ARYAN SCHOOL OF ENGINEERING & ECHNOLOGY

BARAKUDA, PANCHAGAON, BHUBANESWAR, KHORDHA-752050

LECTURE NOTE

SUBJECT NAME- MICROPROCESSOR & MICROCONTROLLER

BRANCH-COMPUTER SCIENCE ENGG.

SEMESTER-4TH SEM

ACADEMIC SESSION-2022-23

PREPARED BY- DIPAK KUMAR SAHOO

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

1.1 INTRODUCTION TO MICROPROCESSOR AND MICROCOMPUTER-

MICROPROCESSOR:

- A Microprocessor is a multipurpose, Programmable clock driven, register based electronic device,
- That read binary instruction from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as outputs.
- Microprocessor is clock driven semiconductor device which for is manufactured by using LSI and VLSI technique.

MICROCOMPUTER:

- A **microcomputer** is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU). It includes a microprocessor, memory, and input/output (I/O) facilities.

- Microcomputers became popular in the 1970s and 80s with the advent of increasingly powerful microprocessors.
- Examples of Microcomputers are Intel 8051 controller-a single board computer,

- IBM PC and Apple Macintosh computer.

1.2 DIFFERENCE BETWEEN MICROCOMPUTER AND MICROPROCESSOR-

General Architecture of Microcomputer System:

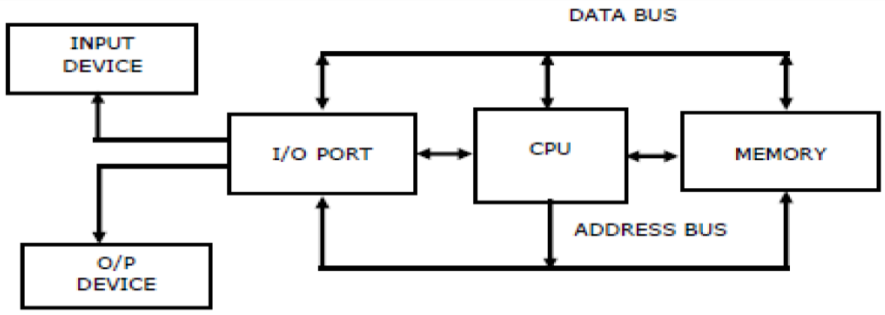


Figure: Block Diagram of a simple Microcomputer

The major parts are CPU, Memory and I/O

There are three buses, address bus, data bus and control bus;

MEMORY:

- Memory consist of RAM and ROM, the purpose of memory is to store binary codes for the sequences of instructions you want the computer to carry out.
- The second purpose of the memory is to store the binary-coded data with which the computer is going to be working.

INPUT / OUTPUT:

- The input/output or I/O Section allows the computer to take in data from the outside world or send data to the outside world.
- Peripherals such as keyboards, video display terminals, printers are connected to I/O Port.

CPU (CENTRAL PROCESSING UNIT):

- In a microcomputer CPU is a microprocessor.
- The fetches binary coded instructions from memory, decodes the instructions into a series of simple actions and carries out these actions in a sequence of steps.
- The CPU also contains an address counter or instruction pointer register, which holds the address of the next instruction or data item to be fetched from memory.

Architecture of microprocessor-

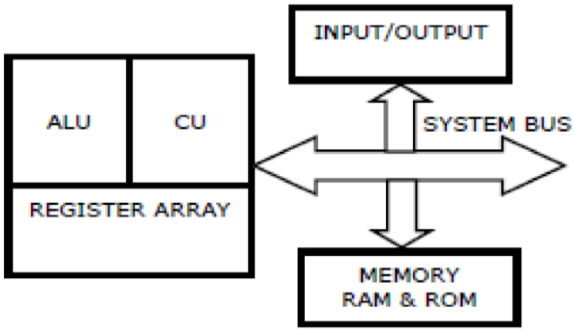


Figure: Microprocessor Based System with Bus Architecture.

Microprocessor is divided into three segments-

1. ALU
2. Register
3. Control Unit

Arithmetic Logic Unit:

- This is the area of Microprocessor where various computing functions are performed on data.

- The ALU performs operations such as addition, subtraction and logic operations such as AND, OR and exclusive OR.

Control Unit:

- The Control Unit Provides the necessary timing and control signals to all the operations in the Microcomputer
- It controls the flow of data between the Microprocessor and Memory and Peripherals.
- The Control unit performs 2 basic tasks
  →Sequencing
  →Execution

Register:

- These are storage devices to store data temporarily.

- There are different types of registers depending upon the microprocessor.
- These registers are primarily used to store data temporarily during the execution of a program and are accessible to the user through the instructions.

1.3 CONCEPT OF ADDRESS, DATA & CONTROL BUS-
ADDRESS BUS:

- The address bus consists of 16, 20, 24 or 32 parallel signal lines.
- On these lines the CPU sends out the address of the memory location that is to be written to or read from. The no of memory location that the CPU can address is determined by the number of address lines.

- If the CPU has N address lines, then it can directly address 2N memory locations i.e. CPU with 16 address lines can address 216 or 65536 memory locations.

DATA BUS:

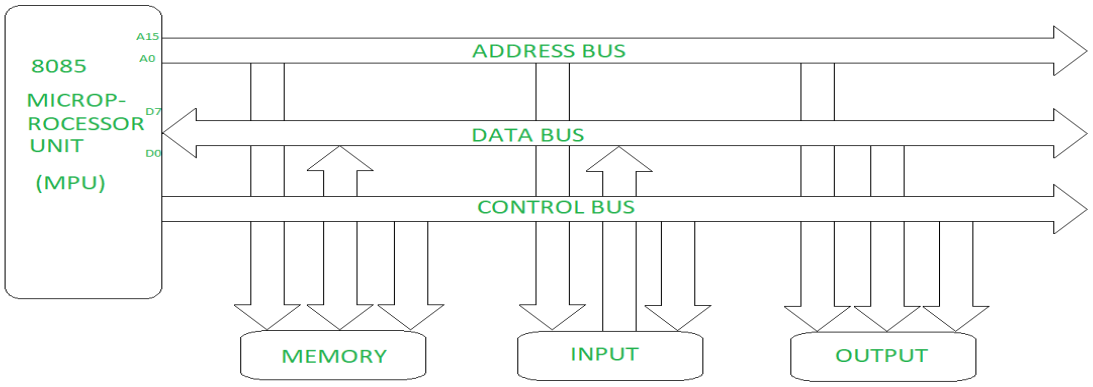
- The data bus consists of 8, 16 or 32 parallel signal lines.
- The data bus lines are bi-directional.
- This means that the CPU can read data in from memory or it can send data out to memory.

CONTROL BUS:

- The control bus consists of 4 to 10 parallel signal lines.

- The CPU sends out signals on the control bus to enable the output of addressed memory devices or port devices.
- Typical control bus signals are Memory Read, Memory Write, I/O Read and I/O Write.

1.4 GENERAL BUS STRUCTURE:



Bus organization system of 8085 Microprocessor

ADDRESS BUS:

- It is a group of conducting wires which carries address only.
- Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices.
- Length of Address Bus of 8085 microprocessor is 16 Bit (i.e. Four Hexadecimal Digits), ranging from 0000 H to FFFF H, (H denotes Hexadecimal).
- The microprocessor 8085 can transfer maximum 16 bit address which means it can address 65,536 different memory location.
- The Length of the address bus determines the amount of memory a system can address.
- Such as a system with a 32-bit address bus can address 2^32 memory locations.
- If each memory location holds one byte, the addressable memory space is 4 GB. However, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

DATA BUS:

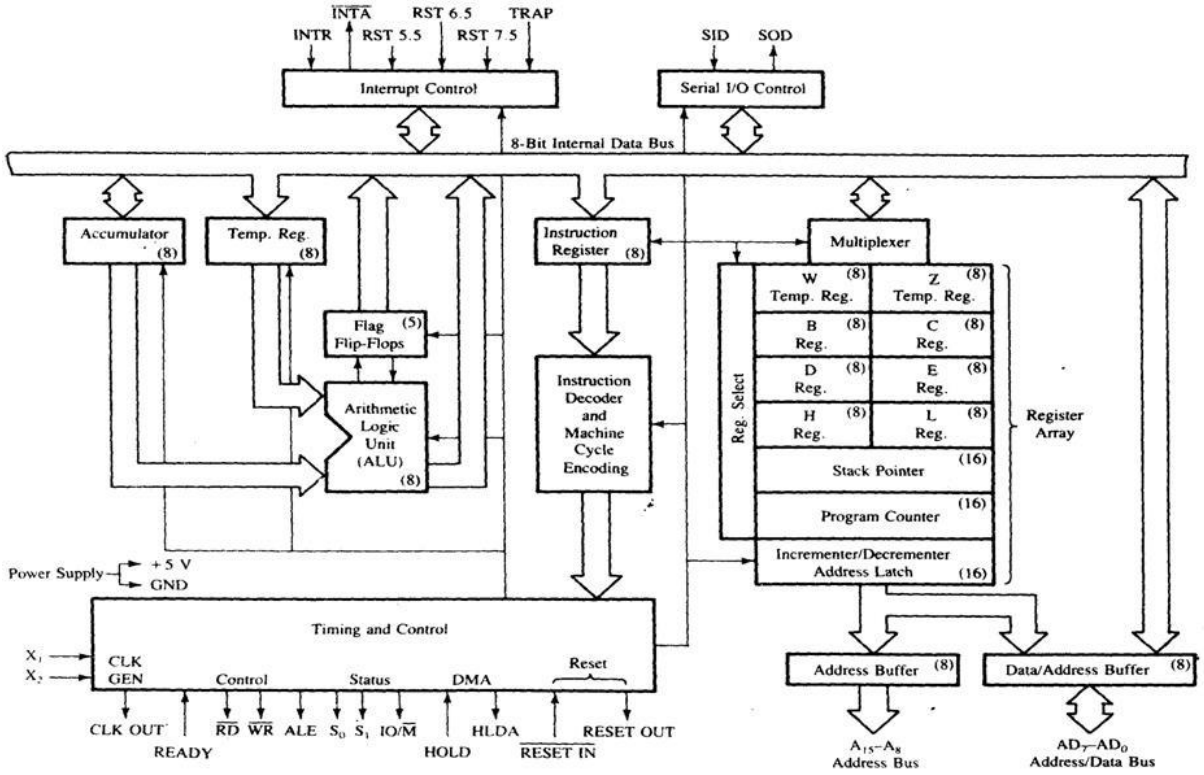
- It is a group of conducting wires which carries Data only.
- Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/output devices and from memory or Input/output devices to microprocessor.
- Length of Data Bus of 8085 microprocessor is 8 Bit (That is, two Hexadecimal Digits), ranging from 00 H to FF H. (H denotes Hexadecimal).

- When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.
- The width of the data bus is directly related to the largest number that the bus can carry, such as an 8 bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255.A 16 bit bus can carry 0 to 65535.

CONTROL BUS:

- It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data i.e. what to do with selected memory location. Some control signals are:
- Memory read
- Memory write
- I/O read
- I/O Write
- Opcode fetch

1.5 ARCHITECTURE OF 8085 MICROPROCESSOR:



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)

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, and 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).

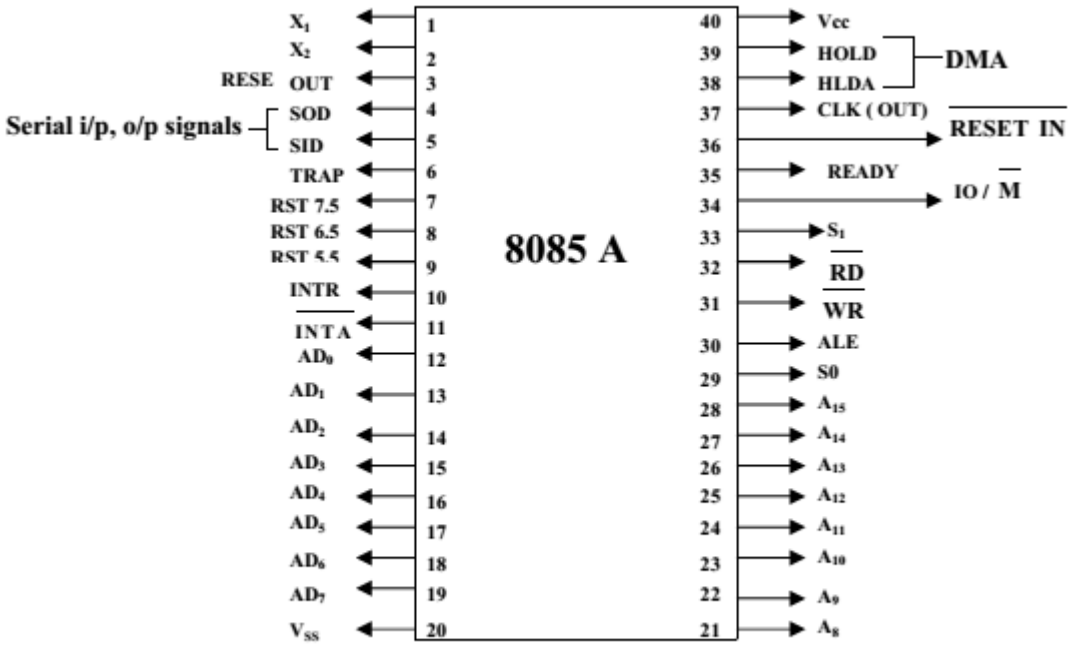
Address buffer and address-data buffer:

- 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.

1.6 SIGNAL DESCRIPTION OF 8085:



Pin Diagram of 8085

The pins of an 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' & IO/M'.

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.

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.

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.

S1 & S0:

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

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

Power supply:

There are 2 power supply signals V_{cc} & V_{ss}. 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.

TRAP:

- It is a non-maskable interrupt, having the highest priority among all interrupts. By default, it is enabled until it gets acknowledged. In case of failure, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H.

RST7.5:

- It is a maskable interrupt, having the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

RST 6.5:

- It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address.

RST 5.5:

- It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.

INTR:

It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor.

When **INTR signal goes high**, the following events can occur:

The microprocessor checks the status of INTR signal during the execution of each instruction.

- When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
- When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

INTA':

It is an interrupt acknowledgment sent by the microprocessor after INTR is received.

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.

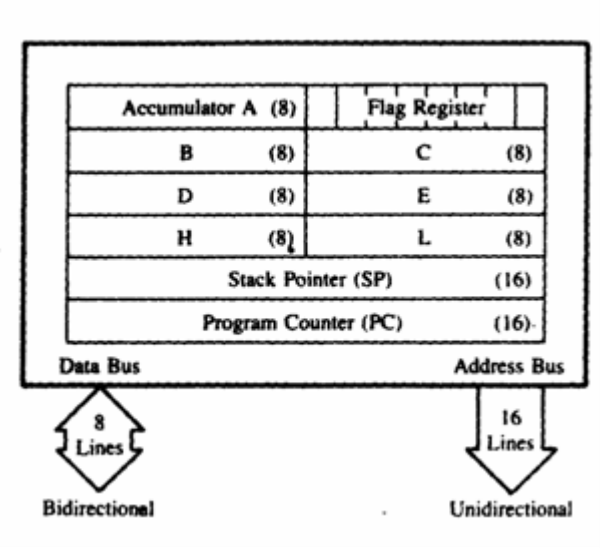
SID (Serial input data line):

The data on this line is loaded into accumulator whenever a RIM instruction is executed.

- When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
- When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

1.7 REGISTER ORGANIZATION:

It has six addressable 8-bit registers: A, B, C, D, E, H, L and two 16-bit registers PC and SP. These registers can be classified as:



- **General Purpose Registers**
- **Temporary Registers:** Temporary data register, W and Z registers
- **Special Purpose Registers:** Accumulator, Flag registers, Instruction register
- **Sixteen-bit Registers:** Program Counter (PC), Stack Pointer (SP)

1. General Purpose Registers:

- Registers B, C, D, E, H, and L are general purpose registers in 8085 Microprocessor. All these GPRS are 8-bits wide. They are less important than the accumulator.
- They are used to store data temporarily during the execution of the program. For example, there is no instruction to add the contents of B and E registers.
- At least one of the operands has to be in A. Thus to add Band E registers, and to store the result in B register, the following have to be done.

➢ Move to A register the contents of B register.

➢ Then add A and E registers. The result will be in A.

➢ Move this result from A register to B register.

- It is possible to use these registers as pairs to store 16-bit information. Only B-C, D-E, and H-L can form register pairs.
- When they are used as register pairs in an instruction, the left register is understood to have the MSB byte and the right registers the LSB byte.

- For example, in D-E register pair, the content of the D register is treated as the MSB byte, and the content of E register is treated as the LSB byte.

2. Temporary Registers:

- **Temporary Data Register: -**
- The ALU has two inputs. One input is supplied by the accumulator and other from the temporary data register.
- The programmer cannot access this temporary data register. However, it is internally used for execution of most of the arithmetic and logical instructions.
- **W and Z register:-** Wand Z registers are temporary registers. These registers are used to hold 8-bit data during the execution of some instructions. These registers are not available for the programmer since 8085Microprocessor Architecture uses them internally.

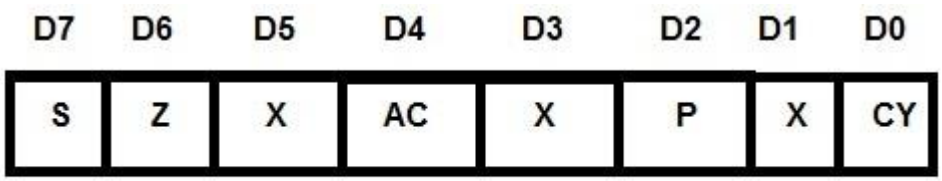
3. Special Purpose Registers:

Accumulator (A):

- 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.

Flag Register:

- It is a 3-bit register, in which five of the bits carry significant information in the form of flags: S (Sign flag), Z (Zero flag), AC (Auxiliary carry flag), P (Parity flag), and CY (carry flag).



Flag Register of 8085

- **S-Sign flag: -** After the execution of arithmetic or logical operations, if bit D7 of the result is 1, the sign flag is set. In a given byte if D7 is1, the number will be viewed as a negative number. If D7 is U, the number will be considered as a positive number.
- **Z-Zero flag**:-The zero flag sets if the result of the operation in ALU is zero and flag resets if the result is non-zero. The zero flags are also set if a certain register content becomes zero following an increment or decrement operation of that register.
- **AC-auxiliary Carry flag: -** This flag is set if there is an overflow out of bit 3 i.e. carry from lower nibble to higher nibble (D3 bit to D4 bit). This flag is used for BCD operations and it is not available for the programmer.
- **P-Parity flag: -** Parity is defined by the number of one's present in the accumulator. After arithmetic or logical operation, if the result has an even number of ones, i.e. even parity, the flag is set. If the parity is odd, the flag is reset.
- **CY-Carry flag:** - This flag is set if there is an overflow out of bit 7. The carry flag also serves as a borrow flag for subtraction. In both the examples shown below, the carry flag is set.

Instruction Register:-

- In a typical processor operation, the processor first fetches the opcode of instruction from memory (i.e. it places an address on the address bus and memory responds by placing the data stored at the specified address on the data bus).
- The CPU stores this opcode in a register called the instruction register. This opcode is further sent to the instruction decoder to select one of the 256 alternatives.

4. Sixteen Bit Registers:

Program counter (PC):-

- Program is a sequence of instructions. Microprocessor fetches these instructions from the memory and executes them.
- The program counter is a special purpose register which, at a given time, stores the address of the next instruction to be fetched.
- Program Counter acts as a pointer to the next instruction.
- How processor increments program counter depends on the nature of the instruction; for one-byte instruction it increments program counter by one, for two-byte instruction it increments program counter by two and for three-byte instruction it increments

program counter by three such that program counter always points to the address of the next instruction.

Stack Pointer (SP):-

The stack is a reserved area of the memory in the RAM where temporary information may be stored. A 16-bit stack pointer is used to hold the address of the most recent stack entry.

1.8 DISTINGUISH BETWEEN GPR AND SPR:

GPR-

- It stands for General purpose registers.
- In these registers data can be accessed directly without requiring any intermediate.
- Examples of GPR are B, C, D, E, H, and L.
- These registers are of 8-bit.
- In order to hold 16 bit data, two 8 bit register can be combined or they can work in pairs such as B-C, D-E and H-L. These pairs are known as register pairs.
- The H-L pair works as a memory pointer.
- A memory pointer holds the address of a particular memory location.

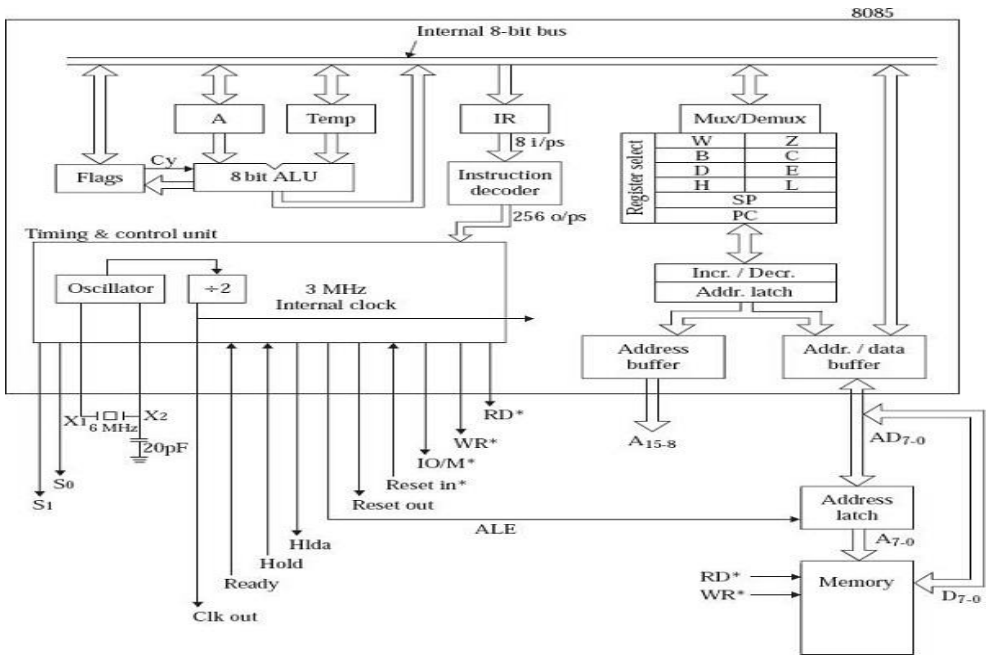
SPR-

- SPR stands for special purpose register.
- In special purpose register data cannot accessed directly and requires an intermediate.
- Examples of SPR are Accumulator, program counter, stack pointer.
- These registers are used only by microprocessor not by users.

1.9 TIMING AND CONTROL UNIT:

- We use Timing and controlling unit in 8085 for the generation of timing signals and the signals to control.
- All the operations and functions both interior and exterior of a microprocessor are controlled by this unit.
- X2 and CLK output pins: To do or rather perform the operations of timing in the microcomputer system, we have a generator called clock generator in the CU of 8085.
- Other than the quartz crystal the complete circuit of the oscillator is within the chip. The two pins namely X1 and X2 are taken out from the chip to give the connection to the crystal externally.
- We connect a capacitor of 20pF between the terminal X2 and ground just to analyze if the crystal is getting started.
- The frequency of the crystal is divided by 2 which divide the counter of the unit of control by 2.
- Internally 8085A works with a frequency of 3 MHz internally with clock frequency. Hence a crystal of frequency of 6-MHz crystal gets connected between X1 and X2.

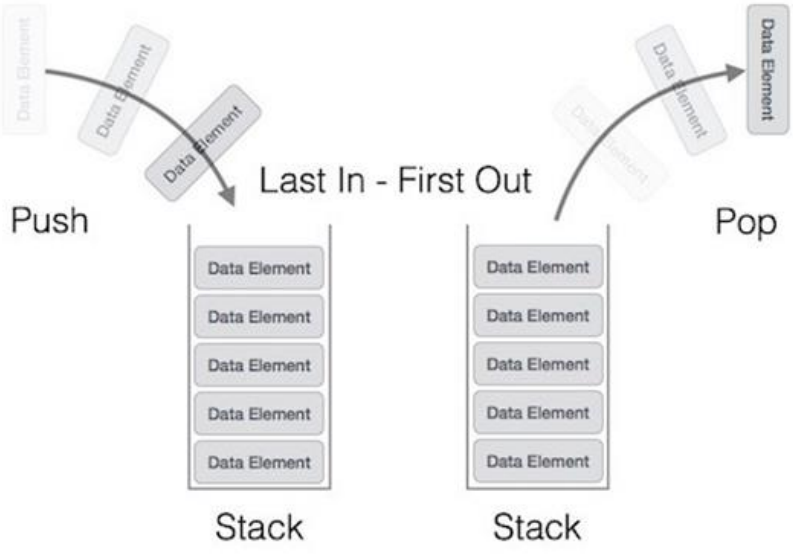
- Every operation in the entire 8085 system occurs with the given synchronization process with the clock. There are Peripheral chips like 8251 USART, which does not operate until a small clock signal is in need.



1.10 STACK, STACK POINTER AND STACK TOP:

STACK:

- The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data when the microprocessor branches to a subroutine.
- Then the return address used to get pushed on this stack. Also to swap values of two registers and register pairs we use the stack as well.



STACK POINTER:

- It is a special purpose 16-bit register that stores the address of the "**top of stack**".
- "8085" provides the **"stack pointer"** which gives the address of the "top of stack". So, whenever you want to store an item it stacks, you just store it at the address provided by the stack pointer.

STACK operation in 8085 microprocessor.

The stack is a reserved area of the memory in RAM where temporary information may be stored. An 8-bit stack pointer is used to hold the address of the most recent stack entry. This location which has the most recent entry is called as the top of the stack.

When the information is written on the stack, the operation is called PUSH. When the information is read from the stack, the operation is called POP. The stack works on the principle of Last in First Out.

1.11 8085 INTERRUPTS:

- Interrupt is a process where an external device can get the attention of the microprocessor**.**
- An interrupt is considered to be an emergency signal that may be serviced.
- The Microprocessor may respond to it as soon as possible.
- The process starts from the I/O device
- The process is asynchronous

Classification of Interrupts:

Interrupts can be classified into two types:

- **Maskable Interrupts** (Can be delayed or Rejected)
- **Non-Maskable Interrupts** (Cannot be delayed or Rejected)

Interrupts can also be classified into:

- **Vectored** (the address of the service routine is hard-wired)
- **Non-vectored** (the address of the service routine needs to be supplied externally by the device)

What happens when MP is interrupted?

- When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an Interrupt Service Routine (ISR) to respond to the incoming interrupt.
- Each interrupt will most probably have its own ISR.
- Responding to an interrupt may be immediate or delayed depending on whether the interrupt is maskable or non-maskable and whether interrupts are being masked or not.

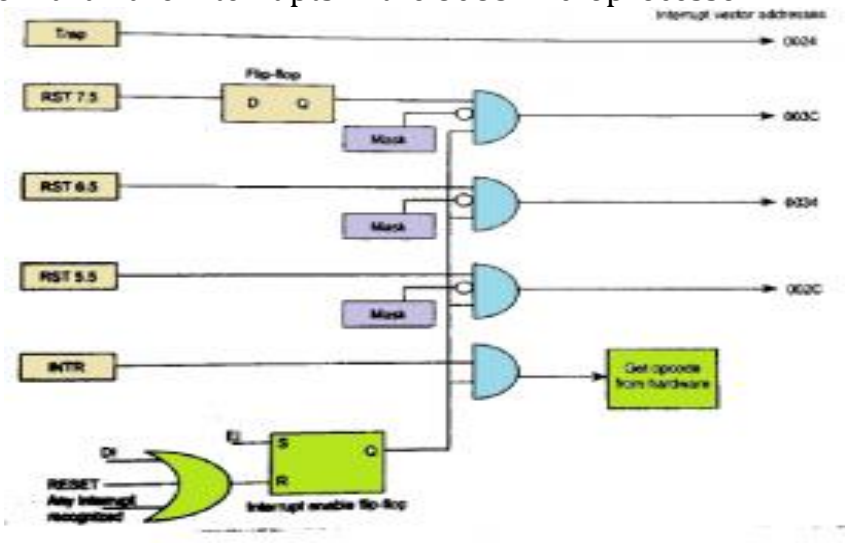
- There are two ways of redirecting the execution to the ISR depending on whether the interrupt is vectored or non-vectored.
- **Vectored:** The address of the subroutine is already known to the Microprocessor.
- **Non Vectored:** The device will have to supply the address of the subroutine to the Microprocessor.
- When a device interrupts, it actually wants the MP to give a service which is equivalent to asking the MP to call a subroutine. This subroutine is called **ISR** (Interrupt Service Routine)
- The 'EI' instruction is a one byte instruction and is used to enable the non-maskable interrupts.
- The 'DI' instruction is a one byte instruction and is used to disable the non-maskable interrupts.
- The 8085 has a single Non-Maskable interrupt. The non-maskable interrupt is not affected by the value of the Interrupt Enable flip flop.

The 8085 has 5 interrupt inputs.

- The **INTR** input is the only non-vectored interrupt. INTR is maskable using the EI/DI instruction pair.
- **RST 5.5, RST 6.5, RST 7.5** are allautomatically vectored and are mask able.
- **TRAP** is the only non-maskable interrupt inthe 8085.it is also automatically vectored.

Masking of interrupt SIM, RIM:

- When we study interrupts in 8085 microprocessor then we should know Masking of Interrupts in 8085 microprocessor.
- In 8085 microprocessor masking of interrupt can be done for four hardware interrupts INTR, RST 5.5, RST 6.5, and RST 7.5.
- The masking of 8085 interrupts is done at different levels. In bellow figure shows the organization of hardware interrupts in the 8085 microprocessor.



- The maskable interrupts are by default masked by the Reset signal. So no interrupt is recognized by the hardware reset.
- The interrupts can be enabled by the EI instruction.
- The three RST interrupts can be selectively masked by loading the appropriate word in the accumulator and executing SIM instruction. This is called software masking.
- All maskable interrupts are disabled whenever an interrupt is recognized.
- All maskable interrupts can be disabled by executing the DI instruction.
- If we talk about RST 7.5 interrupt. It alone has a flip-flop to recognize edge transition. The DI instruction reset interrupt enable flip-flop in the processor and the interrupts are disabled. To enable interrupts, EI instruction has to be executed.

SIM Instruction:

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

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

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

RIM Instruction:

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

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

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

- E.g. write an assembly language program to enables all the interrupts in 8085 after reset**.** **EI** Enable interrupts MVI A, 08H: Unmask the interrupts **SIM:** Set the mask and unmask using SIM instruction.

UNIT-2: INSTRUCTION SET AND ASSEMBLY LANGUAGE PROGRAMMING

2.1 INSTRUCTION WORD SIZE:

- The total memory location required to feed the instruction in memory is called as **instruction word size**.
- The memory location of 8085 microprocessor can accommodate 8-bits of data.
- To store 16-bits data, they are stored in two consecutive memory locations (i.e. 2 Bytes).
- According to the instruction word size in 8085 microprocessor, there are three types of instructions:

a. 1-Byte instruction
b. 2-Byte instruction
c. 3-Byte instruction

1 – Byte Instructions:

- They include opcode and operands in the same byte.
- Operands are internal registers and coded into the instruction.
- Instructions require one memory location to store the single byte in the memory.

Note:
Instructions having the only register or register pair as the operand is 1 – Byte Instructions.
Instructions in the absence of operand are also 1 – Byte Instructions.
Examples:

MOV B, C

LDAX B

NOP

HLT

2 – Byte Instructions:

- 1st byte specifies opcode and 2nd byte specifies operand.
- Instructions require two memory locations to store in the memory.

Note:
Instructions having the 8-bit number either as an address or data as the operand is 2 – Byte Instructions.
Examples:

MVI B, 26 H

IN 56 H

3 – Byte Instructions:

- In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address.
- The 2nd byte holds the low order address.
- The 3rd-byte holds the high order address.
- Instructions require three memory locations to store the single byte in the memory.

Note:
Instructions having the 16-bit number either as an address or data as the operand is 3 – Byte Instructions.
Examples:

LDA 2050 H

JMP 2085 H

2.2 ADDRESSING MODES:

- The various ways of specifying data (or operands) for instructions are called as **addressing modes.**
- The 8085 addressing modes are classified into following types:

1. Immediate addressing mode
2. Direct addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Implicit addressing mode

1. Direct Addressing mode:

- In this addressing mode the address of the operand is specified in the instruction itself.

or

- The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called Direct Addressing mode. i.e., the address of the operand is available in the instruction itself. This is a 3-byte instruction.

Example:
LDA 9525H→ Load the contents of memory location into Accumulator.

STA 8000H→Store the contents of the Accumulator in the location 8000H

IN 01H→ Read the data from port whose address is 01H

2. Register addressing modes:

- In this addressing mode the address of the operand is one of the general purpose register.

or

- In this mode the operands are microprocessor registers only i.e. the operation is performed within various registers of the microprocessor.

Example:

- MOV A, B→Move the contents of B register to A register.
- SUB D→ Subtract the contents of D register from Accumulator.
- ADD B, C→Add the contents of C register to the contents of B register.

3. Register indirect addressing modes:

- In this addressing mode the address of the operand is specified by a register pair.

or

- The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. So it is called Register indirect addressing mode.

Example:

- LXIH 9570H→Load immediate the H-L pair with the address of the location 9570H
- MOV A, M→ Move the contents of the memory location pointed by the H-L pair to accumulator

4. Immediate Addressing mode:

- In this addressing mode the operand is specified in the instruction itself.

or

- In this mode operand is a part of the instruction itself is known as Immediate Addressing mode. If the immediate data is 8-bit, the instruction will be of two bytes. If the immediate data is 16 bit, the instruction is of 3 bytes.

Example:

ADI DATA →Add immediate the data to the contents of the accumulator.
LXIH 8500H→Load immediate the H-L pair with the operand 8500H
MVI 08H → Move the data 08 H immediately to the accumulator
SUI 05H →Subtract immediately the data 05H from the accumulator

5. Implicit Addressing mode:

- In this addressing mode the instruction don't require the address of the operand.

or

- The mode of instruction which do not specify the operand in the instruction but it is implicated, is known as implicit addressing mode. i.e., the operand is supposed to be present generally in accumulator.

Example:
CMA→complement the contents of Accumulator

CMC→ Complement carry

RLC→ Rotate Accumulator left by one bit

RRC→ Rotate Accumulator right by one bit

STC→Set carry.

2.3 INSTRUCTION SET OF 8085:

- An instruction is a binary bit pattern which performs a specific function in a system. The entire group of instructions of a system is called the instruction set.

- Instruction set determines what functions the microprocessor can perform with a single instruction.

- The instruction set in microprocessor 8085 can be classified into five functional categories:

OR

- An instruction is a command to the microprocessor to perform a given task on a specified data.
- Each instruction has two parts: one is task to be performed, called the operation code (opcode), and the second is the data to be operated on, called the operand.
- The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

1. Data transfer (copy) operations
2. Arithmetic operations
3. Logical operations
4. Branching operations and
5. Machine-control operations.

1. DATA TRANSFER INSTRUCTION:

- These instructions move data between registers, or between memory and registers.
- This group of instructions copies data from a location called as source to another location called as destination, without modifying the contents of the source
- These instructions are not the data transfer instructions but data copy instruction because the source is not modified.

Opcode Operand **Copy from source to destination**	**Description**
MOV Rd, Rs	This instruction copies the contents of the source
M, Rs	register into the destination register; the contents of
	The source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. Example: MOV B, C or MOV B, M

Rd, M Move immediate 8-bit

MVI Rd, data	The 8-bit data is stored in the destination register or
M, data	Memory. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: MVI B, 57H or MVI M, 57H

Load accumulator

LDA 16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. Example: LDA 2034H

Load accumulator indirect

LDAX B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B

Load register pair immediate

LXI Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H

Load H and L registers direct

LHLD 16-bit address

The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered.
Example: LHLD 2040H

Store accumulator direct

STA 16-bit address

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

Store accumulator indirect

STAX Reg. pair

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

Store H and L registers direct

SHLD 16-bit address

The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers

HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Example: SHLD 2470H

Exchange H and L with D and E

XCHG none — The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.
Example: XCHG

Arithmetic Operations:

They perform arithmetic operations, such as, addition, subtraction, increment, and decrement.

Addition:

- Addition of any 8-bit number, or the contents of a register or the contents of a memory location is added to the contents of the accumulator and the sum is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- For example the contents of register B cannot be added directly to the contents of the register C. 8085 can also perform 16-bit. It can also perform BCD addition.

Subtraction:

- Subtraction of any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator.
- The subtraction is performed in 2's compliment, and if the results is negative. Then they are expressed in 2's complement.
- No two other registers can be subtracted directly. 8085 do not perform 16-bit subtraction.

Increment or Decrement:

- The 8-bit contents of any register or a memory location can be incremented or decrement by 1.
- Similarly, the 16-bit contents of a register pair can be incremented or decrement by 1.
- These increment and decrement operations can be performed directly in the source itself. It means without using accumulator.

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 R,ADDM
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 R,ADDM
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 55
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 55
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 H, 3025H
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
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 R,SUB M
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 R,SBBM
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 55
SBI	8-bit data	Subtract the immediate from the accumulator with borrow	The 8-bit data and borrow is subtracted from the contents of the accumulator & the result is stored in the accumulator
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 R, INR M

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 R
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 R,DCR M
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 R
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 OPERATIONS:

These type instructions performs various logical operations with the contents of the accumulator. 8085 can perform six logical operation which are:

- AND
- OR
- Exclusive-OR
- NOT
- Compare
- Rotate

A 8-bit number can be logically ANDed with the contents of the accumulator. It can also be a content of register or of a memory location. The results are stored in the accumulator. The content of the accumulator can be complimented.

Rotate:
Each bit of the accumulator can be shifted either left or right to the next position.

Compare:

- Any 8-bit number or the content of a register, or content of a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.
- The result is reflected by zero and carry flags.

Opcode	Operand	Meaning	Explanation
CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	8-bit data	Compare immediate with the accumulator	The second byte data is compared with the contents of the accumulator.
ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and

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

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

BRANCHING OPERATIONS:

This group of instruction transfers the control of microprocessor from one location to another location. 8085 can perform four types of branching operations. These are:

- JMP-Jump within a program.
- CALL-Jump from main program to sub-routine.
- RET-Jump from sub-routine to main program.
- RST-Jump from main program to instruction sub routine.

Jump:

- Conditional jumps are the important aspect of the decision-making process in the programming of a microprocessor.
- These instructions tests for a certain conditions and alter the program sequence when the condition is met.
- For example zero or carry flag, In addition, the instruction set also includes an instruction called unconditional jump.

Call, return, and restart:

- These type of instructions changes the sequence of a program either by calling a sub-routine or returning from a sub-routine.

- The conditional call and return instructions can also test the condition flags.

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 sequences 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 &MACHINE-CONTROL OPERATIONS:

These type of instructions controls the machine functions, such as halt, interrupt, or do nothing.

Opcode	Operand	Meaning	Explanation

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

Stack instructions are as follows:

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

I/O instructions are as follows:

IN - Initiate Input Operation
OUT - Initiate Output Operation

2.4 ASSEMBLY LANGUAGE PROGRAMMING OF 8085:

What is Assembly Language Program?

- Machine language and Hex code instructions are very difficult for the programmer.
- Hence for programmer, the instructions of microprocessor are made in the form of English abbreviation (short form). These instructions are name as Assembly Language instructions or mnemonics.
- The combinations of different mnemonics are known as Assembly Language Program and it is a low level language.

Examples of assembly language program

Loading Register or Memory with Data

Example 1: Write a program to transfer 07 H in register L.

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	2E, 07	MVI	L, 07	Move immediate 07 in register L
2002 H	76	HLT		Stop or terminate the program

- The instruction MVI L, 07 will move the data 07 to the register L.
- The instruction will stop the program.
- The machine code for the instruction MVI L, 07 is 2E, 07.
- The 1st byte of the machine code is 2E which is the Hex code for the instruction MVI L.
- The second byte is the data 07. The machine code for HLT is 76.
- The machine codes are fetch in the memory locations, starting from the memory locations 2000 H.
- Memory location 2000 H contains 2E, 2001 H contains 07 and memory location 2002 H contain 76, After the execution of a program, the contents of Register L can be examined which are 07.

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	3E, 08	MVI	A,08	Get 08 in register A
2002 H	4F	MOV	C,A	Move the contents of register A to register C
2003 H	76	HLT		Halt

Example 2 Write a program to load register A with 08 H and then move it to register C.

- In this program the instruction MVI A, ON H will place the given data 08 1H in the register A.
- The Hex code for MVI A, 08 H is 3E, 08 IH where 3E is the Hex code for MVI A.
- The instruction MOV C, A will move the contents of register A to the register C. Its machine code is 4F.
- With this instruction the data of register A is copies into the register C. It means the given data, is 08 H which was previously placed in register A is now copied into the register C.
- The instruction HLT whose machine code is 76 stops the program.
- The memory locations required for this program are 2000 H to 2003 H. Any other memory locations can be selected. After the execution of a program, the contents of register C can be examined.

Example 3. Write a program to load the contents of memory location 2050 H into accumulator and then move this data into register B

Memory Address	Machine Code	Mnemonics	Operands	Comments
2000 H	3A, 50, 20	LDA	2050 H	Load the contents of memory location 2050 H into the accumulator
2002 H	47	MOV	B,A	Move the contents of register A to register B
2004 H	76	HLT		Stop

- The instruction LDA 2050 H will load the contents of memory location 2050 H into the accumulator.
- The machine code for the instruction LDA is 3A.
- The instruction MOV B. A (Machine code 47) will move the contents of Accumulator to the register B.
- First of all data 07 is fetch in the memory location 2050.
- Then memory locations 2000 H contain 3A, 2001 H contain 50 H, 2002 H contains 20 H, 2003 H contains 47 H and 2004 H contains 76 H.

- After execution of a program, the contents of register B can be examined.

Example 4. Write a program to add two 8-bit numbers.

MEMORY ADDRESS	MACHINE CODE	MNEMONICS	OPERANDS	COMMENTS
2000	21,01,25	LXI	H,2501H	Get address of first number in H-L pair.
2003	7E	MOV	A,M	1st number in accumulator.
2004	23	INX	H	Increment content of H-L pair.
2005	86	ADD	M	Add 1st and 2nd numbers.
2006	32,03,25	STA	2503H	Store sum in 2503H.
2009	76	HLT		Stop the program.

EXPLANATION:

- The 1stnumber was stored in the memory location 2501H.
- 2501 was placed in H-L pair by the execution of the instruction LXI H, 2501H.
- The instruction MOV A,M moved the content of the memory location addressed by H-L pair to the accumulator.
- Thus the 1stnumber 49H which was in the 2501H was placed in the accumulator.
- The INX H increased the content of H-L pair from 2501 to 2502H.
- The instruction ADD M added the content of the memory location addressed by H-L pair with the accumulator.
- The result got stored in the accumulator.
  The instruction STA 2503H stored the sum in the memory location 2503H.
- The instruction HLT ended the program.

Example 5. Write a program to subtract two 8-bit numbers.

MEMORY ADDRESS	MACHINE CODES	MNEMONICS	OPERAND	COMMENTS
2000	21,01,25	LXI	H,2501	Get address of 1st in H-L pair.
2003	7E	MOV	A,M	1st number in accumulator.
2004	23	INX	H	Content of H-L pair increases from 2501 to 2502 H
2005	96	SUB	M	1stnumber- 2ndnumber.
2006	23	INX	H	Content of H-L pair becomes 2503 H.
2007	77	MOV	M,A	Store result in 2503 H.

2008	76	HLT		Stop the program

EXPLANATION:

- The first no. was stored in the memory location 2501 H.
- 2501 H was placed in H-L pair by the execution of the instruction LXI H, 2501 H.
- The instruction MOV A, M moved the content of the memory location addressed by H-L pair to the accumulator.
- Thus the first no. 49H which was in the 2501 H was placed in the accumulator.
- The INX H increased the content of H-L pair from 2501 to 2502 H.
- The instruction SUB M subtracted the content of the memory location addressed by H-L pair from the accumulator.
- The second no. which was in the memory location 2502 H was subtracted from the first no. which was in the accumulator.
- The result got stored in the accumulator.
- The INX H increased the content of H-L pair from 2502 to 2503 H.
- The instruction MOV M, A moved the content of the accumulator to the memory location addressed by H-L pair to the accumulator.
- The result which was stored in the accumulator got stored in the memory location 2503 H.
- The instruction HLT ended the program.

Example 6.Write an assembly language program in 8085 microprocessor to perform AND operation between lower and higher order nibble of 8 bit number.

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

EXPLAINATION:

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

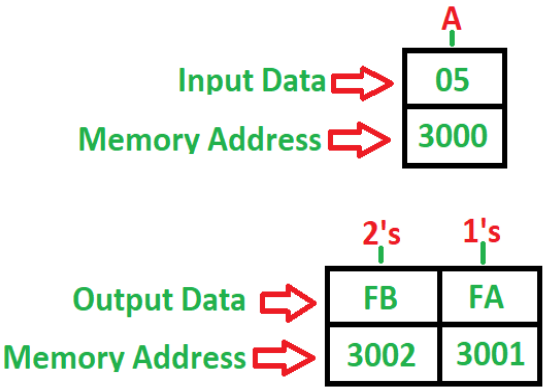
200C	RLC	Rotate accumulator left by one bit without carry
200D	RLC	Rotate accumulator left by one bit without carry
200E	RLC	Rotate accumulator left by one bit without carry
200F	ANA B	A ←A (AND) B
2010	STA 3050	M[3050] ← A
2013	HLT	END

EXPLANATION:

Registers A, B are used for general purpose.

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

Example 7- Write a program to find 1's and 2's complement of 8-bit number where starting address is 2000 and the number is stored at 3000 memory address and store result into 3001 and 3002 memory address.



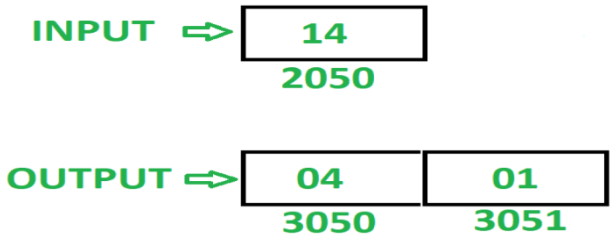
Program –

MEMORY ADDRESS	MNEMONICS	OPERANDS	COMMENT
2000	LDA	[3000]	[A] ← [3000]
2003	CMA		[A] ← [A^]
2004	STA	[3001]	1's complement
2007	ADI	01	[A] ← [A] + 01
2009	STA	[3002]	2's complement
200C	HLT		Stop

EXPLANATION:

1. **A** is an 8-bit accumulator which is used to load and store the data directly
2. **LDA** is used to load accumulator direct using 16-bit address (3 Byte instruction)
3. **CMA** is used to complement content of accumulator (1 Byte instruction)
4. **STA** is used to store accumulator direct using 16-bit address (3 Byte instruction)
5. **ADI** is used to add data into accumulator immediately (2 Byte instruction)
6. **HLT** is used to halt the program

Example8:– Write an assembly language program in 8085 microprocessor to show masking of lower and higher nibble of 8 bit number.
Example –



Assumption: - 8 bit number is stored at memory location 2050. After masking of nibbles, lower order nibble is stored at memory location 3050 and higher order nibble is stored at memory location 3051.

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LDA 2050	A ←M[2050]
2003	MOV B, A	B ←A
2004	ANI 0F	A ←A (AND) 0F

2006	STA 3050	M[3050] ← A
2009	MOV A, B	A ← B
200A	ANI 0F	A ←A (AND) 0F
200C	RLC	rotate content of A left by 1 bit without carry
200D	RLC	rotate content of A left by 1 bit without carry
200E	RLC	rotate content of A left by 1 bit without carry
200F	RLC	rotate content of A left by 1 bit without carry
2010	STA 3051	M[3051] ←A
2013	HLT	END

EXPLANATION:

Registers A, B are used:

1. **LDA 2050:** load the content of memory location 2050 in accumulator A.
2. **MOV B, A:** moves the content of A to B.
3. **ANI 0F:** perform AND operation of A with 0F and store the result back to A.
4. **STA 3050:** store content of A in memory location 3050.
5. **MOV A, B:** moves the content of B in A.
6. **ANI 0F:** perform AND operation of A with 0F and store the result back to A.
7. **RLC:** rotate content of A left by 1 bit without carry. Use this instruction 4 times to reverse the content of A.
8. **STA 3051:** store the content of A in memory location 3051.
9. **HLT:** stops executing the program and halts any further execution.

COUNTER:

- A counter is designed simply by loading appropriate number into one of the registers and using INR or DNR instructions.
- Loop is established to update the count.
- Each count is checked to determine whether it has reached final number; if not, the loop is repeated. C

TIME DELAY:

- Procedure used to design a specific delay.
- A register is loaded with a number, depending on the time delay required and then the register is decremented until it reaches zero by setting up a loop with conditional jump instruction.

Using 8-bit register as counter:

- Counter is another approach to generate a time delay. In this case the program size is smaller. So in this approach we can generate more time delay in less space. The following program will demonstrate the time delay using 8-bit counter.

Program	Time (T-States)
• MVI B,FFH • LOOP: DCR B • JNZ LOOP • RET	7 4 7/10 10

- Here the first instruction will be executed once, it will take 7 T-states. DCR C instruction takes 4 T-states. This will be executed 255 (FF) times. The JNZ instruction takes 10 T-states when it jumps (It jumps 254 times), otherwise it will take 7 T-States. And the RET instruction takes 10 T-States.
- 7 + ((4*255) + (10*254)) + 7 + 10 = 3584. So the time delay will be 3584 * 1/3μs = 1194.66μs. So when we need some small delay, then we can use this technique with some other values in the place of FF.
- This technique can also be done using some nested loops to get larger delays. The following code is showing how we can get some delay with one loop into some other

Using 16-bit register-pair as counter:

- Instead of using 8-bit counter, we can do that kind of task using 16-bit register pair. Using this method more time delay can be generated. This method can be used to get more than 0.5 seconds delay. Let us see and example.

Program	Time (T-States)
LXI B,FFFFH **LOOP: DCX B** **MOV A,B** **ORA C** **JNZ LOOP** **RET**	10 6 4 4 10 (For Jump), 7(Skip) 10

- In the above table we have placed the T-States. From that table, if we calculate the time delay, it will be like this:
- 10 + (6 + 4 + 4 + 10) * 65535H – 3 + 10 = 17 + 24 * 65535H = 1572857. So the time delay will be 1572857 * 1/3μs = 0.52428s. Here we are getting nearly 0.5s delay.

- In different program, we need 1s delay. For that case, this program can be executed twice. We can call the Delay subroutine twice or use another outer loop for two-time execution.

Looping, counting and indexing (Call/JMP)

To perform a repetitive task, commonly used techniques are looping, counting, and indexing. To add data bytes stored in memory, for example, the following steps are necessary.

LOOPING

- The programming technique used to instruct the microprocessor to repeat tasks is called looping.
- This task is accomplished by using jump instructions.
- Define the task to be repeated is called **Looping**.
- A loop is set up by using either a conditional Jump or an unconditional Jump as illustrated in Examples.

COUNTING:

- Specify how many times the task is to be repeated is called **Counting**.
- The counter is set by loading a count (number of times the task is to be repeated) .into a register or a register pair, and the counting is done by decrementing the count every time the loop is repeated. The counter can also be set up to count from 0 to the final count using increment instructions.

INDEXING:

- Specify the location of the data is called **Indexing**.
- The starting location of the data can be specified by loading the memory address into a register pair and using the register pair as a memory pointer or index.

SETTING FLAGS:

- Indicate the end of the repetitive task is called **Setting Flags**.
- The end of repetition is indicated by the flag of the conditional Jump instruction. When the condition is true, the loop is repeated; when the condition is false, the loop execution is terminated, and the execution goes to the next instruction in memory.

CLASSIFICATION OF LOOPS:

1 Conditional loop

2.Unconditional loop

CONTINUOUS LOOP:

- Repeats a task continuously.
- A continuous loop is set up by using the unconditional jump instruction
- A program with a continuous loop does not stop repeating the tasks until the system is reset.

CONDITIONAL LOOP:

- A conditional loop is set up by a conditional jump instructions.
- These instructions check flags (Z, CY, P, S) and repeat the tasks if the conditions are satisfied.
- These loops include counting and indexing.

CONDITIONAL LOOP AND COUNTER:

- A counter is a typical application of the conditional loop.
- A microprocessor needs a counter, flag to accomplish the looping task.
- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either increment or decrement the counter.
- Loop is set up by a conditional jump instruction.
- End of counting is indicated by a flag.

Example:

- Steps to add ten bytes of data stored in memory locations starting ata given location and display the sum.
- The microprocessor needs

1. A counter to count 10 data bytes.
2. An index or a memory pointer to locate where data bytes are stored.
3. To transfer data from a memory location to the microprocessor(ALU)
4. To perform addition
5. Registers for temporary storage of partial answers
6. A flag to indicate the completion of the stack
7. To store or output the result.

Stack and Subroutines programs:

- The stack is a reserved area of the memory in RAM where we can store temporary information.
- Interestingly, the stack is a shared resource as it can be shared by the microprocessor and the programmer.
- The programmer can use the stack to store data. And the microprocessor uses the stack to execute subroutines.

- The 8085 has a 16-bit register known as the 'Stack Pointer.'
- This register's function is to hold the memory address of the stack. This control is given to the programmer.
- The programmer can decide the starting address of the stack by loading the address into the stack pointer register at the beginning of a program.
- The stack works on the principle of First in Last Out. The memory location of the most recent data entry on the stack is known as the Stack Top.

How does a stack work in assembly language?

- We use two main instructions to control the movement of data into a stack and from a stack. These two instructions are **PUSH** and **POP.**
- PUSH – This is the instruction we use to write information on the stack.
- POP – This is the instruction we use to read information from the stack.
- There are two methods to add data to the stack: **Direct method and indirect method**

Direct method:

In the direct method, the stack pointers address is loaded into the stack pointer register directly.

```
LXI SP, 8000H
LXI H, 1234H
PUSH H
POP D
HLT
```

Explanation of the code:

- LXI SP, 8000H – The address of the stack pointer is set to 8000H by loading the number into the stack pointer register.
- LXI H, 1234H – Next, we add a number to the HL pair. The most significant two bits will enter the H register. The least significant two bits will enter the L register.
- PUSH H – The PUSH command will push the contents of the H register first to the stack. Then the contents of the L register will be sent to the stack. So the new stack top will hold 34H.
- POP D – The POP command will remove the contents of the stack and store them to the DE register pair. The top of the stack clears first and enters the E register. The new top of the

stack is 12H now. This one clears last and enters the D register. The contents of the DE register pair is now 1234H.

- HLT – HLT indicates that the program execution needs to stop.

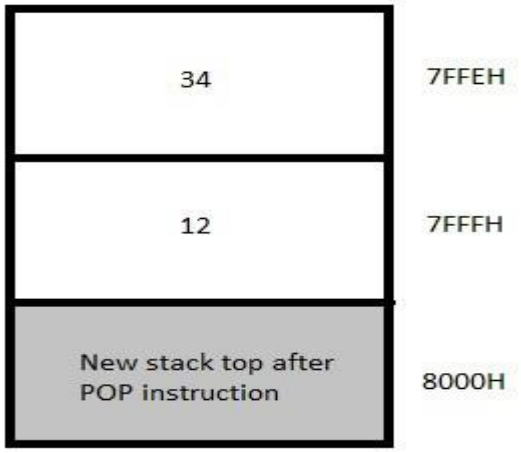
Indirect method:

In the indirect method, the stack pointers address is loaded into the stack pointer register via another register pair.

```
LXI H, 8000H
SPHL
LXI H, 1234H
PUSH H
POP D
HLT
```

Explanation of the code

- LXI H, 8000H – The number that we wish to enter into the stack pointer, 8000H, is loaded into the HL pair register.
- SPHL – This is a special command that we can use to transfer data from HL pair to stack pointer (SP). Now, the contents of the HL pair are in the SP.
- LXI H, 1234H – Next, we add a number to the HL pair. The most significant two bits will enter the H register. The least significant two bits will enter the L register.
- PUSH H – The PUSH command will push the contents of the H register first to the stack. Then the contents of the L register will be sent to the stack. So the new stack top will hold 34H.
- POP D – The POP command will remove the contents of the stack and store them to the DE register pair. The top of the stack clears first and enters the E register. The new top of the stack is 12H now. This one clears last and enters the D register. The contents of the DE register pair is now 1234H.
- HLT – HLT indicates that the program execution needs to stop.
- Both the methods can be shown diagrammatically with the following diagram.



What is a Subroutine is assembly language?

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

BCD to binary conversion in 8085:

(2200H) = 67H
(2300H) = 6 x OAH + 7 = 3CH + 7 = 43H

Source Program:

```
LDA 2200H      : Get the BCD number
MOV B, A       : Save it
ANI OFH        : Mask most significant four bits
MOV C, A       : Save unpacked BCDI in C register
MOV A, B       : Get BCD again
ANI FOH        : Mask least significant four bits
RRC            : Convert most significant four bits into unpacked BCD2
RRC
RRC
RRC
MOV B, A       : Save unpacked BCD2 in B register
```

```
    XRA A              : Clear accumulator (sum = 0)
    MVI D, 0AH         : Set D as a multiplier of 10
Sum: ADD D             : Add 10 until (B) = 0
    DCR B              : Decrement BCD2 by one
    JNZ SUM            : Is multiplication complete? i if not, go back and add again
    ADD C              : Add BCD1
    STA 2300H          : Store the result
    HLT                : Terminate program execution
```

BCD to HEX conversion in 8085 Microprocessor:

Program

```
    LXI H,5000
    MOV A,M     ;Initialize memory pointer
    ADD A       ;MSD X 2
    MOV B,A     ;Store MSD X 2
    ADD A       ;MSD X 4
    ADD A       ;MSD X 8
    ADD B       ;MSD X 10
    INX H       ;Point to LSD
    ADD M       ;Add to form HEX
    INX H
    MOV M,A     ;Store the result
    HLT
```

Result

Input:

Data 0: 02H in memory location 5000

Data 1: 09H in memory location 5001

Output:

Data 0: 1DH in memory location 5002

Program to find larger of two numbers

PROGRAM:

MEMORY ADDRESS	MACHINE CODE	LABELS	MNEMONICS	OPERANDS	COMMENTS
2000	21,01,25		LXI	H,2501H	Address of 1st number in H-L pair.
2003	7E		MOV	A,M	1stnumber in accumulator.
2004	23		INX	H	Address of 2nd number in H-L pair.
2005	BE		CMP	M	Compared 2nd number with 1st number. Is the 2nd number> 1st?
2006	D2,0A,20		JNC	AHEAD	No, larger number is in accumulator. Go to AHEAD
2009	7E		MOV	A,M	Yes, get 2nd number in accumulator.
200A	32,03,25	AHEAD	STA	2503H	store larger number in 2503H
200D	76		HLT		Stop the program.

Example-1:

Data:
2501→98 H
2502→87 H
Result:
2503→98 H and it is stored in the memory location 2503 H.

Program to find smaller of two numbers

PROGRAM:-

ADDRESS	MACHINE CODES	LABELS	MNEMONICS	OPERANDS	COMMENTS
2000	21,01,25		LXI	H,2501H	Address of the 1st number in H-L pair
2003	7E		MOV	A,M	1st number in accumulator
2004	23		INX	H	Address of the 2nd number in H-L pair.
2005	BE		CMP	M	Compare 2nd number with 1st .Is 1st number < 2nd number?
2006	DA,0A,20		JNC	AHEAD	Yes, smaller number is in accumulator. Go to AHEAD.
2009	7E		MOV	A,M	No ,get 2nd number in accumulator
200A	32,03,25	AHEAD	STA	2503H	Store smaller number in 2503H.
200D	76		HLT		stop

EXAMPLE:

DATA:

2501-84H

2502-99H

RESULT:

2503-84H

Program to find the largest number in a data array

PROGRAM:

MEMORY ADDRESS	MACHINE CODES	LABELS	MNEMONICS	OPERANDS	COMMENTS
2000	21,00,25		LXI	H, 2500H	Address for count in H-L pair.
2003	4E		MOV	C,M	Count in register C.
2004	23		INX	H	Address of the 1st number in H-L pair.
2005	7E		MOV	A,M	1st number in accumulator.
2006	OD		DCR	C	Decrement count.
2007	23		INX	H	Address of next number.
2008	BE		CMP	M	Compare next number with previous maximum. Is next number**>** previous maximum.
2009	D2,0D,20		JNC	AHEAD	NO, Larger number is in accumulator. GO to the label AHEAD.
200C	7E		MOV	A,M	Yes, get larger number in accumulator.
200D	0D		DCR	C	Decrement Count.
200E	C2, 07, 20		JNZ	LOOP	
2011	32,04,25		STA	2504H	Store result in 2504H.
2014	76		HLT		Stop the Program.

Example-1:

Data:

2500→03

2501→98

2502→75

2503→99

Result: 2504→99

Program to find the smallest number in a data array

PROGRAM:

MEMORY ADDRESS	MACHINE CODES	LABLES	MNEMONICS	OPERANDS	COMMENTS
2000	21,00,25		LXI	H,2500 H	Get the address for count in the H-L pair
2003	4E		MOV	C,M	Count in register C.
2004	23		INX	H	Get address of 1st number in H-L pair.
2005	7E		MOV	A,M	1stnumber in accumulator.
2006	0D		DCR	C	Decrement count.
2007	23	LOOP	INX	H	Address of next number in H-L pair.
2008	BE		CMP	M	Compare next number with previous smallest. Is previous smallest < next no?
2009	DA,0D,20		JC	AHEAD	Yes, smaller number in the accumulator .Go to AHEAD.
200C	7E		MOV	A,M	No, get next number in accumulator.
200D	0D	AHEAD	DCR	C	Decrement count.
200E	C2,07,20		JNZ	LOOP	
2011	32,50,24		STA	2450 H	Store smallest number in 2450 H.
2014	76		HLT		Stop the program.

2.5 MEMORY AND I/O ADDRESSING-

Memory Addressing-

- A memory address is a unique identifier used by a device or CPU for data tracking.
- This binary address is defined by an ordered and finite sequence allowing the CPU to track the location of each memory byte.

- Modern computers are addressed by bytes which are assigned to memory addresses – binary numbers assigned to a random access memory (RAM) cell that holds up to one byte. Data greater than one byte is consecutively segmented into multiple bytes with a series of corresponding addresses.
- Hardware devices and CPUs track stored data by accessing memory addresses via data buses.
- Before CPU processing, data and programs must be stored in unique memory address locations.

OR

Memory Addressing:

- The bus determines a fixed number of CPU memory addresses assigned according to CPU requirements. The CPU then processes physical memory in individual segments.
- The operating system's read-only memory (ROM) basic input/output system (BIOS) programs and device drivers require memory addresses. Before processing, input device/keyboard data, stored software or secondary storage must be copied to RAM with assigned memory addresses.
- Memory addresses are usually allocated during the boot process. This initiates the startup BIOS on the ROM BIOS chip, which becomes the assigned address. To enable immediate video capability, the first memory addresses are assigned to video ROM and RAM, followed by the following assigned memory addresses:

- Expansion card ROM and RAM chips
- Motherboard dual inline memory modules, single inline memory modules or Rambus inline memory modules
- Other devices

I/O addressing:

- Input/output (I/O) port addresses are used to communicate between devices and software.
- The I/O port address is used to send and receive data for a component.
- As with IRQs, each component will have a unique I/O port assigned.
- There are 65,535 I/O ports in a computer, and they are referenced by a hexadecimal address in the range of 0000h to FFFF H.