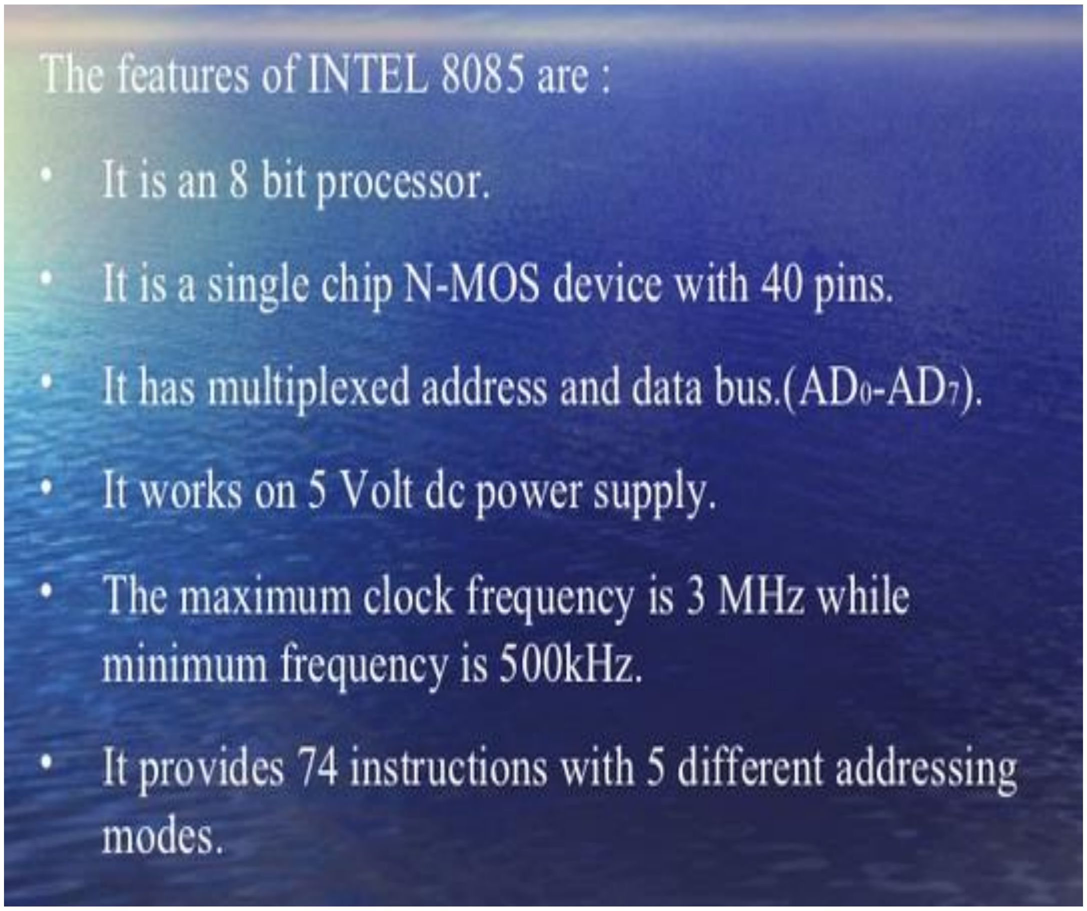
UNIT II 8085 Microprocessor

Lecture 1

Introduction to 8085

1. Introduced in 1977.
2. It is 8-bit MP.
3. It is a 40 pin dual-in-line chip
4. It uses a single +5V supply for its operations.
5. Its clock speed is about 3MHz.



The features of INTEL 8085 are :

- It is an 8 bit processor.
- It is a single chip N-MOS device with 40 pins.
- It has multiplexed address and data bus.(AD_0-AD_7).
- It works on 5 Volt dc power supply.
- The maximum clock frequency is 3 MHz while minimum frequency is 500kHz.
- It provides 74 instructions with 5 different addressing modes.

UNIT II 8085 Microprocessor

Lecture 1,2

8085 Architecture

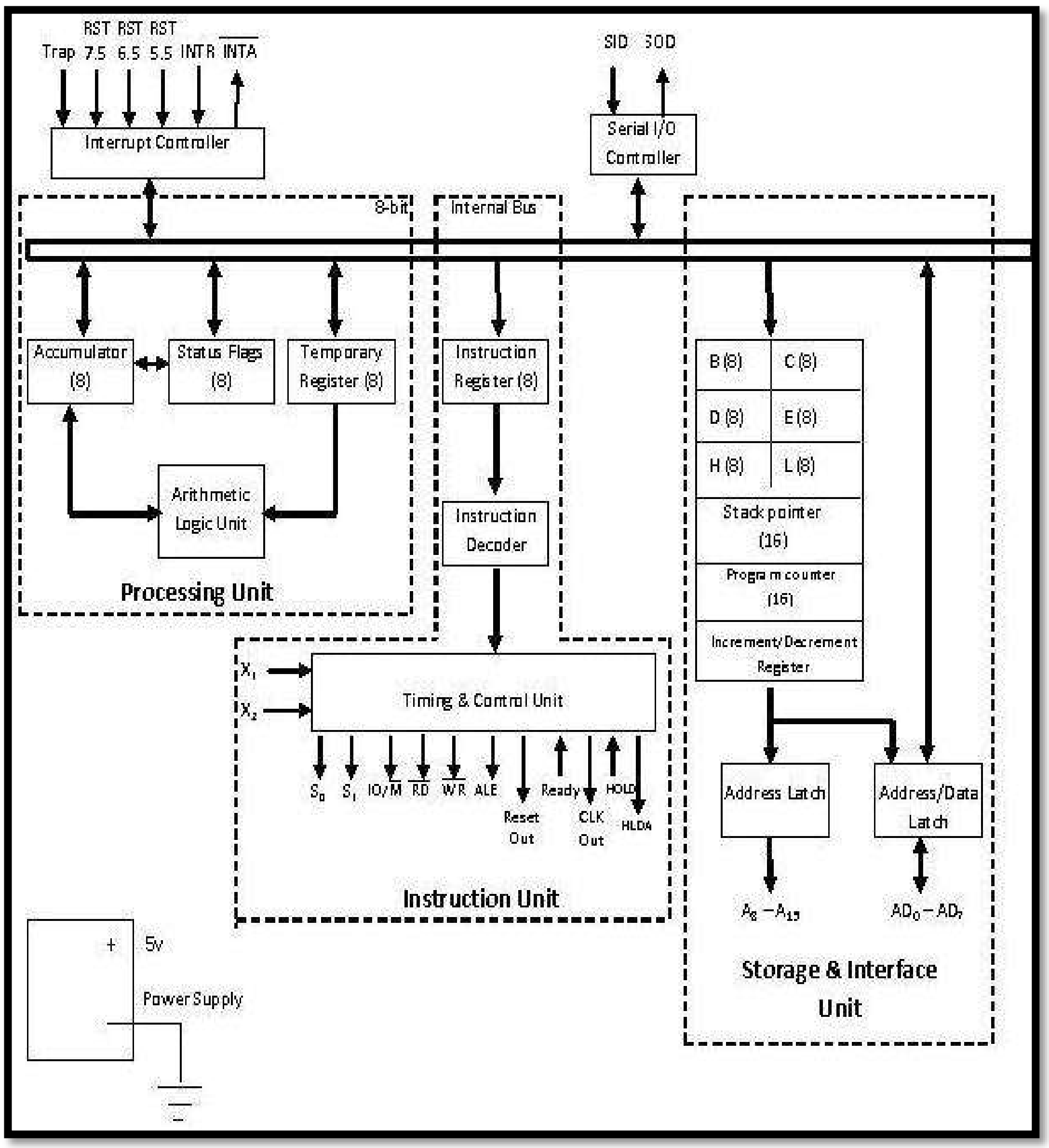


Fig (a) Architecture of 8085

Block Diagram of 8085

Three Units of 8085
 Processing Unit
 Instruction Unit
 Storage and Interface Unit

1.Processing Unit
 Arithmetic and Logic Unit
 Accumulator
 Status Flags
 Temporary Register

2.Instruction Unit
 Instruction Register
 Instruction Decoder
 Timing and Control Unit

3.Storage and Interface Unit
 General Purpose Registers
 Stack Pointer
 Program Counter
 Increment/Decrement Register
 Address Latch
 Address/Data Latch

Three Other Units
 Interrupt Controller
 Serial I/O Controller
 Power Supply

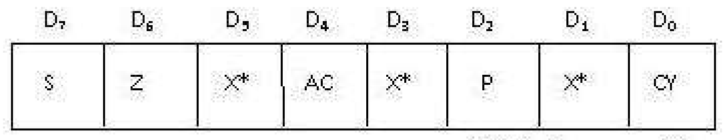
1.1 Accumulator
 It the main register of microprocessor.
 It is also called register 'A'.
 It is an 8-bit register.
 It is used in the arithmetic and logic operations.
 It always contains one of the operands on which arithmetic/logic has to be performed.
 After the arithmetic/logic operation, the contents of accumulator are replaced by the result.

1.2 Arithmetic & Logic Unit (ALU)
 It performs various arithmetic and logic operations.
 The data is available in accumulator and temporary/general purpose registers.
 Arithmetic Operations:

 Addition, Subtraction, Increment, Decrement etc.
 Logic Operations:
 AND, OR, X-OR, Complement etc

1.3 Temporary Register
 It is an 8-bit register.
 It is used to store temporary 8-bit operand from general purpose register.
 It is also used to store intermediate results.

1.4 Status Flags
 Status Flags are set of flip-flops which are used to check the status of Accumulator after the operation is performed.



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

X*=don't care condition

Status Flags
 S = Sign Flag
 Z = Zero Flag
 AC = Auxiliary Carry Flag
 P = Parity Flag
 CY = Carry Flag

Sign Flag (S):
 It tells the sign of result stored in Accumulator after the operation is performed.
 If result is –ve, sign flag is set (1).
 If result is +ve, sign flag is reset (0).

Zero Flag (Z):
 It tells whether the result stored in Accumulator is zero or not after the operation is performed.
 If result is zero, zero flag is set (1).
 If result is not zero, zero flag is reset (0).

Auxiliary Carry Flag (AC):
 It is used in BCD operations.
 When there is carry in BCD addition, we add 0110 (6) to the result.
 If there is carry in BCD addition, auxiliary carry is set (1).
 If there is no carry, auxiliary carry is reset (0).

Parity Flag (P):
 It tells the parity of data stored in Accumulator.
 If parity is even, parity flag is set (1).
 If parity is odd, parity flag is reset (0).

Program Status Word (PSW)
 The contents of Accumulator and Status Flags clubbed together is known as Program Status Word (PSW).
 It is a 16-bit word

ACC (8)	Status Flag (8)

2.1 Instruction Register
 It is used to hold the current instruction which the microprocessor is about to execute.
 It is an 8-bit register.

2.2 Instruction Decoder
 It interprets the instruction stored in instruction register.
 It generates various machine cycles depending upon the instruction.
 The machine cycles are then given to the Timing and Control Unit.

2.3 Timing and Control Unit
 It controls all the operations of microprocessor and peripheral devices.
 Depending upon the machine cycles received from Instruction Decoder, it generates 12 control signals:
 S0 and S1 (Status Signals).
 ALE (Address Latch Enable).

Timing and Control Unit
 RD (Read, active low).
 WR (Write, active low).
 IO/M (Input-Output/Memory).
 READY
 RESET IN
 RESET OUT
 CLK OUT
 HOLD and HLDA

3.1 General Purpose Registers
 There are 6 general purpose registers, namely B, C, D, E, H, L.
 Each of the them is 8-bit register.
 They are used to hold data and results.
 To hold 16-bit data, combination of two 8-bit registers can be used.
 This combination is known as ***Register Pair***.
 The valid register pairs are:
 B – C, D – E, H – L.

3.2 Program Counter
 It is used to hold the address of next instruction to be executed.
 It is a 16-bit register.
 The microprocessor increments the value of Program Counter after the execution of the current instruction, so that, it always points to the next instruction.

3.3 Stack Pointer
 It holds the address of top most item in the stack.
 It is also 16-bit register.
 Any portion of memory can be used as stack.

3.4 Increment/Decrement Register
 This register is used to increment or decrement the value of Stack Pointer.
 During PUSH operation, the value of Stack Pointer is incremented.
 During POP operation, the value of Stack Pointer is decremented.

3.5 Address Latch
 It is group of 8 buffers.
 The upper-byte of 16-bit address is stored in this latch.
 And then it is made available to the peripheral devices.

3.6 Address/Data Latch
 The lower-byte of address and 8-bit of data are multiplexed.
 It holds either lower-byte of address or 8-bits of data.
 This is decided by ALE (Address Latch Enable) signal.
 If ALE = 1 then
 Address/Data Latch contains lower-byte of address.
 If ALE = 0 then
 It contains 8-bit data.

Serial I/O Controller
 It is used to convert serial data into parallel and parallel data into serial.
 Microprocessor works with 8-bit parallel data.
 Serial I/O devices works with serial transfer of data.
 Therefore, this unit is the interface between microprocessor and serial I/O devices.

Interrupt Controller
 It is used to handle the interrupts.
 There are 5 interrupt signals in 8085:
 TRAP
 RST 7.5
 RST 6.5
 RST 5.5

- INTR

Interrupt Controller

- Interrupt controller receives these interrupts according to their priority and applies them to the microprocessor.
- There is one outgoing signal INTA which is called Interrupt Acknowledge.

Power Supply

- This unit provides +5V power supply to the microprocessor.
- The microprocessor needs +5V power supply for its operation.

UNIT II 8085 Microprocessor

Lecture 3

8085 Pin description

Properties

1. Single + 5V Supply
2. 4 Vectored Interrupts (One is Non Maskable)
3. Serial In/Serial Out Port
4. Decimal, Binary, and Double Precision Arithmetic
5. Direct Addressing Capability to 64K bytes of memory

The Intel 8085A is a new generation, complete 8 bit parallel central processing unit (CPU). The 8085A uses a multiplexed data bus. The address is split between the 8bit address bus and the 8bit data bus.

Pin Description

The following describes the function of each pin:

1. A6 - A1s (Output 3 State):-Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 address,3 stated during Hold and Halt modes.
2. AD0 - 7 (Input/output 3state):-Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.
3. ALE (Output):-Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3stated.
4. SO, S1 (Output):-Data Bus Status. Encoded status of the bus cycle:

S1	**S0**	
O	**O**	**HALT**
0	**1**	**WRITE**
1	**0**	**READ**
1	**1**	**FETCH**

S1 can be used as an advanced R/W status.

5. RD (Output 3state):- READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.
6. WR (Output 3state):-WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3stated during Hold and Halt modes.
7. READY (Input) If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high

before completing the read or write cycle.

8. HOLD (Input):-HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU, upon receiving the Hold request. will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.
9. HLDA (Output):-HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.
10. INTR (Input):-INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.
11. INTA (Output):-INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5
RST 6.5 - (Inputs)
RST 7.5

12. RESTART INTERRUPTS; These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.

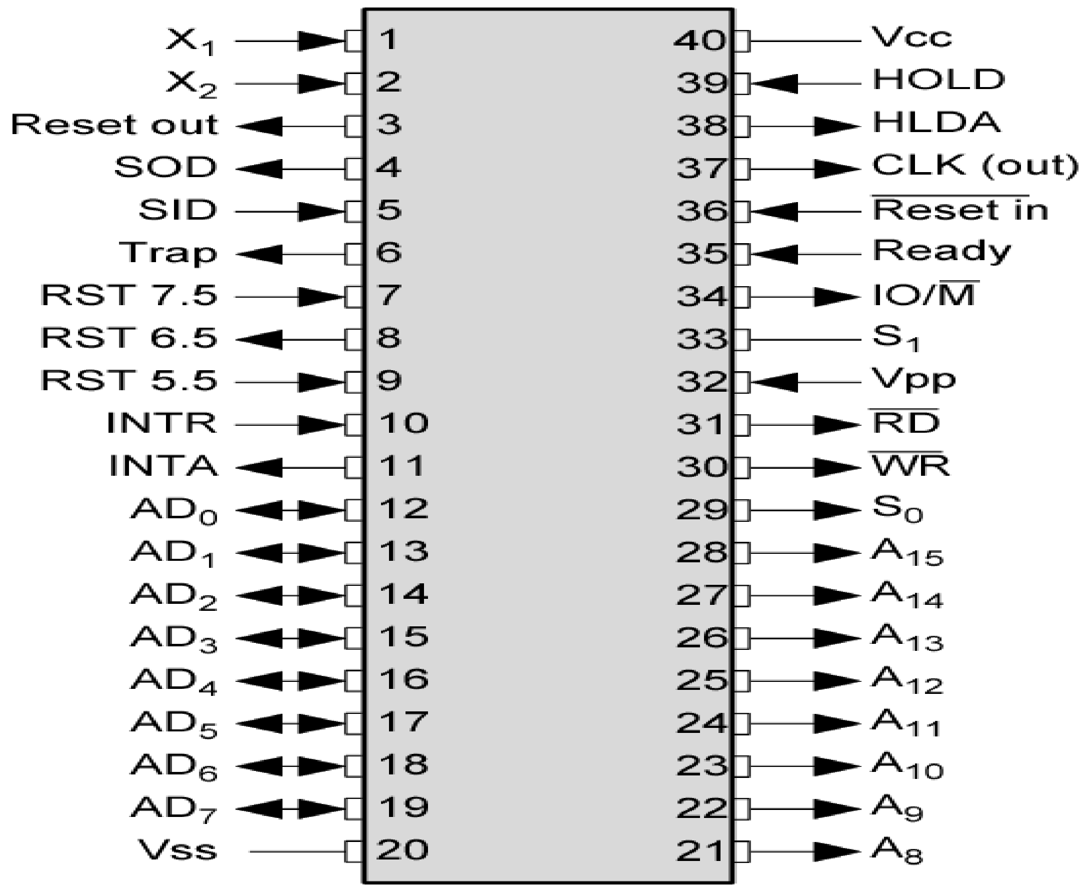
RST 7.5 ~~ Highest Priority
RST 6.5
RST 5.5 o Lowest Priority The priority of these interrupts is ordered as shown above.
These interrupts have a higher priority than the INTR.

13. TRAP (Input):-Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.
14. RESET IN (Input):-Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected The CPU is held in the reset condition as long as Reset is applied.
15. RESET OUT (Output) Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock.
16. X1, X2 (Input):- Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.
17. CLK (Output):-Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.
18. IO/M (Output):-IO/M indicates whether the Read/Write is to memory or l/O Tristated during Hold and Halt modes.
19. SID (Input):- Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
20. SOD (output):- Serial output data line. The output SOD is set or reset as specified by the SIM

instruction.

21. Vcc: - +5 volt supply.

22. Vss:-Ground Reference.



Pin Diagram Of 8085

8085 Functional Description

The 8085A is a complete 8 bit parallel central processor. It requires a single +5 volt supply. Its basic clock speed is 3 MHz thus improving on the present 8080's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU, a RAM/ IO, and a ROM or PROM/IO chip. The 8085A uses a multiplexed Data Bus. The address is split between the higher 8bit Address Bus and the lower 8bit Address/Data Bus. During the first cycle the address is sent out. The lower 8bits are latched into the peripherals by the Address Latch Enable (ALE). During the rest of the machine cycle the Data Bus is used for memory or I/O data.
The 8085A provides RD, WR, and IO/Memory signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. Hold, Ready, and all Interrupts are synchronized. The 8085A also provides serial input data (SID) and serial output data (SOD) lines for simple serial interface. In addition to these features, the 8085A has three maskable, restart interrupts and one non-maskable trap interrupt. The 8085A provides RD, WR and IO/M signals for Bus control.

Status Information:-Status information is directly available from the 8085A. ALE serves as a status strobe. The status is partially encoded, and provides the user with advanced timing of the type of bus transfer being done. IO/M cycle status signal is provided directly also. Decoded So, S1 Carries the following status information:

HALT, WRITE, READ, and FETCH: - S1 can be interpreted as R/W in all bus transfers. In the 8085A the 8 LSB of address are multiplexed with the data instead of status. The ALE line is used as a strobe to enter the lower half of the address into the memory or peripheral address latch. This also frees extra pins for expanded interrupt capability.

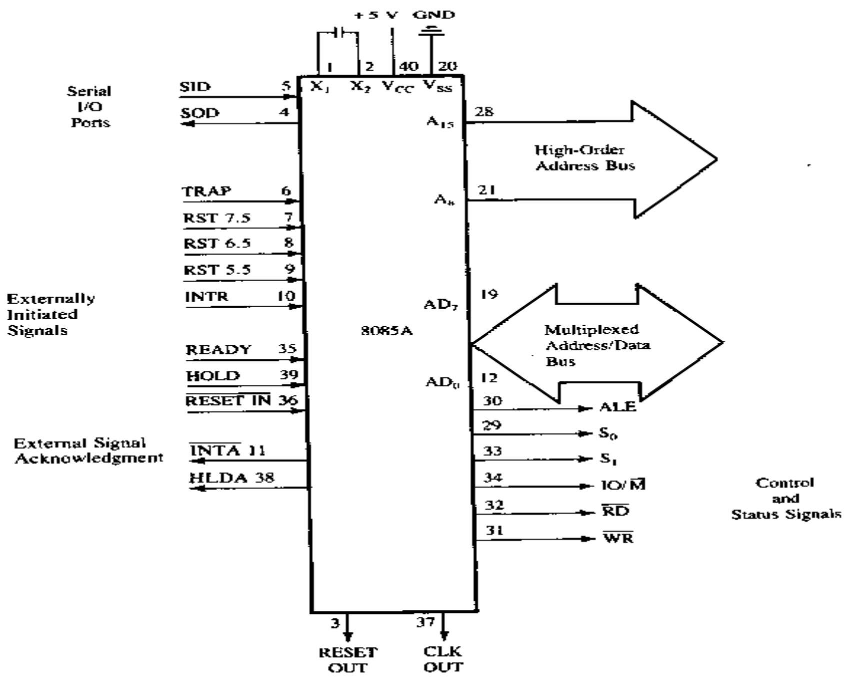
Interrupt and Serial I/O:-The8085A has5 interrupt inputs: INTR, RST5.5, RST6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080 INT. Each of the three RESTART inputs, 5.5, 6.5. 7.5, has a programmable mask. TRAP is also a RESTART interrupt except it is nonmaskable. The three RESTART interrupts cause the internal execution of RST (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RST independent of the state of the interrupt enable or masks. The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP highest priority, RST 7.5, RST 6.5, RST 5.5, INTR lowest priority This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt a RST 7.5 routine if the interrupts were re-enabled before the end of the RST 7.5 routine. The TRAP interrupt is useful for catastrophic errors such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both edge and level sensitive.

Basic System Timing:- The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8bits of address on the Data Bus. Figure 2 shows an instruction fetch, memory read and l/ O write cycle (OUT). Note that during the l/O write and read cycle that the l/O port address is copied on both the upper and lower half of the address. As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085A can be used with slow memory. Hold causes the CPU to relingkuish the bus when it is through with it by floating the Address and Data Buses.

System Interface:-8085A family includes memory components, which are directly compatible to the 8085A CPU. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

1. 2K Bytes ROM
2. 256 Bytes RAM
3. 1 Timer/Counter
4. 4 8bit l/O Ports
5. 1 6bit l/O Port
6. 4 Interrupt Levels
7. Serial In/Serial Out Ports

In addition to standard l/O, the memory mapped I/O offers an efficient l/O addressing technique. With this technique, an area of memory address space is assigned for l/O address, thereby, using the memory address for I/O manipulation. The 8085A CPU can also interface with the standard memory that does not have the multiplexed address/data bus.



UNIT II 8085 Microprocessor

Lecture 5

Addressing Modes of 8085

1. To perform any operation, we have to give the corresponding instructions to the microprocessor.
2. In each instruction, programmer has to specify 3 things:
   - Operation to be performed.
   - Address of source of data.
   - Address of destination of result
3. The method by which the address of source of data or the address of destination of result is given in the instruction is called **Addressing Modes.**

Types of Addressing Modes

Intel 8085 uses the following addressing modes:

1. Direct Addressing Mode
2. Register Addressing Mode
3. Register Indirect Addressing Mode
4. Immediate Addressing Mode
5. Implicit Addressing Mode

1.Direct Addressing Mode

 In this mode, the address of the operand is given in the instruction itself.
 LDA is the operation.
 2500 H is the address of source.
 Accumulator is the destination.

LDA 2500 H	Load the contents of memory location 2500 H in accumulator.

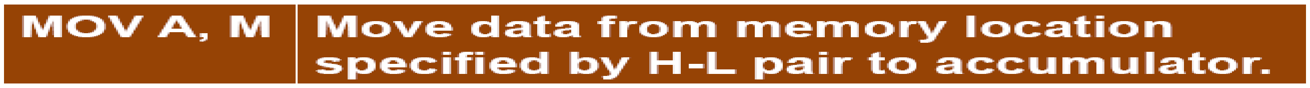
2.Register Addressing Mode

 In this mode, the operand is in general purpose register.
 MOV is the operation.
 B is the source of data.
 A is the destination.

MOV A, B	Move the contents of register B to A.

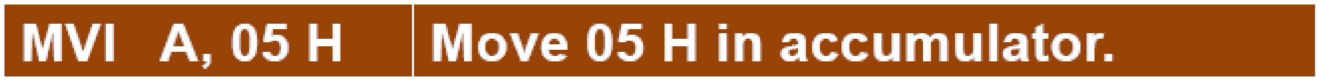
3.Register Indirect Addressing Mode

 In this mode, the address of operand is specified by a register pair.
 MOV is the operation.
 M is the memory location specified by H-L register pair.
 A is the destination



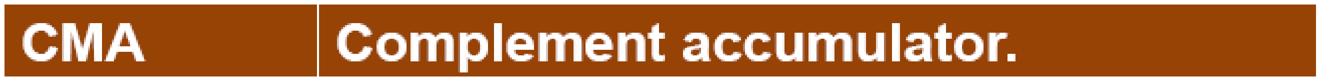
MOV A, M	Move data from memory location specified by H-L pair to accumulator.

4.Immediate Addressing Mode
 In this mode, the operand is specified within the instruction itself.
 MVI is the operation.
 05 H is the immediate data (source).
 A is the destination.



MVI A, 05 H	Move 05 H in accumulator.

5.Implicit Addressing Mode
 If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.
 CMA is the operation.
 A is the source.
 A is the destination.



CMA	Complement accumulator.

UNIT II 8085 Microprocessor

Lecture 6,7,8

Instruction Set of 8085

An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called Instruction Set.
- 8085 has 246 instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value are called Op-Code or Instruction Byte.

Classification of Instruction Set
- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instructions
- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

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.
- **Example:** MOV B, C or MOV B, M

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.

- 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 H-L registers.
- **Example:** MVI B, 57H or MVI M, 57H

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

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

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

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

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

This instruction loads 16-bit data in the register pair.
 Example: LXI H, 2034 H

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

 This instruction copies the contents of memory location pointed out by 16-bit address into register L.
 It copies the contents of next memory location into register H.
 Example: LHLD 2040 H

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

 The contents of accumulator are copied into the memory location specified by the operand.
 Example: STA 2500 H

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

 The contents of accumulator are copied into the memory location specified by the contents of the register pair.
 Example: STAX B

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

 The contents of register L are stored into memory location specified by the 16-bit address.
 The contents of register H are stored into the next memory location.
 Example: SHLD 2550 H

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

 The contents of register H are exchanged with the contents of register D.
 The contents of register L are exchanged with the contents of register E.
 Example: XCHG

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

 This instruction loads the contents of H-L pair into SP.
 Example: SPHL

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

 The contents of L register are exchanged with the location pointed out by the contents of the SP.
 The contents of H register are exchanged with the next location (SP + 1).
 Example: XTHL

Opcode	Operand	Description
XTHL	None	Exchange H–L with top of stack

 The contents of registers H and L are copied into the program counter (PC).
 The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
 Example: PCHL

Opcode	Operand	Description
PCHL	None	Load program counter with H-L contents

 The contents of register pair are copied onto stack.

 SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
 SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.
 Example: PUSH B

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

 The contents of top of stack are copied into register pair.
 The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
 SP is incremented and the contents of location are copied to the high-order register (B, D, H, A).
 Example: POP H

Opcode	Operand	Description
POP	Reg. pair	Pop stack to register pair

 The contents of accumulator are copied into the I/O port.
 Example: OUT 78 H

Opcode	Operand	Description
OUT	8-bit port address	Copy data from accumulator to a port with 8-bit address

 The contents of I/O port are copied into accumulator.
 Example: IN 8C H

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8-bit address

Arithmetic Instructions

These instructions perform the operations like:
 Addition
 Subtract
 Increment
 Decrement

Addition

 Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
 The result (sum) is stored in the accumulator.
 No two other 8-bit registers can be added directly.
 Example: The contents of register B cannot be added directly to the contents of register C.

Subtraction

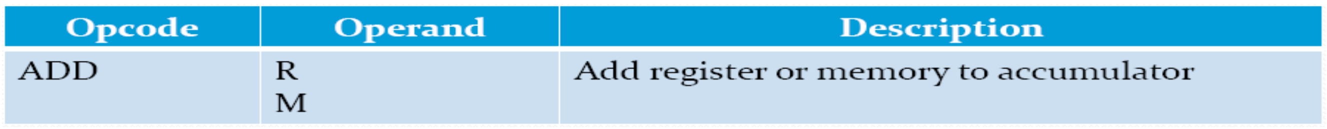
 Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
 The result is stored in the accumulator.
 Subtraction is performed in 2's complement form.
 If the result is negative, it is stored in 2's complement form.
 No two other 8-bit registers can be subtracted directly.

Increment / Decrement

 The 8-bit contents of a register or a memory location can be incremented or decremented by 1.
 The 16-bit contents of a register pair can be incremented or decremented by 1.
 Increment or decrement can be performed on any register or a memory location.

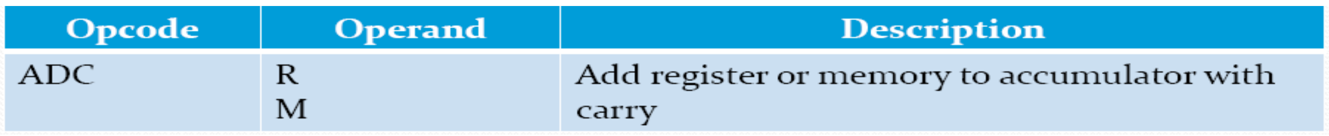
Arithmetic Instructions

 The contents of register or memory are added to the contents of accumulator.
 The result is stored in accumulator.
 If the operand is memory location, its address is specified by H-L pair.
 All flags are modified to reflect the result of the addition.
 Example: ADD B or ADD M



Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

 The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
 The result is stored in accumulator.
 If the operand is memory location, its address is specified by H-L pair.
 All flags are modified to reflect the result of the addition.
 Example: ADC B or ADC M



Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.
- No other flags are changed.
- **Example:** DAD B

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBB B or SBB M

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 45 H

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** INR B or INR M

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.
- **Example:** INX H

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** DCR B or DCR M

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

Logical Instructions

These instructions perform logical operations on data stored in registers, memory and status flags.

- The logical operations are:
- AND
- OR
- XOR
- Rotate
- Compare
- Complement

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
- AND operation
- OR operation
- XOR operation
- With the contents of accumulator.
- The result is stored in accumulator.

Rotate

- Each bit in the accumulator can be shifted either left or right to the next position.

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
- Equality
- Greater Than
- Less Than with the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.

- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

- 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.
- **Example:** CMP B or CMP M

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- The 8-bit data is compared with the contents of 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
- **Example:** CPI 89H

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- **Example:** ANA B or ANA M.

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.

- **Example:** ANI 86H.

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA B or ORA M.

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 86H.

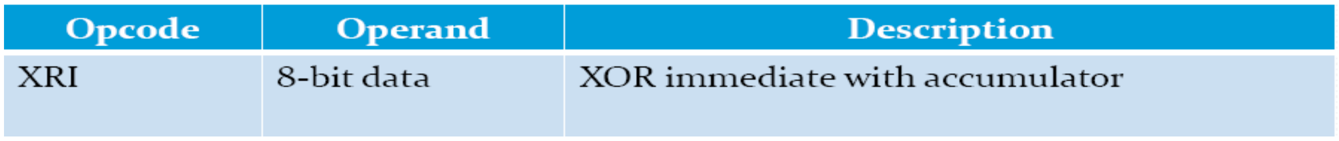
Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

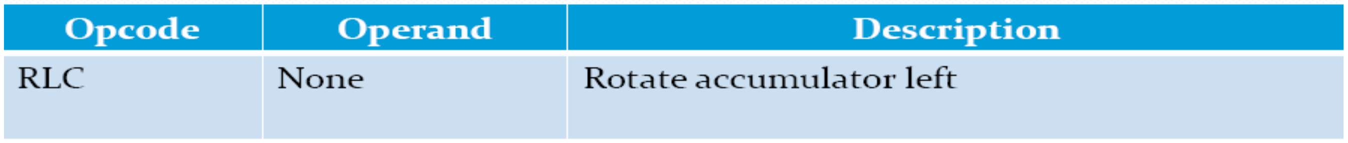
- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.

- **Example:** XRI 86H.



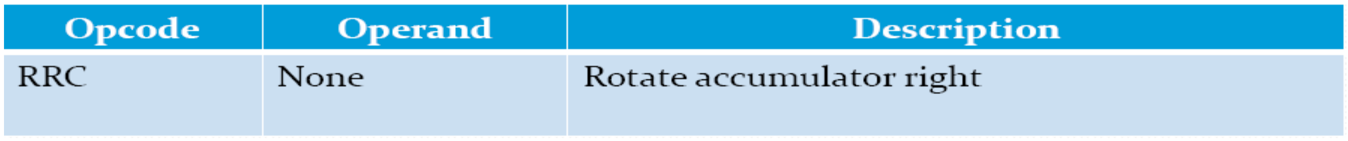
Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

- 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.
- **Example:** RLC.



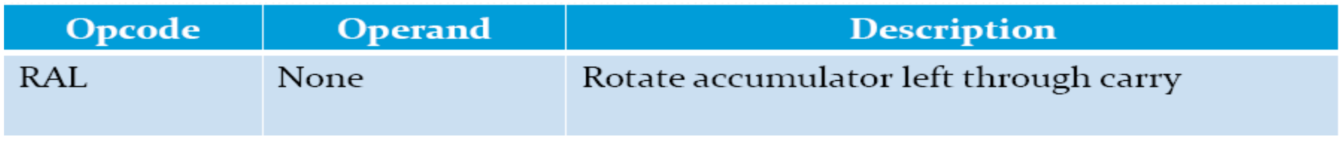
Opcode	Operand	Description
RLC	None	Rotate accumulator left

- 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.
- **Example:** RRC.



Opcode	Operand	Description
RRC	None	Rotate accumulator right

- 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.
- **Example:** RAL.



Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.

 Example: RAR.

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

 The contents of the accumulator are complemented.
 No flags are affected.
 Example: CMA.

Opcode	Operand	Description
CMA	None	Complement accumulator

 The Carry flag is complemented.
 No other flags are affected.
 Example: CMC.

Opcode	Operand	Description
CMC	None	Complement carry

The Carry flag is set to 1.
 No other flags are affected.
 Example: STC.

Opcode	Operand	Description
STC	None	Set carry

Branching Instructions

 The branching instruction alter the normal sequential flow.
 These instructions alter either unconditionally or conditionally.

 The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
 Example: JMP 2034 H.

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

 The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
 Example: JZ 2034 H.

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

 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.
 Example: CALL 2034 H.

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

 The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
 Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
 Example: CZ 2034 H.

Opcode	Operand	Description
Cx	16-bit address	Call conditionally

 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.
 Example: RET.

Opcode	Operand	Description
RET	None	Return unconditionally

 The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
 The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
 Example: RZ.

Opcode	Operand	Description
Rx	None	Call conditionally

 The RST instruction jumps the control to one of eight memory locations depending upon the number.
 These are used as software instructions in a program to transfer program execution to one of the eight locations.
 Example: RST 3.

Opcode	Operand	Description
RST	0 – 7	Restart (Software Interrupts)

Control Instructions

The control instructions control the operation of microprocessor.
 No operation is performed.
 The instruction is fetched and decoded but no operation is executed.
 Example: NOP

Opcode	Operand	Description
NOP	None	No operation

 The CPU finishes executing the current instruction and halts any further execution.
 An interrupt or reset is necessary to exit from the halt state.
 Example: HLT

Opcode	Operand	Description
HLT	None	Halt

 The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.
 No flags are affected.
 Example: DI

Opcode	Operand	Description
DI	None	Disable interrupt

 The interrupt enable flip-flop is set and all interrupts are enabled.
 No flags are affected.
 This instruction is necessary to re-enable the interrupts (except TRAP).
 Example: EI

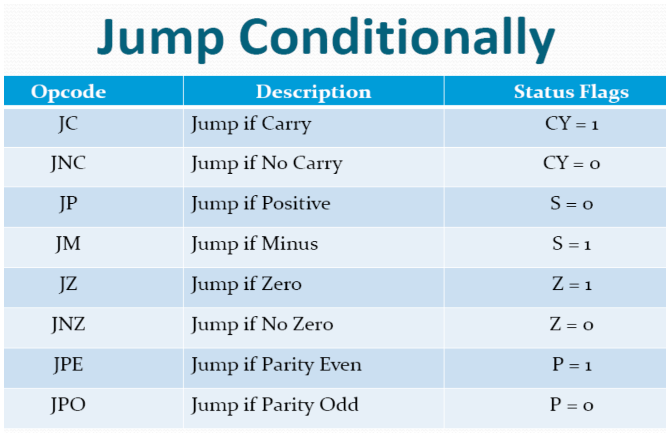
Opcode	Operand	Description
EI	None	Enable interrupt

 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.
 Example: RIM

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

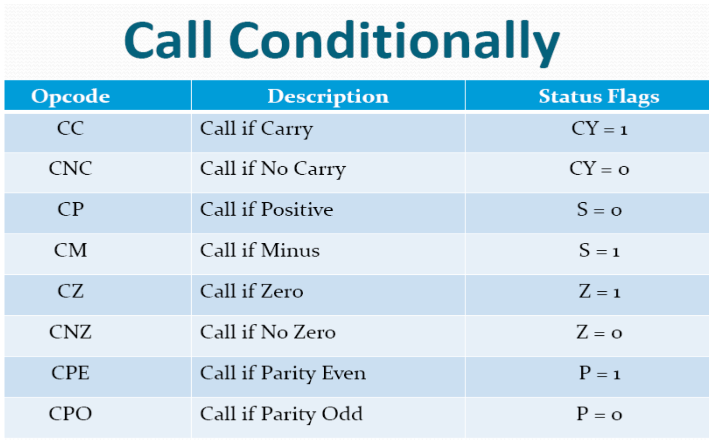
 This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
 The instruction interprets the accumulator contents as follows.
 Example: SIM

Opcode	Operand	Description
SIM	None	Set Interrupt Mask



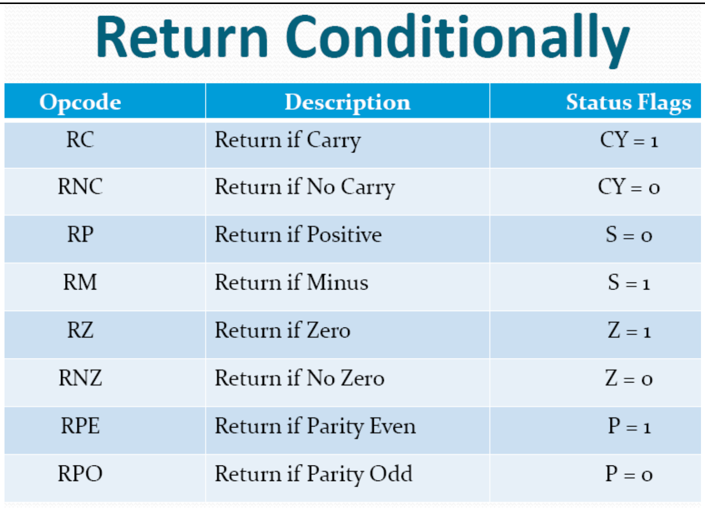
Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0



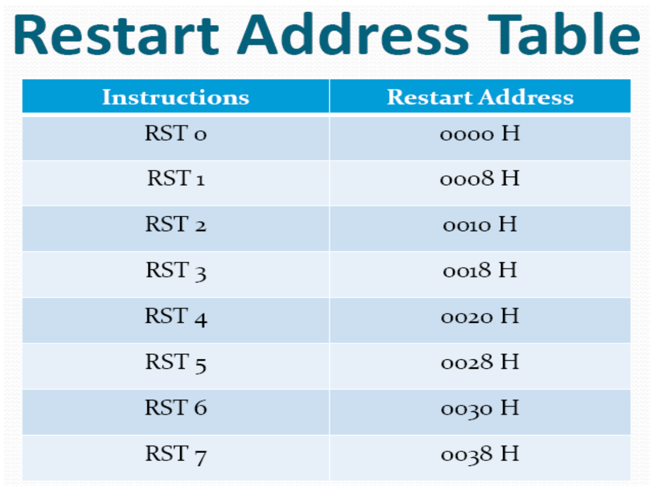
Call Conditionally

Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P = 0



Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0



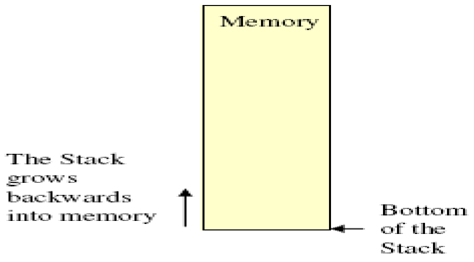
Restart Address Table

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

Lecture 9

Stack

The stack is an area of memory identified by the programmer for temporary storage of information.
The stack is a LIFO structure.
– Last In First Out.
The stack normally grows backwards into memory.
– In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.



Given that the stack grows backwards into memory, it is customary to place the bottom of the stack at the end of memory to keep it as far away from user programs as possible.

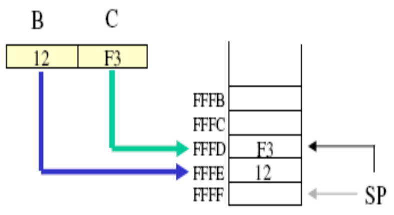
- In the 8085, the stack is defined by setting the SP (Stack Pointer) register.
- LXI SP, FFFFH
- This sets the Stack Pointer to location FFFFH (end of memory for the 8085).
- The Size of the stack is limited only by the available memory

Saving Information on the Stack Information is saved on the stack by PUSHing it on.
– It is retrieved from the stack by POPing it off.
- The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back.

– Both PUSH and POP work with register pairs ONLY.

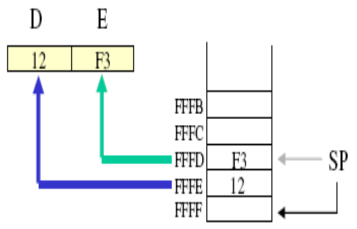
The PUSH Instruction

PUSH B (1 Byte Instruction)
– Decrement SP
– Copy the contents of register B to the memory location pointed to by SP
– Decrement SP
– Copy the contents of register C to the memory location pointed to by SP



The POP Instruction

POP D (1 Byte Instruction)
– Copy the contents of the memory location pointed to by the SP to register E
– Increment SP
– Copy the contents of the memory location pointed to by the SP to register D
– Increment SP



Operation of the Stack

During pushing, the stack operates in a "decrement then store" style.
– The stack pointer is decremented first, then the information is placed on the stack.
• During poping, the stack operates in a "use then increment" style.
– The information is retrieved from the top of the the stack and then the pointer is incremented.
• The SP pointer always points to "the top of the stack".

LIFO

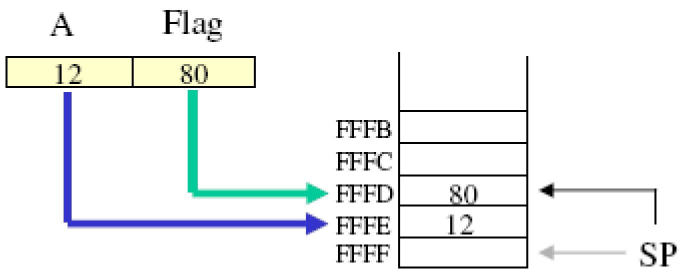
The order of PUSHs and POPs must be opposite of each other in order to retrieve information back into its original location.
PUSH B
PUSH D
...
POP D
POP B
• Reversing the order of the POP instructions will result in the exchange of the contents of BC and DE.

The PSW Register Pair

The 8085 recognizes one additional register pair called the PSW (Program Status Word).
– This register pair is made up of the Accumulator and the Flags register.
• It is possible to push the PSW onto the stack, do whatever operations are needed, then POP it off of the stack.
– The result is that the contents of the Accumulator and the status of the Flags are returned to what they were before the operations were executed.

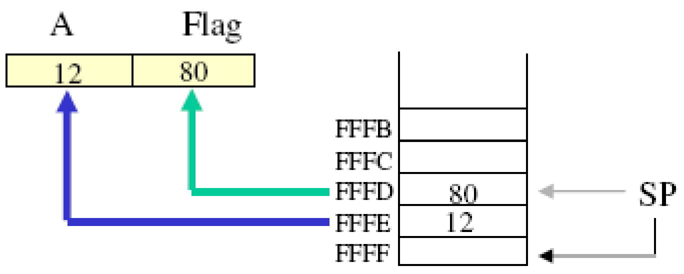
PUSH PSW Register Pair

PUSH PSW (1 Byte Instruction)
– Decrement SP
– Copy the contents of register A to the memory location pointed to by SP
– Decrement SP
– Copy the contents of Flag register to the memory location pointed to by SP



Pop PSW Register Pair

POP PSW (1 Byte Instruction)
– Copy the contents of the memory location pointed to by the SP to Flag register
– Increment SP
– Copy the contents of the memory location pointed to by the SP to register A
– Increment SP



Modify Flag Content using PUSH/POP

Let, We want to Reset the Zero Flag
8085 Flag Program:
– LXI SP FFFF
– PUSH PSW
– POP H
– MOV A L
– ANI BFH (BFH= 1011 1111) * Masking
– MOV L A
– PUSH H
– POP PSW

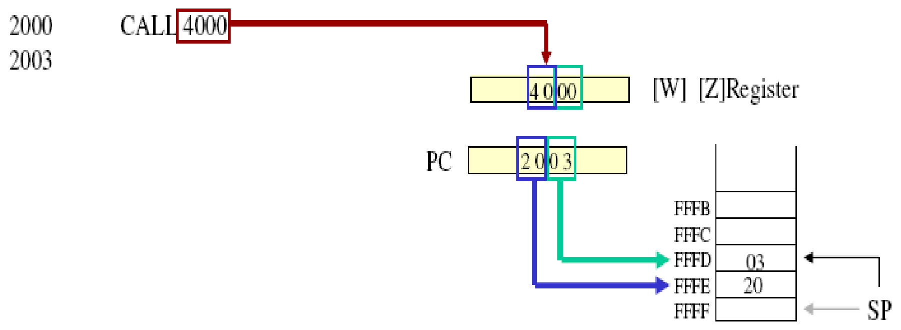
Lecture 10

Subroutines

A subroutine is a group of instructions that will be used repeatedly in different locations of the program.
– Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.
• In Assembly language, a subroutine can exist anywhere in the code.
– However, it is customary to place subroutines separately from the main program.
The 8085 has two instructions for dealing with subroutines.
– The CALL instruction is used to redirect program execution to the subroutine.
– The RET instructions is used to return the execution to the calling routine.

The CALL Instruction

CALL 4000H (3 byte instruction)
– When CALL instruction is fetched, the MP knows that the next two Memory location contains 16bit subroutine address in the memory.



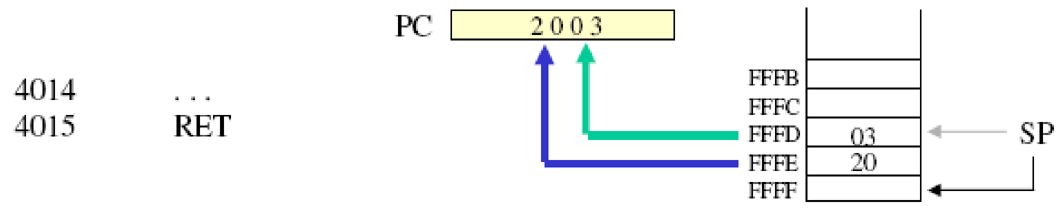
MP Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the W register and stores the lower order 8bit of the address in the Z register
– Push the address of the instruction immediately following the CALL onto the stack [Return address]
– Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register.

The RET Instruction

RET (1 byte instruction)

– Retrieve the return address from the top of the stack

– Load the program counter with the return address.



UNIT II 8085 Microprocessor

Lecture 11

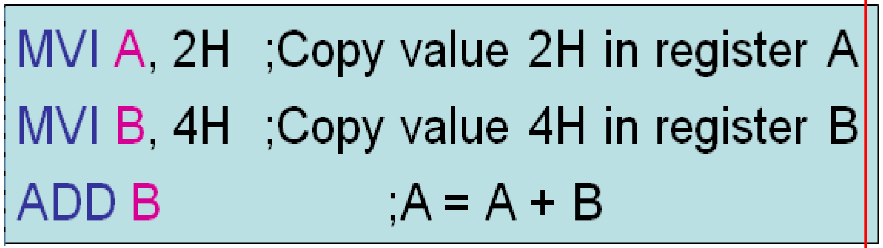
Assembly language programs of 8085 microprocessor

- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only
- This language is called **Machine Language**
- For e.g. 01001111
  - Is a valid machine language instruction of 8085
  - It copies the contents of one of the internal registers of 8085 to another

Assembly Language of 8085

- It uses English like words to convey the action/meaning called as MNEMONICS
- For e.g.

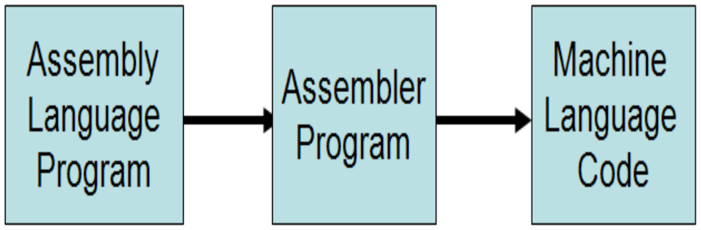
MOV	to indicate data transfer
ADD	to add two values
SUB	to subtract two values



```
MVI A, 2H   ;Copy value 2H in register A
MVI B, 4H   ;Copy value 4H in register B
ADD B              ;A = A + B
```

Microprocessor understands Machine Language only

- Microprocessor cannot understand a program written in Assembly language
- A program known as Assembler is used to convert a Assembly language program to machine language



Low-level/High-level languages

- Machine language and Assembly language are both
  - Microprocessor specific (Machine dependent) so they are called
  - Low-level languages
- Machine independent languages are called
  - High-level languages
  - For e.g. BASIC, PASCAL, C++, C, JAVA, etc.
  - A software called Compiler is required to convert a high-level language program to machine code

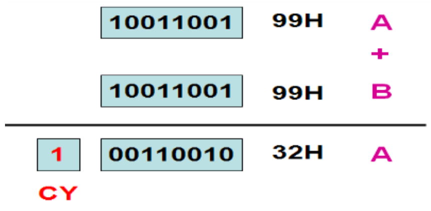
Program 8085 in Assembly language to add two 8-bit numbers. Result can be more than 8-bits.

1. Analyze the problem
   - Result of addition of two 8-bit numbers can be 9-bit
   - Example

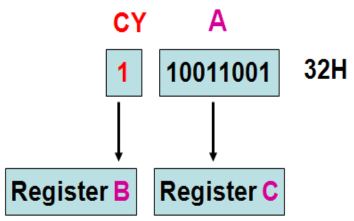
```
  10011001   (99H) A
 +10011001   (99H) B
 100110010 (132H)
```

   - The 9th bit in the result is called CARRY bit.

- How 8085 does it?
  - Adds register A and B
  - Stores 8-bit result in A
  - SETS carry flag (CY) to indicate carry bit



- Storing result in Register memory



Step-1 Copy A to C
Step-2

a) Clear register B
b) Increment B by 1

UNIT II 8085 Microprocessor

Lecture 12

Time-Delay loops

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.
•The table in Appendix F of the book contains a column with the title B/M/T.
B for Number of Bytes
M for Number of Machine Cycles
T for Number of T-State.

Knowing how many T-States an instruction requires, and keeping in mind that a T-State is one clock cycle long, we can calculate the time using the following formula:
Delay = No. of T-States / Frequency
For example a "MVI" instruction uses 7 T-States. Therefore, if the Microprocessor is running at 2 MHz, the instruction would require 3.5 µSeconds to complete.
We can use a loop to produce a certain amount of time delay in a program.
•The following is an example of a delay loop:
MVI C, FFH7 T-States
LOOPDCR C4 T-States
JNZ LOOP10 T-States

The first instruction initializes the loop counter and is executed only once requiring only 7 T-States.
The following two instructions form a loop that requires 14 T-States to execute and is repeated 255 times until C becomes 0.We need to keep in mind though that in the last iteration of the loop, the JNZ instruction will fail and require only 7 T-States rather than the 10.
•Therefore, we must deduct 3 T-States from the total delay to get an accurate delay calculation.
•To calculate the delay, we use the following formula:
Tdelay= TO+ TL
Tdelay= total delay
TO= delay outside the loop
TL= delay of the loop
•TOis the sum of all delays outside the loop.

Using these formulas, we can calculate the time delay for the previous example:
•TO= 7 T-States Delay of the MVI instruction

•TL= (14 X 255) -3 = 3567 T-States14 T-States for the 2 instructions repeated 255 times (FF16= 25510) reduced by the 3 T-States for the final JNZ.

Timing Diagram

Representation of Various Control signals generated during Execution of an Instruction. Following Buses and Control Signals must be shown in a Timing Diagram:

Higher Order Address Bus.

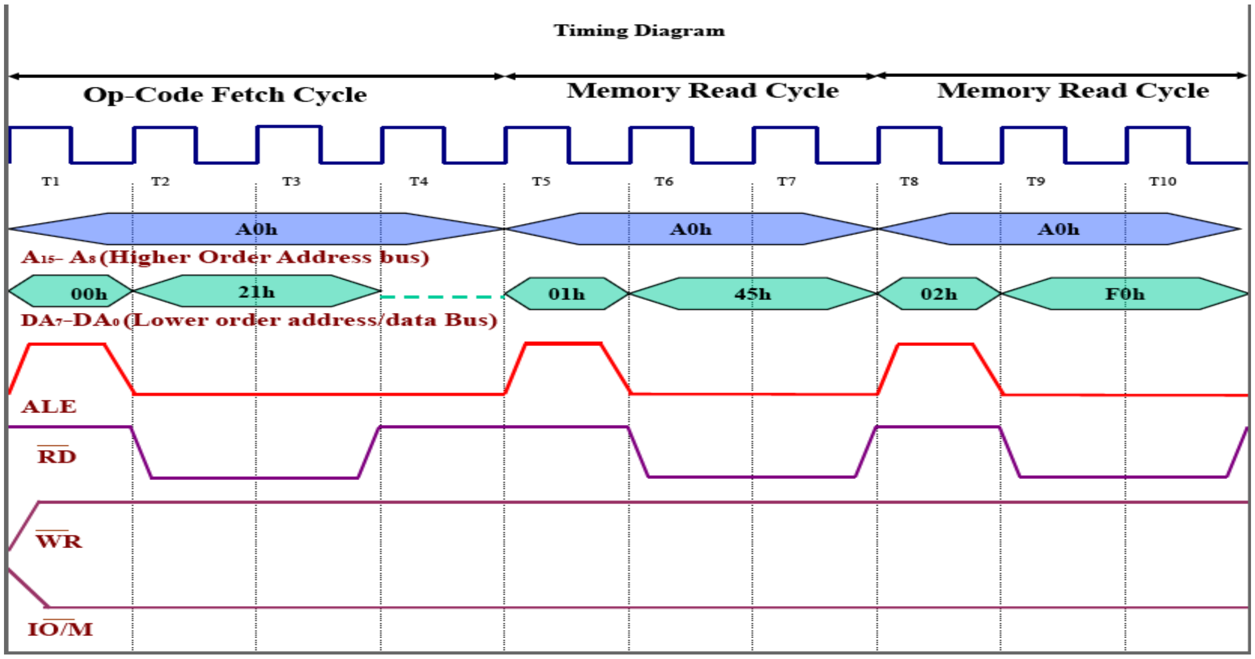
Lower Address/Data bus

ALE

RD

WR

IO/M



Instruction: A000h LXI A,FO45h

Corresponding Coding:

A000h	**21**
A001h	**45**
A002h	**F0**