave remains low, ie, zero. Delay 2 controls the time for which the wave remains High ie, 1. If the time period for Low and High are to be kept equal the counts in register B and register c are made equal. For such a case there is no need of two subroutines only one delay subroutine will be called at two places ie, at 2408 and 240F memory addresses .There will be slight difference in timing of Low and High due to the instruction JMP loop. If accuracy is desired this can be adjusted by suitable adjustment in the cunts of register B and register C. The difference can also be minimized by inserting two NOP instructions in Delay 1 subroutines. The instruction JMP Loop has been used at the end of the program to repeat the whole process to generate equal wave.

5.2.1 To generate square wave or pulse using SOD line:

A square wave can also be generated using SOD line of the µp. zero and one can be outputted at SOD lines using SIM instruction. The execution of SIM instruction loads the content of the 7th bit of the accumulator into SOD latch. While executing SIM instruction the 6th bit of the accumulator is set to 1 to enable SOD lines. To generate square wave the connections are made as shown in Fig 5.5. The SOD terminal is available on a µp kit.

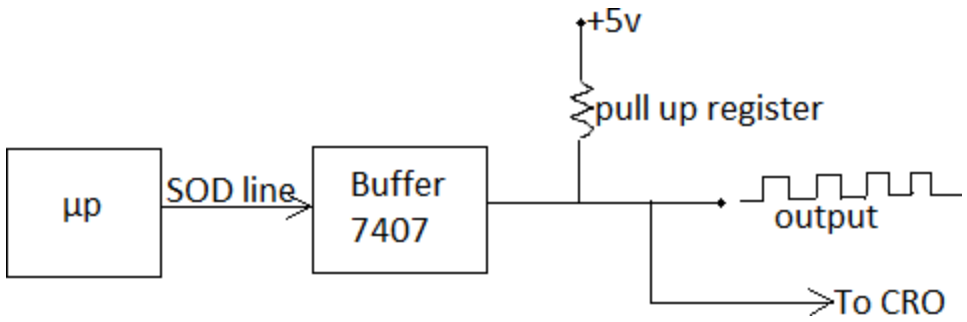


Fig 5.5: To generate square wave using SOD lines

Delay 1 and Delay 2 are subroutines to control time periods for which SOD line remains Low and High respectively. To output Low on the SOD line the 7th bit of the accumulator is set to zero. The 6th bit is set to 1 as it is for enabling the SOD line. Other bits of the accumulator are for setting/resetting of Interrupts. For this case other bits are zero or one, it is immaterial. In the above program they have been set to zero. The first instruction of the program MVI A, 40 indicates that the 7th bit is zero, 6th bit is one and other bits are zero. The SIM instruction outputs the 7th bit of the accumulator on SOD line. CALL Delay 1 is for controlling the time period for which the square wave remain low. The instruction MVI A, CO indicates that the 7th bit of the accumulator is one, 6th bit one and other bits zero. Now the execution of SIM instruction outputs High, ie, 1 on the SOD line. Call Delay 2controls the time period for which the square wave remains High. The JMP loop instruction repeats the whole process to generate a square wave.

5.3 Sensors and Transducers:

5.3.1 Transducers or Transductors:

A general term used for any device receiving input power from one system and supplying output power corresponding to the input certain characteristics(e.g wave from) to another system; which may electrical, mechanical ,or acoustic and thus including transformers, amplifiers, filters, microphones , loud speakers etc.,

Transducers syn.sensors:

Any device that converts a non-electrical parameters ,e.g. sound , pressure, or light, into electrical signals or vice versa. The variations in the electrical signal parameters are functions of the input parameter. Transducers are used in the electro acoustic field. Gramophone pickups, microphones, and loud-speakers are all electro acoustic transducers. The term is also applied to device in which both the input and output electrical signals. Such a device is known as an electric transducers.

For the measurement of physical quantities like temperature, pressure, speed, flow etc., transducers are used to current them to electrical quantities. The electrical output of the transducers is proportional to the input quantity which may be temperature or speed or any other physical quantity.

A transducers used for strain measurement is called a strain Gauge. Strain gauges are used to measure strains and stresses in structures and strain.

Sensors:

A transducers or device whose input is a physical phenomenon and whose output is a quantitative measure of that physical phenomenon.

5.4 Measurement of physical quantities:

Microprocessor – based systems are widely used in industries for the measurement, display and monitoring of physical quantities like temperature, pressure , speed , flow etc., Transducers are used to convert the physical quantities into electrical signal. If the electrical signal is small it is amplified ,using amplifiers. The electrical signal is applied to an A/D converter which is connected to a µc. It more than one physical quantities are to be monitored a multiplexer is included in the interface. A schematic diagram for general interface is shown in Fig.5.6

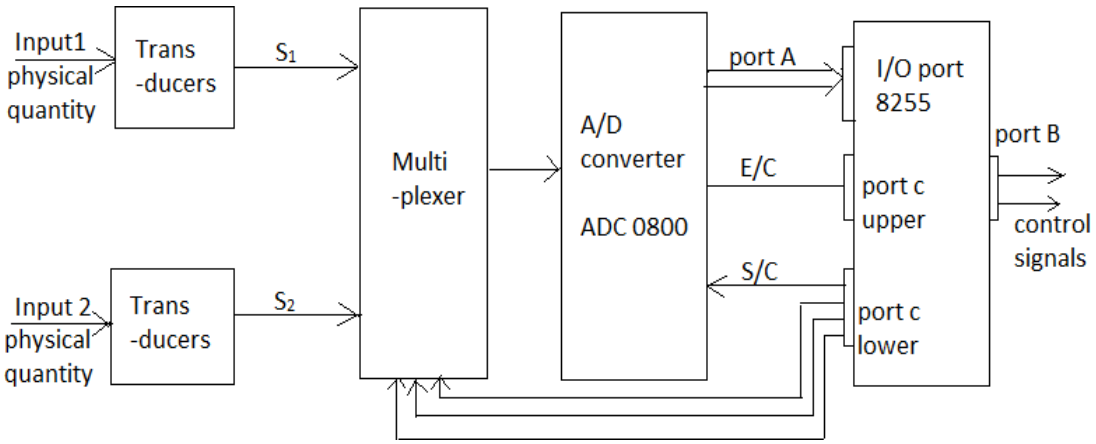


Fig 5.6: Schematic diagram of Interface for physical quantity measurement

5.4.1 Temperature measurement and control:

For the measurements of temperature one of the following devices are used.

(i) Resistance thermometers (-100 to $+300^0$c)

(ii) Thermo couples (-250 to $+200^0$c)

(iii) Thermistor (-100 to $+100^0$c)

(iv) Pyrometers(+100 t0 $+5000^0$c)

(i) Resistance thermometers:

Platinum wires are frequently used in resistance thermometers for industrial applications because of greater resolution, and mechanical and electrical stability as compared to copper or nickel wires.

A change in temperature causes a change in resistance. The resistance thermometer is placed in an arm of a wheatstone bridge to get a voltage proportional to temperature.

(ii) Thermistor:

A thermistor is a semiconductor device fabricated from a sintered mixture of metal alloys having a large negative temperature co efficient. A thermistor is used in a wheatstone bridge to get a voltage proportional to temperature. It can be used in the range of -100^0c to $+100^0$c for greater accuracy as compared to platinum resistance thermometer.

(iii) Thermocouple:

A two-terminal device based on the seeback effect, which is composed of two dissimilar metals that produce a voltage across function that is linearly proportional to the temperature nof the junction.

A thermocouple is the most widely used transducer to measure temperature. Thermocouple materials for the different range of temperatures are as follows:

Material	Temp . range ^{0}c
Iron-constantan	-200 to +1300
Chromel – alnmel	-200 to +1200
Copper – constantan	-200 to +400
Pt – Rh – Platinum	0 to 1500
Tungsten – rhenium	O to 2000

5.4.2 Microprocessor – Based scheme:

Fig 5.7(a) shows a microprocessor-based scheme for temperature measurement and control. The output of a thermocouple proportional to the furnace or oven etc,. is in millivolt. It is to be amplified using multistage amplifier before it is processed by microprocessor. The amplified voltage is applied to an A/D converter. The µp sends a start of conversion signal to the A/D converter through the port of 8255PPI. When A/D converter completes conversion, it sends an end of conversional signal to microprocessor. Having received an end of conversion signal from A/D converter the microprocessor reads the output of the A/D converter which is a

digital quantity proportional to the temperature to be measured. The µp displays the measured temperature. If the temperature of a furnace, oven or water- bath is to be controlled, the µp first measures its temperature and then compares the measured temperature with a reference temperature at which the temperature is to be maintained. If the measured temperature is higher than the reference temperature, µp sends control signal to reduce temperature .If the measured temperature is less than the reference temperature, the µp sends a control signal to increase temperature. The temperature of a furnace or oven can be increased or decreased by increasing or decreasing the final input to the furnace. If heating is done by electric heaters, current in heating element is controlled.

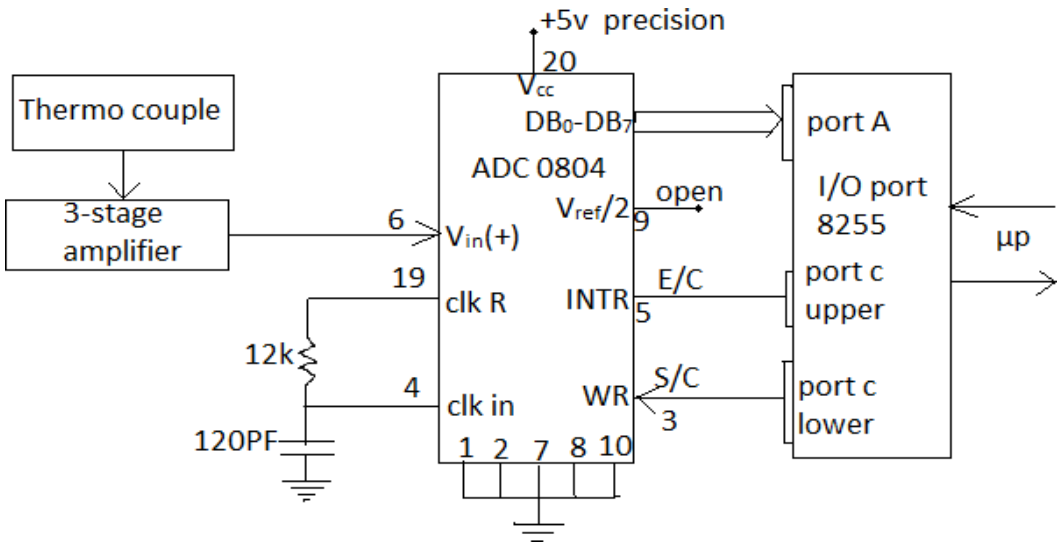


Fig 5.7(a): µp based scheme for temperature measurement

Fig. 5. 7(b) shows an amplifier circuit to amplifier the output of the thermocouple, D.C. level indicator is for initial adjustment.

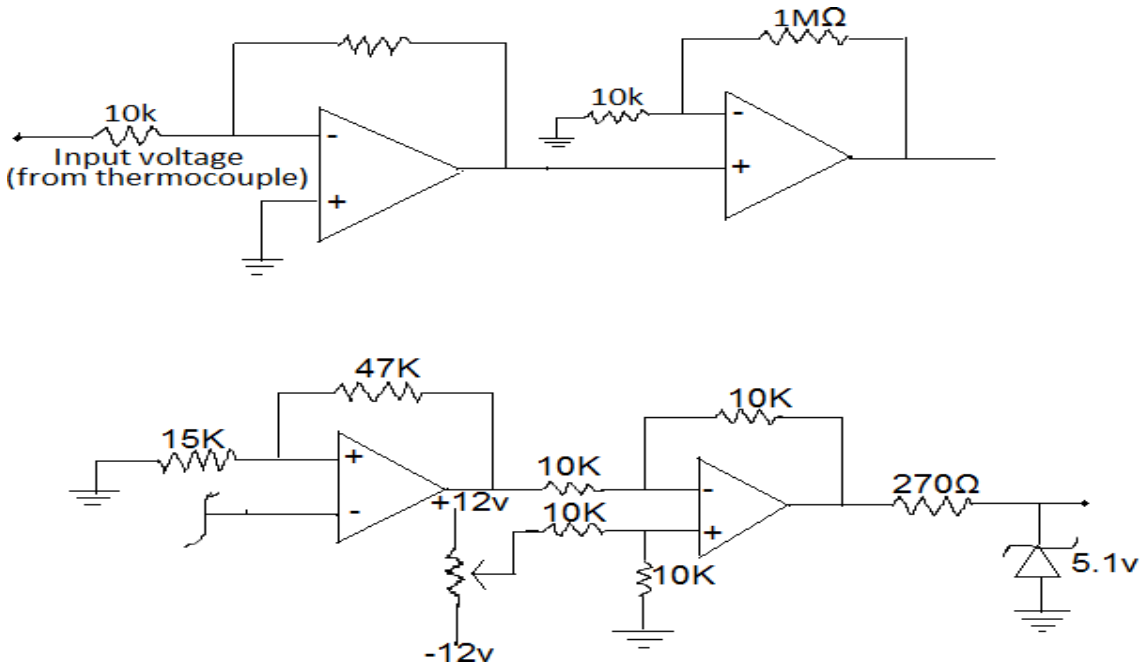


Fig 5.7(b): Three stage Amplifier and D.C. Level Detector

5.4.3 Temperature Monitoring System:

Two transducers have been shown in figure 6. These transducers sense the temperature of two ovens. The temperatures of these two ovens are to be maintained at T_1 and T_2 respectively. The µp sends command to switch on the channel s_1 to get the electrical signal proportional to the temperature of oven no 1. Then it sends start of conversion pulse S/C to the A/D converter. After the conversion is over the A/D converter sends end of conversion signal E/C to the µp. on receiving E/C signal the µp reads the output of the A/D converter. The output of the A/D converter is a digital voltage. This is proportional to t_1, the temperature of the oven no 1. The µp compares t_1 with T_1. If t_1 is less than T_1, the µp issues a control signal to raise the temperature of the oven no 1. If the heating of the oven is electrical and current is controlled by thyristors , the µp will directly control the firing circuit of thyristors to increase current resulting in increase of the temperature. If the heating is done by final, the µp will send a signal to the relay which is controlling the final input. The measurement temperature is also displayed. If $t_1 > T_1$, the µp sends signal to decrease it till it becomes equal to T_1. After this the µp sends commands to switch on the multiplexer channel S_2 to get the electrical voltage proportional to the temperature of over no 2. The µp measures and control its temperature as explained above. After certain internal of time the µp repeats the process of measuring and controlling the temperature of the two ovens. A program flow chart is shown in Fig. 5.8. Similarly any other physical quantity can be measured and monitoring continuously. If a transducers given A.C signal it can be rectified using precision rectifiers.

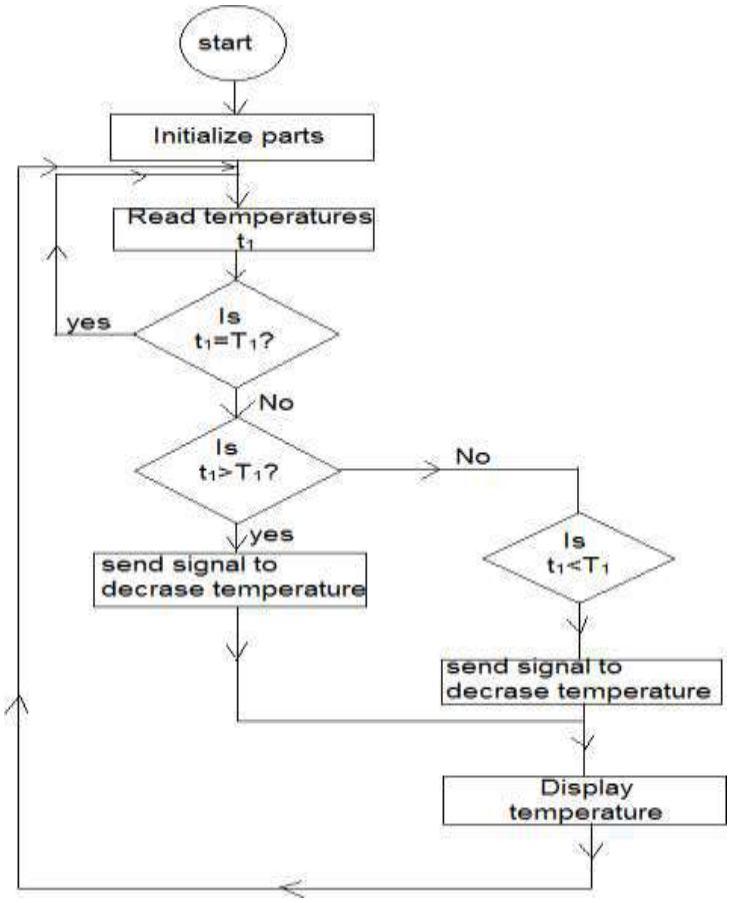


Fig 5.8: Program flow chart for temperature display and monitoring

5.5 Measurements of Electrical quantities:

(i) Frequency measurements:

To measure the frequency of a signal, the time period for half cycle is measured which is inversely proportional to the frequency. A sinusoidal signal is converted to square wave using a voltage comparator LM 311 or operational amplifier LM 747 or LM 324 as shown in Fig.5.9.

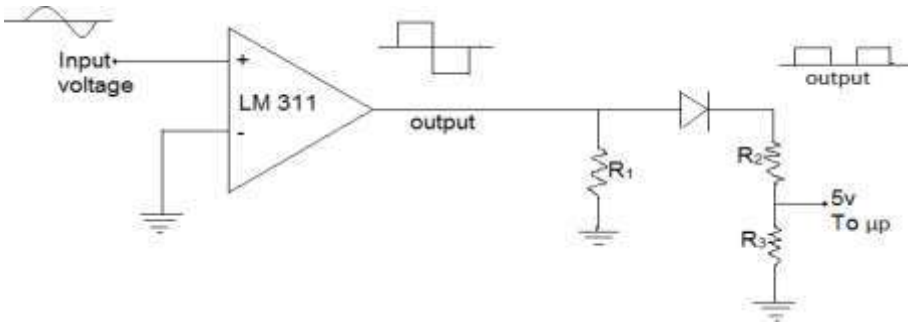


Fig 5.9: Sine to square wave generator

A diode is used to rectify the output signal. A potential divides is used to reduce the magnitude to 5 volts.

A program has been developed to sense the zero instant of the rectified square wave. The µp measures the magnitude of the square wave at two consecutive points as shown in fig 10. The two magnitudes are compared and decision is taken on the basis of carry and zero states flags, whether the point is at zero point.

Various points have been shown in Fig 5.10. Very nearly to P_3 at its left side the magnitude of the square wave is zero, and at P_4 , 5v, at logic „1". The µp subtracts the 1st value from the 2nd, so the result is non- zero and there is no carry. This is the basis for the selection of zero instant point. Suppose the µp takes reading at p_1 and p_2 where both magnitudes are zero. Difference of the two is zero. So this is not the zero instant of the wave. At points p_5 and p_6 the difference of the two values is zero. So it is also not a zero instant point. At p_7 and p_8, the difference is non-zero but that is carry. So it is the end point of the half- square wave.

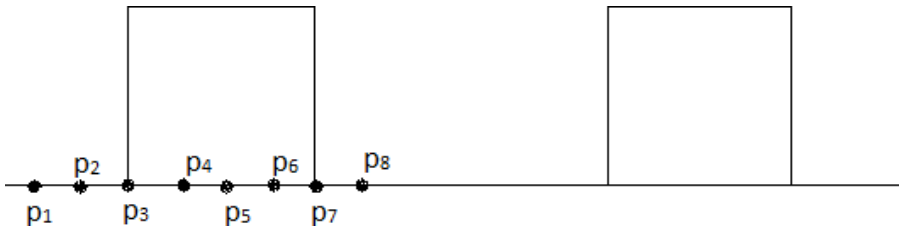


Fig 5.10: Rectified square wave

As soon as the zero instant point is detected the µp initiates a register pair to count the number how many times the loop is executed. The µp reads the magnitude of the square wave again and moves in the loop. It crosses the loop when the magnitude of the square wave becomes zero. Thus the time for half cycle is measured. The count can be compared with the stored numbers in the look-up table and the frequency can be displayed. The count is inversely proportional to the frequency of the input signal can be used for further processing and control as desired. An interfacing circuitary is shown in Fig 5.11. The program flow chart is shown in Fig 5.12. The port is input control word is 98H.

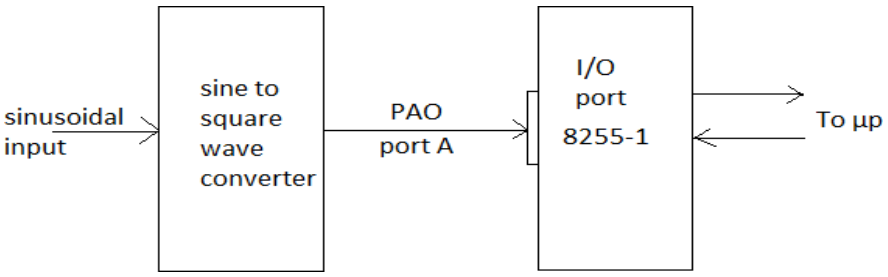
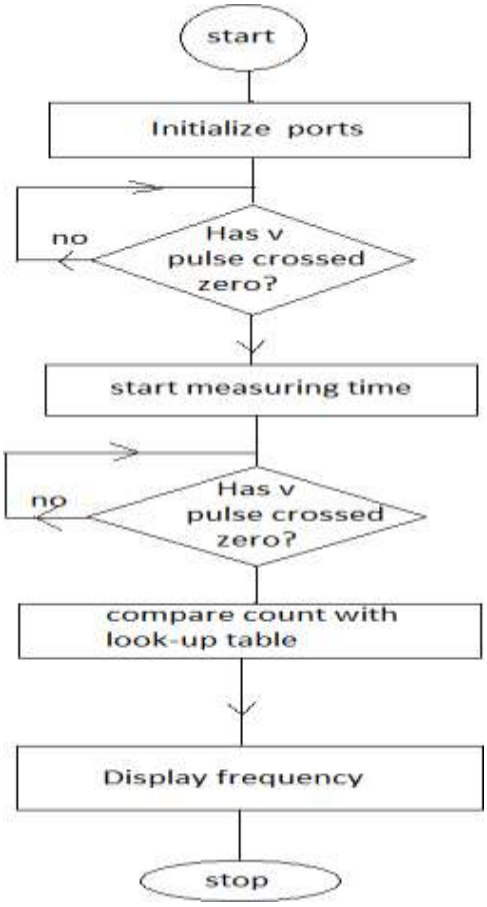


Fig 5.11: Interface for frequency measurement

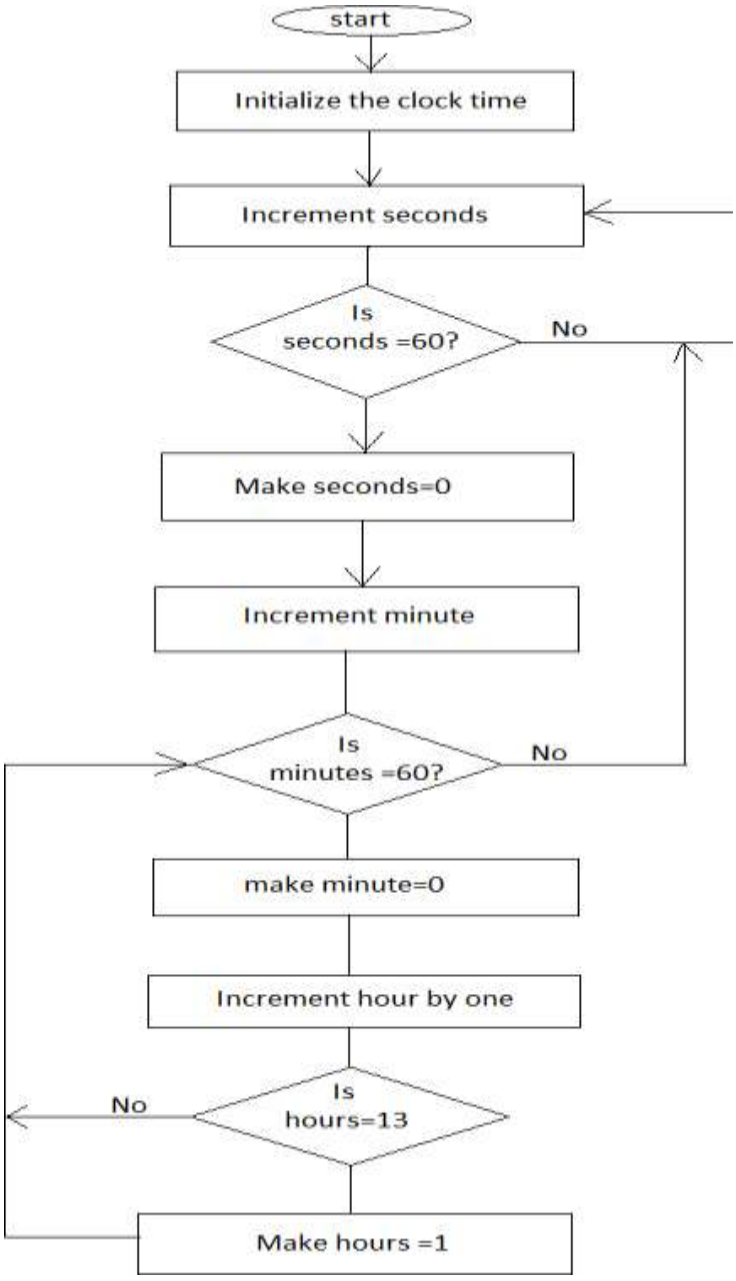
Flow chart for frequency measurements:



5.6 Digital clock:

Many µp applications would require doing certain tasks at specific time of the day or involve the time of day in some other form. For example, switching ON and OFF street lights at specific time in the evening and following morning or punching entry time of every worker for a shift, at a factory gate etc,. A basic requirement of this type of application is a real time digital clock with a display of current time. The flow chart and the corresponding program are shown in Fig.5.15.

Flow Chart for digital clock program:


start
Initialize the clock time
Increment seconds
Is
seconds =60?
No
Make seconds=0
Increment minute
Is
minutes =60?
No
make minute=0
Increment hour by one
Is
hours=13
No
Make hours =1

(i) Display and keyboard:

i. For displaying the speed of the motor the data fields display of the drainer kit is used.
ii. Keyboard of the trainer kit is used for entiring the required speed.

(ii) Software design:

Software has to be designed to implement the following steps:

i. Read the desired speed from the keyboard.
ii. Choose an initial data to be applied to the DAC. So that the motor starts running.
iii. Apply data to the DAC through port A
iv. Generate delay for the motor to attain stable speed.
v. Measure the speed of the motor.
vi. Compare the speed with the required speed. If the speed matches then go to step 7. If the required speed is low and then increment data by 1, otherwise decrement data by 1 and go to step 7.
vii. Display the speed on the display

viii. Go to step 3.

Flow chart:

A flow chart of these operation in the proper sequence is shown in Fig.5.19.

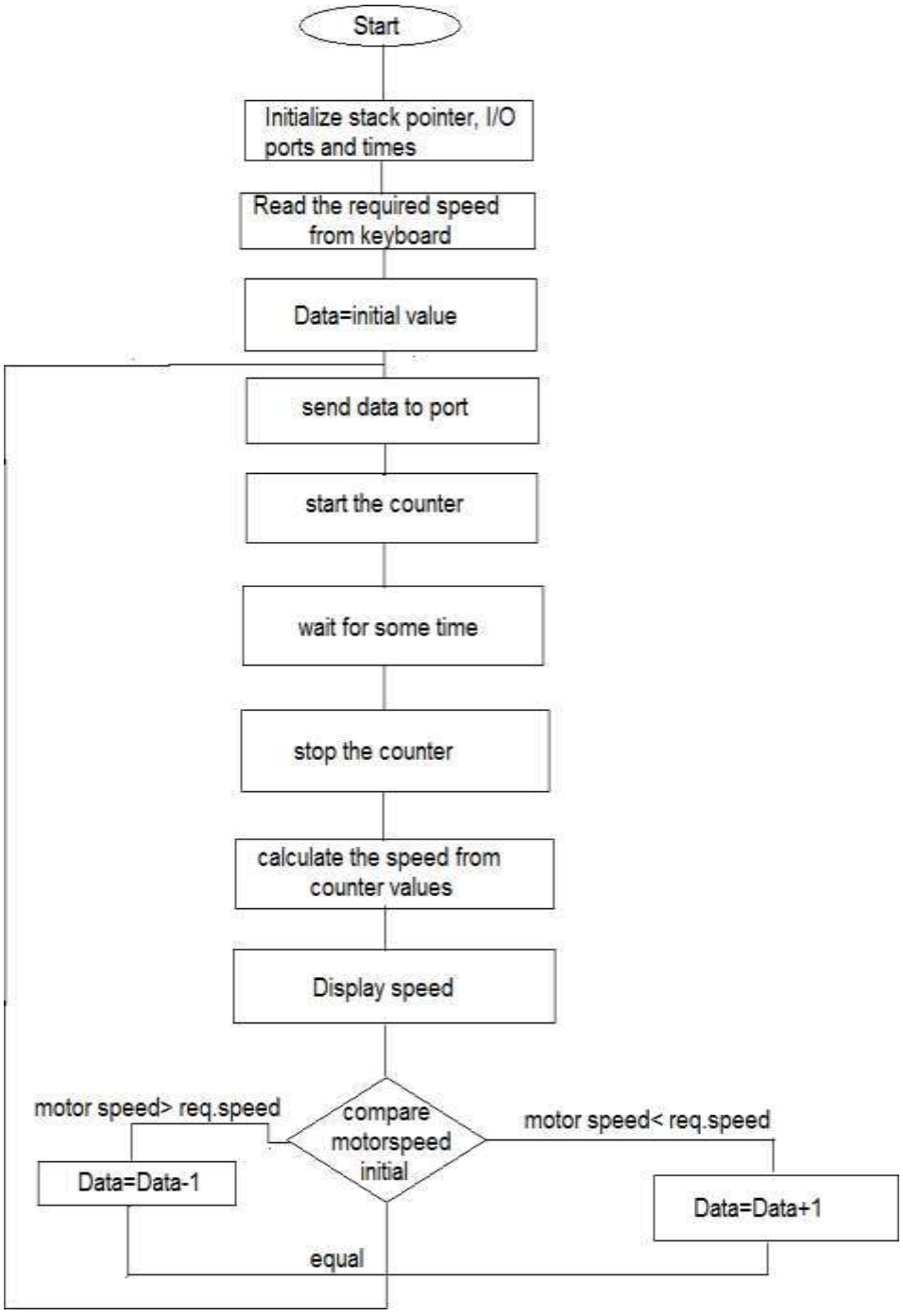


Fig.5.19

(i) Set up registers for display:

- MVI B, 08H : load count
- MVI C, 7FH : load select pattern
- LXI H, 6000B : starting address of message

Display message:

- DISP 1: MOV A, C : select digit
- OUT PB
- MOV A, M : get data
- OUT PA : display data
- CALL DELAY : wait for some time
- DISP 1: MOV A, C
- RRC
- MOV C, A : adjust selection pattern
- INX H
- DCR B : Decrement count
- JNZ DISP 1 : repeat 8 times
- RET

Note: This "display message subroutine" must be called continuously to display the 7-segment coded message stored in the memory from address 6000H.

Delay Subroutine:

- Delay: LXI D, Count
- Back: DCX D
- MOV A, D
- ORA E
- JNZ Back
- RET