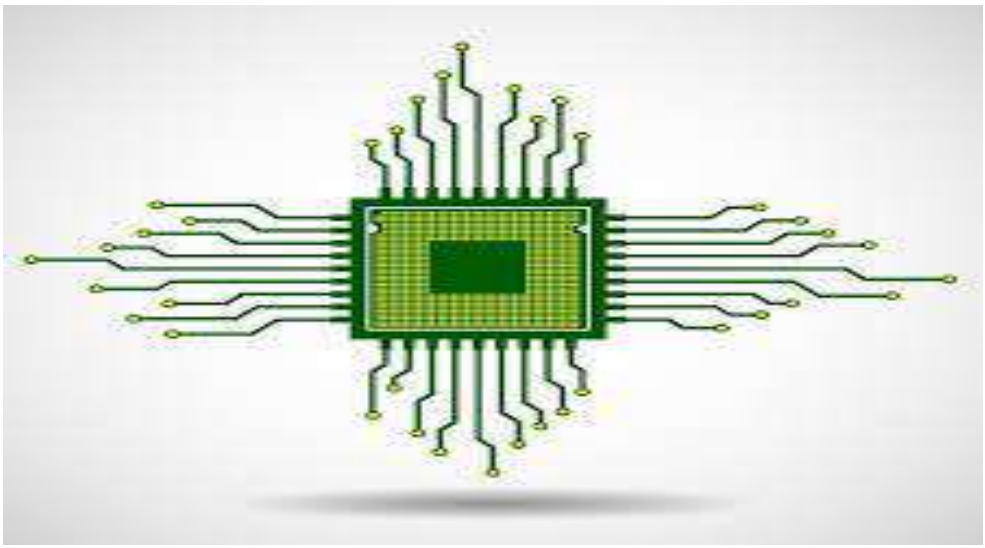
BHADRAK ENGINEERING SCHOOL & TECHNOLOGY (BEST),

ASURALI, BHADRAK

MICROPROCESSOR & MICRO-CONTROLLER (Th- 03)

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



Fourth Semester

E&TC & CSE Engg.

Prepared By: *Er B.R.Nayak*

MICROPROCESSOR & MICRO-CONTOLLER

CHAPTER-WISE DISTRIBUTION OF PERIODS & MARKS

SL. NO.	Chapter/ Unit No.	Topics	Periods as per Syllabus	MARKS
1	**01**	Microprocessor (Architecture and Programming-8 Bit-8085)	15	20
2	**02**	Instruction Set and Assembly Language Programming (8 Bit)	15	15
3	**03**	TIMING DIAGRAMS	07	20
4	**04**	Microprocessor Based System Development Aids	11	15
5	**05**	Microprocessor (Architecture and Programming- 16 Bit-8086)	12	15
6	**06**	Microcontroller (Architecture and Programming-8bit)	15	15
	TOTAL		**75**	**100**

CHAPTER-01

Microprocessor (Architecture and programming-8 bit-8085)

Learning Objectives:

1.1- Introduction to Microprocessor and Microcomputer & distinguish between them.

1.2- Concept of Address bus, data bus, control bus & System Bus

1.3-General Bus structure Block diagram.

1.4- Basic Architecture of 8085 (8 bit) Microprocessor

1.5- Signal Description (Pin diagram) of 8085 Microprocessor

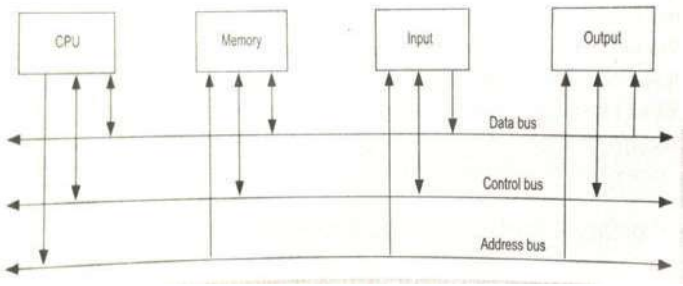
1.6- Register Organizations, Distinguish between SPR & GPR, Timing & Control Module,

1.7- Stack, Stack pointer & Stack top.

1.8- Interrupts: -8085 Interrupts, Masking of Interrupt (SIM, RIM)

1.1- Introduction to Microprocessor and Microcomputer & Distinguish Between Them

- A *microprocessor* is a programmable electronics chip that has computing and decision-making capabilities similar to central processing unit of a computer.
- Any microprocessor- based systems having limited number of resources are called microcomputers. Nowadays, microprocessor can be seen in almost all types of electronics devices like mobile phones, printers, washing machines etc.
- Microprocessors are also used in advanced applications like radars, satellites and flights. Due to the rapid advancements in electronic industry and large-scale integration of devices results in a significant cost reduction and increase application of microprocessors and their derivatives.
- **Bit**: A bit is a single binary digit.
- **Word**: A word refers to the basic data size or bit size that can be processed by the arithmetic and logic unit of the processor. A 16-bit binary number is called a word ina 16-bit processor.
- **Bus**: A bus is a group of wires/lines that carry similar information.



- **Memory Word**: The number of bits that can be stored in a register or memory element is called a memory word.

Difference between microprocessor and microcomputer.

1- Microprocessor is one component of amicrocomputer. **2-** Microprocessor is a programmable	**1-**A digital computer in which one microprocessor is used as a CPU known as microcomputer.

integrated circuit which has its own decision-making capability. Example-8085, INTEL8086,8088,8008,8080.etc.	**2-**Microcomputer uses a microprocessorfor its processing operation. Example-Desktop, Laptop, Note Book etc.

1.2- Concept Of Address Bus, Data Bus, Control Bus & System Bus

Address Bus: It carries the address, which is a unique binary pattern used to identifya memory location or an I/O port. For example, an eight-bit address bus has eight lines and thus it can address 2^8 = 256 different locations. The locations in hexadecimal format can be written as 00H – FFH.

Data Bus: The data bus is used to transfer data between memory and processor or between I/O device and processor. For example, an 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have 16-bit data bus.

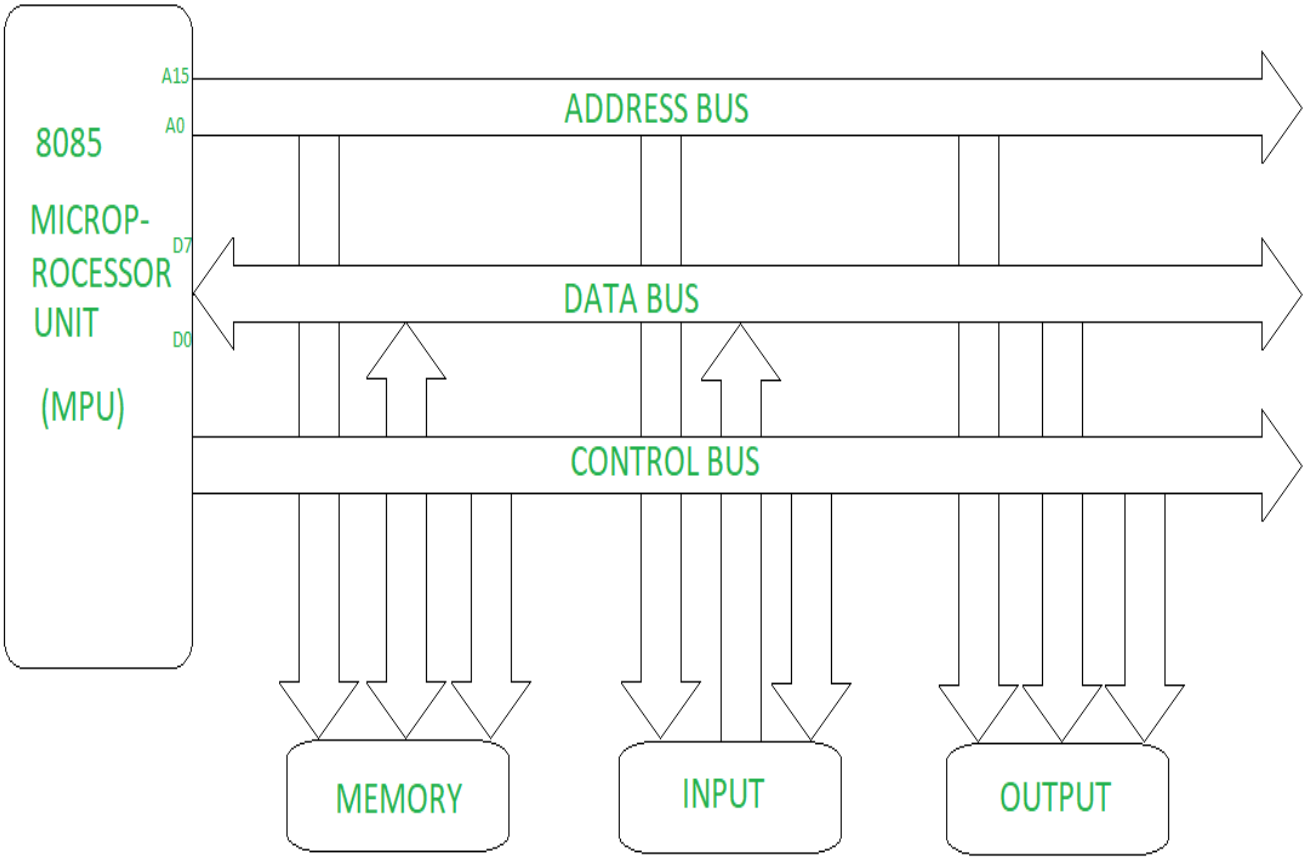
Control Bus: The control bus carry control signals, which consists of signals for selection of memory or I/O device from the given address, direction of data transferand synchronization of data transfer in case of slow devices.

System Bus: The system bus is a group of wires/lines used for communication between the microprocessor and peripherals.

1.3-General Bus Structure Block Diagram.

Bus is a group of conducting wires which carries information, all the peripherals are connected to microprocessor through Bus.

Diagram to represent bus organization system of 8085 Microprocessor.



Bus organization system of 8085 Microprocessor

There are three types of buses.

Address bus –

It is a group of conducting wires which carries address only. Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices (That is, Out of Microprocessor).

Length of Address Bus of 8085 microprocessor is 16 Bit (That is, Four Hexadecimal Digits), ranging from 0000 H to FFFF H, (H denotes Hexadecimal). The microprocessor 8085 can transfer maximum 16-bit address which means it can address 65, 536 different memory location.

The Length of the address bus determines the amount of memory a system can address. Such as a system with a 32-bit address bus can address 2^32 memory locations. If each memory location holds one byte, the addressable memory space is 4 Bowater, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

Data bus –

It is a group of conducting wires which carries Data only. Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to microprocessor.

Length of Data Bus of 8085 microprocessor is 8 Bit (That is, two Hexadecimal Digits), ranging from 00 H to FF H. (H denotes Hexadecimal).

When it is written operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.

The width of the data bus is directly related to the largest number that the bus can carry, such as an 8-bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255.A 16-bit bus can carry 0 to 65535.

Control bus –

It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data, that is what to do with selected memory location. Some control signals are:

- Memory read
- Memory writes
- I/O read
- I/O Write
- Opcode fetch

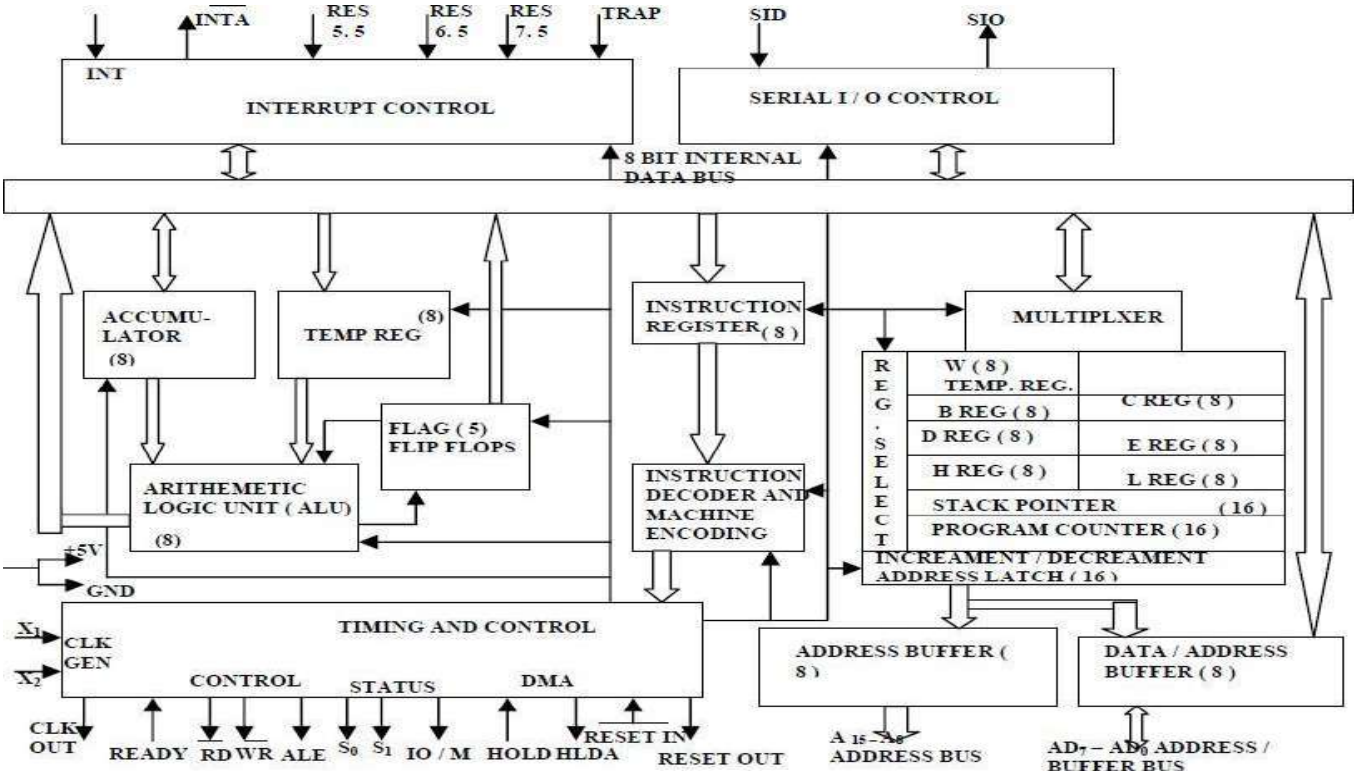
If one line of control bus may be the read/write line. If the wire is low (no electricity flowing) then the memory is read, if the wire is high (electricity is flowing) then the memory is written.

1.4- Basic Architecture Of 8085 (8 Bit) Microprocessor

Architecture of 8 Bit 8085 Microprocessor:

Description of Each Block:

- In 1975 INTEL corporation developed a more power full b bit MP by using NMOS Technology known as INTEL 8085 Microprocessor.
- It is a 40-pin dual package IC fabricated on a single LSI chip.
- The INTEL 8085 uses a single +5-volt supply for its operation and its clock speed is 3 MHZ and clock cycle is 320 nano sec.



Physical components of 8085 Microprocessor:

- ✓ Register set.
- ✓ Bus interface unit (BIU).
- ✓ Arithmetic & logic unite (ALU).
- ✓ Instruction decoder & Machine cycle encoder.
- ✓ Timing and control unit.
- ✓ Interrupt and serial communication.

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.

Increment and Decrement register

- ✓ When the instruction is fetched the value of the program counter is increment and decrement by the increment and decrement Register.

Bus Interface unit (BIU)

- ✓ BIU consist of Address buffer or Address bus , Address and data buffer or Address and data bus.
- ✓ Address buffer or Address bus are A8-A15 and Address and Data bus are AD0-AD7.
- ✓ A8-A15 Address bus are used for MSB bits of memory Address.
- ✓ AD0-AD7, these lines are time multiplexed with address and date. They are used for LSB of memory address.

Instruction Register (IR)

- ✓ The data fetch from memory is stored by IR register.
- ✓ IR only holds that type of data which is fetch from memory.

Instruction Decoder:

- ✓ Instruction Decoder Decode the instruction by 0 or 1

Timing and control unit:

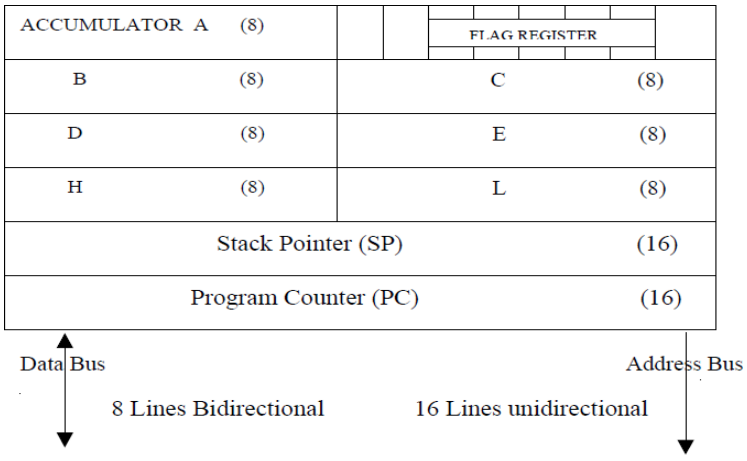
- ✓ Timing and control unit provide the information through control signal to all the register present in microprocessor.
- ✓ It controls the entire operation of the MP. So it is known as brain of the computer.

Arithmetic Logic Unit (ALU)

- ✓ 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.

Register Set:

- ✓ The 8085 includes six registers, one accumulator and one flag register.
- ✓ It has two 16-bit registers: stack pointer and program counter.
- ✓ The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L.
- ✓ They can be combined as register pairs - BC, DE and HL to perform some bit operations.
- ✓ The programmer can use these registers to store or copy data into the register by using data copy instructions.



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.
- ✓ The five-status flag of INTEL 8085 MP are—
- ✓ Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags.
- ✓ Their bit positions in the flag register are shown in Fig.
- ✓ The microprocessor uses these flags to test data conditions.

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

Stack Pointer:

- ✓ 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.
- ✓ **Stack**-The Stack is a sequence of memory location set by a programmer to store different element. So, it is known as Storage device.
- ✓ Stack works by LIFO (Last in First out) Operation.

Interrupt Control:

- ✓ INTA-Interrupt Acknowledgement.
- ✓ INTR-Interrupt Request
- ✓ If there is any Interrupt generated inside MP then the Interrupt signal send the request to the Microprocessor.
- ✓ There are 5 Interrupt signal i.e TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

Serial input and output control:

- ✓ It has two input/output pin that is SID & SOD
- ✓ SID stands for serial input data and SOD for serial output data.

1.5- Signal Description (Pin Diagram) Of 8085 Microprocessor

Properties:

- ✓ It is a 8-bit microprocessor
- ✓ Manufactured with N-MOS technology
- ✓ 40 pin IC package
- ✓ It has 16-bit address bus and thus has 2^{16} = 64 KB addressing capability.
- ✓ Operate with 3 MHz single-phase clock.
- ✓ All the signals are classified into six Group that is

1-address bus

2-Data bus

3-Control & status signals

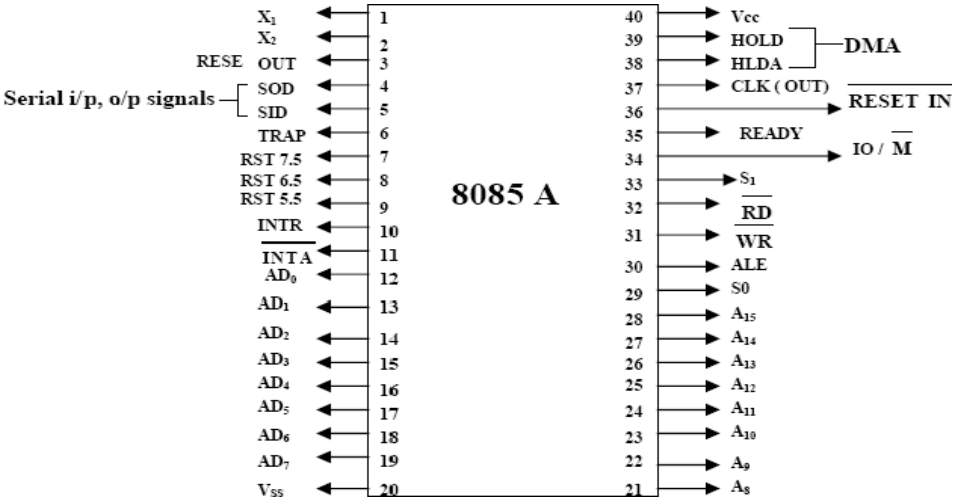
4-Power supply and frequency signals

5-Externally initiated signals

7-Serial I/O signals

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.



Control & Status Signals:

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

- ✓ 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 ofclock cycles until it goes high.
- ✓ HOLD: This signal indicates that a peripheral like DMA (direct memory access)controller is requesting the use of address and data bus.
- ✓ HLDA: This signal acknowledges the HOLD request.
- ✓ INTR: Interrupt request is a general-purpose interrupt.
- ✓ INTA: This is used to acknowledge an interrupt.
- ✓ RST 7.5, RST 6.5, RST 5,5 – restart interrupt: These are vectored interrupts andhave 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 RIMinstruction.
- ✓ SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

1.6- Register Organizations, Distinguish Between SPR & GPR, Timing & Control Module

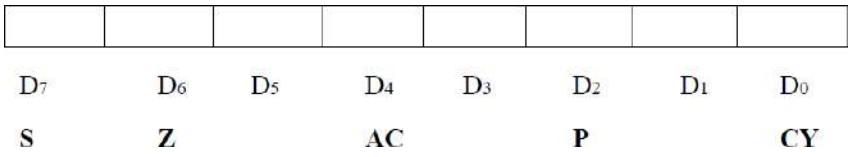
- ✓ The 8085 includes six registers, one accumulator and one flag register,In addition, it has two 16-bit registers: stack pointer and program counter.
- ✓ The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some bit operations. The programmer can use these registers to store or copy data into theregister by using data copy instructions.

Accumulator

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

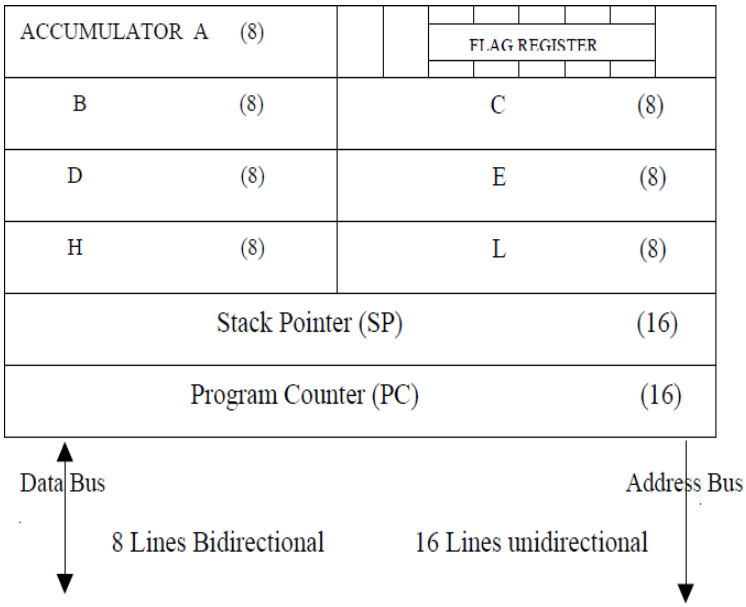
Flag register

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



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

A negative number has a 1 in bit D7 and a positive number has a 0 in 2's



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

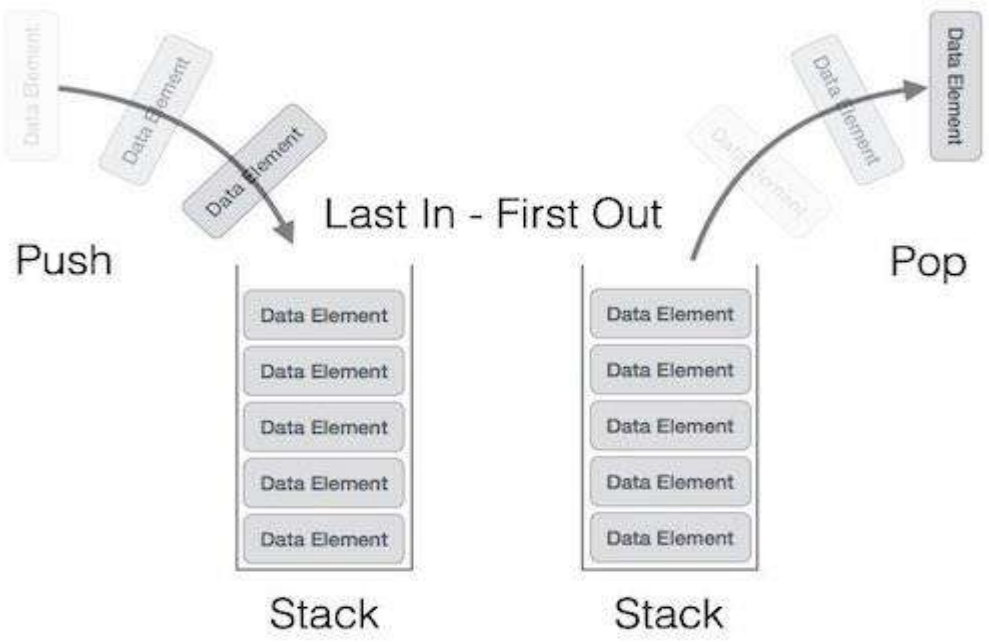
Difference between SPR and GPR

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

1.7- Stack, Stack Pointer & Stack Top

Stack

- ✓ Stack is used to store and retrieve return addresses during function calls.
- ✓ It is also used to transfer arguments to a function. On a microprocessor it is also used to store the status register contents before a context switch. The stack is a temporary store for data.
- ✓ There are two types of stacks they are register stack and the memory stack.



Stack Pointer:

- ✓ 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.
- ✓ A stack (also called a pushdown stack) operates in a last-in/first-out sense.
- ✓ When a new data item is entered or "pushed" onto the top of a stack, the stack pointer increments to the next physical memory address, and the new item is copied to that address.
- ✓ When a data item is "pulled" or "popped" from the top of a stack, the item is copied from the address of the stack pointer, and the stack pointer decrements to the next available item at the top of the stack.

Stack top:

- ✓ The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data.
- ✓ when the microprocessor branches to a subroutine. ... The Stack Pointer register will hold the address of the top location of the stack is known as Stack top.

1.8- Interrupts: -8085 Interrupts, Masking Of Interrupt (SIM, RIM)

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

Types of Interrupts:

- ✓ Interrupts are classified based on their mask ability, IVA and source. They are classified as: Vectored and Non-Vectored Interrupts
- ✓ Vectored interrupts require the IVA to be supplied by the external device thatgives 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.

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.

Software and Hardware Interrupts

- ✓ Software interrupts are special instructions, after execution transfer the control to predefined ISR.
- ✓ Hardware Interrupts and Priorities:
- ✓ 8085 have five hardware interrupts – INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP.

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.
- ✓ 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 instructionis 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. 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.

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.

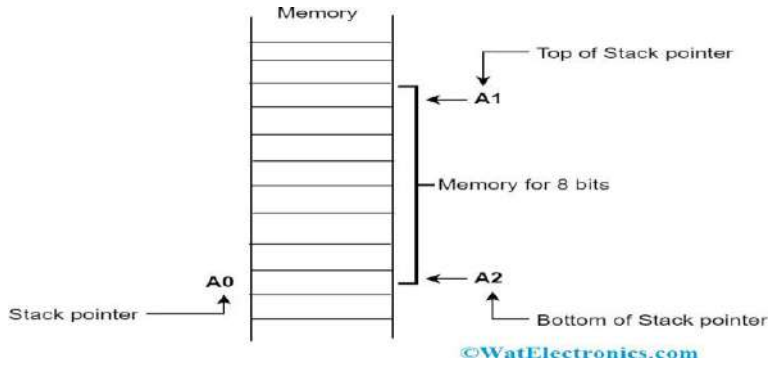
POSSIBLE SHORT TYPE QUESTIONS WITH ANSWERS

1-Define programme counter. (19-S)

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

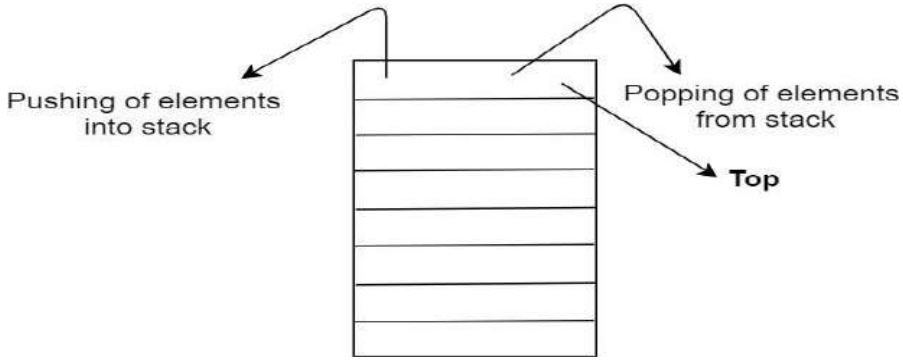
2-What is the function of Stack pointer?

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



3-Define Stack.

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



4-Write the function of ALU?

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

5-Write the different register name of 8085 MP?

In 8085 two types of register are used.

1-General purpose register

2-Special purpose register

General purpose are B,C,D,E,H and L , Accumalotor,Instruction register ,Temporary Register are in 8 bit. Special purpose register are Stack pointer and programme counter are in 16 bit.

6-Show the bit position of Flag register. (19-W)

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

Carry flag (Cy),

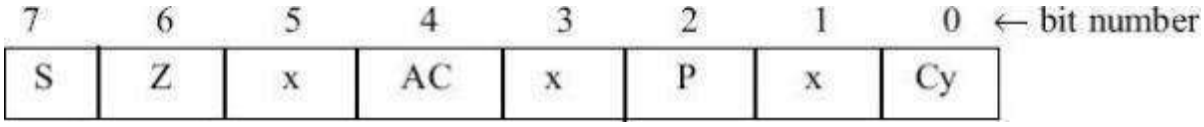
Auxiliary carry flag (AC),

Sign flag (S),

Parity flag (P), and

Zero flag (Z).

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



7-What is carry flag and auxiliary carry flag? (19-W)

Auxiliary carry flag (Ac): Now let us consider the addition of any two 8-bit (2-hex digit) numbers, a carry may be generatedwhen we add the LS hex digits of the two numbers. Such a carry is called intermediate carry also known as half carry,or auxiliary carry.

Carry Flag (CY) – Carry is generated when performing n bit operations and the result is more than n bits, then this flag becomes set i.e. 1, otherwise it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset and if (A<B) itbecomes set. Carry flag is also called borrow flag.

8-What is the function of ALE.

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

9-Write different interrupt are present according to the priority order. (S-23)

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

10-What is the function of SO and S1.

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

11-Define address Bus.

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

12-Define data Bus.

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

13-Write a program to calculate time delay using one register for 8085. (S-24)

Ans: - By using single 8-bit register.

```
        MVI B, 0AH
REPEAT: DCR B
        JNZ: REPEAT
```

14-What do you mean by DMA techniques? Which pin of 8085 belongs to this group? (S-24)

Ans: - DMA (Direct Memory Access) techniques refer to a method where a peripheral device can directly access system memory without needing the CPU for each data transfer.
Allowing faster data movement between devices and memory.
DMA are "HOLD" and "HLDA" pins are the DMA signal for 8085 microprocessors.

15-List the 16 bits register of 8085 microprocessors. (S-24)

Ans: - The 8085 microprocessor has three 16-bit registers:

- Program Counter (PC): Stores the memory address of the next instruction to be executed.
- Stack Pointer (SP): Points to the current position in the hardware stack.
- Incremented/Decremented address latch register: A 16-bit register.

16-Calculate the memory capacity of a microprocessor of 14-bit address line. (S-24)

Ans: - A byte is 8 bits, so if a memory has a 14-bit address, it means it can store 2^14 = 16384 bytes of data Or 16 KB memory capacity.

LONG TYPE QUESTIONS

1-Explain the Architecture of 8085 MP with neat block diagram?

2-Explain the pin configuration of 8085 MP? (19-W)

3-Write soft notes on

Stack, Flag register,8085 Busses

4-Draw the different bits of the flag register of 8085 microprocessor explain the function of each flag. (S-24)

5-State and Explain stack, stack top and stack pointer. (S-24)

6-Draw the pin diagram of 8085 microprocessor and explain the function of each pin. (S-24)

CHAPTER No.-02

Instruction set and Assembly Language Programming(8-Bit)

Learning Objectives:

2.1-Addressing data & differentiate between one-byte, two-byte & three-byte instructions with examples.

2.2- Addressing modes in instructions with suitable examples.

2.3- Instruction Set of 8085(Data Transfer, Arithmetic, Logical, Branching, Stack& I/O , Machine Control

2.4-Simple Assembly Language Programming of 8085- Simple Addition & Subtraction. Logic Operations (AND, OR, Complement 1's & 2's) & Masking of bits. Counters & Time delay (Single Register, Register Pair, more than Two Register). Looping, Counting & Indexing (Call/JMP etc.). Stack & Subroutines programs. Code conversion, BCD Arithmetic & 16 Bit data Operation, Block Transfer. Compare between two numbers. Array Handling (Largest number & smallest number in the array).

2.5-Memory & I/O Addressing

2.1-Addressing Data & Differentiate Between One-Byte, Two-Byte & Three-Byte Instructions with Examples.

One byte, two byte & three-byte instruction with example.

- ✓ According to the length, the instruction of 8085 microprocessor is classified into 3 types.
- ✓ 1-Single byte instruction
- ✓ 2-Two-byte instruction
- ✓ 3-Three-byte instruction

Single byte instruction

- ✓ **In** single byte instruction only opcode is present, there is no operand.
- ✓ Examples-MOV A, B, ADD B, CMAIn these instructions only opcode is present so these are the single byte instruction.
- ✓ The length of this instruction is 8 bits. Each instruction requires one memory location.

Two-byte instruction

- ✓ In two-byte instruction the first 8 bit indicates the opcode and next 8 bit indicates the operand
- ✓ .Example MVI A,32H, ADI A,08H
- ✓ The length of this instruction is 16 bit that is the opcode is 8 bit and operand is 8 bits.Each instruction requires two memory location.

Three-byte instruction

- ✓ Three instruction is the type of instruction in which the first 8 bit indicates the opcode and next 2 bytes specified the operand which is 16 b it address.
- ✓ The low order address is represented in 2nd byte and the higherorders address represented in the 3rd byte.
- ✓ Example-LBA 2050H, JMP 2085H

- ✓ This instruction would require 3 memory location to store the binary code.

2.2- Addressing Modes in Instructions with Suitable Examples.

- ✓ 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 thefollowing five types of addressing modes.
  - o Immediate addressing
  - o Memory direct addressing
  - o Register direct addressing
  - o Indirect addressing
  - o 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 todestination 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 ismoved 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 inthe instruction.

2.3- Instruction Set Of 8085(Data Transfer, Arithmetic, Logical, Branching, Stack& I/O, Machine Control

Instruction set of 8085:

Instruction sets are instruction codes to perform some tasks. It is classified into five categories.

1-Control Instruction

2-Logical Instruction

3-Branching Instruction

4-Arithmatic Instruction

5- Data transfer Instruction

Control Instruction:

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

Logical Instruction:

Opcode	Operand	Meaning	Explanation
CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	8-bit data	Compare immediate with the accumulator	The second byte data is compared with the contents of the accumulator.

ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
XRA	R M	Exclusive OR register or memory with the accumulator	The contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
XRI	8-bit data	Exclusive OR immediate with the accumulator	The contents of the accumulator are Exclusive OR with the 8-bit data and the result is placed in the accumulator.
ORA	R M	Logical OR register or memory with the accumulator	The contents of the accumulator are logically OR with M the contents of the register or memory, and result is placed in the accumulator.
ORI	8-bit data	Logical OR immediate with the accumulator	The contents of the accumulator are logically OR with the 8-bit data and the result is placed in the accumulator.
RLC	None	Rotate the accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
RRC	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
RAL	None	Rotate the accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.

RAR	None	Rotate the accumulator right through carry	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
CMA	None	Complement accumulator	The contents of the accumulator are complemented. No flags are affected.
CMC	None	Complement carry	The Carry flag is complemented. No other flags are affected.
STC	None	Set Carry	Set Carry

Branching Instruction:

<table>
<tr><th colspan="3">Opcode</th><th>Operand</th><th>Meaning</th><th>Explanation</th></tr>
<tr><td colspan="3">JMP</td><td>16-bit address</td><td>Jump unconditionally</td><td>The program sequence is transferred to the memory address given in the operand.</td></tr>
<tr><td colspan="3"><table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>JC</td><td>Jump on Carry</td><td>CY=1</td></tr>
<tr><td>JNC</td><td>Jump on no Carry</td><td>CY=0</td></tr>
<tr><td>JP</td><td>Jump on positive</td><td>S=0</td></tr>
<tr><td>JM</td><td>Jump on minus</td><td>S=1</td></tr>
<tr><td>JZ</td><td>Jump on zero</td><td>Z=1</td></tr>
<tr><td>JNZ</td><td>Jump on no zero</td><td>Z=0</td></tr>
</table></td><td>16-bit address</td><td>Jump conditionally</td><td>The program sequence is transferred to the memory address given in the operand based on the specified flag of the PSW.</td></tr>
</table>

JPE	Jump on parity even	P=1			
JPO	Jump on parity odd	P=0			
Opcode	Description	Flag Status	16-bit address	Unconditional subroutine call	The program sequence is transferred to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.
CC	Call on Carry	CY=1			
CNC	Call on no Carry	CY=0			
CP	Call on positive	S=0			
CM	Call on minus	S=1			
CZ	Call on zero	Z=1			
CNZ	Call on no zero	Z=0			
CPE	Call on parity even	P=1			
CPO	Call on parity odd	P=0			
RET			None	Return from subroutine unconditionally	The program sequence is transferred from the subroutine to the calling program.
Opcode	Description	Flag Status	None	Return from subroutine conditionally	The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW

<table>
<tr>
<td>
<table>
<tr><td>RC</td><td>Return on Carry</td><td>CY=1</td></tr>
<tr><td>RNC</td><td>Return on no Carry</td><td>CY=0</td></tr>
<tr><td>RP</td><td>Return on positive</td><td>S=0</td></tr>
<tr><td>RM</td><td>Return on minus</td><td>S=1</td></tr>
<tr><td>RZ</td><td>Return on zero</td><td>Z=1</td></tr>
<tr><td>RNZ</td><td>Return on no zero</td><td>Z=0</td></tr>
<tr><td>RPE</td><td>Return on parity even</td><td>P=1</td></tr>
<tr><td>RPO</td><td>Return on parity odd</td><td>P=0</td></tr>
</table>
</td>
<td></td>
<td></td>
<td>and the program execution begins at the new address.</td>
</tr>
<tr>
<td>PCHL</td>
<td>None</td>
<td>Load the program counter with HL contents</td>
<td>The contents of registers H & L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low order byte.</td>
</tr>
<tr>
<td>RST</td>
<td>0-7</td>
<td>Restart</td>
<td>The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.
<table>
<tr><td>Instruction</td><td>Restart Address</td></tr>
<tr><td>RST 0</td><td>0000H</td></tr>
</table>
</td>
</tr>
</table>

			RST 1 — 0008H RST 2 — 0010H RST 3 — 0018H RST 4 — 0020H RST 5 — 0028H RST 6 — 0030H RST 7 — 0038H The 8085 has additionally 4 interrupts, which can generate RST instructions internally and doesn't require any external hardware. Following are those instructions and their Restart addresses − Interrupt — Restart Address TRAP — 0024H RST 5.5 — 002CH RST 6.5 — 0034H RST 7.5 — 003CH

Arithmetic Instruction:

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADD K.

ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADC K
ADI	8-bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. Example − ADI 55K
ACI	8-bit data	Add the immediate to the accumulator with carry	The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ACI 55K
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16-bit data into the register pair designated in the operand. Example − LXI K, 3025M
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. Example − DAD K
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. Example − SUB K
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. Example − SBB K
SUI	8-bit data	Subtract the immediate from the accumulator	The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator. Example − SUI 55K

XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example – XCHG
INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. Example − INR K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. Example − INX K
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. Example − DCR K
DCX	R	Decrement the register pair by 1	The contents of the designated register pair are decremented by 1 and their result is stored at the same place. Example − DCX K
DAA	None	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit BCD digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. Example – DAA

Data transfer Instruction:

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination (Dt)	This instruction copies the contents of the source register into the destination register without any alteration. Example − MOV K, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. Example − MVI K, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. Example − LDA 2034K
LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. Example − LDAX K
LXI	Reg. pair, 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory. Example − LXI K, 3225L
LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. Example − LHLD 3225K
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example − STA 325K

STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. Example − STAX K
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example − SHLD 3225K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example – XCHG
SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H register provide the high-order address and the contents of the L register provide the low-order address. Example – SPHL
XTHL	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1). Example – XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence.

			The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example − PUSH K
POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. Example – POPK
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example − OUT K9L
IN	8-bit port address	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example − IN5KL

2.4-Simple Assembly Language Programming Of 8085- Simple Addition & Subtraction. Logic Operations (AND, OR, Complement 1's & 2's) & Masking Of Bits. Counters & Time Delay (Single Register, Register Pair, More Than Two Register). Looping, Counting & Indexing (Call/JMP Etc.). Stack & Subroutines Programs. Code Conversion, BCD Arithmetic & 16 Bit Data Operation, Block Transfer. Compare Between Two Numbers. Array Handling (Largest Number & Smallest Number In The Array).

Assembly language program:

Write 8085 Assembly language program to perform 8-bit addition without carry. The numbers are stored at F100, and F101. Result will be stored at F102.

Input

Address	Data
…	…
F100	CE
F101	21

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 01, F1		LXI H,F100H	Point to get the numbers
F003	7E		MOV A,M	Load first number to A
F004	23		INX H	Point to next operand
F005	86		ADD M	Add M with A
F006	23		INX H	Point to next location
F007	77		MOV M,A	Store result
F008	76		HLT	Terminate the program

Output

Address	Data
…	…
F102	EF

Write 8085 Assembly language program to subtract two 8-bit numbers and store the result at locations 8050H and 8051H.

Input

first input

Address	Data
. . .	. . .
8000	78
8001	5D
. . .	. . .

second input

Address	Data
. . .	. . .
8000	23
8001	CF

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	0E,00		MVIC,00H	Clear C register
F002	21,00, 80		LXIH,8000H	Load initial address to get operand
F005	7E		MOVA, M	Load Acc with the memory element
F006	23		INX H	Point to next location
F007	46		MOVB, M	Load B with the second operand

Address	HEX Codes	Labels	Mnemonics	Comments
F008	90		SUB B	Subtract B from A
F009	D2,0D, F0		JNC STORE	When CY = 0, go to STORE
F00C	0C		INR C	Increase C by 1
F00D	21,50, 80	STORE	LXIH,8050H	Load the destination address
F010	77		MOVM, A	Store the result
F011	23		INX H	Point to next location
F012	71		MOVM, C	Store the borrow
F013	76		HLT	Terminate the program

Output

first output

Address	Data
. . .	. . .
8050	1B
8051	00
. . .	. . .

second output

Address	Data
. . .	. . .
8050	54
8051	01

Write 8085 Assembly language program to multiply two 8-bit numbers stored in memory location and store the 16-bit results into the memory.

Input

Address	Data
. . .	. . .
8000	DC
8001	AC

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 00, 80		LXI H,8000H	Load first operand address
F003	46		MOV B, M	Store first operand to B
F004	23		INX H	Increase HL pair
F005	AF		XRA A	Clear accumulator
F006	4F		MOV C, A	Store 00H at register C

Address	HEX Codes	Labels	Mnemonics	Comments
F007	86	LOOP	ADD M	Add memory element with Acc
F008	D2, 0C, F0		JNC SKIP	When Carry flag is 0, skip next task
F00B	0C		INR C	Increase C when carry is 1
F00C	05	SKIP	DCR B	Decrease B register
F00D	C2, 07, F0		JNZ LOOP	Jump to loop when Z flag is not 1
F010	21, 50, 80		LXI H,8050H	Load Destination address
F013	71		MOV M, C	Store C register content into memory
F014	23		INX H	Increase HL Pair
F015	77		MOV M, A	Store Acc content to memory
F016	76		HLT	Terminate the program

Output

Address	Data
. . .	. . .
8050	93
8051	D0

Program for 1's complement and 2's complement of an 8-bit number. Using 8085 instruction

Input

Address	Data
. . .	. . .
8000	AB
. . .	. . .

Program

Address	HEX Codes	Mnemonics	Comments
F000	3A, 00, 80	LDA 8000H	Load the number from memory
F003	2F	CMA	Complement the accumulator
F004	32, 50, 80	STA 8050H	Store the 1's complemented result
F007	3C	INR A	Increase A by 1
F008	32, 51, 80	STA 8051H	Store the 2's complemented result
F00B	76	HLT	Terminate the program

Output

Address	Data
. . .	. . .

Address	Data
8050	54
8051	55
. . .	. .

Write an assembly language program in 8085 microprocessors to perform AND operation between lower and higher order nibble of 8-bit number.

...

Program –

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

Counters & Time delay (Single Register, Register Pair, more than Two Register)

Counters-

- ✓ A loop counter is set up by loading a register with a certain value
- ✓ Then using the DCR (to decrement) and INR (to increment) the contents of the register are updated.
- ✓ A loop is set up with a conditional jump instruction that loops back or not depending on whether the count has reached the termination count.

- ✓ Simple ALP for implementing a loop Using DCR instruction

```
        MVI C, 15H
LOOP    DCR C
        JNZ  LOOP
```

Using a Register Pair as a Loop Counter-

- ✓ Using a single register, one can repeat a loop for a maximum count of 255 times.
- ✓ It is possible to increase this count by using a register pair for the loop counter instead of the single register. A minor problem arises in how to test for the final count since DCX and INX do not modify the flags.
- ✓ However, if the loop is looking for when the count becomes zero, we can use a small trick by ORing the two registers in the pair and then checking the zero flag.

The following program is an example of a loop set up with a register pair as the loop counter.

```
        LXI B, 1000H
LOOP    DCX B
        MOV A, C
        ORA B
        JNZ  LOOP
```

Time Delays

- ✓ Each instruction passes through different combinations of Fetch, Memory Read, and Memory Write cycles.
- ✓ Knowing the combinations of cycles, one can calculate how long such an instruction would require to complete.
- ✓ B for Number of Bytes
- ✓ M for Number of Machine Cycles
- ✓ T for Number of T-State.

Delay loops

• We can use a loop to produce a certain amount of time delay in a program.

• The following is an example of a delay loop:

```
        MVI C, FFH      7 T-States
LOOP    DCR C           4 T-States
        JNZ  LOOP       10 T-States
```

- ✓ The first instruction initializes the loop counter and is executed only once requiring only 7 T States.
- ✓ The following two instructions form a loop that requires 14 T-States to execute and is repeated 255 times until C becomes 0.

The following is an example of a delay loop set up with a register pair as the loop counter.

```
        LXI B, 1000H    10 T-States
LOOP    DCX B           6 T-States
        MOV A, C        4 T-States
        ORA B           4 T-States
        JNZ  LOOP       10 T-States
```

Looping, Counting & Indexing (Call/JMP etc.)

Looping:

It is used to instruct the microprocessor unit to repeat tasks. A loop is set up by instructing MPU to change sequence of execution and perform the task given.This is accomplished by Jump Instructions.

Loops are of 2 types:

- Continuous (repeats a task continuously)
- Conditional (repeats a task until certain data conditions are met)

Indexing:

It means counting or referencing objects with sequential numbers. Data bytes are stored in memory location, and those data bytes are referred to by their memory location.

Counting:

This programming technique uses INR or DCR instructions. A loop is established to update count and each count is checked to determine whether it has reached final number and if not reached, then the loop is repeated.

Stack & Subroutines programs.

- ✓ The stack is a reserved area of the memory in RAM where we can store temporary information. The stack is a shared resource as it can be shared by the microprocessor and the programmer.
- ✓ The programmer can use the stack to store data. And the microprocessor uses the stack to execute subroutines.
- ✓ The 8085 has a 16-bit register known as the 'Stack Pointer.'
- ✓ The programmer can decide the starting address of the stack by loading the address into the stack pointer register at the beginning of a program.
- ✓ The stack works on the principle of First In Last Out.

Subroutine is assembly language

- ✓ A subroutine is a small program written separately from the main program to perform a particular task that you may repeatedly require in the main program.
- ✓ Essentially, the concept of a subroutine is that it is used to avoid the repetition of smaller programs.
- ✓ Subroutines are written separately and are stored in a memory location that is different from the main program.
- ✓ You can call a subroutine multiple times from the main program using a simple CALL instruction.

The conditional CALL statements in assembly language program.

We can use conditional CALL statements, too, according to our needs. These statements enter a subroutine only when a certain condition is met.

CC	Call at address if cy (carry flag) = 1
CNC	Call at address if cy (carry flag) = 0
CZ	Call at address if ZF (zero flag) = 1
CNZ	Call at address if ZF (zero flag) = 0
CPE	Call at address if PF (parity flag) = 1
CPO	Call at address if PF (parity flag) = 0
CN	Call at address if SF (signed flag) = 1
CP	Call at address if SF (signed flag) = 0

Advantages of using subroutines

- ✓ Subroutines avoid the repetition of instructions.
- ✓ They give an aspect of modular programming to the entire program.
- ✓ Improves efficiency by reducing the size of the memory needed to store the program.

Code conversion, BCD Arithmetic & 16 Bit data Operation, Block Transfer.

To write an assembly language program to convert an 8 bit binary data to BCD using 8085 microprocessor kit.

Memory Location	Hex Code	Label	Mnemonics		Comments
			Op code	Operand	
4100	0E,00		MVI	E,00	Clear „E" register (Hund)
4102	53		MOV	D,E	Clear „D" register (tens)
4103	3A,00,42		LDA	4200	Get the data in „A"
4106	C3,06,41	HUND	CPI	64	Compare the data with 64
4108	DA,11,41		JC	TEN	If content is less jump to ten
410B	D6, 64		SUI	64	Subtract data by 64
410D	IC		INR	E	Increment carry each time
410E	C3,06,41		JMP	HUND	Jump to hundred & repeat
4111	C3, 0A	TEN	CPI	0A	Compare the data with 0A
4113	DA,1C,41		JC	UNIT	If data is less jump to unit
4116	D6, 0A		SUI	0A	Subtract the data by 0A
4118	14		INR	D	Increment „D" each time
4119	C3,11,41		JMP	TEN	Jump to ten & repeat
411C	4F	UNIT	MOV	4A	Move the value „A" to „C"
411D	7A		MOV	A,D	Move the value „D" to „A"
411E	07		RLC		Rotate the value of „A"
411F	07		RLC		Of „A" so that
4120	07		RLC		Lower and upper niddle
4121	07		RLC		Gets exchanged
4122	81		ADD	C	Add „A" and „C"
4123	32,50,42		STA	42,50	Save ten" & units in „M"
4126	7B		MOV	A,E	Move to E to A
4127	32,51,42		STA	4251	Save hundreds unit in „A"
412A	76		HLT		Stop the program execution

Input

Input Address	Value
4200	54

Output

Output Address	Value
4250	84
4251	00

Result:

Thus the binary to BCD conversion was executed successfully

To write an assembly language program to convert BCD data to Binary data using 8085 microprocessor kit.

Memory Location	Hex Code	Label	Mnemonics		Comments
			Op code	Operand	
4100	3A,00,42		LDA	4200	Get the data in „A‟
4103	5E		MOV	E,A	Save in „E‟ register
4104	E6, F0		ANI	F0	Mark the lower nibble
4106	07		RLC		Rotate the upper
4107	07		RLC		To lower nibble
4108	07		RLC		And save in
4109	07		RLC		Register B
410A	47		MOV	B,A	Move it from „A‟ to „B‟
410B	AF		XRA	A	Clear the accumulator
410C	0E,0A		MVI	C,0A	Intialise „C‟ as „0A‟
410E	08		REP		
410F	0D		DCR	C	Decrement „C‟ register
4110	C2,0E,41		JNZ		Jump till value „C‟ is 0
4113	47		MOV	B,A	Move the value A to B
4114	7B		MOV	A,E	Get the BCD in „A‟
4115	E6, 0F		ANI	0F	Mark the upper nibble
4117	80		ADD	B	Add „A‟ and „B‟
4118	32,01,42		STA	4201	Save the binary data
411B	76		HLT		Stop the program Execution

Input

Input Address	Value
4200	68

Output

Output Address	Value
4201	44

16	68
	4-4

Result:

Thus the BCD to binary conversion was executed successfully

Program for Move a block of 16 data stored from 8050H to 805FH to a target location from 8070F to 807FH.

Address	Hex Codes	Label	Mnemonic	T-States	Comment
8000	21 50 80	START:	LXI H, 8050H	10	Setup HL pair as a pointer for source memory.
8003	11 70 80		LXI D, 8070H	10	Set up DE pair as a pointer for destination memory
8006	06 10		MVI B, 10H	7	Set up B to count 16 bytes
8008	7E	LOOP:	MOV A, M	7	Get data byte from source.
8009	12		STAX D	7	Store data byte as destination
800A	23		INX H	6	Point HL to next source location
800B	13		INX D	6	Point DE to next destination
800C	05		DCR B	4	Decrement count
800D	C2 08 80		JNZ LOOP	10	If counter is not 0, go back to transfer next byte.
8010	76		HLT	5	Stop

Total 17 Bytes				Total 69 T-States	

The Main program is started from location 8000H – 8010H.

Source Data Block from 8050H – 805FH.

Destination block from 8070H – 807FH

Input		**Output**	
Address	**Value**	**Address**	**Value**
8050H	00	8070H	00
8051H	11	8071H	11
8052H	22	8072H	22
8053H	33	8073H	33
8054H	44	8074H	44
8055H	55	8075H	55
8056H	66	8076H	66
8057H	77	8077H	77
8058H	88	8078H	88
8059H	99	8079H	99
805AH	AA	807AH	AA
805BH	BB	807BH	BB
805CH	CC	807CH	CC

805DH	DD	807DH	DD
805EH	EE	807EH	EE
805FH	FF	807FH	FF

Write 8085 Assembly language program to find the largest number from a block of bytes.

Input-

Address	**Data**
...	...
8000	06
8001	55
8002	22
8003	44
8004	11
8005	33
8006	66
...	...

Program

Address	**HEX Codes**	**Labels**	**Mnemonics**	**Comments**
F000	21, 00, 80		LXI H,8000H	Point to getarray size
F003	4E		MOV C, M	Get the size of array

Address	HEX Codes	Labels	Mnemonics	Comments
F004	23		INX H	Point to actual array
F005	46		MOV B, M	Load the first number into B
F006	0D		DCR C	Decrease C
F007	23	LOOP	INX H	Point to next location
F008	7E		MOV A, M	Get the next number from memory to Acc
F009	B8		CMP B	Compare Acc and B
F00A	DA, 0E, F0		JC SKIP	if B > A,then skip
F00D	47		MOV B, A	If CY is 0, update B
F00E	0D	SKIP	DCR C	Decrease C
F00F	C2, 07, F0		JNZ LOOP	When count is not 0, go to LOOP
F012	21, 00, 90		LXI H,9000H	Point to destination address
F015	70		MOV M, B	Store the minimum number
F016	76		HLT	Terminate the program

Output

Address	Data
...	...
9000	66
...	...

Program to Find the smallest number in an array of data in 8085 Microprocessor

Input

Address	Data
...	...
8000	06
8001	55
8002	22
8003	44
8004	11
8005	33
8006	66
...	...

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 00, 80		LXI H,8000H	Point to get array size
F003	4E		MOV C, M	Get the size of array
F004	23		INX H	Point to actual array
F005	46		MOV B, M	Load the first number into B

Address	HEX Codes	Labels	Mnemonics	Comments
F006	0D		DCR C	Decrease C
F007	23	LOOP	INX H	Point to next location
F008	7E		MOV A, M	Get the next number from memory to Acc
F009	B8		CMP B	Compare Acc and B
F00A	D2, 0E, F0		JNC SKIP	if B <= A,then skip
F00D	47		MOV B, A	If CY is 1, update B
F00E	0D	SKIP	DCR C	Decrease C
F00F	C2, 07, F0		JNZ LOOP	When count is not 0, go to LOOP
F012	21, 00, 90		LXI H,9000H	Point to destination address
F015	70		MOV M, B	Store the minimum number
F016	76		HLT	Terminate the program

Output

Address	Data
...	...
9000	11
...	.

8085 program to find smallest number between two numbers or compare between two Number

Input

First input

Address	Data
. . .	. . .
8000	FD
8001	23
. . .	. . .

Second input

Address	Data
. . .	. . .
8000	59
8001	75
. . .	. . .

Program

Address	HEX Codes	Label	Mnemonics	Comments
F000	21, 00, 80		LXI H,8000H	Point to the first number
F003	46		MOV B,M	Load the first number to B
F004	23		INX H	Point to next location
F005	7E		MOV A,M	Get the second number to A
F006	B8		CMP B	Compare B with A
F007	DA, 0B, F0		JC STORE	Is CY = 1,jump to Store
F00A	78		MOV A,B	Load A with second number
F00B	32, 50, 80	STORE	STA 8050H	Store the number into memory
F00E	76		HLT	Terminate the program

Output

First output

Address	Data
. . .	. . .
8050	23

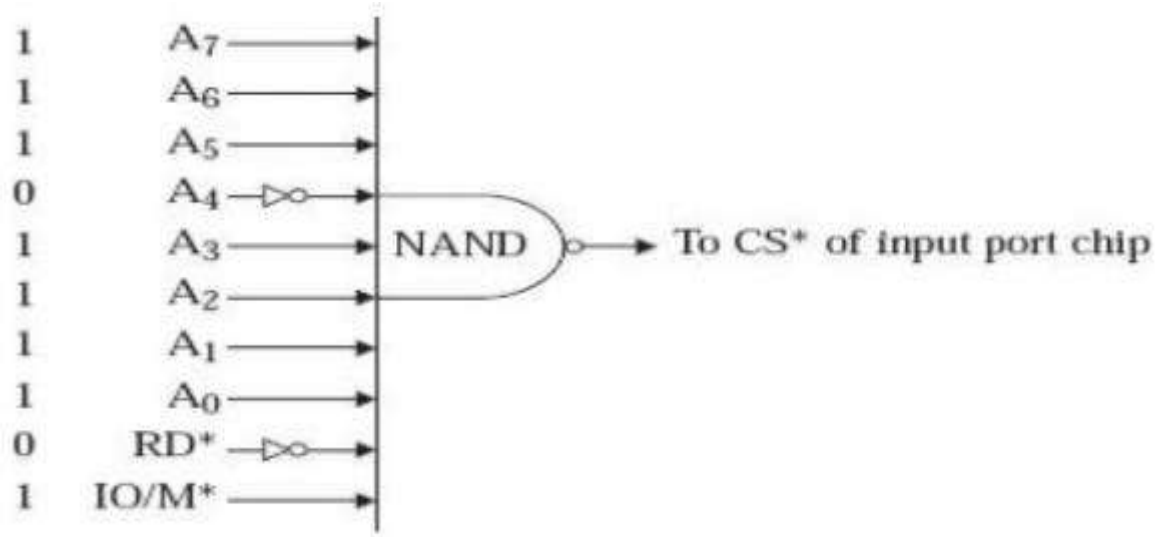
Address	Data
. . .	. . .

Second output

Address	Data
. . .	. . .
8050	59
. . .	. . .

2.5-Memory & I/O Addressing

- ✓ It is possible to address an I/O port as if it were a memory location. For example, let us say, the chip select pin of an I/O port chip is activated when address = FFF0H, IO/M* = 0, and RD* = 0.
- ✓ In the I/O port chip is selected when the 8085 is thinking that it is addressing memory location FFF0H for a read operation.
- ✓ Note that 8085 thinks that it is addressing a memory location because it has sent out IO/M* as a logic 0. But in reality, an input port has been selected, and the input port supplies information to the 8085.
- ✓ Such I/O ports that are addressed by the processor as if they were memory locations are called memory-mapped I/O ports.



- ✓ In the memory location we address an Input Output port. An example to be cited as when address = FFF0H, IO/M* = 0, and RD* = 0.
- ✓ Here we select the Input Output port chip when 8085 microprocessor finds that it is memory allocated location as it is sent out like IO/M* as a logic 0.
- ✓ In real world we select an Input Port which supplies information to 8085 Microprocessor. Like the memory locations 8085 microprocessor gets addressed by the processor which are called memory-mapped Input Output ports.

There is a set of instructions for this memory-mapped I/O operations. E.g. STA, LDA etc.

Mnemonics, Operand	**Opcode (in HEX)**	**Bytes**
STA Address	32	3

- ✓ Let us consider **STA 4050H** as an example instruction of this type. It is a 3-Byte instruction.
- ✓ The first Byte will contain the opcode hex value 32H. As in 8085 assembly language coding supports low order Byte of the address should be mentioned at first then the high order Byte of the address should be mentioned next.
- ✓ So next Byte in memory will hold 50H and after that 40H will be kept in the last third Byte.
- ✓ Let us suppose the initial content of Accumulator is ABH and initial content of memory location 4050H is CDH.
- ✓ So after execution, Accumulator content will remain as ABH and 4050H location's content will become ABH replacing its previous content CDH.
- ✓ The content tracing of this instruction has been shown below −

	Before	**After**
(A)	ABH	ABH
(4050H)	CDH	ABH

POSSIBLE SHORT TYPE QUESTIONS WITH ANSWER.

1-How many types of instruction sets are there.

Ans-According to the length the instruction of 8085 MP is classified into three types.

1-Single byte instruction

2-Two byte instruction

3-Three byte instruction

Q.2-Define Single byte instruction.

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

Q 3- Define addressing mode of 8085 MP

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

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

i-Immediate Addressing Mode

ii-Register Addressing Mode

iii-Direct Addressing Mode

iv-Register Indirect Addressing Mode

v-implied/Implicit Addressing Mode

Q 5-Define Implicit addressing mode of 8085 MP

In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.

Examples- MA (finds and stores the 1's complement of the contains of accumulator A in A) RRC (rotate accumulator A right by one bit)RLC (rotate accumulator A left by one bit)

Q 6- Define Immediate Addressing Mode

In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

Examples:

MVI B 45 (move the data 45H immediately to register B)

LXI H 3050 (load the H-L pair with the operand 3050H immediately)JMP address (jump to the operand address immediately)

Q 7- Define STA 16 bit address

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

Q 8- Define XCHG in 8085 MP. (19-W)

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

Q 09- Define branching instruction with example.

Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executinga branch instruction. The three types of branching instructions are: Jump (unconditional and conditional) Call (unconditional and conditional) Return (unconditional and conditional)

Example-JC, JNC, JP, JM etc

LONG TYPE QUESTIONS

1-Differenciate between 1 byte ,2-byte ,3-byte instruction with examples.

2-Explain different types of addressing mode in 8085 MP with examples. (19-W) (S-24)

3-Explain RIM and Sim instruction.

4-Write an assembly language programme of addition of two 8 bit no result 16 bit.

5-Write an assembly language programme to find smallest number in an array of data.

6-Write an assembly language programming to find the largest number in a given data array using 8085 instructions. (S-24)

Chapter No.- 3:
Timing diagrams

Learning Objectives:

3.1. Define opcode, operand, T-State, Fetch cycle, Machine Cycle, Instruction cycle & discuss the concept of timing diagram.

3.2. Draw timing diagram for memory read, memory write, I/O read, I/O write machine cycle.

3.3. Draw a neat sketch for the timing diagram for 8085 instruction (MOV, MVI, LDA instruction).

3.1-Define Opcode, Operand, T-State, Fetch Cycle, Machine Cycle, Instruction Cycle & Discuss The Concept Of Timing Diagram.

OPCODE

- ✓ Opcode is the first part of an instruction that tells the computer what function to perform and is also called Operation codes.
- ✓ Opcodes are the numeric codes that hold the instructions given to the computer system.
- ✓ These are the instructions that describe the CPU what operations are to be performed. The computer system has an operation code or opcode for each and every function given to it.

OPERAND

- ✓ Operand is another second part of instruction, which indicates the computer system where to find the data or instructions or where to store the data or instructions.
- ✓ The number of operands varies amongst different computer systems. Each instruction indicates the Control Unit of the computer system what to perform and how to perform it.
- ✓ The operations are Arithmetic, Logical, Branch operation, so on depending upon the problem that is provided to the computer.

T-state:

- ✓ One time period of frequency of microprocessor is called t-state.
- ✓ A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.
- ✓ Fetch cycle takes four t-states and execution cycle takes three t-states

Fetch cycle:

- ✓ The first byte of an instruction is its op-code.
- ✓ An instruction may be more than one byte long.
- ✓ The other bytes are data or operand address.
- ✓ The program counter (PC) keeps the memory address of the next instruction to be executed.
- ✓ In the beginning of a fetch cycle the content of the program counter, which is the address of the memory location where op-code is available, is sent to the memory.
- ✓ ·The memory places the op-code on the data bus so as to transfer it to the microprocessor.
- ✓ The entire operation of fetching an op-code takes three clock cycles.

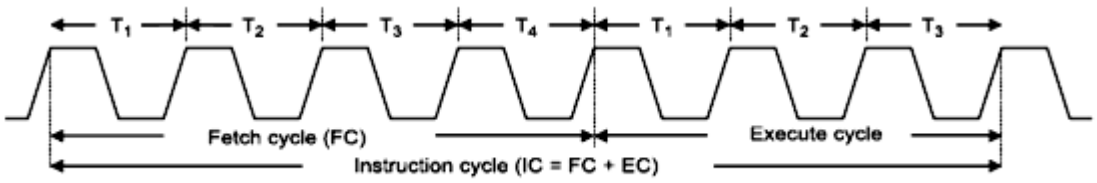
Machine Cycle:

- ✓ Machine cycle is defined as the time required for completing the operation of accessing either memory or I/O device.
- ✓ In the 8085, the machine cycle may consist of three to six T states.
- ✓ The T-state is defined as one sub-division of the operation performed in one clock period.
- ✓ ·In every machine cycle the first operation is op-code fetch and the remaining will be read or write from memory or IO devices.

Instruction Cycle:

- ✓ ·An instruction is a command given to the microprocessor to perform a specific operation on the given data.
- ✓ ·Sequence of instructions written for a processor to perform a particular task is called a program.
- ✓ The microprocessor fetches one instruction from the memory at a time & executes it. It executes all the instructions of the program one by one to produce the final result.
- ✓ In other words, an instruction cycle is defined as the time required completing the execution of an instruction.
- ✓ An instruction cycle consists of a fetch cycle and an execute cycle.
- ✓ The time required to fetch an opcode (fetch cycle) is a fixed slot of time while the time required to execute an instruction (execute cycle) is variable which depends on the type of instruction to be executed.

Instruction cycle (IC) = Fetch cycle (FC) Execute cycle (EC)



Timing diagram of various signals

Address latch enable:

- ✓ Address latch enable is an active high signal. (i.e.) the latch becomes enabled when the signal is high.
- ✓ It is activated during the beginning of T1 state of each machine cycle and it remains active in the T1state. But in case of bus idle machine cycle it is not activated.

Data Bus (D0-D7):

- ✓ Two types of data flow are possible. The data can be transferred from memory to microprocessor and vice versa.
- ✓ This process occurs during the T2 and T3 states.
- ✓ There are 2 cycles. One is Read machine cycle and the other is write machine cycle. In read machine cycle, the data will appear during the later part of T2 state, while in Write machine cycle the data will appear on the beginning of T2 state.

Lower byte address (A0-A7):

- ✓ The lower byte of address is available on the time multiplexed address/date bus during the T1 state of machine cycle, except the bus idle machine cycle.

Higher byte addresses (A8-A15):

- ✓ The higher byte addresses (A8-A15) is available for T1, T2 and T3 states of each machine cycle, except the bus idle machine cycle.

IO/M', S0, S1:

- ✓ From the previous discussions about 8085 microprocessors, we very well know that IO/M', S0, S1 are the status signals of the microprocessor.
- ✓ These status signals decide the type of machine cycle is to be executed.
- ✓ So they remain activated from the beginning T1 state of a particular machine cycle and remains till the end of that machine cycle.

RD' and WR':

- ✓ These 2 signals RD' and WR' decides the direction of the data transfer.
- ✓ RD' is Active: When RD' goes active, the data is transmitted from memory, I/O device or any other peripherals to the microprocessor.

- ✓ WR' is active: When WR' goes active, the data is transmitted from microprocessor to the memory or any other peripheral devices.
- ✓ In 8085 microprocessors either RD' goes high or WR' goes high. Both cannot take place at same time.
- ✓ The data transfer both RD' and WR' takes place during T2 and T3 states of machine cycle. So these signals are activated during the T2 and T3 states.

3.2-Draw Timing Diagram For Memory Read, Memory Write, I/O Read, I/O Write Machine Cycle.

Various operation of 8085 MP

To execute a program, 8085 performs various operations as:

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

Opcode Fetch Machine Cycle:

- ✓ It is the first step in the execution of any instruction.
- ✓ 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

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

T2 clock cycle

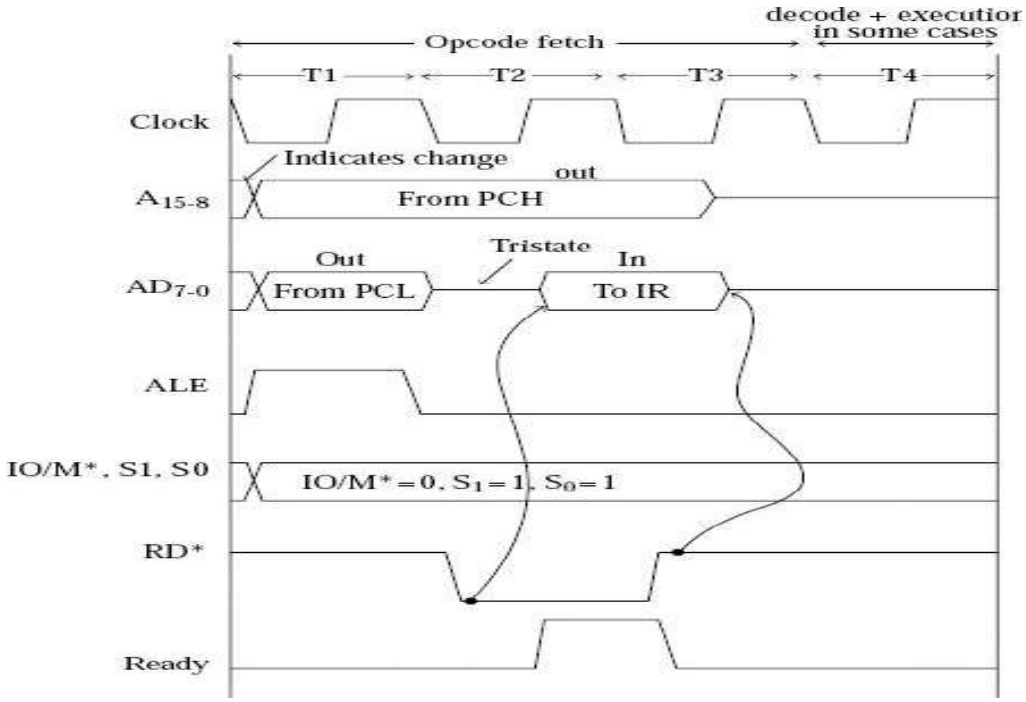
- ✓ Multiplexed address bus is now change to Data bus.
- ✓ The RD signal is made low by the processor. This signal makes the memory device load the data bus with the contents of the location addressed by the processor.

T3 clock cycle

- ✓ The opcode available on the data bus is read by the processor and moved to the instruction register.
- ✓ The RD signal is deactivated by making it logic 1.

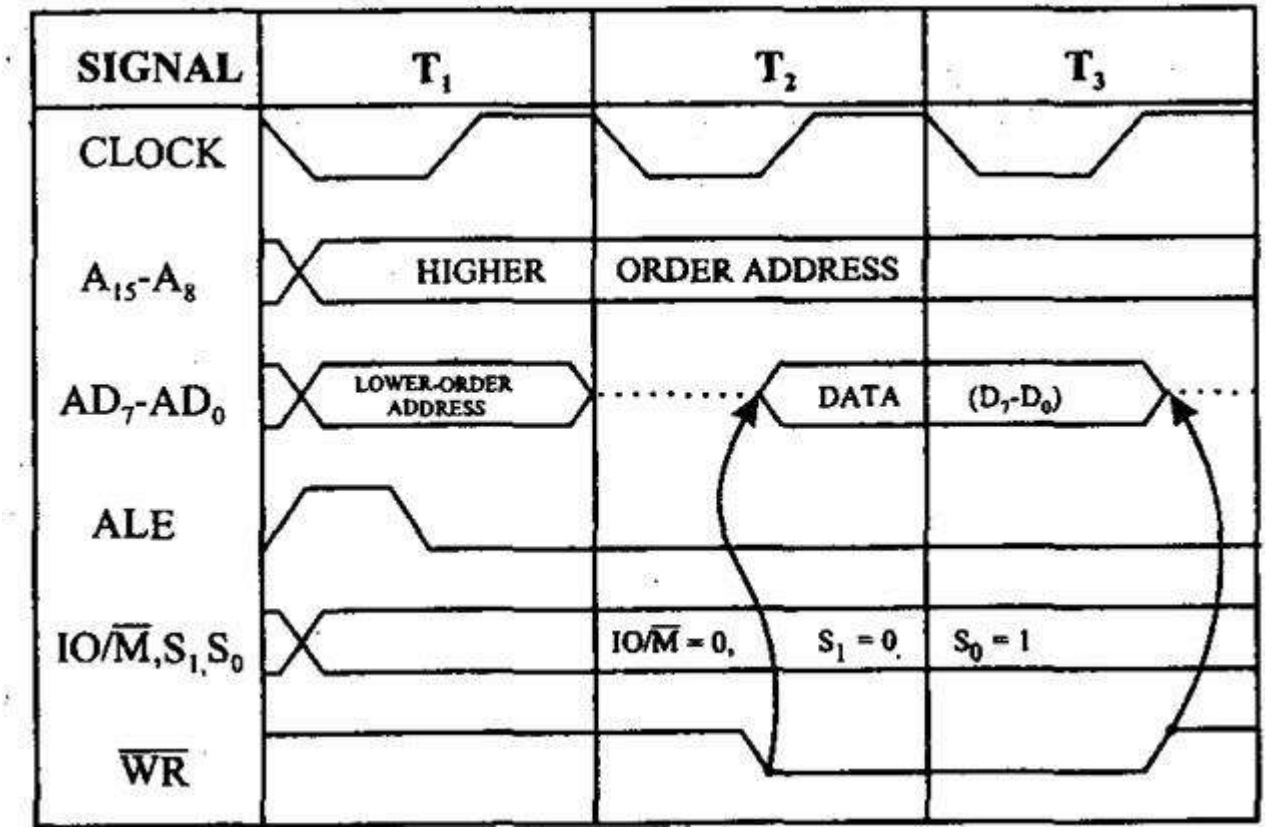
T4 clock cycle

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



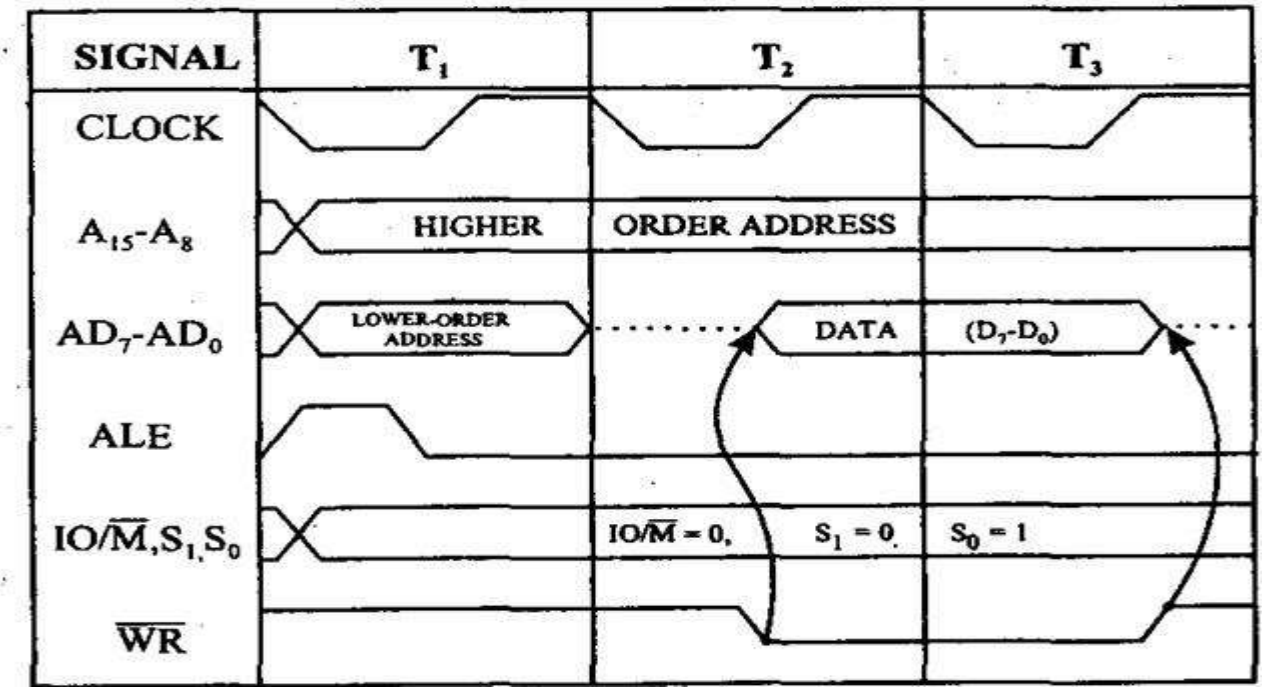
Memory Read Machine Cycle of 8085:

- ✓ Single byte instructions require only Opcode Fetch machine cycles. But, 2-byte and 3-byte instructions require additional machine cycles to read the operands from memory.
- ✓ The additional machine cycle is called Memory Read machine cycle.
- ✓ For example, the instruction MVI A, 50H requires one OF machine cycle to fetch the operand from memory and one MR machine cycle to read the operand (50H) from memory.
- ✓ The MR machine cycle takes 3 T-states.



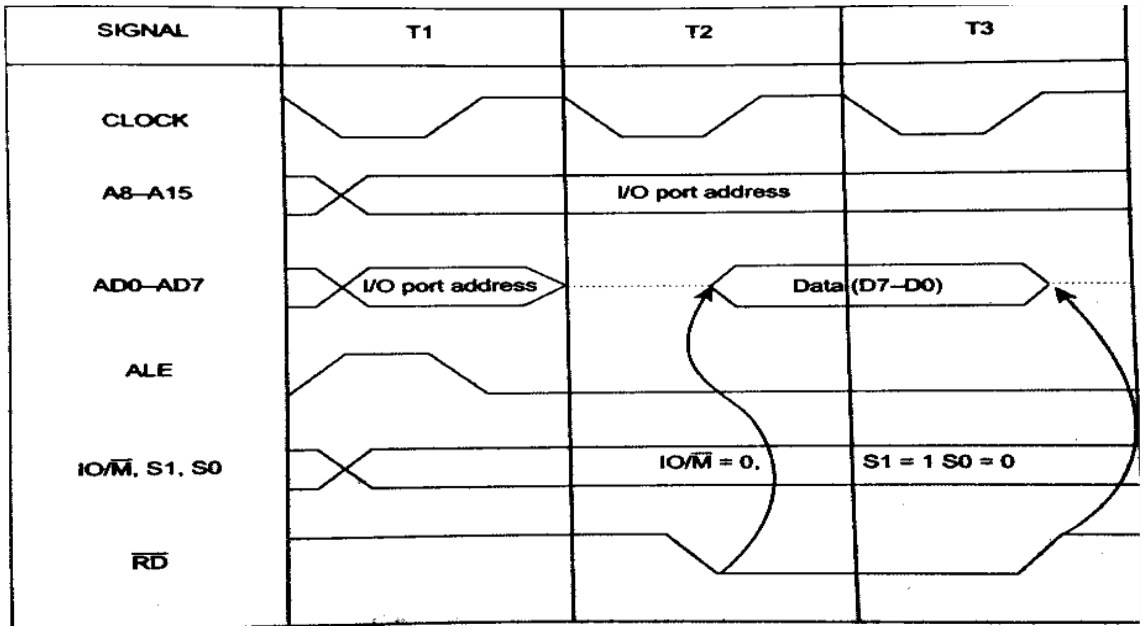
Memory Write Machine cycle:

- ✓ 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 WR signal is made low.
- ✓ The timing diagram of this cycle is given in Fig.



I/O Read Cycle:

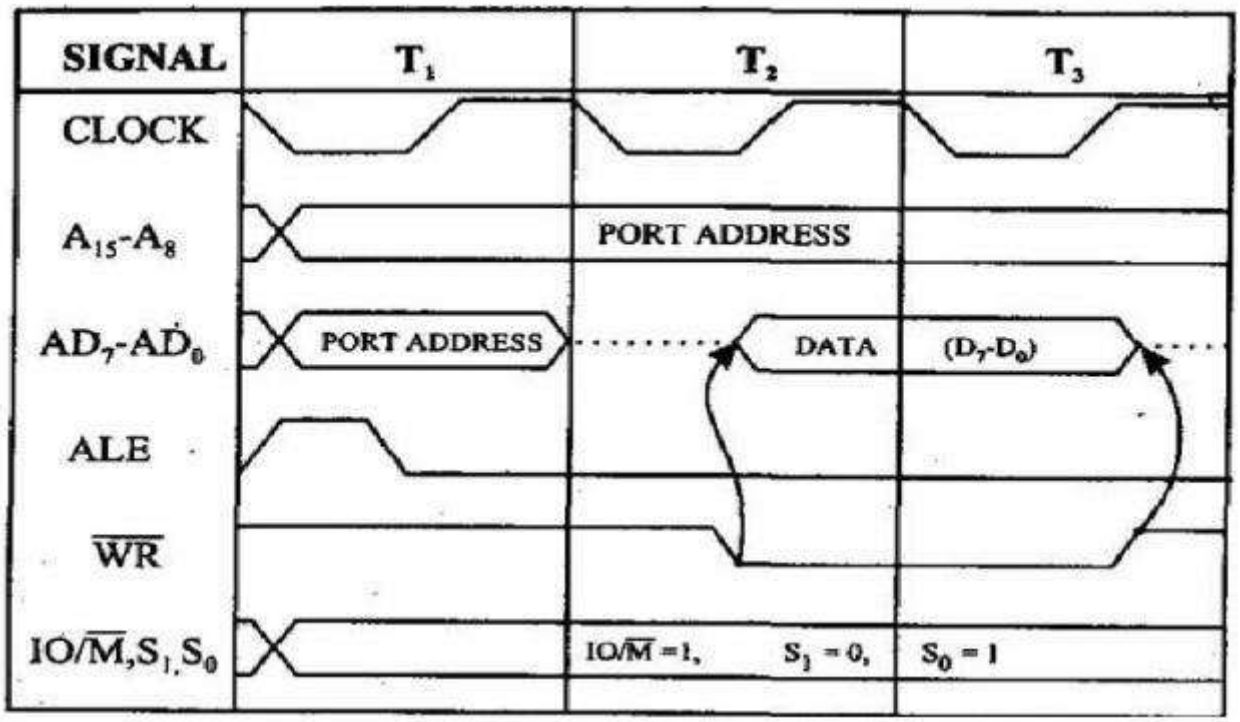
- ✓ The I/O read cycle is executed by the processor to read a data byte from I/O port or from peripheral, which is I/O mapped in the system.
- ✓ The 8-bit port address is placed both in the lower and higher order address bus.
- ✓ The processor takes three T-states to execute this machine cycle. The timing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



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 executethis machine cycle.

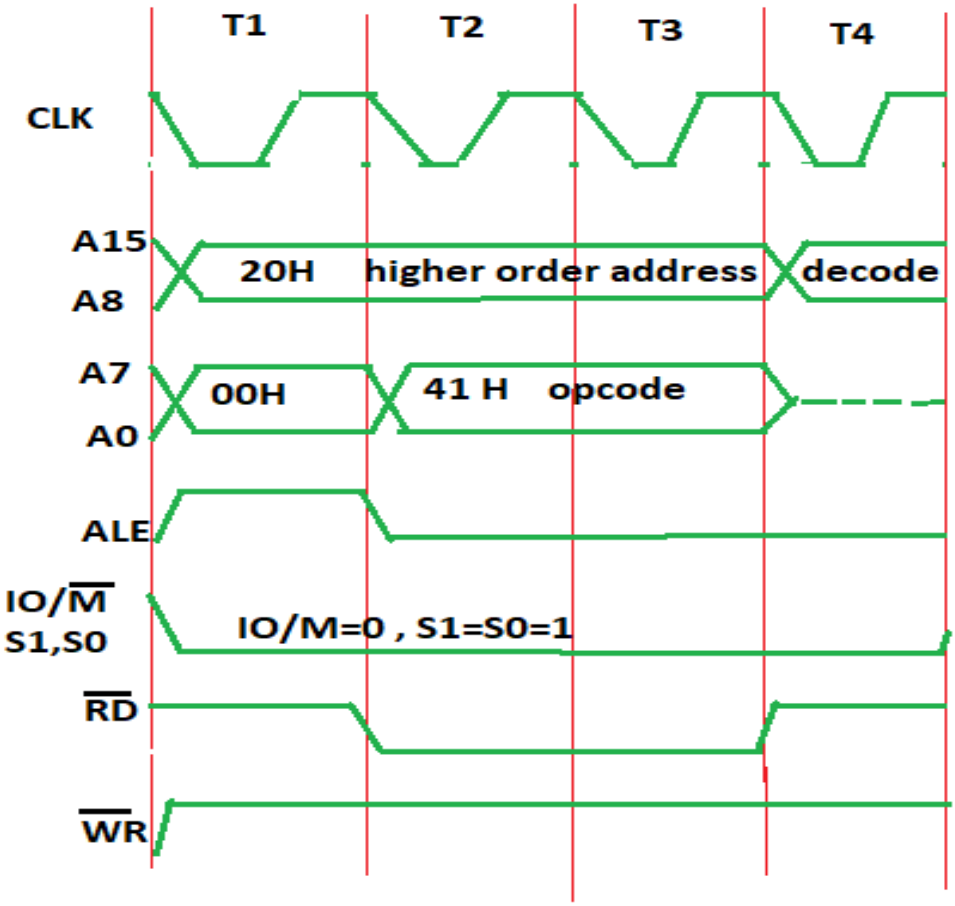
- ✓ The timing diagram of this cycle is given in Fig.



3.3-Draw A Neat Sketch For The Timing Diagram For 8085 Instruction (MOV, MVI, LDA Instruction).

Draw the timing diagram of the given instruction in 8085

```
MOV B, C
```



- ✓ Only opcode fetching is required for this instruction and thus we need 4 T states for the timing diagram. For the opcode fetch the IO/M (low active) = 0, S1 = 1 and S0 = 1.

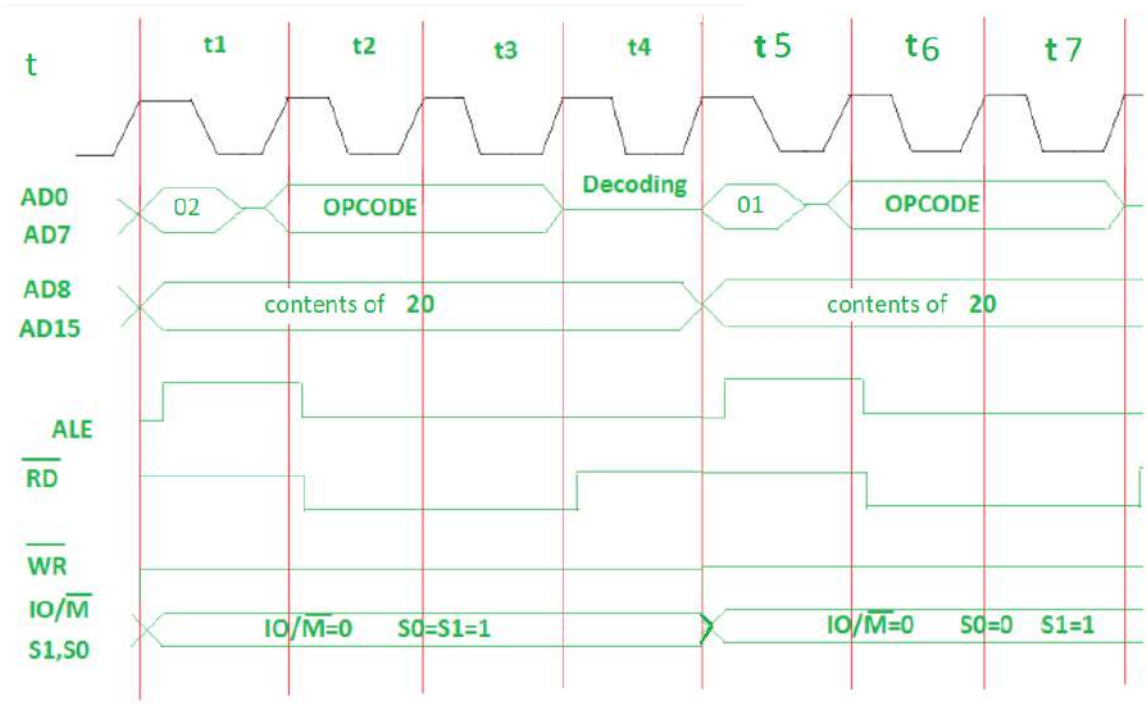
In Opcode fetch (t1-t4 T states):

- ✓ 00 – lower bit of address where opcode is stored, i.e., 00
- ✓ 20 – higher bit of address where opcode is stored, i.e., 20.
- ✓ ALE – provides signal for multiplexed address and data bus. Only in t1 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- ✓ RD (low active) – signal is 1 in t1 & t4 as no data is read by microprocessor. Signal is 0 in t2 & t3 because here the data is read by microprocessor.
- ✓ WR (low active) – signal is 1 throughout, no data is written by microprocessor.
- ✓ IO/M (low active) – signal is 1 in throughout because the operation is performing on memory.
- ✓ S0 and S1 – both are 1 in case of opcode fetching.

Draw the timing diagram of the following code,

MVI B, 45

- ✓ Opcode: MVI
- ✓ Operand: B is the destination register and 45 is the source data which needs to be transferred to the register.
- ✓ 45’ data will be stored in the B register.
- ✓ The opcode fetch will be same in all the instructions.
- ✓ Only the read instruction of the opcode needs to be added in the successive T states.
- ✓ For the opcode fetch the IO/M (low active) = 0, S1 = 1 and S0 = 1. Also, 4 T states will be required to fetch the opcode from memory.
- ✓ For the opcode read the IO/M (low active) = 0, S1 = 1 and S0 = 0. Also, only 3 T states will be required to read data from memory.



In Opcode fetch (t1-t4 T states):

- ✓ 00 – lower bit of address where opcode is stored, i.e., 00
- ✓ 20 – higher bit of address where opcode is stored, i.e., 20.
- ✓ ALE – provides signal for multiplexed address and data bus. Only in t1 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- ✓ RD (low active) – signal is 1 in t1 & t4 as no data is read by microprocessor. Signal is 0 in t2 & t3 because here the data is read by microprocessor.
- ✓ WR (low active) – signal is 1 throughout, no data is written by microprocessor.
- ✓ IO/M (low active) – signal is 1 in throughout because the operation is performing on memory.
- ✓ S0 and S1 – both are 1 in case of opcode fetching.

Timing diagram of LDA Instruction

- ✓ In 8085 Instruction set, **LDA** is a mnemonic that stands for LoaD Accumulator with the contents from memory.
- ✓ In this instruction Accumulator will get initialized with 8-bit content from the 16-bit memory address as indicated in the instruction as a16.
- ✓ This instruction uses absolute addressing for specifying the data.
- ✓ It occupies 3-Bytes in the memory. First Byte specifies the opcode, and the successive 2-Bytes provide the 16-bit address, i.e. 1-Byte each for each memory location.

Mnemonics, Operand	Opcode(in HEX)	Bytes
LDA Address	3A	3

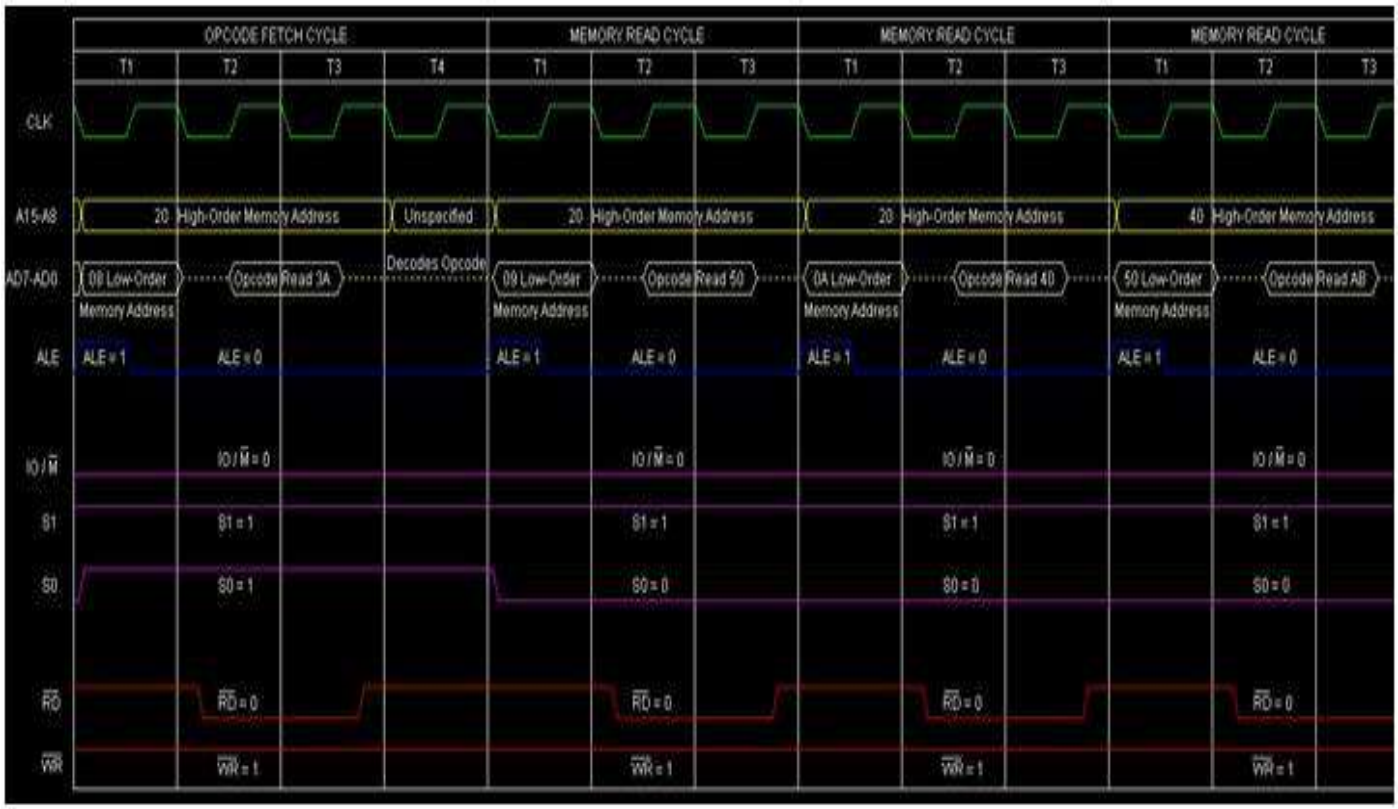
- ✓ Let us consider **LDA 4050H** as an example instruction of this type. It is a 3-Byte instruction.
- ✓ The initial content of memory address 4050H is ABH. Initially Accumulator content is CDH.
- ✓ As after execution A will be initialized with value ABH. Memory location 4050H will still remain with the content ABH.

	Before	After
(4050)	ABH	ABH

A	CDH	ABH

Address	Hex Codes	Mnemonic	Comment
2008	3A	LDA 4050H	A <- Content of the memory location 4050H
2009	50		Low order Byte of the address
200A	40		High order Byte of the address

Here is the timing diagram of the instruction **LDA 4050H**



POSIBLE SHORT TYPE QUESTIONS WITH ANSWERS

Q 1-Define Opcode and Operand. (19-W)

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

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

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

Machine cycle:

The basic microprocessor operation such as reading a byte from I/O port or writing a byte to memory is called asmachine cycle.

The time TCY in the above figure is called as the machine cycle. Thus a machine cycle consists of several T-states.

T-state:

One complete cycle of clock is called as T-state as shown in the above figure. The time intervals T1T1 orT2T2 are theexamples of T-state.

A T-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse. Various versions of 8086 have maximum clock frequency from 5MHz to 10MHz. Hence the minimum time for one T-state is between 100 to 200 n sec.

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

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

Q4- Write number of machine cycle and addressing mode for the given instruction of 8085 microprocessor MVI B, 10H. (S-24)

Ans: -The "MVI B, 10H" instruction requires 2 machine cycles that is
Opcode fetch and Memory read machine cycle.
Immediate addressing mode is used.

POSSIBLE LONG TYPE QUESTIONS

1-Draw the timing diagram of the instruction LDAX B.
2-Draw the taming diagram of Opcode fetch machine cycle.
3-Draw and explain the timing diagram of the instruction IN 82H.
4-Draw a timing diagram of LXI D,2500H with a neat sketch.

Chapter No.: 4
Microprocessor Based System Development Aids.

Learning Objectives-

4.1-Concept of interfacing
4.2-Define Mapping &Data transfer mechanisms - Memory mapping & I/O Mapping.
4.3- Concept of Memory Interfacing: - Interfacing EPROM & RAM Memories
4.4- Concept of Address decoding for I/O devices
4.5- Programmable Peripheral Interface: 8255.
4.6- ADC & DAC with Interfacing.
4.7- Interfacing Seven Segment Displays
4.8- Generate square waves on all lines of 8255
4.9- Design Interface a traffic light control system using 8255.
4.10- Design interface for stepper motor control using 8255.
4.11- Basic concept of other Interfacing DMA controller, USART

4.1-Concept Of Interfacing

Interface-

- ✓ Interface is the path for communication between two components.
- ✓ Interfacing is of two types, memory interfacing and I/O interfacing.

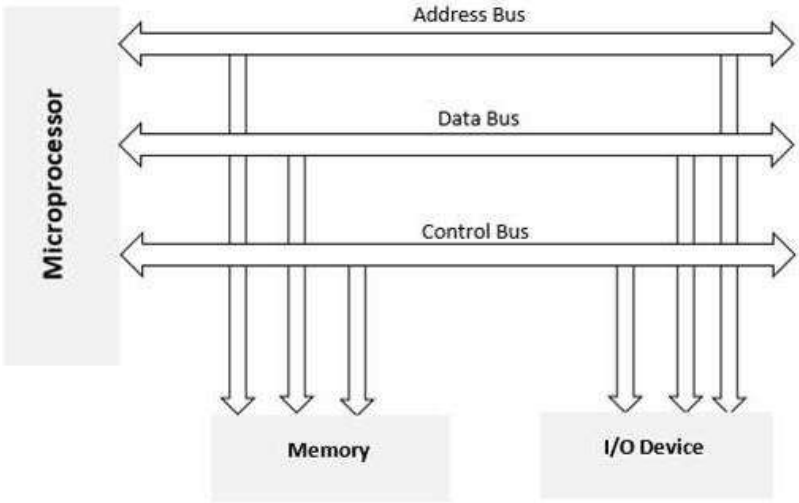
Memory Interfacing

- ✓ When we are executing any instruction, we need the microprocessor to access the memory for reading instruction codes and the data stored in the memory.
- ✓ For this, both the memory and the microprocessor requires some signals to read from and write to registers.
- ✓ The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.

IO Interfacing

- ✓ There are various communication devices like the keyboard, mouse, printer, etc.
- ✓ So, we need to interface the keyboard and other devices with the microprocessor by using latches and buffers. This type of interfacing is known as I/O interfacing.

Block Diagram of Memory and I/O Interfacing



4.2-Define Mapping &Data Transfer Mechanisms - Memory Mapping & I/O Mapping.

Mapping.

- ✓ The memory mapping is used to transfer the logical address space into physical memory but sometimes physical memory is a smaller size.
- ✓ The microprocessor can access external memory. The memory mapping used for increased access to physical memory.

Memory mapping

- ✓ In memory mapping of I/O devices, the I/O ports are assigned 16-bit address within the memory.
- ✓ Here each bus is common thus the same set of instructions is used for memory and I/O devices.
- ✓ Thus, I/O is considered as memory and the same address space is used by both memory and I/O devices.
- ✓ This reduces the addressing capability of the memory.
- ✓ In this case, the processor considers the I/O ports as memory locations for the purpose of reading and writing.

I/O Mapping.

- ✓ It is also known **as** Isolated I/O mapping and the reason for the same is that here the address space of memory and I/O are separated from each other.
- ✓ Thus, different read and write instructions are used for I/O and memory and there is a common bus for I/O devices and memory however, individual read and write control lines are used for I/O.
- ✓ Here the operation takes place in a way that, if the data over which operation is to be performed is to be collected from the I/O devices then address is placed on the address line and I/O read and I/O write control lines will get activated so that data transfer can be performed between the processor and I/O.
- ✓ For the transfer of data between the processor and I/O devices, only IN and OUT instructions are used in the isolated mapping. The required chip select signals in this case are generated by an individual decoding unit.

4.3- Concept Of Memory Interfacing:- Interfacing Eprom & Ram Memories

Concept of Memory Interfacing

For Memory Interfacing in 8085

1. Microprocessor 8085 can access 64Kbytes memory since address bus is 16-bit. But it is not always necessary to use full 64Kbytes address space. The total memory size depends upon the application.
2. Generally, EPROM (or EPROMs) is used as a program memory and RAM (or RAMs) as a data memory. When both, EPROM and RAM are used, the total address space 64Kbytes is shared by them.
3. The capacity of program memory and data memory depends on the application.
4. It is not always necessary to select 1 EPROM and 1 RAM. We can have multiple EPROMs and multiple RAMs as per the requirement of application.
5. We can place EPROM/RAM anywhere in full 64 Kbytes address space. But program memory (EPROM) should be located from address 0000H since reset address of 8085 microprocessor is 0000H.

Memory Interfacing: -

- ➢ Interfacing is a technique to be used for connecting the Microprocessor to Memory.
- ➢ Now a days Semiconductor memories are used for storing purpose.
- ➢ There are some of the advantages of the semiconductor memory.

  Small size

  High speed

Better reliability
Low cost

- Generally, RAM or ROM is used for memory interfacing.

Memory: -

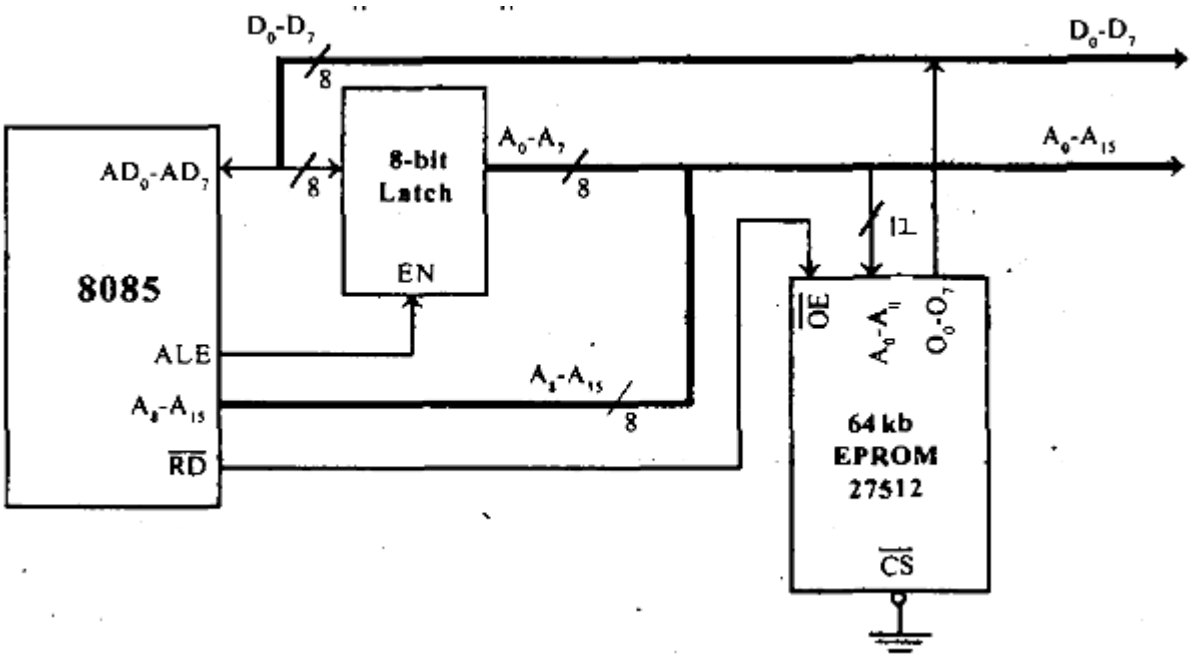
- A memory is a digital IC which stores the data in binary form.

Memory Size: -

- The number of location and number of bits per word will vary from memory to memory.
- For example, if a particular memory chip is capable of storing M words with each word having N-bits. Then the size of the memory will be M× N.

Interfacing a ROM memory of 4096*8 with 8085 Microprocessor: -

- Given memory size = 4096 * 8
  4096 =2^12.
  So 12 lines will be used for interfacing. A0 to A11
- In this system A0 to A11 lines of Microprocessor will be connected to the address lines of the memory. And D0 to D7 of the 8085 microprocessor will be connected to the data bus of the memory.
- As we know that the it is EPROM, so only RD pin is connected to the microprocessor. There is not the facility for writing data.
- In case if you are using RAM then you have to connect one more pin for writing operation.



- As we see that there is a pin named CS. Generally this pin is used for Selection for the chip in case of two or more than memory chip.
- Latch has been used to separate the data and address bus.
- Now come to the main part when the 8085 wants to read from and write into memory, it activates IO/M, RD and WR signals as shown in Table.

IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	Operation
0	0	1	8085 reads data from memory
0	1	0	8085 writes data into memory

- Table shows the status of IO/M, RD and WR signals during memory read and write operations.

4.4- Concept Of Address Decoding For I/O Devices

Address Decoding

- Microprocessor system includes memory devices and I/O devices.
- It is important to note that microprocessor can communicate (read/write) with only one device at a time, since the data, address and control buses are common for all the devices.
- In order to communicate with memory or I/O devices, it is necessary to decode the address from the microprocessor.
- Due to this each device (memory or I/O) can be accessed independently.
- The following section describes common address decoding techniques.

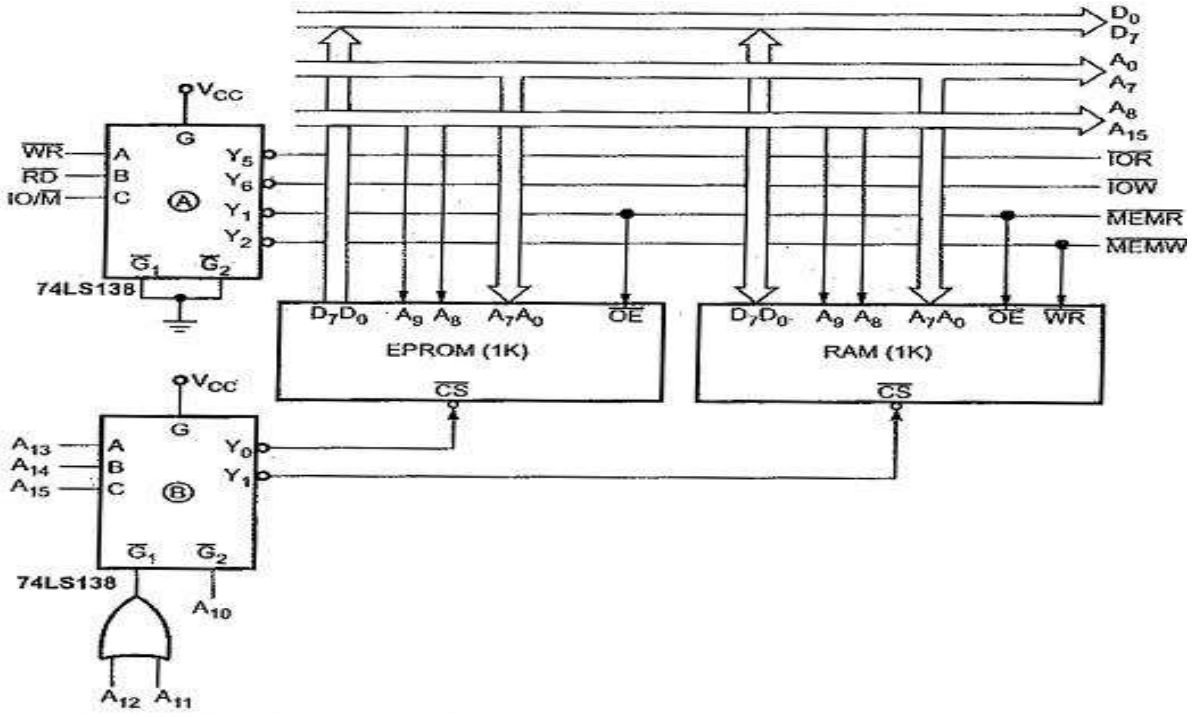
Address Decoding Techniques:

Address decoding technique are two types:

1. **Absolute decoding/Full Decoding**
2. **Linear decoding/Partial Decoding**

Absolute decoding:

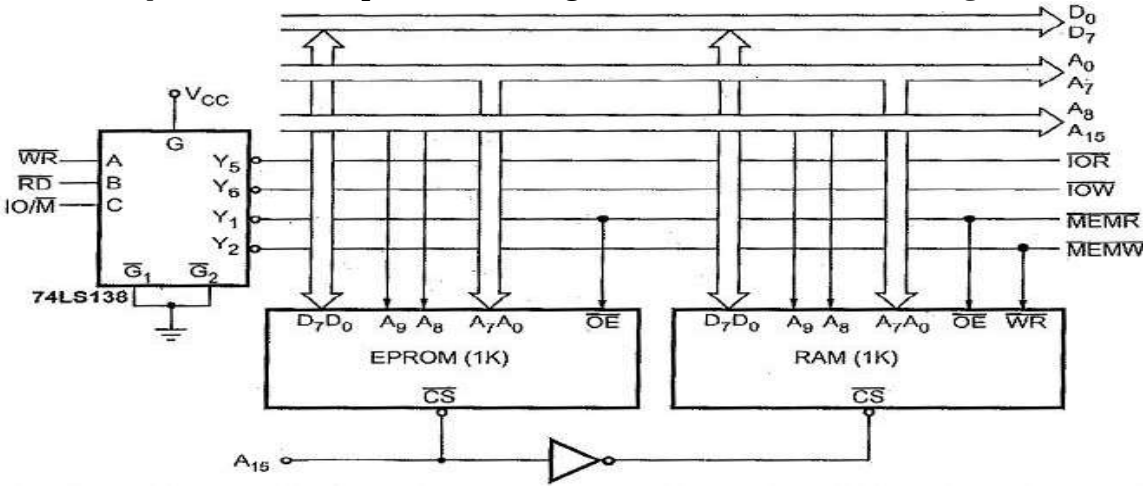
- In absolute decoding technique, all the higher address lines are decoded to select the memory chip, and the memory chip is selected only for the specified logic levels on these high-order address lines; no other logic levels can select the chip.
- This addressing technique is normally used in large memory systems.



Linear decoding:

- In small systems, hardware for the decoding logic can be eliminated by using individual high-order address lines to select memory chips.
- This is referred to as linear decoding. Figure shows the addressing of RAM with linear decoding technique.

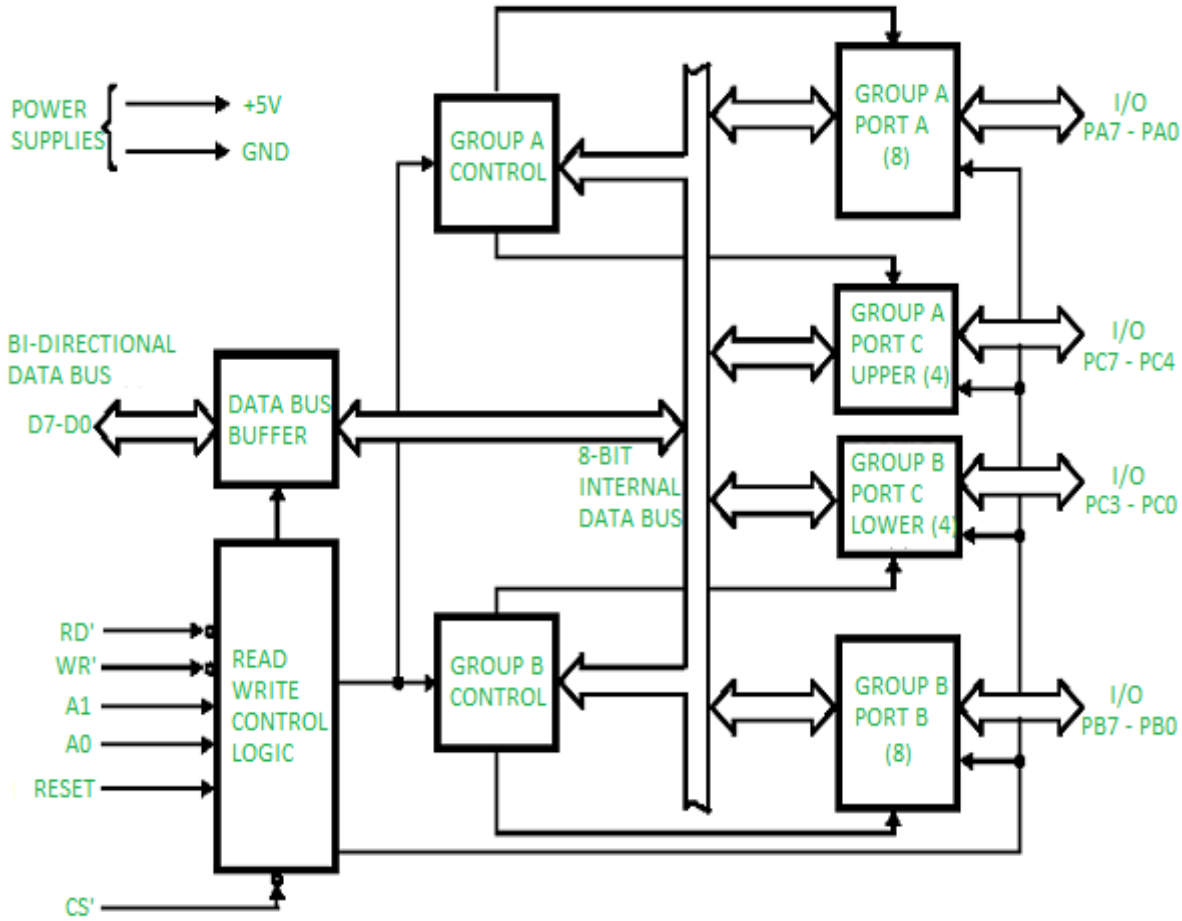
- This technique is also called **partial decoding**. It reduces the cost of decoding circuit.



4.5-Programmable Peripheral Interface: 8255.

- PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc.
- It consists of three 8-bit bidirectional I/O ports i.e. PORT A, PORT B and PORT C. We can assign different ports as input or output functions.

Block diagram –



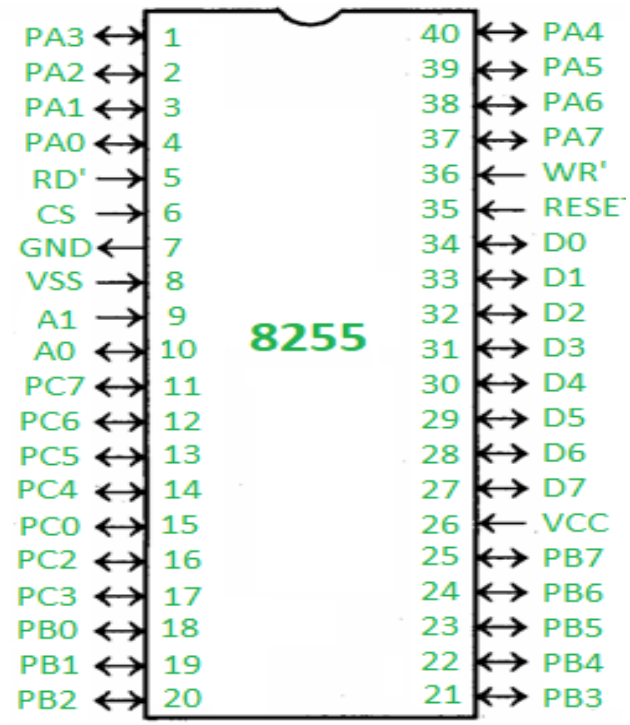
- It consists of 40 pins and operates in +5V regulated power supply.
- Port C is further divided into two 4-bit ports i.e. port C lower and port C upper
- Port C can work in either BSR (bit set rest) mode or in mode 0 of input-output mode of 8255.
- Port B can work in either mode 0 or in mode 1 of input-output mode.

- Port A can work either in mode 0, mode 1 or mode 2 of input-output mode.
- It has two control groups, control group A and control group B.
- Control group A consist of port A and port C upper.
- Control group B consists of port C lower and port B.

Depending upon the value if CS', A1 and A0 we can select different ports in different modes as input-output function or BSR.
This is done by writing a suitable word in control register (control word D0-D7).

CS'	A1	A0	Selection	Address
0	0	0	PORT A	80 H
0	0	1	PORT B	81 H
0	1	0	PORT C	82 H
0	1	1	Control Register	83 H
1	X	X	No Selection	X

Pin diagram –

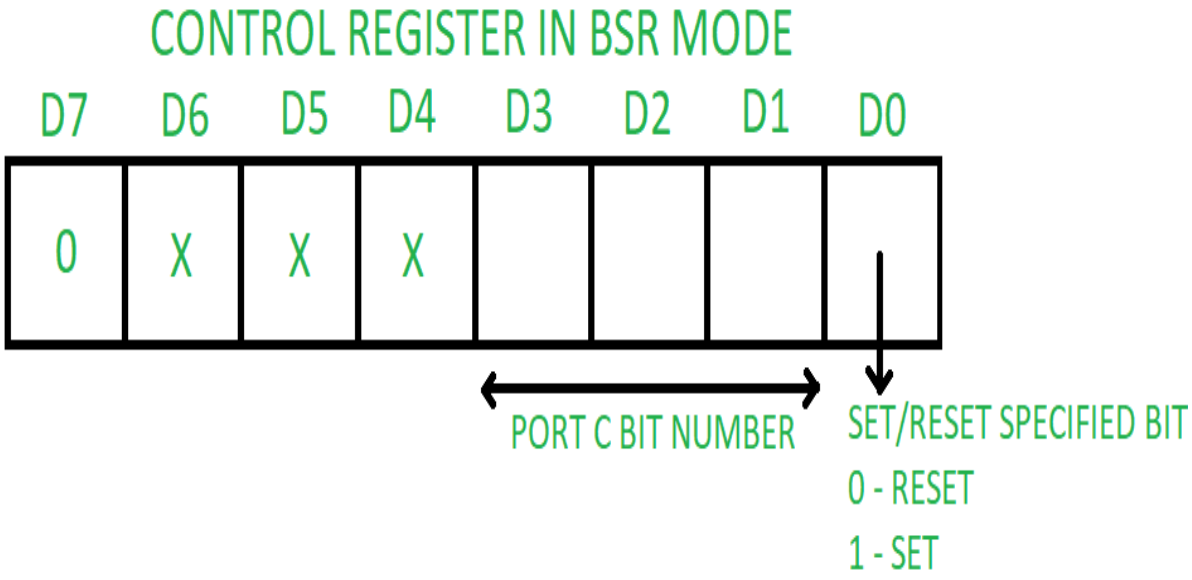


- **PA0 – PA7 –** Pins of port A
- **PB0 – PB7 –** Pins of port B
- **PC0 – PC7 –** Pins of port C
- **D0 – D7 –** Data pins for the transfer of data
- **RESET –** Reset input
- **RD' –** Read input
- **WR' –** Write input
- **CS' –** Chip select
- **A1 and A0 –** Address pins

Operating modes –

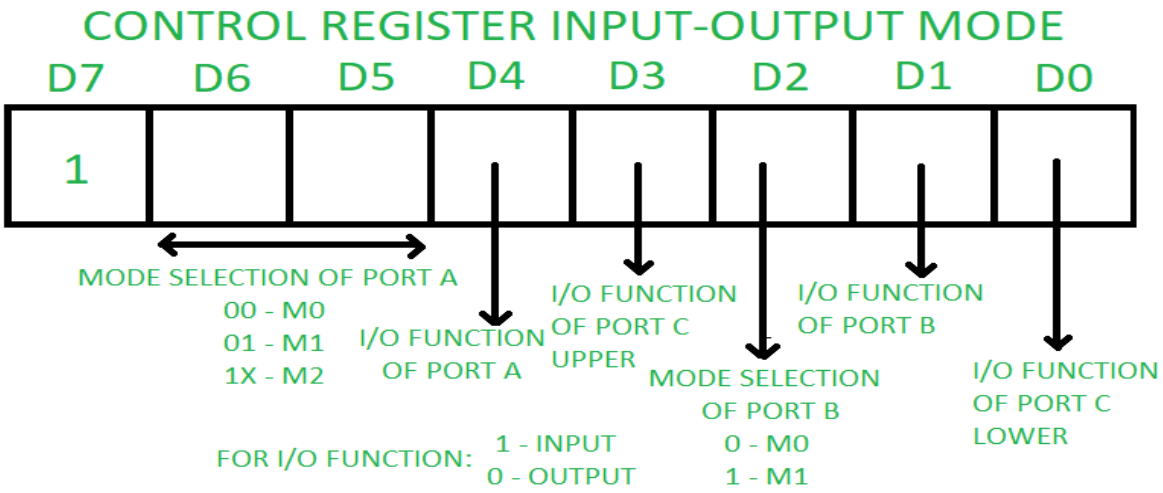
1-Bit set reset (BSR) mode

If MSB of control word (D7) is 0, PPI works in BSR mode. In this mode only port C bits are used for set or reset.



2-Input-Output mode

- If MSB of control word (D7) is 1, PPI works in input-output mode.
- This is further divided into three modes



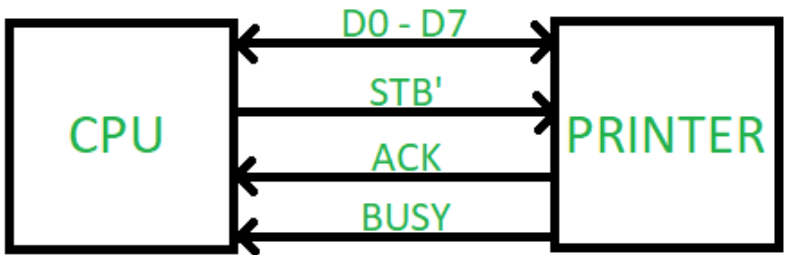
Mode 0 –

- In this mode all the three ports (port A, B, C) can work as simple input function or simple output function.
- In this mode there is no interrupt handling capacity.

Mode 1 –

- Handshake I/O mode or strobed I/O mode.
- In this mode either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission.
- It has interrupt handling capacity and input and output are latched.

Example: A CPU wants to transfer data to a printer. In this case since speed of processor is very fast as compared to relatively slow printer, so before actual data transfer it will send handshake signals to the printer for synchronization of the speed of the CPU and the peripherals.



Mode 2 –

- Bi-directional data bus mode.
- In this mode only port A works, and port B can work either in mode 0 or mode 1.
- 6 bits port C are used as handshake signals.
- It also has interrupt handling capacity.

4.6- ADC And DAC With Interfacing

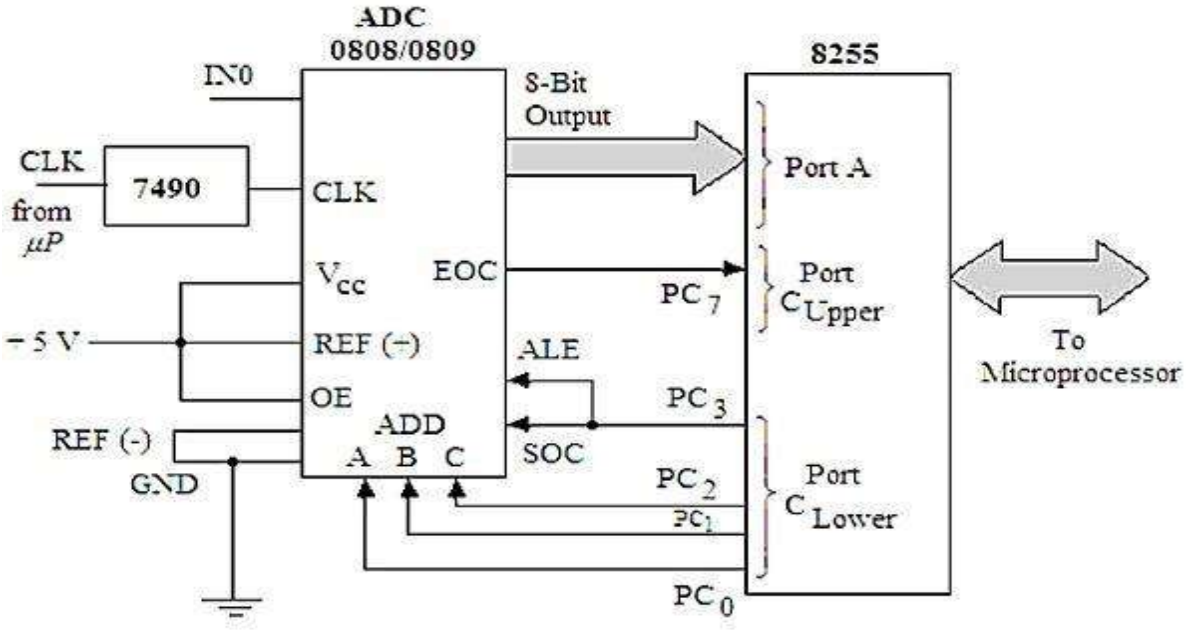
ADC and DAC Interfacing

- The Analog to Digital Conversion is a quantizing process.
- Here the analog signal is represented by equivalent binary states.
- The A/D converters can be classified into two groups based on their conversion techniques.
- In the first technique it compares given analog signal with the initially generated equivalent signal..
- In another technique it determines the changing of analog signals into time or frequency.
- This process includes integrator-converters and voltage-to frequency converters.
- The first process is faster but less accurate, the second one is more accurate.
- As the first process uses flash type, so it is expensive and difficult to design for high accuracy.
- Good features of ADC 0808/0809:
  - The conversion speed is much higher
  - The accuracy is also high
  - It has minimal temperature dependence
  - Excellent long term accuracy and repeatability
  - Less power consumption

Interfacing ADC with 8085 Microprocessor

- To interface the ADC with 8085, we need 8255 Programmable Peripheral Interface chip with it.

- The circuit diagram of connecting 8085, 8255 and the ADC converter.

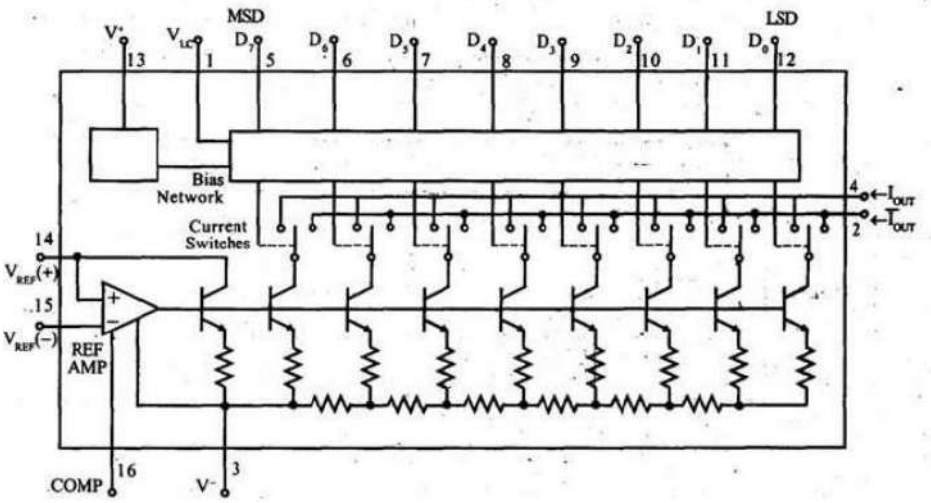


- The Port A of 8255 chip is used as the input port.
- The PC7 pin of Port Cupper is connected to the End of Conversion (EOC) Pin of the analog to digital converter. This port is also used as input port.
- The C lower port is used as output port. The PC2-0 lines are connected to three address pins of this chip to select input channels.
- The PC3 pin is connected to the Start of Conversion (SOC) pin and ALE pin of ADC 0808/0809

DAC Interfacing with 8085 Microprocessor

DAC 0800 Features-

- To convert the digital signal to analog signal a Digital-to-Analog Converter (DAC)has to be employed.
-
- The DAC will accept a digital (binary) input and convert to analog voltage orcurrent



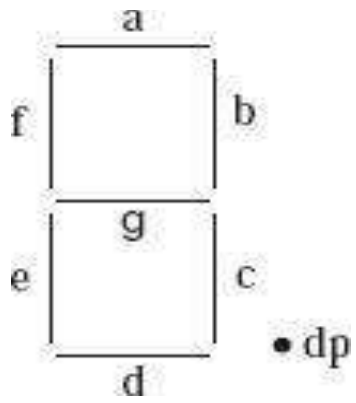
- Every DAC will have "n" input lines and an analog output.
- The DAC require a reference analog voltage (V ref) or current (I ref) source.
- The smallest possible analog value that can be represented by the n-bit binary codeis called resolution
- The DAC0800 require a positive and a negative supply voltage in the range of ±5V to ±18V.
- It can be directly interfaced with TTL, CMOS, PMOS and other logic families.
- For TTL input, the threshold pin should be tied to ground (VLC = 0V).
- The reference voltage and the digital input will decide the analog output current, which can be converted to a voltage by simply connecting a resistor to output terminal or by using an op-amp I to V converter.

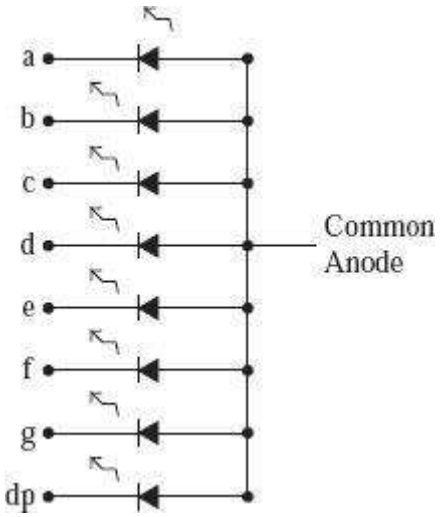
- The DAC0800 is available as a 16-pin IC in DIP.

4.7-Interfacing Seven Segment Displays

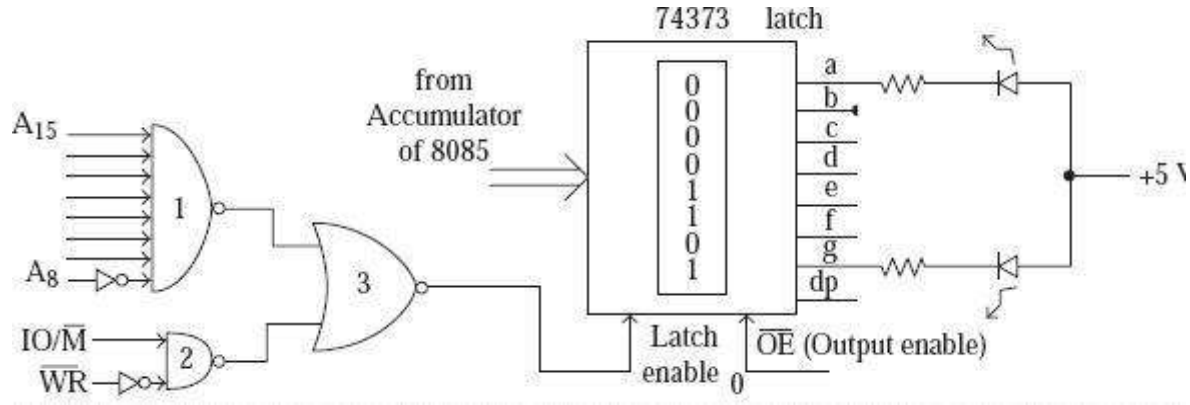
Seven segment Display

- It is an output device which is very commonly used in the kit of 8085 microprocessors.
- It is the Light Emitting Diode consisting of seven segments.
- We denote the segments as 'a, b, c, d, e, f, g, and dp' where dp signifies '.' which is the decimal point.
- A 7-segment display is as shown in the following Figure.





- There are two types of 7-segment LED:
- First is the common anode type and 2nd is the common cathode type.
- In the LED which is common anode and is 7-segmented, here we connect all the eight LED anodes together and the eight external pin is brought to display. And this pin gets connected to a DC supply of +5 Volt.
- The cathode ends of the eight segments are brought out on the pins of the display.
- The use of 74373 latch for interfacing a 7-segment display is shown in the following Figure.



- In the 74373 latch is used as an I/O mapped I/O port with the port address as FEH.
- This could be easily verified from the chip select circuit used in the figure.
- The following instructions are to be executed to display character '3' on the 7-segment display.
- The corresponding program to send 0DH to the port FEH will be -

MVI A, 0DH
OUT FEH

- Using MVI instruction we are initializing Accumulator (A) with Byte 0DH i.e 0000 1101. Then it will be sent to the port FEH by the instruction OUT.

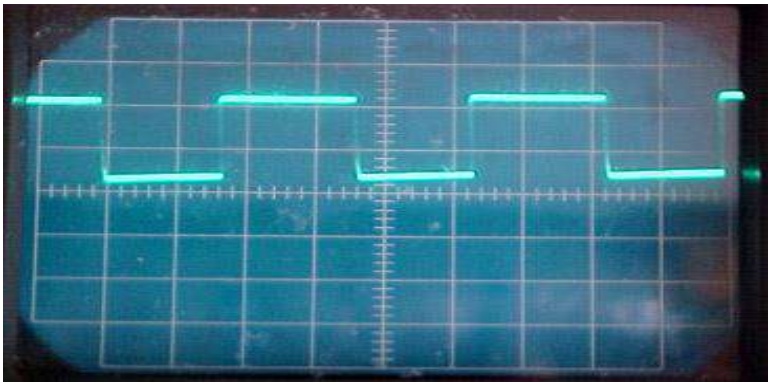
4.8-Generate Square Waves On All Lines Of 8255

- The following is the assembly language using DAC to interface with 8255 and generate a square wave on CRO.
- Here in the code, we use two delay elements one for the rising part of the wave and the other delay element to reach zero i.e. decrement.
- Certain value chosen is delayed or sustained for a time period to form the square wave. The two loops used in the program to repeat cycles of a square wave.

Code:

```
MOV DX,8807 : DX is loaded with control word register address of 8255
MOV AL,80
OUT DX,AL : Contents of AL are transferred to port A of 8255
MOV DX,8801 : DX is loaded with Port A address of 8255
Begin MOV AL,00
OUT DX,AL ; Contents of AL are transferred to port  A of 8255
MOV CX,00FF
Delay1 Loop Delay1
MOV AL,FF
OUT DX,AL : Contents of AL are transferred to port A of 8255
MOV CX,00FF : CX is loaded with 00FFH
Delay2 Loop Delay2 : Repeat until CX=0
JMP Begin ; Repeat the same
```

- Thus we programed in assembly language to interface DAC using 8255 to generate a square waveform



4.9-Design Interface A Traffic Light Control System Using 8255.

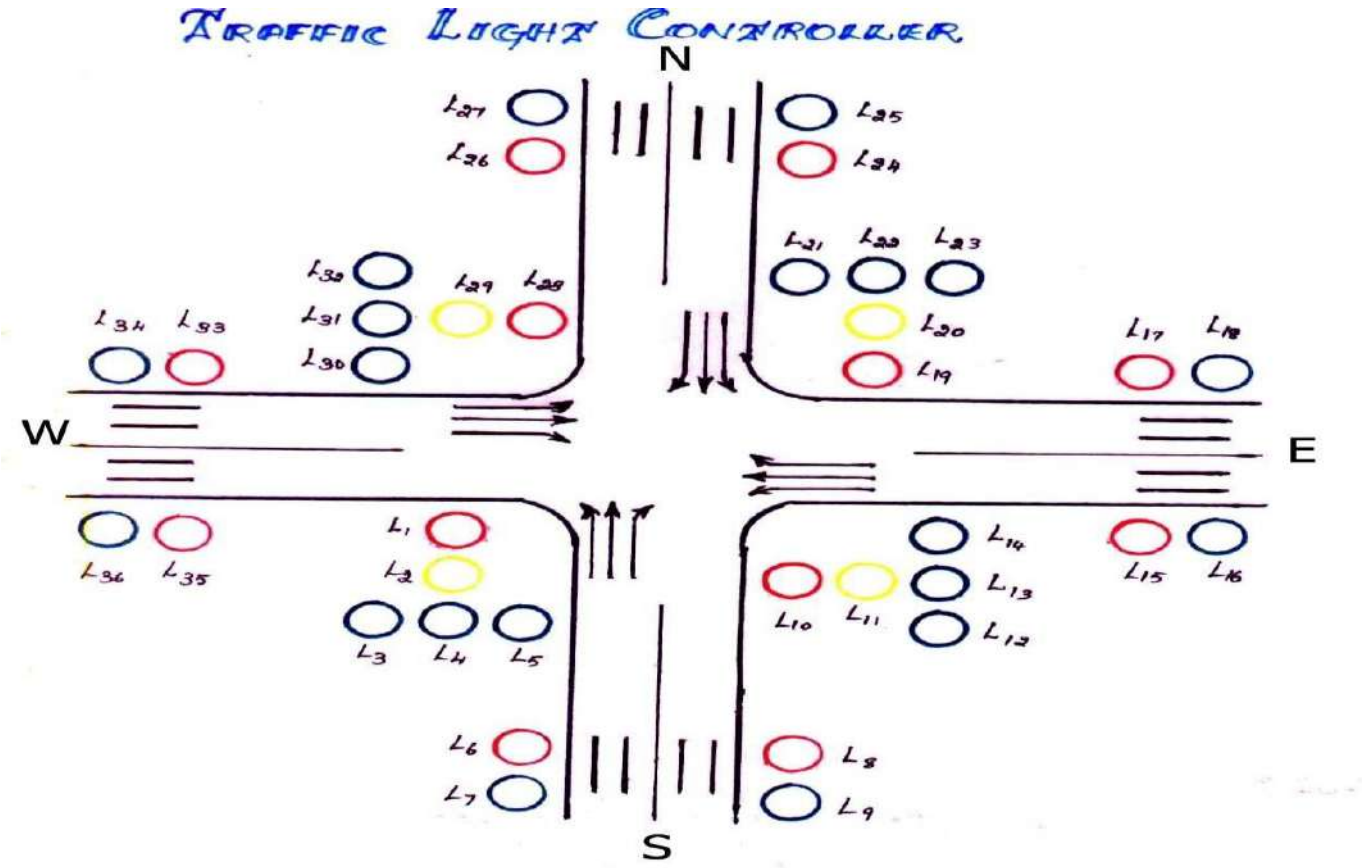
DESCRIPTION

- Combination of Red, Amber and Green LEDs are provided to indicate Halt, Wait and Go states for vehicles.
- Combination of Red and Green LEDs are provided for pedestrian crossing. 36 LEDs are arranged in the form of an intersection.
- At the left corner of each road, a group of 5 LEDs (Red, Amber and Green) are arranged in the form of a T section to control the traffic of that road.
- Each road is named as North N, South S, East E and West W.
- L1, L10, L19 and L28 (Red) are for stop signal for the vehicles on the road N, S, W and E respectively.
- L2, L11, L20 and L29 (Amber) indicate wait state for the vehicles on the road N, S, E and W respectively.
- L3, L4 and L5 (Green) are for left, straight and right turn for the vehicles on the road S.
- Similarly L12 - L13 - L14, L23 - L22 - L21 and L32 - L31 - L30 simulates same function for the roads E, N & W respectively.
- A total of 16 LEDs (2 Red & 2 Green at each road) are provided for pedestrian crossing. L7 - L9, L16 - L18, L25 - L27 & L34 - L36 (Green) when on allows pedestrians to cross and L6 - L8, L15 - L17, L24 - L26 & L33 - L35 (Red) when on alarms the pedestrians to wait.

ADDRESS	LABEL	MNEMONICS	OPCODE/OPERAND
C000		**MVI A,80H**	3E 80
C002		**OUT CWR**	D3 DB
C004	**REPEAT**	**MVI E,03H**	06 03
C006		**LXI H,C100H**	21 00 C1
C009	**NEXTSTAT**	**MOV A,M**	7E
C00A		**OUT PORTA**	D3 D8
C00C		**INX H**	23
C00D		**MOV A,M**	7E
C00E		**OUT PORTB**	D3 D9
C010		**INX H**	23
C011		**MOV A,M**	7E
C012		**OUT PORTC**	D3 DA
C014		**CALL DELAY**	CD 1F C0
C017		**INX H**	23
C018		**DCR E**	05
C019		**JNZ NEXTSTAT**	C2 09 C0
C01C		**JMP REPEAT**	C3 04 C0

C01F	**DELAY**	**LXI D,3000H**	11 00 30
C022	**L2**	**MVI C,FFH**	0E FF
C024	**L1**	**DCR C**	0D
C025		**JNZ L1**	C2 24 C0
C028		**DCX D**	1B
C029		**MOV A,D**	7A
C02A		**ORA E**	B3
C02B		**JNZ L2**	C2 22 C0
C02E		**RET**	C9

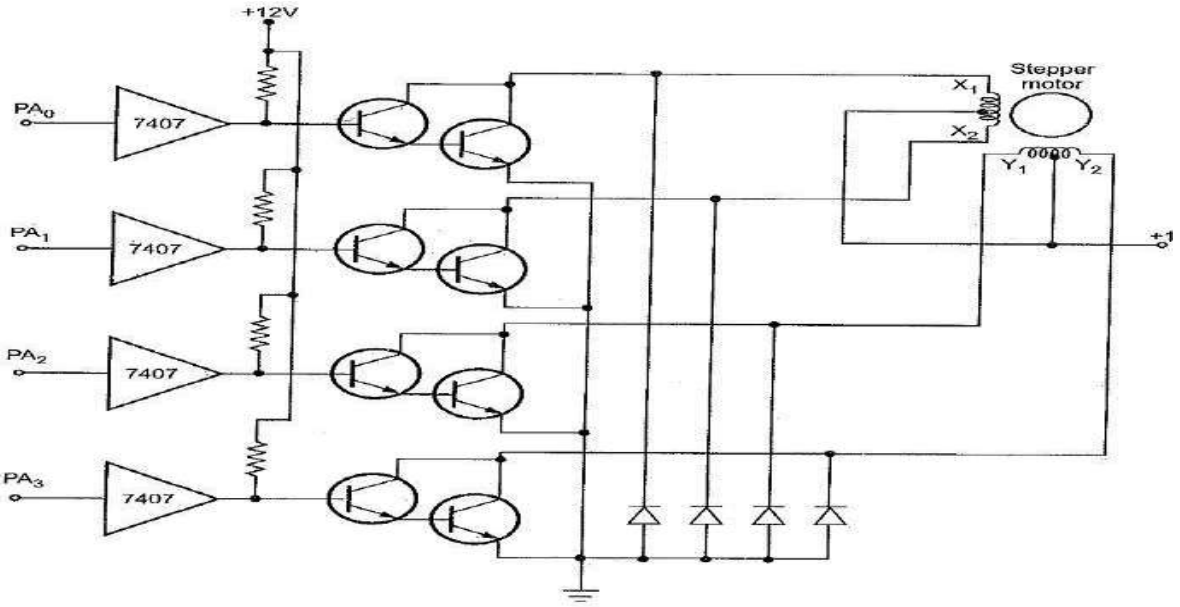
Traffic controller with Four way



4.10-Design Interface For Stepper Motor Control Using 8255

Stepper Motor Interface:

- A stepper motor is a digital motor. It can be driven by digital signal.
- Figure shows the typical 2 phase Stepper Motor Interface using 8255. Motor shown in the circuit has two phases, with center-tap winding.
- The center taps of these windings are connected to the 12V supply.
- Due to this, motor can be excited by grounding four terminals of the two windings.
- Motor can be rotated in steps by giving proper excitation sequence to these windings.
- The lower nibble of port A of the 8255 is used to generate excitation signals inthe proper sequence.



- The given excitation sequence rotates the motor in clockwise direction. To rotate motor in anticlockwise direction we have to excite motor in a reverse sequence.
- The excitation sequence for Stepper Motor Interface many change due to change in winding connections.

Step	X_1	X_2	Y_1	Y_2
1	0	1	0	1
2	1	0	0	1
3	1	0	1	0
4	0	1	1	0
1	0	1	0	1

- The excitation sequence given in Table is called full step sequence. In which excitation ends of the phase are changed in one step.
- The excitation sequence given in table takes two steps to change theexcitation ends of the phase. Such a sequence is called half step sequence and in each step the motor is rotated by 0.9°.

POSSIBLE SHORT TYPE QUESTIONS WITH ANSWERS

1-Define interfacing.

Ans.-In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these.

2-Define memory mapping and I/O mapping. (W-23)

Ans.-Memory-mapped I/O (MMIO) and I/O mapped I/O (PMIO) are two complementary methods of performing input/output (I/O) between the central processing unit (CPU) and peripheral devices in a computer. An alternative approach is using dedicatedI/O processors, commonly known as channels on mainframe computers, which execute their own instructions.

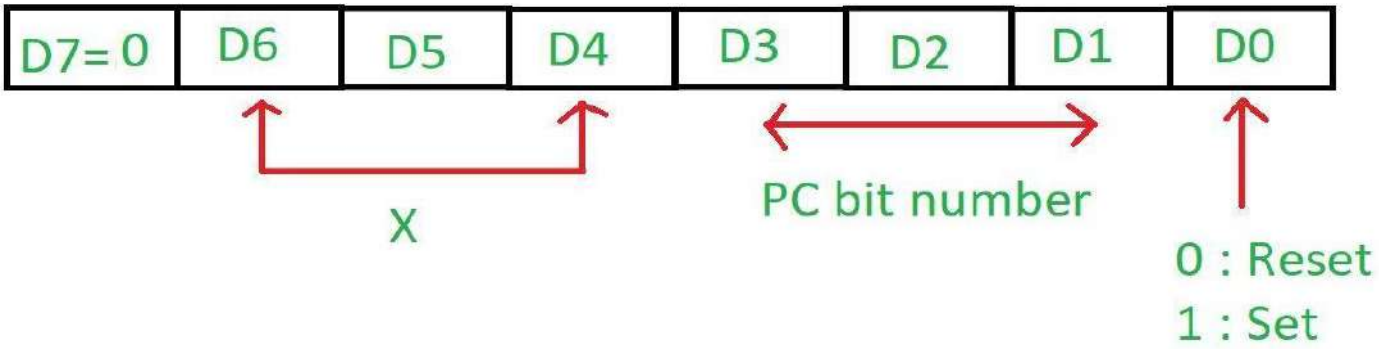
3-Define EPROM

- **EPROM**, in full erasable programmable read-only memory, Form of computer memory that **does** not lose itscontent when the power supply is cut off and that **can** be erased and reused State different modes of 8255

4-State different modes of 8255. (19-W)

- There are 2 modes in 8255 microprocessor:

Bit set reset (BSR) mode – This mode is used to set or reset the bits of port C only, and selectedwhen the most significant bit (D7) in the control register is 0. Control Register is as follows:



Input/output mode (I/O) – This mode is selected when the most significant bit (D7) in the controlregister is 1.

Mode 0 – Simple or basic I/O mode:

Port A, B and C can work either as input function or as output function. The outputs are latchedbut the inputs are not latched. It has interrupt handling capability.

Mode 1 – Handshake or strobed I/O:

In this either port A or B can work and port C bits are used to provide handshaking. The outputs as well as inputs are latched. It has interrupt handling capability. Before actual data transfer thereis transmission of signal to match speed of CPU and printer.

Q 5-Define ADC

- In electronics, an analog-to-digital converter (ADC, A/D, or A-to-D) is a system that converts an analog signal, such asa sound picked up by a microphone or light entering a digital camera, into a digital signal.
- An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number representing the magnitude of the voltage or current. Typically, the digital output is a two's complement binary number that is proportional to the input.

Q6-Why interfacing is required in Microprocessor. (S-24)

Ans: - Interfacing is required in microprocessors to enable communication between the microprocessor and external devices.

It allows the microprocessor to interact with devices like memory, sensors, and input/output (I/O) devices.

Ttypes of interfacing

Memory interfacing

I/O interfacing

POSSIBLE LONG TYPE QUESTIONS

1- Draw the block diagram of 8255.
2- Generate square wave using 8255.
3- Design Interface a traffic light control system using 8255.
4- Design interface for stepper motor control using 8255.
5-Draw the functional block diagram of 8255 and explain each block. (S-24)
6-Develop a traffic light controller program with a neat block diagram. (S-24)

CHAPTER 5.0
MICROPROCESSOR (ARCHITECTURE AND PROGRAMMING- 16 BIT-8086)

Learning Objectives:

5.1- Register Organization of 8086.
5.2- Internal architecture of 8086
5.3-Signal Description of 8086
5.4-General Bus Operation& Physical Memory Organization
5.5-Minimum Mode &Timings,
5.6-Maximum Mode &Timings,
5.7-Interrupts and Interrupt Service Routines, Interrupt Cycle, Non-Mask able Interrupt, Mask able Interrupt
5.8-8086 Instruction Set & Programming: Addressing Modes, Instruction Set, Assembler Directives and Operators,
5.9-Simple Assembly language programming using 8086 instructions

5.1-Register Organization Of 8086.

General 16-bit registers

- The registers AX, BX, CX, and DX are the general 16-bit registers.

AX Register*:*

- Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16- bit register AX.
- AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations, rotate and string manipulation.

BX Register*:*

- This register is mainly used as a base register.
- It holds the starting base location of a memory region within a data segment.
- It is used as offset storage for forming physical address in case of certain addressing mode.

CX Register*:*

- It is used as default counter or count register in case of string and loop instructions.

DX Register*:*

- Data register can be used as a port number in I/O operations and implicit operand or destination in case of few instructions.
- In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

Segment registers:

- To complete 1Mbyte memory is divided into 16 logical segments.
- The complete 1Mbyte memory segmentation, Each segment contains 64Kbyte of memory.
- There are four segment registers.

1-Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions.

- The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register.
- CS register cannot be changed directly.
- The CS register is automatically updated during far jump, far call and far return instructions.
- It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

2-Stack segment *(SS)* is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment.

- SS register can be changed directly using POP instruction. The stack segment is that segment of memory, which is used to store stack data.

3-Data segment *(DS)* is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment.

- DS register can be changed directly using POP and LDS instructions. It points to the data segment memory where the data is resided.

4-Extra segment *(ES)* is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions.

- ES register can be changed directly using POP and LES instructions. It also refers to segment which essentially is another data segment of the memory. It also contains data.

Pointers and index registers.

The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively

Stack Pointer (SP) is a 16-bit register pointing to program stack in stack segment.

Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.

Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Conditional Flags

Conditional flags are as follows:

1-Carry Flag (CY): This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

2-Auxiliary Flag (AC): If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AC flag is set i.e. carry given by D3 bit to D4 is AC flag.

3-Parity Flag (PF): This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's,
the Parity flag is reset.

4-Zero Flag (ZF): It is set; if the result of arithmetic or logical operation is zero else it is reset.

5-Sign Flag (SF): In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

6-Control Flags: Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

7-Trap Flag (TF): It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

5.2-Internal Architecture Of 8086

Bus Interface Unit

- The Bus Interface Unit of 8086 microprocessor manages all the data and addresses transfers for the Execution Unit (EU) like sending address, fetching an instruction, reading data from the memory/ports as well as writing data to the ports/memory.
- The BIU and EU are connected by an internal bus as shown in the below block diagram .

The BIU of 8086 microprocessor has the following functional parts:

1- Instruction Queue

- The Bus Interface Unit of the 8086 microprocessor contains an instruction queue.

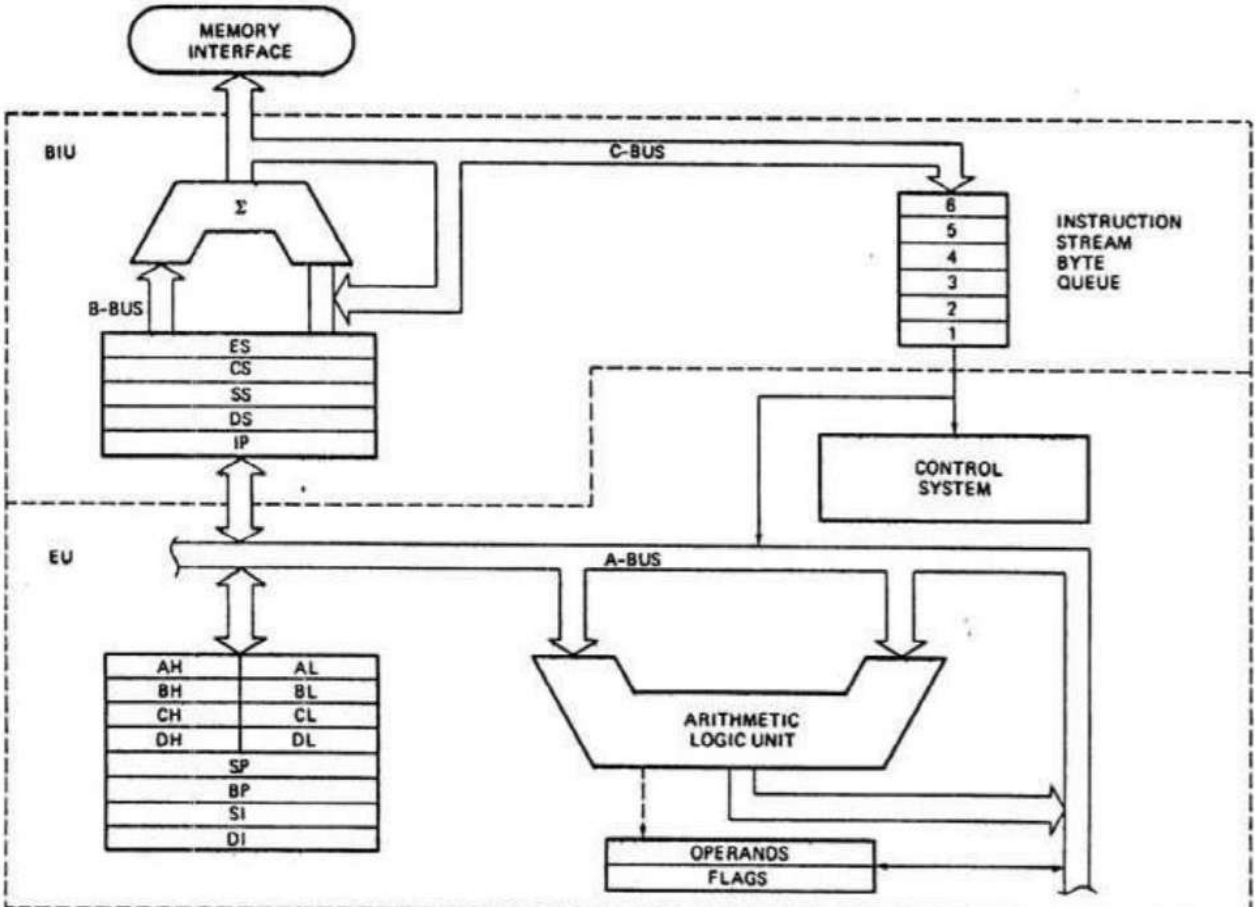
- BIU gets up to six bytes of the next instructions and stores them in the instruction queue.
- When the Execution unit executes an instruction and is ready for the next instruction, it simply reads the instruction from the instruction queue resulting in increased execution speed.
- Pipelining is the process of fetching the next instruction cycle when the current instruction is being executed.

2. Segment Register

- The BIU of 8086 microprocessor has 4 segment buses i.e. CS, DS, SS & ES. Ts holds the address of instructions and data in the memory which are used by the processor to access memory locations. It also contains one pointer register IP.

(i) Code Segment Register (CS)

- The base of the current code segment is contained by the code segment register of the 8086 microprocessor.
- It finds its application in addressing a memory location in the code segment of the memory where the executable program is stored.



ii) Data Segment Register (DS)

- The data segment contains the starting address of the program's data segment. It is accessed in the data segment by an offset address or the content of the other register that holds the offset address.

(iii) Stack Segment Register (SS)

- The stack segment register of the 8086 microprocessor contains the starting address of a program's stack segment. The current word in the stack that is addressed is indicated by the segment address plus an offset value in the stack pointer.

(iv) Extra Segment Register (ES)

- The ES of the 8086 microprocessor is an additional data segment used by some strings to hold extra data. The string instructions always use the ES and destination index (DI) to determine the 20-bit physical address.

(v) Instruction Pointer Register (IP)

- The Instruction Pointer is a 16-bit register. IP is responsible for holding the distance or offset from this address to the next instruction byte to be fetched.

Execution Unit (EU)

- The execution unit of the 8086 gives instructions to the bus interface unit stating from where to fetch the instructions or data from, decode the instructions, and execute those instructions.
- The main function of the EU of the 8086 microprocessor is to control operations on data using an instruction decoder and ALU. It has no direct connection with the system and performs over data through BIU.
- The EU includes an arithmetic and logic unit (ALU), a control unit, and a set of registers.

1. ALU

- The arithmetic and logic unit performs all the arithmetic and logical operations like add, subtract, OR, AND, NOT in the 8086 microprocessor.

2. General-Purpose Registers

- The 8086 microprocessor has 8 registers AH, AL, BH, BL, CH, CL, DH, DL. Individually, these registers can store 8-bit data, and in pairs, they can store 16-bit data.
- The valid register pairs are AHAL, BHBL, CHCL, DHDL. They can be represented as AX, BX, CX, and DX.

a. AX Register

- The AX register is called a 16-bit accumulator and AL is called the 8-bit accumulator. For inputting/outputting data from or to I/O ports, the I/O (IN or OUT) instructions often use AX or AL.

b. BX Register

- The BX register of the 8086 microprocessor is known as the base register. It is used to store the starting base address of the memory field within the data section.
- It is also used as an index to extend addressing. BX can also be used as a foundation register in conjunction with DI or SI for special addressing.

c. CX Register

- The contents of the CX register of the 8086 microprocessor are used as a counter in certain instructions such as SHIFT, ROTATE, and LOOP.

d. DX Register

- The DX register of the 8086 microprocessor is known as the data register. It is used to hold the I/O port address during the I/O instructions.

3. Stack Pointer Register (SP) and Base Pointer Register (BP)

- Both the SP and BP registers in the 8086 microprocessor are used to access data in the stack segment. During the execution of instructions, the SP is used as an offset from the current stack section.
- The offset address in the current stack section is stored in the BP. Instructions that use the based addressing mode make use of this offset.

4. Index Register

- The two index registers SI (source index) and DI (destination index) o the 8086 microprocessor are used in indexed addressing.
- The instructions that process data strings use the SI and DI index registers, along with DS and ES, to differentiate between the source and destination address.

5. Flag Register

- The flag register of the 8086 microprocessor is also known as the status register.
- It is a 16-bit register that can be used as a flipflop as its status changes according to the result.
- The control bit in the flag register can be set or reset by the programmer. It has 9 flags and they are divided into two groups: Conditional flags and Control flags.

X	X	X	X	O	D	I	T	S	Z	X	A	X	P	X	C
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

5.3-Signal Description Of 8086

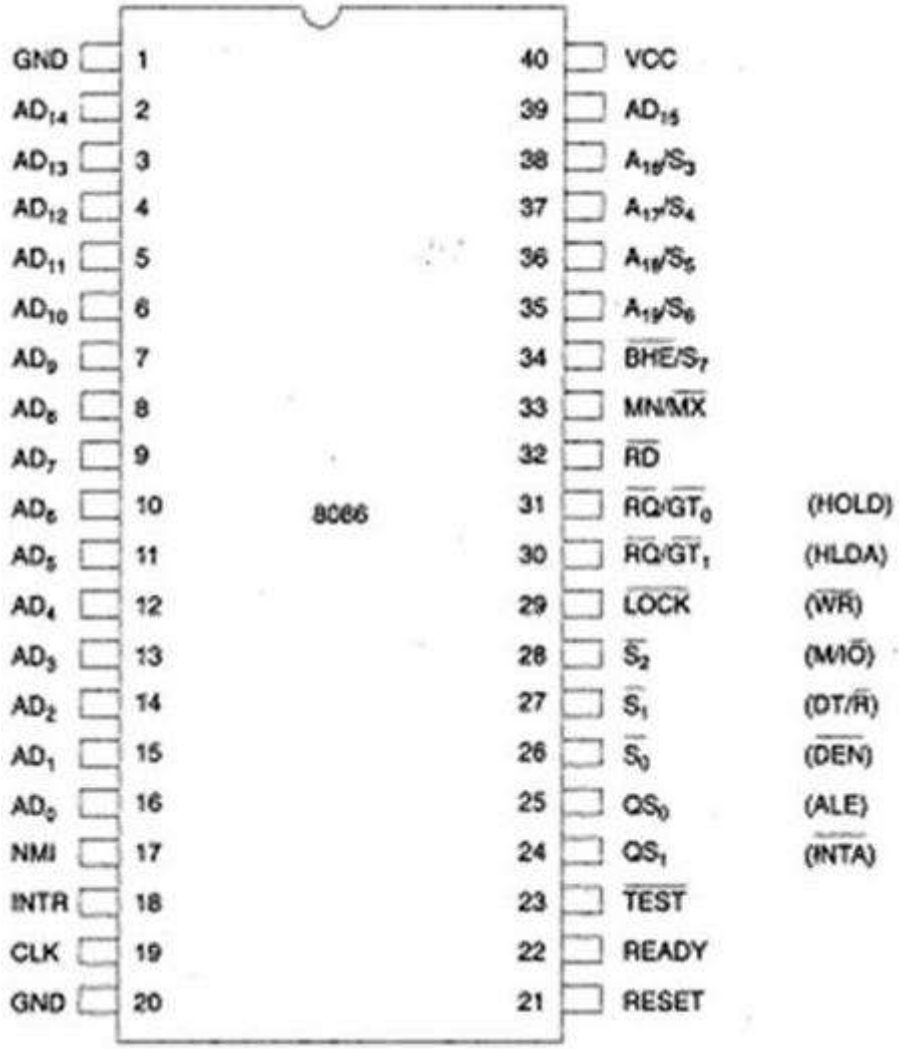
8086 Pin Diagram

Here is the pin diagram of 8086 microprocessors –

- Power supply and frequency signals
- It uses 5V DC supply at V_{CC} pin 40, and uses ground at V_{SS} pin 1 and 20 for its operation.

Clock signal

- Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.



Address/data bus

- AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

- A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

S7/BHE

- BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Ready

- It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

- It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

- It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

- It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

INTA

- It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

- It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

- It stands for Data Enable and is available at pin 26. It is used to enable Trans receiver 8286. The trans receiver is a device used to separate data from the address/data bus.

DT/R

- It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the trans receiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

- This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicating the memory operation. It is available at pin 28.

WR

- It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

- It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

- This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS1 and QS0

- These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table –

QS0	QS1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S0, S1, S2

- These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status −

S2	S1	S0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK

- When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

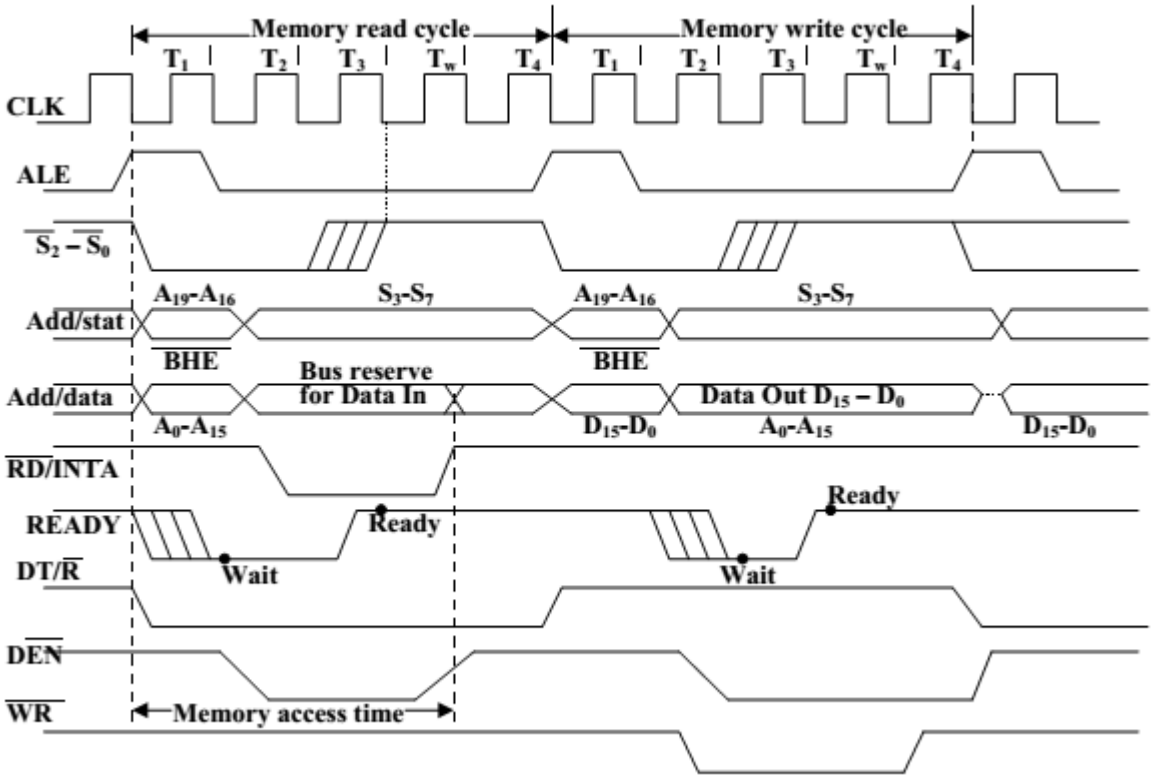
RQ/GT1 and RQ/GT0

- These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT0 has a higher priority than

5.4-General Bus Operation& Physical Memory Organization

Bus operation of 8086 MP

- The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus.
- The main reason behind multiplexing address and data over the same pins is the maximum utilisation of processor pins and it facilitates the use of 40 pin standard DIP package.
- The bus can be DE multiplexed using a few latches and trans receivers, whenever required.
- Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as T1, T2, T3, T4. The address is transmitted by the processor during T1. It is present on the bus only for one cycle.
- The negative edge of this ALE pulse is used to separate the address and the data or status information. In maximum mode, the status lines S0, S1 and S2 are used to indicate the type of operation.
- Status bits S3 to S7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T1 while status bits S3 to S7 are valid during T2 through T4.



General Bus Operation Cycle in Maximum Mode

Maximum mode

- In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information.
- In the maximum mode, there may be more than one microprocessor in the system configuration.

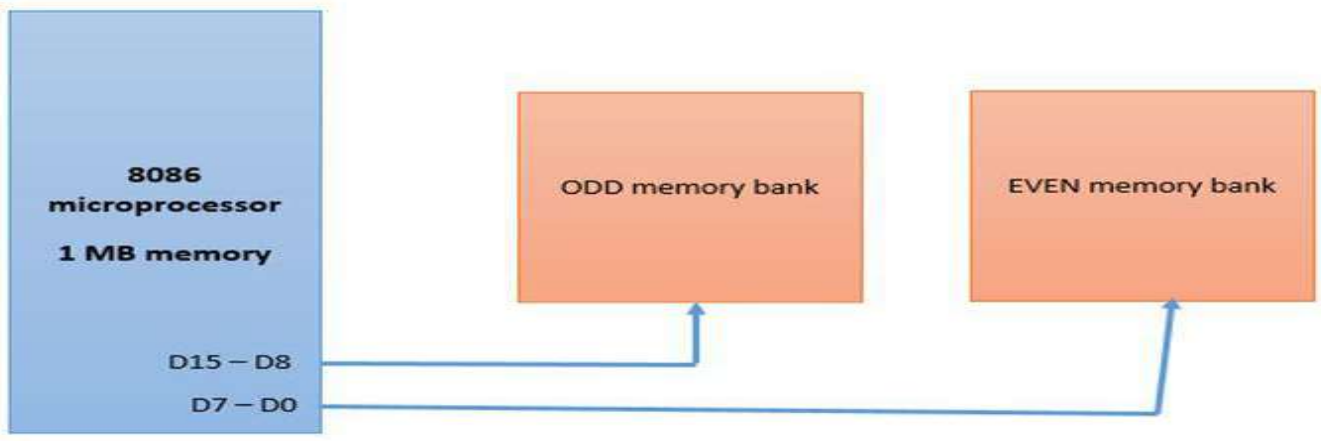
Minimum mode

- In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.
- In this mode, all the control signals are given out by the microprocessor chip itself.
- There is a single microprocessor in the minimum mode system.

Physical Memory organisation of 8086 MP

- There are 20 address lines in the 8086 microprocessor.
- This gives us 220 different memory locations. Hence the total size is 220 Bytes (as each memory location is Byte Addressable, i.e. one byte of data can be stored at every single location), which is equal to 1MB.
- Even the memory is byte-addressable, yet the 8086 microprocessor an easily handle up to 16 bits of data at a time through its 16 data lines.
- So the entire memory in 8086 is divided into two memory banks: odd bank and the even bank.

- The way in which data is read or written is decided by the value of BHE, and the last address bit, that is the A0 line. It is done in the following way:



BHE'	A0	Operation performed on memory
0	0	16 bits of data will be read or written into the memory
0	1	8 bits of data will be read/written into the odd memory bank
1	0	8 bits of data will be read/written into the even memory bank
1	1	No operation is performed

- If the lower byte of the word is stored at even memory bank and the upper byte is stored at odd memory bank then the CPU will require only 1 memory cycle.
- If the lower byte of the word is located at an odd memory address, then the CPU will require 2 memory cycles.

- The first memory cycle is required for accessing the lower byte of the word through the higher data bus, i.e. D15 to D8, and the second memory cycle is required for accessing the upper byte of the word through the lower data bus, i.e. D7 to D0.

5.5-Minimum Mode &Timings.

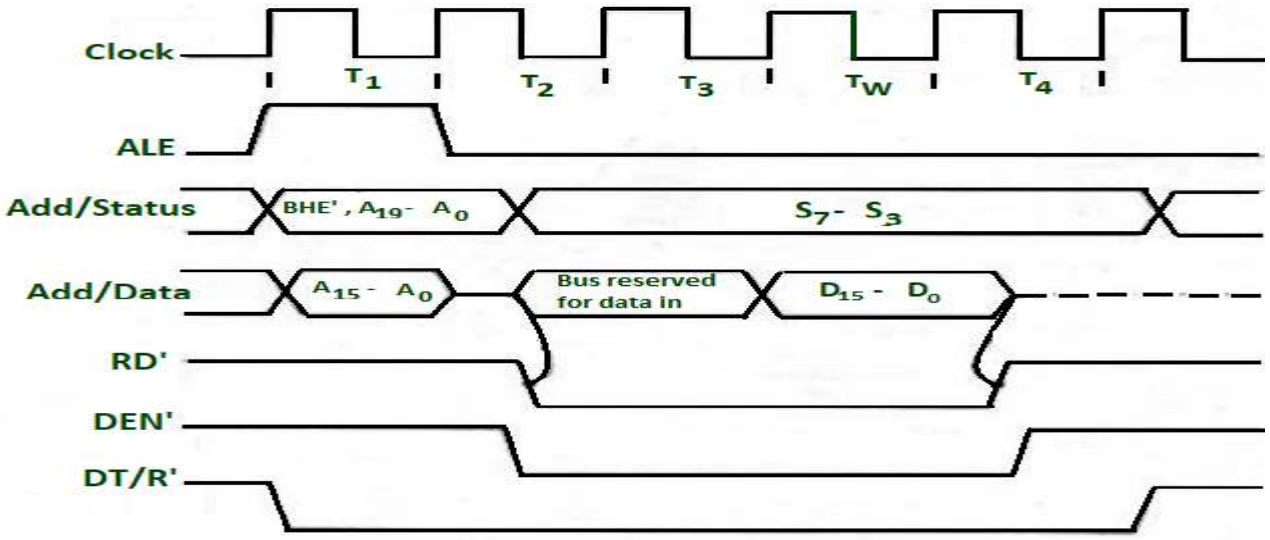
Minimum mode

- In a minimum mode the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1.
- In this mode, all the control signals are given out by the microprocessor chip itself.
- The opcode fetch and read cycles are similar.
- Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.

Read cycle timing diagram for Minimum mode.

The working of min mode can be easily understood by timing diagrams.

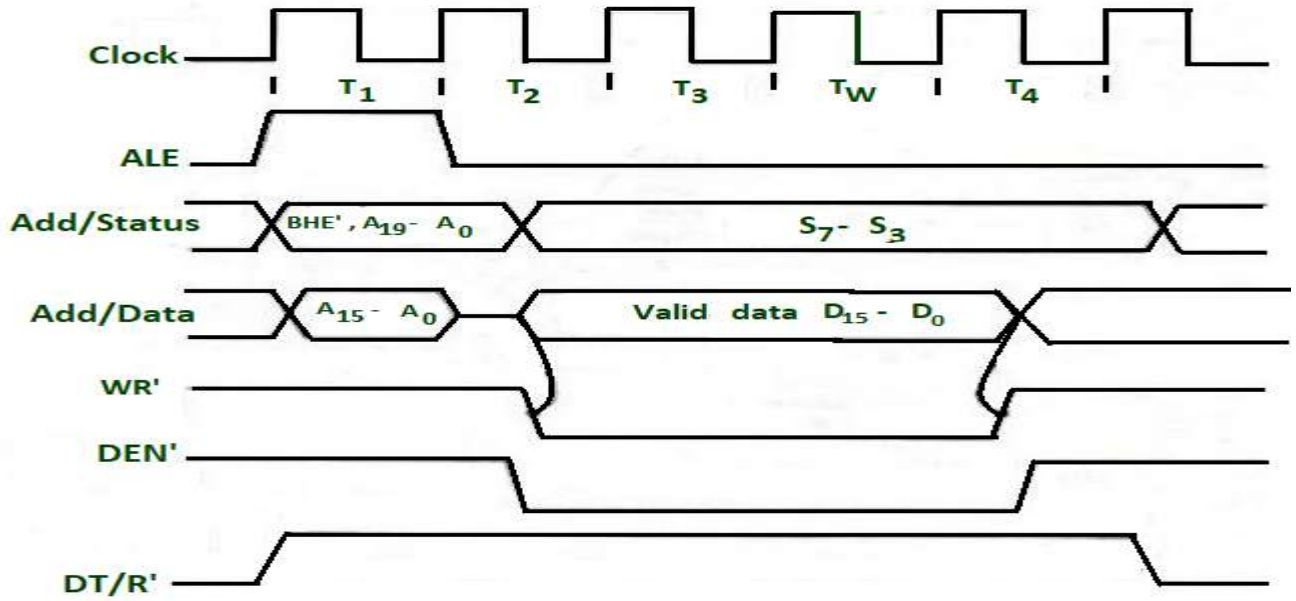
- All processors bus cycle is of at least 4 T-states (T_1, T2, T3, T4) .The address is given by processor in the T1 state. It is available on the bus for **one T-state.**
- In T_2, the bus is tristated for changing the direction of the bus (in the case of a data read cycle.)
- The data transfer takes place between T_3 and T_4.
- If the addressed device is slower, then the wait state is inserted between T_3 and T_4.



Opcode fetch or read timing diagram

- At T_1 state ALE =1, this indicates that a valid address is latched on the address bus and also M / IO'= 1, which indicates the memory operation is in progress.
- In T2, the address is removed from the local bus and is sent to the addressed device. Then the bus is tristated.
- When RD' = 0, the valid data is present on the data bus.
- During T_2 DEN' =0, which enables transceivers and DT/R' = 0, which indicates that the data is received.
- During T_3, data is put on the data bus and the processor reads it.
- The output device makes the READY line high.
- This means the output device has performed the data transfer process. When the processor makes the read signal to 1, then the output device will again tristate its bus drivers.

Write cycle timing diagram for Minimum mode.



- At T_1 state ALE =1, this indicates that a valid address is latched on the address bus and also M / IO'= 1, which indicates the memory operation is in progress.
- In T_2, the processor sends the data to be written to the addressed location.
- The data is buffered on the bus until the middle of T_4 state.
- The WR'=0 becomes at the beginning of T_2.
- The BHE' and A0 signals are used to select the byte or bytes of memory or I/O word.
- During T_2 DEN' =0, which enables, transceivers and DT/R' = 1, which indicates that the data is transferred by the processor to the addressed device.

5.6-Maximum Mode &Timings

Maximum mode of 8086 Microprocessor

- A processor is in the Maximum Mode Configuration of 8086 when its MN/MX pin is grounded. The maximum mode defines pins 24 to 31 as follows.

1. **QS_1, QS_0 (output):** These two output signals reflect the status of the instruction queue. This status indicates the activity in the queue during the previous clock cycle.

QS_1	QS_0	Status
0	0	No operation (queue is idle)
0	1	First byte of an opcode
1	0	Queue is empty
1	1	Subsequent byte of an opcode

2. **S_2, S1, S0 (output):** These three status signals indicate the type of transfer to be take place during the current bus cycle.

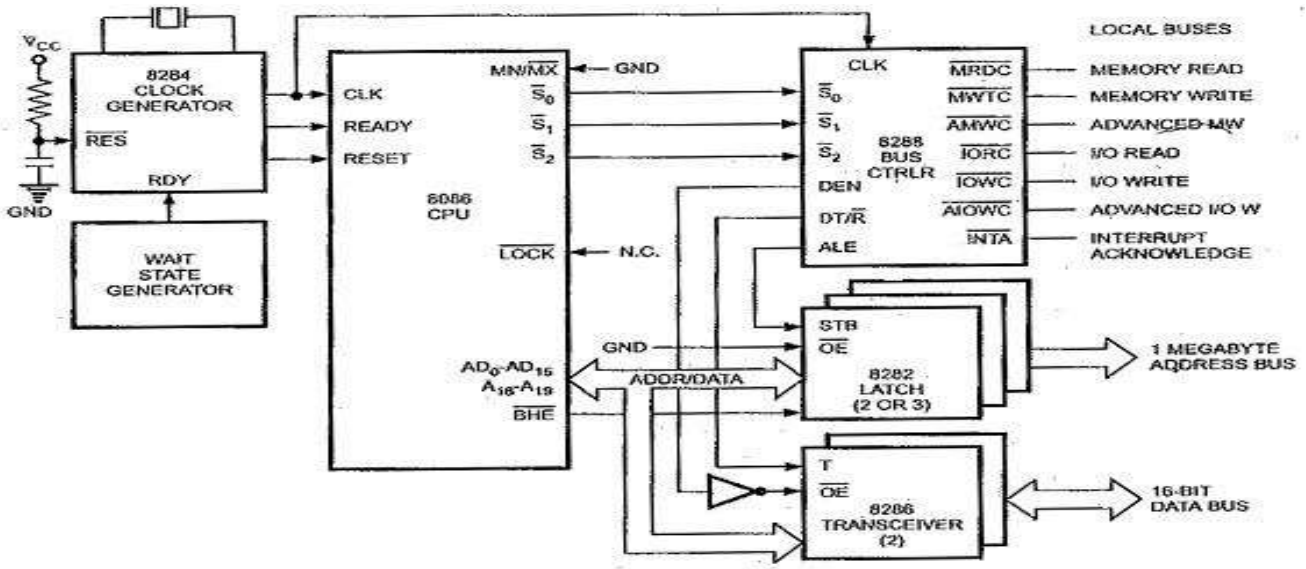
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Machine cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Machine cycle
1	0	0	Instruction fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Inactive-Passive

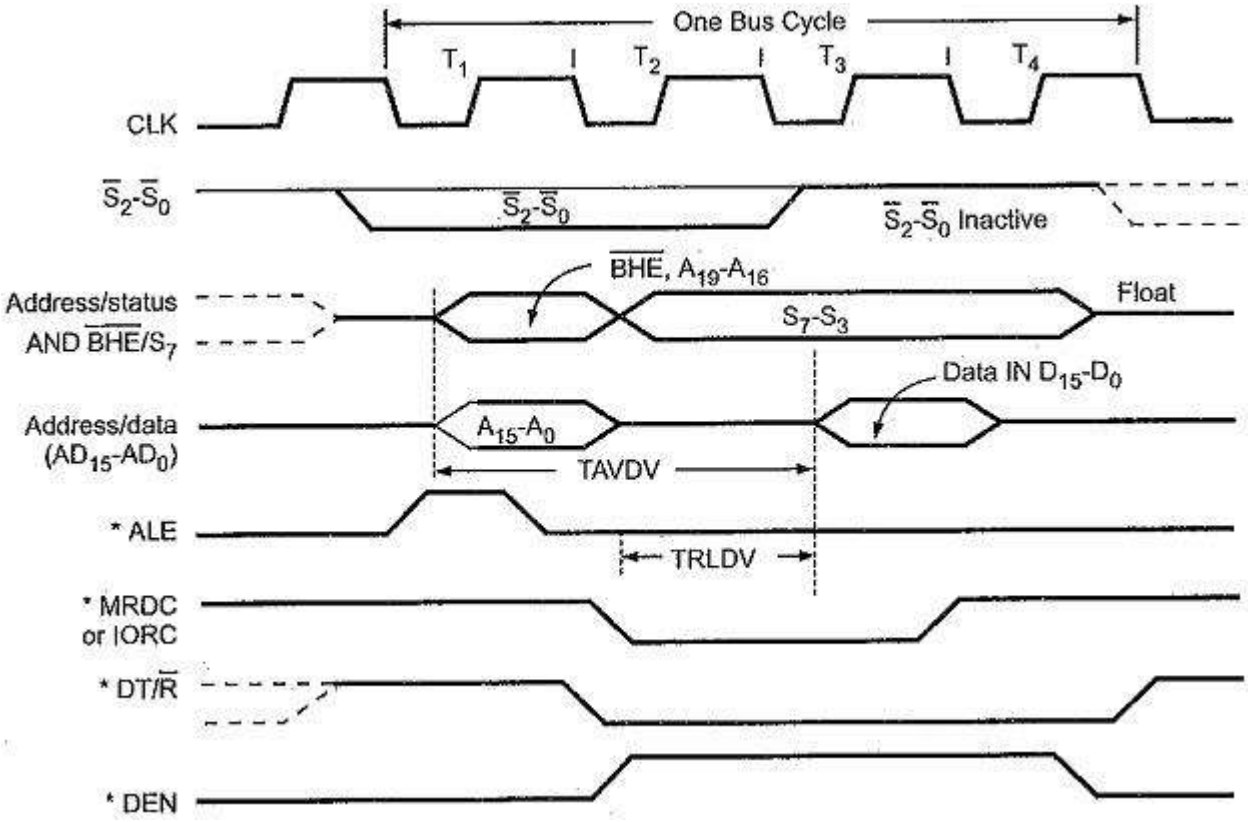
3. LOCK: This signal indicates that an instruction with a LOCK prefix is being executed and the bus is not to be used by another processor:

4. RQ/GT_1 and RQ/GT_0:

- In the Maximum Mode Configuration of 8086, HOLD and HLDA pins are replaced by RQ (Bus request)/GT_0 (Bus Grant), and RQ/GT_1 signals.
- By using bus request signal another master, can request for the system bus and processor communicate. That the request is granted to the requesting master by using bus grantnal. Both signals are similar except the RQ/GT_0 has higher priority than RQ/GT_1.
- In the maximum mode additional circuitry is required to translate the control signals. The additional circuitry converts the status signals (S_2-S_0) into the I/O and memory transfer signals.
- It also generates the control signals required to direct the data flow and for controlling 8282 latches and 8286 transceivers. The Intel 8288 bus controller is used to implement this control circuitry.



Timing Diagram for Input Read operation of 8086

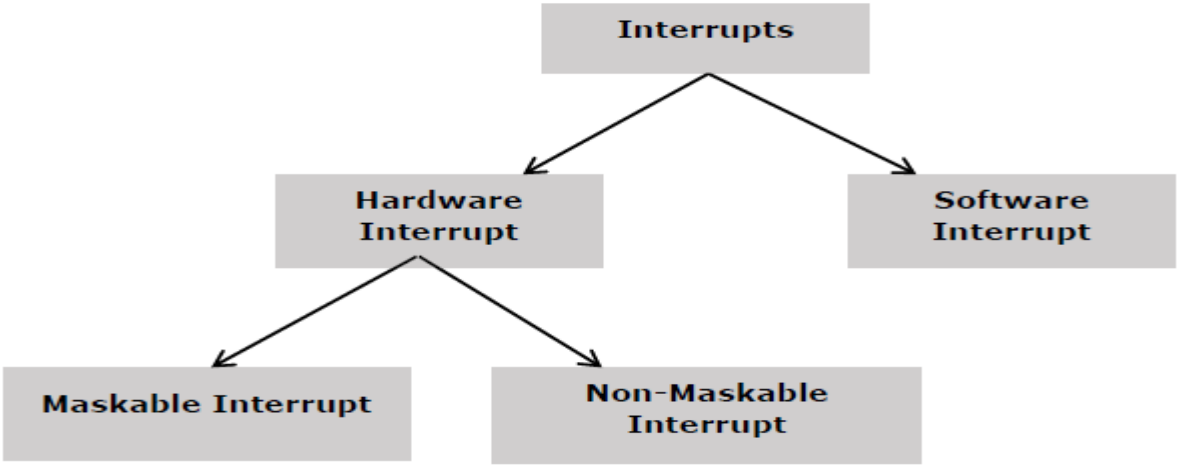


- S_0, S1, S2 are set at the beginning of bus cycle. On detecting the change on passive state $S_0 = S_1 = S_2 = 1$, the 8288 bus controller will output a pulse on its ALE and apply a required signal to its DT/R pin during T_1.
- In T_2, 8288 will set DEN = 1 thus enabling transceiver. For an input, 8288 it will activates MRDC or IORC.
- These signals are activated until T_4. For an output, the AMWC or AIOWC is activated from T_2 to T_4 and MWTC or IOWC is activated from T_3 to T_4.
- The status bits S_0 to S_2 remain active until T_3, and become passive during T_3 and T_4.
- If ready input is not activated before T3, wait state will be inserted between T_3 and T_4.

5.7-Interrupts And Interrupt Service Routines, Interrupt Cycle, Non-Mask Able Interrupt, Mask Able Interrupt

Interrupt

- **Interrupt** is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.
- The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.
- The following image shows the types of interrupts we have in a 8086 microprocessor −



Hardware Interrupts

- Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
- The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-mask able interrupt and INTR is a mask able interrupt having lower priority.
- One more interrupt pin associated is INTA called interrupt acknowledge.

Software Interrupts

- Some instructions are inserted at the desired position into the program to create interrupts.
- These interrupt instructions can be used to test the working of various interrupt handlers.
- 256 interrupts are there

INT n is invoked as software interrupts- n is the type no in the range 0 to 255(00 to FF)

Interrupts are divided into three groups

Type 0 to Type4 (Dedicated Interrupts)

TYPE 0 interrupt represents division by zero situation.

TYPE 1 interrupt represents single-step execution during the debugging of a program.

TYPE 2 interrupt represents non-mask able NMI interrupt.

TYPE 3 interrupt represents break-point interrupt.

TYPE 4 interrupt represents overflow interrupt.

Interrupt Service Routine (ISR)

- Stands for "Interrupt Service Routine." An ISR (also called an interrupt handler) is a software process invoked by an interrupt request from a hardware device.
- It handles the request and sends it to the CPU, interrupting the active process. When the ISR is complete, the process is resumed.
- A basic example of an ISR is a routine that handles keyboard events, such as pressing or releasing a key.
- Each time a key is pressed, the ISR processes the input. For example, if you press and hold the right arrow key in a text file, the ISR will signal to the CPU that the right arrow key is depressed.
- The CPU sends this information to the active word processor or text editing program, which will move the cursor to the right.
- When you let go of the key, the ISR handles the "key up" event. This interrupts the previous "key down" state, which signals to the program to stop moving the cursor.
- The group of memory locations set aside to hold the addresses of ISRs is called the interrupt vector table

NMI (Non mask-able interrupt)-

- This is a non-mask-able, edge triggered, high priority interrupt.
- On receiving an interrupt on NMI line, the microprocessor executes INT.
- Microprocessor obtains the ISR address from location 2 x 4 = 00008H from the IVT (Interrupt vector Table).
- It reads 4 locations starting from this address to get the values for IP and CS to execute the ISR.

MI (Mask-able Interrupt)-

- If the interrupts are disabled using clear interrupt Flag instruction, the microprocessor will not get interrupted even if INTR is activated.
- That is, INTR can be masked.
- INTR is a non-vectored interrupt, which means, the 8086 does not 6 know where to branch to service the interrupt.
- The 8086 has to be told by an external device like a Programmable Interrupt controller regarding the branch.
- Whenever the INTR pin is activated by an I/O port, if Interrupts are enabled and NMI is not active at that time, the microprocessor finishes the current instruction that is being executed and gives out a '0' on INTA pin twice.
- When INTA pin goes low for the first time, it asks the external device to get ready. In response to the second INTA the microprocessor receives the 8 bit.

5.8-8086 Instruction Set & Programming: Addressing Modes, Instruction Set, Assembler Directives And Operators

Addressing Modes of 8086 MP

1-Immediate addressing mode-

- In this mode, the operand is specified in the instruction itself. Instructions are longer but the operands are easily identified.
- Example:
  MOV CL, 12H
- This instruction moves 12 immediately into CL register. CL ← 12H

2-Register addressing mode-

- In this mode, operands are specified using registers. This addressing mode is normally preferred because the instructions are compact and fastest executing of all instruction forms.
- Registers may be used as source operands, destination operands or both.
- Example:
  MOV AX, BX

- This instruction copies the contents of BX register into AX register. AX ← BX

3-Direct memory addressing mode-

- In this mode, address of the operand is directly specified in the instruction. Here only the offset address is specified, the segment being indicated by the instruction.
- Example:
  MOV CL, [4321H]
- This instruction moves data from location 4321H in the data segment into CL.
- The physical address is calculated as
  DS * 10H + 4321
  Assume DS = 5000H
  ∴PA = 50000 + 4321 = 54321H
  ∴CL ← [54321H]

5-Register based indirect addressing mode-

- In this mode, the effective address of the memory may be taken directly from one of the base register or index register specified by instruction. If register is SI, DI and BX then DS is by default segment register.
- If BP is used, then SS is by default segment register.
- Example:
  MOV CX, [BX]
- This instruction moves a word from the address pointed by BX and BX + 1 in data segment into CL and CH respectively.
  CL ← DS: [BX] and CH ← DS: [BX + 1]
  Physical address can be calculated as DS * 10H + BX.

6-Register relative addressing mode-

- In this mode, the operand address is calculated using one of the base registers and an 8 bit or a 16 bit displacement.
- Example:
  MOV CL, [BX + 04H]
- This instruction moves a byte from the address pointed by BX + 4 in data segment to CL.
  CL ← DS: [BX + 04H]
  Physical address can be calculated as DS * 10H + BX + 4H.

7-Base indexed addressing mode-

- Here, operand address is calculated as base register plus an index register.
- Example:
  MOV CL, [BX + SI]
- This instruction moves a byte from the address pointed by BX + SI in data segment to CL.
  CL ← DS: [BX + SI]
  Physical address can be calculated as DS * 10H + BX + SI.

8-Relative based indexed addressing mode-

- In this mode, the address of the operand is calculated as the sum of base register, index register and 8 bit or 16 bit displacement.
- Example:
  MOV CL, [BX + DI + 20]
- This instruction moves a byte from the address pointed by BX + DI + 20H in data segment to CL.
  CL ← DS: [BX + DI + 20H]
  Physical address can be calculated as DS * 10H + BX + DI + 20H.

9-Implied addressing mode-

- In this mode, the operands are implied and are hence not specified in the instruction.
- Example:
  STC
  This sets the carry flag.

Instruction set of 8086 Microprocessor-

The 8086 microprocessor supports 8 types of instructions −

1-Data Transfer Instructions
2-Arithmetic Instructions
3-Bit Manipulation Instructions
4-String Instructions
5-Program Execution Transfer Instructions (Branch & Loop Instructions)
6-Processor Control Instructions
7-Iteration Control Instructions
8-Interrupt Instructions

1-Data Transfer Instructions

- These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

MOV − Used to copy the byte or word from the provided source to the provided destination.
PPUSH − Used to put a word at the top of the stack.
POP − Used to get a word from the top of the stack to the provided location.
PUSHA − Used to put all the registers into the stack.
POPA − Used to get words from the stack to all registers.
XCHG − Used to exchange the data from two locations.
XLAT − Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

IN − Used to read a byte or word from the provided port to the accumulator.
OUT − Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

LEA − Used to load the address of operand into the provided register.
LDS − Used to load DS register and other provided register from the memory
LES − Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

LAHF − Used to load AH with the low byte of the flag register.
SAHF − Used to store AH register to low byte of the flag register.
PUSHF − Used to copy the flag register at the top of the stack.
POPF − Used to copy a word at the top of the stack to the flag register.

2-Arithmetic Instructions

- These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Instructions to perform addition

ADD − Used to add the provided byte to byte/word to word.
ADC − Used to add with carry.
INC − Used to increment the provided byte/word by 1.
AAA − Used to adjust ASCII after addition.
DAA − Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

SUB − Used to subtract the byte from byte/word from word.
SBB − Used to perform subtraction with borrow.
DEC − Used to decrement the provided byte/word by 1.
NPG − Used to negate each bit of the provided byte/word and add 1/2's complement.
CMP − Used to compare 2 provided byte/word.
AAS − Used to adjust ASCII codes after subtraction.
DAS − Used to adjust decimal after subtraction.

Instruction to perform multiplication

MUL − Used to multiply unsigned byte by byte/word by word.
IMUL − Used to multiply signed byte by byte/word by word.
AAM − Used to adjust ASCII codes after multiplication.

Instructions to perform division

DIV − Used to divide the unsigned word by byte or unsigned double word by word.
IDIV − Used to divide the signed word by byte or signed double word by word.
AAD − Used to adjust ASCII codes after division.
CBW − Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
CWD − Used to fill the upper word of the double word with the sign bit of the lower word.

3-Bit Manipulation Instructions

- These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Instructions to perform logical operation

NOT − Used to invert each bit of a byte or word.
AND − Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
OR − Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
XOR − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
TEST − Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

SHL/SAL − Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
SHR − Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
SAR − Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

ROL − Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
ROR − Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
RCR − Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
RCL − Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

4-String Instructions

- String is a group of bytes/words and their memory is always allocated in a sequential order.

REP − Used to repeat the given instruction till CX ≠ 0.
REPE/REPZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
REPNE/REPNZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
MOVS/MOVSB/MOVSW − Used to move the byte/word from one string to another.
COMS/COMPSB/COMPSW − Used to compare two string bytes/words.
INS/INSB/INSW − Used as an input string/byte/word from the I/O port to the provided memory location.
OUTS/OUTSB/OUTSW − Used as an output string/byte/word from the provided memory location to the I/O port.
SCAS/SCASB/SCASW − Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
LODS/LODSB/LODSW − Used to store the string byte into AL or string word into AX.

5-Program Execution Transfer Instructions (Branch and Loop Instructions)

- These instructions are used to transfer/branch the instructions during an execution. It includes the Instructions to transfer the instruction during an execution without any condition

CALL − Used to call a procedure and save their return address to the stack.
RET − Used to return from the procedure to the main program.
JMP − Used to jump to the provided address to proceed to the next instruction.
Instructions to transfer the instruction during an execution with some conditions −
JA/JNBE − Used to jump if above/not below/equal instruction satisfies.
JAE/JNB − Used to jump if above/not below instruction satisfies.
JBE/JNA − Used to jump if below/equal/ not above instruction satisfies.
JC − Used to jump if carry flag CF = 1

JE/JZ − Used to jump if equal/zero flag ZF = 1
JG/JNLE − Used to jump if greater/not less than/equal instruction satisfies.
JGE/JNL − Used to jump if greater than/equal/not less than instruction satisfies.
JL/JNGE − Used to jump if less than/not greater than/equal instruction satisfies.
JLE/JNG − Used to jump if less than/equal/if not greater than instruction satisfies.
JNC − Used to jump if no carry flag (CF = 0)
JNE/JNZ − Used to jump if not equal/zero flag ZF = 0
JNO − Used to jump if no overflow flag OF = 0
JNP/JPO − Used to jump if not parity/parity odd PF = 0
JNS − Used to jump if not sign SF = 0
JO − Used to jump if overflow flag OF = 1
JP/JPE − Used to jump if parity/parity even PF = 1
JS − Used to jump if sign flag SF = 1

6-Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values.
STC − Used to set carry flag CF to 1
CLC − Used to clear/reset carry flag CF to 0
CMC − Used to put complement at the state of carry flag CF.
STD − Used to set the direction flag DF to 1
CLD − Used to clear/reset the direction flag DF to 0
STI − Used to set the interrupt enable flag to 1, i.e., enable INTR input.
CLI − Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

7-Iteration Control Instructions

These instructions are used to execute the given instructions for number of times.
LOOP − Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
LOOPE/LOOPZ − Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
LOOPNE/LOOPNZ − Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
JCXZ − Used to jump to the provided address if CX = 0

8-Interrupt Instructions

These instructions are used to call the interrupt during program execution.
INT − Used to interrupt the program during execution and calling service specified.
INTO − Used to interrupt the program during execution if OF = 1
IRET − Used to return from interrupt service to the main program

Assembler directives of 8086 microprocessor

Introduction:

- Assembler directives are the directions to the assembler .which indicate how an operand or section of the program is to be processed.
- These are also called pseudo operations which are not executable by the microprocessor.

Assembler directives:

The various directives are-

1. ASSUME: The ASSUME directive is used to inform the assembler the name of the logical segment it should use for a specified segment.

Ex: ASSUME DS: DATA tells the assembler that for any program instruction which refers to the data segment, it should use the logical segment called DATA.

2. DB -Define byte. It is used to declare a byte variable or set aside one or more storage locations of type byte in memory.

For example, CURRENT_VALUE DB 36H tells the assembler to reserve 1 byte of memory for a variable named CURRENT_ VALUE and to put the value 36 H in that memory location when the program is loaded into RAM .

3. DW -Define word. It tells the assembler to define a variable of type word or to reserve storage locations of type word in memory.

4. DD(define double word) :This directive is used to declare a variable of type double word or restore memory locations which can be accessed as type double word.
5.DQ (define quad word) :This directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage in memory .
6.DT (define ten bytes): It is used to inform the assembler to define a variable which is 10 bytes in length or to reserve 10 bytes of storage in memory.
7. EQU –Equate it is used to give a name to some value or symbol. Every time the assembler finds the given name in the program, it will replace the name with the value or symbol we have equated with that name
8. ORG (**Originate)**: The ORG statement changes the starting offset address of the data.
It allows to set the location counter to a desired value at any point in the program. For example the statement ORG 3000H tells the assembler to set the location counter to 3000H.
9 .PROC- Procedure: It is used to identify the start of a procedure. Or subroutine.
10. END- End program .This directive indicates the assembler that this is the end of the program module. The assembler ignores any statements after an END directive.
11. **ENDP**- End procedure: It indicates the end of the procedure (subroutine) to the assembler.
12. ENDS-End Segment: This directive is used with the name of the segment to indicate the end of that logical segment.
Ex: CODE SEGMENT: Start of logical segment containing code
CODE ENDS : End of the segment named CODE.

Different Assembler Operator in 8085 Microprocessor

Different types of operators are:

1) Arithmetic: + , - , * , /
2) Logical : AND, OR, XOR, NOT
3) SHL and SHR: Shift during assembly
4) [ ]: index
5) HIGH: returns higher byte of an expression
6) OFFSET: returns offset address of a variable
7) SEG: returns segment address of a variable
8) PTR: used with type specifications BYTE, WORD, RWORD, DWORD, QWORD E.g. INC BYTE PTR [BX]
9) Segment override MOV AH, ES: [BX]

5.9-Simple Assembly Language Programming Using 8086 Instructions

1-Write a Program for Read a Character from the Keyboard

```
MOV ah, 1h      //keyboard input subprogram
INT 21h         // character input
// character is stored in al
MOV c, al       //copy character from alto c
```

2-Write a Program for Reading and Displaying a Character

```
MOV ah, 1h      // keyboard input subprogram
INT 21h         //read character into al
MOV dl, al      //copy character to dl
MOV ah, 2h      //character output subprogram
INT 21h         // display character in dl
```

3-Assembly language programming using 8086 Microprocessor.

Addition

```
ORG0000h
MOV DX, #07H    // move the value 7 to the register AX//
MOV AX, #09H    // move the value 9 to accumulator AX//
Add AX, 00H     // add CX value with R0 value and stores the result in AX//
```

END

Multiplication

```
ORG0000h
MOV DX, #04H     // move the value 4 to the register DX//
MOV AX, #08H     // move the value 8 to accumulator AX//
MUL AX, 06H       // Multiplied result is stored in the Accumulator AX //
END
```

Subtraction

```
ORG 0000h
MOV DX, #02H     // move the value 2 to register DX//
MOV AX, #08H     // move the value 8 to accumulator AX//
SUBB AX, 09H     // Result value is stored in the Accumulator A X//
END
```

Division

```
ORG 0000h
MOV DX, #08H     // move the value 3 to register DX//
MOV AX, #19H     // move the value 5 to accumulator AX//
DIV AX, 08H     // final value is stored in the Accumulator AX //
END
```

POSSIBLE SHORT TYPE QUESTION WITH ANSWER

Q 1-What is the size of address and data bus in 8086 MP. (19-W)

Ans-

- Size of 8085 is 8-bit microprocessor,
- Whereas 8086 is 16-bit microprocessor.
- Address Bus is 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- Memory − 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.

Q2-Write different memory segment used in 8086 MP and their functions. (S-23)

Ans-

- The Bus Interface Unit (BIU) contains four 16 bit special purpose registers (mentioned below) called as Segment Registers.
- **Code segment register (CS):** is used for addressing memory location in the code segment of the memory, where the executable program is stored.
- **Data segment register (DS):** points to the data segment of the memory where the data is stored.
- **Extra Segment Register (ES):** also refers to a segment in the memory which is another datasegment in the memory.
- **Stack Segment Register (SS):** is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data

Q3-Difference between the minimum and maximum mode of 8086 MP

In minimum mode there can be only one processor i.e. 8086.	In maximum mode there can be multiple processors with 8086, like 8087 and 8089.
ALE for the latch is given by 8086 as it is the only processor in the circuit.	ALE for the latch is given by 8288 bus controller as there can be multiple processors in the circuit.
Multiprocessing cannot be performed hence performance is lower.	As multiprocessing can be performed, it can give very high performance.

Q5-Mention the name of flags available in status register of 8086 MP. (S-24)

Flags is a 16-bit register containing 9 one-bit flags.

- Overflow Flag (OF)
- Direction Flag (DF)

- Interrupt-enable Flag (IF).
- Single-step Flag (TF)
- Sign Flag (SF
- Zero Flag (ZF)
- Auxiliary carry Flag (AF)
- Parity Flag (PF)
- Carry Flag (CF)

Q6-Define NMI in 8086 MP

- **NMI** is a non-mask able interrupt.
- Interrupt is processed in the same way as the INTR interrupt. Interrupt type ofthe NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h.
- This interrupt has higher priority than the mask able interrupt.

Q7- What is the maximum memory size that can be addressed by 8086 MP. (S-24)

Ans: - The maximum memory size that can be addressed by an 8086 microprocessor is 1 (MB).

Explanation: - The 8086 has a 20-bit address bus, which allows it to access 2^20 memory location1 MB.

Address bus size: 20 bits

Maximum addressable memory: 1 MB

POSSIBLE LONG TYPE QUESTION

Q1-With neat diagram explains architecture of 8086 MP. (S-24)

Q2-Explain pin description of 8086 MP.

Q3-Explain function of different flags in 8086 MP.

Q4-Explain different addressing mode in 8086 MP With examples.

Q5-What is the different between the MUL and IMUL instruction in 8086 MP.(S-23)

Q6- What is different between the DIV and IDIV instruction in 8086 MP.

Q7-Explain minimum mode of 8086 MP.

Q8-Explain maximum mode of 8086 MP.

Q9- Write an assembly language programming to add two 16-bit number. (S-23)

Q10- Write an assembly language programming to subtract two 16-bit number.

Q11- Write an assembly language programming for multiplication of 16-bit numbers using 8086 instructions. (S-24)

Chapter No- 06

Microcontroller (architecture and programming-8bit)

Learning Object-

6.1-Distinguish between Microprocessor & Microcontroller.

6.2-8 bit & 16-bit microcontroller.

6.3-CISC & RISC processor.

6.4-Architecture of 8051Microcontroller.

6.5-Signal Description of 8051Microcontrollers.

6.6-Memory Organization-RAM structure, SFR.

6.7-Registers, timers, interruptsof8051Microcontrollers.

6.8-Addressing Modes of 8051.

6.9-Simple 8051 Assembly Language Programming Arithmetic& Logic Instructions, JUMP, LOOP, CALL Instructions, I/O Port Programming.

6.10-Interrupts, Timer & Counters.

6.11-Serial Communication.

6.12-Microcontroller Interrupts and Interfacing to 8255.

6.1- Distinguish Between Microprocessor &Microcontroller

Microprocessor	Microcontroller
Microprocessor is the heart of Computer system.	Micro Controller is the heart of an embedded system.
It is only a processor, so memory and I/O components need to be connected externally	Micro Controller has a processor along with internal memory and I/O components.
Memory and I/O has to be connected externally, so the circuit becomes large.	Memory and I/O are already present, and the internal circuit is small.
You can't use it in compact systems	You can use it in compact systems.
Cost of the entire system is high	Cost of the entire system is low
Due to external components, the total power consumption is high. Therefore, it is not ideal for the devices running on stored power like batteries.	As external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries.

6.2- 8-bit & 16-bit microcontroller

- A microcontroller is a single and small chip having integrated circuit, processor core and embedded programmable I/O peripherals. 1 bit to 512 bit are common example of microcontrollers.

8-bit Microcontroller

- 8-bit microcontroller is kind of microcontroller having all characteristics of microcontroller and its data units are mostly 8 bits wide.
- 8 bits wide means your CPU can use 8-bit data bus or pipe and can access the same size data by a single device instruction.
- For every cycle of instruction its range is 0 to 255. It requires 20mA current to work. Intel 8008 was the first model having 8-bit micro-controller.

16-bit Microcontroller

- 16-bit microcontroller is more advanced than 8-bit microcontroller.
- It is more accurate and precise in performing mathematical and technical tasks.
- For every cycle of instruction its bit range is extended from 0 to 65,535.
- As 16-bit controller is two time more than 8-bit controller, it can work on two 16 bit numbers. It requires 10mA current to perform.

6.3-CISC & RISC Processor

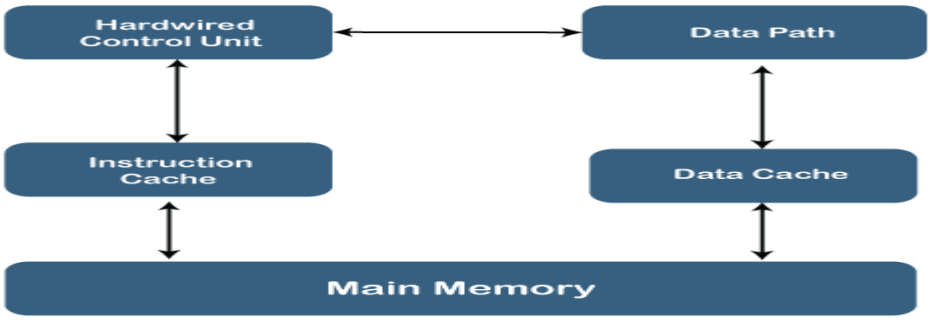
Difference between the RISC and CISC Processors

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It emphasizes on software to optimize the instruction set.	It emphasizes on hardware to optimize the instruction set.
It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.
It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
RISC has simple decoding of instruction.	CISC has complex decoding of instruction.
Uses of the pipeline are simple in RISC.	Uses of the pipeline are difficult in CISC.
It uses a limited number of instruction that requires less time to execute the instructions.	It uses a large number of instruction that requires more time to execute the instructions.

RISC Processor

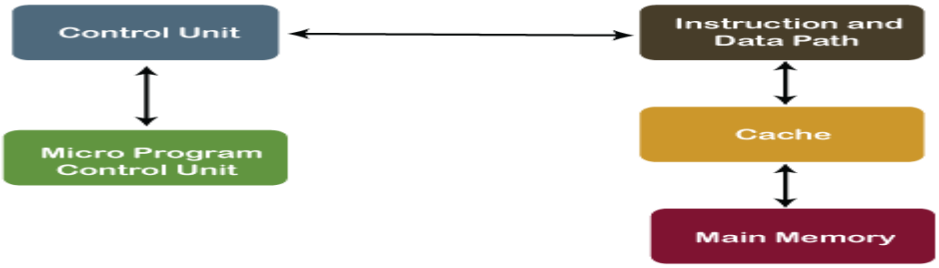
- RISC stands for **Reduced Instruction Set Computer Processor**, a microprocessor architecture with a simple collection and highly customized set of instructions.
- It is built to minimize the instruction execution time by optimizing and limiting the number of instructions. It means each instruction cycle requires only one clock cycle, and each cycle contains three parameters: fetch, decode and execute.
- The RISC processor is also used to perform various complex instructions by combining them into simpler ones.
- RISC chips require several transistors, making it cheaper to design and reduce the execution time for instruction. Examples of RISC processors are SUN's SPARC, PowerPC, Microchip PIC processors, RISC-V.

RISC Architecture



CISC Processors Architecture

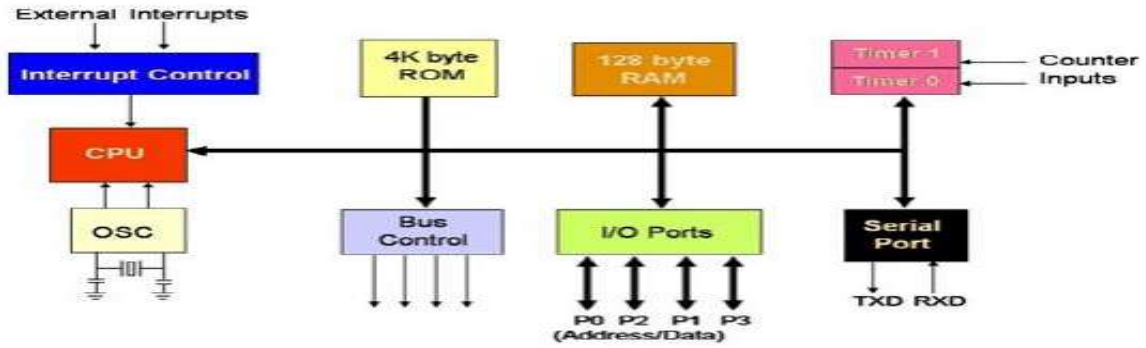
- The CISC architecture helps reduce program code by embedding multiple operations on each program instruction, which makes the CISC processor more complex.
- The CISC architecture-based computer is designed to decrease memory costs because large programs or instruction required large memory space to store the data,



6.4-Architecture of 8051Microcontroller

8051 Microcontroller Architecture

The internal architecture of 8051 Microcontroller represented in form of block diagram.



CPU (Central Processing Unit):

- CPU act as a mind of any processing machine. It synchronizes and manages all processes that are carried out in microcontroller.
- User has no power to control the functioning of CPU.
- CPU manage the different types of registers available in 8051 microcontroller.

Interrupts:

- Interrupts is a sub-routine call that given by the microcontroller when some other program with high priority is request for acquiring the system buses.

Types of interrupt in 8051 Microcontroller:

The five sources of interrupts in 8051 Microcontroller:

1-Timer 0 overflow interrupt - TF0

2-Timer 1 overflow interrupt - TF1

3-External hardware interrupt - INT0

4-External hardware interrupt - INT1

5-Serial communication interrupt - RI/TI

Memory:

- For operation Micro-controller required a program. This program guides the microcontroller to perform the specific tasks.
- Microcontroller also required memory for storage of data and operands for the short duration.

Bus:

- Bus is a group of wires which uses as a communication canal or acts as means of data transfer.
- The different bus configuration includes 8, 16 or more cables.
- Therefore, a bus can bear 8 bits, 16 bits all together.

Types of buses in 8051 Microcontroller:

Address Bus:

- 8051 microcontrollers are consisting of 16-bit address bus.
- It is generally be used for transferring the data from Central Processing Unit to Memory.

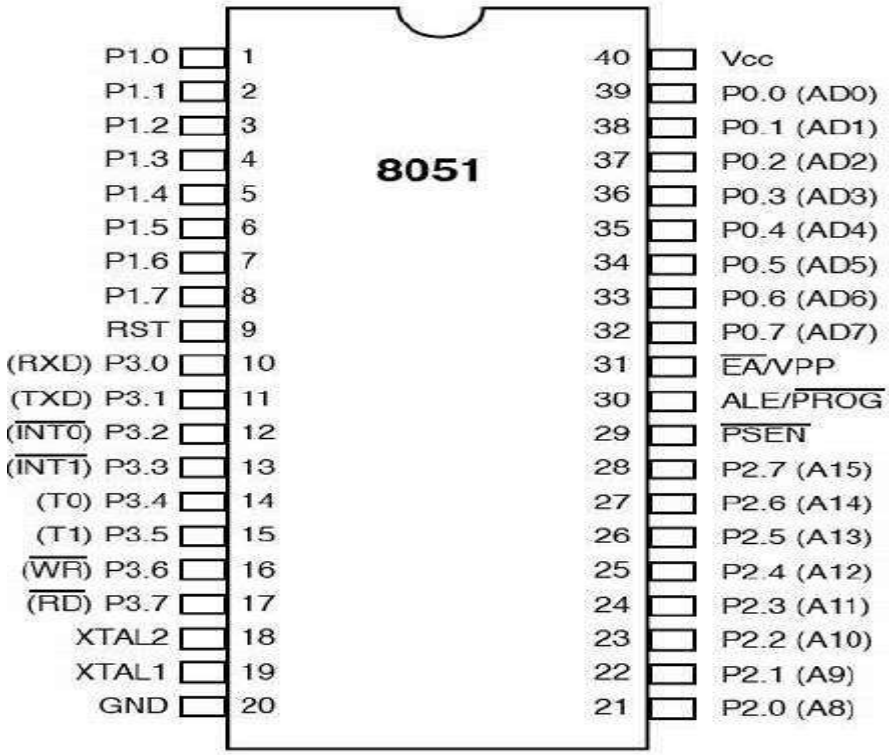
Data bus:

- 8051 microcontrollers are consisting of 8 bits' data bus.
- It is generally be used for transferring the data from one peripherals position to other peripherals.

Oscillator:

- As the microcontroller is digital circuit therefore it needs timer for their operation.
- To perform timer operation inside microcontroller it required externally connected or on-chip oscillator.
- Microcontroller is used inside an embedded system for managing the function of devices. Therefore, 8051 uses the two 16 bit counters and timers.
- For the operation of this timers and counters the oscillator is used inside microcontroller.

6.5- Signal Description of 8051Microcontrollers



Pins 1-8: These pins belongs to Port 1 of microcontroller. Port 1 is used as domestically pulled up, quasi bi directional input/output port.

Pin 9: It is a RESET pin which is utilized to set the microcontroller 8051 to its primary value. During the beginning of an application the RESET pin is to be set elevated for two machine rotations.

Pins 10-17: These pins belong to Port 3 of microcontroller. Port 3 can be used for number of functions such as timer input, interrupts, serial communication indicator for transmitting (TxD) and receiving (RxD). It is also known as domestic pull up port with quasi bi direction port embedded within.

Pins 18 and 19: These pins are generally be used for interfacing outer crystal oscillator with given system clock.

Pin 20: This pin titled as V_{ss}. It symbolizes ground voltage or 0 V is connected to this pin of microcontroller.

Pin 21-28: These pins belong to port 2 of microcontroller. Port 2 can be used as Input/output port, senior order address bus is multiplexed with this quasi bi directional port.

Pin 29: This pin belongs to Program Store Enable or PSEN. It is used for interpreting the sign from outer program memory.

Pin 30: This pin belongs to External Access or EA input is used for permit or prohibits outer memory interfacing. If there is no outer memory need, this pin is set to high by linking it with supply voltage .

Pin 31: This pin belongs to Address Latch Enable or ALE is used for de-multiplexing the address data indication of port 0 for outer memory interfacing

Pin 32-39: These pins belong to Port 0 of the microcontroller. Port 0 can be used as input/output port, lower order address and data bus signals are multiplexed with this port. This pin act as bi directional Input/output port and outer connected pull up resistors are necessary for utilizing these ports as Input/output.

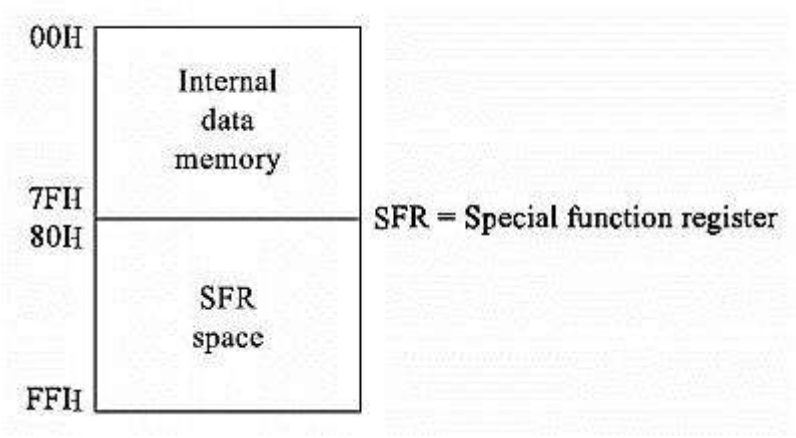
Pin 40: This pin is used to provide power supply to the circuit.

6.6-Memory Organization-RAM structure, SFR

Memory Organization of RAM

- The internal data memory of 8051 is divided into two groups.
- These are a set of eight registers and a scratch pad memory.
- These eight registers are R0 toR7.
- The address range 00H to 07H is used to access the registers, and the rest are scratch pad memory.

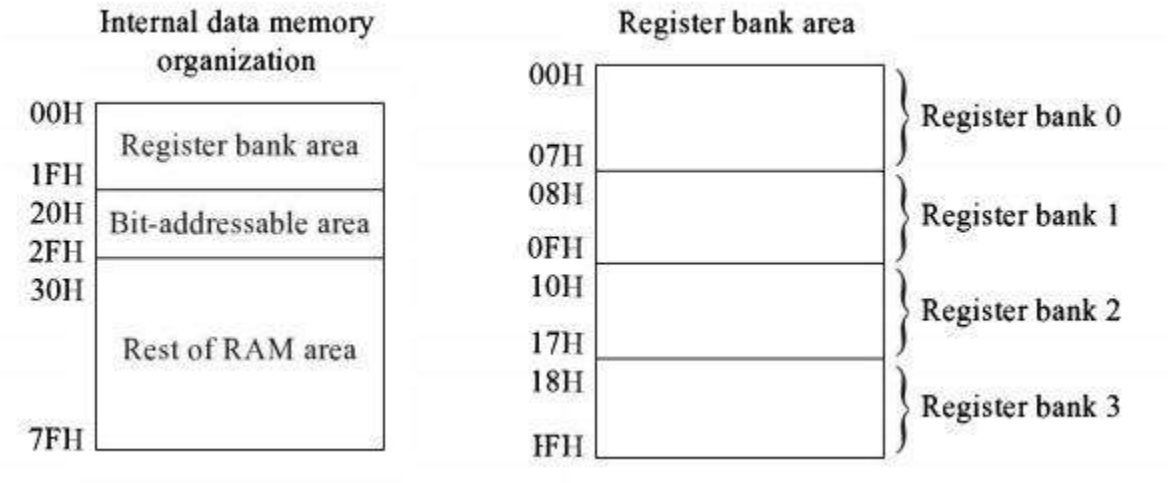
8051 Provides four register bank, but only one register bank can be used at any point in time. To select the register bank, two bits of PSW (Program Status Word) are used.



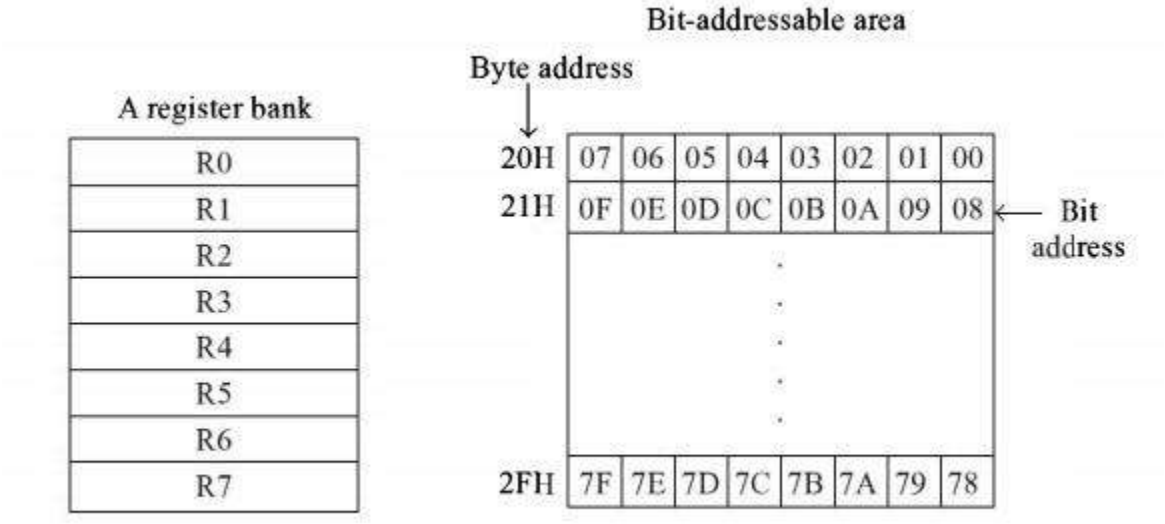
So the following addressing can be used to select register banks.

Address Range	Register Bank
00H to 07H	Register Bank 0
08H to 0FH	Register Bank 1
10H to 17H	Register Bank 2

Address Range	Register Bank
18H to 1FH	Register Bank 3

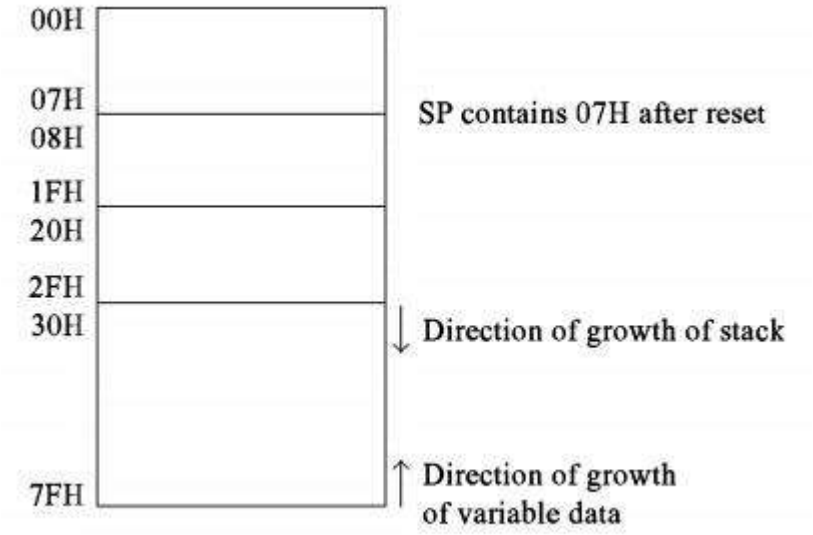


- When all of the register banks are being used, the scratch pad area will be 20H to 7FH. But from 20H to 2FH (16 bytes or 128 bits) can be used as bit addressable RAM.
- By using some simple instructions with 8-bit memory address we can check the bit addressing. For an example the instruction **CLR 6FH**, using this instruction it clears the location 6FH.



Stack

- The stack area in 8051always can be implemented in the internal data memory.
- Here the stack pointer (SP) is an only 8-bit register, because the internal RAM area is only in range 00H to 7FH, and when all register banks are being used, the stack location will be in range 30H to 7FH.
- So in such a case, the SP will be initialized with 2FH.



- The stack pointer SP increases before each PUSH operation and decreases after each pop instruction.
- When the 8051 is reset, the Stack Pointer will point to 07H.
- The location 08H to 7FHcan be used as a stack. We are assuming that the register bank 0 is in use and 20H to 27H are not like bit-addressable area.

8051 Microcontroller Special Function Registers (SFRs)

- The 8051 Microcontroller Special Function Registers act as a control table that monitor and control the operation of the 8051 Microcontroller.
- In Internal RAM Structure, the Address Space from 80H to FFH is allocated to SFRs.
- Out of these 128 Memory Locations (80H to FFH), there are only 21 locations that are actually assigned to SFRs.
- Each SFR has one Byte Address and also a unique name which specifies its purpose.
- All the 21 8051 Microcontroller Special Function Registers (SFRs) along with their functions and Internal RAM Address is given in the following table.

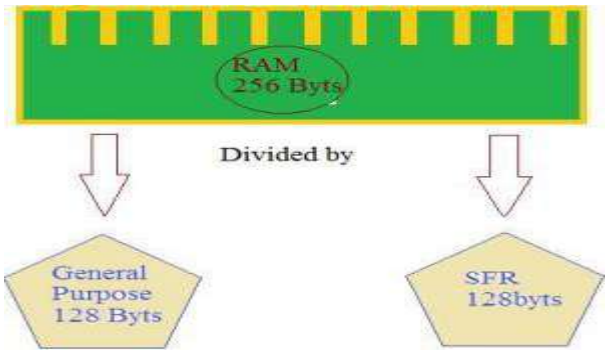
Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

6.7-Registers, timers, interruptsof8051Microcontrollers

Types of Registers of 8051 MC

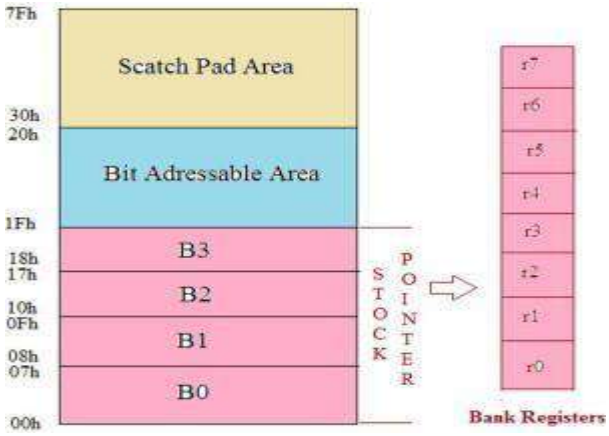
The 8051 microcontroller contains mainly two types of registers:

- The 8051 microcontroller consists of 256 bytes of RAM,
- General-purpose registers (Byte addressable registers)
- Special function registers (Bit addressable registers)



General purpose Register

- The general-purpose memory is called as the RAM of the 8051 microcontrollers, which is divided into 3areas such as banks, bit-addressable area, and scratch-pad area.
- The banks contain different general- purpose registers such as R0-R7, and all such registers are byte-addressable registers that store or remove only 1-byte of data.



PSW (Program Status Word) Register

- The PSW register is a bit and byte-addressable register.
- This register reflects the status of the operationthat is carried out in the controller.
- The physical address of the PSW starts from D0h and the individual bits are accessed with D0h to D7h.

D7	D6	D5	D4	D3	D2	D1	D0
C	---	AC	RS1	RS0	OV	---	P

Carry Flag (C): The Address of the Carry flag is D7. This carry flag is affected when the bit isgenerated from the

When C=0 carry resetsC=1 carry sets

```
  1 0 0 0 0 0 0 0
  1 0 1 1 0 1 0 1
1 ---------------
  0 0 1 1 0 1 0 1
```

C Carry Flag

Auxiliary Flag(AC): The address of the auxiliary carry is D5. This auxiliary carry is affected when abit is generated from the 3rd position to the 4th position.

AC=0 auxiliary is resetAC=1 auxiliary is set

```
      AC
      ↓
  0 0 0 0 1 0 1 0
  0 0 1 0 1 1 0 1
-----------------
  0 0 1 1 0 1 1 1
```

Overflow Flag (OV): The address of the overflow flag is D2. When a bit is generated from the 6thposition to the 7th position, then the overflow flag is affected.

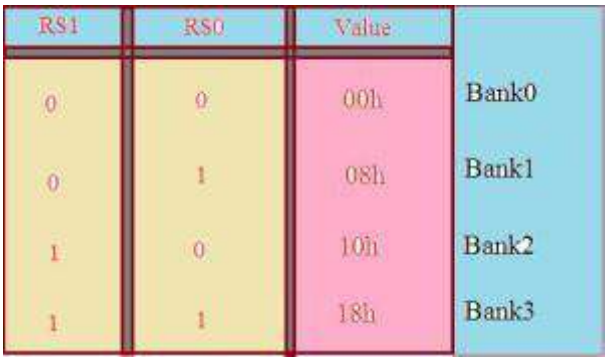
OV=0 overflow flag resetsOV=1 overflow flag sets

```
    OV
    ↓
  0 1 0 0 1 1 0 1
  0 1 1 0 1 0 1 0
-----------------
1 0 1 1 0 1 1 1
```

Parity Flag (P): The address of the parity flag is D0. While performing arithmetic operations, if theresult is 1, then the parity flag is set – otherwise, reset.

RS1 and RS0

The RS1 and RS0, the bits in the PSW register, are used to select different memory locations (bank0 to bank4) in the RAM.



RS1	RS0	Value	
0	0	00h	Bank0
0	1	08h	Bank1
1	0	10h	Bank2
1	1	18h	Bank3

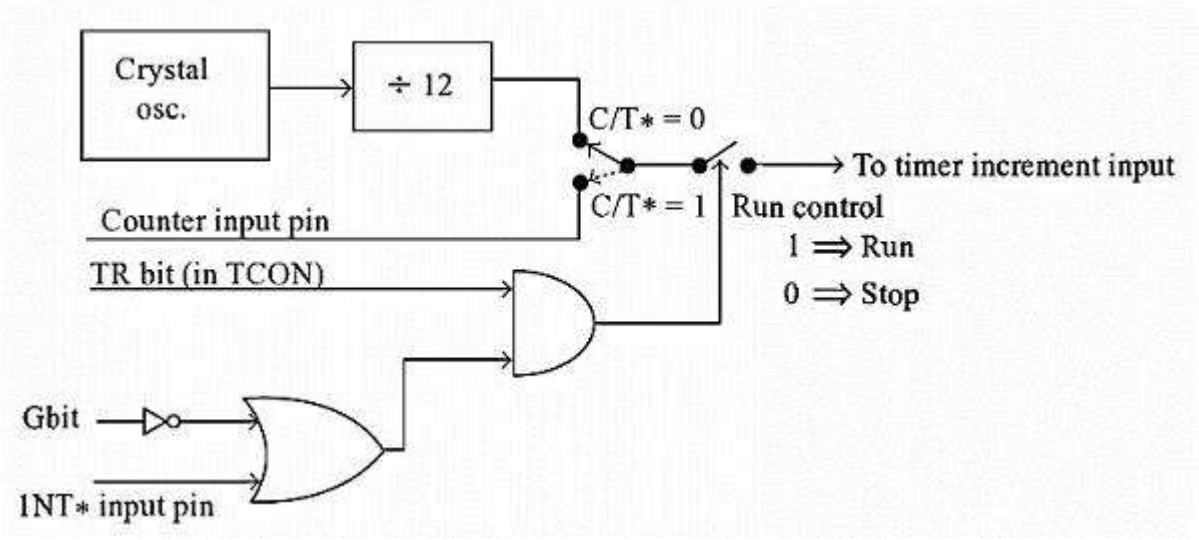
Times of 8051

- In Intel 8051, there are two 16-bit timer registers.
- These registers are known as Timer0 andTimer1.
- The timer registers can be used in two modes.
- These modes are Timer mode and the Counter mode.
- There are four different modes of the Timer or Counter.
- The Mode 0 to Mode 2 are for both of the Timer/Counter.
- Mode 3 has a different meaning for each timer register.
- There is a register called TMOD. This register can be programmed to configure these timers or counters.

TMOD Register

TMOD (Timer Mode) is an SFR. The address of this register is 89H.

Timer	**Timer1 Mode**				**Timer0 Mode**			
Bit Details	Gate (G)	C/T	M1	M0	Gate (G)	C/T	M1	M0



In the following table, we will see the bit details and their different operations for high or low value.

Bit Details	**High Value(1)**	**Low Value(0)**
C/**T**	Configure for the Counter operations	Configure for the Timer operations
Gate (G)	Timer0 or Timer1 will be in Run Mode when TRX bit of TCON register is high.	Timer0 or Timer1 will be in Run Mode when TRX bit of TCON register is high and INT0 or INT1 is high.

Bit Details	**00**	**01**	**10**	**11**
M1 M0	This is for Mode 0. (8-bit timer/counter, with 5-bit pre-scaler)	This is Mode 1. (16-bit timer/counter)	This is Mode 3 (8-bit auto reload-timer/counter)	This is Mode 3 (The function depends on Timer0 or Timer1)

Types of Interrupts in 8051 Microcontroller

- The 8051 microcontroller can recognize five different events that cause the main program to interrupt from thenormal execution. These five sources of interrupts in 8051are:
- Timer 0 overflow interrupt- TF0
- Timer 1 overflow interrupt- TF1
- External hardware interrupt- INT0
- External hardware interrupt- INT1
- Serial communication interrupt- RI/TI
- The Timer and Serial interrupts are internally generated by the microcontroller, whereas the external interrupts are generated by additional interfacing devices or switches that are externally connected to the microcontroller.
- These external interrupts can be edge triggered or level triggered. When an interrupt occurs, the microcontroller executes the interrupt service routine so that memory location corresponds to the interrupt that enables it.

6.8-Addressing Modes of 8051

- In 8051 there are 1-byte, 2-byte instructions and very few 3-byte instructions are present.
- The opcodes are 8-bitlong. As the opcodes are 8-bit data, there are 256 possibilities.
- Among 256, 255 opcodes are implemented.
- The clock frequency is12MHz, so 64 instruction types are executed in just 1 µs, and rest are just 2 µs. TheMultiplication and Division operations take 4 µs to execute.

In 8051 There are six types of addressing modes.

- Immediate Addressing Mode
- Register Addressing Mode
- Direct Addressing Mode
- Register Indirect Addressing Mode
- Indexed Addressing Mode
- Implied Addressing Mode

Immediate Addressing mode

- In this Immediate Addressing Mode, the data is provided in the instruction itself.
- The data is providedimmediately after the opcode.
- These are some examples of Immediate Addressing Mode.

  MOVA, #0AFH; MOVR3, #45H; MOVDPTR, #FE00H;

Register addressing mode

- In the register addressing mode the source or destination data should be present in a register (R0 to R7). Theseare some examples of Register Addressing Mode.

  MOVA, R5; MOVR2, #45H;MOVR0, A;

- In 8051, there is no instruction like **MOVR5, R7**. But we can get the same result by using this instruction **MOV R5,**

Direct Addressing Mode

- In the Direct Addressing Mode, the source or destination address is specified by using 8-bit data in the instruction.
- Only the internal data memory can be used in this mode.
- Here some of the examples of directAddressing Mode.
  MOV80H, R6;MOVR2, 45H;MOVR0, 05H;

Register indirect addressing Mode

- In this mode, the source or destination address is given in the register.
- By using register indirect addressing mode, the internal or external addresses can be accessed.
- The R0 and R1 are used for 8-bit addresses, and DPTRis used for 16-bit addresses, no other registers can be used for addressing purposes.
- some examples ofthis mode.
  MOV0E5H, @R0;MOV@R1, 80H
  In the instructions, the @ symbol is used for register indirect addressing

Indexed addressing mode

- In the indexed addressing mode, the source memory can only be accessed from program memory only.
- Thedestination operand is always the register A.
- These are some examples of Indexed addressing mode.
  MOVCA, @A+PC; MOVCA, @A+DPTR;

Implied Addressing Mode

- In the implied addressing mode, there will be a single operand.
- These types of instruction can work on specificregisters only.
- These types of instructions are also known as register specific instruction.
- Here are some examples of Implied Addressing Mode.
  RLA; SWAPA;
- These are 1- byte instruction. The first one is used to rotate the A register content to the Left. The second one isused to swap the nibbles in A.

6.9-Simple 8051 Assembly Language Programming Arithmetic& Logic Instructions, JUMP, LOOP, CALL Instructions, I/O Port Programming.

Program

MOVR0, #20H; set source address 20H to R0

MOVR1, #30H; set destination address 30H to R1

MOVA, @R0; take the value from source to register A

MOVR5, A; Move the value from A to R5

MOVR4, #00H; Clear register R4 to store carry

INCR0; Point to the next location

MOVA, @R0; take the value from source to register A

ADDA, R5; Add R5 with A and store to register A

JNC SAVE

INCR4; Increment R4 to get carry

MOVB, R4; Get carry to register B

MOV@R1, B; Store the carry first

INCR1; Increase R1 to point to the next address

SAVE: MOV@R1, A; Store the result

HALT: SJMP HALT; Stop the program

So by adding 5FH + D8H, the result will be 137H. The 01H will be stored at 30H, and 37 is stored at 31H.

Output

Address	Value
	. . .
20H	5FH
21H	D8H
	. . .
30H	01H
31H	37H

Write a program to subtract two 8-bit numbers using 8051 microcontroller

Program

```
MOV R0, #20H; set source address 20H to R0
MOV R1, #30H; set destination address 30H to R1
MOV A, @R0; take the value from source to register A
MOV R5, A; Move the value from A to R5
MOV R4, #00H; Clear register R4 to store borrow
INC R0; Point to the next location
MOV A, @R0; take the value from source to register A
MOV R3, A; store second byte
MOV A, R5; get back the first operand
SUBB A, R3; Subtract R3 from A
JNC SAVE
INC R4; Increment R4 to get borrow
MOV B, R4; Get borrow to register B
MOV @R1, B; Store the borrow first
INC R1; Increase R1 to point to the next address
 SAVE: MOV @R1, A; Store the result
```

HALT: SJMP HALT; Stop the program

Output

Address	Value
	…
20H	73H
21H	BDH
	…
30H	01H
31H	B6H

The XOR Instruction

The XOR instruction implements the bitwise XOR operation. The XOR operation sets the resultant bit to 1, if and only if the bits from the operands are different. If the bits from the operands are same (both 0 or both 1), the resultant bit is cleared to 0.

For example,

Operand1: 0101

Operand2: 0011

After XOR -> Operand1: 0110

XORing an operand with itself changes the operand to 0. This is used to clear a register.

```
XOR EAX, EAX
```

The TEST Instruction

The TEST instruction works same as the AND operation, but unlike AND instruction, it does not change the first operand. So, if we need to check whether a number in a register is even or odd, we can also do this using the TEST instruction without changing the original number.

```
TEST    AL, 01H
JZ      EVEN_NUMBER
```

The NOT Instruction

The NOT instruction implements the bitwise NOT operation. NOT operation reverses the bits in an operand. The operand could be either in a register or in the memory.

For example,

Operand1: 0101 0011

After NOT -> Operand1: 1010 1100

Example 1:

Write a program to clear Accumulator [A], then Add 5 to the accumulator 20 times:

Answer:

```
;This program adds value 3 to the Accumulator 20 times
        MOV A,#0            ;Accumulator=0, Clear Accumulator
        MOV R2,#20          ;Load counter R2=20
AGAIN:  ADD A,#05           ;Add 05 to Accumulator
        DJNZ R2,AGAIN       ;Repeat Until R2=0 [20 times]
        MOV R5,A            ;Save Accumulator in R5
```

- The R2 register is used as a counter.
- The counter R2 is first set to 20. In each iteration, the instruction DJNZ decrements R2 and checks its value.
- If R2 is not zero, it jumps to the target address associated with label "AGAIN".
- This looping action continues until R2 becomes zero.
- After R2 becomes zero, it falls through the loop and executes the instruction immediately below it, in this case the "MOV R5, A" instruction.
- In the DJNZ instruction that the registers can be any of R0-R7. The counter can also be a RAM location.
- In the **Example # 1** The R2 holds the count and R2 is an 8-bit register, it can hold a maximum of FFH [255 Decimal]. The loop can be repeated a maximum of 256 times.

Loop Inside a Loop [NESTED LOOP]:

- The maximum count is 256. But if we want to repeat an action more times than 256, we use a loop inside a loop, which is called nested loop. In nested loop, we use two registers to hold the count.

Example # 2:

Write a program to load the accumulator with the value 66H, and complement the accumulator ACC 600 times:

Answer:

- 700 is larger than 255 [the maximum capacity of any register], we use two registers to hold the count. We are using R2, R3 for the count.

```
MOV A,$66H        ;Accumulator A=66H

MOV R3,#10        ;R3=10, the outer loop count

NEXT: MOV R2,#60        ;R2=60, the inner loop count

AGAIN: CPL A              ;Complement A register

DJNZ R2,AGAIN             ;Repeat inner loop 60 times

DJNZ R3,NEXT              ;Repeat outer loop 10, so 10X60=600 times
CPL A run
```

- R2 is used to keep the inner loop count. In the instruction "DJNZ R2, AGAIN", whenever R2 becomes zero, it falls through and "DJNZ R3, NEXT" is executed.
- This instruction forces the CPU to load R2 with the count 60 and the inner loop starts again. This process will continue until R3 becomes zero and the outer loop is finished.

8051 Conditional Jump Instructions:

Example # 3:

```
MOV A,R0         ;A = R0

JZ OVER          ;Jump if accumulator A = 0

MOV A,R1         ;A = R1

JZ OVER          ;Jump if accumulator A = 0

.....

OVER:
```

- Either R0 or R1 is zero, it jumps to the label OVER. The JZ instruction can be used only for register accumulator A.
- It can used only check to see whether the Accumulator is zero. It does not apply to any other register.

Example # 4:

Write a program in which if R4 register contains the value 0. Then put 55H in R4 register:

Answer:

```
MOV A,R4        ;Copy R4 to accumulator A

JNZ NEXT        ;Jump if accumulator is not zero

MOV R4,#55H     ;Put value 55H into register R4

NEXT: .....
```

JNC [Jump if No Carry i.e. Jumps if Carry Flag = 0]:

6.10-Interrupts, Timer & Counters

Timer & Counter

- A **timer** is a specialized type of clock which is used to measure time intervals.
- A timer that counts from zero upwards for measuring time is often called a **stopwatch**.
- It is a device that counts down from a specified time interval and used to generate a time delay, for example, an hour glass is a timer.
- A **counter** is a device that stores (and sometimes displays) the number of times a particular event or process occurred, with respect to a clock signal.
- It is used to count the events happening outside the microcontroller. In electronics, counters can be implemented quite easily using register-type circuits such as a flip-flop.

Difference between a Timer and a Counter

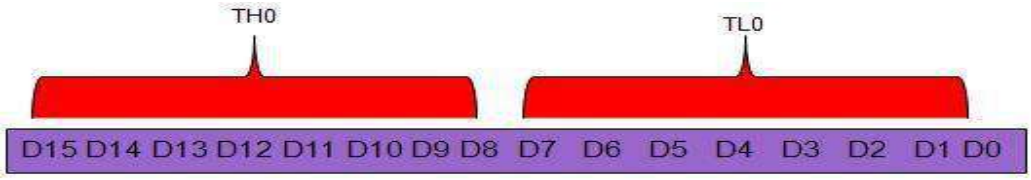
Timer	**Counter**
The register incremented for every machine cycle.	The register is incremented considering 1 to 0 transition at its corresponding to an external input pin (T0, T1).
Maximum count rate is 1/12 of the oscillator frequency.	Maximum count rate is 1/24 of the oscillator frequency.
A timer uses the frequency of the internal clock, and generates delay.	A counter uses an external signal to count pulses.

Timers of 8051 and their Associated Registers

- The 8051 has two timers, Timer 0 and Timer 1.
- They can be used as timers or as event counters.
- Both Timer 0 and Timer 1 are 16-bit wide.
- Since the 8051 follows an 8-bit architecture, each 16 bit is accessed as two separate registers of low-byte and high-byte.

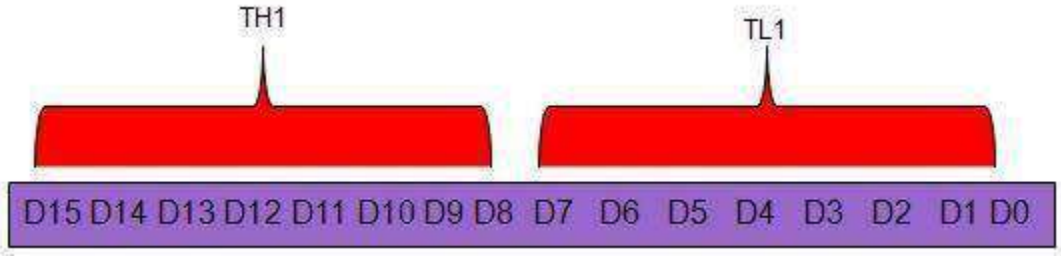
Timer 0 Register

- The 16-bit register of Timer 0 is accessed as low- and high-byte.
- The low-byte register is called TL0 (Timer 0 low byte) and the high-byte register is called TH0 (Timer 0 high byte).
- These registers can be accessed like any other register. For example, the instruction **MOV TL0, #4H** moves the value into the low-byte of Timer #0.



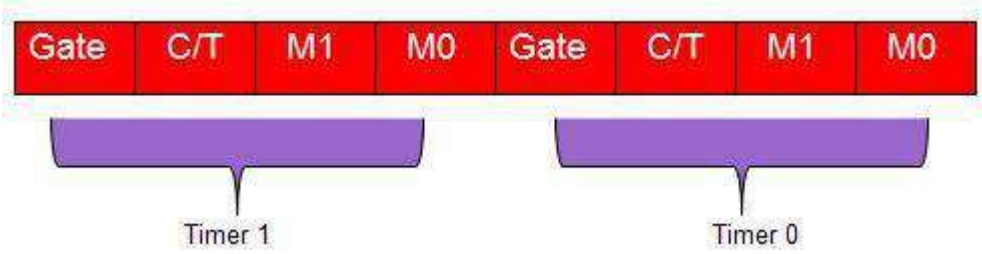
Timer 1 Register

- The 16-bit register of Timer 1 is accessed as low- and high-byte.
- The low-byte register is called TL1 (Timer 1 low byte) and the high-byte register is called TH1 (Timer 1 high byte).
- For example, the instruction **MOV TL1, #4H** moves the value into the low-byte of Timer 1.



TMOD (Timer Mode) Register

- Both Timer 0 and Timer 1 use the same register to set the various timer operation modes.
- It is an 8-bit register in which the lower 4 bits are set aside for Timer 0 and the upper four bits for Timers.



Gate − When set, the timer only runs while INT(0,1) is high.

C/T − Counter/Timer select bit.

M1 − Mode bit 1.

M0 − Mode bit 0.

6.11-Serial Communication

Data communication

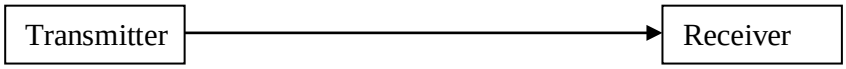
- The 8051 microcontroller is parallel device that transfers eight bits of data simultaneously over eight data lines to parallel I/O devices. Parallel data transfer over a long is very expensive.
- Hence, a serial communication is widely used in long distance communication.

- In serial data communication, 8-bit data is converted to serial bits using a parallel in serial out shift register and then it is transmitted over a single data line.

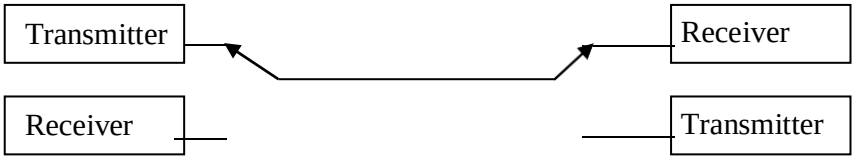
Basics of serial data communication,

Communication Links

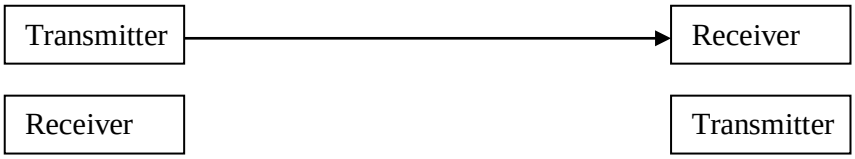
Simplex communication link: In simplex transmission, the line is dedicated for transmission. The transmitter sends and the receiver receives the data.



Half duplex communication link*:* In half duplex, the communication link can be used for either transmission or reception. Data is transmitted in only one direction at a time.



- Full duplex communication link:
- If the data is transmitted in both ways at the same time, it is a full duplex i.e. transmission and reception can proceed simultaneously.



- This communication link requires two wires for data, one for transmission and one for reception.

6.12-Microcontroller Interrupts and Interfacing to 8255

Circuit Diagram to interface 8255 PPI with 8051

Step 1:

- Connect the power pins of 8255 by connecting VCC and GND pins to the appropriate sources.
- The EA of 8051 should be connected to VCC as well.

Step 2:

- Pins P3.6 (WR) and P3.7 (RD) are to be connected to the WR and RD pins of 8255, respectively.

Step 3:

- Connect Port 0 to 74LS373 decoder, i.e. P0.0 – P0.7 to 1D – 8D.
- Connect the ALE pin of 8051 to the Enable pin (G) of 74LS373 to activate it.

Step 4:

- Tap the AD0 – AD7 lines of Port 0 and connect it directly to D0 – D7 pins of 8255.

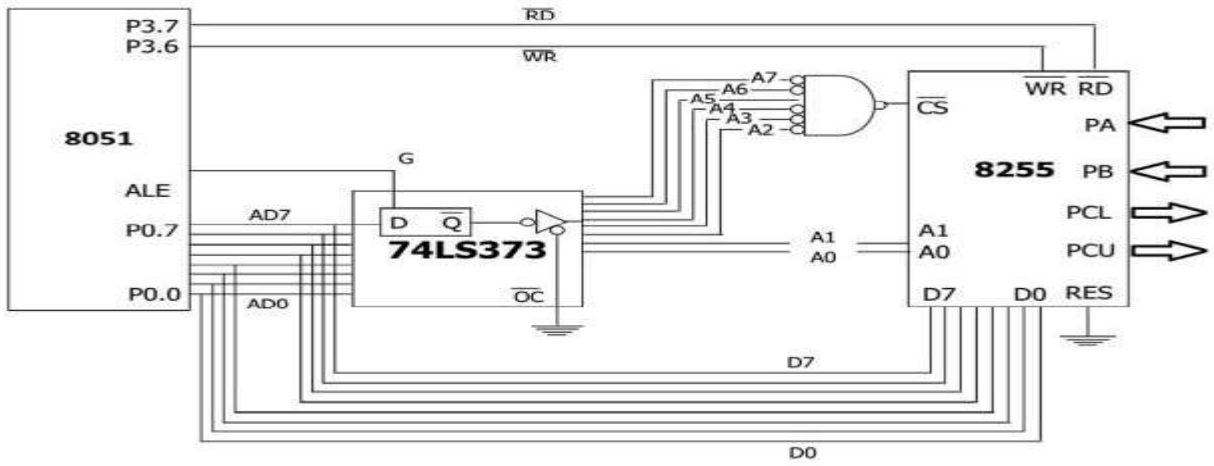
Step 5:

- After DE multiplexing, connect the pins A0 and A1 (from the output of IC74LS373 to the pins A0 and A1 of 8255, respectively.)

- These two pins indicate the mode in which 8255 is operating.

Step 6:

- The final step is to connect the chip select combinational logic that we designed.



Interrupt in 8051 Microcontroller

- 8051 has five interrupts. These interrupts are INT0, INT1,TO ,T1 , TI/RI. All of the interrupts can be enabled or disabled by using the IE (interrupt enable) register.
- The interrupt addresses of these interrupts are like below −

Interrupt	Address
INT0	0003H
INT1	000BH
T0	0013H
T1	001BH
TI/RI	0023H

Interrupt Enable (IE)Register

- This register can be used to enable or disable interrupts programmatically.
- This register is an SFR. The address is A8H.
- This byte is bit addressable. So it can be programmed by the user. The bits in this register has a different meaning. The register structure is looking like this:

Bit Address	AF	AE	AD	AC	AB	AA	A9	A8
Bit Details	EA	X	X	ES	ET1	EX1	ET0	EX0

Now, let us see the bit details and different operations when the value is low (0) and high (1).

Bit Details	High Value (1)	Low Value (0)
EA	Least significant 5 bits can decide enable or disable of these five interrupts.	Disable all five interrupts. It just ignores the rest five bits.
ES	Enable Serial Port Interrupt	Disable Serial Port Interrupt
ET1	Enable Timer1 interrupt	Disable Timer1 interrupt
EX1	Enable external interrupt 1 (INT1)	Disable external interrupt 1 (INT1)
ET0	Enable Timer0 interrupt	Disable Timer0 interrupt
EX0	Enable external interrupt 0 (INT0)	Disable external interrupt 0 (INT0)

POSSIBLE SHORT TYPE QUESTIONS WITH ANSWER

Q1 Difference between microprocessor and micro controller

- Microprocessor consists of only a Central Processing Unit, whereas Micro Controller contains a CPU, Memory, I/O all integrated into one chip.
- Microprocessor uses an external bus to interface to RAM, ROM, and other peripherals, on the other hand, Microcontroller uses an internal controlling bus.

Q2 Define SFR

- The Special Function Register (**SFR**) is the upper area of addressable memory, from address 0x80 to 0xFF
- Different status registers are mapped into the **SFR**, for use in checking the status of the 8051, and changing some operational parameters of the 8051

Q3-Which port of 8051 is used as address/data bus

- While connecting an 8051 to external memory, Port 0 can provide both address and data.
- The 8051 microcontrollers then multiplex the input as address or data in order to save pins.
- Dual role of Port 2 − Besides working as I/O, Port P2 is also used to provide 16-bit address bus for external memory along with Port 0.

Q4 Difference between CISC and RISC. (S-23)

- One of the major differences between RISC and CISC is that RISC emphasizes efficiency in cycles per instruction and CISC emphasizes efficiency in instructions per program.
- RISC needs more RAM, whereas CISC has an emphasis on smaller code size and uses less RAM overall than RISC.

Q5 What are the interrupt are present in 8051. (S-23)

- 8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI.
- Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

Q6- Define serial communication in 8051 mc

- Serial communication is a form of I/O in which the bits of a byte begin transferred appear one after the other in a timed sequence on a single wire.
- Serial communication has become the standard for inter computer communication. In this lab, we'll try to build a serial link between 8051 and PC using RS232

Q7- what does @ and # symbol indicate in 8051 micro controllers? Give one example of Each. (S-24)

Ans:-The "@" symbol is typically used to indicate a register address, while the "#" symbol represents an immediate value that is directly loaded into a register during an instruction.

Example: - **"MOV A, @R0"**
"MOV R1, #50"

Q8-Write various ports available in 8051. (S-24)

Ans: - The 8051 microcontroller has four 8-bit ports, labeled P0, P1, P2, and P3.
These ports are bidirectional, meaning they can act as both inputs and outputs.
Port 0-Can be used as a general-purpose input.
Port 1-Used for interfacing activities, such as connecting to switches, LEDs.
Port 2-Can be used as a general-purpose input/output port.
Port 3-Can be used as an input/output port

POSSIBLE LONG TYPE QUESTIONS

Q1-Explain internal architecture of 8051 with a neat diagram. (S-24)

Q2-Explain pin description of 8051 mc

Q3 Describe the memory organisation in the 8051 mc. (S-24)

Q4-Explain different addressing mode of 8051mc

Q5-Explain 8051 mc is interfacing to 8255 with neat diagram.

Q6-Write short note on timer and counter in 8051 mc