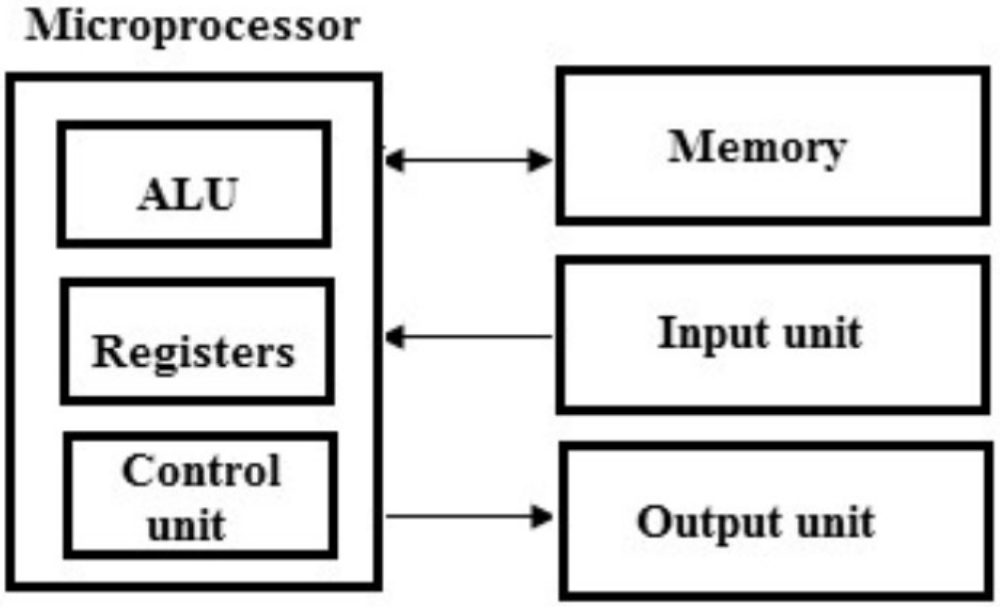
UNIT – I
MICROPROCESSOR – 8085

Microprocessor – Introduction

Microprocessor is a multipurpose programmable integrated circuit (IC) chip. It has computing and decision-making capabilities similar to the central processing unit (CPU) of a computer. Figure shows the parts of a microprocessor based system. The microprocessor works as per the program stored in memory. Data from the external world enters the microprocessor through the input unit. Data may be sent to the external world through the Output input.



Some of the important features of microprocessor are:

The microprocessor IC consists of ALU, Registers and Control unit.

1. ALU – Arithmetic and Logic Unit. ALU performs the computing and decision making operations.
2. Registers–Registers are used for storing the internal temporary data.
3. Control unit – The control unit controls the operation of the microprocessor and the devices connected to the microprocessor.
4. The microprocessor can understand a set of basic commands (instruction set).
5. The microprocessor has several pins for transmitting address signals to the memory and I/O (Input / Output) devices. These pins are known as address bus.
6. The microprocessor has several pins for transmitting data signals to the memory and I/O devices. These pins are known as data bus.
7. The microprocessor has few pins for controlling the memory and I/O devices. These pins are known as control bus.

Evolution of microprocessor

The history of the development of microprocessor is given below:

4-bit microprocessors:

- 4004 was the first microprocessor introduced in 1971 by Intel Corporation, USA.
- Operating on 4-bits of data at a time.
- Has the capabilities for addition, subtraction, comparison and logical (AND and OR) operations.
- Examples: Intel's 4004, Intel's 4040, Rockwell International's PPS4, Toshiba's T3472

8-bit microprocessors:

- 8008 was the first 8-bit microprocessor introduced in 1973 by Intel Corporation, USA.
- Perform arithmetic and logical operations on 8-bit data.
- Examples: Intel's 8008, Intel's 8080, Intel's 8085, Motorola's M6800, National Semiconductor's NSC 800, Zilog Corporation's Z80, Fairchild's F8, Hitachi's 6809.

12-bit microprocessors:

- Performs arithmetic and logical operations on 12-bit data
- Examples : Intersil's IM6100, Toshiba's T3190

16-bit microprocessors:

- Performs arithmetic and logical operations on 16-bit data
- Examples : Intel's 8086, Intel's 8088, Intel's 80286, Fairchild's 9440, Data General's mN601, Texas Instrument's TMS9900, Motorola's M68000, Zilog's Z8000

32-bit microprocessors:

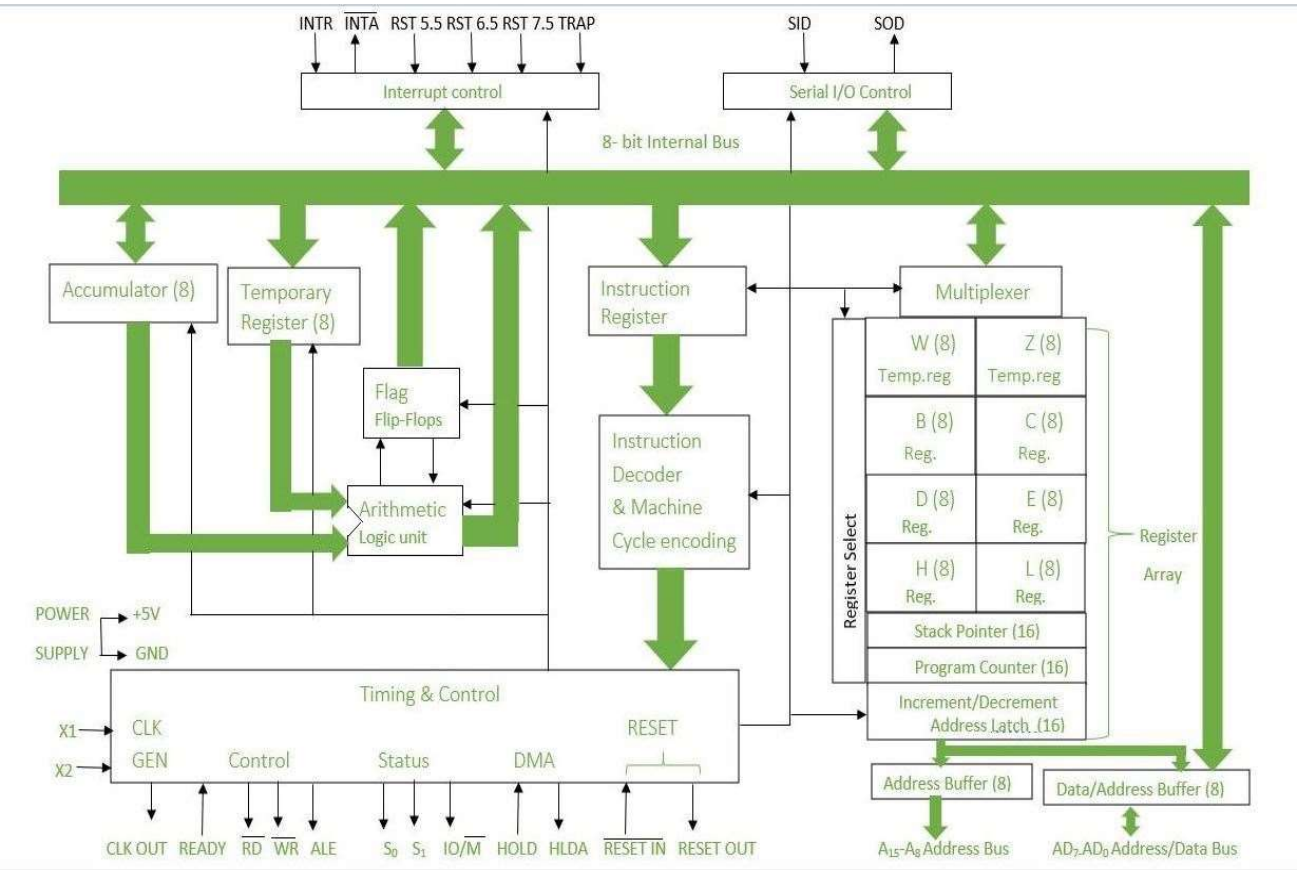
- Performs arithmetic and logical operations on 32-bit data
- Examples: Intel's 80386, Intel's 80486, Intel's iAPX432, Motorola's 68020, Motorola's 68030, National's 32032, National's 32523, Inmos' T414, Inmos' T800

64-bit microprocessors:

- Performs arithmetic and logical operations on 64-bit data
- Intel's Pentium microprocessor executes 100 million instructions per second (MIPS).
- Examples: Intel's Pentium (80586), Intel's Pentium Pro, Intel's Pentium II, Celeron, Intel's Pentium III and Intel's Pentium IV

Architecture of 8085 Microprocessor

The internal architecture (block diagram) of 8085 Microprocessor is shown in figure.



The following are the functional blocks in the 8085Microprocessor.

1. Accumulator
2. Temporary register
3. Arithmetic and Logic Unit (ALU)
4. Flag register
5. Instruction Register
6. Instruction Decoder and Machine cycle encoder
7. General purpose registers
8. Stack Pointer
9. Program Counter
10. Incrementer / Decrementer
11. Timing and Control unit
12. Interrupt control
13. Serial I/O control
14. Address buffer and Address / Data buffer

1. Accumulator (A-register)

It is an 8-bit register. It is associated with ALU. The accumulator is also called A-register. During the arithmetic / logic operations, one of the operand is available in Accumulator. The result of the arithmetic / logic operations is also stored in the Accumulator.

2. Temporary (TEMP) register

It is an 8-bit register. It is also associated with ALU. This register is used to hold one of the data (from memory or general purpose registers) during an arithmetic / logic operation.

3. Arithmetic and Logic Unit (ALU)

The Arithmetic and Logic Unit includes Accumulator, Temporary register, arithmetic and logic circuits and flag register. The ALU can perform arithmetic (such as addition and subtraction) and logic operations (such as AND, OR and EX-OR) on 8-bit data. It receives the data from accumulator and or TEMP register. The result is stored in the accumulator. The conditions of the result (such as carry, zero) are indicated in the flags.

4. Flag register

It is an 8-bit register. But only five bits are used. The flag positions in the flag register are shown in figure

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

The flags are affected by the arithmetic and logic operations in the ALU. The flag register is also known as Status register or Condition code register. There are five flags namely Sign (S) flag, Zero (Z) flag, Auxiliary Carry (AC) flag, Parity (P) flag and Carry (CY) flag.

- Sign (S) flag: Sign flag is set (1) if the bit D7 of the result in the accumulator is 1, otherwise it is reset (0). This flag is set when the result is negative. This flag is used only for signed numbers.
- Zero (Z) flag: Zero flag is set (1) if the result in the accumulator is zero, otherwise it is reset (0).
- Auxiliary Carry (AC): Auxiliary Carry flag is set (1) if there is a carry from bit position D3of result in the accumulator, otherwise it is reset (0). This flag is used for BCD operations.
- Parity (P) flag: Parity flag is set (1) if the result in the accumulator has even number of 1s, otherwise it is reset (0).

- Carry (CY) flag: Carry flag is set (1) if the result of an arithmetic operation results in a carry from bit position D7, otherwise it is reset (0). This flag is also used to indicate a borrow condition during subtraction operations.

5. Instruction register

When an instruction is fetched from memory, it is stored in the Instruction register. It is an 8-bit register. This resister cannot be used in the programs.

6. Instruction Decoder and Machine cycle encoding

This unit decodes the instruction stored in the Instruction register. It determines the nature of the instruction and establishes the sequence of events to be followed by the Timing and control unit.

7. General purpose registers

There are six 8-bit general purpose registers namely B, C, D, E, H and L registers. B and C registers are combined together as BC register pair for 16-bit operations. Similarly D and E registers can be used as DE resister pair and H and L as HL register pair. The HL register pair is also used as memory pointer (M-register) for storing 16-bit address in some instructions. There are two more 8-bit temporary registers W and Z. These registers are used to hold data during the execution of some instructions. W and Z registers cannot be used in programs.

8. Stack Pointer (SP)

Stack is a portion of memory (RAM) used as FILO (First In Last Out) buffer. This is mainly used during subroutine operations. Stack Pointer is a 16- bit register used as a memory pointer (16-bit address) for denoting the stack position in memory. The Stack pointer is decremented each time when data is loaded into the stack and incremented when data is retrieved from the stack. Stack pointer always points to the top of the stack memory.

9. Program Counter (PC)

The Program Counter (PC) is a 16-bit register. It is used to point the address of the next instruction to be fetched from the memory. When one instruction is fetched from memory, PC is automatically incremented to point out the next instruction.

10. Incrementer / Decrementer

This unit is used to increment or decrement the contents of the 16-bit registers.

11. Timing and Control unit

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between microprocessor and peripherals. The internal clock generator is available in this unit. This unit has the micro programs for all the instructions to carry out the micro steps required in completing the instructions. This unit receives signals from the Instruction decoder and Machine cycle encoding unit and generates control signals according to the micro-program for the instruction.

12. Interrupt control

There are five hardware interrupts available in 8085 Microprocessor namely TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR for interfacing the peripherals with the microprocessor. These interrupts are handled by the Interrupt control unit. $INTA$ Signal is generated by the Interrupt control unit as an acknowledgement for an interrupting device. If two or more interrupts occur at the same time, service is given according to the priority basis.

13. Serial I/O control

Serial data is transmitted to the peripherals through SOD pin and received through the SID pin. The SOD and SID pins are handled by the Serial I/O control unit using the SIM and RIM instructions.

14. Address buffer and Address / Data buffer

The Address buffer is an 8-bit unidirectional buffer from which the higher order address bits $A_8 – A_{15}$ leaves the microprocessor to the memory and peripherals. The Address /

Data buffer is an 8-bit bidirectional buffer used for sending the lower order address bits A_0 – A_7 and sending and receiving the data bits D_0 – D_7 to the memory and peripherals.

Instruction set and addressing modes

Instruction format

The format of 8085 microprocessor instructions is shown in figure.

The instruction has two parts, Opcode and Operand.

Opcode: Represents the operation to be performed on the operand. It is also called mnemonic.

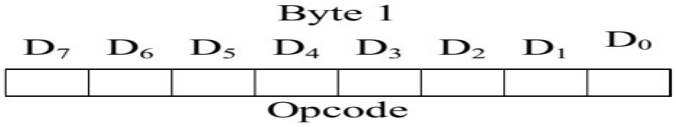
Operand: Data or address is given in this part. If the operand is an 8-bit data, only Operand-1 is present in the instruction. If the operand is a 16-bit data or address, Operand-1 and Operand-2 are specified in the instruction. Both Operand-1 and Operand-2 are optional.

Classification of instructions based on size

There are three groups of instructions in 8085 microprocessor based on the length or size of the instruction. They are,

1. Single byte (or 1 byte) instructions
2. Two byte instructions
3. Three byte instructions

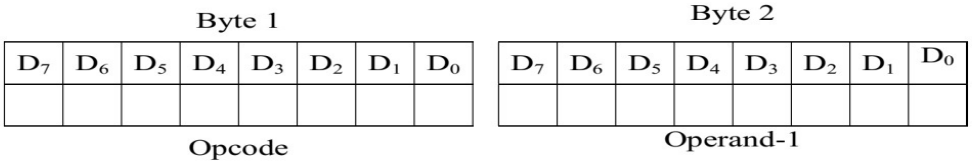
Single byte instructions



This type of instruction has only Opcode and the operand is specified within the Opcode itself.

Example: 1) MOV B, C 2) ADD B

Two byte instructions

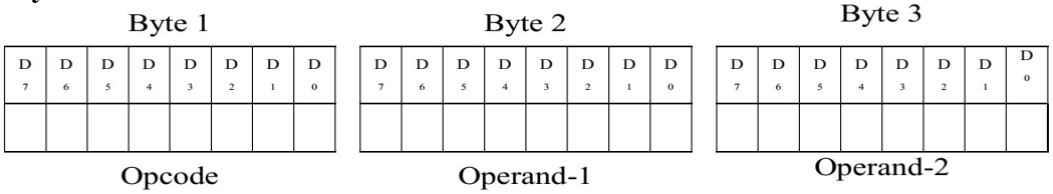


This type of instruction has Opcode and one operand. The first byte represents the Opcode and the second byte represents the 8-bit operand data or 8-bit port address.

This type of instruction has Opcode and one operand. The first byte represents the Opcode and the second byte represents the 8-bit operand data or 8-bit port address.

Example : 1) MVI A, 50H 2) OUT 50H

Three byte instructions



This type of instruction has Opcode and two operands. The first byte represents the Opcode, the second byte presents the lower order 8-bits of data or address and the third byte represents the higher order 8-bits of data or address.

Example: 1) STA 5000H 2) LXI B, 5000H

Classification of instructions based on function

There are 246 instructions (74 types) in the 8085 microprocessor. Based on the function of the instruction, the instructions are classified into the following five types.

1. Data transfer instructions
2. Arithmetic instructions
3. Logic and bit manipulation instructions
4. Branch instructions
5. Machine control instructions

Data transfer instructions

These instructions move (or copy) data from source to destination. The source and destination are registers and memory. Memory to memory transfer is not possible. After the data transfer, the content of the source is not modified and the earlier content of the destination is altered. No flags are affected.

Examples: 1) MOV A, B 2) MOV A, M

Arithmetic instructions

Arithmetic operations like addition, subtraction, increment and decrement are performed by this category of instructions. One of the operand is taken from the Accumulator and the other operand may be from registers or memory. The result of the arithmetic operations is stored in the Accumulator. All the flags are affected.

Examples: 1) ADD B 2) INR A

Logic and bit manipulation instructions

Logical functions like AND, OR and EX-OR are performed by this instructions. All logic functions are performed in relation with the contents of the Accumulator. All the flags are affected.

Examples: 1) ANA B 2) CMA

Branch instructions

Branch instructions change the sequence of the program execution unconditionally or conditionally. The condition of flags is used to take the decision for conditional branches. No flags are affected.

Examples: 1) JMP 5000H 2) JNZ 5000H

Machine control instructions

The instructions dealing with interrupt handling and system operations are classified into this category. No flags are affected.

Examples: 1) HLT 2) EI

Instruction set

Data Transfer Instructions

S.No	Instruction	Example

1.	**Move - Copy from source to destination**
	MOV Rd, Rs
	MOV M, Rs
	MOV Rd, M
		
		MOV B, C
		MOV M, A
		MOV B, M

	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered.	
2.	**Move immediate 8-bit** MVI Rd, data MVI M, data	 MVI B, 50H MVI M, 50H
	The 8-bit data is stored in the destination register or memory.	
3.	**Load accumulator direct** LDA 16-bit address	LDA 5000H
	The contents of a memory location, specified by a 16-bit address in the operand,	

	are copied to the accumulator.	
4.	**Load accumulator indirect** LDAX Reg. pair	LDAX B LDAX D
	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator.	
5.	**Load register pair immediate** LXI Reg. pair, 16-bit data	LXI B, 5000H LXI D, 5000H LXI H, 5000H LXI SP, 5000H
	The instruction loads 16-bit data in the register pair designated in the operand**.**	
6.	**Load H and L registers direct** LHLD 16-bit address	LHLD 5000H
	The instruction copies the contents of the memory location pointed by the 16-bit address into register L and copies the contents of the next memory location into register H.	
7.	**Store accumulator direct** STA 16-bit address	STA 5000H
	The contents of the accumulator are copied into the memory location specified by the operand.	
8.	**Store accumulator indirect** STAX Rx	STAX B STAX D
	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair).	
9.	**Store H and L registers direct** SHLD 16-bit address	SHLD 5000H
	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.	
10.	**Exchange H and L with D and E** XCHG	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.	
11.	**Copy H and L registers to the stack pointer** SPHL	SPHL
	Loads the contents of the H and L registers into the stack pointer register.	
12.	**Exchange H and L with top of stack pointer** XTHL	XTHL
	The contents of the L register are exchanged with the stack location pointed by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1).	
13.	**Push register pair onto stack** PUSH Reg. pair(PSW 'Processor Status Word' means Accumulator and Flag register)	PUSH B PUSH D PUSH H PUSH PSW
	The contents of the register pair designated in the operand are copied onto the stack.	
14.	**Pop off stack to register pair**	POP B

	POP Reg. pair	POP D POP H POP PSW
	The contents of the memory location pointed out by the stack pointer register are copied to registers specified.	
15.	**Output data from accumulator to a port with 8-bit address** OUT 8-bit port address	 OUT 50H
	The contents of the accumulator are copied into the I/O port specified by the operand.	
16.	**Input data to accumulator from a port with 8-bit address** IN 8-bit port address	 IN 50H
	The contents of the input port designated in the operand are read and loaded into the accumulator.	

Arithmetic Instructions

S.No	Instruction	Example
1.	**Add register or memory to accumulator** ADD R ADD M	 ADD B 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.	
2.	**Add register to accumulator with carry** ADC R ADC M	 ADC B 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.	
3.	**Add immediate to accumulator** ADI 8-bit data	 ADI 45H
	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator.	
4.	**Add immediate to accumulator with carry** ACI 8-bit data	 ACI 45H
	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.	
5.	**Add register pair to H and L registers** DAD Reg. pair	 DAD H
	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.	
6.	**Subtract register or memory from accumulator** SUB R SUB M	 SUB B 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.	
7.	**Subtract source and borrow from accumulator** SBB R	 SBB B

	SBB M	SBB M
	The contents of the operand (register or memory) and the Borrow (carry flag) are subtracted from the contents of the accumulator and the result is placed in the accumulator.	
8.	**Subtract immediate from accumulator** SUI 8-bit data	 SUI 45H
	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator.	
9.	**Subtract immediate from accumulator with borrow** SBI 8-bit data	 SBI 45H
	The 8-bit data (operand) and the Borrow (carry flag) are subtracted from the contents of the accumulator and the result is stored in the accumulator.	
10.	**Increment register or memory by 1** INR R INR M	 INR B INR M
	The content of the designated register or memory is incremented by 1 and the result is stored in the same place.	
11.	**Increment register pair by 1** INX Reg. pair	INX B INX D INX H
	The contents of the designated register pair are incremented by 1 and the result is stored in the same place.	
12.	**Decrement register or memory by 1** DCR R DCR M	 DCR B DCR M
	The content of the designated register or memory is decremented by 1 and the result is stored in the same place.	
13.	**Decrement register pair by 1** DCX Reg. pair	DCX B DCX D DCX H
	The contents of the designated register pair are decremented by 1 and the result is stored in the same place.	
14.	**Decimal adjust accumulator** DAA	 DAA
	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6to the high-order four bits.	

Logic and bit manipulation instructions

S.No	Instruction	Example
1.	**Compare register or memory with accumulator** CMP R CMP M	 CMP B 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	

2.	**Compare immediate with accumulator** CPI 8-bit data	CPI 50H

	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	

3.	**Logical AND register or memory with accumulator** ANA R ANA M	 ANA B ANA M
	The contents of the accumulator are logically ANDed (bitwise) with the contents of the operand (register or memory), and the result is placed in the accumulator.	
4.	**Logical AND immediate with accumulator** ANI 8-bit data	ANI 50H
	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator.	
5.	**Exclusive OR register or memory with accumulator** XRA R XRA M	 XRA B 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.	
6.	**Exclusive OR immediate with accumulator** XRI 8-bit data	XRI 50H
	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator.	
7.	**Logical OR register or memory with accumulator** ORA R ORA M	 ORA B 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.	
8.	**Logical OR immediate with accumulator** ORI 8-bit data	ORI 50H
	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator.	
9.	**Rotate accumulator left** RLC	RLC
	Each binary bit of the accumulator is rotated left by one position. Bit D_7 is placed in the position of D_0 as well as in the Carry flag.	

10.	**Rotate accumulator right** RRC	RRC
	Each binary bit of the accumulator is rotated right by one position. Bit D_0 is placed in the position of D_7 as well as in the Carry flag.	
11.	**Rotate accumulator left through carry** RAL	RAL
	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D_7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D_0.	
12.	**Rotate accumulator right through carry** RAR	RAR
	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D_0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D_7.	
13.	**Complement accumulator** CMA	CMA
	The contents of the accumulator are complemented.	
14.	**Complement carry** CMC	CMC
	The Carry flag is complemented.	
15.	**Set Carry** STC	STC
	The Carry flag is set to 1.	

Branch instructions

S.No	Instruction	Example
1.	**Jump unconditionally** JMP 16-bit address	JMP 5000H
	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.	
2.	**Jump conditionally J** condition 16-bit address	JC 5000H JNC 5000H JP 5000H JM 5000H JZ 5000H JNZ 5000H JPE 5000H JPO 5000H
	The program sequence is transferred to the memory location specified by the 16- bit address given in the operand based on the specified flag.	
3.	**Unconditional subroutine call**	CALL 5000H

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

	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.	
4.	**Call conditionally C** condition 16-bit address	CC 5000H CNC 5000H CP 5000H CM 5000H CZ 5000H CNZ 5000H CPE 5000H CPO 5000H

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 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.

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

5.	**Return from subroutine unconditionally** RET	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.	
6.	**Return from subroutine conditionally R** condition 16-bit address	RC 5000H RNC 5000H RP 5000H RM 5000H RZ 5000H RNZ 5000H RPE 5000H RPO 5000H

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.

Opcode	Description	Flag Status
RC	**Return on Carry**	**CY = 1**
RNC	**Return on No Carry**	**CY = 0**
RP	**Return on Positive**	**S = 0**
RM	**Return on Minus**	**S = 1**
RZ	**Return on Zero**	**Z = 1**

	RNZ	**Return on No Zero**	**Z = 0**
	RPE	**Return on Parity Even**	**P = 1**
	RPO	**Return on Parity Odd**	**P = 0**

7.	**Load Program Counter with HL Contents** PCHL	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.	
8.	**Restart** RST 0-7	RST 0 RST 1 RST 2 RST 3 RST 4 RST 5 RST 6 RST 7

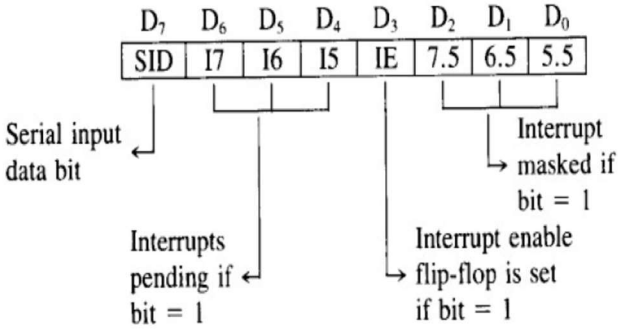
The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The addresses are:

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

Machine control instructions

S.No	Instruction	Example
1.	No operation **NOP**	NOP
	No operation is performed. The instruction is fetched and decoded. However no Operation is executed.	
2.	**Halt and enter wait state** **HLT**	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.	
3.	**Disable interrupts** **DI**	DI
	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.	
4.	**Enable interrupt** **EI**	EI
	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected.	
5.	**Read interrupt mask** **RIM**	RIM
	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5,	

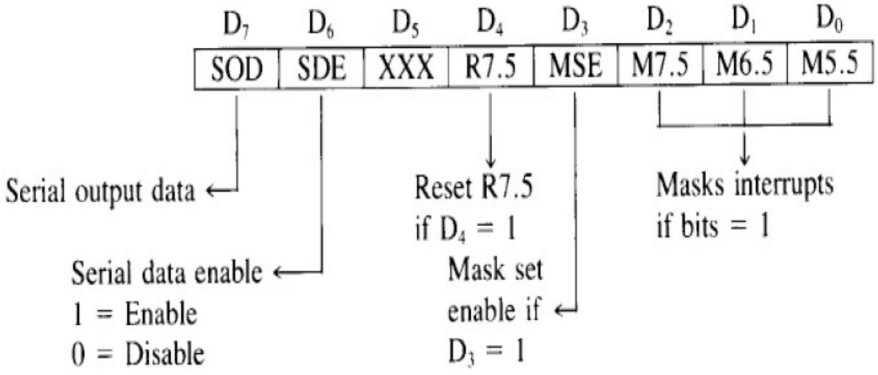
5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.



6.

Set interrupt mask
SIM | SIM

This is a multipurpose instruction and used to implement the8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.



- SOD—Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE—Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX—Don't Care
- R7.5—Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE—Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5—D_2 = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5—D_1 = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5—D_0 = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.

Addressing modes

The method of specifying the location of operand in an instruction is called addressing mode. There are five types of addressing modes in 8085 microprocessor.

1. Direct addressing mode
2. Immediate addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Implicit (or) implied addressing mode

Direct addressing mode

In direct addressing mode, the address of the operand is directly specified in the instruction. In this addressing mode, the instruction is two or three bytes long. The first byte is the Opcode. The operand may be a 16-bit (2 bytes) memory address or an 8-bit (1 byte) port address.

Examples:

S. No	Instruction	Remarks
1	STA 5000	This instruction stores the content of the accumulator in memory location 5000. Here, the memory address is given directly in the instruction.
2	LDA 5000	This instruction loads the data from memory location 5000 accumulator. Here, the memory address is given directly in the instruction.
3	IN 80	This instruction loads the data from input port 80. Here, the port address is directly given in the instruction.

Immediate addressing mode

In immediate addressing mode, the operand itself is immediately given after the Opcode. The instruction is two or three bytes long. The first byte is the Opcode. The operand may be a 16-bit (2 bytes) immediate data or an 8-bit (1 byte) immediate data.

Examples:

S. No	Instruction	Remarks
1	MVI A, 50	This instruction immediately moves the data 50 into the accumulator. Here, the data is given immediately after the Opcode.
2	LXI B, 2050	This instruction immediately moves the data 2050 into the register pair BC. 20 to B register and 50 to C register. Here, the data is given immediately after the Opcode.

Register addressing mode

In register addressing mode, a register is specified as the operand in the instruction. The instruction is one byte long. The register name is specified in the Opcode itself.

Examples:

S. No	Instruction	Remarks
1	ADD B	This instruction adds the content of B register with the accumulator. Here, the data is in the register B.
2	MOV C, D	This instruction moves the D register value to C

		register. Here, C and D registers are specified as the operands.

Register indirect addressing mode

In register indirect addressing mode, the content of the register pair is used as the address of the operand in the instruction. The instruction is one byte long. The register pair contains the 16-bit address of the memory location where the actual operand is stored.

Examples:

S. No	Instruction	Remarks
1	STAX B	This instruction stores the accumulator value in memory location whose address is specified by the BC register pair. Here, the address is indirectly specified in the register pair.
2	MOV A, M	This instruction moves the data from memory to accumulator. M means memory whose address is specified in HL register pair. Here, address of the operand is indirectly specified in the HL register pair.

Implicit or Implied addressing mode

In implied addressing mode, a particular register is implicitly specified as the operand in the instruction. The instruction is one byte long. This addressing mode is also known as implied addressing mode and inherent addressing mode.

Examples:

S. No	Instruction	Remarks
1	CMA	This instruction complements the contents of the accumulator. Here, Accumulator is implicitly specified in the instruction.
2	RLC	This instruction rotates the contents of the accumulator left one time. Here, Accumulator is implicitly specified in the instruction.

Machine cycle and Instruction cycle

Machine cycle

Machine cycle is defined as the time required for completing one operation of accessing memory, I/O or acknowledging an external request. Machine cycle is comprised of T-states. T-state is defined as one subdivision of the operation performed in one clock period. The following are the various machine cycles of 8085 microprocessor.

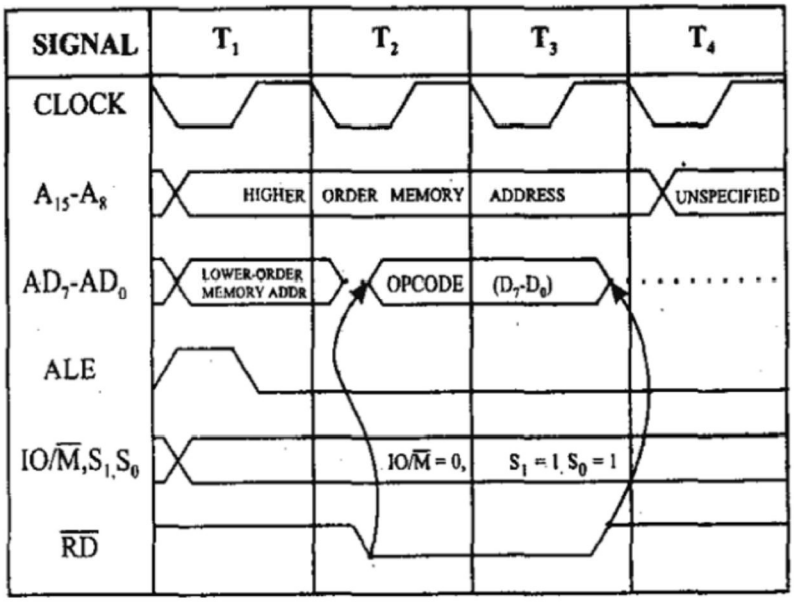
1. Opcode Fetch (OF)
2. Memory Read (MR)
3. Memory Write (MW)
4. I/O Read (IOR)
5. I/O Write (IOW)
6. Interrupt Acknowledge (IA)
7. Bus Idle (BI)

All instructions have at least one Opcode Fetch machine cycle. Depending on the type of instruction one or more other machine cycles are required to complete the execution of the instruction. The number and type of machine cycles for different instructions are shown in table.

S.No	Instruction	Number of machine cycles	Machine cycle – 1	Machine cycle - 2	Machine cycle - 3	Machine cycle - 4
1	MOV A, B	1	OF	-	-	-
2	MVIA,50H	2	OF	MR	-	-
3	LDA 5000H	4	OF	MR	MR	MR
4	STA 5000H	4	OF	MR	MR	MW
5	IN 80H	3	OF	MR	IOR	-
6	OUT 80H	3	OF	MR	IOW	-

Opcode Fetch (OF) machine cycle of 8085

Each instruction of the microprocessor has one byte Opcode. The Opcode is stored in memory. So, the processor executes the Opcode Fetch machine cycle to fetch the Opcode from memory. Hence, every instruction starts with Opcode Fetch machine cycle. The time taken by the microprocessor to execute the Opcode Fetch cycle is 4T (T- states). The first 3 T-states are used for fetching the Opcode from memory and the remaining T-state is used for internal operations by the microprocessor. The timing diagram for Opcode Fetch machine cycle is shown in figure



The steps in Opcode Fetch machine cycle are given in table.

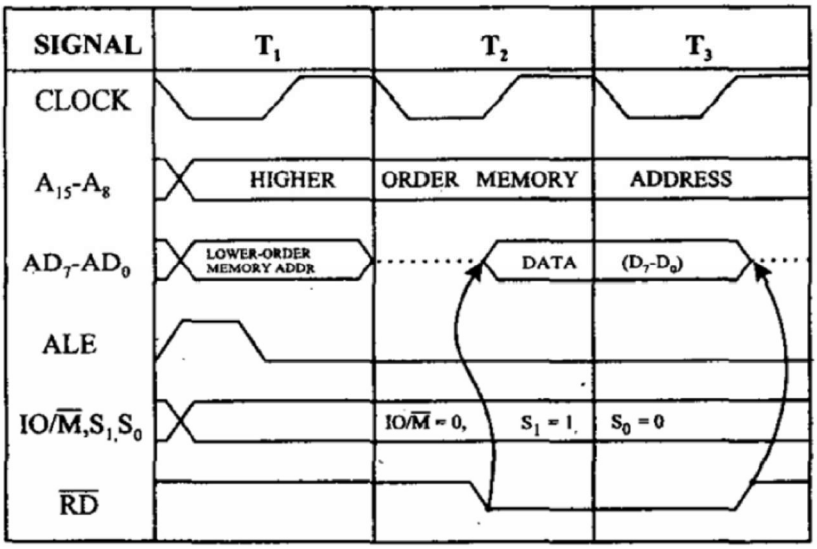
S. No	T state	Operation
1	T_1	The microprocessor places the higher order 8-bits of the memory address on A15 – A8 address bus and the lower order 8-bits of the memory address on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =1 and S0 = 1. These status signals do not change throughout the OF machine cycle.
4	T_2	The microprocessor makes the $\overline{RD}$ line LOW to enable memory read and increments the Program Counter.
5		The contents on D7 – D0 (i.e. the Opcode) are placed on

		the address / data bus.
6	T_3	The microprocessor transfers the Opcode on the address / data bus to Instruction Register (IR).
7		The microprocessor makes the $\overline{RD}$ line HIGH to disable memory read.
8	T_4	The microprocessor decodes the instruction.

Memory Read Machine Cycle of 8085

Single byte instructions require only Opcode Fetch machine cycles. But, 2-byte and 3-byte instructions require additional machine cycles to read the operands from memory. The additional machine cycle is called Memory Read machine cycle. For example, the instruction MVI A, 50H requires one OF machine cycle to fetch the operand from memory and one MR machine cycle to read the operand (50H) from memory. The MR machine cycle takes 3 T-states.

The timing diagram for Memory Read machine cycle is shown in figure

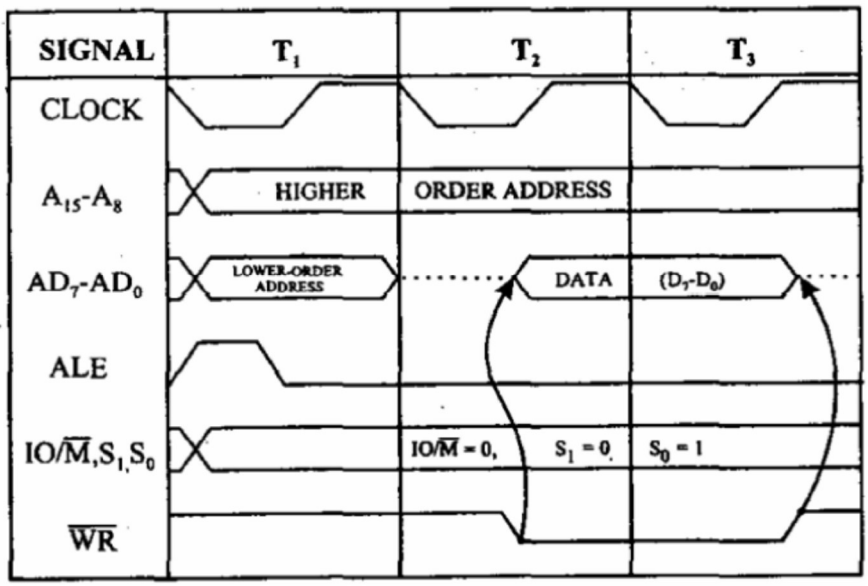


Timing Diagram for Memory Read Machine Cycle The steps in Memory Read machine cycle are given in table.

S. No	T state	Operation
1	T_1	The microprocessor places the higher order 8-bits of the memory address on A15 – A8 address bus and the lower order 8-bits of the memory address on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =1 and S0 = 0. These status signals do not change throughout the memory read machine cycle.
4	T_2	The microprocessor makes the $\overline{RD}$ line LOW to enable memory read and increments the Program Counter.
5		The contents on D7 – D0 (i.e. the data) are placed on the address / data bus.
6	T_3	The data loaded on the address / data bus is moved to the microprocessor.
7		The microprocessor makes the $\overline{RD}$line HIGH to disable the memory read operation.

Memory Write Machine Cycle of 8085

Microprocessor uses the Memory Write machine cycle for sending the data in one of the registers to memory. For example, the instruction STA 5000H writes the data in accumulator to the memory location 5000H. The MW machine cycle takes 3 T-states.
The timing diagram for Memory Write machine cycle is shown in figure

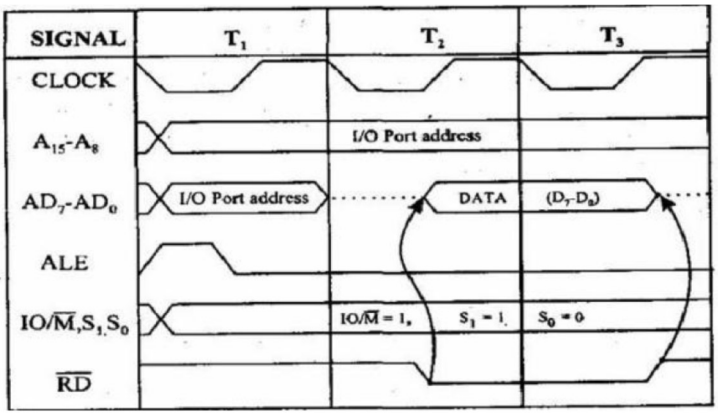


The steps in Memory Write machine cycle are given in table.

S. No	T state	Operation
1	T_1	The microprocessor places the higher order 8-bits of the memory address on A15 – A8 address bus and the lower order 8-bits of the memory address on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =0 and S0 = 1. These status signals do not change throughout the memory write machine cycle.
4	T_2	The microprocessor makes the $\overline{WR}$ line LOW to enable memory write.
5		The contents of the specified register are placed on the address / data bus.
6	T_3	The data placed on the address / data bus is transferred to the specified memory location.
7		The microprocessor makes the $\overline{WR}$ line HIGH to disable the memory write operation.

I/O Read Machine Cycle of 8085

Microprocessor uses the I/O Read machine cycle for receiving a data byte from the I/O port or from the peripheral in I/O mapped I/O systems. The IN instruction uses this machine cycle during execution. The IOR machine cycle takes 3 T-states.
The timing diagram for I/O Read machine cycle is shown in figure



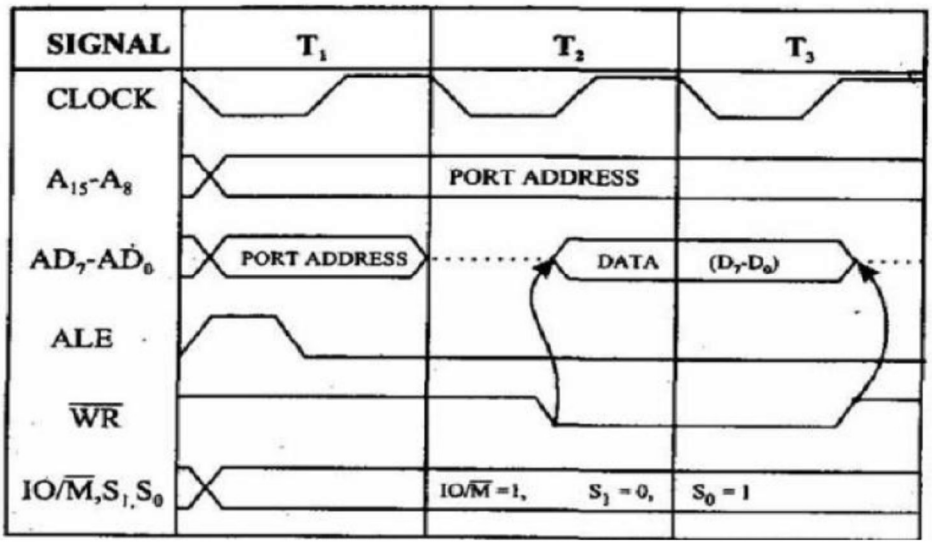
The steps in I/O Read machine cycle are given in table.

S. No	T state	Operation
1	T_1	The microprocessor places the address of the I/O port specified in the instruction on A15 – A8 address bus and also on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =1 and S0 = 0. These status signals do not change throughout the I/O read machine cycle.
4	T_2	The microprocessor makes the $\overline{RD}$ line LOW to enable I/O read.
5		The contents on D7 – D0 (i.e. the data) are placed on the address / data bus.
6	T_3	The data loaded on the address / data bus is moved to the microprocessor ie., to the accumulator.
7		The microprocessor makes the $\overline{RD}$ line HIGH to disable the I/O read operation.

I/O Write Machine Cycle of 8085

Microprocessor uses the I/O Write machine cycle for sending a data byte to the I/O port or to the peripheral in I/O mapped I/O systems. The OUT instruction uses this machine cycle during execution. The IOR machine cycle takes 3 T-states.

The timing diagram for I/O Write machine cycle is shown in figure



The steps in I/O Read machine cycle are given in table.

S. No	T state	Operation
1	T_1	The microprocessor places the address of the I/O port specified in the instruction on A15 – A8 address bus and also on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =0 and S0 = 1. These status signals do not change throughout the I/O write machine cycle.
4	T_2	The microprocessor makes the $\overline{WR}$ line LOW to enable I/O write.
5		The contents of the Accumulator are placed on the address / data bus.
6	T_3	The data placed on the address / data bus is transferred to the specified I/O port.
7		The microprocessor makes the $\overline{WR}$ line HIGH to disable the I/O write operation.

Instruction cycle

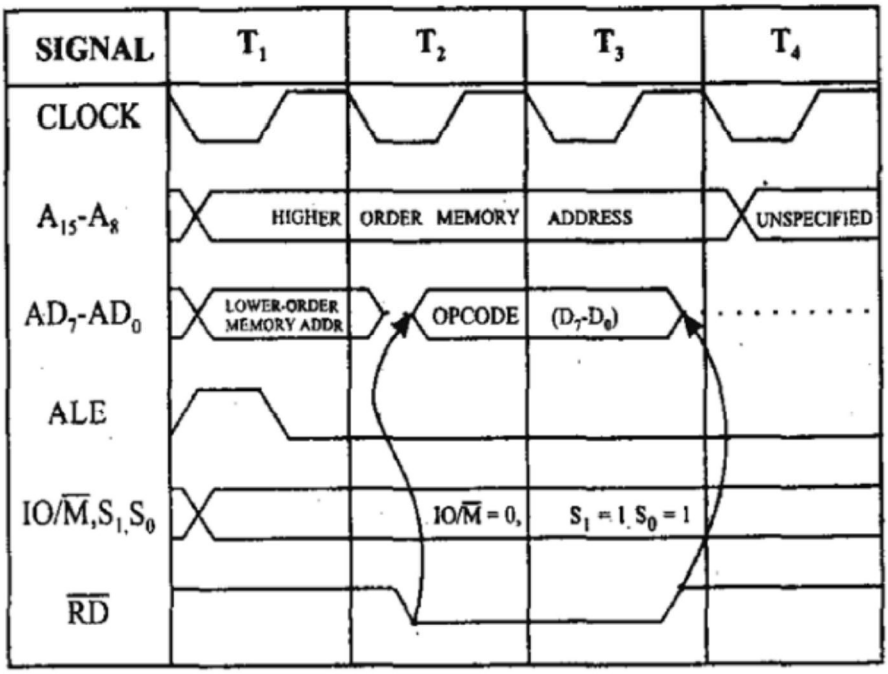
Timing diagram for MOV Rd, Rs (or MOV r_1, r_2) instruction

MOV Rd, Rs instruction moves (copies) the contents of the source register (Rs) into the destination register (Rd). It is a single byte instruction. It has only Opcode Fetch machine cycle.

Some examples for MOV Rd, Rs instruction:

1. MOV A, B
2. MOV C, L

The time taken by the processor to execute the Opcode Fetch cycle is 4T (T states). The first 3 T-states are used for fetching the Opcode from memory and the remaining T-state is used for internal operations by the microprocessor. The timing diagram for MOV Rd, Rs (Opcode Fetch machine cycle) is shown in figure. It has 4 T states.



The steps in I/O Read machine cycle are given in table.

S. No	T state	Operation
1	T_1	The microprocessor places the higher order 8- bits of the Program Counter on A15 – A8 address bus and the lower order 8-bits of the Program Counter on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/$\overline{M}$ = 0, S1 =1 and S0 = 1. These status signals do not change throughout the OF machine cycle.
4	T_2	The microprocessor makes the $\overline{RD}$ line LOW to enable memory read (opcode fetch) and increments the Program Counter.
5		The contents on D7 – D0 (i.e. the Opcode) are placed on the address / data bus.
6	T_3	The microprocessor transfers the Opcode on the address / data bus to Instruction Register (IR).
7		The microprocessor decodes the instruction.
8	T_4	The data in the register Rs (r_2) is moved to the register Rd (r_1).

I/O Mapping and Interrupts

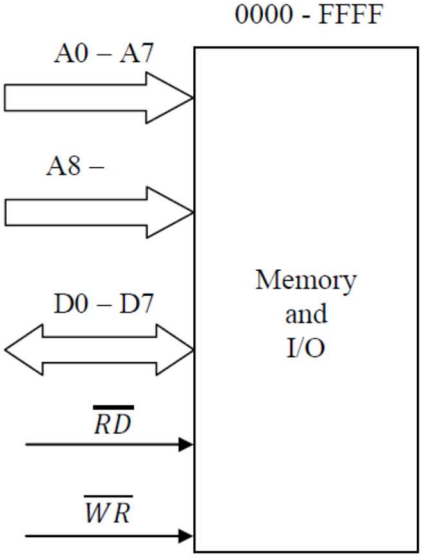
I/O interfacing

There are two methods of interfacing the Input / Output devices with the microprocessor. They are,

1) Memory mapped I/O
2) I/O mapped I/O.

Memory mapped I/O

In this method the I/O devices are treated like the memory. A part of the memory address space is used for the I/O devices. The memory mapped I/O scheme is shown in figure



In memory mapped I/O scheme, the same address space is used for both memory and I/O devices.

- The microprocessor uses the sixteen address line A0 – A7 and A8 – A15 for the memory as well as for the I/O devices.
- The I/O devices share the address space with the memory. All the memory related instructions are used for addressing I/O devices also.
- No separate IN and OUT instructions are required in memory mapped I/O scheme.
- IO/$\overline{M}$ pin is not required.

Steps for memory operations (memory read and memory write):

1. When the memory related instructions like LDA and STA are used, the microprocessor places the 16-bit address on the address bus.
2. $\overline{RD}$is activated for read operation and $\overline{WR}$ is activated for write operation.

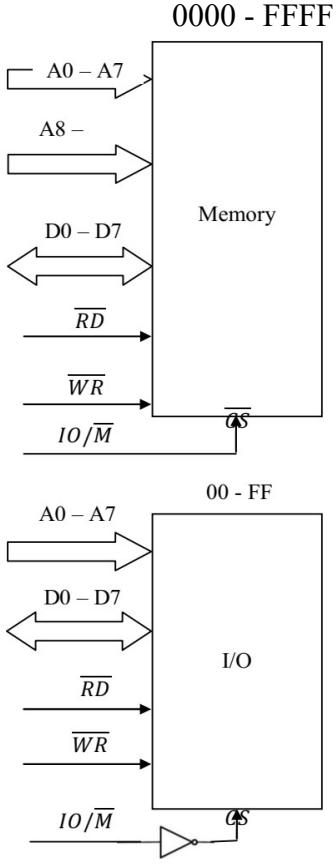
Steps for I/O operations (I/O read and I/O write):

The same steps used for memory operations are used for I/O operations also.

I/O mapped I/O

In this method, I/O devices are treated as I/O devices and memory is treated as memory. Separate address space is used for memory and I/O. The I/O mapped I/O scheme is shown in figure.

- In I/O mapped I/O scheme, the microprocessor uses the sixteen address lines A_0 – A_7 and A_8 – A_{15} for the memory and eight address lines A0 to A7to identify an input / output device.
- Here, the full address space 0000 – FFFF is used for the memory and a separate address space 00 – FF is used for the I/O devices.
- Hence, the microprocessor can address 65536 (2^{16}) memory locations 256 (2^8) input devices and 256 (2^8) output devices separately.
- IN and OUT instructions are used to activate the IO/$\overline{M}$ signal.
- When IO/$\overline{M}$ is low, the memory is selected for reading and writing operations.
- When IO/$\overline{M}$ is high, the I/O port is selected for reading and writing operations.



Steps for memory operations (memory read and memory write):

1. When the memory related instructions like LDA and STA are used, the microprocessor places the 16-bit address on the address bus.
2. The microprocessor makes the IO/$\overline{M}$ line low.
3. The microprocessor makes the $\overline{RD}$ low for read operation and $\overline{WR}$ low for write operation.

Steps for I/O operations (I/O read and I/O write):

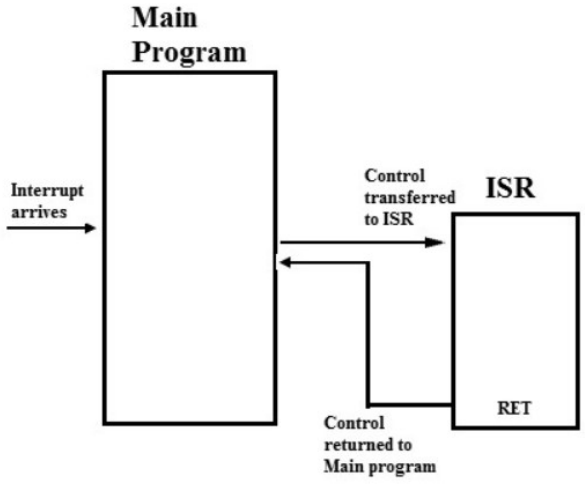
1. When the I/O related instructions like IN and OUT are used, the microprocessor places the 8-bit address on the address bus $A_0 - A_7$ as well as $A_8 - A_{15}$.
2. IO/M line is made high.
3. The microprocessor makes the $\overline{RD}$ low for read operation and $\overline{WR}$ low for write operation.

Differences between Memory mapped I/O ad I/O mapped I/O

S.No	Memory mapped I/O	I/O mapped I/O
1.	16-bit device address.	8-bit device address.
2.	Data is transferred between any general-purpose register and I/O port.	Data is transferred only between accumulator and I/O port.
3.	The memory map (64K) is shared between I/O device and system memory.	The I/O map is independent of the memory map; 256 input devices and 256 output devices can be connected.
4.	More hardware is required to decode 16- bit address.	Less hardware is required to decode 8-bit address.
5.	Arithmetic or logic operation can be directly performed with I/O data.	Arithmetic or logical operation cannot be directly performed with I/O data.
6.	IO/M pin is not required.	IO/M pin is required.
7.	Instructions like LDA, STA, MOV R,M and ADD M are used.	IN and OUT instructions are used.

Interrupts

Interrupts are the signals send by an external device to the microprocessor to request the microprocessor to perform a particular task or work. Interrupts are used for data transfer between the peripheral and the microprocessor. The microprocessor will check the interrupts always at the 2nd T-state of last machine cycle. If there is any interrupt, it accepts the interrupt and sends the $INTA$ signal to the peripheral. The microprocessor executes an interrupt service routine (ISR) stored in memory. It returns to the main program by RET instruction, after the ISR is executed. The interrupt process is shown in figure



Types of interrupts

There are six types of interrupts. They are,

1. Hardware interrupts
2. Software interrupts
3. Maskable interrupts
4. Non-maskable interrupts
5. Vectored interrupts
6. Non-vectored interrupts

Hardware interrupts: These interrupts are given by the peripheral devices to the interrupt pin (hardware) of the microprocessor. Hardware interrupts are also called external interrupts.

Software interrupts: These interrupts are internally generated within the microprocessor using software instructions. Software interrupts are also called internal interrupts.

Maskable interrupts: These external interrupts can be delayed or rejected by the microprocessor.

Non-maskable interrupts: These external interrupts cannot be delayed or rejected by the microprocessor. Non-maskable interrupts are used for handling emergency situations.

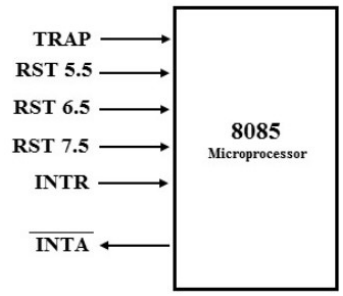
Vectored interrupts: When the address of the Interrupt Service Routine (ISR) is fixed within the microprocessor itself, then the interrupt is called Vectored interrupt.

Non-vectored interrupts: When the address of the Interrupt Service Routine (ISR) is supplied by the peripheral device, then the interrupt is called Non-vectored interrupt.

8085 interrupts

In 8085 microprocessors, there are 5 interrupts as shown in figure

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



In additional to these hardware interrupts, 8085 microprocessor has eight software interrupts. The RESTART instructions RST 0 to RST 7 are software interrupt instructions.

Interrupt priority

The microprocessor can respond to only one interrupt at one time. When multiple (more than one) interrupts occur simultaneously, the microprocessor will service the interrupts in their fixed priority order. Interrupt having the highest priority level will be serviced first. In 8085, TRAP interrupt has the highest priority and INTR has the lowest priority.

TRAP

- This interrupt is a non-maskable interrupt. It is unaffected by any mask or interrupt enable.
- It is a vectored interrupt. The interrupt vector address is 0024H.
- TRAP has the highest priority level.

- TRAP interrupt is edge and level triggered. This means that the TRAP must go high and remain high until it is acknowledged.
- In emergency situations like sudden power failure, it executes an ISR and sends the data from main memory to backup memory.

RST 7.5

- The RST 7.5 interrupt is a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 003CH.
- It has the second highest priority.
- It is edge triggered. ie. Input goes to high and no need to maintain high state until it is recognized and acknowledged.

RST 6.5

- The RST 6.5 interrupt is a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 0034H.
- It has the third highest priority.
- It is level triggered. ie. Input goes to high and stays high until it is recognized and acknowledged.

RST 5.5

- The RST 5.5 interrupt is a maskable interrupt.
- It is a vectored interrupt. The interrupt vector address is 002CH.
- It has the fourth highest priority.
- It is level triggered. ie. Input goes to high and stays high until it is recognized and acknowledged.

INTR

- INTR is a maskable interrupt.
- It is a non- vectored interrupt. After receiving $\overline{INTA}$, the peripheral has to supply the address of ISR.
- It has the lowest priority.
- It is a level triggered. ie. Input goes to high and it is necessary to maintain
- high state until it is recognized and acknowledged.

Process of INTR interrupt

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If INTR is high, the microprocessor completes current instruction, disables the interrupt and sends $INTA$ signal to the peripheral device.
4. $INTA$ allows the peripheral device to send an RST instruction through data bus.
5. Upon receiving the $INTA$ signal, the microprocessor saves the memory location of the next instruction on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction.
6. Microprocessor executes the ISR.
7. ISR must include the 'EI' instruction to enable the further interrupt within the program.
8. The RET instruction at the end of the ISR retrieves the return address from the stack and the program is transferred back to main program which was interrupted.

Instructions for Interrupts handling in 8085 microprocessors

There are four instructions available for interrupts handling. They are,

1. DI (Disable Interrupt)
2. EI (Enable Interrupt)
3. SIM (Set Interrupt Mask)
4. RIM (Read Interrupt Mask)

DI (Disable Interrupt)

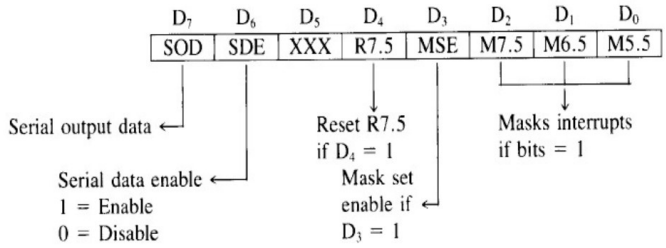
This instruction resets the Interrupt Enable Flip-flop inside the microprocessor. All the interrupts except the TRAP are disabled.

EI (Enable Interrupt)

This instruction sets the Interrupt Enable Flip-flop inside the microprocessor. All the interrupts are enabled.

SIM (Set Interrupt Mask)

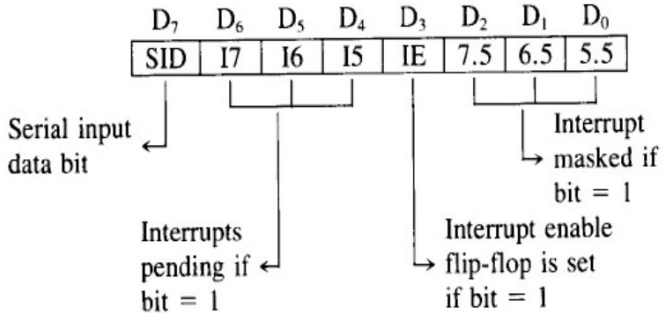
This instruction is used to selectively mask (disable) and unmask (enable) RST 7.5, RST 6.5 and RST 5.5 interrupts. This instruction is also used for serial data output. The SIM the instruction uses the accumulator contents for masking and unmasking the interrupts.



- SOD — Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE — Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX — Don't Care
- R7.5 — Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE — Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5 — D_2 = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5 — D_1 = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5 — D_0 = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.

RIM (Read Interrupt Mask)

This instruction is used to read the status of RST 7.5, RST 6.5 and RST 5.5 interrupts like pending and enable / disable details. This instruction is also used for reading the serial data. When the RIM instruction is given, the microprocessor loads the details into the accumulator.



Bits D6, D5 and D4 give the pending details of RST 7.5, RST 6.5 and RST 5.5 interrupts respectively. Bits D2, D1 and D0 give the masked / unmasked details of RST 7.5, RST 6.5 and RST 5.5 interrupts respectively. Bit D3 gives the status of Interrupt enable Flip-flop.

Summary of 8085 interrupts

Interrupt	Vector address	Priority	Type
TRAP	0024_H	1 (Highest priority)	Hardware interrupt Vectored interrupt Non-maskable interrupt
RST7.5	$003C_H$	2	Hardware interrupt Vectored interrupt Maskable interrupt
RST6.5	0034_H	3	Hardware interrupt Vectored interrupt Maskable interrupt
RST5.5	$002C_H$	4	Hardware interrupt Vectored interrupt Maskable interrupt
INTR	--	5 (Lowest priority)	Hardware interrupt Non-vectored interrupt Maskable interrupt
RST instruction	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	--	Software interrupt Vectored interrupt Maskable interrupt