TYBSc
Applied Component

Digital Electronics, Microprocessor and C++

Microprocessor 8085

Unit 2 & Unit 3

2016-2017

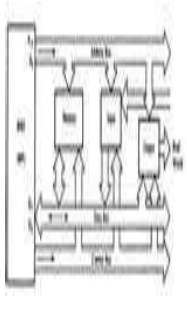
Pratibha P. Pai
Associate Professor

Department of Physics

SIES College of Arts, Science & Commerce

Sion(west)





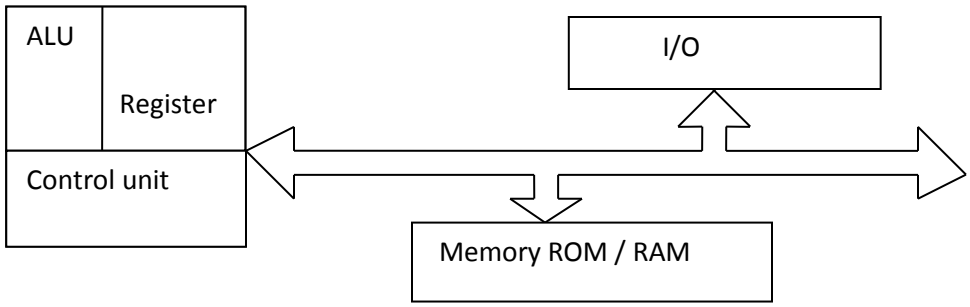

MICROPROCESSOR - 8085

A microprocessor is a multi-purpose, programmable, clock-driven, register- based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input, processes data according to the instructions and provides the result as output.

In general, a system consists of 3 components. **i) microprocessor ii) memory iii) I/O**. These 3 components interact with one another to perform a given task. The physical components of this system are called hardware. A set of instructions written for the µP to perform a task is called a program and a group of programs is called as software.

Microprocessor applications are broadly divided into 2 categories. a)**Reprogrammable systems** in which the microprocessor is accessible for reprogramming e.g microwave oven, washing machine etc. and b) **Embedded systems** in which the microprocessor is not accessible to the user. e.g traffic signal, smoke detector, factory siren etc.

The µP is a digital device designed with registers, flip-flops and timing elements. The µP has a set of instructions designed internally to manipulate data and communicate with peripherals. This process of data manipulation and communication is determined by the logic design of the µP. This design is called architecture.



General block diagram of a Micro computer

The µP can be programmed to perform functions on a given data by using necessary instructions from its set. These instructions are given to the µP by writing them into its memory. Writing (or entering) instruction and data is done through input device. e.g. key board(In µP, the key board is a hex pad containing 16 data keys and some additional function keys while in a computer, the key board is similar to that of a typewriter. It is an ASCII key board. The µP reads or transfers each instruction one at a time, matches it with the instruction set and performs data manipulation. Result can be sent to output device. e.g. LED,CRT terminal, printer, magnetic tape.

In addition, the µP can respond to external signals. It can be interrupted, reset or asked to wait to synchronize with slower peripherals.

All the functions performed by the µP are broadly classified as

i) µP initiated operations.
ii) Internal data operations.
iii) Peripheral (or externally initiated) operations.

µP initiated operations and 8085 bus organization

The µP performs mainly 4 operations. They are part of communication process between the MPU(microprocessor unit) and the peripherals. Following are the operations:

i) Memory Read : Reads data (or instructions) from memory.
ii) Memory Write : Writes data (or instructions) into memory.
iii) I /O Read : Accepts data from I /P devices.
iv) I /O Write : Sends data to O / P devices.

Types of buses and their structure -- with respect to 8085 µP

In 8085 microprocessor, there are 3 sets of communication lines called buses and for communication, the MPU needs to perform the following steps—

Step 1: Identify the peripheral or the memory location with its address

Step 2: Transfer binary information (data & instruction)

Step 3: Provide timing and synchronization signals

The 3 types of buses are the **address bus, data bus and the control bus**.

Address bus: This bus is a group of 16 lines identified as A_0 to A_{15}.The address bus is unidirectional.i.e from the MPU to the peripheral devices and the memory. The MPU uses the address bus to perform the first step – i.e identifying a peripheral or memory location.

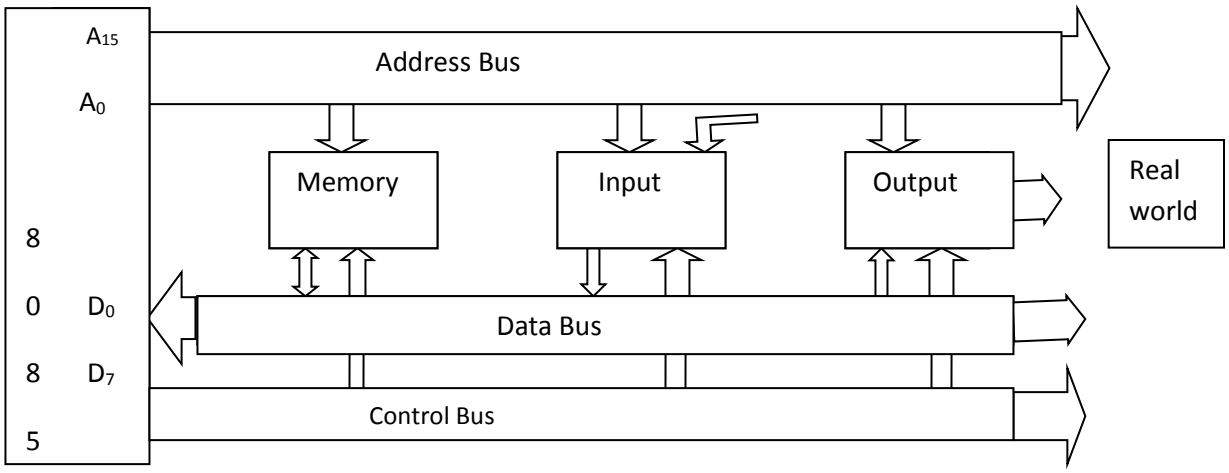
In computer system, each peripheral or memory location is identified with a binary number called an address and the address bus is used to carry a 16- bit address. With 16 lines, 8085 MPU is capable of addressing 2^{16} =65,536(generally known as 64 KB, 2^{10} = 1024 bytes= 1KB; 2^{12} = 4KB; 2^{20} = 1MB; 2^{30} = 1GB) memory locations. Most 8-bit µPs have 16 address lines.

Data bus: The data bus is a group of 8 lines D_0 to D_7 that allows data to flow from one chip to another. These lines are bidirectional i.e information on the data bus travels to and from the micro- processor; to and from the memory and the peripherals. The µP uses the data bus to

perform the second step – i.e transferring data. The data bus influences the architecture considerably. It determines the word length and the register size of the µP. 8085 is an 8- bit µP and the maximum number that can appear on the data bus is FFH ≡ 1111 $1111_2 \equiv 255_{10.}$

Control bus: This bus comprises of various single lines that carry synchronization signals. µP uses the control bus to perform the third step — i.e providing timing signals. The µP generates specific control signals for every operation it performs.(e.g memory read, I/O write)

To read an instruction from a memory location, the µP places the 16-bit address on the address bus. The address on the bus is decoded by an external circuit and the memory location is identified. The µP sends a pulse called Memory Read as the control signal. The pulse activates the memory chip. Then the contents of memory location are placed on the data bus and brought inside the µP. Following is the diagram.



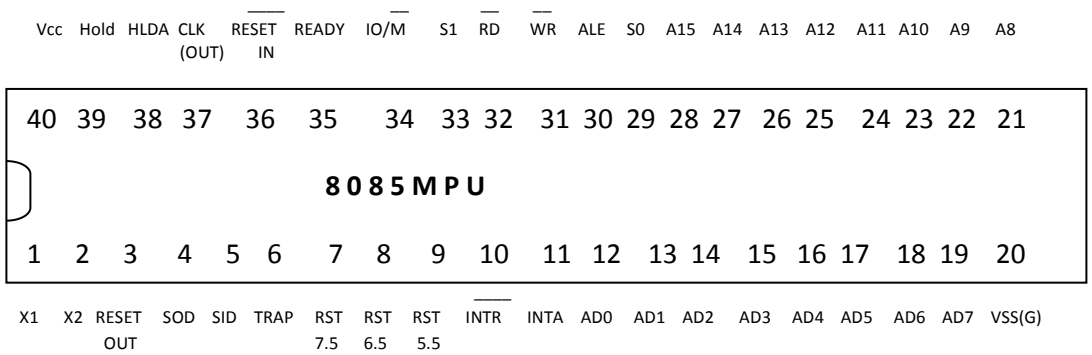
The 8080A MPU

The 8080A µP is the predecessor of 8085. The instruction set of 8080A and 8085 are practically the same. 8080A is not a complete functional unit as a processor. It requires 2 address chips, 8224 clock generator driver and the 8228 system controller and the bus driver.

8080A µP is manufactured on a single LSI chip using n- channel silicon gate MOS process. It is 40- pin dual-inline-package (DIP). It has 16 address lines, 8 data lines and requires 3 power supplies — +5V, - 5V ,+ 12V and a clock with 2 phases. The control signals are generated using system controller.

The 8224 clock generator is designed to provide a 2- phase clock. The crystal oscillator provides the basic frequency divided by 9 inside the chip. The 8228 system controller is a single chip integrated circuit. It is employed for 2 purposes.(1) As a bidirectional bus driver for the data bus and (2) as a controller to generate the necessary control signals.

The 40 pin diagram of 8085 and the functions of various pins



Important Features of 8085

- It is an 8-bit microprocessor.
- It is operated on a single +5V power supply.
- The operating frequency is 3MHz.
- It has 16 address lines, so can access 64Kbytes (2^{16}) of memory.
- The address and data bus are multiplexed.
- It provides 8 bit I/O addresses to access 256 ports.
- It supports 74 instructions.
- It has 6 general purpose registers, an accumulator, a flag register, two 16 bit registers(program counter and stack pointer)
- It provides 5 hardware interrupts **RST 5.5**, **RST 6.5, RST 7.5, TRAP and INTR.**

Functions: (number shown in brackets is pin number)

Functional unit 1: ***POWER SUPPLY AND CLOCK FREQUENCY***

Vcc(40):- 5V power supply.

Vss(20):-Ground reference.

X_1(1) & **X_2**(2):-A crystal is connected at these pins. The frequency is internally divided by 2. So to operate the system at 3 MHz, the crystal should have a frequency of 6 MHz.

CLK(out)(37):-This signal can be used as a system clock for other devices.

A_8 to A_{15}(21 to 28):-These are 8 signal lines. These are uni –directional and used as the high order address bus.

AD_0 to AD_7(12 to 19):-These are 8 signal lines. These are bi-directional and serve a dual purpose. They are used as low order address bus as well as the data bus.

Functional unit 2: CONTROL AND STATUS SIGNALS

This group of signals includes 2 control signals ($\overline{RD}$ & $\overline{WR}$), 3 status signals (IO/$\overline{M}$, S_1 & S_0) to identify the nature of operation and one special signal(ALE) to indicate the beginning of the operation.

$\overline{RD}$(32) Read :-This is a READ control signal(active low).This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.

$\overline{WR}$(31)Write:-This is a WRITE control signal(active low).This signal indicates that the data on the data bus are to be written on a selected memory or I/O location.

IO/$\overline{M}$(34):-This is a status signal used to differentiate between I/O & memory operations. When it is high, it indicates I/O operation. When it is low, it indicates memory operation. This signal is combined with RD & WR to generate I/O and memory control signals.

S_0(29) & **S_1**(33):- These are status signals. They can identify various operations but are rarely used in small systems

ALE(30) Address Latch Enable:-This is a positive going pulse generated every time the 8085 begins an operation.(machine cycle).It indicates that the bits AD_0 to AD_7 are address bits. This is primarily used to latch the low order address from the multiplexed bus & generate a separate set of 8 address lines A_0 to A_7.

Functional unit 3: INTERRUPTS & EXTERNALLY INITIATED OPERATIONS

8085 has 5 interrupt signals that can be used to interrupt a program execution.

The µP can be interrupted from the normal execution of instructions and asked to execute some other instructions. When it is complete, the µP resumes its operation.

INTR(10) Interrupt Request:- This is used as a general purpose interrupt.

INTA(11) Interrupt Acknowledge:- It acknowledges the interrupt request.

RST 5.5(9) , **RST 6.5**(8) & **RST 7.5**(7) Restart Interrupts:-These are vectored interrupts and transfer the program control to specify memory locations. They have higher priority than INTR. The order of priority is 7.5, 6.5, 5.5.

TRAP(INPUT)(6):-This also has a high priority. This is a non-maskable interrupt.

In addition to the interrupts, 3 pins- Reset, Hold, Ready - accept the externally initiated signals as inputs. **Reset , when activated, all internal operations are suspended and the program counter is cleared (i.e it holds 0000H).Now the program execution can again begin at the zero memory address.**

RESET IN(36):-When the signal at this pin is low, the program counter is set to zero and the MPU is reset.

RESET OUT(3):-This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

HOLD(39):-It accepts the externally initiated signals as inputs. When this pin is activated, the µP allows the external peripheral to use the control bus.

READY(35):-It is used to delay the µP READ or WRITE cycles. If the signal at this pin is low, the µP enters into a WAIT state. This signal is used primarily to synchronize slower peripherals with the µP.

HLDA(38) Hold Acknowledge:-This signal acknowledges the HOLD Request.

Functional unit 4: ***SERIAL I/O PORTS***

8085 has 2 signals to implement serial transmission.

SID(5) Serial Input Data } The SID & the SOD lines in 8085 eliminate the need for an

SOD(4) Serial Output Data } i/p port & an o/p port in the software controlled serial I/O.

The SID is a 1-bit i/p port and SOD is 1-bit o/p port.

µP Architecture , internal data operations and 8085 registers.

The internal architecture of 8085 µP determines how and what operations can be performed with the data. These operations are listed as follows:

1. Store 8 –bit data. ***2. Perform arithmetic & logical operations.***

3. Test for conditions. ***4. Sequence the execution of instructions.***

5. Store data temporarily during execution in the defined R/W memory locations called the stack.

The above operations are performed with data using ALU, registers, control logic and internal buses.

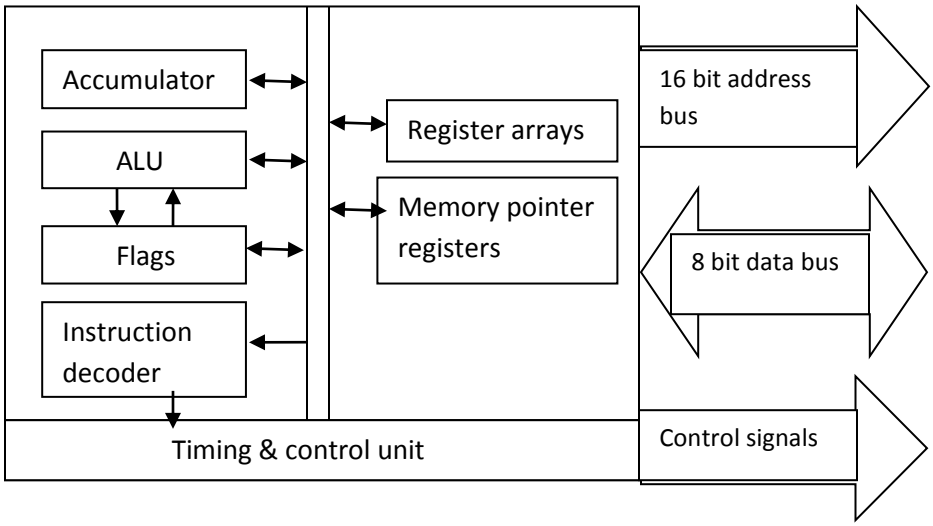
The 8085 Model

A model is a conceptual representation of a real object. It can take many forms, such as a text description, a drawing or a built structure. The µP can be represented in terms of hardware model and programming model.

8085 Hardware Model

There are 2 major segments in this model. One segment includes ALU, an 8 bit register called Accumulator, instruction decoder and flags. The second segment has 8bit and 16 bit registers. Both the segments are connected with various internal connections called an internal bus. Arithmetic and logic operations are performed in the ALU and the results are stored in the accumulator and the flags.

Following is the 8085 hardware model:



The 8085 Programming Model -Programmable Registers with number of bits in brackets

Accumulator A (8)	Flag register(8) S | Z | - | AC | - | P | - | CY
B (8)	C (8)
D (8)	E (8)
H (8)	L (8)
Stack pointer SP (16)	
Program counter PC (16)	

data Bus (8 lines) Address Bus(16 lines)

This model does not reflect the physical structure of the 8085 but includes the information that is required in writing assembly language programs. The model includes 6 registers, one accumulator, one flag register, 16 bit program counter and 16 bit stack pointer.

The ALU:- ALU stands for Arithmetic and Logic Unit. It is one of the major parts of the µP. It performs the computing functions on 8-bit data. It includes the **accumulator, the temporary registers, the arithmetic and the logic circuits and the flag register.** The temporary register is used to hold data during an arithmetic and logic operation. The result is stored in the accumulator and the flags (actually the flip flops) are set or reset according to the result of the operation. Flags generally reflect data conditions in the accumulator. **ALU can perform the following arithmetic and logical functions —**

i) add 8 bit/16 bit data ii) subtract 8 bit data iii) logical AND, OR, XOR, NOT(complement) iv) shift left/right , rotate v) increment/decrement etc.

ALU does not store data. It only processes data. The data size is 8 bits and so operation on 8 bit data can only be performed.

The Accumulator:- The accumulator is a storage place or a register. It is an 8-bit general purpose register. Sometimes it is referred to as register A. It is a part of the ALU. It is used to store data (first operation in the list above) and perform arithmetic & logical operations (second operation in the list).The result of the operation is stored in the accumulator.

Temporary registers:- The second input to ALU is given by the temporary registers. These are not available to the user. These registers are named as w and z registers. They are used by µP for internal operations such as to store operand, immediate operand or address of memory.

Status register or Flag register:- This is a special register which keeps track of certain facts about the outcome of arithmetic, logical and other operations. This register makes it possible for the µP to be able to test for certain conditions (third operation in the list) and then perform alternate functions based on those conditions. The flag register is adjacent to the accumulator. It is also called as status register. It is an 8-bit register containing 5 flags. These are flip-flops which are set or reset according to data conditions in the accumulator. They are as follows:- **Sign flag(S), Zero flag(Z), Auxiliary carry flag(AC), Parity flag(P) and Carry flag(CY).**The position of the flags are as shown where — ⇒ unspecified bit.

Bit position→	D7	D6	D5	D4	D3	D2	D1	D0
Symbol →	S	Z	—	AC	—	P	—	Cy
Name of the flag →	Sign	Zero	unused	Auxiliary	unused	Parity	unused	Carry

Sign flag:- After the execution of arithmetic or logic operation, if the bit D_7 is 1,the sign flag is SET(S=1),otherwise it is RESET(S=0). When 8-bit signed binary numbers are used if after the operation, the D_7 bit is 1, then the number is negative. If D_7 bit is 0, the number is positive. (The sign of a number is indicated by MSB bit. Remember: "**Sign flag is SET on negative numbers.**")

Zero flag:- If the result of an ALU operation is 0 i.e $0000\ 0000_2$, then Z flag is SET(Z=1) otherwise it is RESET(Z=0). Ask the question **"Is the result zero?"** If the answer is 'YES', then Z=1.

Auxiliary carry flag:- This is also called as Half carry. This flag is used internally for BCD (Binary Coded Decimal) numbers. In an arithmetic operation, when a carry is generated by D_3 bits and passed on to D_4 position,(i.e from lower nibble to higher nibble) then AC flag is SET(AC=1).Otherwise it is RESET(AC=0). This flag is not available for the programmer.

Parity flag:- This indicates the parity of the result. After an arithmetic or logic operation, if the result contains even number of 1s, parity flag is SET(P=1).If it has odd number of 1s, then parity flag is RESET (P=0).

Carry flag:- If the arithmetic operation results in a carry (i.e D_8 position , resulting number has 9 bits), the CY flag is SET(CY=1), otherwise it is RESET(CY=0).The carry flag serves as borrow flag during subtraction.

Most commonly used flags are CY, S and Z.

General-purpose registers (Register array):- There are 6 general purpose registers. They are B, C, D, E, H and L. They are all 8- bit registers. They can be combined as register pairs like BC, DE & HL. The register pair can store 16 –bit data. These registers are programmable i.e a programmer may use them to load or copy the data from the registers by using respective instructions.

The HL register pair is usually used for a different purpose than the BC & DE pairs. w and z are temporary registers used to hold 8 – bit data during the execution of some instructions. They are not available to the programmer.

Special purpose registers: There are 3 special purpose registers in 8085. They are PC, SP and increment/decrement latch.

Program counter (PC) & Stack Pointer(SP):- They are part of register array. **Both are 16 bit registers.**

Program counter deals with sequencing the execution of instructions (fourth operation in the list). Memory locations have 16- bit addresses & this register is a memory pointer. The function

of PC is to "point" to the memory address from which the next byte is to be fetched and executed. When a byte is being fetched, the PC is incremented by 1 to point to the next memory location.

When the µP is reset, the content of PC is 0000H and so execution always starts from 0000H.

The stack is a special place in memory. SP points to a memory location in the R/W memory called the stack. It is used as a memory pointer to define the stack starting address.(top of the stack) The beginning of the stack is defined by loading a 16 –bit address in the stack pointer. The structure of the stack is a First- In – Last- Out (FILO type).

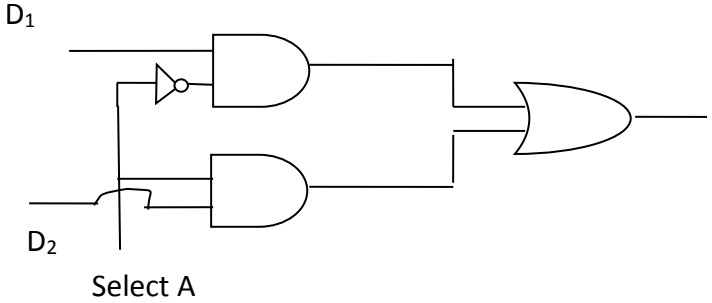
increment/decrement latch : This is again a 16 bit register used to increment or decrement the contents of PC and SP registers.

Timing & control unit:- This unit synchronizes all the µP operations with the clock and generates the control signals necessary for communication between the µP and the peripherals. The control signals are similar to a synchronizing pulse in an oscilloscope.

Instruction register & Decoder:- Both are part of ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequence of events to follow. The instruction register is not programmable and cannot be accessed through any instruction.

Need to multiplex address and data bus in 8085

Let us first understand what a multiplexer (Mux) means. Basic Mux is 2:1.
A multiplexer has many inputs but only one output. By using proper control input, only one of the many inputs can be steered at the output. It is also called as data selector.



G1 & G2 are AND gates. G3 is an OR gate. A is control signal. D1 and D2 are i/ps. Y is o/P.

If A=0, Y=D1. If A=1, Y=D2.i.e by controlling A, data D1 or D2 is passed at the o/p.

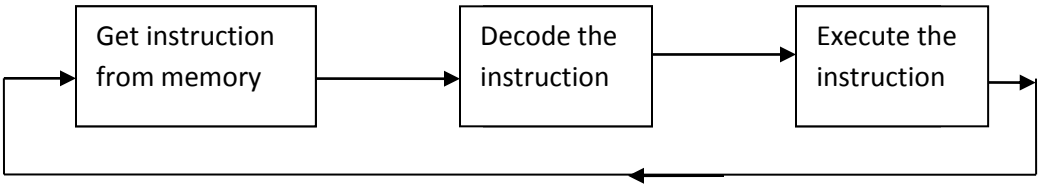
In 8085, the signal lines AD_7 to AD_0 are bidirectional, they serve a dual purpose. They are used as lower order address bus as well as data bus. A_{15} to A_8 are unidirectional. They are used as high order address bus. In executing an instruction, during the earlier part of the cycle, AD_7 to

AD_0 are used as low order address bus while during the latter part of the cycle, these lines are used as data bus. This process is known as multiplexing the bus.
Multiplexing helps to reduce the number of pins in the IC. The disadvantage is that because of time sharing, the speed decreases.

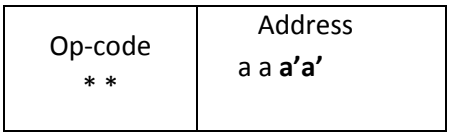
INSTRUCTIONS and their CLASSIFICATION

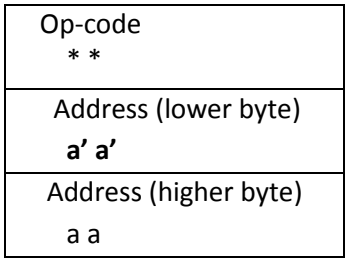
A µP instruction is a binary word. When it is read as an instruction, this binary word tells the µP to do a simple task. The set of all such instructions the µP knows how to use is known as instruction set .Each µP has its own, unique instruction set. This means that the instruction set of 2 different µPs is different.

In an 8 –bit µP, the instruction word is 8- bit (8 bits make 1 byte) long. But a complete instruction may take 1, 2 or 3 words. During the execution of the instruction, the following steps are followed – **fetch, decode and execute**.



An instruction can be broken into 2 parts, called the operation code (op-code in short) and the address. The op-code tells the µP what to do and the address tells the µP where to take the action.





Mnemonics

We know that a µP instruction is a binary number. It is difficult to remember instructions consisting of 2, 3 or 4 byte binary numbers. It is easier with hexadecimal numbers but we do not understand what they stand for. So, we use what are called as Mnemonics. A mnemonic is an abbreviation that reminds us of what it stands for .In other words, partly the meaning in English, the language we understand. In most instruction code mnemonics, the op-code is abbreviated to 3/4 letters. Mnemonics in 8085 programming are written in upper case letters.

e.g:-

Machine language	Hex code (op-code)	Mnemonics (assembly language)	Operand
1000 0000	80	ADD	B
0100 1111	4F	MOV	C, A
0011 0010	32	STA	Address aaa'a'

The µP does not process mnemonics, op-codes or hex numbers. It works only on binary words. An assembler is a program(software) that converts mnemonics(assembly language) and symbolic addresses into binary instructions and binary addresses. Thus, before the instructions are executed in a program, all the shorthand notations must be converted into binary numbers. The set of binary numbers which make up a program(object code) is called a machine language program.

e.g- store the data in accumulator to the memory location C220H.
What we write: STA C220H
For the CPU :32, 20, C2
(32 is the op-code for STA followed by the lower order address and then the higher order address)

Programming languages

A programming language is a special language used to write µP /computer programs. There are 2 categories of programming languages. They are -- low level and high level. Low level language is machine dependent while high level language is machine independent. In brief,

a) Machine level language : e.g. 1010 1100, 1110 0011, 1000 0001 etc.

- Only language µP / computer directly understands.
- "Natural language "of µP / computer.
- Defined by hardware design--- machine dependent.
- Generally consist of strings of numbers – ultimately 0s and 1s.
- Instruct µP /computers to perform elementary operations—one at a time.
- Cumbersome for humans.

b) Assembly level language: e.g. LDA C200H, ADD B, INR A

- English –like abbreviations representing elementary µP / computer operations.
- machine dependent.
- clearer to humans.
- cannot be understood by µP / computer.
- So require Translator program (called assembler or cross- assembler), converts

mnemonics to machine language.

c) High level language: e.g. BASIC, COBOL, PASCAL, C, C++, JAVA etc.

- Similar to everyday English, use common mathematical notations.
- Single statements accomplish substantial tasks.
  (Assembly language requires many instructions to accomplish simple tasks.)
- Translator programs(compilers)(Convert to machine language).
- Interpreter programs(Directly execute high- level language programs).
- Machine independent.
- Compiler & Interpreter require large memory space (because instruction in English requires several machine codes to translate in binary).
- Translation: source code→ Compiler / Interpreter→ Object code.

µP instruction format and width of an instruction

An instruction is a command to the µP to perform a given task on a specified data. Every instruction consists of 2 parts (i) task to be performed called the op-code and (ii) the data to be operated on called the operand. The op-code tells the µP what to do while the operand tells the µP where to take the action. The operand may be 8- bit data, an internal register, a memory location or a 16- bit address. In some instructions, the operand is implicit.

Width of an instruction means the number of memory locations the instruction occupies. Each memory location can store only one byte. This byte may be op-code, data, lower byte of the address or the higher byte of the address.

One byte instructions : -

These instructions occupy one memory location. It includes the op-code and the operand in the same byte. e.g MOV R_d ,R_s; MOV R , M; ADD R; SUB R; INR R; DCR M; HLT; ORA R; XRA R; etc. Op-code for HLT is 76.

R_d is the destination register, Rs is the source register (both are any one of registers A, B, C, D, E, H, L), R any register, M memory location pointed to by the HL register pair.

Two byte instructions:-

These instructions occupy two memory locations. The first byte specifies the op-code and the second byte necessarily specifies the 8-bit data. e.g MVI R_d, 8-bit ; ADI 8-bit ;

SUI 8-bit; ANI 8-bit; ACI 8-bit etc.

Task	Mnemonic	operand	Width-2 byte
Load 8-bit data into register B	MVI	B,28H	06 op-code 28 data
OR immediately 8-bit data with the content of accumulator	ORI	CBH	F6 op-code CB data

H stands for Hex and the Op-code is in hex.

Three byte instructions:-

These instructions occupy 3 memory locations. The first byte specifies the op-code , the following 2 bytes specify the 16-bit address. Of the 2 bytes, the first one is the lower order address and the second byte is the higher order address. e.g LXI R_p,16-bit; STA 16-bit; LDA 16-bit ; JMP 16- bit; all conditional jump instructions like JNC, JNZ, JC, JZ, JPE, JPO…

Task	Mnemonic	operand	Width -3 byte
Store the accumulator content in the memory location D125H	STA	D1 25	32 op-code 25 Low byte, D1 High byte
Jump if no zero to C210H	JNZ	C2 10	C2 op-code 10 Low byte, 11 C2 High byte

---

FUNCTIONAL CLASSIFICATION

According to the functions the instructions perform, they can be classified into 5 categories. They are as follows.

(1) Data transfer instructions (2) Arithmetic instructions

(3) Logical instructions (4) Branch instructions and

(5) Stack, I/O, machine control instructions.

Apart from these, in some µPs i) exchanges, block transfers & searches ii) rotates and shifts iii) bit set, reset, bit test iv) calls, returns & restarts are taken as extra classification.

1.Data transfer instructions :- These group of instructions transfer (technically copy data) data from one place (called source) to another place(called destination), without modifying the content of the

source. The data can be transferred from register to memory, memory to register, register to register etc.

MOV Rd, Rs	Copy data from Rs(source register) into Rd(destination register)
MVI Rd, 8 bit	Load 8bit data in the register Rd
LXI Rp,16-bit	Load 16bit data in the register pair
MOV R,M	Copy data byte from memory location into a register
MOV M,R	Copy data byte from register into memory location
STA 16-bit	Copy data byte from accumulator into memory location specified by 16 bit address
LDA 16-bit	Copy data byte into accumulator from the memory location specified by 16 bit address

2. **Arithmetic instructions**:- These group of instructions perform arithmetic operation on data. The arithmetic operation may be addition, subtraction, increment, decrement etc.(8085 does not have multiplication and division instructions, multiplication is repeated addition while division is repeated subtraction.)

Any 8- bit number or contents of a register or the contents of a memory location can be added to /subtracted from the contents of accumulator and the result stored in accumulator. **Register to register addition or subtraction is not allowed**. Subtraction is performed in 2's complement form. The 8-bit contents of register or memory location and the 16-bit contents of register pair can be incremented or decremented by 1

ADD R	Add contents of register to contents of accumulator
ADI 8bit	Add 8bit data to contents of accumulator
SUB R	Subtract contents of register from the contents of accumulator
SUI 8bit	Subtract 8bit data from contents of accumulator
INR R	Increment contents of register
DCR R	Decrement contents of register
ADD M	Add contents of memory location to contents of accumulator
SUB M	Subtract contents of memory location from contents of accumulator
INR M/DCR M	Increment/decrement contents of memory location
INX R_p/DCX R_P	Increment/decrement contents of register pair(BC, DE, HL)

3. **Logical instructions**:- These group of instructions perform various logical operations with the contents of accumulator.

AND, OR, XOR:- 8-bit number or content of register or memory location can be logically ANDed,ORed, or Xclusively ORed with the content of accumulator and the result stored in the accumulator.(The process is by bit-by-bit ANDing, ORing, XORing)

Rotate and shift:- Each bit in the accumulator can be shifted left or right by one position(RLC-rotate accumulator left, RRC-rotate accumulator right, RAL-rotate accumulator left through carry and RAR-rotate accumulator right through carry).

Compare:- 8-bit number or contents of register or content of memory location can be compared (═ ,> ,<) with the content of accumulator.

Complement:-Content of accumulator can be complemented (0 to 1 and 1 to 0).

ANA R/M (ORA, XRA)	AND accumulator content with content of register /memory location
ANI 8bit (ORI, XRI)	AND 8bit data with accumulator content
CMA	Complement content of accumulator
CMP R/M	Compare contents of register/memory location with content of accumulator
CPI 8bit	Compare 8bit data with the content of accumulator

4. **Branching instructions**:- These instructions change the sequence of program execution conditionally or unconditionally. In the former case, control is transferred to another memory location only if a particular condition is satisfied.(e.g zero or carry flag).In the latter case ,without any condition, control is transferred to another location.(all 16-bit addresses).

JMP 16bit address — change the program sequence to the location given by the 16bit address.

JZ (jump if zero), JNZ(jump if no zero), JC(jump if carry), JNC(jump if no carry), JPO(jump if parity odd), JPE(jump if parity even), JP(jump if plus) and JM(jump if minus) all with 16-bit addresses.

Call return and restart:-These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. e.g CALL 16 bit address, RET

5**. Machine control instructions**:- These instructions control machine functions such as Halt(HLT), Interrupt or No operation performed(NOP).These are used to produce delay time.

Note: While subtraction, the carry flag is actually the borrow and microprocessor shows the complemented borrow i.e if cy=0, borrow is present while if cy=1, borrow is absent.

ADDRESSING MODES IN 8085— CLASSIFICATION

An instruction consists of op-code followed by zero, one or two operands. While the op-code specifies the operation to be performed, the operand specifies the source or destination of data operated on. The operand can specify µP register, memory location or Input/Output port. The various ways in which the µP generates these operand addresses are called addressing modes.

{ source ≈ from where the data is transferred. Destination ≈ to where the data is transferred. Source and destination are together called operand.}

There are 5 addressing modes in 8085 µP. They are **i)Immediate addressing ii)Direct addressing iii)Register addressing iv) Indirect addressing and v)Implied addressing.**

1.Immediate addressing mode:- In this mode, the operand is specified within the instruction itself. These are all 2-byte instructions. The first byte is the op-code and the second byte is the 8-bit data.(In the mnemonic 'I' is seen.)

e.g MVI A,56H ;MVI B,00H

2.Direct addressing mode:- In this mode, the address of the operand is given in the instruction itself. These are all 3-byte instructions.(16 bit address is seen here.)

e.g STA C050H ;LDA C050H

3.Register addressing mode:- In this mode, the name of the register in which the data is stored is given in the instruction itself. They are 1-byte long.(register name is seen.)

e.g MOV B,C ;SUB D

4.Indirect addressing mode:- In this mode, the address of the operand is given indirectly. i.e the address is stored in a particular register pair or a memory location.

(M is seen here or register pair).These are 1-byte long.

e.g ADD M, DCR M, MVI M,2BH

5.Implied addressing mode:- In this mode, operand is not needed. It is implied within the mnemonic itself. These are 1- byte long instructions.

e.g STC-set carry flag

Examples of different modes(figures used are just for examples)

Immediate	MVI B,23H; LXI H,C550H; MVI C,00H; SUI ABH; ADI 12H; ANI 16H; XRI B2H; MVI M,90H; CPI 25H ; J condition address
Direct	STA C000H; LDA C770H; LHLD D855H; SHLD D010H
Register	**MOV** B,C; SUB B; ADD A; ADC B; INR B; DCR D; ANA C; CMP B; INX H; ADD C;
Indirect	MOV M,A; MOV A,M; ADC M; LDAX B; STAX D; ADD M; SUB M; PUSH B; ANA M; DCR M; INR M; MVI M,25H
Implied	STC; HLT; RAL; RAR; RLC; RRC; CMA;NOP;XCHG;RET;CMC;RST

Examples of different functions

Data Transfer	MOV R_d,R_s; MOV R,M; MOV M,R; MVI R,data; MVI M,data; LXI R_p,16bit data; LDA aaaa; STA aaaa; LHLD aaaa; SHLD aaaa; LDAX R_p; STAX R_p; XCHG; IN 8bit address; OUT 8bit address
Arithmetic	ADD R; ADD M;ADC R; ADC M; ADI data; ACI data; DAD R_p; SUB R; SUB M; SBB R; SBB M; SUI data; SBI data; DAA; INR R; INR M; DCR R; DCR M; INX R_p; DCX R_p
Logical	ANA R;ANA M; ANI data; ORA R; ORA M; ORI data; XRA R; XRA M; XRI data; CMA; CMC; STC; CMP R; CMP M; CPI data; RLC; RRC; RAL ;RAR
Branch	JMP address; conditional jump instructions; PCHL; CALL address; conditional call instructions; RET; RST N(restart instructions)
Machine Control & stack	HLT; NOP; PUSH R_p; POP R_p; SPHL; XTHL; EI; DI; RIM; SIM

WRITING AND EXECUTING AN ASSEMBLY LANGUAGE PROGRAM

We know that a program is a set of logically related instructions written in a specific sequence to accomplish a task. To manually write and execute an assembly language program on a single board computer, with a hex keyboard for input and LEDs for the output, the following steps are necessary:

1.Write the instructions in mnemonics obtained from the instruction set supplied by the manufacturer.
2.Find the hexadecimal machine code for each instruction by searching through set of instructions.
3.Enter (load) the program in the user memory in a sequential order by using the hex keyboard as the input device.
4.Execute the program by pressing the 'execute' key. The answer will be displayed by the LEDs. This procedure is called either manual or hand assembly.

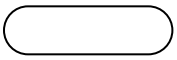
When the user program is entered by the keys, each entry is interpreted and converted into its binary equivalent by the monitor program, and the machine code is stored as 8 bits in each memory location in a sequence. When the Execute command is given, the microprocessor fetches each instruction, decodes it and executes it in a sequence until the end of the program.

FLOW CHART

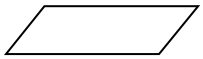
The set of instructions to perform a task is written in a sequence. This sequence is known as a program. The steps necessary to write a program can be represented in a pictorial format called flow chart. It represents the logical approach in solving the problem. A flow chart is similar to a block diagram, representing the structure of the program. Flow charts break large, complex programs into smaller logical units and make it easier, to make changes and corrections. Flow chart is used for 2 purposes-

1.To assist and clarify the thinking process 2. To communicate the programmer's logic to others.

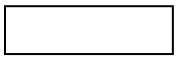
Following symbols are used in flow charting-



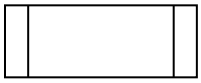
start or end



input or output

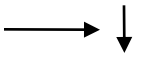


process



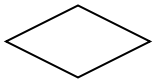
predefined process

(subroutine)



flow lines

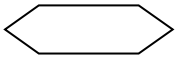
comment



Decision box

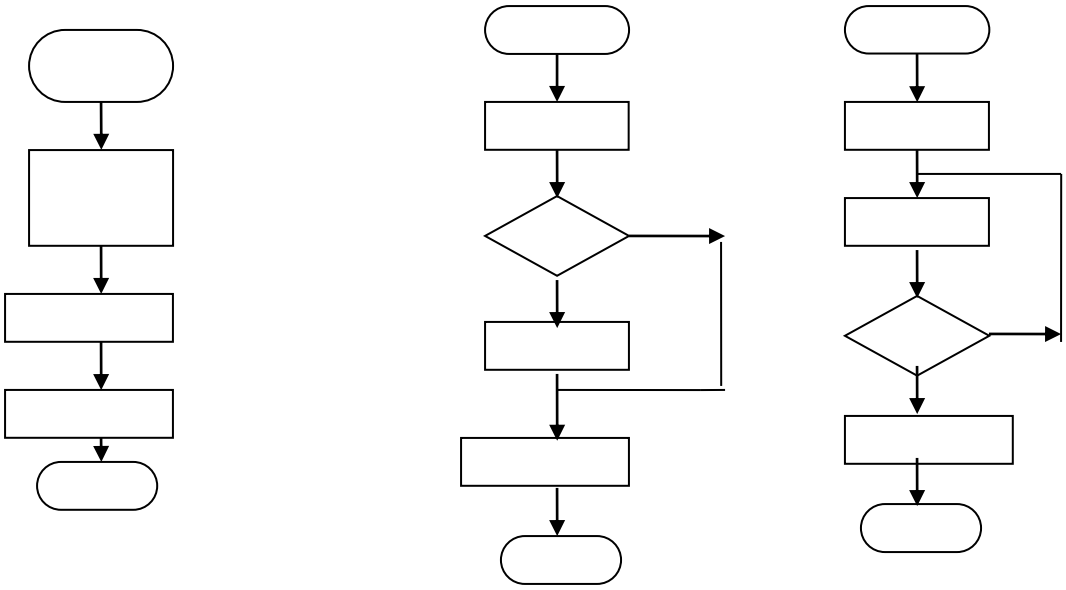
connector to connect parts in different pages

connector to connect different parts in the same page



Preparation

There are 3 types of flow charts. They are as shown below.



fig(1) fig(2) fig(3)

fig(1) shows a linear sequence. fig(2) shows a forward branching(if-then-else type).

fig(3) shows a reverse branching(repeat until type).

LOOPING, COUNTING & INDEXING:-

The programming technique used to instruct the µP to repeat tasks is called looping. A loop is set up by giving instructions to the µP to change the sequence of execution and perform the task again. It is achieved by using jump instructions. The loop is finally broken or exited from, when the condition is met.

In addition, techniques such as counting & indexing are also used in setting a loop. Loops can be classified into 2 groups. i)**continuous loop**(unconditional branching i.e repeat tasks continuously) and ii) **conditional loop**(conditional branching i.e repeat tasks until certain data conditions are met).

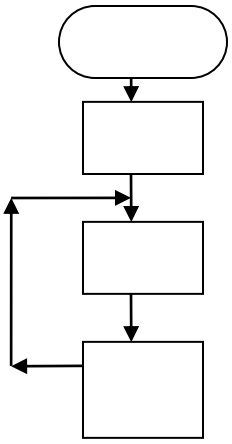
Continuous loop:-

This is the simplest type of branching. Here, the program will jump to the indicated memory location every time this part of the program is run. The jump can be forward or backward. The task is repeated continuously.

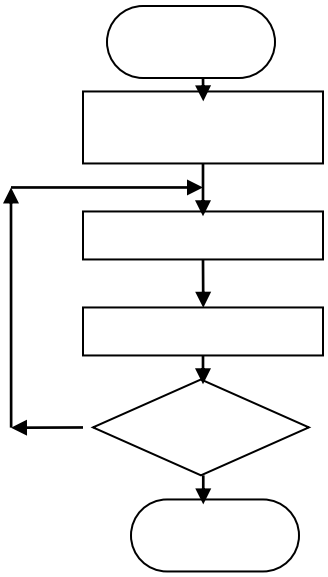
A backward jumping instruction forms an infinite loop. To come out of such a loop, the user has to reset the system.

For forward jumping, the user has to simply indicate the address of the next instruction to be executed. e.g JMP aaaa; aaaa stands for 16bit address and JMP instruction is a 3-byte instruction.

e.g:- see fig (4)



fig(4)



fig(5)

Conditional loop:- Here the task is repeated till certain conditions are met. These are set up by conditional Jump instructions. These instructions check the flags (S, Z, CY etc). These loops include counting and indexing.

Counter:- This is a backward jumping loop. Here,

i)counter is set up by loading an appropriate count in a register.

ii) counting is performed by incrementing or decrementing the counter.

(It is always easy to count down than to count up as the zero flag is set when the counter becomes zero. With up counting, one has to use 'compare' instruction.)

iii) A loop is set up by a conditional jump instruction.

iv)End of counting is indicated by a flag.

e.g :see fig(5) program for adding the number 2 five times where S is the sum.

Indexing with a counter:- This is another type of loop where conditional jump instructions are used. Indexing means pointing to or referencing objects with sequential numbers. e.g In a book store or a library where books are stored, numbers are given to shelves or books. Books are sorted out according to numbers .This is indexing.

In a µP where data are stored, the data are referred to by their memory locations. It is to be noted that the conditional jump instructions allow the µP to make decisions based on conditions indicated by the flags. After the logic and arithmetic operations, the flip flops are set or reset to reflect data conditions. The conditional jump instructions check the flag conditions and make decisions to change or not to change the sequence of a program. The 4 flags used by jump instructions are CY, Z, S and P. (AC flag is used internally.)With each flag, 2 jump instructions are associated.

Form:- J condition aaaa . C220H is 16 bit address taken as an example.

Sr. no	Mnemonic operand	description
1	JNZ C220H	Jump if no zero (Z=0)
2	JZ C220H	Jump if zero (Z=1)
3	JNC C220H	Jump if no carry (CY=0)
4	JC C220H	Jump if carry (CY=1)
5	JPO C220H	Jump if parity odd (P=0)
6	JPE C220H	Jump if parity even (P=1)
7	JP C220H	Jump if plus (S=0)
8	JM C220H	Jump if minus (S=1)

All the conditional jump instructions are 3-byte instructions. The first byte is the op-code, the second byte is the lower byte address and the third byte is the higher byte address.

———————————————————————

ADDITIONAL DATA TRANSFER & 16-BIT ARITHMETIC INSTRUCTIONS

(data transfer between micro-processor and memory)

Following are the 4 types of data transfer and 2 types of arithmetic instructions that we come across.

1.Loading 16-bit data in register pair (R_p):-

LXI R_p : Load register pair immediate. Here, R_p can be B, D, H or stack pointer.

(LXI B ,16bit; LXI D, 16bit; LXI H, 16bit; and LXI SP,16bit)

e.g LXI B,C550H . These instructions are all 3 byte long. 2nd byte is loaded in low-order register of the pair(C here) .3rd byte is loaded in high-order register pair(B here).

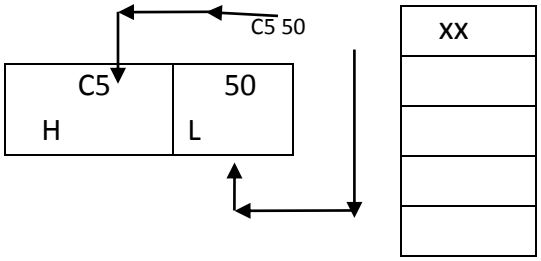
LXI B,C550H ⟹ Load the address C550H in B and C registers with C5 in B and 50 in C.

Similarly LXI H, C550H and LXI D,C550H (C550H is just an e.g.)

Difference between LXI and MVI:

LXI H,C550H ⟹
Uses 3 bytes

MVI H,C5 } uses 4 bytes
MVI L,50 }



A program to store the data A5H in the memory location C110H:

```
LXI H, C110H
MVI M, A5H
HLT
```

2. Data transfer from memory to µP:-

MOV R,M : move from memory to register
LDA 16bit : load accumulator direct
LDAX B/D : load accumulator indirect

MOV R,M—are 1-byte instructions. The addressing mode is indirect. M stands for the memory location pointed by the HL register pair.

(MOV A,M; MOV B,M; MOV C,M; MOV D,M; MOV E,M; MOV H,M; MOV L,M)

The instruction copies the data from the memory location to a register.

LDA 16bit:- This is 3 byte instruction. The addressing mode is direct. It copies the data from the memory location specified by the 16bit address into the accumulator.

LDAX R_p:- These are 1-byte instructions. The addressing mode is indirect.

(LDAX B; LDAX D)

It copies the data from the memory location given in BC or DE pair into the accumulator.

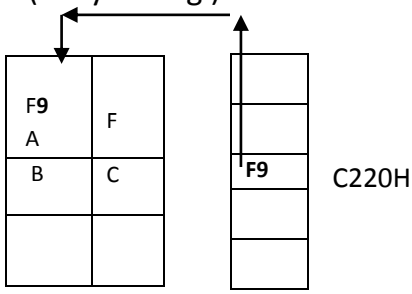
Understanding:

Memory location C220H contains data A9H .Transfer the data into the accumulator .

a) using LDA **b) using MOV and** **c) using LDAX instructions**.

a) LDA C220H

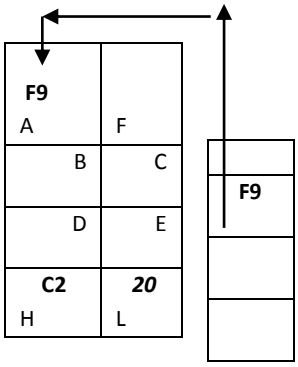
(3 byte long)



b) LXI H,C220H

MOV A,M

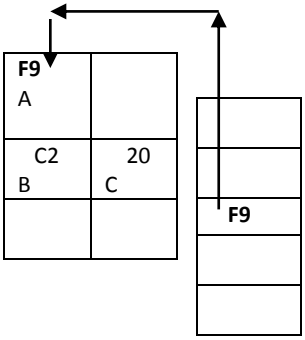
(4 byte long)



c)LXI B,C220H

LDAX B

(4 byte long)



e.g- Write a program to copy the data 89H stored in the memory location(C125H) mentioned in the BC pair to the accumulator.

```
LXI H, C125H        ;load the address
MVI M ,89H          ; load the data
MVI B,C1H
MVI C, 25H
LDAX  B
HLT
```

3.Data transfer from µP to memory:-

MOV M, R: move from register to memory

STA 16bit: store accumulator direct

STAX B/D: store accumulator indirect

MOV M,R:- is one byte instruction. Indirect addressing mode. M is the memory location specified by the contents of HL registers.

(MOV M,A; MOV M,B; MOV M,C; MOV M,D; MOV M,E; MOV M,H;

MOV M, L; MOV M,M)

This instruction copies data from the register R into the memory location specified by the HL pair.

STA 16bit:- This is a 3-byte instruction. Direct addressing mode. This instruction stores(copies) the accumulator contents in the memory location whose address is given by the 16 bit operand.

STAX Rp:- These are 1 byte instructions. Indirect addressing mode. R_p can be BC or DE register pair. This instruction copies the accumulator contents into the memory location specified by the contents of BC or DE registers.

e.g- Register B contains B8H. Copy the data into D120H using MOV and STAX.

a) LXI H,D120H
MOV M,B
(4 byte long)

b) LXI D, D120H
MOV A, B
STAX D
(5 byte long)

e.g- Accumulator has ABH . Copy the data into memory location C550H using direct as well as indirect mode

a) direct: STA C550H.

b) indirect: LXI B, C550H → immediate addressing mode
STAX B → indirect addressing mode

e.g- A program to move the content in accumulator (50H) to the memory location (DO15H) whose address is in DE pair-

```
MVI A, 50H
MVI D, D0H
MVI E,15H
STAX D
HLT
```

4.MVI M, 8bit:- This is an instruction to load 8-bit data immediately into the memory location pointed by the HL pair register. This is 2-byte instruction. The 1st byte is the op-code and the 2nd byte is the 8 bit data.

5. Arithmetic operations related to 16-bits or register pair:-

INX R_p/DCX R_p : Increment/decrement register pair

These are 1-byte instructions. They treat the content of 2 registers as one 16bit number and increment/decrement it by 1. These instructions do not affect the flags.

Addressing mode is register addressing.

(INX B ; INX D; INX H; INX SP; DCX B; DCX D; DCX H; DCX SP)

e.g- Load the number D440H in register pair BC. Increment using INX B. Check if INX B is equivalent to INR B and INR C.

(Note: INR increments the register content while INX increments the address in the register pair)

a) LXI B,D440H

D4	40
B	C

INX B gives D4 in B & 41 in C.
∴ contents in BC pair are D441.

b) INR B increments D4 to D5.

INR C increments 40 to 41
∴ new contents are D541.

6. Arithmetic operations related to memory:-

ADD M : Adds (M) to (A) and stores the sum in A.
SUB M : Subtracts(M) from (A) and stores the difference in A.
INR M : Increments the contents of memory location by 1
DCR M : Decrements the contents of memory location by 1.

All the above instructions are 1 byte instructions and addressing mode is register indirect.
M stands for content of memory location.
ADD M /SUB M implicitly assume that one of the operands is A .All the flags are modified according to the data conditions.

With INR M/DCR M, all flags **except CY flag** are affected.

ADDITIONAL LOGIC OPERATIONS : ROTATE

(instructions relating rotating the accumulator bits)

RLC: Rotate accumulator left. **RAL: Rotate accumulator left through carry.**

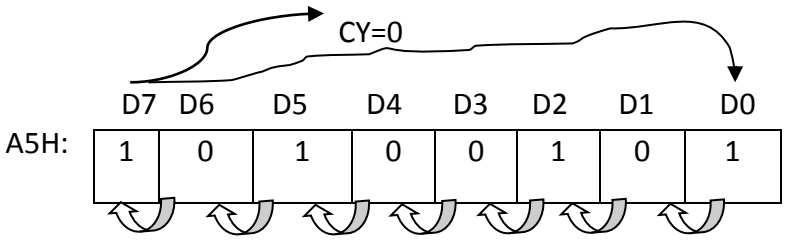
RRC: Rotate accumulator right. **RAR: Rotate accumulator right through carry.**

Description :-

RLC: Each bit of the accumulator is shifted to left by one position.

Bit D_7 is placed in the position of D_0 as well as in the carry flag. S, Z, P, AC flags are not affected.

e.g- Before the instruction, let accumulator = A5H and CY=0



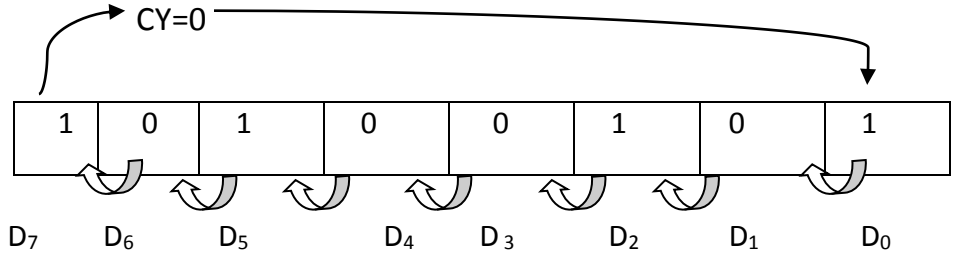
After the execution of RLC ,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D0
0	1	0	0	1	0	1	1

Accumulator will contain now 4BH.and CY=1

RAL: Each bit of the accumulator is rotated left by one position through the CY flag. D_7 is placed in CY position. CY bit is placed in D_0 position. S, Z, AC, P are not affected. CY is modified.(like 9 bit rotation)

Eg. Accumulator has A5H and CY=0



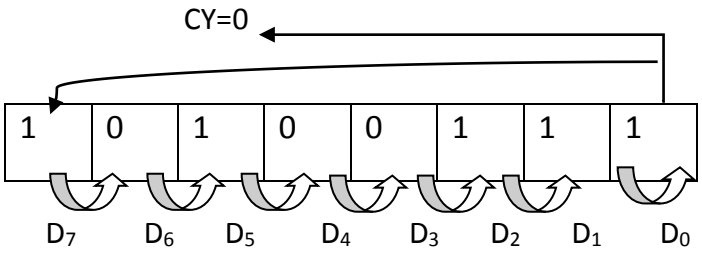
After RAL instruction is executed,

0	1	0	0	1	0	1	0

i.e Accumulator will contain 4AH and CY=1.

RRC:- Each bit of the accumulator is shifted right to adjacent position. Bit D_0 is placed in D_7 position as well as CY position. Only CY flag is modified. Does not affect S, Z, AC, P flags.

e.g- Let accumulator contain A7H and let CY=0



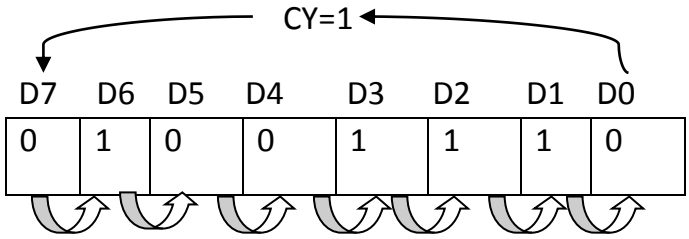
After RRC,

1	1	0	1	0	0	1	1

Accumulator will contain D3H and CY =1.

RAR:- Each bit of accumulator is rotated to the right through carry. D_0 is placed in CY and CY is placed in D_7. Only CY flag is modified.

Let accumulator=4EH and CY=1



After RAR,

1	0	1	0	0	1	1	1

Accumulator will contain A7H and CY =0.

Applications of rotate instructions: Rotate instructions are used in arithmetic multiply and divide operations and also in serial data transfer.

1. Rotating right is equivalent to dividing
2. Rotating left is equivalent to multiplying
3. To check whether the data is positive /negative.
4. To check whether the data is even or odd.

LOGIC OPERATIONS : COMPARE

8085 µP has 2 types of compare operations. They are CMP and CPI.

CMP R/M : compare register /memory with accumulator

CPI 8bit : compare immediately 8bit data with accumulator

CMP R/M is one byte instruction while **CPI 8bit** is 2- byte instruction. With compare instructions, the contents of register or memory location are not affected. Only the flags get modified. The process is as follows:-

The µP compares a data byte (from register/memory location/direct data) with the contents of accumulator by **subtracting** the data byte from (A) and indicates whether the data byte is > or < or = (A) by modifying the flags.

If (A) < (R/M) ,CY flag is set. (Read as: content of register R or content of memory location)
If (A) > (R/M) , CY flag is reset.
If (A) = (R/M), Zero flag is set.
If (A) ≠ (R/M), Zero flag is reset.
If memory is an operand, its address is specified by (HL).
Similar case with CPI 8-bit data .

SUB ROUTINE

A subroutine is a group of instructions that performs a sub task of repeated occurrence. eg. Time delays or arithmetic operations. The subroutine is written as a separate unit, apart from the main program, and the microprocessor transfers the program execution from the main program to the subroutine whenever it is called to perform the task. After the completion of the subroutine task, the microprocessor returns to the main program.. e.g: If a time delay is required between 3 successive events, 3 delays can be written in the main program. To avoid repetition of the same delay instructions, the subroutine technique is used. Delay instructions are written once, separately from the main program, and are called by the main program when needed.

Before implementing the subroutine technique, the stack must be defined. The stack is used to store he memory address of the instruction in the main program that follows the subroutine call.

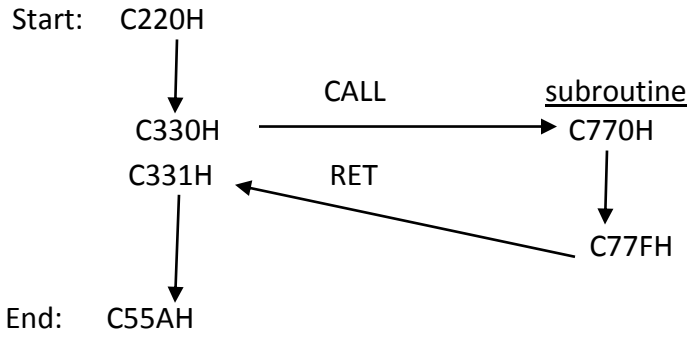
The 8085 µP has 2 instructions to implement subroutines: **CALL** (call a subroutine) and **RET**(return to main program from the subroutine). The CALL instruction is used in the main

program to call a subroutine, and the RET instruction is used at the end of the subroutine to return to the main program.

When a subroutine is called, the content of the program counter, which is the address of the instruction following the CALL instruction, is stored on the stack and the program execution is transferred to the subroutine address.

When the RET instruction is executed at the end of the subroutine, the memory address stored on the stack is retrieved and the sequence of execution is resumed in the main program.

CALL 16 bit address : is unconditional call instruction. It is a 3 byte instruction when the stack pointer is decremented by 2
RET : is a 1 – byte instruction(unconditional return)
Example: main program



A simple program is available on Page 42 for reference.

The following are conditional CALL and RET instructions ---

Conditional CALL instructions	Conditional RETURN instructions
CC: Call subroutine if carry flag is set (Cy =1) **CNC** : Call subroutine if carry flag is reset(Cy =0) **CZ** : Call subroutine if zero flag is set (Z = 1) **CNZ** : Call subroutine if zero flag is reset (Z = 0) **CM** : Call subroutine if sign flag is set (S =1,negative number) **CP**: Call subroutine if sign flag is reset (S = 0, positive number) **CPE** : Call subroutine if Parity flag is set (P=1) **CPO**: Call subroutine if Parity flag is reset(P=0)	**RC** : Return if Carry flag is set **RNC**: Return if Carry flag is not set(or reset) **RZ** : Return if Zero flag is set **RNZ** : Return if Zero flag is reset **RM** : Return if sign flag is set(S=1,negative number) **RP** : Return if sign flag is reset (S=0, positive number) **RPE** : Return if Parity flag is set(even parity) **RPO**: Return if Parity flag is reset(odd parity)

Meaning and function performed by some advanced instructions:

1. LHLD F000H: Load HL registers direct. This instruction transfers the contents of F000H to L register and that of F001H to H register.
2. SHLD F000H: Store HL registers direct. This instruction stores contents of L register into F000H and that of H register into F001H.
3. XCHG: Exchange the contents of HL and DE. H ↔ D and L ↔ E.
4. ADC R: Add with carry contents of register R and A(R stands for any register).This is used in addition of 16 bit data.
5. ACI 8 bit : Add accumulator content to 8 bit data and the carry.
6. SBB R: Subtract the content of R and the borrow from the content of A.(sum of borrow and subtrahend is subtracted from content of register A)
7. SBI 8 bit: Subtract 8 bit data and the borrow from the content of A.
8. DAD Rp: Double register add. Adds the contents of the register pair(BC / DE / HL / SP) to the contents of HL register and the result placed in HL registers.
9. XTHL: Exchange top of the stack with H and L. The contents of L are exchanged with the contents of memory location shown by the stack pointer and the contents of H are exchanged with the contents of memory location of the stack pointer + 1.
10. SPHL: Copy H and L registers into the Stack Pointer Register, contents of H specify the high order byte and that of L specify low order byte, HL contents are not affected.
11. PCHL: Copy H and L registers into the Program Counter.
12. CMC: Complement the carry flag.
13. STC: Set the carry flag.

Restart instructions(RST)

RST 0: Call 0000H	RST 1: Call 0008H	RST 2: Call 0010H	RST 3: Call 0018H
RST 4: Call 0020H	RST 5: Call 0028H	RST 6: Call 0030H	RST 7: Call 0038H

The 8255A Programmable Peripheral Interface (PPI)

Features 1: 1.The 8255A is a widely used, programmable, parallel I/O device.

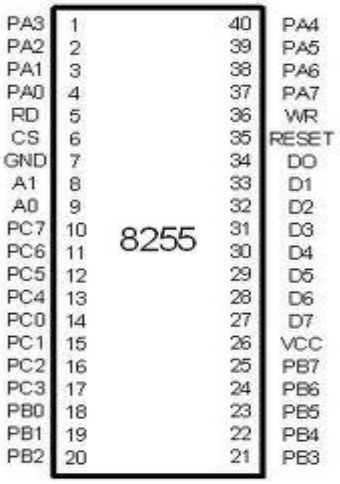
2. It can be programmed to transfer data under various conditions, simple I/O to interrupt I/O.

3. It is flexible, versatile, economical and complex.

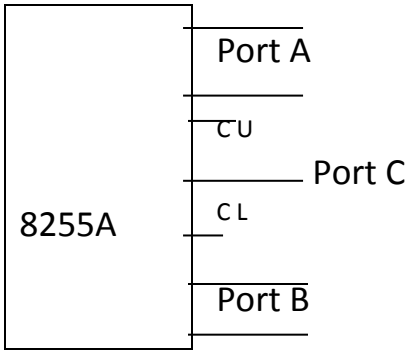
4. It is an important general purpose I/O device that can be used with almost any micro-processor.

5. It is TTL compatible and has three 8- bit parallel ports **Port A, Port B and Port C** .

The 8255A is a 40 pin IC. It has 24 I/O pins. The I/O pins are grouped in two 8-bit parallel **Port A and Port B** with the remaining 8 bits as **Port C**. The 8 bits of Port C can be used as individual bits or can be grouped in two 4-bit Ports -- **$C_{upper}(C_U)$ and $C_{lower}(C_L)$.** The functions of these Ports are defined by writing a **control word in a register called the control register**.



Pin lay out diagram of 8255



Pin names

Pin names

PA→ Port A; PB→ Port B; PC→ Port C;

A_1, A_0→ Port address;

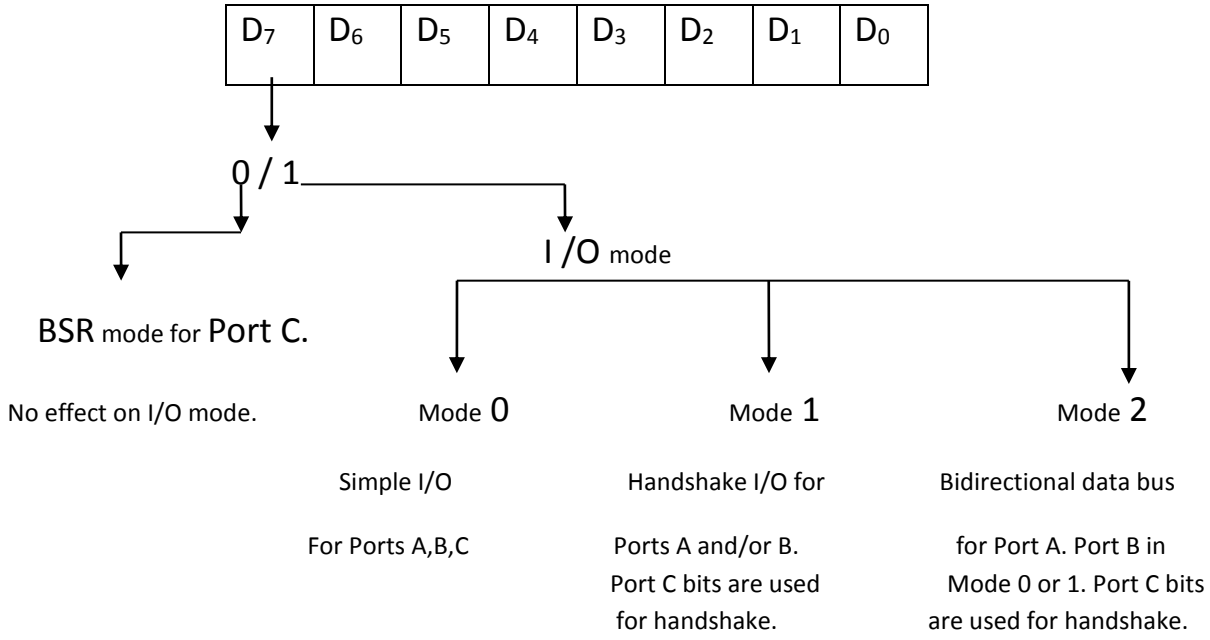
RD→ Read i/p; WR→ Write i/p ; CS→ chip select; RESET→ Reset i/p;

$D_0 – D_7$→ data bus(bidirectional);

Vcc and Ground { a total of 40 pins}

All the functions of 8255 are classified according to 2 modes. They are (i) BSR mode (i.e Bit Set Reset mode) and (ii) Simple I/O mode (i.e Input/Output).

Control word : It is 8 bit long. It is represented as $D_7D_6D_5D_4D_3D_2D_1D_0$.



Features 2:

1.**BSR (Bit Set/ Reset) mode**: This mode is used to set or reset the bits in Port C.

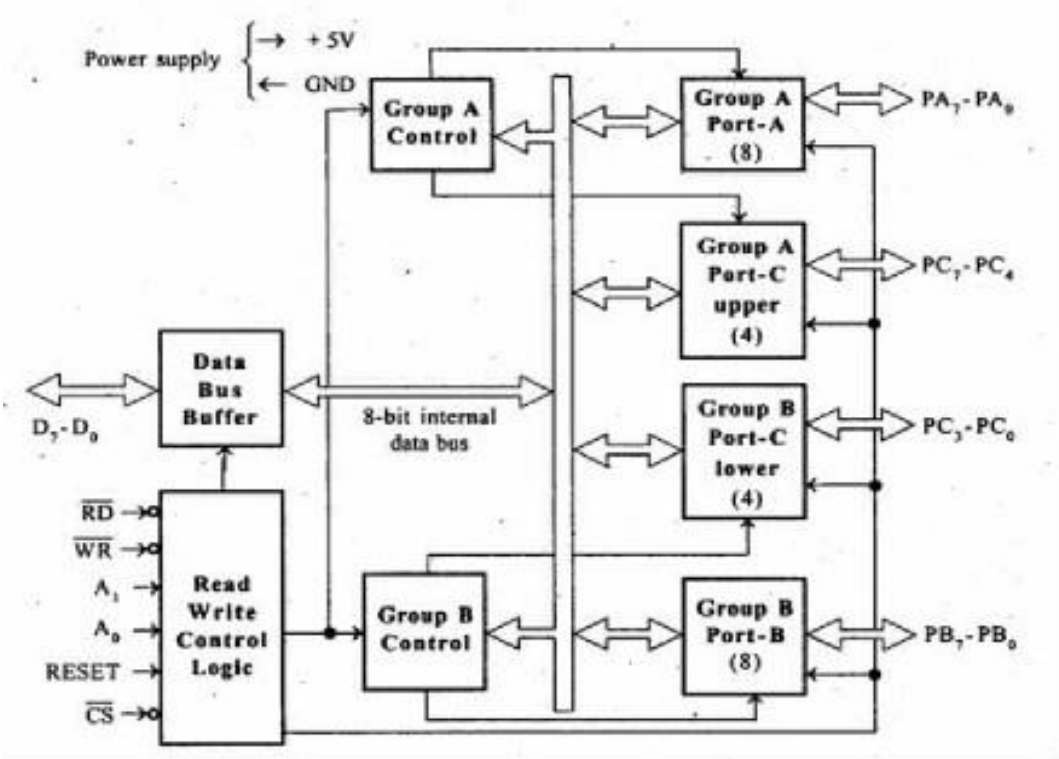
2**. I/O(Input/ Output) mode**: This is further divided into 3 modes –Mode 0, Mode 1 and Mode 2.

In Mode 0, all ports function as simple I/O ports. Output is latched. Inputs are not latched but buffered.

Mode 1 is a handshake mode. Here ports A and / or B use bits from port C as handshake signals.

In Mode 2, port A can be set up for bidirectional data transfer using handshake signals from port C and port B can be set up either in Mode 0 or Mode 1.

The block diagram of 8255



It shows all the elements of the programmable device- two 8-bit ports PA and PB; two 4- bit ports PC_U and PC_L; the data bus buffer; control logic and the power supply.

Control logic: This section has 6 lines. They are $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_1, A_0, and RESET. The function of each line is as follows:

$\overline{RD}$(Read):- This control signal enables the Read operation. When the signal at this pin is low, the MPU reads data from a selected I/O port of the 8255.

$\overline{WR}$(Write):- This control signal enables the Write operation. When the signal at this pin goes low, the MPU writes into a selected I/O port or the control register.

RESET(Reset):- This is an active high signal. It clears the control register and sets all the ports in the input mode.

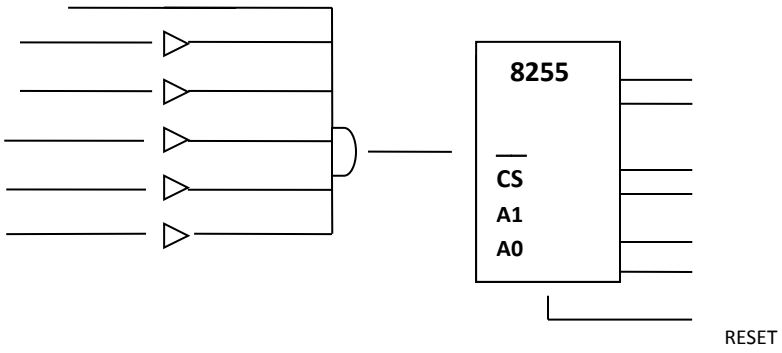
$\overline{CS}$, A_0, A_1 :- CS is chip select . All the 3 are device select signals. $\overline{CS}$ is connected to a decoded

address. A_0 and A_1 are generally connected to A_0 and A_1 of the MPU address lines respectively.

The $\overline{CS}$ signal is master Chip Select. Its function is shown in the following table

$\overline{CS}$	A_1	A_0	Selected
1	X	X	8255 is not selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control register

Example:- Write the port addresses for the given circuit.



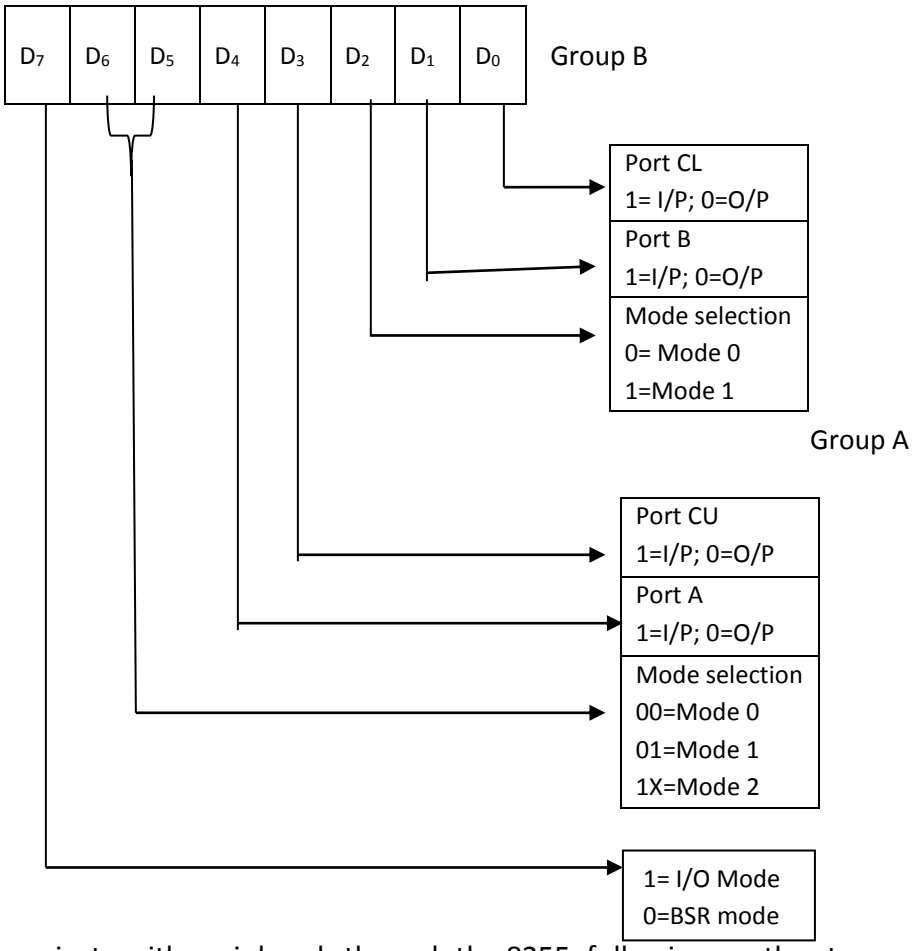
We know that the port addresses are determined by $\overline{CS}$, A_1 and A_0 lines. Also the chip select line is active low.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	address	Port
1	**0**	**0**	**0**	0	0	0	0	**8**0	A
1	**0**	**0**	**0**	0	0	0	1	**8**1	B
1	**0**	**0**	**0**	0	0	1	0	**8**2	C
1	**0**	**0**	**0**	0	0	1	1	**8**3	Control register

The $\overline{CS}$ line goes low when all the i/ps to NAND gate are 1. In other words, A_6, A_5, A_4, A_3, A_2, the i/ps to NOT gates are 0 and A_7 is 1. When these signals are combined with A_1 and A_0, the port addresses would be as follows.

CONTROL WORD:

The content of the control register is called the control word. It specifies an I/O function for each port. This register can be accessed to write a control word when A_0=1 and A_1=1. But this register is not accessible for Read operation. The bit D_7 of the control register specifies either the Bit Set / Reset function or the I/O function. The D_6 to D_0 bits determine I/O functions in various modes when D_7=1. If D_7=0, then port C operates in the BSR mode. The BSR control word does not affect the functions of port A and port B. The control word format with different modes is shown.



To communicate with peripherals through the 8255, following are the steps.

Step 1:- Determine the addresses of Ports A, B, C and the control register according to the chip select logic and the address lines A_0 and A_1.

Step 2:- Write a control word in the control register and

Step 3:- Write I/O instructions to communicate with peripherals through ports A, B, C.

MODE 0 : Simple Input or Output

In this mode, each port (or half port in case of C) can be programmed to function as simply an input port or an output port.

The Input / Output **features in Mode 0** are as follows:-

a) Inputs are not latched.
b) Outputs are latched.
c) Ports do not have handshake or interrupt capability.

Example 1: Consider Port A and Port C upper as o/p ports and Port B and Port C lower as i/p ports. Write the control word when CS is low in Mode 0.

From the control word format

D_7	D_6 D_5	D_4	D_3	D_2	D_1	D_0
I/O	Mode 0	o/p A	CU o/p	Mode select	Port B i/p	Port CL i/p
1	0 0	0	0	0	1	1

i.e 1000 0011 = 83H.

Example 2: Write the control word if Port A and Port B are o/p ports and Port C is i/p port in Mode 0. CS is low.

D_7	D_6 D_5	D_4	D_3	D_2	D_1	D_0
I/O	Mode 0	o/p A	CU i/p	Mode sel	Port B o/p	Port CL i/p
1	0 0	0	1	0	0	1

i.e 1000 1001 = 89H.

Example 3: Write the control word if Port A is o/p and Port B and C are i/ps. CS is low.

D_7	D_6 D_5	D_4	D_3	D_2	D_1	D_0
1	0 0	0	1	0	1	1

i.e 1000 1011 = 8BH.

Example 4: The control word of 8255 PPI in I/O mode is 92H. Identify the mode of operation and I/O status of all ports.

Ctrl word is 92 H = 1001 0010_2 i.e.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	Mode 0		Port A o/p	Port CU o/p	Mode sel	Port B o/p	Port CL i/p
1	0	0	1	0	0	1	0

D_7 =1 implies the mode of operation is simple I/O.

Port A : I/p port, mode 0

Port B : I/p port, mode 0

Port C_U : O/p port, mode 0

Port C_L : O/p port, mode 0

Problem 1: Write a program to get the o/p at Port A and Port B and i/p at Port C in Mode 0.

From example 2 ,we have the control word = 89H.

MVI A, 89H
OUT control reg.
IN Port C
OUT Port A
OUT Port B
HLT

Address	Op code	Mnemonic	Comment
C000	3E	MVI A,89H	; transfer CTRL word to accumulator
C001	89		
C002	D3	OUT Ctrl Reg.	;CTRL word from acc. to CTRL REG
C003	**33**		
C004	DB	IN Port C	; give C as i/p port
C005	**32**		
C006	D3	OUT Port A	; o/p at Port A
C007	**30**		
C008	D3	OUT Port B	; o/p at Port B
C009	**31**		
C00A	76	HLT	; Stop the program

Note: From the manual, if 8255 is connected externally, then the addresses of the ports are

Port A → 30; Port B → 31; Port C → 32; CTRL REG → 33.

If used internally, they are respectively 10,11,12 and 13.

Problem 2: The control word of 8255 PPI in I/O mode is 83H. Write a program to input 8 bit data from Port B. Mask the upper 4 bits of data (accumulator contents) and then display it at port A.

Ctrl word is 83 H = $1000\ 0011_2$ i.e.

D_7	D_6 D_5	D_4	D_3	D_2	D_1	D_0
I/O	Mode 0	Port A o/p	Port CU o/p	Mode sel	Port B o/p	Port CL i/p
1	0 0	0	0	0	1	1

D_7 =1 implies the mode of operation is simple I/O.

Port A : o/p port, mode 0

Port B : I/p port, mode 0

Port C_U : o/p port, mode 0

Port C_L : I/p port, mode 0

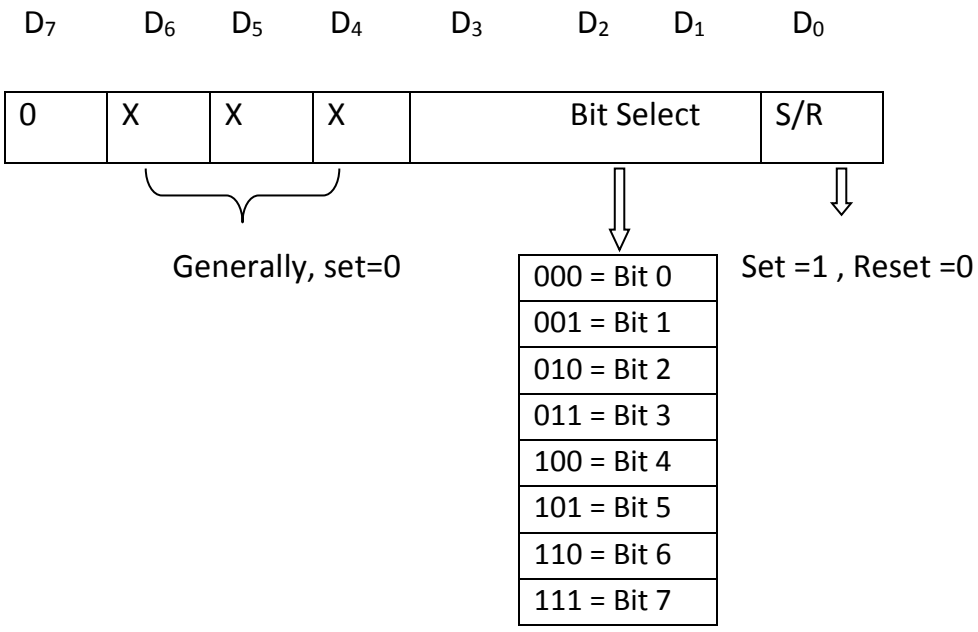
Program:

```
MVI A, 83H          1 0 0 0  0 0 1 1     logically AND
OUT CWR             0 0 0 0  1 1 1 1
IN Port B           0 0  0 0  0 0 1 1
ANI 0FH             ==========
OUT Port A
RST 1
```

BSR (BIT SET / RESET) MODE:

This mode is concerned only with 8 bits of Port C. The bits can be set or reset by writing an appropriate control word in the control register. A control word with D7 = 0 is recognized as a BSR control word and it does not alter any previously transmitted control word with bit D7 = 1. Thus the I/P – O/p operations of ports A and B are not affected. Also, individual bits of port C can be used for applications such as an ON/OFF switch.

BSR control word: The control word is written in the control register. One can set or reset one bit at a time using this control word. The format is as shown below-



Example to SET and RESET specific bits:

Write a BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10mS. Assume that a delay subroutine is available. Given: Address of control register is 83H.

To write BSR control words—

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
To set bit PC_7:	0	0	0	0	1	1	1	1	i.e. 0FH
To set bit PC_3:	0	0	0	0	0	1	1	1	i.e. 07H
To reset bit PC_7:	0	0	0	0	1	1	1	0	i.e. 0EH
To reset bit PC_3:	0	0	0	0	0	1	1	0	i.e. 06H

Subroutine:

```
MVI A, 0FH      ;load byte in accumulator to set PC7
OUT 83H         ;Set PC7 =1
MVI A, 07H      ;load byte in accumulator to set PC3
OUT 83H         ;Set PC3 =1
CALL DELAY      ;delay for 10mS
MVI A, 06H      ; load byte in accumulator to reset PC3
OUT 83H         ;Reset PC3
MVI A, 0EH      ; load byte in accumulator to reset PC7
OUT 83H         ;Reset PC7
RET
```

From the above, following points can be noted-

1.To set/reset bits in Port C, control word is written in control register and not port C.
2. A BSR control word affects only one bit in Port C.
3. The BSR control word does not affect the I/O mode.

****

STACK and STACK POINTER(SP) ----- PUSH & POP instructions

1. Stack is a part of memory defined in the program by the user. The stack is achieved by initializing Stack Pointer (SP). SP points to stack area only.

2. Stack area is normally defined in RAM only. It is used to store information (data) temporarily.

3. To initialize SP, we use the instruction LXI SP, 16 bit address.

4. While storing data(PUSHing) onto the stack, SP decrements but while retrieving data(POPing) from the stack, SP increments.

5. The current location where SP points is referred to as "Top of the Stack".

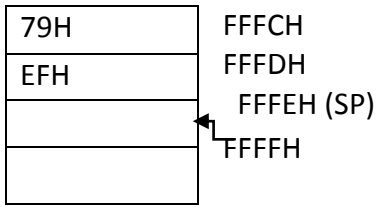
6. Stack follows the rule" FIRST IN - LAST OUT".

Simple Program to understand SP, PUSH and POP

MVI A, 77H	;load 77H in accumulator
MVI D,78H	;Load 78H in Reg. D
MVI C,79H	;Load 79H in Reg. C
ADD D	;(A) <- (A) + (D)
MOV B,A	;(B) <- (A)
LXI SP, FFFEH	; initialize stack to address FFFEH
PUSH B	;load the content of B in FFFDH and that of C in FFFCH
MVI D,55H	;load D with 55H
MVI A,65H	;load A with 65H
ADD D	;(A) <- (A) + (D)
POP B	;retrieve the contents and load B and C again
ADD B	;(A) <- (A) + (B)
STA D000H	;store the acc. Content to memory location D000H
RST 1	;restart the program

Working: 77H +
78H
A= E FH ; B = EFH

PUSH B



The SP gets decremented to store the data in high register, decremented again to store the data in low register. During PUSH operation, Acc. content will be lost.

POP B

During POP, the data in lower mem. location goes to lower register while the data In higher mem. location goes to higher register and every time the SP gets incremented.

A	F
B EFH	C 79H
D	E

65H +
55H
BAH in reg.A. Now, BAH is added to EFH to give the sum = 1A9H . So D000H contains A9H with CY=1.

Simple program to understand subroutine.

MAIN PROGRAM: (Start at C000H)

```
MVI B,55H
MVI E,ABH
MOV A,E
ADD B
MOV B,A        ; B= A=00H
CALL D000H
STA D100H      ;D100=25H
ADD B
 STA D101H     ;D101=25H
RST 1
```

SUBROUTINE:

```
D000H:   MVI C, 12H
D002H    MVI D, 13H
D004H    MOV A,D
D005H    ADDC        ; A=25H
D006H    RET
```

Result in D100: 25H and in D101=25H

==========

2 mark questions

1. What is Microprocessor? Give the power supply & clock frequency of 8085?

Ans:A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to those instructions and provides result as output. The power supply of 8085 is +5V and clock frequency in 3MHz.

2. List few applications of microprocessor-based system.

Ans: It is used:

i. For measurements, display and control of current, voltage, temperature, pressure, etc.
ii. For traffic control and industrial tool control.
iii. For speed control of machines.

3. What are the functions of an accumulator?

Ans: The accumulator is the register associated with the ALU operations and sometimes I/O operations. It is an integral part of ALU. It holds one of data to be processed by ALU. It also temporarily stores the result of the operation performed by the ALU.

4. List the 16 – bit registers of 8085 microprocessor.

Ans: Stack pointer (SP) and Program counter (PC).

5. List the allowed register pairs of 8085.

Ans:

- B-C register pair
- D-E register pair
- H-L register pair

11. List out the five categories of the 8085 instructions. Give examples of the instructions for each group.

Ans:

- Data transfer group – MOV, MVI, LXI.
- Arithmetic group – ADD, SUB, INR.
- Logical group –ANA, XRA, CMP.
- Branch group – JMP, JNZ, CALL.
- Stack I/O and Machine control group – PUSH, POP, IN, HLT.

12. Explain the difference between a JMP instruction and CALL instruction.

Ans: A JMP instruction permanently changes the program counter. A CALL instruction leaves

information on the stack so that the original program execution sequence can be resumed.

13. Explain the purpose of the I/O instructions IN and OUT.

Ans: The IN instruction is used to move data from an I/O port into the accumulator. The OUT instruction is used to move data from the accumulator to an I/O port. The IN & OUT instructions are used only on microprocessor, which use a separate address space for interfacing.

14. What is the difference between the shift and rotate instructions?

Ans: A rotate instruction is a closed loop instruction. That is, the data moved out at one end is put back in at the other end. The shift instruction loses the data that is moved out of the last bit locations.

16. What are the Control signals used for DMA operation?

Ans:-HOLD & HLDA.

17. What is meant by Wait State?

Ans:-This state is used by slow peripheral devices. The peripheral devices can transfer the data to or from the microprocessor by using READY input line. The microprocessor remains in wait state as long as READY line is low. During the wait state, the contents of the address, address/data and control buses are held constant.

18. List the four instructions which control the interrupt structure of the 8085 microprocessor.

Ans:-

- DI (Disable Interrupts)
- EI (Enable Interrupts)
- RIM (Read Interrupt Masks)
- SIM (Set Interrupt Masks)

19. What is meant by polling?

Ans:-Polling or device polling is a process which identifies the device that has interrupted the microprocessor.

20. What is meant by interrupt?

Ans:-Interrupt is an external signal that causes a microprocessor to jump to a specific subroutine.

21. Explain priority interrupts of 8085.

Ans:-The 8085 microprocessor has five interrupt inputs. They are TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. These interrupts have a fixed priority of interrupt service. If two or more interrupts go high at the

same time, the 8085 will service them on priority basis. The TRAP has the highest priority followed by RST 7.5, RST 6.5, RST 5.5. The priority of interrupts in 8085 is shown in the table.

TRAP	**1**
RST 7.5	**2**
RST 6.5	**3**
RST 5.5	**4**
INTR	**5**

22. What is a microcomputer?

Ans:-A computer that is designed using a microprocessor as its CPU is called microcomputer.

23. What is the signal classification of 8085

Ans:-All the signals of 8085 can be classified into 6 groups

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

24. What are operations performed on data in 8085

Ans:- The various operations performed are

- Store 8-bit data
- Perform arithmetic and logical operations
- Test for conditions
- Sequence the execution of instructions
- Store data temporarily during execution in the defined R/W memory locations called the stack

25. Steps involved to fetch a byte in 8085

Ans:-
i. The PC places the 16-bit memory address on the address bus
ii. The control unit sends the control signal RD to enable the memory chip
iii. The byte from the memory location is placed on the data bus
iv. The byte is placed in the instruction decoder of the microprocessor and the task is carried out according to the instruction

26. How many interrupts does 8085 have, mention them

Ans:-The 8085 has 5 interrupt signals; they are INTR, RST7.5, RST6.5, RST5.5 and TRAP

27. Basic concepts in memory interfacing

Ans:-The primary function of memory interfacing is that the microprocessor should be able to read from and write into a given register of a memory chip. To perform these operations the microprocessor should

- Be able to select the chip
- Identify the register
- Enable the appropriate buffer

28. Define instruction cycle, machine cycle and T-state

Ans:-Instruction cycle is defined, as the time required completing the execution of an instruction. Machine cycle is defined as the time required completing one operation of accessing memory, I/O or acknowledging an external request. Tcycle is defined as one subdivision of the operation performed in one clock period

29. What is an instruction?

Ans:-An instruction is a binary pattern entered through an input device to command the microprocessor to perform that specific function

30. What is the use of ALE

Ans:-The ALE is used to latch the lower order address so that it can be available in T2 and T3 and used for identifying the memory address. During T1 the ALE goes high, the latch is transparent ie, the output changes according to the input data, so the output of the latch is the lower order address. When ALE goes low the lower order address is latched until the next ALE.

31. How many machine cycles does 8085 have, mention them

Ans: The 8085 have seven machine cycles. They are

- Opcode fetch
- Memory read
- Memory write
- I/O read
- I/O write
- Interrupt acknowledge
- Bus idle

32. Explain the signals HOLD, READY and SID

Ans:HOLD indicates that a peripheral such as DMA controller is requesting the use of address bus, data bus and control bus. READY is used to delay the microprocessor read or write cycles until a slow responding peripheral is ready to send or accept data.SID is used to accept serial data bit by bit

33. Mention the categories of instruction and give two examples for each category.

Ans: The instructions of 8085 can be categorized into the following five categories

- Data transfer Instructions -MOV Rd,Rs STA 16-bit
- Arithmetic Instructions -ADD R DCR M
- Logical Instructions -XRI 8-bit RAR
- Branching Instructions -JNZ CALL 16-bit
- Machine control Instructions -HLT NOP

34. Explain LDA, STA and DAA instructions

Ans:LDA copies the data byte into accumulator from the memory location specified by the 16-bit address. STA copies the data byte from the accumulator in the memory location specified by 16-bit address. DAA changes the contents of the accumulator from binary to 4-bit BCD digits.

35. Explain the different instruction formats with examples

Ans: The instruction set is grouped into the following formats

- One byte instruction -MOV C,A
- Two byte instruction -MVI A,39H
- Three byte instruction -JMP 2345H

36. What is the use of addressing modes, mention the different types

Ans: The various formats of specifying the operands are called addressing modes, it is used to access the operands or data. The different types are as follows

- Immediate addressing
- Register addressing
- Direct addressing
- Indirect addressing
- Implicit addressing

37. What is the use of bi-directional buffers?

Ans: It is used to increase the driving capacity of the data bus. The data bus of a microcomputer system is bi-directional, so it requires a buffer that allows the data to flow in both directions.

38. Give the register organization of 8085

Ans:

W(8)	Temp. Reg
Z(8)	Temp. Reg

B(8)	**Register**
C(8)	**Register**
D(8)	**Register**
E(8)	**Register**
H(8)	**Register**
L(8)	**Register**
Stack Pointer	**(16)**
Program Counter	**(16)**

39. Define stack and explain stack related instructions

Ans: The stack is a group of memory locations in the R/W memory that is used for the temporary storage of binary information during the execution of the program. The stack related instructions are PUSH & POP

40. Why do we use XRA A instruction

Ans: The XRA A instruction is used to clear the contents of the Accumulator and store the value 00H.

41. Compare CALL and PUSH instructions

Ans:

CALL	PUSH
1.When CALL is executed the microprocessor automatically stores the 16-bit address of the instruction next to CALL on the stack.	**1.PUSH The programmer uses the instruction to save the contents of the register pair on the stack**
2.When CALL is executed the stack pointer is decremented by two	**2. When PUSH is executed the stack pointer is decremented by two**

42. What is Microcontroller and Microcomputer

Ans: Microcontroller is a device that includes microprocessor; memory and I/O signal lines on a single chip, fabricated using VLSI technology. Microcomputer is a computer that is designed using microprocessor as its CPU. It includes microprocessor, memory and I/O.

43. Define Flags

Ans: The flags are used to reflect the data conditions in the accumulator. The 8085 flags are S-Sign flag, Z-Zero flag, AC-Auxiliary carry flag, P-Parity flag, CYCarry flag, D7 D6 D5 D4 D3 D2 D1 D0

44. How does the microprocessor differentiate between data and instruction?

Ans: When the first m/c code of an instruction is fetched and decoded in the instruction register, the microprocessor recognizes the number of bytes required to fetch the entire instruction. For example MVI A, Data, the second byte is always considered as data. If the data byte is omitted by mistake whatever is in that memory location will be considered as data & the byte after the "data" will be treated as the next instruction.

45. Compare RET and POP

Ans:

RET	POP
1.RET transfers the contents of the top two locations of the stack to the PC	1.POP transfers the contents of the top two locations of the stack to the specified register pair
2.When RET is executed the SP is incremented by two	2. When POP is executed the SP is incremented by two
3.Has 8 conditional RETURN instructions	3.No conditional POP instructions

71. What is interfacing?

Ans: An interface is a shared boundary between the devices which involves sharing information. Interfacing is the process of making two different systems communicate with each other.

72. List the operation modes of 8255

Ans: a) I.O Mode

i. Mode 0-Simple Input/Output.
ii. Mode 1-Strobed Input/Output (Handshake mode)
iii. Mode 2-Strobed bidirectional mode

b) Bit Set/Reset Mode.

73. What is a control word?

Ans: It is a word stored in a register (control register) used to control the operation of a program digital device.

74. What is the purpose of control word written to control register in 8255?

Ans: The control words written to control register specify an I/O function for each I.O port. The bit D7 of the control word determines either the I/O function of the BSR function.

75.What is the size of ports in 8255?

Ans:

- Port-A : 8-bits
- Port-B : 8-bits
- Port-CU : 4-bits
- Port-CL : 4-bits
- **76. Distinguish between the memories mapped I/O peripheral I/O?**
- **Ans:**

Memory Mapped I/O	Peripheral MappedI/O
16-bit device address	8-bit device address
Data transfer between any general-purpose register and I/O port.	Data is transfer only between accumulator and I.O port
The memory map (64K) is shared between I/O device and system memory.	The I/O map is independent of the memory map; 256 input device and 256 output device can be connected
More hardware is required to decode 16-bit address	Less hardware is required to decode 8-bit address
Arithmetic or logic operation can be directly performed with I/O data	Arithmetic or logical operation cannot be directly performed with I/O data

- **77. What is memory mapping?**
- **Ans:** The assignment of memory addresses to various registers in a memory chip is called as memory mapping.
- **78. What is I/O mapping?**
- **Ans:**The assignment of addresses to various I/O devices in the memory chip is called as I/O mapping.
- **79. What is an USART?**
- **Ans:**USART stands for universal synchronous/Asynchronous Receiver/Transmitter. It is a programmable communication interface that can communicate by using either synchronous or asynchronous serial data.

==========

1.What are the various registers in 8085?

Ans: - Accumulator register, Temporary register, Instruction register, Stack Pointer, Program Counter are the various registers in 8085 .

2.In 8085 name the 16 bit registers?

Ans:- Stack pointer and Program counter all have 16 bits.

3.What are the various flags used in 8085?

Ans:- Sign flag, Zero flag, Auxillary flag, Parity flag, Carry flag.

4.What is Stack Pointer?

Ans:- Stack pointer is a special purpose 16-bit register in the Microprocessor, which holds the address of the top of the stack.

5.What is Program counter?

Ans:- Program counter holds the address of either the first byte of the next instruction to be fetched for execution or the address of the next byte of a multi byte instruction, which has not been completely fetched. In both the cases it gets incremented automatically one by one as the instruction bytes get fetched. Also Program register keeps the address of the next instruction.

6.Which Stack is used in 8085?

Ans:- LIFO (Last In First Out) stack is used in 8085.In this type of Stack the last stored information can be retrieved first.

7.What happens when HLT instruction is executed in processor?

Ans:- The Micro Processor enters into Halt-State and the buses are tri-stated.

8.What is meant by a bus?

Ans:- A bus is a group of conducting lines that carriers data, address, & control signals.

9.What is Tri-state logic?

Ans:- Three Logic Levels are used and they are High, Low, High impedance state. The high and low are normal logic levels & high impedance state is electrical open circuit conditions. Tri-state logic has a third line called enable line.

10.Give an example of one address microprocessor?

Ans:- 8085 is a one address microprocessor.

11.In what way interrupts are classified in 8085?

Ans:- In 8085 the interrupts are classified as Hardware and Software interrupts.

12.What are Hardware interrupts?

Ans:- TRAP, RST7.5, RST6.5, RST5.5, INTR.

13.What are Software interrupts?

Ans:- RST0, RST1, RST2, RST3, RST4, RST5, RST6, RST7.

14.Which interrupt has the highest priority?

Ans:- TRAP has the highest priority.

15.Name 5 different addressing modes?

Ans:- Immediate, Direct, Register, Register indirect, Implied addressing modes.

16.How many interrupts are there in 8085?

Ans:- There are 12 interrupts in 8085.

17.What is clock frequency for 8085?

Ans:- 3 MHz is the maximum clock frequency for 8085.

18.What is the RST for the TRAP?

Ans:- RST 4.5 is called as TRAP.

19.In 8085 which is called as High order / Low order Register?

Ans:- Flag is called as Low order register & Accumulator is called as High order Register.

20.What are input & output devices?

Ans:- Keyboards, Floppy disk are the examples of input devices. Printer, LED / LCD display, CRT Monitor are the examples of output devices.

==========