1. Introduction to Microprocessor

Definition:

- *"The microprocessor is a multipurpose, clock driven, register based, digital-integrated circuit which accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output."*
- "Microprocessor is a computer Central Processing Unit (CPU) on a single chip that contains millions of transistors connected by wires."

Introduction:

- A microprocessor is designed to perform arithmetic and logic operations that make use of small number-holding areas called registers.
- Typical microprocessor operations include adding, subtracting, comparing two numbers, and fetching numbers from one area to another.

2. Components of Microprocessor

- Microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution.
- The microprocessor can be divided into three segments as shown in the figure, Arithmetic/logic unit (ALU), register array, and control unit.
- These three segment is responsible for all processing done in a computer

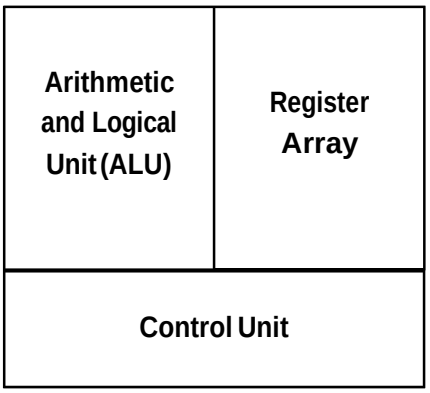


Figure: Components of Microprocessor

Arithmetic and logic unit (ALU)

- It is the unit of microprocessor where various computing functions are performed on the data.
- It performs arithmetic operations such as addition, subtraction, and logical operations such as OR,AND, and Exclusive-OR.
- It is also known as the brain of the computer system.

Register array

- It is the part of the register in microprocessor which consists of various registers identified by letters such as B, C, D, E, H, and L.
- Registers are the small additional memory location which are used to store and transfer data and programs that are currently being executed.

Control unit

- The control unit provides the necessary timing and control signals to all the operations in the microcomputer.
- It controls and executes the flow of data between the microprocessor, memory and peripherals.
- The control bus is bidirectional and assists the CPU in synchronizing control signals to internal devices and external components.
- This signal permits the CPU to receive or transmit data from main memory.

3. System bus (data, address and control bus).

- This network of wires or electronic pathways is called the 'Bus'.
- A system bus is a single computer bus that connects the major components of a computer system.
- It combines the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation.
- The technique was developed to reduce costs and improve modularity.

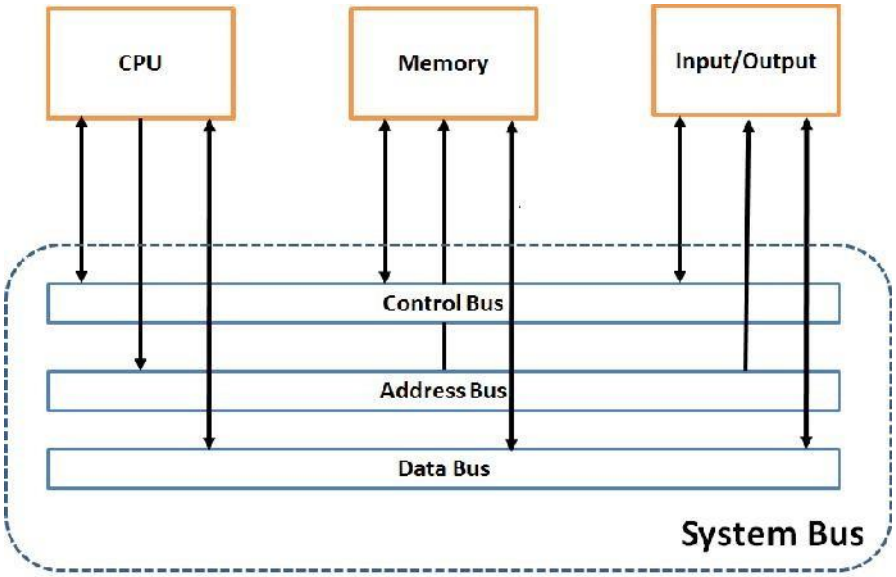


Figure: System Bus

Address Bus

- It is a group of wires or lines that are used to transfer the addresses of Memory or I/O devices.
- It is unidirectional.
- The width of the address bus corresponds to the maximum addressing capacity of the bus, or the largest address within memory that the bus can work with.
- The addresses are transferred in binary format, with each line of the address bus carrying a single binary digit.
- Therefore the maximum address capacity is equal to two to the power of the number of lines present (2^lines).

Data Bus

- It is used to transfer data within Microprocessor and Memory/Input or Output devices.
- It is bidirectional as Microprocessor requires to send or receive data.
- Each wire is used for the transfer of signals corresponding to a single bit of binary data.
- As such, a greater width allows greater amounts of data to be transferred at the same time.

Control Bus

- Microprocessor uses control bus to process data, i.e. what to do with the selected memory location.
- Some control signals are Read, Write and Opcode fetch etc.
- Various operations are performed by microprocessor with the help of control bus.
- This is a dedicated bus, because all timing signals are generated according to control signal.

4. Microprocessor systems with bus organization

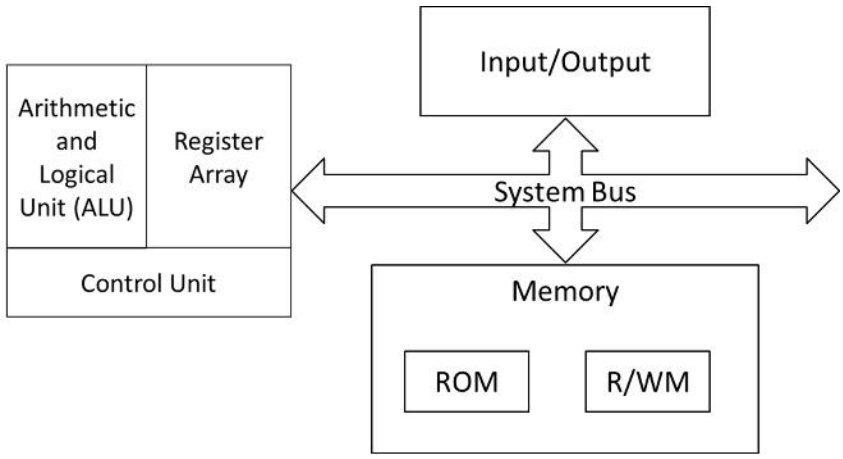


Figure: Microprocessor systems with bus organization

- To design any meaningful application microprocessor requires support of other auxiliary devices.
- In most simplified form a microprocessor based system consist of a microprocessor, I/O (input/output) devices and memory.
- These components are interfaced (connected) with microprocessor over a common communication path called system bus. Typical structure of a microprocessor based system is shown in Figure.
- Here, microprocessor is master of the system and responsible for executing the program and coordinating with connected peripherals as required.
- Memory is responsible for storing program as well as data. System generally consists of two types of memories ROM (Read only and non-volatile) and RAM (Read/Write and volatile).
- I/O devices are used to communicate with the environment. Keyboard can be example of input devices and LED, LCD or monitor can be example of output device.
- Depending on the application level of sophistication varies in a microprocessor based systems. For example: washing machine, computer.

1. Explain Classification of Memory

Ans.

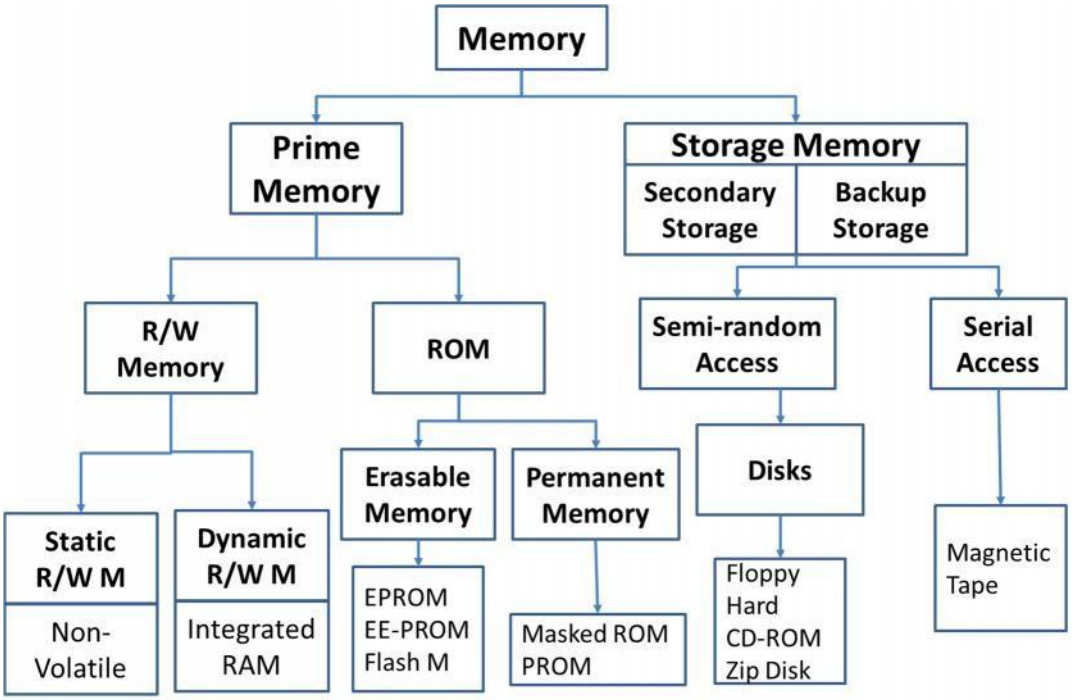


Figure: Classification of Memory

ROM (Read Only Memory):

The first classification of memory is ROM. The data in this memory can only be read, no writing is allowed. It is used to store permanent programs. It is a nonvolatile type of memory.

The classification of ROM memory is as follows:

1. **Masked ROM**: the program or data are permanently installed at the time of manufacturing as per requirement. The data cannot be altered. The process of permanent recording is expensive but economic for large quantities.

2. **PROM (Programmable Read Only Memory):** The basic function is same as that of masked ROM. but in PROM, we have fuse links. Depending upon the bit pattern, the fuse can be burnt or kept intact. This job is performed by PROM programmer.
   To do this, it uses high current pulse between two lines. Because of high current, the fuse will get burnt; effectively making two lines open. Once a PROM is programmed we cannot change connections, only a facility provided over masked ROM is, the user can load his program in it. The disadvantage is a chance of re-growing of the fuse and changes the programmed data because of aging.

3. **EPROM (Erasable Programmable Read Only Memory):** the EPROM is programmable by the user. It uses MOS circuitry to store data. They store 1's and 0's in form of charge. The information stored can be erased by exposing the memory to ultraviolet light which erases the data stored in all memory locations. For ultraviolet light, a quartz window is provided which is covered during normal operation. Upon erasing it can be reprogrammed by using EPROM programmer. This type of memory is used in a project developed and for experiment use. The advantage is it can be programmed erased and reprogrammed. The disadvantage is all the data get erased even if you want to change single data bit.

4. **EEPROM:** EEPROM stands for electrically erasable programmable read only memory. This is similar to EPROM except that the erasing is done by electrical signals instead of ultraviolet light. The main advantage is the memory location can be selectively erased and reprogrammed. But the manufacturing process is complex and expensive so do not commonly used.

R/W Memory (Read/Write Memory):

The RAM is also called as read/write memory. The RAM is a volatile type of memory. It allows the programmer to read or write data. If the user wants to check the execution of any program, user feeds the program in RAM memory and executes it. The result of execution is then checked by either reading memory location contents or by register contents.

Following is the classification of RAM memory.
It is available in two types:

1. **SRAM (Static RAM):** SRAM consists of the flip-flop; using either transistor or MOS. for each bit we require one flip-flop. Bit status will remain as it is; unless and until you perform next write operation or power supply is switched off.

   Advantages of SRAM:
   - Fast memory (less access time)
   - Refreshing circuit is not required.

   Disadvantages of SRAM:
   - Low package density
   - Costly

2. **DRAM (Dynamic RAM):** In this type of memory a data is stored in form of charge in capacitors. When data is 1, the capacitor will be charged and if data is 0, the capacitor will not be charged. Because of capacitor leakage currents, the data will not be held by these cells. So the DRAMs require refreshing of memory cells. It is a process in which same data is read and written after a fixed interval.

   Advantages of DRAM:

   - High package density
   - Low cost

   Disadvantages of DRAM:

   - Required refreshing circuit to maintain or refresh charge on the capacitor, every after few milliseconds.

Secondary Memory

- **Magnetic Disk**: The Magnetic Disk is Flat, circular platter with metallic coating that is rotated beneath read/write heads. It is a Random access device; read/write head can be moved to any location on the platter

- **Floppy Disk**: These are small removable disks that are plastic coated with magnetic recording material. Floppy disks are typically 3.5″ in size (diameter) and can hold 1.44 MB of data. This portable storage device is a rewritable media and can be reused a number of times. Floppy disks are commonly used to move files between different computers. The main disadvantage of floppy disks is that they can be damaged easily and, therefore, are not very reliable. The following figure shows an example of the floppy disk. Figure 3 shows a picture of the floppy disk.

- **Hard Disk**: Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters. The hard disk platters spin at 5 a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).

- **Optical Disks:** Optical Mass Storage Devices Store bit values as variations in light reflection. They have higher area density & longer data life than magnetic storage. They are also standardized and relatively inexpensive. Their Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed.

Types of Optical Disk

1. CD-ROM (read only)
2. CD-R: (record) to a CD
3. CD-RW: can write and erase CD to reuse it (re-writable)
4. DVD(Digital Video Disk)

2. Explain I/O devices and their Interfacing

Ans. Input / Output (I/O)

- MPU communicates with outside word through I/O device.
- There are 2 different methods by which MPU identifies and communicates With I/O devices these methods are:
  1- Direct I/O (Peripheral)
  2- Memory-Mapped I/O

The methods differ in terms of the

- No. of address lines used in identifying an I/O device.
- Type of control lines used to enable the device.
- Instructions used for data transfer.

Direct I/O (Peripheral):-

- This method uses two instructions (IN & OUT) for data transfer.
- MPU uses 8 address lines to send the address of I/O device (can identify 256 input devices & 256 output devices).
- The (I/P & O/P devices) can be differentiated by control signals I/O Read (IOR) and I/O Write (IOW).
- The steps in communicating with an I/O device are similar to those in communicating with memory and can be summarized as follows:
  1- The MPU places an 8-bit device address on address bus then decoded.
  2- The MPU sends a control signal (IOR or IOW) to enable the I/O device.
  3- Data are placed on the data bus for transfer.

Memory-Mapped I/O:-

- The MPU uses 16 address lines to identify an I/O device.
- This is similar to communicating with a memory location.
- Use the same control signals (MEMR or MEMW) and instructions as those of memory.
- The MPU views these I/O devices as if they were memory locations.
- There are no special I/O instructions.
- It can identify 64k address shared between memory & I/O devices.

1. Write down main features of 8085 microprocessor.

- It is an 8 bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A0-A15
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD0 – AD7
- Data bus is a group of 8 lines D0 – D7
- It supports external interrupt request. .
- A 16 bit program counters (PC)
- A 16 bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).

2. Explain 8085 microprocessor architecture.

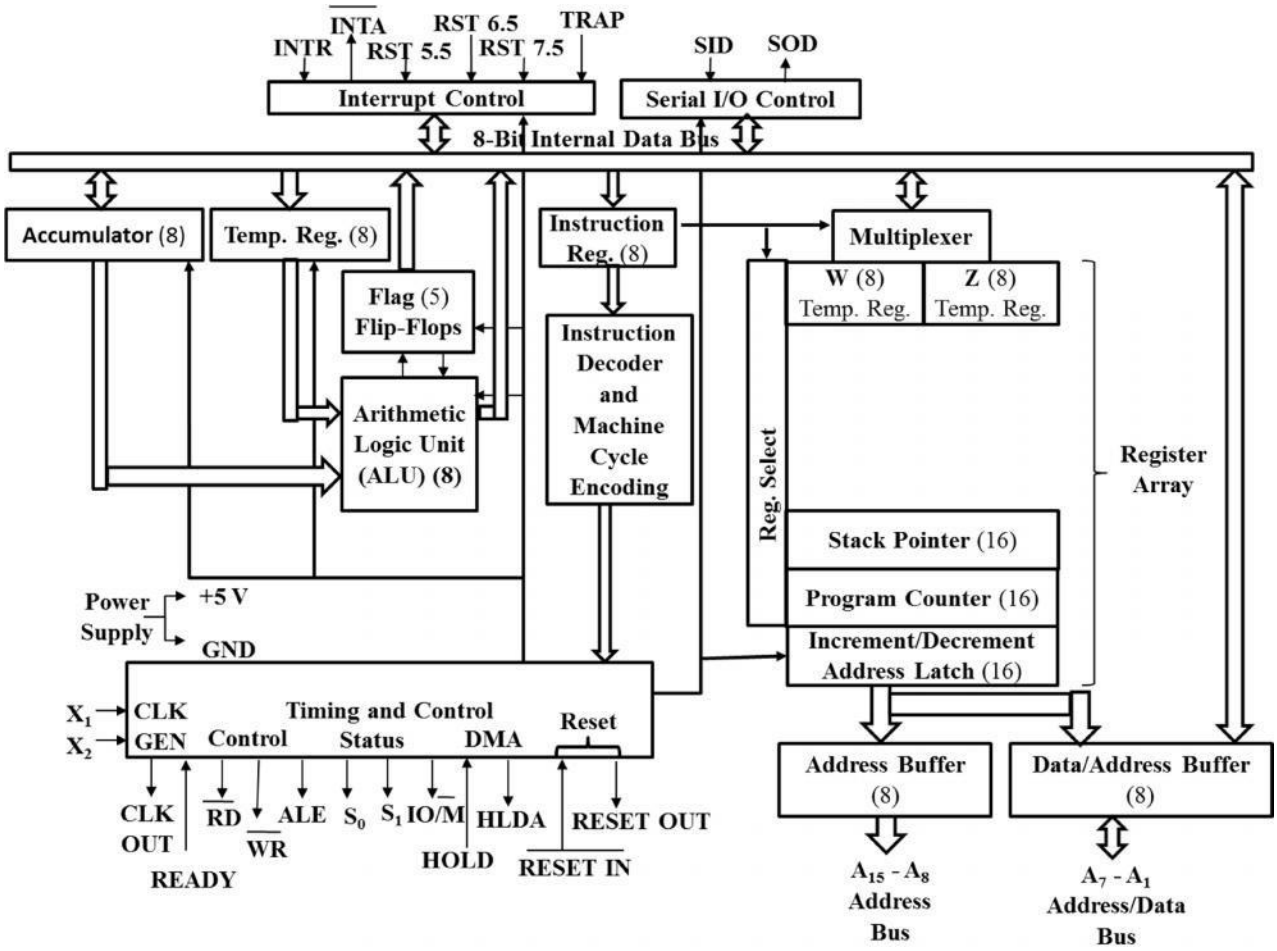


Figure: 8085 microprocessor architecture.

- The architecture of microprocessor 8085 can be divided into seven parts as follows:

Register Unit:

General Purpose Data Register

- 8085 has six general purpose data registers to store 8-bit data.
- These registers are named as B, C, D, E, H and L as shown in fig. 1.
- The user can use these registers to store or copy a data temporarily during the execution of a program by using data transfer instructions.
- These registers are of 8 bits but whenever the microprocessor has to handle 16-bit data, these registers can be combined as register pairs – BC, DE and HL.
- There are two internal registers – W and X. These registers are only for internal operation like execution of CALL and XCHG instructions and not available to the user.

Program Counter (PC)

- 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer.
- Memory locations have 16-bit addresses which are why this is a 16-bit register.
- The microprocessor uses this register to sequence the execution of the instructions.
- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

- SP is also a 16-bit register used as a memory pointer.
- It points to a memory location in R/W memory, called the stack.
- The beginning of the stack is defined by loading 16-bit address in the stack pointer.

MUX/DEMUX unit

- This unit is used to select a register out of all the available registers.
- This unit behaves as a MUX when data is going from the register to the internal data bus.
- It behaves as a DEMUX when data is coming to a register from the internal data bus of the microprocessor.
- The register select will behave as the function selection lines of the MUX/DEMUX.

Address Buffer Register & Data/Address Buffer Register

- These registers hold the address/data, received from PC/internal data bus and then load the external address and data buses.
- These registers actually behave as the buffer stage between the microprocessor and external system buses.

Control Unit:

- The control unit generates signals within microprocessor to carry out the instruction, which has been decoded.
- In reality it causes connections between blocks of the microprocessor to be opened or closed, so that the data goes where it is required and the ALU operations occur.
- The control unit itself consists of three parts; the instruction registers (IR), instruction decoder and machine cycle encoder and timing and control unit.

Instruction Register

- This register holds the machine code of the instruction.
- When microprocessor executes a program it reads the opcode from the memory, this opcode is stored in the instruction register.

Instruction Decoder & Machine Cycle Encoder

- The IR sends the machine code to this unit.
- This unit, as its name suggests, decodes the opcode and finds out what is to be done in response of the coming opcode and how many machine cycles are required to execute this instruction.

Timing & Control unit

- The control unit generates signals within microprocessor to carry out the instruction, which has been decoded.
- In reality, it causes certain connections between blocks of the microprocessor to be opened or closed, so that the data goes where it is required and the ALU operations occur.

Arithmetic & Logical Unit:

- The ALU performs the actual numerical and logical operation such as ‘add’, ‘subtract’, ‘AND’, ‘OR’, etc.
- ALU uses data from memory and from accumulator to perform the arithmetic operations and always stores the result of the operation in accumulator.
- ALU consists of accumulator, flag register and temporary register.

Accumulator

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

Flags register

- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- They are called zero (Z), carry (CY), sign (S), parity (P) and auxiliary carry (AC) flags; their bit positions in the flag register are shown in fig.
- The microprocessor uses these flags to set and test data conditions.

Interrupt Control

- The interrupt control unit has 5 interrupt inputs TRAP,RST 7.5, RST 6.5, RST 5.5 & INTR and one acknowledge signal INTA.

- It controls the interrupt activity of 8085 microprocessor.

Serial IO control

- 8085 serial IO control provides two lines, SOD and SID for serial communication.
- The serial output data (SOD) line is used to send data serially and serial input data line (SID) is used to receive data serially.

3. Explain Flags Registers in 8085

- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- They are called zero (Z), carry (CY), sign (S), parity (P) and auxiliary carry (AC) flags; their bit positions in the flag register are shown in fig.
- The microprocessor uses these flags to set and test data conditions.

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

Figure: Flags registers in 8085.

- The flags are stored in the 8-bit register so that the programmer can examine these flags by accessing the register through an instruction.
- These flags have critical importance in the decision-making process of the microprocessor.
- The conditions (set or reset) of the flags are tested through the software instructions.
- For instance, JC (jump on carry) is implemented to change the sequence of a program when CY flag is set.

Z (Zero) Flag:

- This flag indicates whether the result of mathematical or logical operation is zero or not.
- If the result of the current operation is zero, then this flag will be set, otherwise reset.

CY (Carry) Flag:

- This flag indicates, whether, during an addition or subtraction operation, carry or borrow is generated or not, if generated then this flag bit will be set.

AC (Auxiliary Carry) Flag:

- It shows carry propagation from D3 position to D4 position.

- As shown in the fig., a carry is generated from D3 bit position and propagates to the D4 position. This carry is called auxiliary carry.

S (Sign) Flag:

- Sign flag indicates whether the result of a mathematical operation is negative or positive.
- If the result is positive, then this flag will reset and if the result is negative this flag will be set.
- This bit, in fact, is a replica of the D7 bit.

P (Parity) Flag:

- Parity is the number of 1's in a number.
- If the number of 1's in a number is even then that number is known as even parity number.
- If the number of 1's in a number is odd then that number is known as an odd parity number.
- This flag indicates whether the current result is of even parity (set) or of odd parity (reset).

4. Explain 8085 pin diagram.

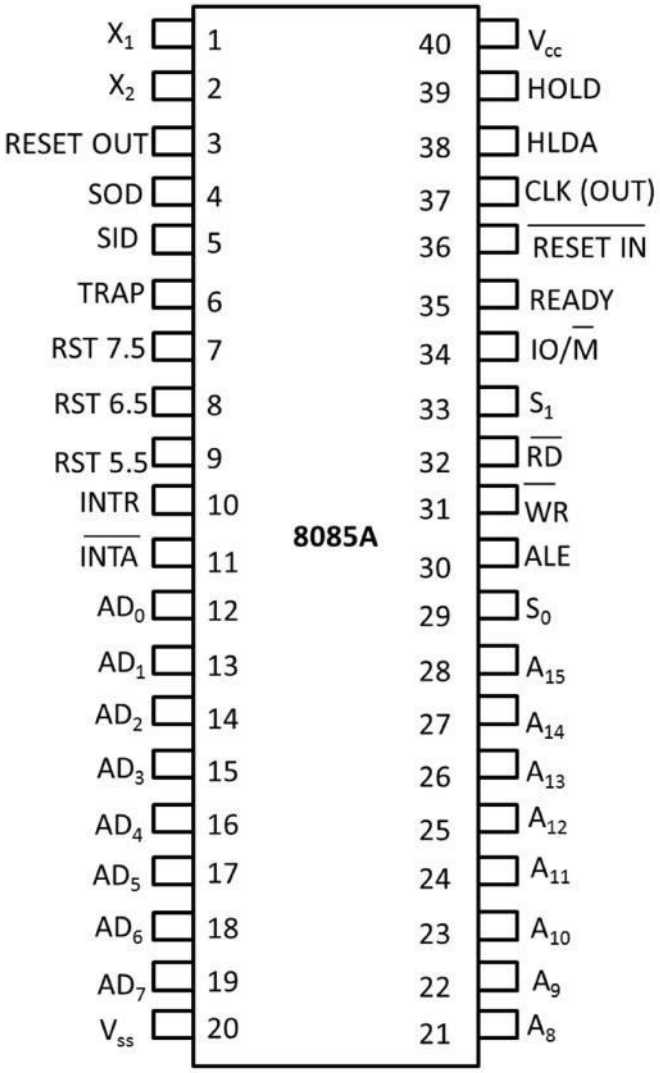


Figure: 8085 pin diagram.

- All signals can be classified into six groups:
  1. Address Bus
  2. Data Bus
  3. Control & Status Signals
  4. Power Supply & Frequency signals
  5. Externally initiated signals
  6. Serial I/O Ports

1) Address Bus (pin 12 to 28)

- 16 signal lines are used as address bus.
- However these lines are split into two segments: A_{15} - A_8 and AD_7 - AD_0
- A_{15} - A_8 are unidirectional and are used to carry high-order address of 16-bit address.
- AD_7 - AD_0 are used for dual purpose.

2) Data Bus/ Multiplexed Address (pin 12 to 19)

- Signal lines AD7-AD0 are bidirectional and serve dual purpose.
- They are used as low-order address bus as well as data bus.
- The low order address bus can be separate from these signals by using a latch.

3) Control & Status Signals

- To identify nature of operation
- Two Control Signals
  1) RD' (Read-pin 32)
     - ✓ This is a read control signal (active low)
     - ✓ This signal indicates that the selected I/O or Memory device is to be read & data are available on data bus.
  2) WR' (Write-pin 31)
     - ✓ This is a write control signal (active low)
     - ✓ This signal indicates that the selected I/O or Memory device is to be write.
- Three Status Signals
  1) S_1 (pin 33)
  2) S_0 (pin 29)
     - ✓ S_1 and S_0 status signals can identify various operations, but they are rarely used in small systems.

S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

  3) IO/M' (pin 34)
     - ✓ This is a status signal used to differentiate I/O and memory operation
     - ✓ When it is high, it indicates an I/O operation
     - ✓ When it is low, it indicates a memory operation
     - ✓ This signal is combined with RD' and WR' to generate I/O & memory control signals
- To indicate beginning of operation
  - o One Special Signal called ALE (Address Latch Enable-Pin 30)
  - o This is positive going pulse generated every time the 8085 begins an operation (machine cycle)
  - o It indicates that the bits on AD7-AD0 are address bits
  - o This signal is used primarily to latch the low-address from multiplexed bus & generate a separate set of address lines A7-A0.

4) Power Supply & FrequencySignal

- V_{cc}→Pin no. 40, +5V Supply
- V_{ss} → Pin no.20, Ground Reference
- X1, X2 → Pin no.1 & 2, Crystal Oscillator is connected at these two pins. The frequency is internally divided by two;
  - Therefore, to operate a system at 3MHz, the crystal should have a frequency of 6MHz.
- CLK (OUT) → Clock output. Pin No.37: This signal can be used as the system clock for other devices.

5) Externally Initiated Signals including Interrupts

- INTR (Input) → Interrupt Request. It is used as general purpose interrupt
- INTA' (Output) → Interrupt Acknowledge. It is used to acknowledge an interrupt.
- RST7.5, RST6.5, RST5.5 (Input) → Restart Interrupts.
  - These are vector interrupts that transfer the program control to specific memory locations.
  - They have higher priorities than INTR interrupt.
  - Among these 3 interrupts, the priority order is RST7.5, RST6.5, RST5.5
- TRAP (Input) → This is a non maskable interrupt & has the highest priority.
- HOLD (Input) → This signal indicates that a peripheral such as DMA Controller is requesting the use of address & data buses
- HLDA (Output) → Hold Acknowledge. This signal acknowledges the HOLD request
- READY (Input) → This signal is used to delay the microprocessor read or write cycles until as low-responding peripheral is ready to send or accept data. When the signal goes low, the microprocessor waits for an integral no. of clock cycles until it goes high.
- RESET IN' (Input) → When the signal on this pin goes low, the Program Counter is set to zero, the buses are tri-stated & microprocessor is reset.
- RESET OUT (Output) → This signal indicates that microprocessor is being reset. The signal can be used to reset other devices.

6) Serial I/O Ports

- Two pins for serial transmission
  1) SID (Serial Input Data-pin 5)
  2) SOD (Serial Output Data-pin 4)
- In serial transmission, data bits are sent over a single line, one bit at a time.

5. Explain Instruction Cycle

- Instruction Cycle is defined as time required to complete execution of an instruction.
- 8085 instruction cycle consists of 1 to 6 Machine Cycles or 1 to 6 operations.

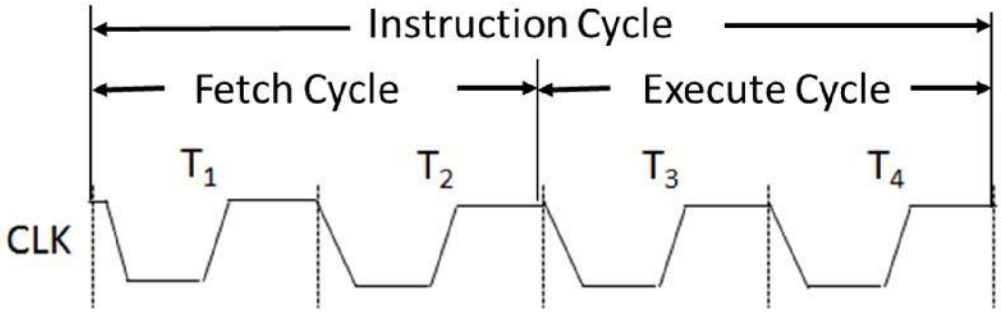


Figure: Instruction Cycle.

6. Explain Machine Cycle

- Machine Cycle is defined as time required by the microprocessor to complete operation of accessing memory device or I/O device.
- This cycle may consist 3 to 6 T-states.
- The basic microprocessor operation such as reading a byte from I/O port or writing a byte to memory is called as machine cycle.

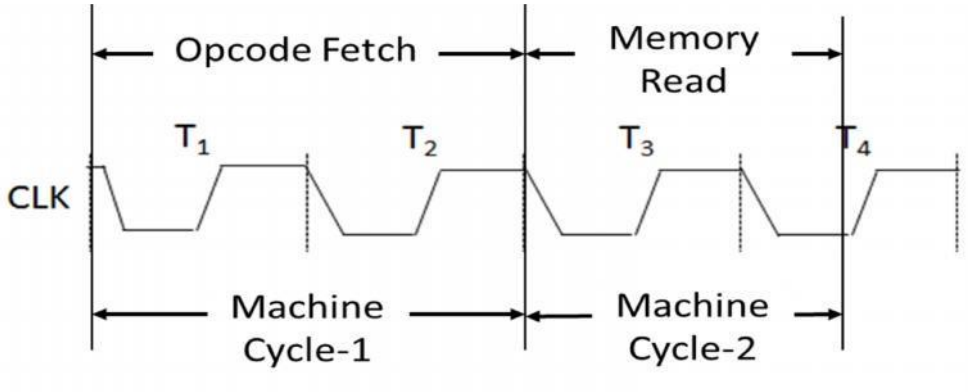


Figure: Machine Cycle.

7. Explain T-States

- T-States are defined as one subdivision of operation performed in one clock period.
- These sub divisions are internal states synchronized with system clock & each T-state is precisely equal to one clock period.

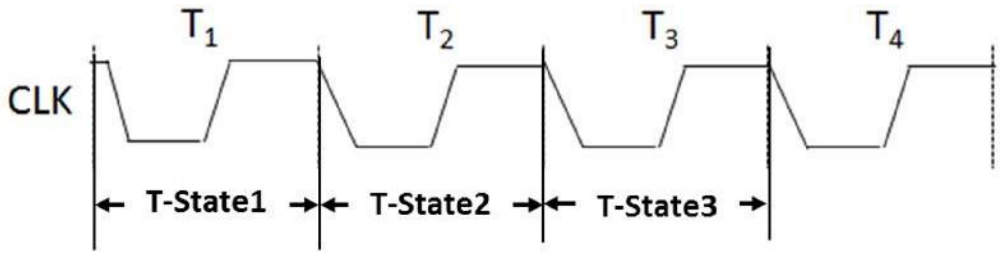


Figure: T-States.

8. Compare Instruction Cycle, Machine Cycle and T-States

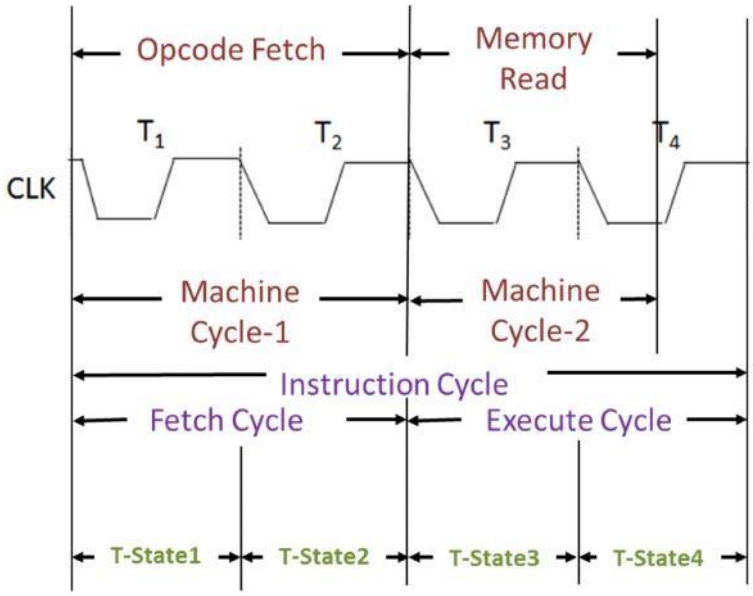


Figure: Comparison between Instruction Cycle, Machine Cycle and T-States.

- Instruction Cycle: Time required to complete execution of an instruction.
- Machine Cycle: Time required by the microprocessor to complete an operation.
- T-States: One subdivision of operation performed in one clock period.

9. Explain 8085 Programming Model

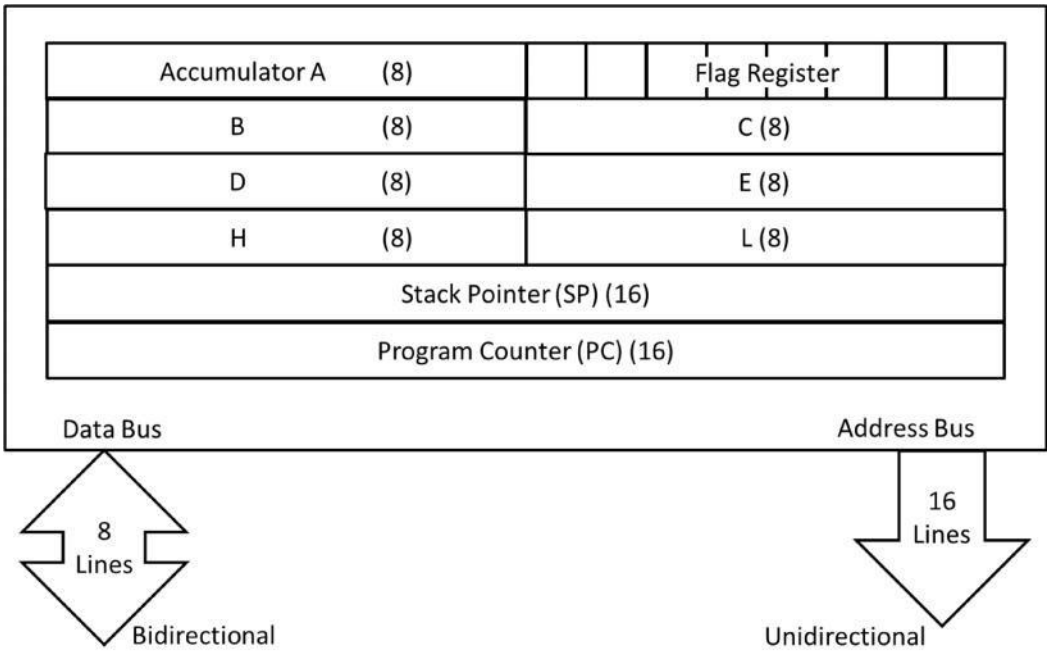


Figure: 8085 Programming Model.

Registers

- 6 general purpose registers to store 8-bit data B, C, D, E, H & L.
- Can be combined as register pairs – BC, DE, and HL to perform 16-bit operations.
- Used to store or copy data using data copy instructions.

Accumulator

- 8 - bit register, identified as A
- Part of ALU
- Used to store 8-bit data to perform arithmetic & logical operations.
- Result of operation is stored in it.

Flag Register

- ALU has 5 Flag Register that set/reset after an operation according to data conditions of the result in accumulator & other registers.
- Helpful in decision making process of Microprocessor
- Conditions are tested through software instructions
- For e.g.
- JC (Jump on Carry) is implemented to change the sequence of program when CY is set.

Program Counter

- 16-bit registers used to hold memory addresses.
- Size is 16-bits because memory addresses are of 16-bits.

- Microprocessor uses PC register to sequence the execution of instructions.
- Its function is to point to memory address from which next byte is to be fetched.
- When a byte is being fetched, PC is incremented by 1 to point to next memory location.

Stack Pointer

- Used as memory pointer
- Points to the memory location in R/W memory, called Stack.
- Beginning of stack is defined by loading a 16-bit address in the stack pointer.

10. Explain Bus Organization of 8085

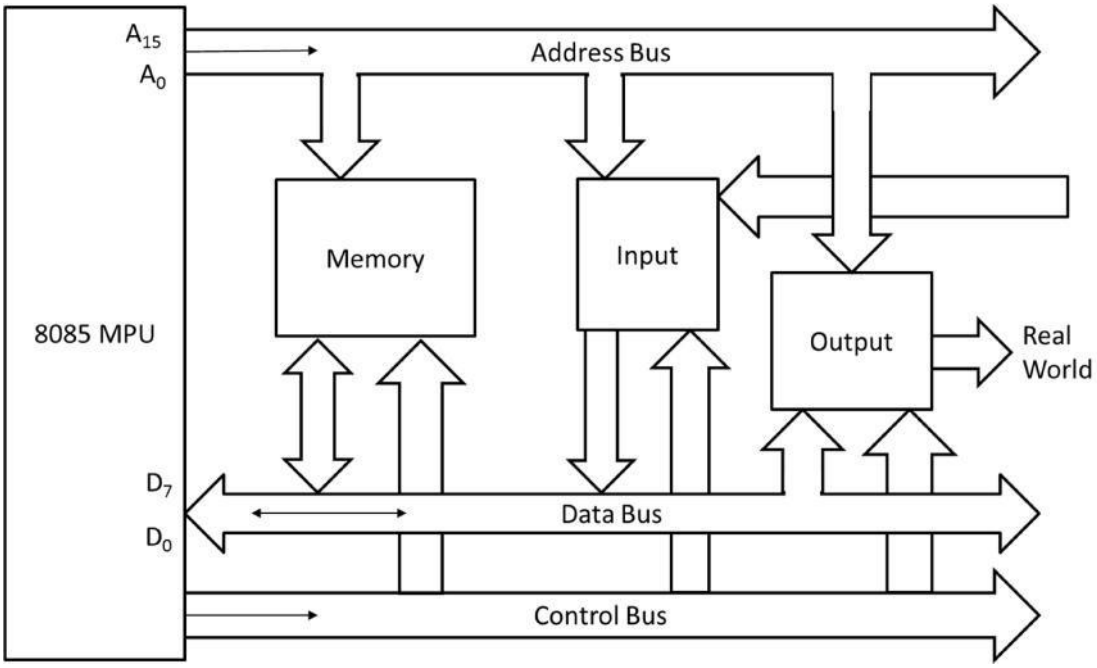


Figure: Bus Organization of 8085.

Address Bus

- Group of 16 lines generally identified as A0 to A15.
- It is unidirectional i.e. bits flow from microprocessor to peripheral devices.
- 16 address lines are capable of addressing 65536 memory locations.
- So, 8085 has 64K memory locations.

Data Bus

- Group of 8 lines identified as D0 to D7.
- They are bidirectional i.e. data flow in both directions between microprocessor, memory & peripheral.
- 8 data lines enable microprocessor to manipulate data ranging from 00H to FFH (28=256 numbers).
- Largest number appear on data bus is 1111 1111 => (255)10.
- As Data bus is of 8-bit, 8085 is known as 8-bit Microprocessor.

Control Bus

- It comprises of various single lines that carry synchronization, timing & control signals.

- These signals are used to identify a device type with which MPU intends to communicate.

11. Explain Demultiplexing AD0-AD7

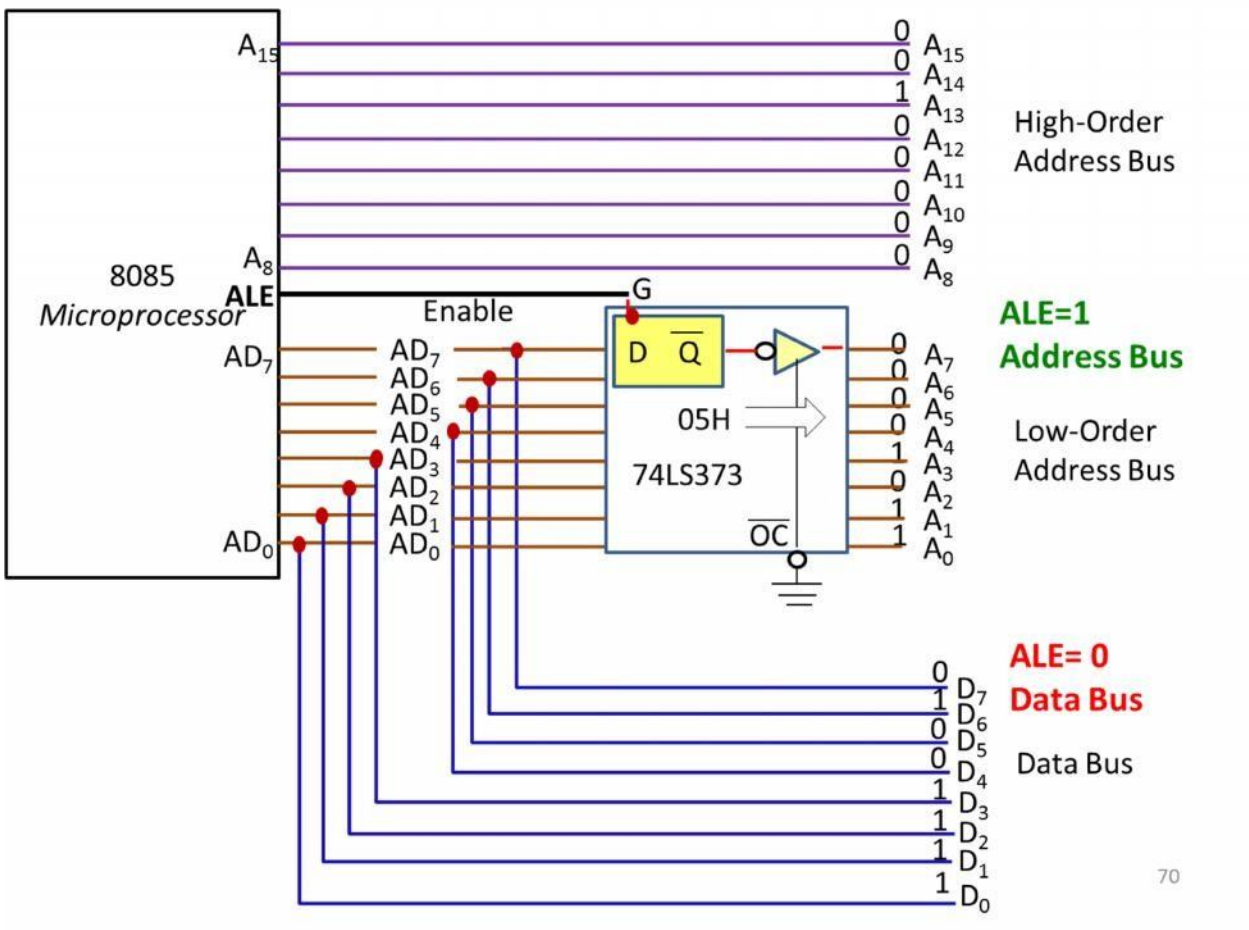


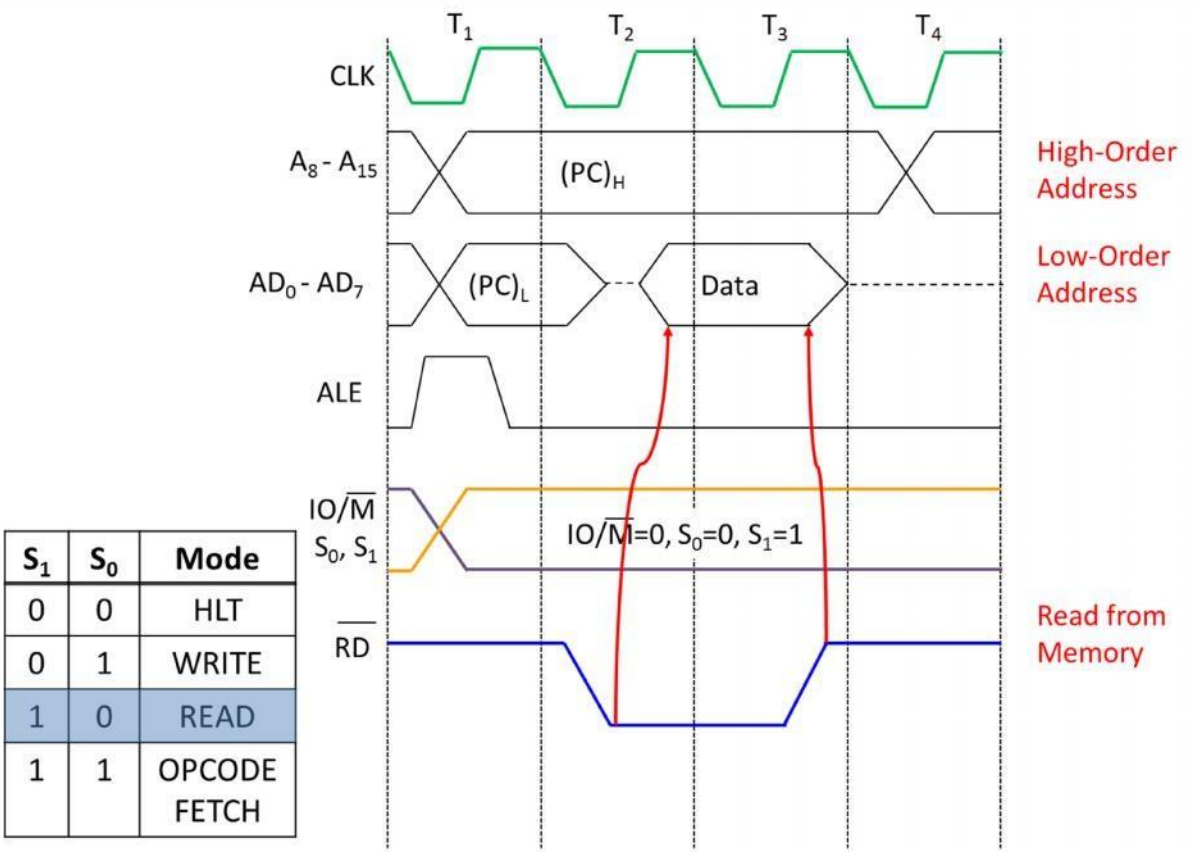
Figure: Demultiplexing AD0-AD7.

- The higher-order bus remains on the bus for three clock periods. However, the low-order address is lost after the first clock period.
- This address need to be latched and used for identifying the memory address. If the bus AD7-AD0 is used to identify the memory location (2005H), the address will change to 204FH after the first clock period.
- Figure shows a schematic that uses a latch and the ALE signal to demultiplex the bus.
- The bus AD7-AD0 is connected as the input to the latch.
- The ALE signal is connected to the Enable pin of the latch, and the output control signal of the latch is grounded.
- Figure shows that the ALE goes high during T1. And during T1 address of lower-order address bus is store into the latch.

12. Explain Memory Interfacing

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

Memory Read Cycle

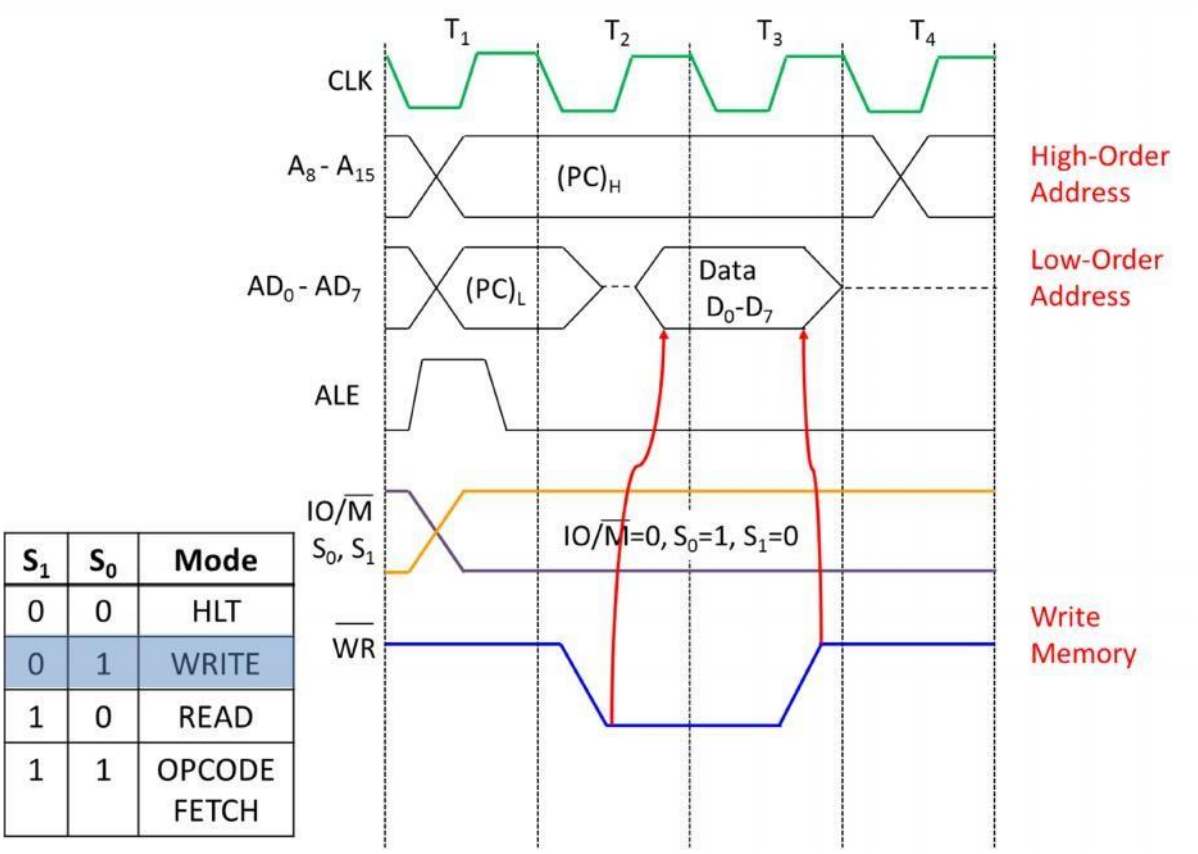


S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

Figure: Memory Read Cycle.

- It is used to fetch one byte from the memory.
- It requires 3 T-States.
- It can be used to fetch operand or data from the memory.
- During T1, A8-A15 contains higher byte of address. At the same time ALE is high. Therefore Lower byte of address A0-A7 is selected from AD0-AD7.
- Since it is memory ready operation, IO/M (bar) goes low.
- During T2 ALE goes low, RD (bar) goes low. Address is removed from AD0-AD7 and data D0-D7 appears on AD0-AD7.
- During T3, Data remains on AD0-AD7 till RD (bar) is at low signal.

Memory Write Cycle



S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

Figure: Memory Write Cycle.

- It is used to send one byte into memory.
- It requires 3 T-States.
- During T1, ALE is high and contains lower address A0-A7 from AD0-AD7.
- A8-A15 contains higher byte of address.
- As it is memory operation, IO/M (bar) goes low.
- During T2, ALE goes low, WR (bar) goes low and Address is removed from AD0-AD7 and then data appears on AD0-AD7.
- Data remains on AD0-AD7 till WR (bar) is low.

13. Explain how Control Signals Generated in 8085

Operation	I0/M'	RD'	WR'
MEMR'	0	0	X
MEMW'	0	X	0
IOR'	1	0	X
IOW'	1	X	0

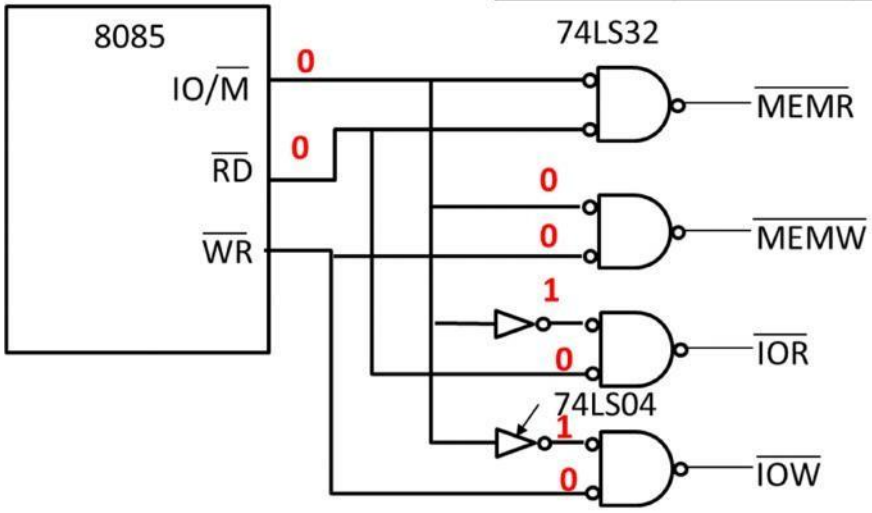


Figure: Control Signals Generated in 8085.

- Figure shows that four different control signals are generated by combining the signals RD (bar), WR (bar), and IO/M (bar).
- The signal IO/M (bar) goes low for the memory operation. This signal is ANDed with RD (bar) and WR (bar) signals b using the 74LS32 quadruple two-input OR gates, as shown in figure 4.5.
- The OR gates are functionally connected as negative NAND gates. When both input signals go low, the output of the gates go low and generate MEMR (bar) and MEMW (bar) control signals.
- When the IO/M (bar) signal goes high, it indicates the peripheral I/O operation.
- Figure shows that this signal is complemented using the Hex inverter 74LS04 and ANDed with the RD (bar) and WR (bar) signals to generate IOR (bar) and IOW (bar) control signals.

1. 8085 instruction set.

Sr.	Instruction	Description	Example
DATA TRANSFER INSTRUCTIONS			

1.	MOV R_d, R_s
	MOV M, R_s
	MOV R_s, M
		This instruction copies the contents of the source 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.	MOV B, C
			MOV B, M

2.	MVI Rd, data MVI M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers.	MVI B, 57H MVI M, 57H
3.	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.	LDA 2034H
4.	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.	LDAX B
5.	LXI Reg.-pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand.	LXI H, 2034H LXI H, XYZ
6.	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.	LHLD 2040H
7.	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.	STA 4350H
8.	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.	STAX B

Sr.	Instruction	Description	Example
9.	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.	SHLD 2470H
10.	XCHG	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.	XCHG
11.	SPHL	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.	SPHL
12.	XTHL	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.	XTHL
13.	PUSH Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.	PUSH B PUSH A
14.	POP Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.	POP H POP A
15.	OUT 8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand.	OUT F8H
16.	IN 8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator.	IN 8CH

ARITHMETIC INSTRUCTIONS

Sr.	Instruction	Description	Example
17.	ADD R ADD M	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADD B ADD M
18.	ADC R ADC M	The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADC B ADC M
19.	ADI 8-bit data	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.	ADI 45H
20.	ACI 8-bit data	The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.	ACI 45H
21.	DAD Reg. pair	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.	DAD H
22.	SUB R SUB M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SUB B SUB M
23.	SBB R SBB M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SBB B SBB M

Sr.	Instruction	Description	Example
24.	SUI 8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SUI 45H
25.	SBI 8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SBI 45H
26.	INR R INR M	The contents of the designated register or memory are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	INR B INR M
27.	INX R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place.	INX H
28.	DCR R DCR M	The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	DCR B DCR M
29.	DCX R	The contents of the designated register pair are decremented by 1 and the result is stored in the same place.	DCX H
30.	DAA	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. 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.	DAA

Sr.	Instruction	Description	Example
BRANCHING INSTRUCTIONS			
31.	JMP 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.	JMP 2034H JMP XYZ
	Jump conditionally specified	*The program sequence is transferred to the memory location by the 16-bit address given in the operand based on the specified flag of the PSW as described below.*	
32.	JC 16-bit address	Jump on Carry, Flag Status: CY=1	JC 2050H
33.	JNC 16-bit address	Jump on no Carry, Flag Status: CY=0	JNC 2050H
34.	JP 16-bit address	Jump on positive, Flag Status: S=0	JP 2050H
35.	JM 16-bit address	Jump on minus, Flag Status: S=1	JM 2050H
36.	JZ 16-bit address	Jump on zero, Flag Status: Z=1	JZ 2050H
37.	JNZ 16-bit address	Jump on no zero, Flag Status: Z=0	JNZ 2050H
38.	JPE 16-bit address	Jump on parity even, Flag Status: P=1	JPE 2050H
39.	JPO 16-bit address	Jump on parity odd, Flag Status: P=0	JPO 2050H
40.	CALL 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.	CALL 2034H CALL XYZ
	Call conditionally specified	*The program sequence is transferred to the memory location by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.*	
41.	CC 16-bit address	Call on Carry, Flag Status: CY=1	CC 2050H
42.	CNC 16-bit address	Call on no Carry, Flag Status: CY=0	CNC 2050H
43.	CP 16-bit address	Call on positive, Flag Status: S=0	CP 2050H
44.	CM 16-bit address	Call on minus, Flag Status: S=1	CM 2050H
45.	CZ 16-bit address	Call on zero, Flag Status: Z=1	CZ 2050H
46.	CNZ 16-bit address	Call on no zero, Flag Status: Z=0	CNZ 2050H
47.	CPE 16-bit address	Call on parity even, Flag Status: P=1	CPE 2050H
48.	CPO 16-bit address	Call on parity odd, Flag Status: P=0	CPO 2050H

Sr.	Instruction	Description	Example
49.	RET	The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.	RET
Return conditionally	*from subroutine*	*The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.*	
50.	RC	Return on Carry, Flag Status: CY=1	RC
51.	RNC	Return on no Carry, Flag Status: CY=0	RNC
52.	RP	Return on positive, Flag Status: S=0	RP
53.	RM	Return on minus, Flag Status: S=1	RM
54.	RZ	Return on zero, Flag Status: Z=1	RZ
55.	RNZ	Return on no zero, Flag Status: Z=0	RNZ
56.	RPE	Return on parity even, Flag Status: P=1	RPE
57.	RPO	Return on parity odd, Flag Status: P=0	RPO
58.	PCHL	The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.	PCHL
59.	RST 0-7	The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:	RST 3

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

Sr.	Instruction	Description	Example
The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware.			
60.	TRAP	It restart from address 0024H	TRAP
61.	RST 5.5	It restart from address 002CH	RST 5.5
62.	RST 6.5	It restart from address 0034H	RST 6.5
63.	RST 7.5	It restart from address 003CH	RST 7.5
LOGICAL INSTRUCTIONS			

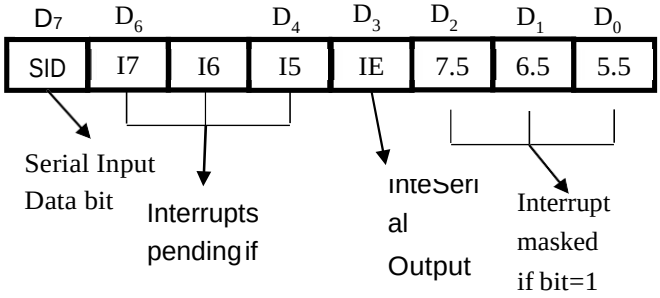
64.	CMP R
	CMP M
		The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows:
		if (A) < (reg/mem): carry flag is set
		if (A) = (reg/mem): zero flag is set
		if (A) > (reg/mem): carry and zero flags are reset
			CMP B
			CMP M

65.	CPI 8-bit data	The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
		if (A) < data: carry flag is set
		if (A) = data: zero flag is set
		if (A) > data: carry and zero flags are reset
			CPI 89H

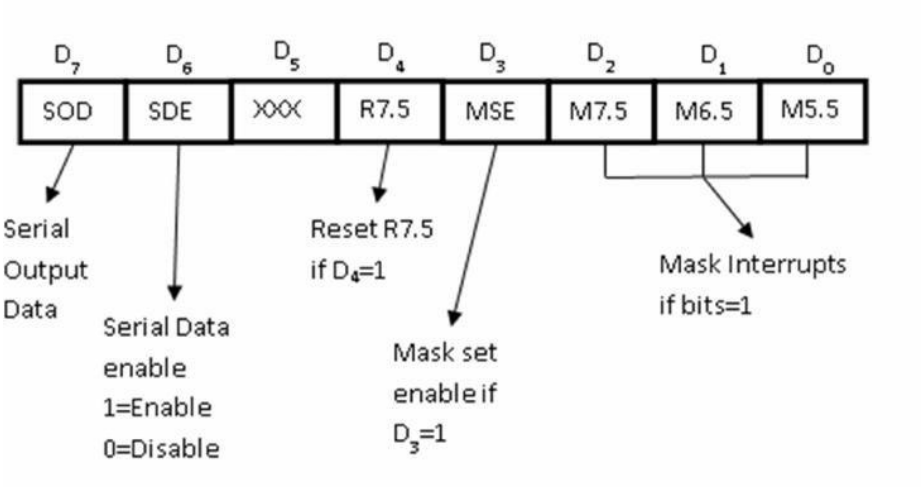
66.	ANA R ANA M	The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANA B ANA M
67.	ANI 8-bit data	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANI 86H
68.	XRA R XRA M	The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRA B XRA M

Sr.	Instruction	Description	Example
69.	XRI 8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRI 86H
70.	ORA R ORA M	The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORA B ORA M
71.	ORI 8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORI 86H
72.	RLC	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. S, Z, P, AC are not affected.	RLC
73.	RRC	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. S, Z, P, AC are not affected.	RRC
74.	RAL	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. S, Z, P, AC are not affected.	RAL
75.	RAR	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.	RAR
76.	CMA	The contents of the accumulator are complemented. No flags are affected.	CMA
77.	CMC	The Carry flag is complemented. No other flags are affected.	CMC
78.	STC	The Carry flag is set to 1. No other flags are affected.	STC
CONTROL INSTRUCTIONS			
79.	NOP	No operation is performed. The instruction is fetched and decoded. However no operation is executed.	NOP

Sr.	Instruction	Description	Example
80.	HLT	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.	HLT
81.	DI	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.	DI
82.	EI	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to re enable the interrupts (except TRAP).	EI
83.	RIM	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.	RIM



84.	SIM	This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.	SIM



1. Stack

- Stack is a group of memory location in the R/W memory that is used for temporary storage of binary information during execution of a program.
- The starting memory location of the stack is defined in program and space is reserved usually at the high end of memory map.
- The beginning of the stack is defined in the program by using instruction **LXI SP, 16-bit memory address**. Which loads a 16-bit memory address in stack pointer register of microprocessor.
- Once stack location is defined storing of data bytes begins at the memory address that is one less then address in stack pointer register. LXI SP, 2099h the storing of data bytes begins at 2098H and continues in reversed numerical order.

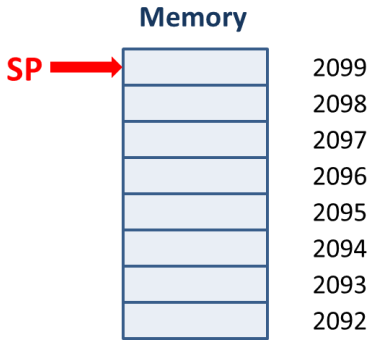


Fig. Stack

- Data bytes in register pair of microprocessor can be stored on the stack in reverse order by using the PUSH instruction.
- PUSH B instruction sore data of register pair BC on sack.

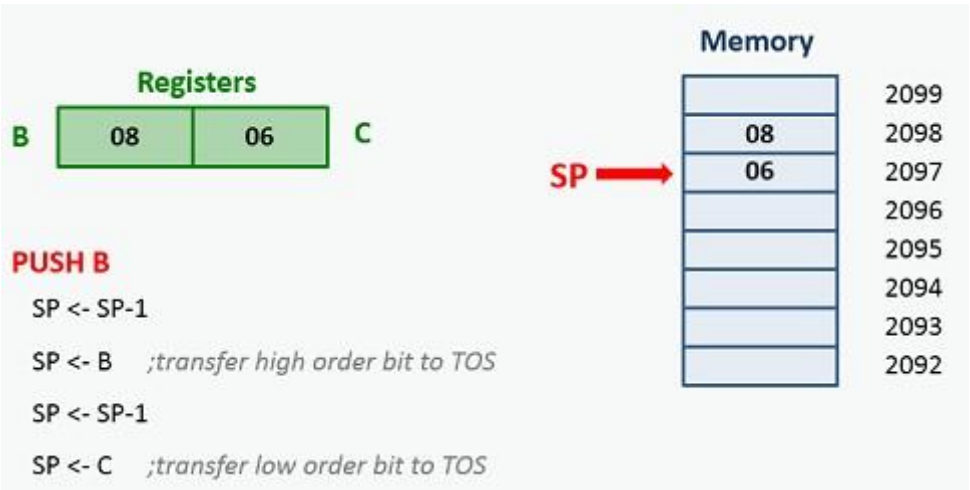


Fig. PUSH operation on stack

- Data bytes can be transferred from the stack to respective registers by using instruction POP.

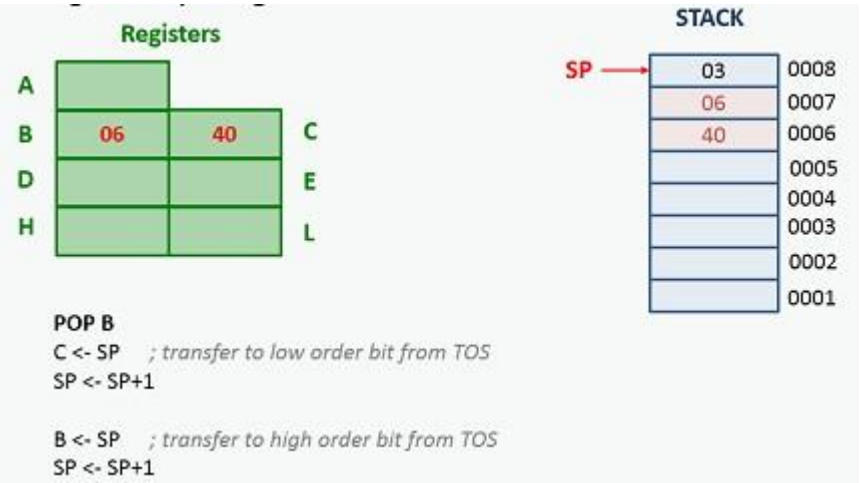


Fig. POP operation on stack

Instruction necessary for stack in 8085

LXI SP, 2095	Load the stack pointer register with a 16-bit address.
PUSH B/D/H	It copies contents of B-C/D-E/H-L register pair on the stack.
PUSH PSW	Operand PSW represents Program status word meaning contents of accumulator and flags.
POP B/D/H	It copies content of top two memory locations of the stack in to specified register pair.
POP PSW	It copies content of top two memory locations of the stack in to B-C accumulator and flags respectively.

2. Subroutine

- A subroutine is a group of instruction that performs a subtask of repeated occurrence.
- A subroutine can be used repeatedly in different locations of the program.

Advantage of using Subroutine

- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.

Where to write Subroutine?

- In Assembly language, a subroutine can exist anywhere in the code.
- However, it is customary to place subroutines separately from the main program.

Instructions for dealing with subroutines in 8085.

- The **CALL** instruction is used to redirect program execution to the subroutine.
  - When CALL instruction is fetched, the Microprocessor knows that the next two **new** Memory location contains 16bit subroutine address.
  - Microprocessor Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the **W** register and stores the lower order 8bit of the address in the **Z** register.
  - Push the **Older** address of the instruction immediately following the CALL onto the stack [Return address]
  - Loads the program counter (**PC**) with the **new** 16-bit address supplied with the CALL instruction from **WZ** register.
- The **RET** instruction is used to return.

- Number of PUSH and POP instruction used in the subroutine must be same, otherwise, RET instruction will pick wrong value of the return address from the stack and program will fail.

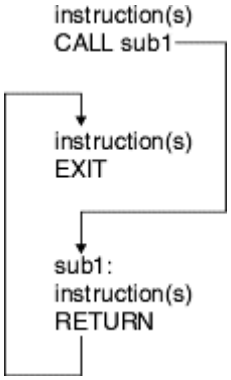


Fig. Subroutine

- Example: write ALP to add two numbers using call and subroutine.

```
LXI H 2000 ; Load memory address of operand
MOV B M ; Store first operand in register B
INX H ;Increment H-L pair
MOV A M ; Store second operand in register A
CALL ADDITION ; Call subroutine ADDITION
STA 3000 ; Store answer
HLT

ADDITION: ADD B ; Add A and B
RET ; Return
```

Conditional call and return instruction available in 8085

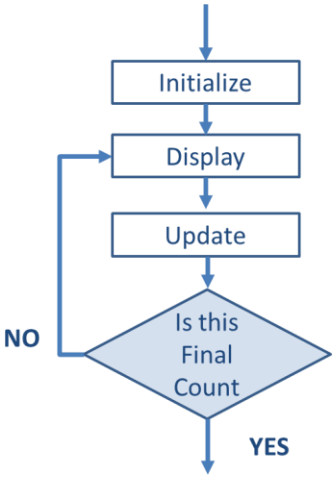
CC 16-bit address	Call on Carry, Flag Status: CY=1
CNC 16-bit address	Call on no Carry, Flag Status: CY=0
CP 16-bit address	Call on positive, Flag Status: S=0
CM 16-bit address	Call on minus, Flag Status: S=1
CZ 16-bit address	Call on zero, Flag Status: Z=1
CNZ 16-bit address	Call on no zero, Flag Status: Z=0
CPE 16-bit address	Call on parity even, Flag Status: P=1
CPO 16-bit address	Call on parity odd, Flag Status: P=0
RC	Return on Carry, Flag Status: CY=1
RNC	Return on no Carry, Flag Status: CY=0
RP	Return on positive, Flag Status: S=0
RM	Return on minus, Flag Status: S=1
RZ	Return on zero, Flag Status: Z=1
RNZ	Return on no zero, Flag Status: Z=0
RPE	Return on parity even, Flag Status: P=1
RPO	Return on parity odd, Flag Status: P=0

3. Applications of Counters and Time Delays

1. Traffic Signal
2. Digital Clocks
3. Process Control
4. Serial data transfer

4. Counters

- 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.




```
1. MVI C,05
2. LOOP: MOV A,C
3.       OUT 01
4.       DCR C
5.    JNZ LOOP
6. HLT
```

Fig. Counter

5. Time Delay

- Each instruction passes through different combinations of Fetch, Memory Read, and Memory Write cycles.
- Knowing the combinations of cycles, one can calculate how long such an instruction would require to complete.
- It is counted in terms of number of T–states required.
- Calculating this time we generate require software delay.

Time Delay Using Single Register

Label	Opcode	Operand	Comment	T-states
	MVI	C,05h	; Load Counter	**7**
LOOP:	DCR	C	; Decrement Counter	**4**
	JNZ	LOOP	; Jump back to Decr. C	**10/7**

MVI C 05 Mchine Cycle: F + R = 2 T-States: 4T + 3T = 7T	DCR C Mchine Cycle: F = 1 T-States: 4T = 4T	JNZ LOOP (true) Mchine Cycle: F + R + R = 3 T-States: 4T + 3T + 3T = 10T	JNZ LOOP (false) Mchine Cycle: F + R = 3 T-States: 4T + 3T = 7T

- Instruction MVI C, 05h requires 7 T-States to execute. Assuming, 8085 Microprocessor with 2MHz clock frequency. How much time it will take to execute above instruction?
  Clock frequency of the system (f) = 2 MHz
  Clock period (T) = 1/f = ½ * 10-6 = 0.5 µs
  Time to execute MVI = 7 T-states * 0.5 µs
  = 3.5 µs
- Now to calculate time delay in loop, we must account for the T-states required for each instruction, and for the number of times instructions are executed in the loop. There for the next two instructions:
  DCR: 4 T-States
  JNZ: 10 T-States
  14 T-States
- Here, the loop is repeated for 5 times.
- Time delay in loop T_L with 2MHz clock frequency is calculated as:
  T_L = T * Loop T-sates * N_{10} ---------- (1)
  T_L : Time Delay in Loop
  T : Clock Frequency
  N_{10} : Equivalent decimal number of hexadecimal count loaded in the delay register.
- Substituting value in equation (1)
  T_L = (0.5 * 10^{-6} * 14 * 5)
  = 35 µs
- If we want to calculate delay more accurately, we need to accurately calculate execution of JNZ instruction i.e
  If **JNZ = true**, then **T-States = 10**
  Else if **JNZ =false**, then **T-States = 7**
- Delay generated by last clock cycle:
  = 3T * Clock Period
  = 3T * (1/2 * 10^{-6})
  = 1.5 µs
- Now, the accurate loop delay is:
  T_{LA} = T_L - Delay generated by last clock cycle
  T_{LA} = 35 µs - 1.5 µs
  T_{LA} = 33.5 µs
- Now, to calculate total time delay
  Total Delay = Time taken to execute instruction outside loop + Time taken to execute loop instructions
  T_D = T_O + T_{LA}
  = (7 * 0.5 µs) + 33.5 µs
  = 3.5 µs + 33.5 µs
  = 37 µs
- In most of the case we are given time delay and need to find value of the counter register which decide number of times loop execute.
- For example: write ALP to generate 37 µs delay given that clock frequency if 2 MHz.
- Single register loop can generate small delay only for large delay we use other technique.

Time Delay Using a Register Pair

- Time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (FFFF h).
- A 16-bit is decremented by using DCX instruction.
- Problem with DCX instruction is DCX instruction doesn't set **Zero** flag.
- Without test flag, Jump instruction can't check desired conditions.
- Additional technique must be used to set Zero flag.

Label	Opcode	Operand	Comment	T-states
	LXI	B,2384 h	; Load BC with 16-bit counter	**10**
LOOP:	DCX	B	; Decrement BC by 1	**6**
	MOV	A, C	; Place contents of C in A	**4**
	ORA	B	; OR B with C to set Zero flag	**4**
	JNZ	LOOP	; if result not equal to 0, 10/7 jump back to loop	**10/7**

- Here the loop includes four instruction:
  Total T-States = 6T + 4T + 4T + 10T
  = 24 T-states
- The loop is repeated for 2384 h times.
- Converting $(2384)_{16}$ into decimal.
  2384 h = $(2 * 16^3) + (3 * 16^2) + (8 * 16^1) + (4 * 16^0)$
  = 8192 + 768 + 128 + 4 = **9092**
- Clock frequency of the system (f)= 2 MHz
- Clock period (T) = 1/f = ½ * 10^{-6} = 0.5 μs
- Now, to find delay in the loop
  T_L = T * Loop T-sates * N_{10}
  = 0.5 * 24 * 9092
  = 109104 μs = 109 ms (without adjusting last cycle)

Time Delay Using a LOOP within a LOOP

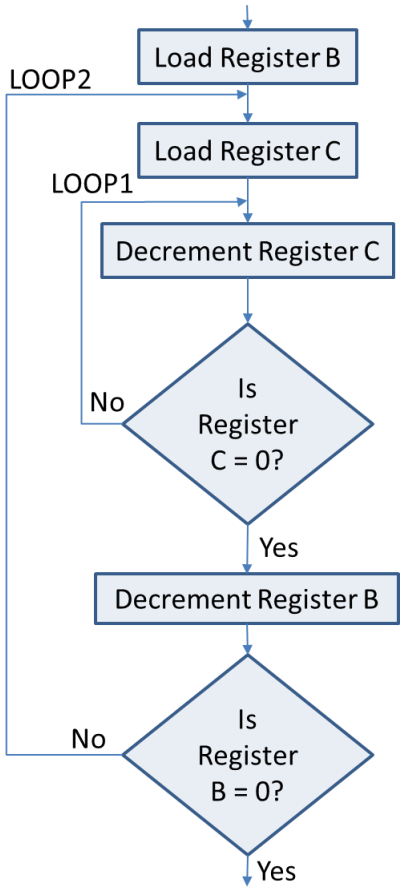


Fig. Time Delay Using a LOOP within a LOOP

Label	Opcode	Operand	T-states
	MVI	B,38h	7T
LOOP2:	MVI	C,FFh	7T
LOOP1:	DCR	C	4T
	JNZ	LOOP1	10/7 T
	DCR	B	4T
	JNZ	LOOP2	10/7 T

- Calculating delay of inner LOOP1: T_{L1}
  T_L= T * Loop T-states * N_{10}
  = 0.5 * 14* 255
  = 1785 μs = 1.8 ms
  T_{L1}= TL – (3T states* clock period)
  = 1785 – (3 * ½ * 10^{-6})
  = 1785-1.5=**1783.5 μs**
- Now, Calculating delay of outer LOOP2: T_{L2}
- Counter B : $(38)_{16}$ = **$(56)_{10}$** So loop2 is executed for 56 times.
  T-States = 7 + 4 + 10 = 21 T-States
  T_{L2} = 56 (T_{L1} + 21 T-States * 0.5)
  = 56(1783.5 μs + 10.5)

= 100464 µs

T_{L2} = 100.46 ms

Disadvantage of using software delay

- Accuracy of time delay depends on the accuracy of system clock.
- The Microprocessor is occupied simply in a waiting loop; otherwise it could be employed to perform other functions.
- The task of calculating accurate time delays is tedious.
- In real time applications timers (integrated timer circuit) are commonly used.
- Intel 8254 is a programmable timer chip that can be interfaced with microprocessor to provide timing accuracy.
- The disadvantage of using hardware chip include the additional expense and the need for extra chip in the system.

6. Counter design with time delay

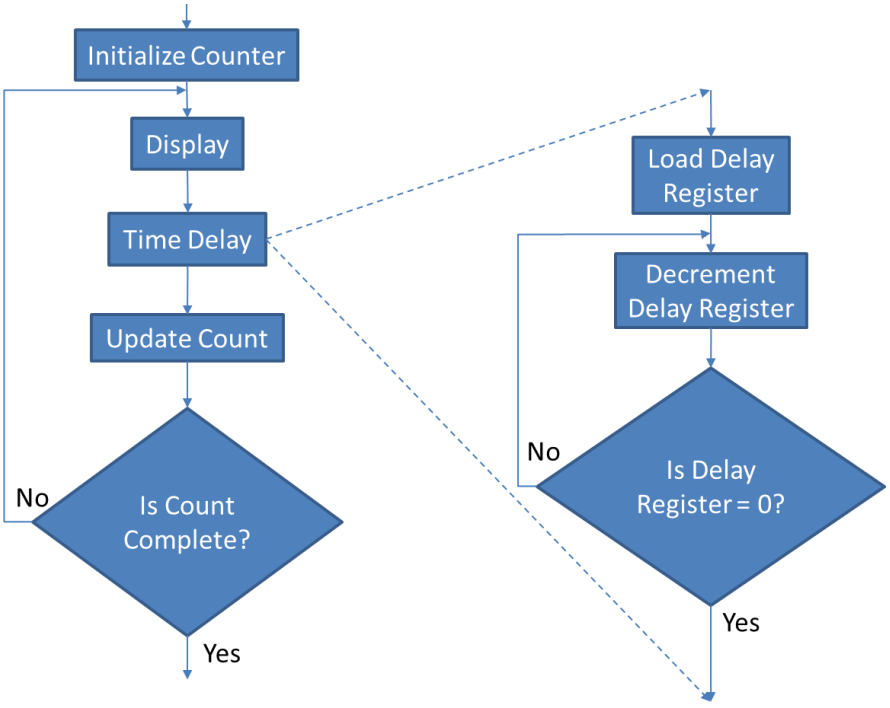


Fig. 6. Counter design with time delay

- It is combination of counter and time delay.
- I consist delay loop within counter program.

7. Hexadecimal counter program

- Write a program to count continuously in hexadecimal from **FFh** to **00h** with **0.5 µs** clock period. Use register C to set up 1 ms delay between each count and display the number at one of the output port.
- **Given:**
- Counter= FF h
- Clock Period T=0.5 µs

- Total Delay = 1ms
- **Output:**
- To find value of delay counter
- **Program**

```
MVI B,FF
LOOP:MOV A,B
    OUT 01
    MVI C, COUNT; need to calculate delay count
    DELAY: DCR C
            JNZ DELAY
    DCR B
    JNZ LOOP
HLT
```

- Calculate Delay for Internal Loop
  TI = T-States * Clock Period * COUNT
  = 14 * 0.5 * 10-6 * COUNT
  TI = (7.0 * 10-6) * COUNT
- Calculate Delay for Outer Loop:
  TO = T-States * Clock Period
  = 35 * 0.5 * 10-6
  TO = 17.5 ?s
- Calculate Total Time Delay:
  TD = TO + TL
  1 ms = 17.5 * 10-6 + (7.0 * 10-6)* COUNT
  1 * 10-3 = 17.5 * 10-6 + (7.0 * 10-6)* COUNT
  COUNT="1 * 10−3 − 17.5 * 10−6" /"7.0 * 10−6"
  COUNT= $(140)_{10} = (8C)_{16}$

8. 0-9 up/down counter program

- Write an 8085 assembly language program to generate a decimal counter (which counts 0 to 9 continuously) with a one second delay in between. The counter should reset itself to zero and repeat continuously. Assume a crystal frequency of 1MHz.
- **Program**

```
    START: MVI B,00H
    DISPLAY: OUT 01
            LXI H, COUNT
    LOOP: DCX H
            MOV A, L
            ORA H
            JNZ LOOP
            INR B
            MOV A,B
            CPI 0A
            JNZ DISPLAY
```

JZ START

9. Code Conversion

Two Digit BCD Number to Binary Number

1. Initialize memory pointer to given address (2000).
2. Get the Most Significant Digit (MSD).
3. Multiply the MSD by ten using repeated addition.
4. Add the Least Significant Digit (LSD) to the result obtained in previous step.
5. Store the HEX data in Memory.

- **Program**

```
LXI H 2000
MOV C M
MOV A C
ANI 0F ; AND operation with 0F (00001111)
MOV E A
MOV A C
ANI F0 ; AND operation with F0 (11110000)
JZ SB1 ; If zero skip further process and directly add LSD
RRC ; Rotate 4 times right
RRC
RRC
RRC
MOV D A
MVI A 00
L1: ADI 0A ; Loop L1 multiply MSD with 10
DCR D
JNZ L1
SB1: ADD E
STA 3000 ; Store result
HLT
```

8-bit Binary Number to Decimal Number

1. Load the binary data in accumulator
2. Compare ‘A’ with 64 (Dicimal 100) if cy = 01, go step 5 otherwise next step
3. Subtract 64H from ‘A’ register
4. Increment counter 1 register
5. Go to step 2
6. Compare the register ‘A’ with ‘0A’ (Dicimal 10), if cy=1, go to step 10, otherwise next step
7. Subtract 0AH from ‘A’ register
8. Increment Counter 2 register
9. Go to step 6
10. Combine the units and tens to from 8 bit result
11. Save the units, tens and hundred’s in memory
12. Stop the program execution

- **Program**

```
MVI B 00
```

```
LDA 2000
LOOP1: CPI 64 ; Compare with 64H
JC NEXT1 : If A is less than 64H then jump on NEXT1
SUI 64 ; subtract 64H
INR B
JMP LOOP1
NEXT1: LXI H 2001
MOV M B ; Store MSD into memory
MVI B 00
LOOP2: CPI 0A ; Compare with 0AH
JC NEXT2 ; If A is less than 0AH then jump on NEXT2
SUI 0A ; subtract 0AH
INR B
JMP LOOP2
NEXT2: MOV D A
MOV A B
RLC
RLC
RLC
RLC
ADD D
STA 2002 ; Store packed number formed with two leas significant digit
HLT
```

Binary Number to ASCII Number

- Load the given data in A - register and move to B - register
- Mask the upper nibble of the Binary decimal number in A - register
- Call subroutine to get ASCII of lower nibble
- Store it in memory
- Move B - register to A - register and mask the lower nibble
- Rotate the upper nibble to lower nibble position
- Call subroutine to get ASCII of upper nibble
- Store it in memory
- Terminate the program.

```
LDA 5000 Get Binary Data
    MOV B, A
    ANI 0F      ; Mask Upper Nibble
    CALL SUB1   ; Get ASCII code for upper nibble
    STA 5001
    MOV A, B
    ANI F0      ; Mask Lower Nibble
    RLC
    RLC
    RLC
    RLC
    CALL SUB1  ; Get ASCII code for lower nibble
    STA 5002
```

```
        HLT         ; Halt the program.

SUB1: CPI 0A
        JC SKIP
        ADI 07
SKIP: ADI 30
        RET         ; Return Subroutine
```

ASCII Character to Hexadecimal Number

1. Load the given data in A - register
2. Subtract 30H from A - register
3. Compare the content of A - register with 0AH
4. If A < 0AH, jump to step6. Else proceed to next step
5. Subtract 07H from A - register
6. Store the result
7. Terminate the program

- **Program**

```
LDA 2000
CALL ASCTOHEX
STA 2001
HLT

ASCTOHEX: SUI 30 ; This block Convert ASCII to Hexadecimal.
CPI 0A
RC
SUI 07
RET
```

10. BCD Arithmetic

Add 2 8-bit BCD Numbers

1. Load firs number into accumulator.
2. Add second number.
3. Apply decimal adjustment to accumulator.
4. Store result.

- **Program**

```
LXI H, 2000H
MOV A, M
INX H
ADD M
DAA
INX H
MOV M, A
HLT
```

Subtract the BCD number stored in E register from the number stored in the D register

1. Find 99's complement of data of register E
2. Add 1 to find 100's complement of data of register E
3. Add Data of Register D
4. Apply decimal adjustment

- **Program**

```
MVI A, 99H
SUB E        : Find the 99's complement of subtrahend INR A      :
Find 100's complement of subtrahend
ADD D        : Add minuend to 100's complement of subtrahend DAA
             : Adjust for BCD
HLT          : Terminate program execution
```

11. 16-Bit Data operations

Add Two 16 Bit Numbers

1. Initialize register C for using it as a counter for storing carry value.
2. Load data into HL register pair from one memory address (9000H).
3. Exchange contents of register pair HL with DE.
4. Load second data into HL register pair (from 9002H).
5. Add register pair DE with HL and store the result in HL.
6. If carry is present, go to 7 else go to 8.
7. Increment register C by 1.
8. Store value present in register pair HL to 9004H.
9. Move content of register C to accumulator A.
10. Store value present in accumulator (carry) into memory (9006H).
11. Terminate the program.

- **Program**

```
MVI C, 00H
LHLD 9000H
XCHG ; Exchange contents of register pair HL with DE LHLD
9002H
DAD D ; Add register pair DE with HL and store the result in HL JNC
AHEAD ; If carry is present, go to AHEAD
INR C
AHEAD: SHLD 9004H ; Store value present in register pair HL to 9004H MOV A, C
STA 9006H ; Store value present in accumulator (carry) into memory (9006H) HLT
```

Subtract Two 16 Bit Numbers

1. Load first data from Memory (9000H) directly into register pair HL.
2. Exchange contents of register pair DE and HL.
3. Load second data from memory location (9002H) directly into register pair HL.