DIPLOMA IN ELECTRONICS & TELECOMMUNICATION ENGINEERING

(Effective FROM 2019-20 Sessions)

Th.3 MICROPROCESSOR & MICROCONTROLLER

LEARNING NOTES

UNIT-1:MICROPROCESSOR (ARCHITECTURE AND PROGRAMMING-8085-8-BIT)

1.1 Introduction to Microcomputer AND Microprocessor & distinguish between them.

Microcomputer

A microcomputer can be defined as a small sized, inexpensive, and limited capability computer. It has the same architectural block structure that is present on a computer. Present-day microcomputers are having smaller sizes. Nowadays, they are of the size of a notebook. But in the coming days also their sizes will get more reduced as well. Due to their lower costs, individuals can possess them as their personal computers. Because of mass production, they are becoming still cheaper. Initially, in the earlier days, they were not very much powerful. Their internal operations and instructions were very much limited and restricted. But at present days, microcomputers have not only multiplied and divide instructions on unsigned and signed numbers but are also capable of performing floating point arithmetic operations. In fact, they are becoming more powerful than the minicomputers and main computers of yesteryear.

As an example, the Commodore 64 was one of the most popular microcomputers of its era and is the best-selling model of home computer of all time.

So a microcomputer is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU). It includes a single printed circuit board containing a microprocessor, memory, and minimal input/output(I/O) circuitry mounted. With the advent of increasingly powerful microprocessors, microcomputers became popular in the 1970s and1980s. The predecessors to these computers, mainframes, and minicomputers, were comparatively much larger and more expensive(though indeed present-day mainframes such as the IBM System z machines use one or more custom microprocessors as their CPUs). Also, we can mention that many microcomputers, in the generic sense, (when equipped with a keyboard and screen for input and output) are also personal computers.

Microprocessor

The processor on a single chip is called a Microprocessor which can process micro-instructions. Instructions in the form of 0sand 1s

are called micro-instructions. The microprocessor is the CPU part of a microcomputer, and it is also available as a single integrated circuit. Thus as main components, the microprocessor will have theControl Unit (CU) and the Arithmetic Logic Unit (ALU) of a microcomputer. An example is Intel 8085 microprocessor. In addition to the microprocessor features, a microcomputer will have the following additional features:

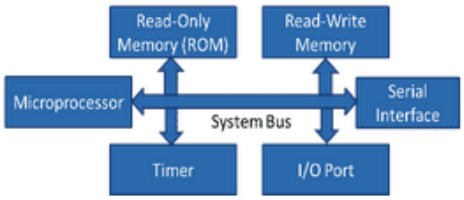
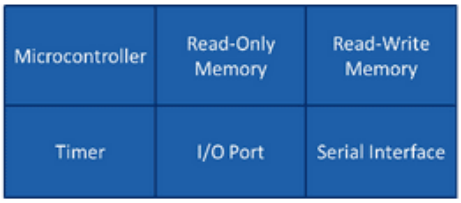
- ROM/PROM/EPROM/EEPROM for storing program;
- RAM for storing data, intermediate results, and final results;
- I/O devices for communication with the outside world;
- I/O ports for communication with the I/O devices.

In the present-day world, Microprocessors are extensively used. Before the microprocessor's invention, the logic design was done by hardware using gates, flip-flops, etc. A mini-computer was too much costly. With the advent of the microprocessor, logic design using hardware has been mostly replaced. It provides flexibility instrumentation where the characteristics of the system can be changed just by changing the software. Also, new generations of applications have surfaced, which were not thought of earlier because of the prohibitive cost of a minicomputer or the complexity of logic design using hardware.

Some of the applications where microprocessors have been used are listed below –

- Business applications such as desktop publishing;
- Industrial applications such as power plant control;
- Measuring instruments such as multimeter;
- Household equipment such as washing machine;
- Medical equipment such as blood pressure monitor;
- Defense equipment such as light combat aircraft;

- Computers such as a personal computer.

Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

1.2 Concept of Address bus, Data bus, Control bus & System Bus

SYSTEM BUSES

- Set of wires, that interconnects all the components (subsystems) of a computer
  - A source component sources out data onto the bus
  - A destination component inputs data from the bus
- May have a hierarchy of buses
  - Address, data and control buses to access memory and an I/O controller.
  - Second set of buses from I/O controller to attached devices/peripherals
  - Peripheral Component Interconnect(PCI) bus is an example of a very common local bus

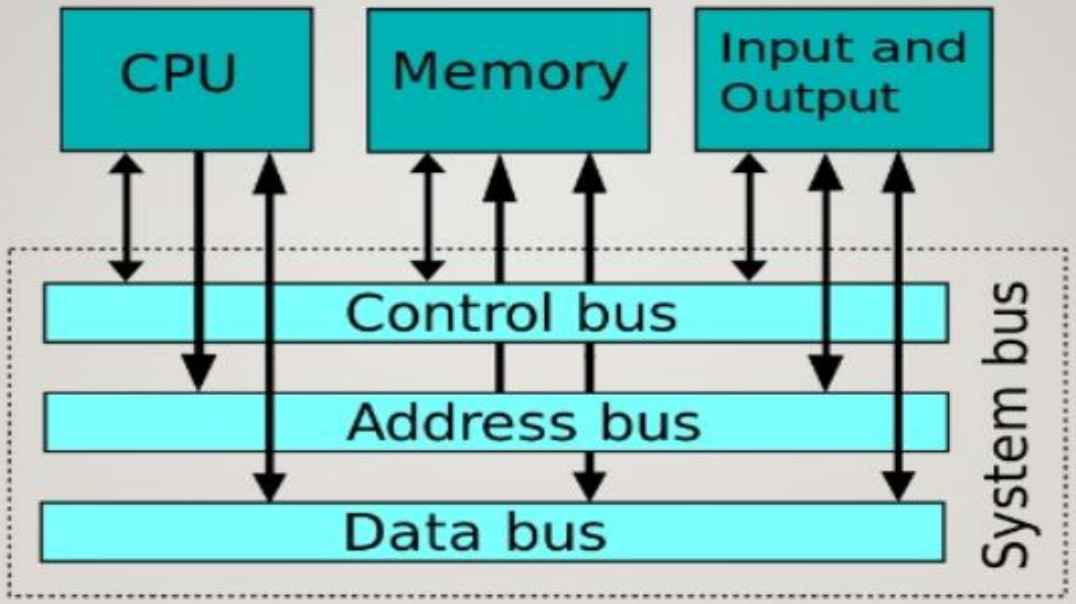


Fig: System Bus (Data, Address and Control Bus)

ADDRESS BUS

- It is a channel which transmits addresses of data (not the data) from the CPU to memory.
- The address bus consists of 16,24, or 32 parallel signal lines.
- The number of lines (wires) determines the amount of memory that can be directly addressed as each line carries one bit of the address.
- If the CPU has N address lines, then it can directly address 2^N address lines.
- For example, a computer with 32 bit address can address 4GB of physical memory.

- CPU reads/writes data from the memory by addressing a unique location; outputs the location of the data (aka address) on the address bus; memory uses this address to access the proper data.
- Each I/O device (such as monitor, keypad, etc) has a unique address as well (or a range of addresses); when accessing a I/O device, CPU places its address on the address bus. Each device will detect if it is its own address and act accordingly
- Devices always receive data from the CPU; CPU never reads the address buss (it is never addressed)

DATA BUS

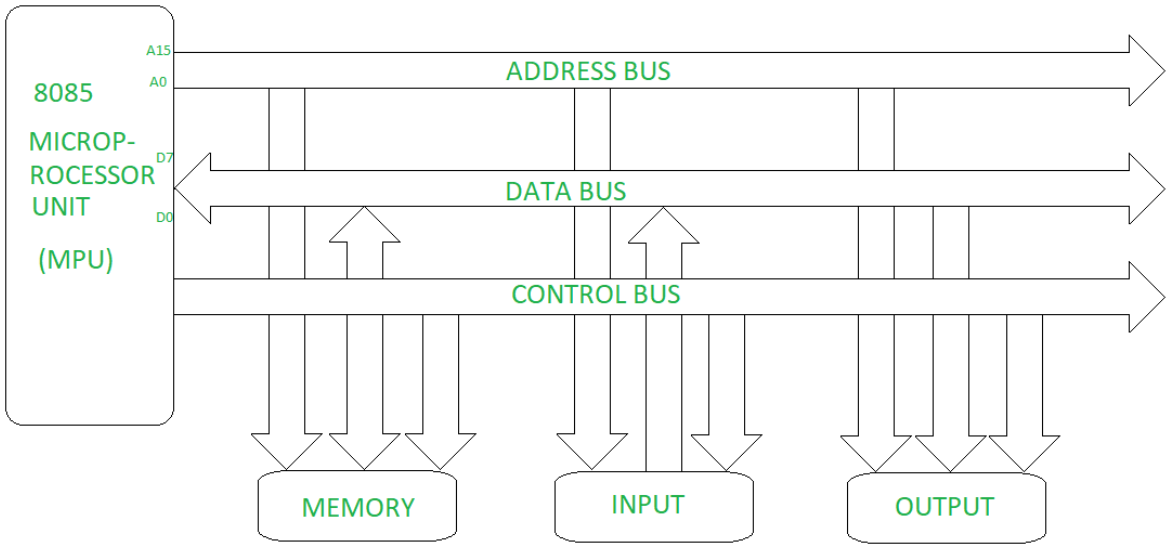
- Data bus is a channel across which **actual data are transferred** between the CPU, memory and I/O devices.
- The data bus consists of 8, 16, 32 or 64 parallel signal lines. Because each wire can transfer 1 bit of data at a time, an 8 wire bus can move 8 bits at a time which is a full byte.
- The **number of wires in the bus affects the speed** at which data can travel between hardware components. The wider the data bus, more data it can carry at one time.
- The data bus is **bidirectional** this means that the CPU can read data in from memory or it can send data out to memory.

- When the CPU fetches data from memory, it first outputs the address on the address bus, then the memory outputs the data onto the data bus; the CPU reads the data from data bus
- When writing data onto the memory, the CPU outputs first the address on the address bus, then outputs the data onto the output bus; memory then reads and stores the data at the proper location
- The process to read/write to a I/O device is similar

CONTROL BUS

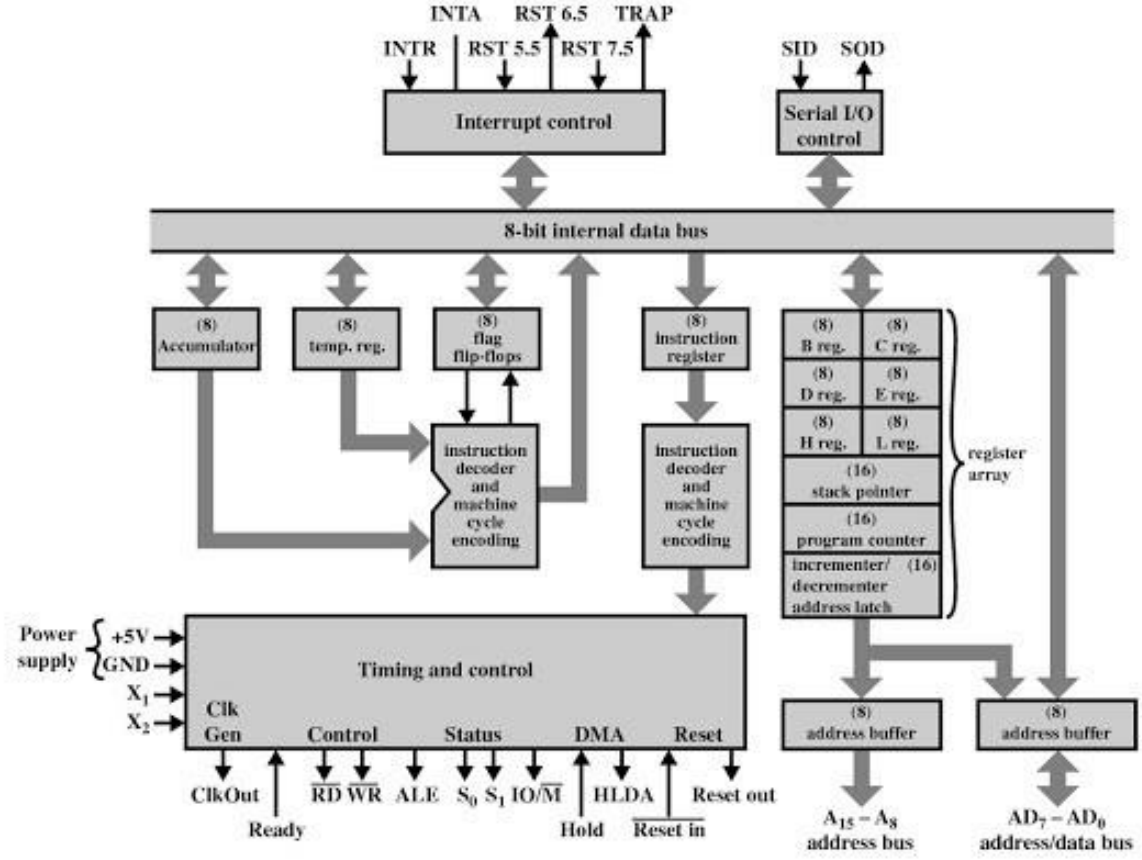
- The physical connections that carry **control information** between the CPU and other devices within the computer. This bus is mostly a collection of unidirectional signals.
- It is the **path for all timing and controlling functions** sent by the control units to other units of the system.
- It carries signals that **report the status of various devices**.
  - These signals indicate whether the data is to be read into or written out the CPU, whether the CPU is accessing memory or an IO device, and whether the I/O device or memory is ready for the data transfer
- For example, one line of the bus is used to indicate whether the CPU is currently reading from or writing to main memory. Others are *I/O Read/Write*

1.3 General Bus structure Block diagram



Bus organization system of 8085 Microprocessor

1.4 BASIC ARCHITECTURE OF 8085 (8 BIT) MICROPROCESSOR



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

It has the following configuration −

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

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

8085 Microprocessor – Functional Units

8085 consists of the following functional units −

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops −

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

Its bit position is shown in the following table −

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

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits –

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Serial Input/output control

It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

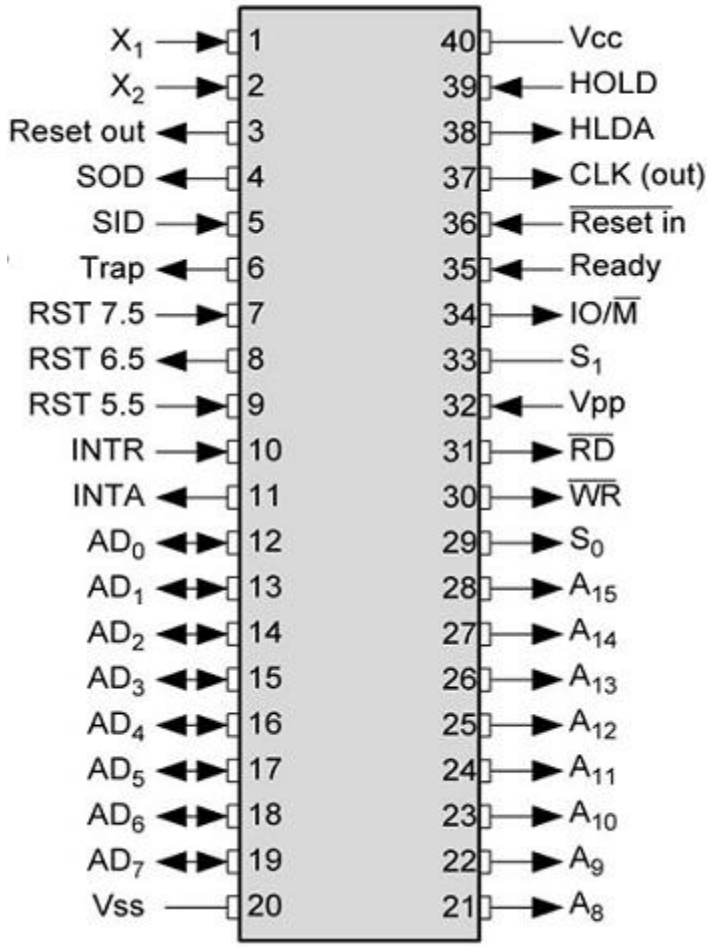
Address buffer and address-data buffer

The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

Address bus and data bus

Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

1.5 Signal Description (Pin diagram) of 8085 Microprocessor



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

Address bus

A15-A8, it carries the most significant 8-bits of memory/IO address.

Data bus

AD7-AD0, it carries the least significant 8-bit address and data bus.

Control and status signals

These signals are used to identify the nature of operation. There are 3 control signal and 3 status signals.

Three control signals are RD, WR & ALE.

- **RD** − This signal indicates that the selected IO or memory device is to be read and is ready for accepting data available on the data bus.
- **WR** − This signal indicates that the data on the data bus is to be written into a selected memory or IO location.

- **ALE** − It is a positive going pulse generated when a new operation is started by the microprocessor. When the pulse goes high, it indicates address. When the pulse goes down it indicates data.

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

IO/M

This signal is used to differentiate between IO and Memory operations, i.e. when it is high indicates IO operation and when it is low then it indicates memory operation.

S1 & S0

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

Power supply

There are 2 power supply signals − VCC & VSS. VCC indicates +5v power supply and VSS indicates ground signal.

Clock signals

There are 3 clock signals, i.e. X1, X2, CLK OUT.

- **X1, X2** − A crystal (RC, LC N/W) is connected at these two pins and is used to set frequency of the internal clock generator. This frequency is internally divided by 2.
- **CLK OUT** − This signal is used as the system clock for devices connected with the microprocessor.

Interrupts & externally initiated signals

Interrupts are the signals generated by external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. We will discuss interrupts in detail in interrupts section.

- **INTA** − It is an interrupt acknowledgment signal.
- **RESET IN** − This signal is used to reset the microprocessor by setting the program counter to zero.
- **RESET OUT** − This signal is used to reset all the connected devices when the microprocessor is reset.
- **READY** − This signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.
- **HOLD** − This signal indicates that another master is requesting the use of the address and data buses.

- **HLDA (HOLD Acknowledge)** − It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed.

Serial I/O signals

There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.

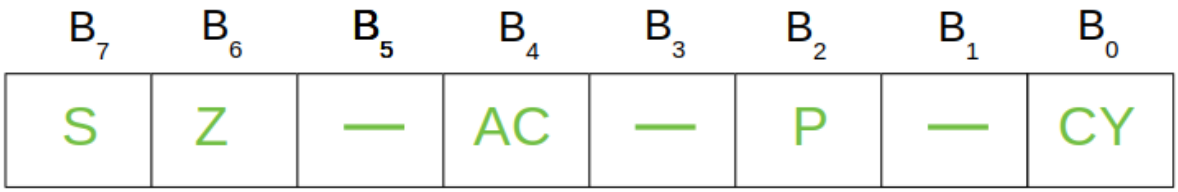
- **SOD** (Serial output data line) − The output SOD is set/reset as specified by the SIM instruction.
- **SID** (Serial input data line) − The data on this line is loaded into accumulator whenever a RIM instruction is executed.

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

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

(b) Specific Purpose Registers –

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



fig(a)-Bit position of various flags in flag registers of 8085

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

1. **Sign Flag:** It occupies the seventh bit of the flag register, which is also known as the most significant bit. It helps the programmer to know whether the number stored in the accumulator is positive or negative. If the sign flag is set, it means that number stored in the accumulator is negative, and if reset, then the number is positive.
2. **Zero Flag:**: It occupies the sixth bit of the flag register. It is set, when the operation performed in the ALU results in zero(all 8 bits are zero), otherwise it is reset. It helps in determining if two numbers are equal or not.
3. **Auxillary Carry Flag:** It occupies the fourth bit of the flag register. In an arithmetic operation, when a carry flag is generated by the third bit and passed on to the fourth bit, then Auxillary Carry flag is set. If not flag is reset. This flag is used internally for BCD(Binary-Coded decimal Number) operations.
   Note – This is the only flag register in 8085 which is not accessible by user.
4. **Parity Flag:** It occupies the second bit of the flag register. This flag tests for number of 1’s in the accumulator. If the accumulator holds even number of 1’s, then this flag is set and it is said to even parity. On the other hand if the number of 1’s is odd, then it is reset and it is said to be odd parity.
5. **Carry Flag:** It occupies the zeroth bit of the flag register. If the arithmetic operation results in a carry(if result is more than 8 bit), then Carry Flag is set; otherwise it is reset.

(c) Memory Registers –

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

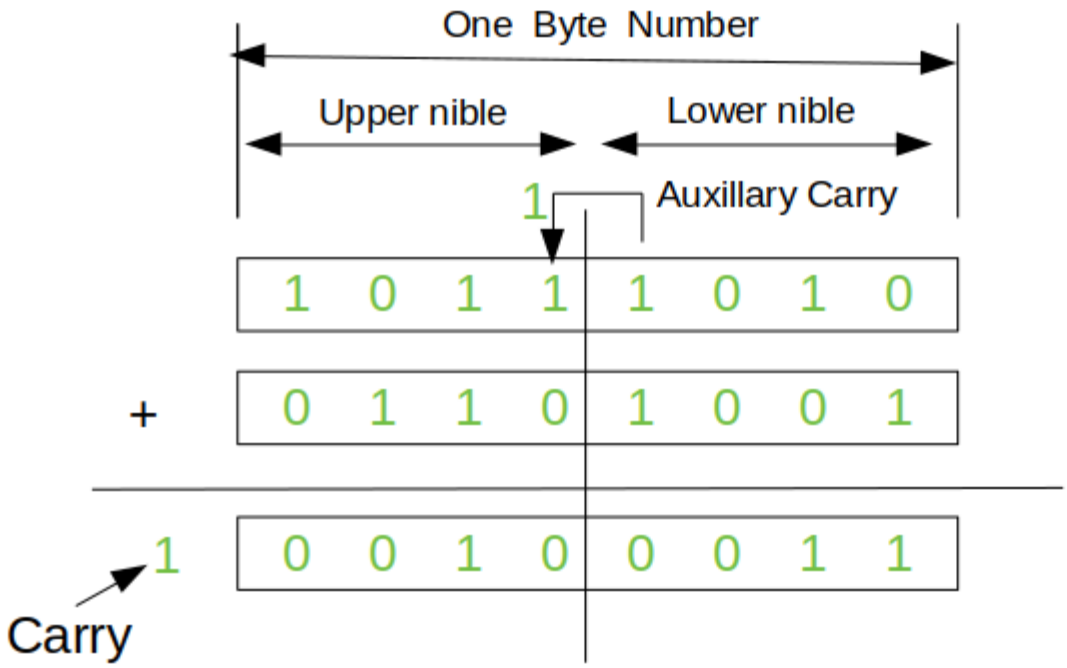
- **Program Counter:** This register is used to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being

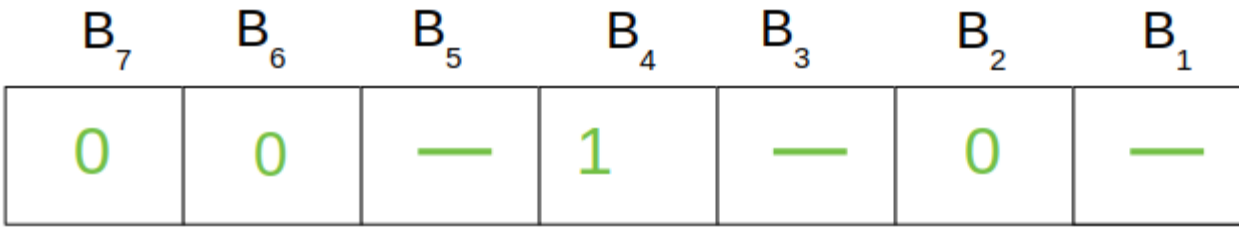
fetched, the program counter is incremented by one to point to the next memory location.

- **Stack Pointer:** It is used as a memory pointer. It points to a memory location in read/write memory, called the stack. It is always incremented/decremented by 2 during push and pop operation.

Example –

Here two binary numbers are added. The result produced is stored in the accumulator. Now lets check what each bit means. Refer to the below explanation simultaneously to connect them with the example.





Value of flags

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

Distinguish between SPR & GPR

Segment Registers:

• Segments are specific areas clear in a program for containing data, code and stack.

• There are 3 main segments − Code Segment − It contains all the instructions to be executed. A 16-bit Code Segment register or CS register supplies the starting address of the code segment.

General purpose registers:

• General purpose registers are used to store momentary data within the microprocessor.

• It is of sixteen bits and is divided into two eight-bit registers

1.7 Stack, Stack pointer &Stack top.

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

A stack pointer is a small register that stores the address of the last program request in a stack. A stack is a specialized buffer which stores data from the top down. As new requests come in, they "push down" the older ones. The most recently entered request always resides at the top of the stack, and the program always takes requests from the top.

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

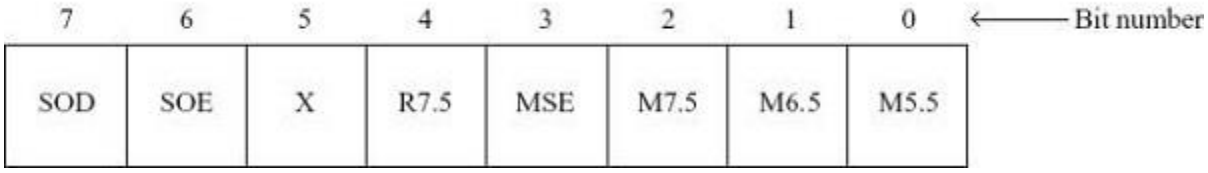
In 8085 Instruction set, **SIM** (Set Interrupt Mask) and **RIM** (Read Interrupt Mask) instructions can perform mask and unmask RST7.5, RST6.5, and RST5.5 interrupt pins and can also read their status.

In 8085 Instruction set, **SIM** stands for “Set Interrupt Mask”. It is 1-Byte instruction and it is a multi-purpose instruction. The main uses of **SIM** instruction are –

- Masking/unmasking of RST7.5, RST6.5, and RST5.5
- Reset to 0 RST7.5 flip-flop
- Perform serial output of data

Mnemonics, Operand	Opcode(in HEX)	Bytes
SIM	30	1

When SIM instruction is executed then the content of the Accumulator decides the action to be taken. So before executing the SIM instruction, it is mandatory to initialize Accumulator with the required value. The meaning and purpose of the various bits of the accumulator when SIM is executed has been depicted below –



Note that except bit 5, which is a don't care bit, the other bits of the Accumulator decide the effect of executing the SIM instruction. Masking of interrupts: Only the LS 4 bits of the accumulator are used for masking or unmasking of interrupts.

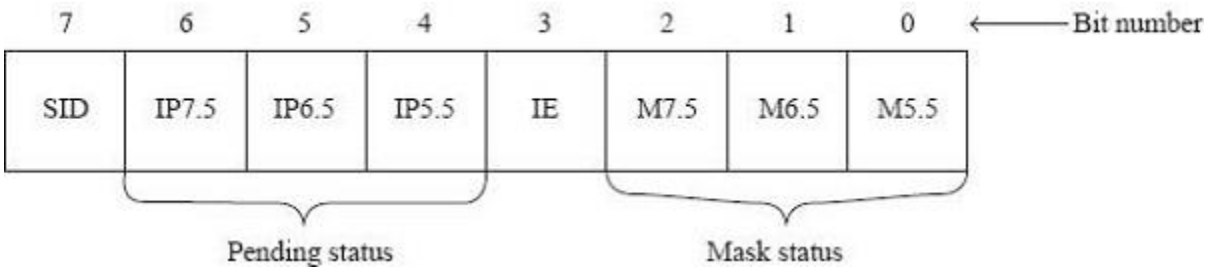
In 8085 Instruction set, **RIM** stands for “Read Interrupt Mask”. It is a 1-Byte multi-purpose instruction. It is used for the following purposes.

- To check whether RST7.5, RST6.5, and RST5.5 are masked or not;
- To check whether interrupts are enabled or not;
- To check whether RST7.5, RST6.5, or RST5.5 interrupts are pending or not;
- To perform serial input of data.

Mnemonics, Operand	Opcode(in HEX)	Bytes
RIM	20	1

To get the status information about the interrupt system, RIM instruction provides status information about interrupt system and this instruction can be used for serial input of data. Through this RIM instruction, 8085 can know which interrupt is masked or unmasked, etc. The contents of the Accumulator after the execution of the RIM instruction provide this information.

Thus, it is essential to look into the Accumulator contents after the RIM instruction is executed. The meaning of the various bits of the Accumulator after RIM is executed is shown in the following figure –



Mask status of interrupts: The LS 3 bits of the accumulator are used to provide mask status of interrupts. Note that they are not used for masking or unmasking. Masking or unmasking has to be done using the SIM instruction.

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

& 2.2 Addressing modes in instructions with suitable examples

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

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

- **Example-1:**
  Task- Copy the contents of accumulator in register B.
- **Mnemonic- MOV B, A**
- Opcode- MOV
- Operand- B, A
- Hex Code- 47H
- Binary code- 0100 0111

- **Example-2:**
  Task- Add the contents of accumulator to the contents of register B.
- **Mnemonic- ADD B**
- Opcode- ADD
- Operand- B
- Hex Code- 80H
  Binary code- 1000 0000

- **Example-3:**
  Task- Invert (complement) each bit in the accumulator.
- **Mnemonic- CMA**
- Opcode- CMA
- Operand- NA
- Hex Code- 2FH
  Binary code- 0010 1111

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

2. Two-byte instructions –

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

- **Example-1:**
  Task- Load the hexadecimal data 32H in the accumulator.

- **Mnemonic- MVI A, 32H**
- Opcode- MVI
- Operand- A, 32H
- Hex Code- 3E
- 32
- Binary code- 0011 1110
  0011 0010

- **Example-2:**
  Task- Load the hexadecimal data F2H in the register B.

- **Mnemonic- MVI B, F2H**
- Opcode- MVI
- Operand- B, F2H
- Hex Code- 06
- F2
- Binary code- 0000 0110
  1111 0010

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

3. Three-byte instructions –

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

- **Example-1:**
  Task- Load contents of memory 2050H in the accumulator.

- **Mnemonic- LDA 2050H**
- Opcode- LDA
- Operand- 2050H
- Hex Code- 3A
- 50
- 20

- Binary code- 0011 1010
- 0101 0000
  0010 0000
- **Example-2:**
  Task- Transfer the program sequence to the memory location 2050H.
- **Mnemonic- JMP 2085H**
- Opcode- JMP
- Operand- 2085H
- Hex Code- C3
- 85
- 20
- Binary code- 1100 0011
- 1000 0101
  0010 0000

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

- **Mnemonic- MOV B, A**
- Opcode- MOV
- Operand- B, A
- Hex Code- 47H
- Binary code- 0100 0111
- **Example-2:**
  Task- Add the contents of accumulator to the contents of register B.
- **Mnemonic- ADD B**
- Opcode- ADD
- Operand- B
- Hex Code- 80H
  Binary code- 1000 0000
- **Example-3:**
  Task- Invert (complement) each bit in the accumulator.
- **Mnemonic- CMA**
- Opcode- CMA
- Operand- NA
- Hex Code- 2FH
  Binary code- 0010 1111

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

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

- **Example-1:**
  Task- Load the hexadecimal data 32H in the accumulator.

- **Mnemonic- MVI A, 32H**
- Opcode- MVI
- Operand- A, 32H
- Hex Code- 3E
- 32
- Binary code- 0011 1110
  0011 0010

- **Example-2:**
  Task- Load the hexadecimal data F2H in the register B.

- **Mnemonic- MVI B, F2H**
- Opcode- MVI
- Operand- B, F2H
- Hex Code- 06
- F2
- Binary code- 0000 0110
  1111 0010

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

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

- **Example-1:**
  Task- Load contents of memory 2050H in the accumulator.

- **Mnemonic- LDA 2050H**
- Opcode- LDA
- Operand- 2050H
- Hex Code- 3A
- 50
- 20
- Binary code- 0011 1010
- 0101 0000
  0010 0000

- **Example-2:**
  Task- Transfer the program sequence to the memory location 2050H.
- **Mnemonic- JMP 2085H**
- Opcode- JMP
- Operand- 2085H
- Hex Code- C3
- 85
- 20
- Binary code- 1100 0011
- 1000 0101
  0010 0000

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

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

Data transfer instructions in 8085 microprocessor

Data tranfer instructions are the instructions which transfers data in the microprocessor. They are also called copy instructions.

Following is the table showing the list of logical instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MOV	Rd, Rs	Rd = Rs	MOV A, B
MOV	Rd, M	Rd = Mc	MOV A, 2050
MOV	M, Rs	M = Rs	MOV 2050, A
MVI	Rd, 8-bit data	Rd = 8-bit data	MVI A, 50
MVI	M, 8-bit data	M = 8-bit data	MVI 2050, 50

OPCODE	OPERAND	EXPLANATION	EXAMPLE
LDA	16-bit address	A = contents at address	LDA 2050
STA	16-bit address	contents at address = A	STA 2050
LHLD	16-bit address	directly loads at H & L registers	LHLD 2050
SHLD	16-bit address	directly stores from H & L registers	SHLD 2050
LXI	r.p., 16-bit data	loads the specified register pair with data	LXI H, 3050
LDAX	r.p.	indirectly loads at the accumulator A	LDAX H
STAX	16-bit address	indirectly stores from the accumulator A	STAX 2050
XCHG	none	exchanges H with D, and L with E	XCHG
PUSH	r.p.	pushes r.p. to the stack	PUSH H
POP	r.p.	pops the stack to r.p.	POP H
IN	8-bit port address	inputs contents of the specified port to A	IN 15
OUT	8-bit port address	outputs contents of A to the specified port	OUT 15

Following is the table showing the list of Arithmetic instructions with their meanings.

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
SBI	8-bit data	Subtract the immediate from the accumulator with borrow	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

Logical instructions in 8085 microprocessor

Logical instructions are the instructions which perform basic logical operations such as AND, OR, etc. In 8085 microprocessor, the destination operand is always the accumulator. Here logical operation works on a bitwise level.

Following is the table showing the list of logical instructions:

OPCODE	OPERAND	DESTINATION	EXAMPLE
ANA	R	A = A AND R	ANA B
ANA	M	A = A AND Mc	ANA 2050

OPCODE	OPERAND	DESTINATION	EXAMPLE
ANI	8-bit data	A = A AND 8-bit data	ANI 50
ORA	R	A = A OR R	ORA B
ORA	M	A = A OR Mc	ORA 2050
ORI	8-bit data	A = A OR 8-bit data	ORI 50
XRA	R	A = A XOR R	XRA B
XRA	M	A = A XOR Mc	XRA 2050
XRI	8-bit data	A = A XOR 8-bit data	XRI 50
CMA	none	A = 1’s compliment of A	CMA
CMP	R	Compares R with A and triggers the flag register	CMP B
CMP	M	Compares Mc with A and triggers the flag register	CMP 2050
CPI	8-bit data	Compares 8-bit data with A and triggers the flag register	CPI 50
RRC	none	Rotate accumulator right without carry	RRC
RLC	none	Rotate accumulator left without carry	RLC

OPCODE	OPERAND	DESTINATION	EXAMPLE
RAR	none	Rotate accumulator right with carry	RAR
RAL	none	Rotate accumulator left with carry	RAR
CMC	none	Compliments the carry flag	CMC
STC	none	Sets the carry flag	STC

Branching instructions in 8085 microprocessor

Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction.

The three types of branching instructions are:

1. Jump (unconditional and conditional)
2. Call (unconditional and conditional)
3. Return (unconditional and conditional)

1. Jump Instructions – The jump instruction transfers the program sequence to the memory address given in the operand based on the specified flag. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Jump Instructions: Transfers the program sequence to the described memory address.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JMP	address	Jumps to the address	JMP 2050

(b) Conditional Jump Instructions: Transfers the program sequence to the described memory address only if the condition in satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JC	address	Jumps to the address if carry flag is 1	JC 2050
JNC	address	Jumps to the address if carry flag is 0	JNC 2050
JZ	address	Jumps to the address if zero flag is 1	JZ 2050
JNZ	address	Jumps to the address if zero flag is 0	JNZ 2050
JPE	address	Jumps to the address if parity flag is 1	JPE 2050
JPO	address	Jumps to the address if parity flag is 0	JPO 2050
JM	address	Jumps to the address if sign flag is 1	JM 2050
JP	address	Jumps to the address if sign flag 0	JP 2050

2. Call Instructions – The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

(a) Unconditional Call Instructions: It transfers the program sequence to the memory address given in the operand.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
CALL	address	Unconditionally calls	CALL 2050

(b) Conditional Call Instructions: Only if the condition is satisfied, the instructions executes.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
CC	address	Call if carry flag is 1	CC 2050
CNC	address	Call if carry flag is 0	CNC 2050
CZ	address	Calls if zero flag is 1	CZ 2050
CNZ	address	Calls if zero flag is 0	CNZ 2050
CPE	address	Calls if parity flag is 1	CPE 2050
CPO	address	Calls if parity flag is 0	CPO 2050
CM	address	Calls if sign flag is 1	CM 2050
CP	address	Calls if sign flag is 0	CP 2050

3. Return Instructions – The return instruction transfers the program sequence from the subroutine to the calling program. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RET	none	Return from the subroutine unconditionally	RET

(b) Conditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program only is the condition is satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RC	none	Return from the subroutine if carry flag is 1	RC
RNC	none	Return from the subroutine if carry flag is 0	RNC
RZ	none	Return from the subroutine if zero flag is 1	RZ
RNZ	none	Return from the subroutine if zero flag is 0	RNZ
RPE	none	Return from the subroutine if parity flag is 1	RPE
RPO	none	Return from the subroutine if parity flag is 0	RPO
RM	none	Returns from the subroutine if sign flag is 1	RM
RP	none	Returns from the subroutine if sign flag is 0	RP

Stack I-O and Machine Control Instructions

The following instructions affect the Stack and/or Stack Pointer:

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

The I/0 instructions are as follows:

IN	Initiate Input Operation
OUT	Initiate Output Operation

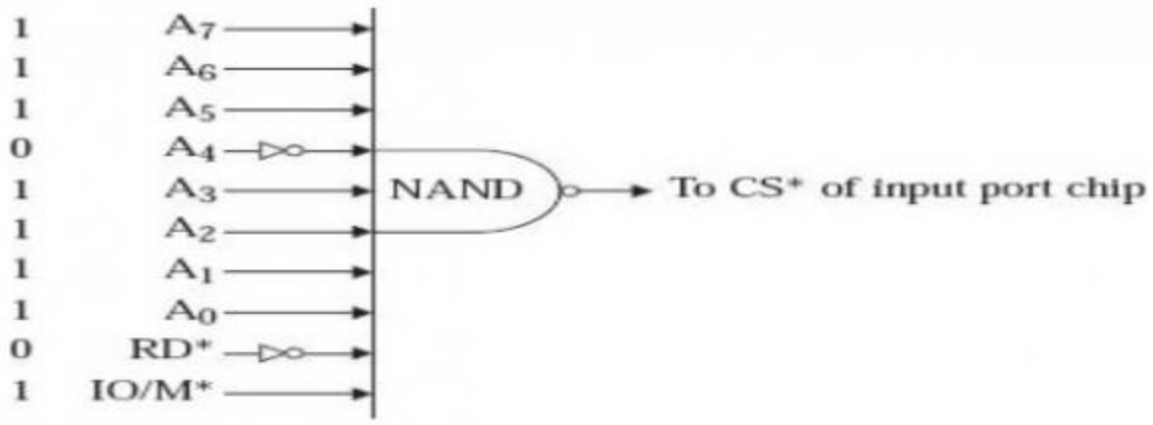
The Machine Control instructions are as follows:

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

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. This is shown in the following fig.

In this case, 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.

But 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. Let us discuss STA instruction in detail for better understanding.

Register A is an 8-bit register used in 8085 to perform arithmetic, logical, I/O & LOAD/STORE operations. Register A is quite often called as an Accumulator. An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (Central Processing Unit). In an arithmetic operation involving two operands, one operand has to be in this register. And the result of the arithmetic operation will be stored or accumulated in this register. Similarly, in a logical operation involving two operands, one operand has to be in the accumulator. Also, some other operations, like complementing and decimal adjustment, can be performed only on the accumulator.

Let us now consider a program segment which involves content of Accumulator only. In 8085 Instruction set, **STA** is a mnemonic that stands for STore Accumulator contents in memory. In this instruction, Accumulator 8-bit content will be stored to a memory location whose 16-bit address is indicated in the instruction as a16. This instruction uses absolute addressing for specifying the destination. This instruction occupies 3-Bytes of memory. First Byte is required for the opcode, and next successive 2-Bytes provide the 16-bit address divided into 8-bits each consecutively.

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

The content tracing of this instruction has been shown below

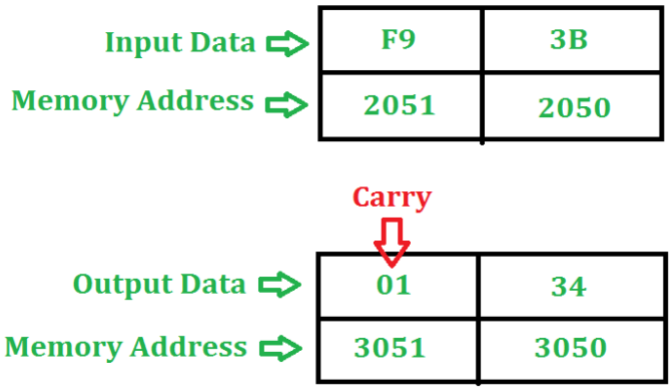
Address	Hex Codes	Mnemonic	Comment
2008	2A	STA 4050H	Content of the memory location 4050H A
2009	50		Low order Byte of the address
200A	40		High order Byte of the address

2.4 Simple Assembly Language Programming of 8085

8085 program to add two 8 bit numbers

Problem – Write an assembly language program to add two 8 bit numbers stored at address 2050 and address 2051 in 8085 microprocessor. The starting address of the program is taken as 2000.

Example –



Algorithm –

1. Load the first number from memory location 2050 to accumualtor.
2. Move the content of accumulator to register H.
3. Load the second number from memory location 2051 to accumaltor.
4. Then add the content of register H and accumulator using “ADD” instruction and storing result at 3050
5. The carry generated is recovered using “ADC” command and is stored at memory location 3051

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LDA 2050	A<-[2050]
2003	MOV H, A	H<-A
2004	LDA 2051	A<-[2051]

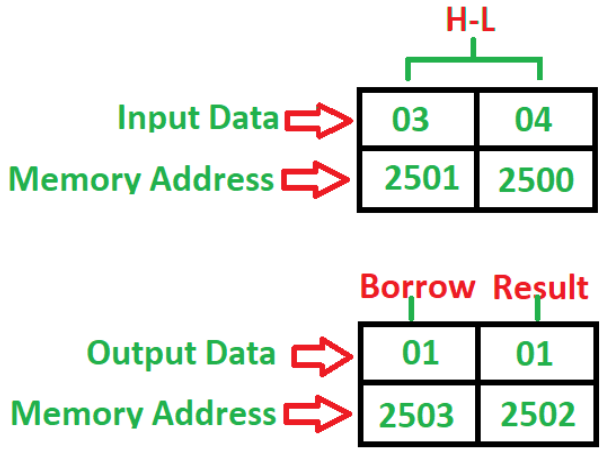
MEMORY ADDRESS	MNEMONICS	COMMENT
2007	ADD H	A<-A+H
2006	MOV L, A	L←A
2007	MVI A 00	A←00
2009	ADC A	A←A+A+carry
200A	MOV H, A	H←A
200B	SHLD 3050	H→3051, L→3050
200E	HLT	

Explanation –

1. **LDA 2050** moves the contents of 2050 memory location to the accumulator.
2. **MOV H, A** copies contents of Accumulator to register H to A
3. **LDA 2051** moves the contents of 2051 memory location to the accumulator.
4. **ADD H** adds contents of A (Accumulator) and H register (F9). The result is stored in A itself. **For all arithmetic instructions A is by default an operand and A stores the result as well**
5. **MOV L, A** copies contents of A (34) to L
6. **MVI A 00** moves immediate data (i.e., 00) to A
7. **ADC A** adds contents of A(00), contents of register specified (i.e A) and carry (1). As ADC is also an arithmetic operation, A is by default an operand and A stores the result as well
8. **MOV H, A** copies contents of A (01) to H
9. **SHLD 3050** moves the contents of L register (34) in 3050 memory location and contents of H register (01) in 3051 memory location
10. **HLT** stops executing the program and halts any further execution

8085 program to subtract two 8-bit numbers with or without borrow

Problem – Write a program to subtract two 8-bit numbers with or without borrow where first number is at **2500** memory address and second number is at **2501** memory address and store the result into **2502** and borrow into **2503** memory address.
Example –



Algorithm –

1. Load 00 in a register C (for borrow)
2. Load two 8-bit number from memory into registers
3. Move one number to accumulator
4. Subtract the second number with accumulator
5. If borrow is not equal to 1, go to step 7
6. Increment register for borrow by 1
7. Store accumulator content in memory
8. Move content of register into accumulator
9. Store content of accumulator in other memory location
10. Stop

Program –

MEMORY	MNEMONICS	OPERANDS	COMMENT
2000	MVI	C, 00	[C] <- 00
2002	LHLD	2500	[H-L] <- [2500]

MEMORY	MNEMONICS	OPERANDS	COMMENT
2005	MOV	A, H	[A] <- [H]
2006	SUB	L	[A] <- [A] – [L]
2007	JNC	200B	Jump If no borrow
200A	INR	C	[C] <- [C] + 1
200B	STA	2502	[A] -> [2502], Result
200E	MOV	A, C	[A] <- [C]
2010	STA	2503	[A] -> [2503], Borrow
2013	HLT		Stop

Explanation – Registers A, H, L, C are used for general purpose:

1. **MOV** is used to transfer the data from memory to accumulator (1 Byte)
2. **LHLD** is used to load register pair directly using 16-bit address (3 Byte instruction)
3. **MVI** is used to move data immediately into any of registers (2 Byte)
4. **STA** is used to store the content of accumulator into memory(3 Byte instruction)
5. **INR** is used to increase register by 1 (1 Byte instruction)
6. **JNC** is used to jump if no borrow (3 Byte instruction)
7. **SUB** is used to subtract two numbers where one number is in accumulator(1 Byte)
8. **HLT** is used to halt the program

2.4.2 Logic Operations (AND, OR, Complement 1's & 2's) & Masking of bits

logical operation is a special symbol or word that connects two or more phrases of information. It is most often used to test whether a certain relationship between the phrases is true or false.

In computing, logical operations are necessary because they can be used to model the way that information flows through electrical circuits, such as the circuits inside a CPU. These types of operations are called boolean operations.

The elements in a circuit which behave according to Boolean logic are called logic gates.

AND

The AND logic operation returns true only if either of its inputs are true. If either of the inputs is false, the output is also false.

In computer programming, the AND operation is usually written as **&&** (two ampersands).

In Boolean algebra, the AND operation of two inputs A and B can be written as **AB**.

Below is the truth table for an AND operation, and the circuit diagram of an AND logic gate.

AND		
A	B	AB
0	0	0
1	0	0
0	1	0
1	1	1

OR

The OR logic operation returns true if either of its inputs are true. If all inputs are false, the output is also false.

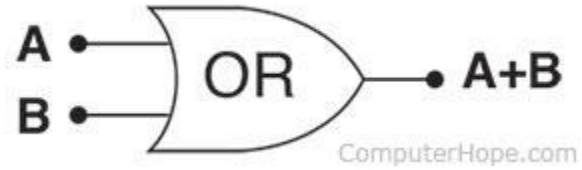
In computer programming, the OR operation is usually written as **||** (two vertical bars).

In Boolean algebra, the OR value of two inputs A and B can be written as **A+B**.

Note

Do not mistake the OR operation for arithmetic addition, even though they both use the "+" symbol. They are distinct operations.

Below is the truth table for an OR operation, and the circuit diagram of an OR logic gate.



OR		
A	B	A+B
0	0	0
1	0	1
0	1	1
1	1	1

Unit-3: TIMING DIAGRAMS.

CONCEPT OF TIMING DIAGRAM:-

An instruction is a command given to the computer to perform a specific operation. The sequence of instructions is called as a program. Program and data are stored in the CPU fetches one instruction from the memory at a time and executes it.

To fetch the instruction CPU constitutes an instruction cycle consists of a fetch cycle and execute cycle.

In fetch cycle the CPU fetch opcode from the memory. The necessary steps which are carried out to fetch an opcode from the memory, constitutes a fetch cycle.

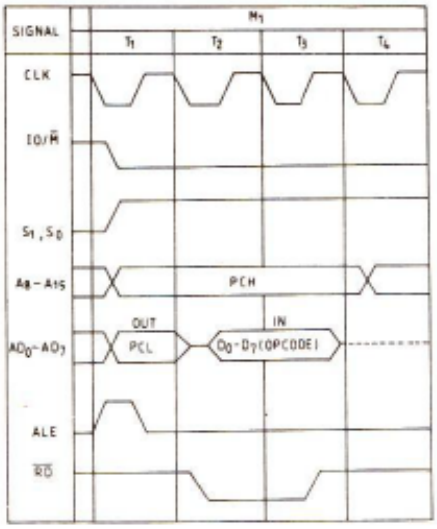
The necessary steps which are carried out to get data, if any from the memory and to perform the specific operation specified in an instruction, constitutes an execute cycle.

The necessary steps carried out tin a machine cycle can be represent graphically. The graphical representation of machine cycle is called as Timing Diagram.

The necessary steps carried out to perform the operation of accessing either memory or I/O device, constitute a machine cycle. In other words the necessary steps carried out to perform a fetch, a read or a write operation is called Machine cycle.

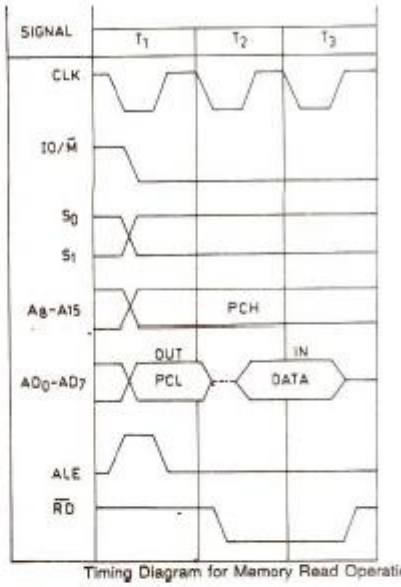
The subdivision of an operation performed in one clock cycle is called T-state. For fetch timing diagram the no. of T-state is 4 and if the data is in register pair, than the no. of T-state are 6.

1. Timing Diagram for Opcode fetches operation.



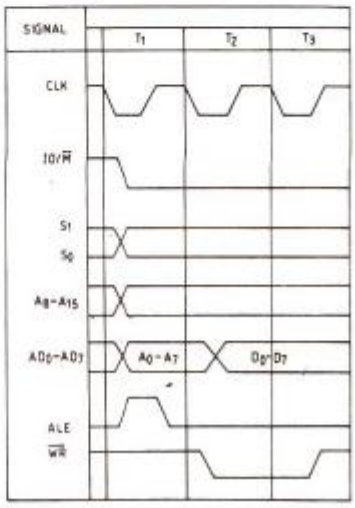
Timing Diagram for Opcode Fetch Operation.

2. Timing Diagram for Memory Read.



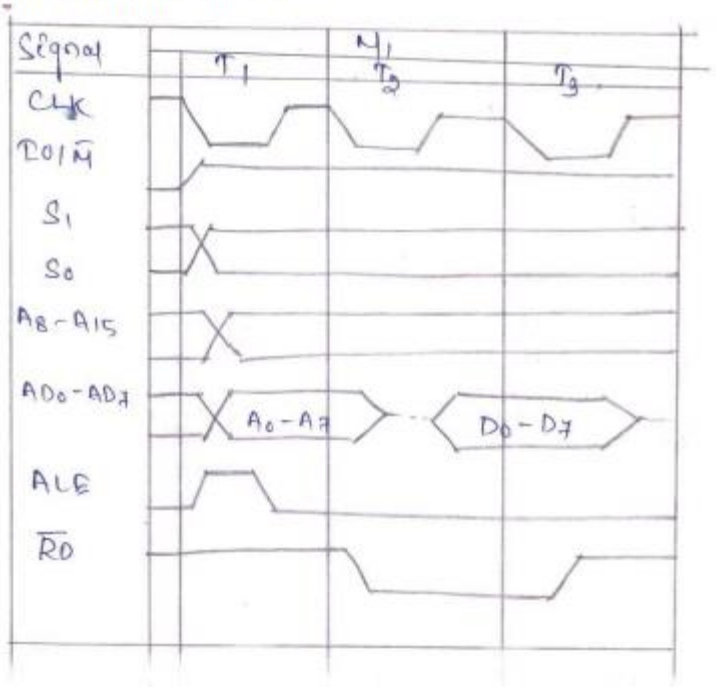
Timing Diagram for Memory Read Operation.

3. Timing Diagram for Memory Write.

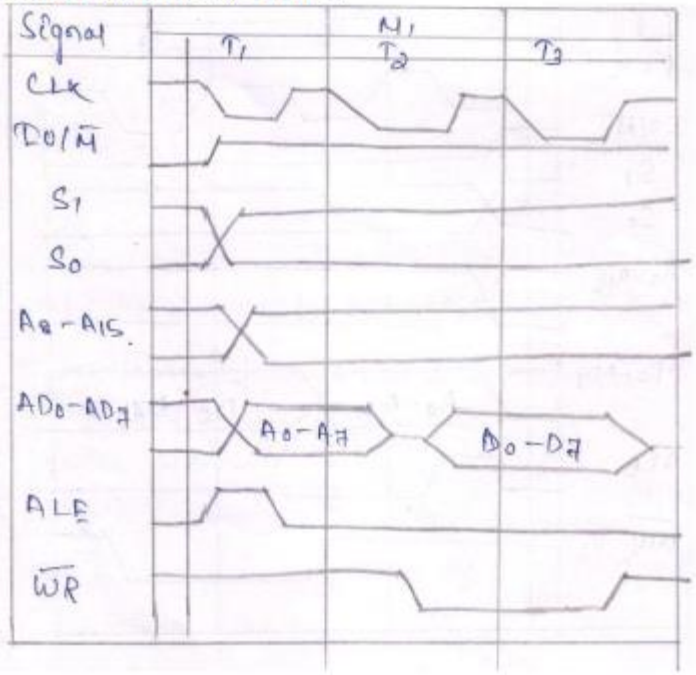


Timing Diagram for Memory Write Operation.

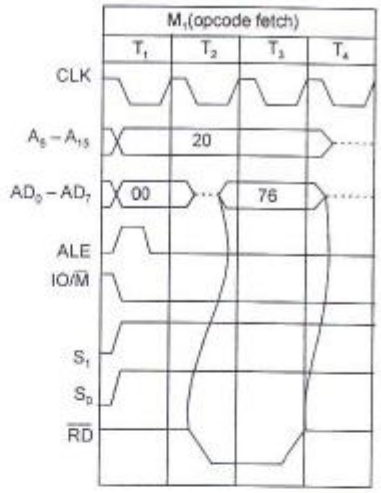
4. Timing Diagram for I/O Read.



5. Timing Diagram for I/O Write.

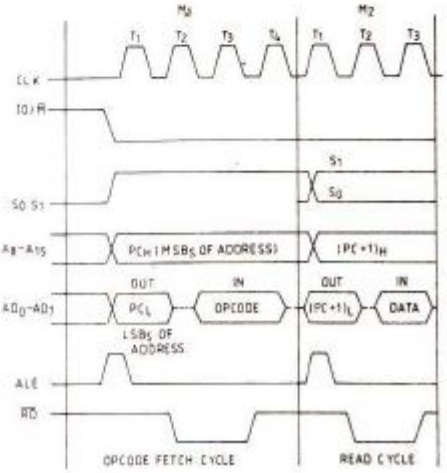


6. Timing Diagram for MOV A, B.



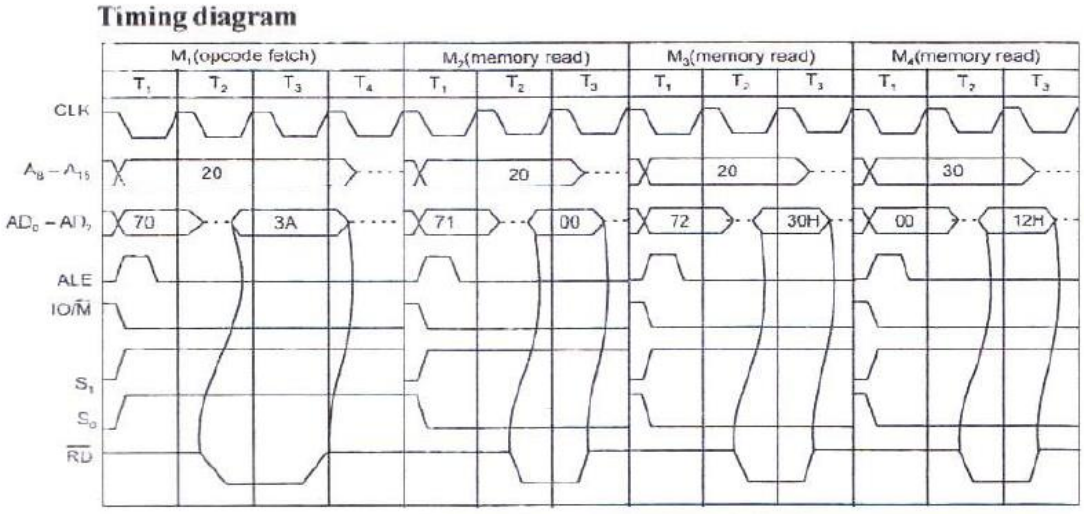
Timing diagram for MOV A, B

Timing Diagram for MVI r, Data.



Timing Diagram for MVI r, Data

8. Timing Diagram for LDA 3000H.



Timing diagram for LDA 3000H

INTERFACING I/O AND MEMORY PROGRAMMING

INTRODUCTION:

Some addresses are assigned to memories and some addresses to input device. Each memory location is assigned by an address.

Suppose that memory location are assigned that the address 2000 to 24FF. One address is assigned to one memory location. Any one of these address is not assigned to on input device.

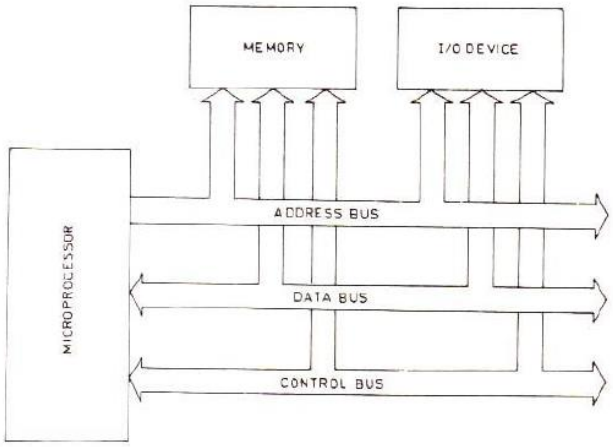
The address which is not assigned to memory is assigned to input devices. Ex: 2500, 2501, 2502, and 2503... etc. and one address is assigned to each input device. In this scheme all the data transform instruction of the microprocessor can be used for data memory as well as input devices. Ex: MOV A,M This is valid for data transfer from the memory to accumulator or form input device to accumulator. The memory mapped I/P scheme is suitable for a small system.

INPUT MAPPED I/P SCHEME:-

In this scheme the addresses assigned to memory location can also be assigned to input devices. Since the same address may be assigned to memory location or an input device, the microprocessor must issue a signal to distinguish whether the address on the Bus is for the memory location or I/O device. For this purpose Intel 8085 microprocessor issues on (IO/$\overline{M}$) signal for this purpose, when this signal is high it indicates that it is for I/O device, and when this signal is low, it indicates that it is for memory device and two extra instruction input and output are used to address I/O device. The "IN" instruction is used to read the data of an input device. The "'OUT" instruction is used to send the data to an output device. This scheme is suitable for a large system.

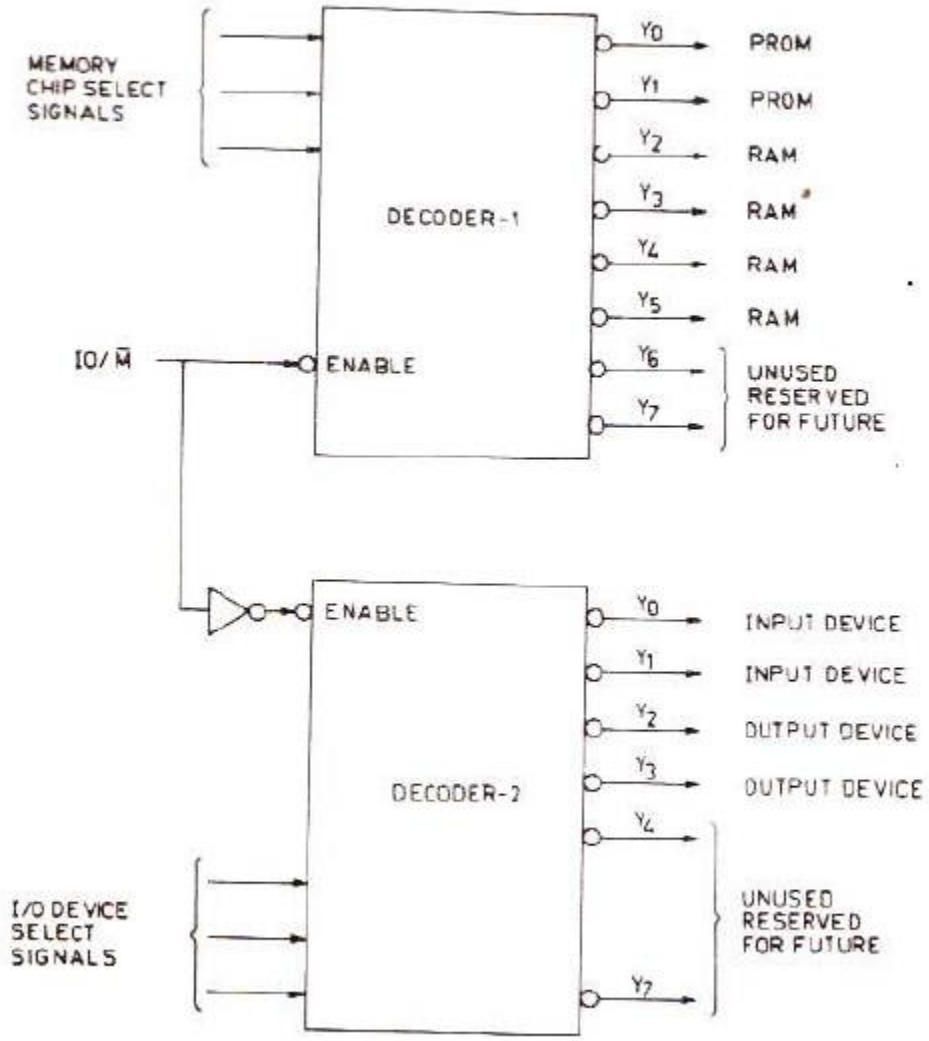
MEMORY AND I/O INTERFACING:

Several memory chips and I/O devices are connected to the microprocessor on address decoding circuit is employed to select the required I/O device or a memory chip.



Schematic Diagram for Memory and I/O Interfacing.

MEMORY INTERFACING:-

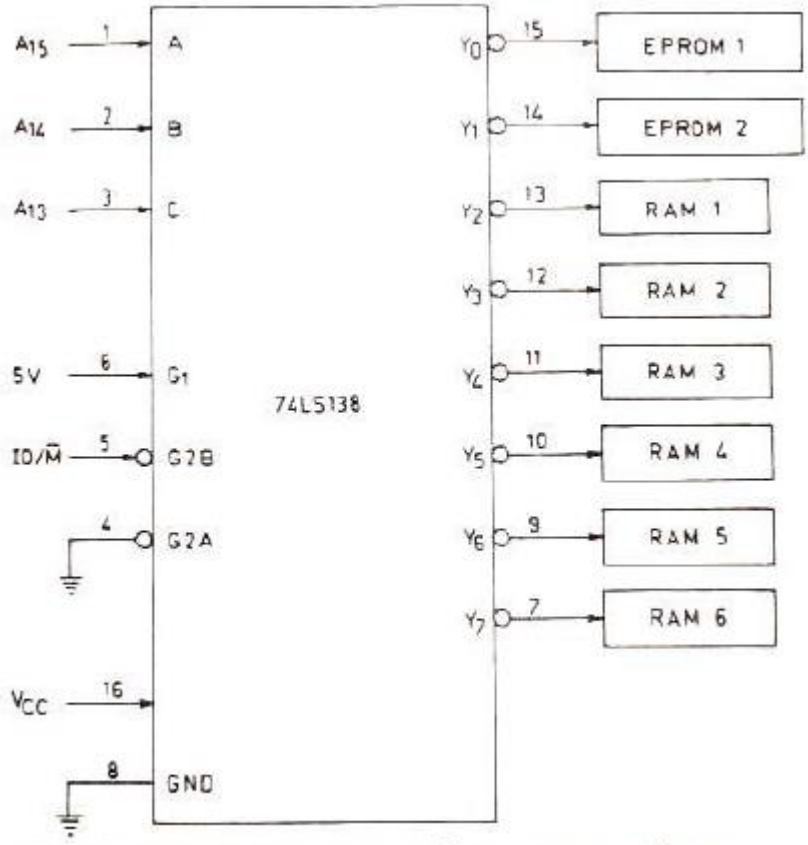


Interfacing of Memory and I/O Devices.

Table 7.1 Truth table for 74LS138

INPUTS						OUTPUTS							
ENABLE			SELECT										
G1	G2A	G2B	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
X	H	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

X Denotes irrelevant



Interfacing of Memory Chips using 74LS138.

Memory Locations for Various Zones

Decoder Output	*Memory Device*	*Zones of the Address Space*	*Memory Locations Address*
Y_0	EPROM 1	ZONE 0	0000 to 1FFF
Y_1	EPROM 2	ZONE 1	2000 to 3FFF
Y_2	RAM 1	ZONE 2	4000 to 5FFF
Y_3	RAM 2	ZONE 3	6000 to 7FFF
Y_4	RAM 3	ZONE 4	8000 to 9FFF
Y_5	RAM 4	ZONE 5	A000 to BFFF
Y_6	RAM 5	ZONE 6	C000 to DFFF
Y_7	RAM 6	ZONE 7	E000 to FFFF

Here to decoders are employed i.e. decoder 1 and decoder 2.If the (IO/$\overline{M}$) is high the decoder 2 is activated and required I/O device is selected. If (IO/$\overline{M}$) is low, decoder 1 is activated and required memory chip will be selected.

The address of a memory location or an I/O device is sent by the microprocessor, the corresponding memory chip or I/O device is selected by the decoding circuit the decoder task is performed by a bipolar PROM, PLA(Programmable logic array),Comparator etc. IN this section we are using an interface of memory chip through 74LS138.

G1, G2A and G2B are the enable signals A,B,C are the selected lines by Appling the proper logic to select a line one output can be selected. i.e. Y_0, Y1, Y2....Y_7. These are the 8 output lines. When it is selected it goes low.

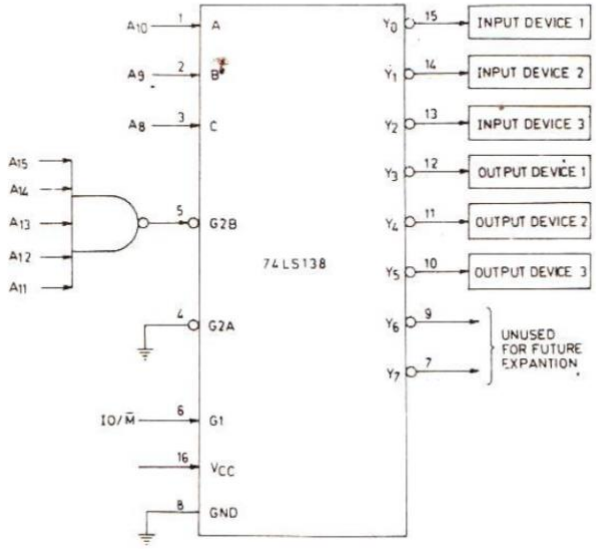
From the truth table when the G1 is low or G2A is high or G2B is high all output lines are high and similarly when G1 is high and G2A and G2B are low it act as a decoder.

The memory location for EPROM 1 will be lie in the range of 0000 to 1FFF, these are the memory location for ZONE 0 for the memory chip which is connected to the output line Y_0 of the decoder. Similarly for other ZONES.

The inter memory address (64kb for 8085) has been divided into 8 ZONES address lines *A*15, A14,A13 are applied to selected the lines A,B,C of the 74LS138. The logic applied to these select a particular memory device, an EPROM or a RAM other address

lines are A_0,A_1....A_{12} goes directly to the chip IO/$\bar{M}$ is connected to G2B when IO/M goes low it is to memory read or write operation. G1 is connected to +5V_{dc} is supply and G2A is grounded.

I/O INTERFACING:-



i Interfacing of I/O Devices Using 74LS138.

Address of I/O Devices connected to 74LS138

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	*Selected Output Lines*	*Corres-ponding Address*	*I/O Device*
1	1	1	1	1	0	0	0	Y_0	F8	Input Device 1
1	1	1	1	1	0	0	1	Y_1	F9	Input Device 2
1	1	1	1	1	0	1	0	Y_2	FA	Input Device 3
1	1	1	1	1	0	1	1	Y_3	FB	Output Device 1
1	1	1	1	1	1	0	0	Y_4	FC	Output Device 2
1	1	1	1	1	1	0	1	Y_5	FD	Output Device 3
1	1	1	1	1	1	1	0	Y_6	FE	Unused
1	1	1	1	1	1	1	1	Y_7	FF	Unused

This is the interface of I/O device through decoder 74LS138.

As the address of an I/O device is 8 bits so only A8-A15 lines of address Bus are used for I/O addressing.

The address lines A8, A9, A10 are used as select lines A, B and C of the decoder.

The address lines of A11 –A15 are applied to the G2B through NAND gate. When all the address lines (A11-A15) are high, the G2B is low.

IO/$\overline{M}$ is applied to G1 when it goes high and is indicates for I/O read or I/O write operation.

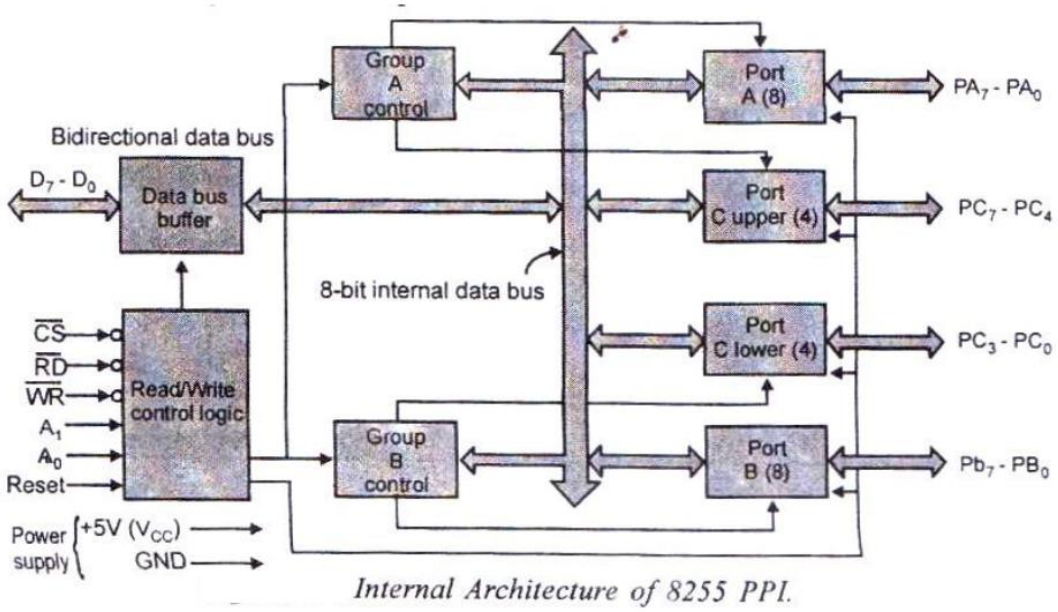
ARCHITECTURE OF 8255(PPI) :-

PPI= Programmable Peripheral Interface.

8255 PPI chip is also called as parallel input output port chip.

There are 24 I/P –O/P lines which can be individually programmed in two groups of 12 lines each these I/O lines are arranged as 3 ports i.e. port A, port B and port C. There are 2 individual groups of I/O pins called as group A and group B.

All the ports of 8255 can function independently as input or output by programming the bits of an internal register of 8255 called controlled word register (CWR).



Internal Architecture of 8255 PPI.

The functional block diagram contains

The functional block diagram contains

1. Data Bus buffer
2. Read/Write control logic
3. Group A and Group B control.
4. Port A, Port B, Port C.

DATA BUS BUFFER :-

It is 8 bit bidirectional data bus buffer it is used to interface the 8255 data bus with the system data bus.

Its outer pins are $D_0 - D_7$ and it is connected to system data bus. The direction of the data bus is to decided by the read/ write control signals. In read operation it transmits data to the system data bus and in write operation it receives data from the system data bus.

READ/ WRITE CONTROL LOGIC :

This block's function is to accept i/ps from system control bus and address bus.

Here the control signals like $\overline{RD}$,$\overline{WR}$, $\overline{CS}$ and address signals Ao, A1 are used.

Among these 5 signals $\overline{RD}$ and $\overline{WR}$ are connected to $\overline{IOR}$, $\overline{IOW}$ or $\overline{MEMR}$, $\overline{MEMW}$ depending upon the mapping Ao, A1 are directly connected to address lines Ao, A1 of the

---

system address lines. The selection of 8255 is enabled or disenabled by $\overline{CS}$ signal. If

$\overline{CS}$=0 the 8255 is selected and if $\overline{CS}$=1 8255 is rejected.

GROUP A AND GROUP B CONTROL:

8255 IS DIVIDED INTO 2 GROUPS I.E. Group-A and Group-B. Group-A consists of port A and port cupper and Group-B consists of Port B and port C lower. That means each group consists of 12 pins. The selection of ports are done by mode operation.

Mode Operation- Mode 0 - Simple I/O -(Port A,B,C)

Mode 1 - Stroved I/O -(Port A, B)

Mode 2 - Bidirectional Port - (Port C)

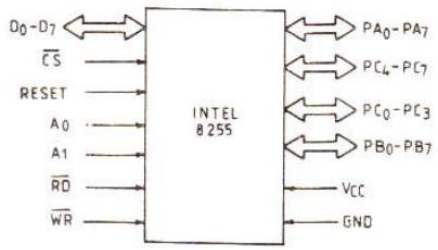
PORT A, PORT B AND PORT C :

The ports of 8255 PPI are Port A, Port B and Port C each port consists of an 8-bit data I/P buffer. The function of port A, Port B and Port C are decided by the control bit pattern of control word Register (CWR). The port C is divided into 2 groups PC upper and PC lower. Port C pins can be used as simple i/o, hand shake signals and status signals. It is used for coordination between port A and Port B.

A_1	Ao	Port/ Register Selection
0	0	Port A
0	1	Port B
1	0	Port C
1	1	CWR (Control Word Register)

PIN CONFIGURATION OF 8255 PPI CHIP :

Intel 8255 is a 40 Pin I.C. package. It operates on a single 5 vdc supply



Schematic Diagram of Intel 8255 A.

$D_0 - D_7$:

Pins = 27 to 34

Types= i/p - o/p

These are the 8-bit bidirectional data bus lines which are used to carry data or control word to / from the microprocessor.

$\overline{CS}$:

Pins = 6

Types = i/p

These are above low signals which is used to select 8255.

If $\overline{CS}$ = 0 =8255 is selected

$\overline{CS}$ = 1 =8255 is rejected

$\overline{RD}$:

Pin = 5

Types = i/p

It is a active low signal. It is used to send the data or status information to the mp on the data bus.

$\overline{WR}$:

Pin = 36

Types = I/P

This is also an active low signal. It is used to write data or control words into the 8255 PPI.

A1 and Ao :

Pin = 8 and 9

Types = I/P

It is used to select the different ports and control word Register (CWR).

The operation of $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ are :

$\overline{RD}$	$\overline{WR}$	A1	Ao	$\overline{CS}$	I/P operation (Read)
0	1	0	0	0	Port A to data bus
0	1	0	1	0	Port B to data bus
0	1	1	0	0	Port C to data bus
0	1	1	1	0	Not allowed

RESET :

Pin = 35

Types = I/P

This is an active high signal. It is used to reset the mp. After reset the control word register is cleared and all ports are to be set to i/p mode.

PAo - PA_7 :(Port A Pins)

Pins = 1 to 4and 37 to 40

Types = I/P or O/P

These are the 8-bit bidirectional pins used to send data to the device or to read data from device connected with the 8255 chip.

PBo – PB_7 :- (Port B Pins):

Pins = 18 - 25

Types = i/p or o/p

These are also 8-bit bidirectional pins and it is used to send data to the device or to read data from device connected with the 8255 chip.

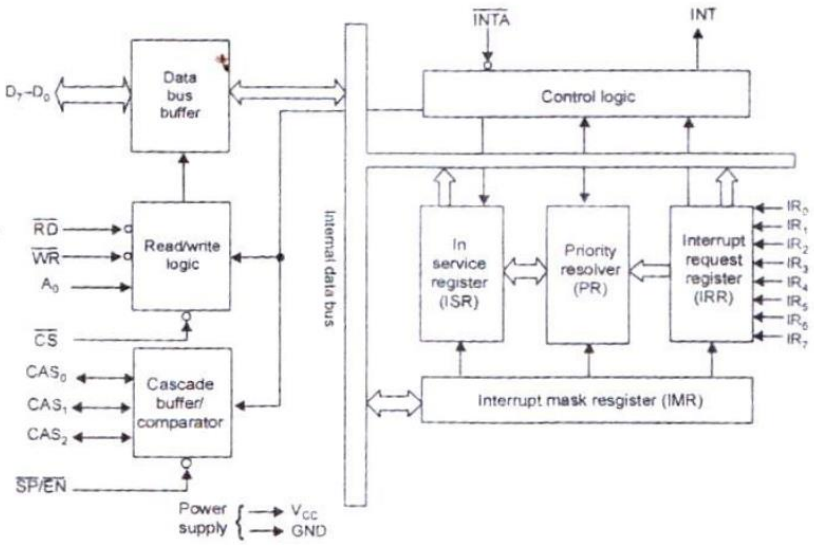
PCo – PC_7 : (Port-C Pins) :

Pins = 10 - 17

Types = I/P or O/P

These are the 8-bit bidirectional pins. These are divided to 2 sections i.e. (PC_4, PC_5, PC_6, PC_7 PC upper and PC Lower (i.e. PCo, PC_1, PC_2, PC_3)

FUNCTIONAL BLOCK DIAGRAM OF 8259 PIC (PROGRAMMABLE INTERRUPT CONTROLLER):



Architecture of 8259 PIC.

(ARCHITECTURE OF 8259 PIC)

It is a 28 pin programmable interrupt controller.

The 8259 PIC contains 8 blocks

(1) Data bus buffer
(2) Read / Write logic
(3) Control logic
(4) Interrupt request register (IRR)
(5) In service register (ISR)
(6) Priority resolves (PR)
(7) Interrupt Mask register (IMR)
(8) Cascade Buffer / Comparator

DATA BUS BUFFER:

It is a 8-bit bidirectional data bus is used to interface the 8259 PIC data bus with the system data bus.

Control words and status information are transferred through the data bus buffer. It is internally connected to the data bus and its outer pi8ns Do –D_7 are connected to the system data bus directly. The direction OP the data bus is decided by the read / write control signal.

When the read signal is activated, then it transmits data to the system and when the write signal is activated then it receives data from the system data bus. The reading and writing operation is achieved by IN and OUT mp instruction.

READ / WRITE LOGIC :

The block accepts i/p from the system control bus and address bus. The control signals are $\overline{RD}$ and $\overline{WR}$, $\overline{CS}$ and address signal Ao is used. For this 5 signal $\overline{RD}$ and $\overline{WR}$

are connected to $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{MEMW}$ depending upon the mapping. $\overline{RD}$ and $\overline{WR}$ decides the operation is to performed i.e. write data to the 8259 or read data from the

8259. Ao is directly connected to the address lines Ao of the system address lines $\overline{CS}$ is connected to the Chip select decoder. It means the selection of the 8259 is enabled or

disenabled by $\overline{CS}$ signal.

If $\overline{CS}$= 0 8259 is selected

$\overline{CS}$= 1 8259 is rejected.

CONTROL LOGIC :

This block has 2 pins $\overline{INTA}$ and INT as an i/p. $\overline{INTA}$ is connected to the interrupt

Pin of the mp when a valid interrupt is occurs it goes high $\overline{INTA}$ is a interrupt acknowledgement signal from the mp.

INTERRUPT REQUEST REGISTER (IRR):

The interrupts at the Interrupt request (IR) lines are handless by the interrupt request register internally. IRR is used to store all the interrupt levels which are requesting service in it in order to serve them one by one on the priority basis.

IN SERVICE REGISTER (ISR) :

ISR is used to store all the interrupt levels which are being serviced. Each bit of this register is set by the priority resolver and reset by the end of the interrupt command word. The mp can read the contents of this register by issuing appropriate command word.

PRIORITY RESOLVER (PR):

PR determines the priorities of the bits set in the IRR. To made decision the priority resolver looks at the ISR. If the highest priority bit in the ISR is set, then it ignores the new request. If the priority resolver finds that the new interrupt has a higher priority that the interrupt currently being serviced, then it will set all the appropriate bit in the ISR and send the INT signal to the mp for the new interrupt request.

INTERRUPT MASK REGISTER (IMR):

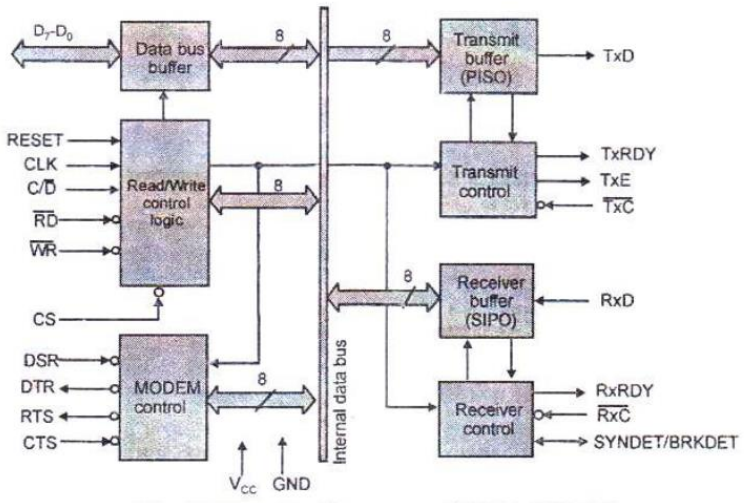
It is a programmable register. It is used to mask unwanted interrupt request by writing appropriate control word. The IMR operates on the IRR. Masking of a higher priority i/p will not affect the interrupt request lines of the lower priority. The mp can read contents of this register without issuing any command word.

CASCADE BUFFER / COMPARATOR :

This functional block stores and compares the identification number (IDS) of all 8259s used in the system. This associate 3 Pins i.e. CAS_2 – CASo. These are the O/P when the 8259 is used as a master and the I/p when the 8259 is used as a Slave. As a master, the 8259 sends the ID of the interrupting Salve device on to the CAS_2 –CASo lines. The salve thus selected will send its programmed subroutine address on to the

data bus during the next one or two consecutive $\overline{INTA}$ Pulses. In buffer mode, it generates an $\overline{EN}$ signal.

FUNCTIONAL BLOCK DIAGRAM / ARCHITECTURE OF 8251 USART:



Architecture of 8251 USART

DATA BUS BUFFER :

A 8-bit bidirectional data bus IS USED TO CONNECT 8251 usart DATA BUS TO THE SYSTEM DATA BUS. It is internally connected to the data bus and its outer pins Do-D7 are connected to the system data bus.

- For Read signal it transmits data and
- For write signal it receives data
- IN and OUT instructions are used for reading and writing operation and it is depends upon the read / write control logic.
- USART = Universal synchronous / Asynchronous receiver / transmitter.

READ / WRITE CONTROL LOGIC :

This device is used to control all the devices. After getting control signals from the control bus it generates control signals for device operation.

$\overline{CS}$ Pin is used to active 8251 USART. When $\overline{CS}$=0, 8251 USART is activated. The C/$\overline{D}$ (Control / Data) signal decides which part to active.

This device is used to control all the devices. After getting control signals from the control bus it generates control signals for device operation.

$\overline{CS}$ Pin is used to active 8251 USART. When $\overline{CS}$=0, 8251 USART is activated. The C/$\overline{D}$ (Control / Data) signal decides which part to active.

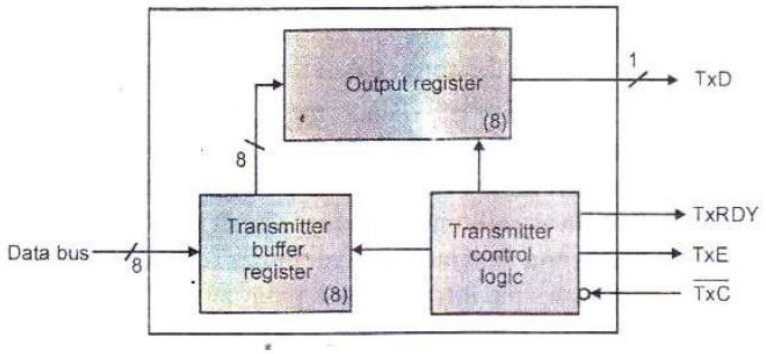
If C/$\overline{D}$=1, the control part is selected.

C/$\overline{D}$=0, the data part is selected.

The status of all signal are:

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	**Data transfer**
0	0	0	1	Receiver data register of 8251 to data bus
0	0	1	0	Data bus to 8251 transmitter data register.
0	1	0	1	Status word to data bus
0	1	1	0	Data bus to control word
1	X	X	X	Data bus tristated

TRANSMITTER SECTION:



The transmitting sections are having the signals:

TxD - Transmit Data

TxRDy Transmitter Ready

RxE Transmitter Empty

TxC Transmitter Clock

The Transmitting Section consists of transmit buffer, transmit control block and O/P register.

The transmitter Buffer register accepts data from the data bus buffer through internal data bus if Cs=0, CID =0, RD=1 and WR=0

The contents of the transmitter buffer are automatically transferred to the O/P register, if the O/P reg. is empty, the data is shifted serially on the TxD Pin, along with the appropriate bits are also added depending upon the mode selection i.e. synchronous (Synchronous Char. Are sent) or Asynchronous mode (Start and stop bits are sent).

If transmitter buffer reg. does not contain any data then TxRDY=1 and allow mp to sent next data for transmit control to indicate peripheral about in availability of data for transmission.

If TxRDY=1, it indicates transmitter buffer is empty. Here the data 1st cone to the transmit buffer reg. and then it transfers to the O/P register and finally from O/P reg it transfers one bit at a time on TDx Pin.

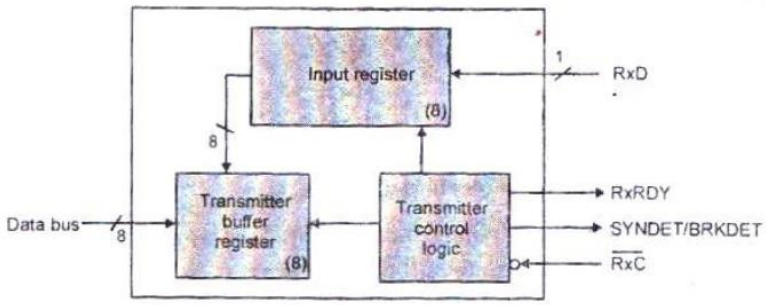
If TxRDY=0, it indicates that the transmitter buffer is not empty and contains some data when last data is transferred from mp to the transmit buffer register at the same time this last data is transferred from O/P register to the TxD Pin. It means the transmit buffer reg and O/P reg. are both empty, then it gives TxE=1

TxC is used to apply –ve edge clock pulses.

The condition for making TxE=1 are

(i) Transmit buffer reg. is empty
(ii) TxE=1
(iii) CTS=o
(iv) Upon master reset signal.

RECEIVER SECTION:-



This section contains the signals

RxD= Receive data

RXRDY= Receiver ready

SYNDET/BRKDET=Synchronous detect / break detect

$\overline{RxC}$= Receiver Clock

This section consists of Receiver buffer Register, receiver control logic and i/p reg. The function of i/p reg. is to accept the serial data through RxD. The function of i/p reg. is to convert the serial data to parallel from.

In synchronous mode it check the start bit and if the start bit is detected then the bit after start are converted to parallel form and then transferred to receiver buffer reg.

In asynchronous mode the i/p reg. will accept the data and converts it to parallel form and load into receiver buffer reg. if the SYNDET signal is 1.

In asynchronous mode after the data bits are accepted receiver checks programmed parity bit. If both are not same then a error signal is generated.

After the data bytes are transferred from i/p reg. to receiver buffer reg. the control logic generates a signal RxRDY to signal the mp about availability of data bytes to read by mp.

MODEM CONTROL :

Telephone lines are used to send the data over long distance. The telephone lines are analog in nature, so MODEMs (Modulator-Demodulator) are used to convert datas. To communicate MOIDEMs with 8251, the 8251 USART provides a block called MODEM control. The various signals under this unit are $\overline{RTS}$, $\overline{CTS}$, $\overline{DTR}$ and $\overline{DSR}$.

$\overline{DTR}$ (DATA TERMINAL READY):-

After the terminal is mode ON, different operation will be performed. At the time of data transmission / reception it generates a signal DTR to indicate its readiness for data transfer.

$\overline{DSR}$ (DATA SET READY):

This i/p is used as a general purpose one bit inverting i/p port. Its status can be checked by mp using a status read operation. This bit is used to check, if the data set is ready while communicating with a modem.

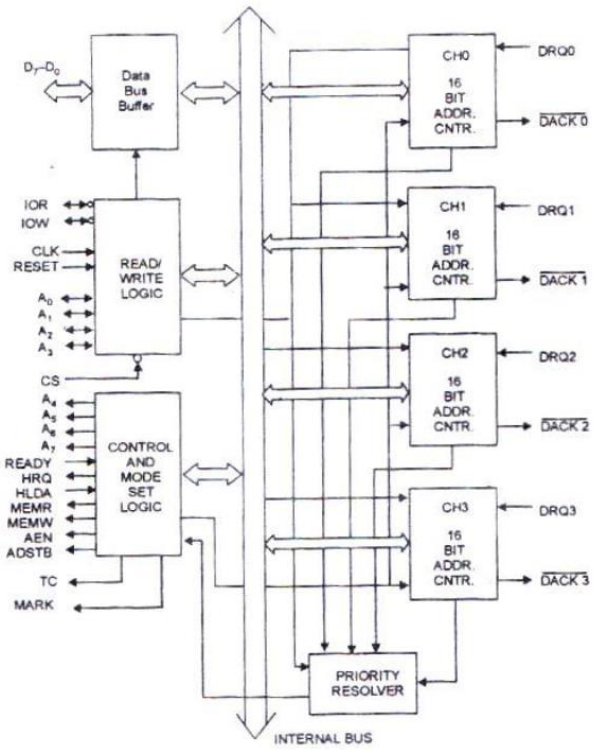
$\overline{CTS}$ (CLEAR TO SEND):

When MODEM is ready to transmit data it makes $\overline{CTS}$ signal low . The terminal after receiving $\overline{CTS}$ signal sends serial data character to the MODEM.

$\overline{RTS}$ (REQUEST TO SEND) :

Before the terminal is ready to transmit data and has a data character to be transmitted the terminal makes $\overline{RTS}$ signal low.

FUNCTIONAL BLOCK DIAGRAM / ARCHITECTURE DIAGRAM OF 8257 DMA CONTROLLER:



The 8257 supports 4 DMA channels. That means 4 peripheral devices can request for DMA data transfer through these channels at a time.

It has

(1) 8-bit internal Data buffer
(2) A read / write unit
(3) A Control unit and
(4) Priority Resolving unit along with a set of registers.

REGISTER ORGANISATION OF 8257:

The 8257 has 4 independent DMA channels . Each channel has a pair of 2 16-bit registers:

There are 2 common registers for all the channels i.e. mode set registers and status register. The address lines Ao-A3 are used to select one of the registers.

DMA ADDRESS REGISTERS:

Every DMA channel has one DMA address register. The Primary function this register is to store the address of the starting memory location which is access by the DMA channel.

TERMINAL COUNT REGISTERS:

Each channel of 8257 DMA has one terminal count register (TC) . It is a 16 bit register. It ascertains that the data transfer through a DMA channel stops after the required number of DMA cycles.

It means this register should be appropriately written before the actual DMA operation starts.

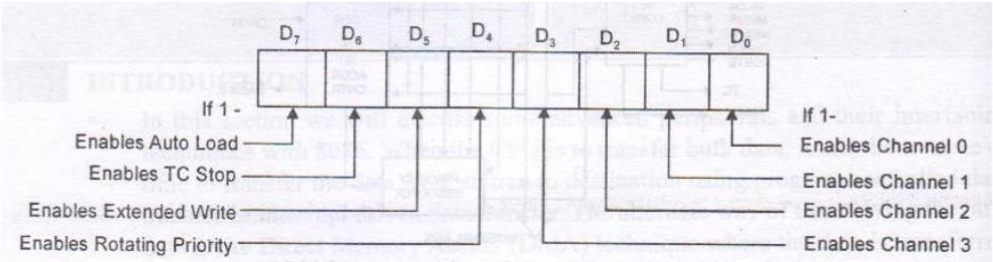
This terminal count register are initialized with the binary equivalent of the no. of required DMA cycles minus 1

After each DMA operation the terminal count register content will be decremented by one and finally it becomes zero after required no of DMA cycles are evered.

MODE SET REGISTERS:

As per the requirements of system the mode set register (MSR) is used for programming the 8257.

The Format of MSR is



Bit Definitions of the Mode Set Register.

Do - D3 Used to enable the 4 DMA channels of 8257

D6 If IC bit stops then the selected channel is disenabled. The IC stop bit is zero.

D4	It is set when the rotating priority is enables. Otherwise fixed priority is enabled.
D7	If it set, enables channel 2 for repeat block chaining operation.
D5	If it set, then the duration of $\overline{MEMW}$ and $\overline{IOW}$ signal is activated.

ADSTB (ADDRESS STOBE):

It is the higher byte of the memory address generated by the DMA controller in to the Latches.

AEN (ADDRESS LATCH ENABLE):-This o/p is used to disable the system data bus and control bus given by the CPU.

TC (TERMINAL COUNT): It indicates to the currently selected peripheral.

MARK (The Modulo-128): It indicates to the selected peripheral that the current DMA cycle is the 128th cycle since the previous MARK O/P . It will be activated after 128th cycles.

PROGRAMMABLE DMA CONTROLLER:

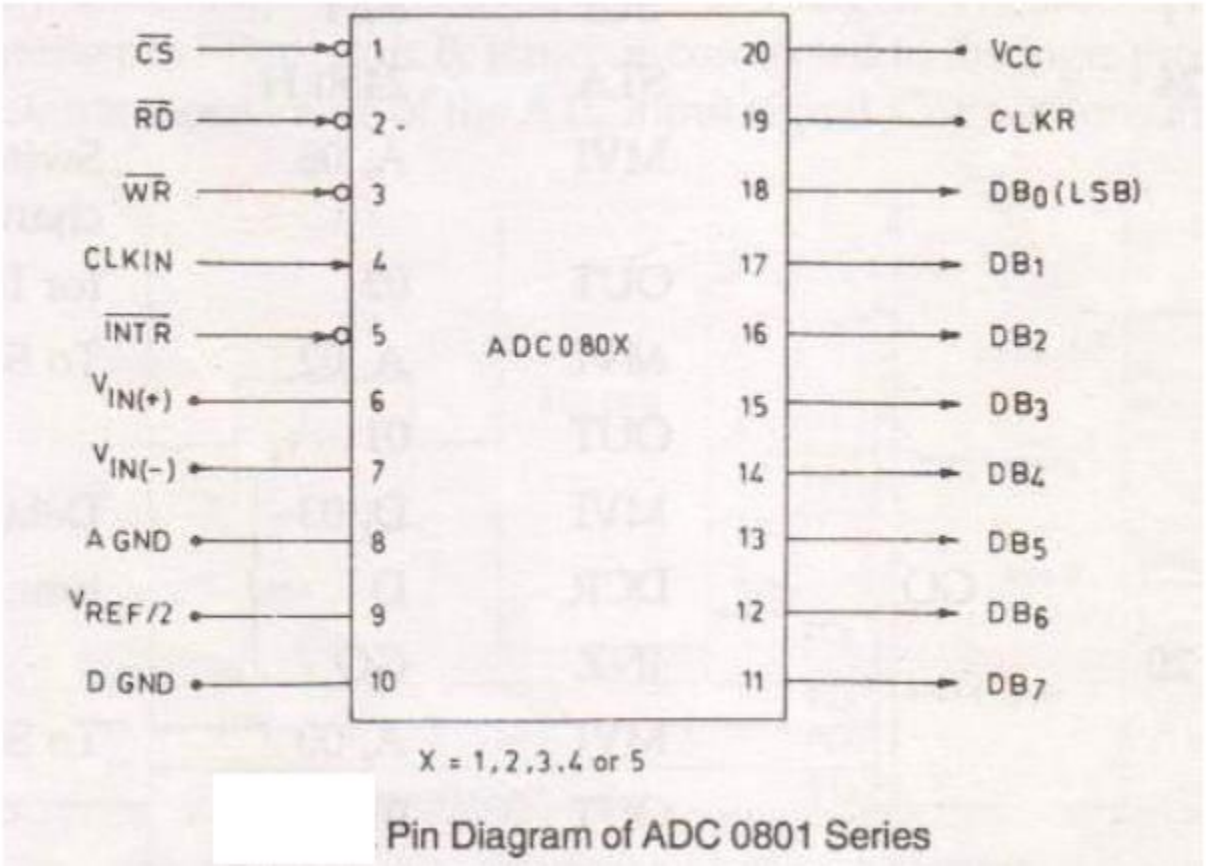
The bulk data transfer from I/O devices to memory or from memory to I/O devices through the accumulator is a time consuming process. For this process Direct Memory Access (DMA) technique is preferred.

In DMA data transfer scheme, data are directly transferred from an I/O device to RAM or from RAM to I/o device.

OPERATING PRINCIPLE OF ADC 0801 SERIES:-

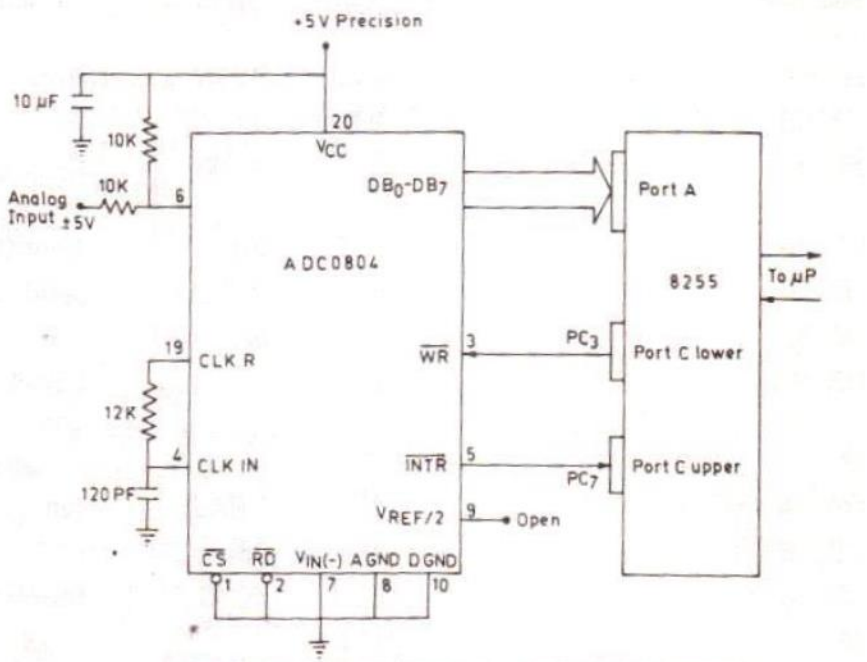
It's a 8-bit successive approximation A/D converters. Its important features are: on-chip clock generator, differential analog voltage input and no requirement of zero-adjustment. To start interfacing the $\overline{CS}$ should be low. The range of clock frequency is 100KHZ to 800KHZ.The clock frequency is: F=1/(1.1 RC).

A typically range of R=10KΩ to 50KΩ. Corresponding to R=12kΩ and C=120pf, the clock frequency=632KHZ.



Pin Diagram of ADC 0801 Series

INTERFACING OF ADC:



Interfacing of ADC 0804 for ± 5 V Analog Input Voltage.

It is a interfacing circuit of ADC 0808/0809 to intel 8085 microprocessor. An a input is connected to IN3. 5Vdc is applied to pin no 12 i.e.(ref +ve).It should not be from a stabilized power unit. The pin no 9 and 11 are connected to the 5Vdc stab power unit.

PROGRAM:

Mnemonics	Operands	Comments
MVI	A,98H	Initialize I/O ports of 8255
OUT	0B	
MVI	A,03	Switch ON multiplexer channel IN3
OUT	0A	
MVI	A,0B	Start of conversion pulse without affecting multiplexer's channel.
OUT	0A	
MVI	A,03	
OUT	0A	
IN	0A	Read E/C signal.
RAL		Rotate accumulator left.
JNC	READ	Is conversion over? No,

IN	08	jump to READ. Read digital o/p of A/D converter.
STA	FC50H	Store the result.
HLT		Stop.

OPERATING PRINCIPLE OF DAC:

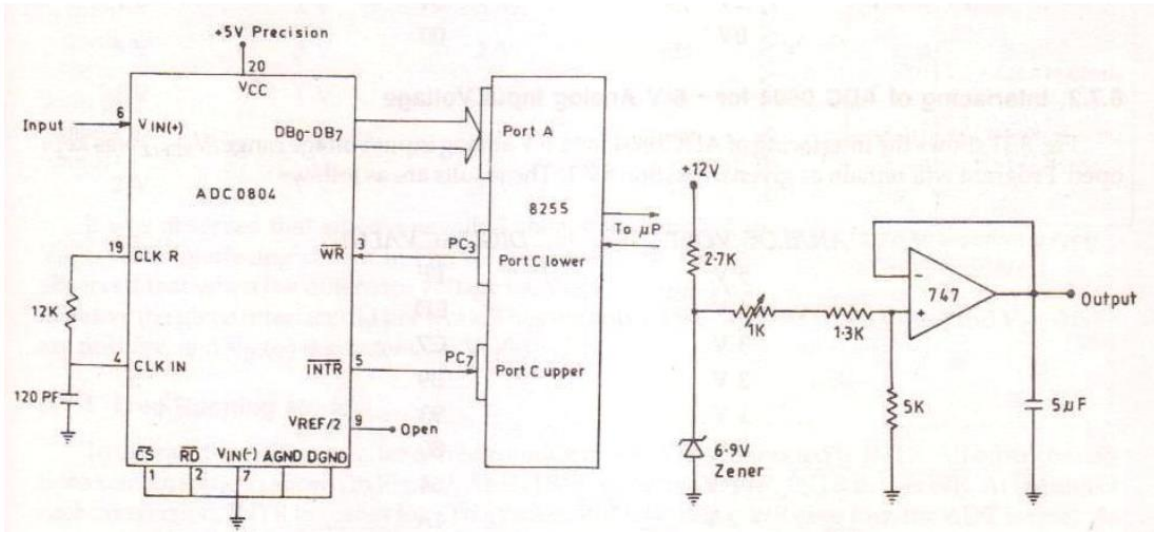
A DAC contains a ladder n/w. The n/w has i/p for binary bits of the digital word. When the MSB=1, it produces an o/p current, $(I_{REF})/4$. The bit next to the MSB produces $(I_{REF})/4$ and so on. It produces an o/p current or voltage proportional to the magnitude

$I_{out} = I_{REF} (1/2\ B_{n-1} + 1/4\ B_{n-2} + \ldots + 1/2^n B_0)$

Where B_0, B_{n-1}, B_{n-2}.... Are the binary bits of digital word applied to DAC.

$I_{REF} = (V_{ref})/R$ Where R=2.5K

INTERFACING OF DAC:



The DAC 0800 is a simple monolithic 8-bit D/A converter. It has fast settling time,100ns. It can directly interface to TTL,CMOS,PMOS and others. It operates at4.5V to +18V supply. The supply V^+ may be either 5V or +12V. V^- is kept -12V being easily available on standard power supply unit.

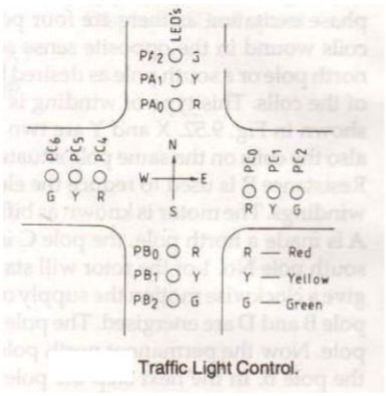
PROGRAM:

Mnemonics	Operands	Comments
MVI	A,98H	Get the control word for 8255
OUT	0B	Initialize ports.
MVI	A,80	Get 80 for digital i/p to DAC.
OUT	09	I/P 80to DAC through port B
HLT		Stop.

For bipolar operation the following modification in the circuit has to be done.

Connect pin 2 of DAC to non-inverting terminal of Op-amp. This common point is earthed through a 5KΩ resistor. The connection of pin4 of DAC and inverting terminal Op-amp will remain as before.

INTERFACING OF TRAFFIC LIGHT CONTROL SYSTEM USING 8255:



Traffic Light Control.

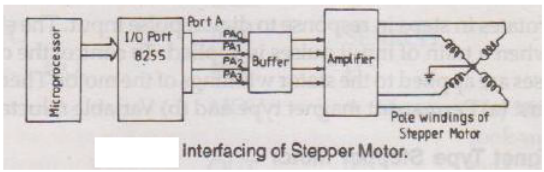
All the ports of 8255 have been programmed as O/P ports. The control word to make all the ports o/p ports in mode 0 operation is 80H. The connection of pins of the ports to LED have been made through buffer(7407).+ve logic have been used to switch on LEDs. Three types of LEDs have been used to switch on LEDs.

3 types of LEDs are Yellow, Red, Green.
Yellow to make alert
Red does not allow crossing.
Green allow crossing.

PROGRAM:

Mnemonics	Operands	Comments
MVI	A,80H	Get control word for 8255
OUT	0B	Initialize ports of 8255
MVI	A,01	
OUT	09	Red ON for South
OUT	08	Red ON for North
MVI	A,44	Green ON for East and West.
OUT	0A	
CALL	DELAY 1	
MVI	A,22	Yellow ON for East and West.
OUT	0A	
MVI	A,02	
OUT	09	Yellow ON for South
OUT	08	Yellow ON for North
CALL	DALAY 2	
MVI	A,11	Red ON for East and West
OUT	0A	
MVI	A,04	
OUT	08	Green ON for North
OUT	09	Green ON for South
CALL	DELAY 1	
MVI	A,22	Yellow ON for East and West
OUT	0A	
MVI	A,02	
OUT	09	Yellow ON for South
OUT	08	Yellow ON for North.
CALL	DELAY 2	

INTERFACING OF STEPPER MOTOR CONTROL:



Interfacing of Stepper Motor.

The necessary steps to interface the 8051 with the stepper motor.

1. The ohm meter is used to measure the resistance of the leads. It is used to identify which COM lads are connected to which winding leads.

2. The common wires are connected to the +5v side of the motor power supply.

3. The 4 bit of 8051 i.e. P1.0, P1.1, P1.2, P1.3 are used to control the 4 leads of the stator winding used a driver to energized the stator. The driver has an internal diode to take care of back emf.

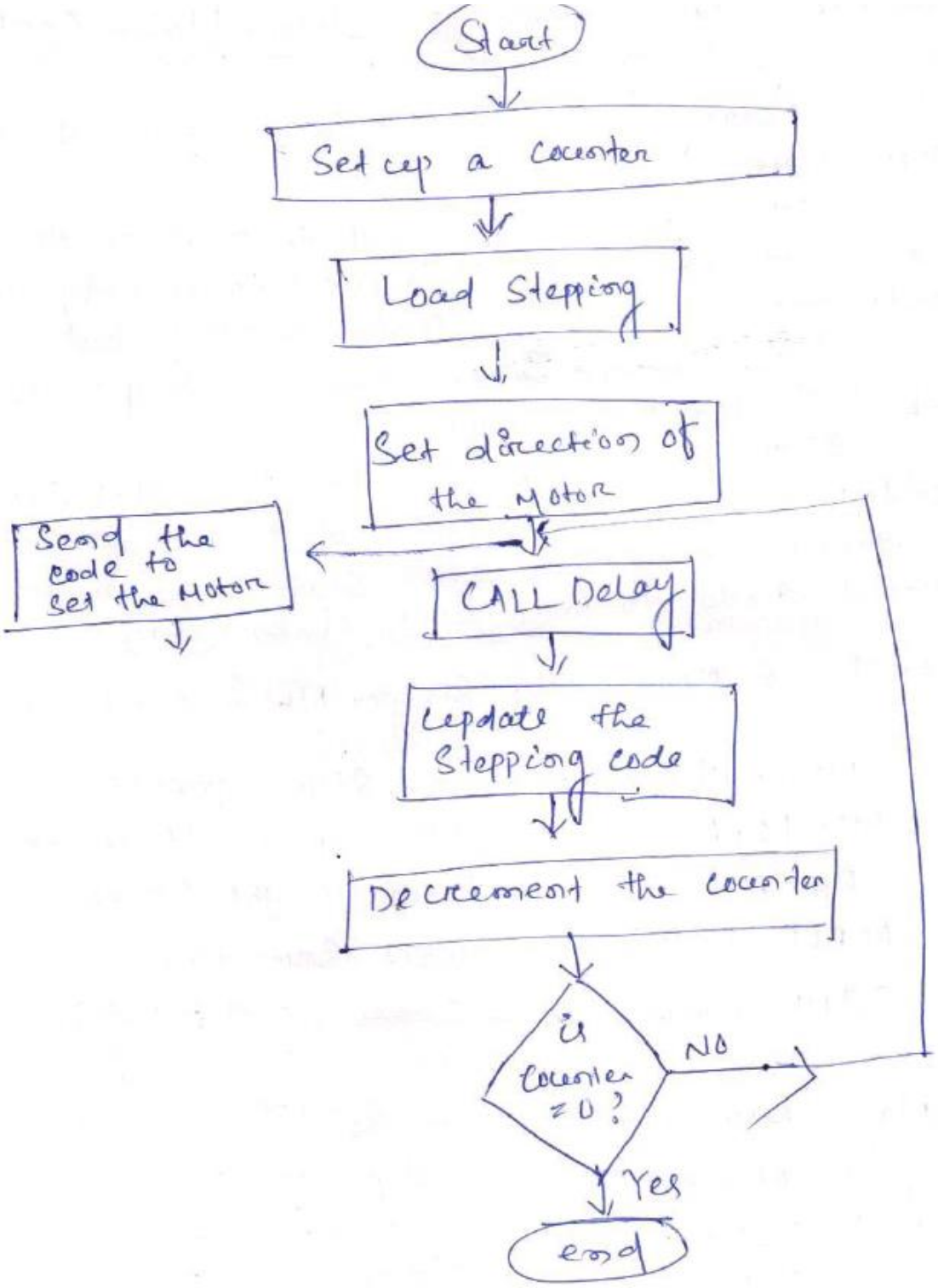
PROGRAM: To rotate the stepper motor continuously.

MOV A,# 66H	Load step sequence
Back: MOV P1,A	Issue sequence to motor
RR A	Rotate right clockwise
A CALL Delay	Wait some time
S JMP Back	Continue doing this

Delay MOV R_2, # 100	R_2=100
L1: MOV R_3, # 255	R_3=255
L2: DJNZ R_3, L_2	
DNZ R_2,L_1	
RET	

Relationship between steps per second and rpm

Steps per second= (Rpm * Steps per revolution)/60.


Start
Set up a Counter
Load Stepping
Set direction of the Motor
Send the code to set the Motor
CALL Delay
Update the Stepping code
Decrement the Counter
is Counter = 0 ?
NO
Yes
end

16-Bit Microprocessor

Introduction:

8086 is a 16-bit of microprocessors. Its internal data lines are of 16. The 8088 and 80188 have their internal architecture of 16 bit but their data lines are only 8. All these microprocessors come under Intel 8086's family.

8086, 80186,80286,8088 and 80188 microprocessor have the same basic set of registers, instruction and addressing modes.

The 8086 is a 16 bit, N-channel, HMOS microprocessor the term HMOS is used for "high-speed MOS". It is a set of 40 pins IC package. The types of packaging are DIP (dual inline package).

AD_0-AD_{15} are the 16 low order address line 8 LSB of data are transmitted on AD_0-AD_7 & 8 MSBs of data on AD_8-AD_{15}.

PIN CONFIGURATION OF 8086

8086 pin configuration

AD_0-AD_{15}:

Address / data lines. There are low-order address buses. They are multiplexed with data.

When AD lines are used to transmit memory address the symbols A is used instead of AD, EXP: A0-A15 & when the data are transmitted D is used instead of AD. EXP: D_0-D7, D_8-D_{15} or D_0-D_{15}.

A_{16}-A_{19} (output):

There are the high order address lines those are multiplexed with status signals.

A_{16}/S_3, A_{17}/s_4=A_{16} & A_{17} are multiplexed with segment identified signal S_3 & S_4.

A_{18}/s_5=A_{18} is multiplexed with the status signal S5.

A_{19}/S_6=A_{19} is multiplexed with status signal.

$\overline{BHE}$/S_7 (o/p):-

Bus high enable /status signal .during T1 is low ,it is to enable data into the most significant half of data bus D_8-D_{15} 8-bit device connected to upper half of data bus use $\overline{BHE}$ signal is available during T3 and T4.

$\overline{RD}$ (read):-

It is used for read operation. It is active when it is low

READY (input):-

The addresses I /O or memory sends acknowledgement through this pin. When it is high it indicates that the peripheral is ready to transfer the data.

RESET(i/p):-

Used for system reset. It is active when it is high

CLK (i/p):-

Clock .10 MHz

INTR:-

Interrupt request

NMI (i/p):-

Non-maskable interrupt request

$\overline{TEST}$ (i/p):-

Wait for test control when it is low the microprocessor continuous execution otherwise waits.

VCC:-

Power supply +5v dc

GND:-

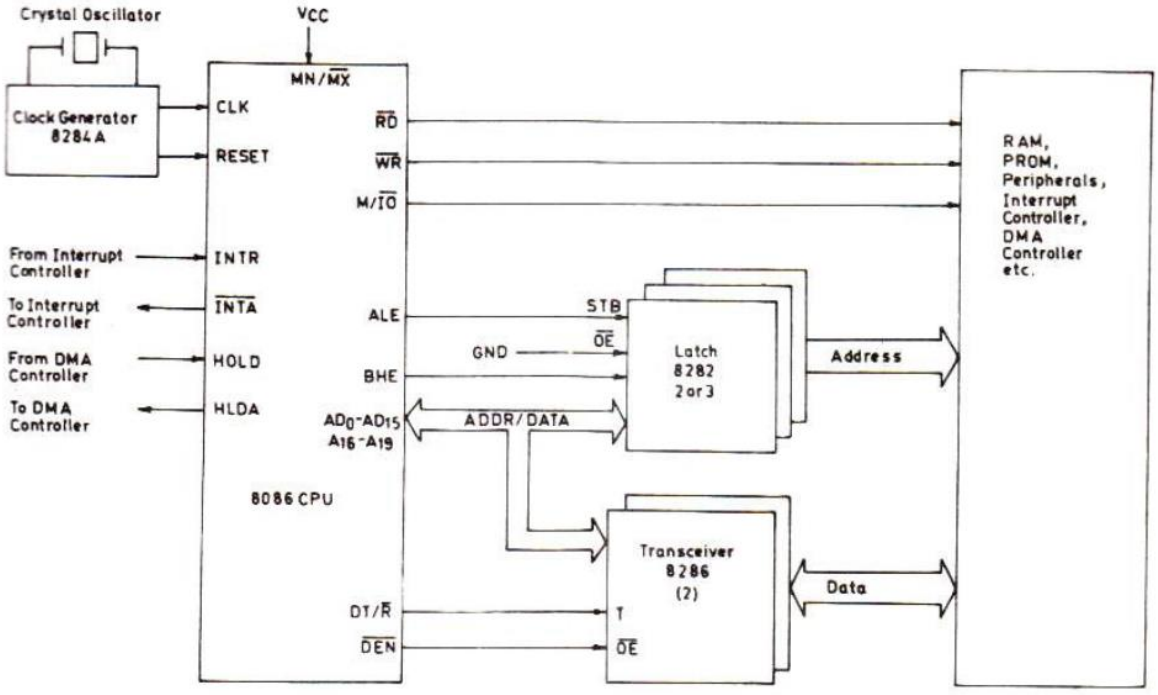
Ground

MINIMUM MODE-MAXIMUM MODE:-

There are 2 available modes of operation for the 8086/8088 microprocessor. i.e. maximum and minimum mode maximum mode operation is obtained by connecting the mode selection pin . MN/$\overline{MX}$ to +5v and maximum mode to ground that pin.

MINIMUM MODE OPERATION:-

Minimum mode operation is the least-expensive way to operate the 8086/8088 microprocessor. all the control signal for the memory and D10 are generated by the microprocessor the minimum mode allows the 8085A ,8bit peripheral to be used with the 8086/8088 with the out any special consideration.



Typical 8086-based Computer System in Minimum Mode Configuration

For the minimum mode operation the pin MN/$\overline{MX}$ is connected to 5v dc supply IC. MN/$\overline{MX}$=VCC

$\overline{INTA}$ (o/p):-

Interrupt acknowledge on receiving interrupt signal the process issues an interrupt acknowledge signal. It is an active low signal.

ALE (o/p):-

Address latch enable .it goes high during T1 the microprocessor sends this signal to latch the addresses.

$\overline{DEN}$(o/p):-

Data enable .it is used for as output enable signal .it is active low.

DT/$\overline{R}$ (o/p):

Data transmit/receive. It is used to control the direction of data flow through the transceiver when it is high bit sent the data and when it is low it receives the data

M/$\overline{IO}$ (o/p):-

When it is high the microprocessor wants to access the memory and when it is low the microprocessor access the I/p device

$\overline{WR}$ (o/p):-

When this signal is low it perform the write operation for memory and i/o device

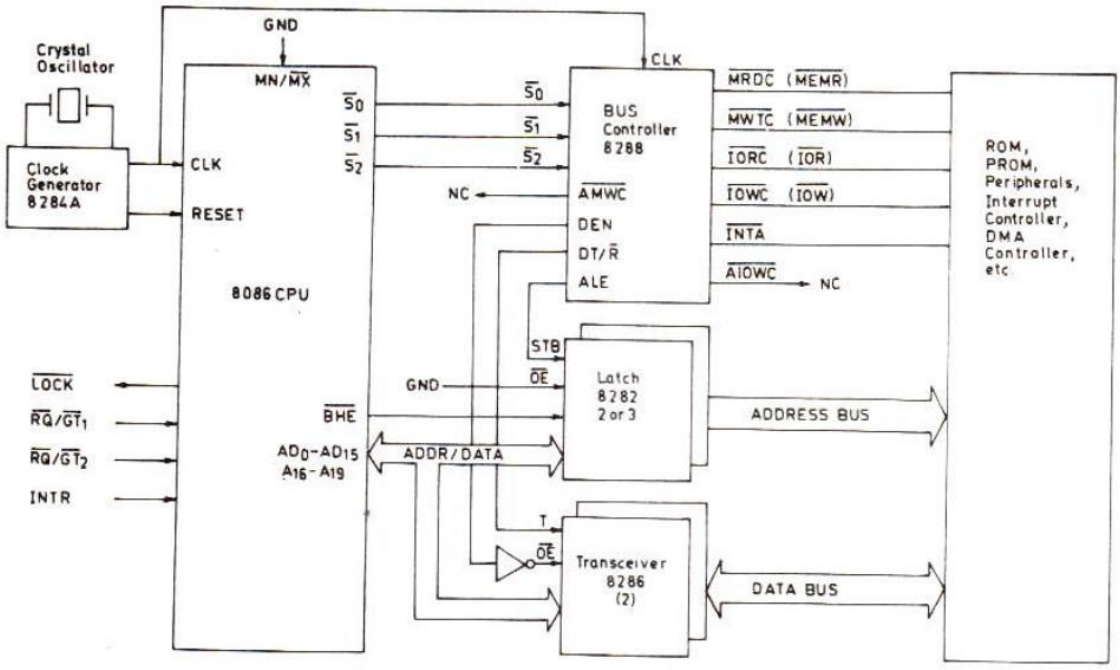
HLDA (o/p):-

Hold acknowledge .it is issued by the microprocessor when it receives hold signal .it is an active high signal. When the hold request is removed HLDA goes low

HOLD (I/O):-

When another device in microprocessor system wants to use the address and data bus , it sends a HOLD request to CPU through this pin , it is also a active high signal

MAXIMUM MODE OPERATION:



Typical Intel 8086-based Computer System in Maximum Mode Configuration

Maximum modes operation differs from minimum mode in that same ctrl signal must be externally generated.

This requires the addition external bus controller there are not enough pins on 8086/8088 for bus ctrl, so for maximum mode operation same new pins and few features have replaced some of them. This maximum mode is used when the system contains external co-processors such as the 8087 arithmetic co-processor.

For the maximum mode operation the pin MN/$\overline{MX}$ is mode low it is grounded.

QS_1, QS_0 (o/p):-

Instruction queue status.

QS_1	QS_0	
0	0	no operation

0	0	1st byte of opcode from queue
1	0	empty the queue
1	1	subsequent byte from queue

$\overline{S0}, \overline{S1}, \overline{S2}$ (o/p):-

These are the signal connected to the controller 8288

S_2	S_1	S_0	
0	0	0	interrupt acknowledge
0	0	1	read data from i/o port
0	1	0	write data from i/o port
0	1	1	halt
1	0	0	opcode fetch
1	0	1	memory read
1	1	0	memory write
1	1	1	passive status

$\overline{LOCK}$ (o/p):-

It is an active low signal when it is low all interrupts are masked and no hold request are generated

$\overline{RQ}/\overline{GT1}, \overline{RQ}/\overline{GT2}$;-

Local bus priority control other processers ask the CPU though these lines to release the local bus $\overline{RQ}/\overline{GT0}$ has higher priority than $\overline{RQ}/\overline{GT1}$,

In maximum mode operation $\overline{WR}$, ALE,$\overline{DEN}$, DT/$\overline{R}$ etc, are not available directly from the processer . These signals are available from the controller 8288.

CLK:-

Clock I/p

ALE:-

Address latch enable.

$\overline{DEN}$**:-**

Data bus enable.

DT/$\overline{R}$**:-**

Data transmit/receive.

$\overline{AEN}$**:-**

Address enable.

CEN:-

Control enable.

$\overline{ALOWC}$**:-**

Advanced I/O write command.

$\overline{IOWC}$**:-**

I/p write command.

$\overline{IORC}$**:-**

I/O read command.

$\overline{AMWC}$**:-**

Advance memory write control.

***$\overline{MWTC}$*:-**

Memory write control.

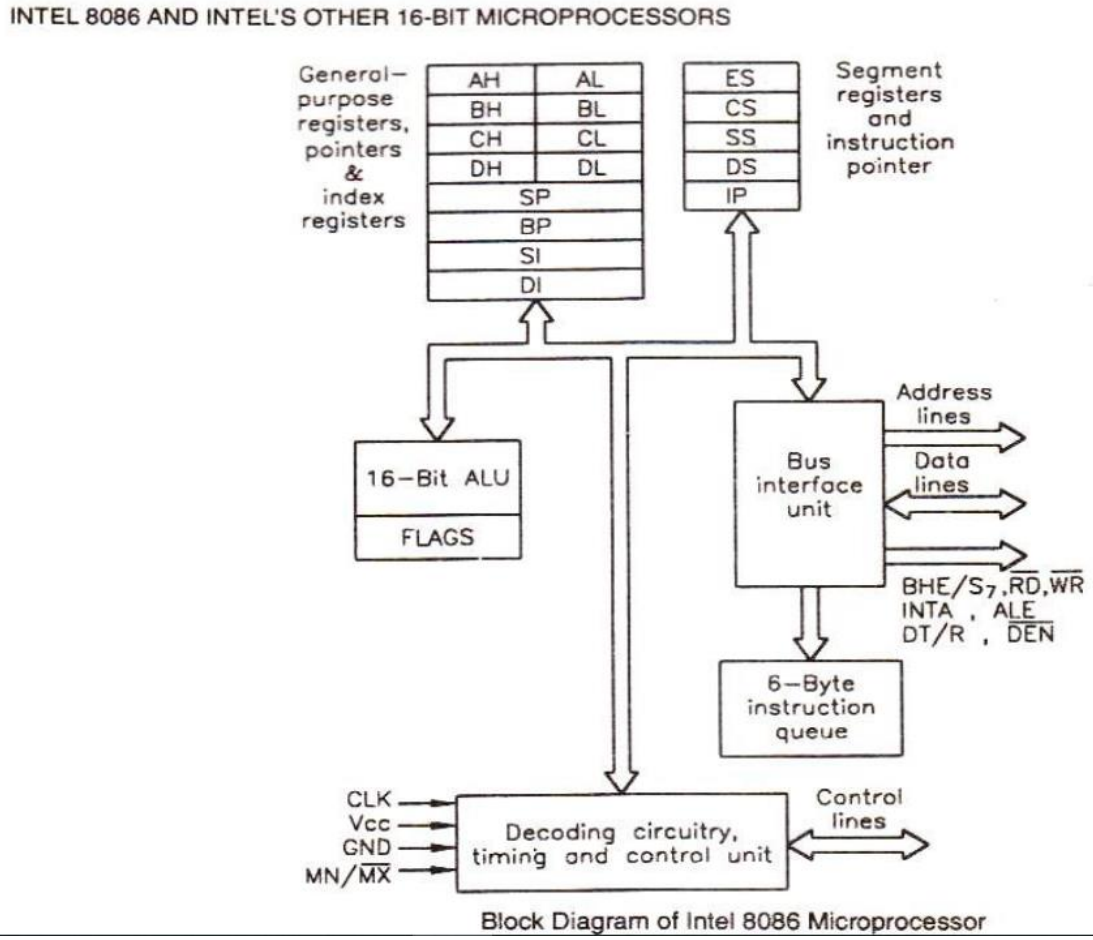
***$\overline{MRDC}$*:-**

Memory read control.

***$\overline{INTA}$*:-**

Interrupt acknowledge.

Architecture of 8086 microprocessor:-



Block Diagram of Intel 8086 Microprocessor

The 8086 microprocessor contains 2 independent units i.e. bus interface unit bus interface unit (BIU) and an execution unit (EU).

The general purpose registers, stack pointer, base pointer and index registers, ALU, Flag register, instruction decoder and timing and control unit constitute execution unit (EU).The segment register, instruction pointer,6-byte instruction queue are associated with the bus interface unit (BIU).

The BIU handles transfer of data and address between the processor and memory i/o device.

While EU executes instruction the BIU fetches instruction. This type of overlapped operation of the functional units of a microprocessor is called pipeline.

REGISTERS OF INTEL 8086:-

The 8086 contains the following registers

(1) General purpose registers
(2) Pointer and index registers
(3) Segment register
(4) Instruction pointer
(5) Status flags.

(1) GENERAL PURPOSE REGISTERS:-

There are four 16-bit general purpose register: AX, BX, CX and DX. Each of these 16-bit register are further subdivided into 2 8-bit registers i.e.

REGISTERS OF INTEL 8086:-

The 8086 contains the following registers

(1) General purpose registers
(2) Pointer and index registers
(3) Segment register
(4) Instruction pointer
(5) Status flags.

(1) GENERAL PURPOSE REGISTERS:-

There are four 16-bit general purpose register: AX, BX, CX and DX. Each of these 16-bit register are further subdivided into 2 8-bit registers i.e.

16-bit register	8-bit high order register	8-bit low order register
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

Register AX is serves as an accumulator .register BX CX, and DX are used as general purpose registers sometimes it serves as special purpose register. BX serves as a base register for the computation of memory address. XC is used counter in case of multi iteration instruction. When the content of CX becomes zero such instruction terminate the execution. DX is also used for memory addressing when the data are transferred between i/o port and memory using certain i/o instruction.

POINTER AND INDEX REGISTER:-

(1) stack pointer ,SP
(2) Base pointer , BP
(3) Source index ,SI
(4) Destination index ,DI

STACK POINTER:-
A 16 bit register which is used to store top of the stack , it is pointing to a memory location inside the stack segment in main memory .
The contain of the stack pointer is added with the stack segment to get the physical address of the top of stack pointer .
BASE POINTER:-
16 bit register which is also holds the offset address of stack segment.
It may locate any memory location inside the stack segment.
It may locate any memory location inside the stack segment.

SOURCE POINTER:-

These are used to store the offset address of data and extra segment.

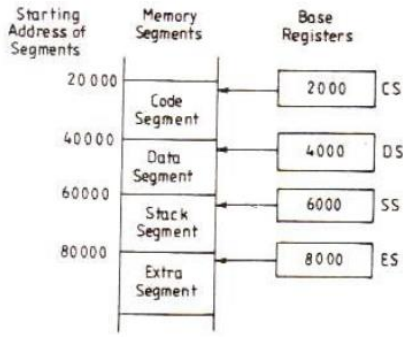
Source index store the offset Address of the instruction and the destination index stores the extra segment of instruction.

SEGMENT REGISTER:-

There are 4 segment register in 8086 i.e.

(1) Code segment
(2) Data segment
(3) Stack segment
(4) Extra segment

In 8086 the memory of 8086 are divided in to 4 segment i.e. code, data, stack and extra .



Memory Segments

CODE SEGMENT:-

The code segment of the memory holds instruction codes of a program.

Code segment points to the starting address of the codes segment.

DATA SEGMENT:-

The data variable and constants given in the program are held in the segment of memory .

Data segment points to the starting address of the data segment.

STACK SEGMENT:-
Stack segment holds address and data of subroutines. it also holds the contents of registers or memory location given in PUSH operation .
EXTRA SEGMENT:-
The extra segment holds the destination address of some of certain string instruction and so on.
A segment register pointer to the starting address of memory segment currently being used.
The content of the stack pointer and the content of stack segment register are used to compute the address the stack lo0cation to be accessed.
The index register i.e. SI and DI together with segment register DS and ES used to perform string operation
.

INSTRUCTION POINTER (ip):-

It also refers as a program counter.

It holds the address of the next instruction that means it point to the memory location in the code register to get the instruction address.

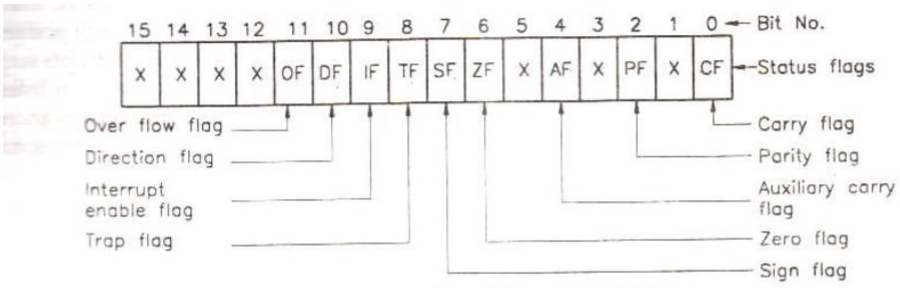
After fetching each instruction the instruction pointer is incremented according to the size of the instruction.

STATUS REGISTERS:-

8086 microprocessor contains a 16 bit status register, it is also called as flag register or program status word (PSW).

There are 9 status flags and other 7bit are not used. The status flag are:

(1) Carry flag
(2) Parity flag
(3) Auxiliary carry flag
(4) Zero flag
(5) Sign flag
(6) Trap flag
(7) Interrupt enable flag
(8) Direction flag
(9) Over flow flag.



Status Flags of Intel 8086

Out of these 9, 6 are the conditional flag and other 3 are the control flag.

The 6 conditional flag are CF, AF, ZF, SF, PF and OF. These flag are set or reset by the processor after the execution of arithmetic or logical instruction.

The 3 control flag are TF, IF and DF . These flag are set or reset by the programmer as required by certain instruction in the program.

All the flags except TF, OF, IF and DF are same as in 8085.

The OF set to 1, when result of signed operation is out of range.

The TF is set to 1 that means a program can be run in single-step mode.

The IF is set to 1, when the INTR of 8086 is enabling.

The DF is used for string operation .it is set to 1, when the string byte are accessed from higher memory address to lower memory address.

INTERRUPTS:-

An interrupt may occurs at normal program execution of a microprocessor .an interrupt caused by a external device is called as hardware interrupt. A microprocessor can also be interrupted by the internal abnormal condition like overflow, division by zero etc.

A programmer can also interrupt microprocessor by inserting INT instruction at desired point in the program while debugging a program. Such interrupt is called software interrupt.

8086 microprocessor can handle up to 256, hardware and software interrupts to store the starting address ISS for 256 interrupts 1kb memory from 0000 to 003FF is set . The starting address of an ISS stored in the 1kb memory space is called interrupt vector or interrupt pointer . From 4bytes memory 2 bytes for store the CS and 2bytes for IP values.

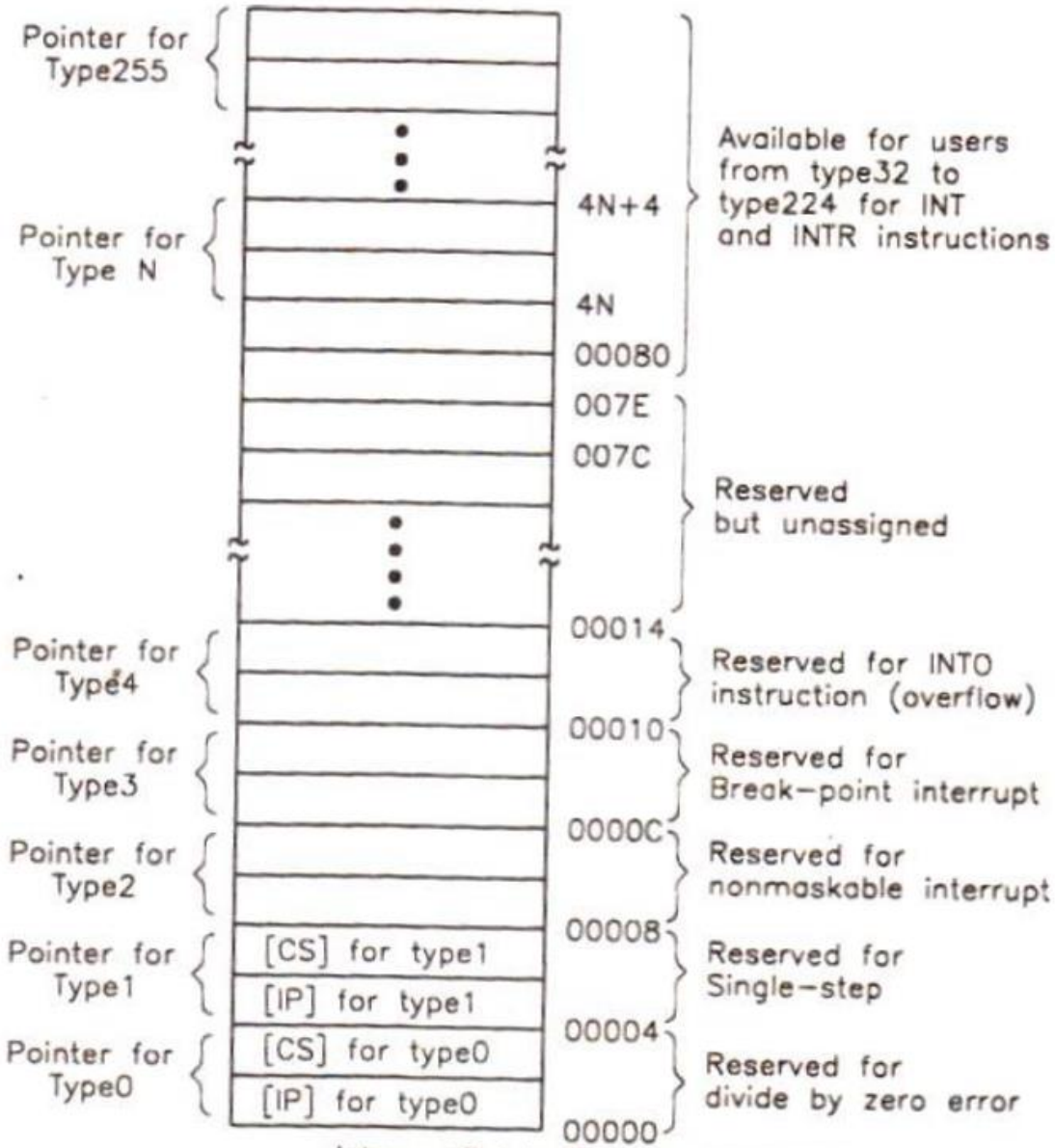
1kb memory space act as a table to contain interrupt vector and hence it is called interrupt vector table or interrupt pointer table .the256 IP is number from 0 to 255 .

1st 5 interrupt are specific interrupt such as divide-by-zero, single-step control, non-maskable interrupt NMI, break point and overflow interrupt. The next 5 to 31 are reserved for other advanced microprocessor of Intel Corporation. The upper 224 interrupt from type 32 to type 255 are available to user for hardware and software interrupt.

The 8086 has 2 hardware interrupt. It can't be disabled by user by using software. It is used by the processor to handle emergency condition.

The ISS for type 2 interrupt saves program after power failure in some RAM provided from the point at which it was interrupt.

INTR is a maskable interrupt it can be enable /disenabled using interrupt flag (IF) .after receiving INTR interrupt from an external device, 8086 acknowledges through INTA signal.



Interrupt Pointer Table for Intel 8086

ADDRESSING MODE OF INTEL 8086:-

An instruction performs specific operation on the specified data .hence, the programmer must specified the required data for an instruction. The way by which an operand is specified for an instruction is called addressing mode.

The 8086 has 8 addressing modes i.e.

(1) register addressing mode
(2) immediate addressing mode
(3) direct addressing mode
(4) register indirect addressing mode
(5) based addressing mode
(6) indexed addressing mode
(7) based index addressing mode
(8) based indexed with displacement

(1) REGISTER ADDRESING MODE:

In this addressing mode the operand is placed in 1 of the 16bit or 8bit general purpose registers.

Ex: MOV AX, CX

ADD AL, BL

ADD CX, DX

(2)IMMEDIATE ADDRESSING MODE:

In immediate addressing the operand is specified in the instruction itself ,

Ex: MOV AL, 35H

MOV BX, 0301H

MOV [0401], 3598H

ADD AX, 4836H

Displacement: it is an 8bit or 16bit immediate value given in the instruction

Base: it is the content of the base register, BX or BP

Index: it is the content of the index register, SI or DI

(3)DIRECT ADDRESSING MODE:

In direct addressing mode the operands offset is given in the instruction as an 8bit or 16bit displacement element.

Ex: ADD AL, [0301]

ADD [0301], AX

(4)REGISTER INDIRECT ADDRESSING MODE:

The operand offset is placed in any 1 of the register BX, BP, SI or DI as specified in the instruction

Ex: MOV AX, [BX]

ADD AL, [SI]

(5) BASED ADDRESSING MODE :

The operand offset is the sum of and 8bit or 16bit displacement and the contents of the based register BX or BP .BX is used as base register for data segment and BP is used as a base register for stack segment.

Ex: MOV AL, [BX+05]

MOV AL, [BX+1346H]

(6) INDEX ADDRESSING MODE:

The operand offset is the sum of the content of an index register SI and DI and 8bit or 16bit displacement

(7) BASED INDEXED ADDRESSING MODE:

The operand offset is the sum of the content of a base register BX or BP and an index register SI or DI

Ex: ADD AX, [BX+SI]

MOV CX, [BX+SI]

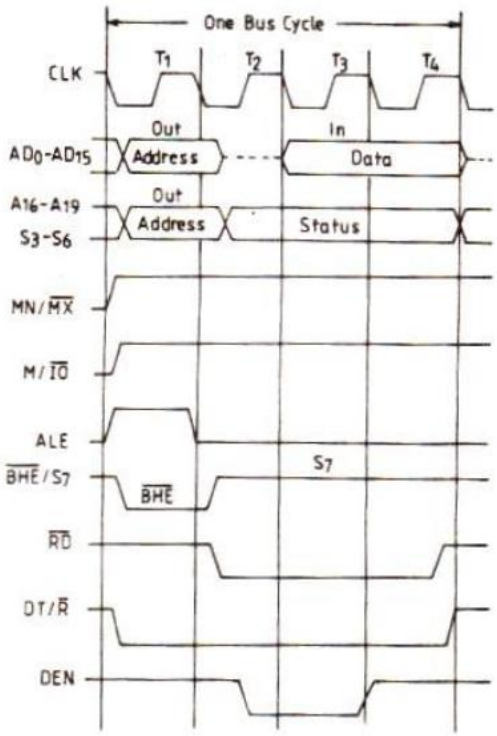
(8)BASED INDEXED WITH DISPLACEMENT:

In this addressing mode the operand offset is given by offset = [BX or BP]+[SI or DI]+8bit or 16bit displacement

Ex: MOV AX, [BX+SI+05]

MOV AX [BX+SI+1235H]

BASIC TIMING DIAGRAM OF INTEL 8086 MICROPROCESSOR:



' Timing Diagram of Intel 8086 for Memory Read in Minimum Mode

CLASSIFICATION OF 8086 INSTRUCTIONS:

Instruction are classified a the basic of function of they perform. There are categorized into:-

1. Data transfer instruction
2. Arithmetic instruction
3. Logical instruction
4. Program Execution transfer/Branch instruction

1.DATA TRANSFER INSTRUCTIONS :

This instruction performs all the data movement operation like MOV, Load, Store, Exchange, I/P,O/P,PUSH,POP instruction.

The source of the data may be a register memory location, port etc . The destination may be also the same.

Ex: MOV, XCHG, PUSH, POP, LDS, LEA , IN,OUT ETC .

2. ARITHMATIC INSTRUCTIONS:

The instruction performs all the arithmetic operation like addition, substraction, multiplication, division, increment, decrement, comparison, etc.

Ex: ADD, SUB, INC, MUL, DIV, CMP, etc

3. LOGICAL INSRTUCTION :

This perform all the logical operation like AND, OR, X-OR, NOT, TEST ETC

4. PROGRAM EXECUTION TRANSFER OR BRANCH INSTRUCTION:

The instruction of this group transfers program execution from the normal sequence of instruction .to the specified destination or target.

After the execution of such instruction, the processor starts instruction

Ex: JMP, JC, JZ, CALL, RET etc

ITERATION CONTROL INSTRUCTIONS:

Instruction like loop loopz, loopne etc are come under this group

INTERRUPT INSTRUCTION:

Instruction such as INT, INTO and IRET are comes under this group.

PROCESS CONTROL INSTRUCTION :

Instruction like flag manipulation and machine ctrl. Are comes under this group.

PROCESS CONTROL INSTRUCTION :

Instruction like flag manipulation and machine ctrl. Are comes under this group.

Ex : CLC< CLD< STC <LOK etc

STRING INSTRUCTION:

It handles the string operation such as string movement, comparison, scan, load and store.

Ex.; MOV S/MOV SB/ MOV SW, CMPS/CMPSB/CMPSW, SCAS/SCAB/SCASW, LODS/LODSB/LOSDW etc.