MODULE 1

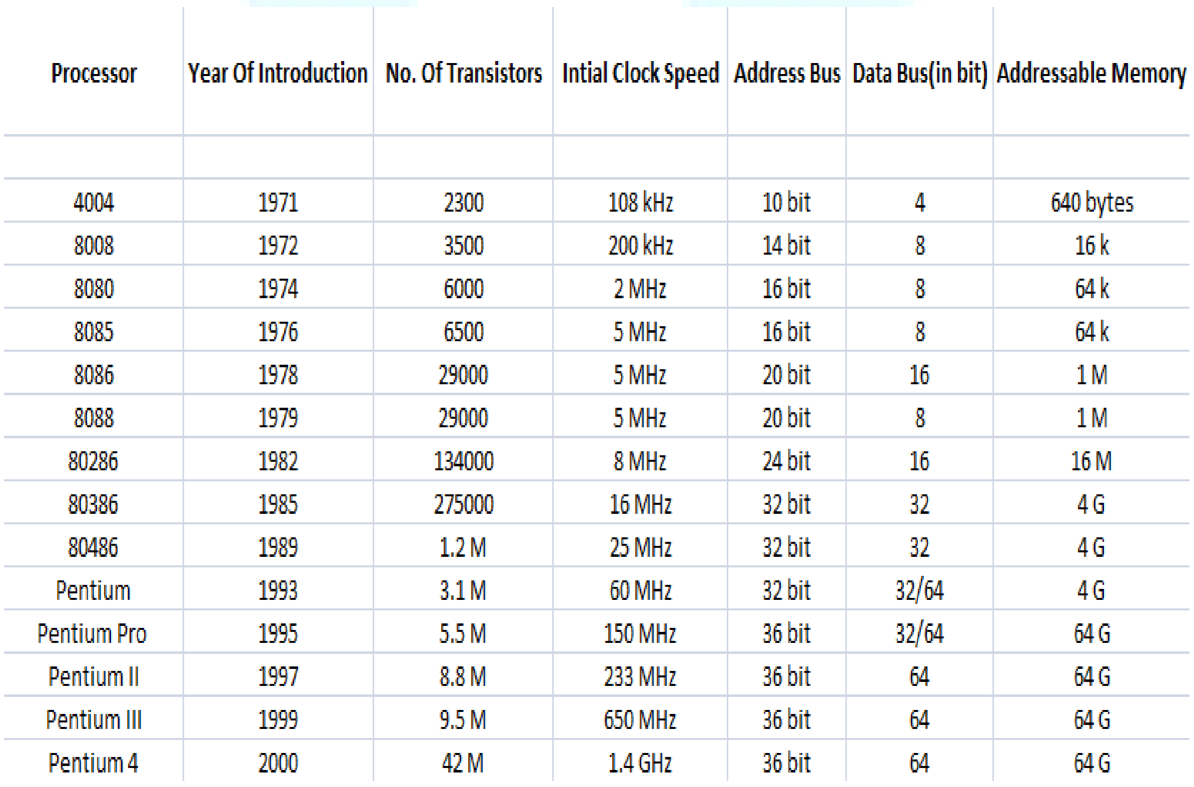
Processor-8085-Architecture-instruction set, fetch & execute, addressing mode, interrupts

MICROPROCESSOR

- A microprocessor is any of a type of miniature electronic device that contains the arithmetic, logic, and control circuitry necessary to perform the functions of a digital computer's central processing unit

- The microprocessor, also known as the Central Processing Unit (CPU), is the brain of all computers and many household and electronic devices.
- It is actually a silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably.
- The microprocessor is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output.

EVOLUTION OF MICROPROCESSOR



Processor	Year Of Introduction	No. Of Transistors	Intial Clock Speed	Address Bus	Data Bus(in bit)	Addressable Memory
4004	1971	2300	108 kHz	10 bit	4	640 bytes
8008	1972	3500	200 kHz	14 bit	8	16 k
8080	1974	6000	2 MHz	16 bit	8	64 k
8085	1976	6500	5 MHz	16 bit	8	64 k
8086	1978	29000	5 MHz	20 bit	16	1 M
8088	1979	29000	5 MHz	20 bit	8	1 M
80286	1982	134000	8 MHz	24 bit	16	16 M
80386	1985	275000	16 MHz	32 bit	32	4 G
80486	1989	1.2 M	25 MHz	32 bit	32	4 G
Pentium	1993	3.1 M	60 MHz	32 bit	32/64	4 G
Pentium Pro	1995	5.5 M	150 MHz	36 bit	32/64	64 G
Pentium II	1997	8.8 M	233 MHz	36 bit	64	64 G
Pentium III	1999	9.5 M	650 MHz	36 bit	64	64 G
Pentium 4	2000	42 M	1.4 GHz	36 bit	64	64 G

GENERATION OF MICROPROCESSOR

FIRST GENERATION

-This was the period during 1971 to 1973 of microprocessor's history.

-In 1971, INTEL created the first microprocessor 4004 that would run at a clock speed of 108 KHz.

-With only 4 bits as the word size, the 4004 could only represent signed numbers in the range -8 to +7, which is indeed very small.

-So, it was not really of practical use for arithmetic calculations. However, it found applications in controlling devices.

SECOND GENERATION

-Intel 8008 was the next in the evolution, the first 8-bit microprocessor. This was in the year 1972.

- This was soon followed by Intel 8080, also an 8-bit microprocessor.

-Intel 8080 was the first commercially popular 8bit microprocessor.

-With 8 bits as the word size, it could represent signed numbers in the range of −128 to +127.

-This is also not a good enough range for performing arithmetic calculations. Thus, the 8080 also was used only for control applications.

-Some other microprocessors like 6800 from Motorola, Z-80 from Zilog were also popular at this time.

-They were costly as they were based on NMOS technology fabrication and also for their superfast speed.

THIRD GENERATION

-Around 1978, Intel released 8086, the first 16-bit microprocessor.

- With 16-bit word size, it was possible to represent signed numbers in the range of −32,768 to +32,767, which is quite a decent range for performing arithmetic calculations.

-As such, this processor became very popular not only for control applications, but also for number crunching operations.

- Speeds of those processors were four times better than the 2nd generation processors.

- Not to be outdone, Motorola came out with 68000, their 16-bit processor.

-Zilog released Z-8000, again a 16-bit processor. These are the most popular 16-bit processors.

FOURTH GENERATION

-In the early 80s, Intel released the 32-bit processor, the Intel 80386, by using HCMOS fabrication.

-With 32-bit word size, it was possible to represent signed numbers in the range ±2×109, which is quite a large range for performing arithmetic calculations.

- If floating point notation is used, it can represent much larger numbers.

- As such, this processor became verypopular as the CPU in computers for number crunching operations.

-At this time, Motorola came out with 68020, their 32-bit processor.

- Intel released 80486, which was basically an 80386 processor and 80387 numeric co-processor on a single chip.

-Motorola released 68030. In the early 90s, Intel released 80586 by the name Pentium processor.

-It is extremely fast in performing arithmetic calculations and executing instructions.

- The Pentium 4 released in 2000 has 42 Million transistors worked with a clock frequency of 1.5 GHz and is rated for 1500 MIPS (Million instructions per second).

FIFTH GENERATION

-From 1995 to until now this generation has been bringing out high-performance and high-speed processors that make use of 64-bit processors.

-The present-day computers based on microprocessors are already faster than the mini computers and sometimes the main frame computers of yesteryear, and they are available at a small fraction of the cost of such main frame computers.

GENERAL FEATURES OF MICROPROCESSOR

- **Low Cost** - Due to integrated circuit technology microprocessors are available at very low cost. It will reduce the cost of a computer system.
- **High Speed** - Due to the technology involved in it, the microprocessor can work at very high speed. It can execute millions of instructions per second.

- **Small Size** - A microprocessor is fabricated in a very less footprint due to very large scale and ultra large scale integration technology. Because of this, the size of the computer system is reduced.

- Versatile - The same chip can be used for several applications, therefore, microprocessors are versatile.
- Low Power Consumption - Microprocessors are using metal oxide semiconductor technology, which consumes less power.
- Less Heat Generation - Microprocessors uses semiconductor technology which will not emit much heat as compared to vacuum tube devices.
- Reliable - Since microprocessors use semiconductor technology, therefore, the failure rate is very less. Hence it is very reliable.
- Portable - Due to the small size and low power consumption microprocessors are portable.

COMMON TERMS USED IN A MICROPROCESSOR

- Bus

-A bus is a set of conductors intended to transmit data, address or control information to different elements in a microprocessor.

- Usually a microprocessor will have 3 types of buses: Data Bus, Control Bus and Address Bus.

-An 8-bit processor will be using 8-bit wide bus.

- Instruction Set

-Instruction set is the group of commands that a microprocessor can understand. So instruction set is an interface between hardware and software (program).

-An instruction commands the processor to switch relevant transistors for doing some processing in data.

- For e.g.: ADD A, B; is used to add two numbers stored in the register A and B.

- Word Length

-Word Length is the number of bits in the internal data bus of a processor or it is the number of bits a processor can process at a time.

-For e.g. An 8-bit processor will have an 8-bit data bus, 8-bit registers and will do 8-bit processing at a time.

-For doing higher bits (32-bit, 16-bit) operations, it will split that into a series of 8-bit operations.

- Cache Memory

-Cache memory is a random access memory that is integrated into the processor. So the processor can access data in the cache memory more quickly than from a regular RAM.

-It is also known as CPU Memory.

-Cache memory is used to store data or instructions that are frequently referenced by the software or program during the operation. So it will increase the overall speed of the operation.

- Clock Speed

-Microprocessors uses a clock signal to control the rate at which instructions are executed, synchronize other internal components and to control the data transfer between them.

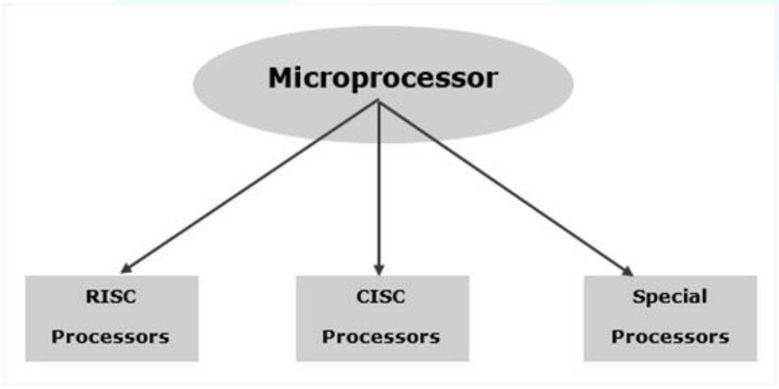
- So clock speed refers to the speed at which a microprocessor executes instructions.

-It is usually measured in Hertz and is expressed in megahertz (MHz), gigahertz (GHz) etc.

CLASSIFICATION OF MICROPROCESSORS

Based on Word Length

Based on the word length of a processor we can have 8-bit, 16-bit, 32-bit and 64-bit processors.



RISC – Reduced Instruction Set Computer

RISC is a type of microprocessor architecture which uses small, general purpose and highly optimized instruction set rather than more specialized set of instructions found in others. RISC offers high performance over its opposing architecture CISC (see below). In a processor, execution of each instruction requires a special circuit to load and process the data. So by reducing instructions, the processor will be using simple circuits and faster in operation.

- Simple instruction set
- Larger program

- Consists of large number of registers
- Simple processor circuitry (small number of transistors)
- More RAM usage
- Fixed length instructions
- Simple addressing modes
- Usually fixed number of clock cycles for executing one instruction

CISC – Complex Instruction Set Computer

CISC is the opposing microprocessor architecture for RISC. It is made to reduce the number of instructions per program, ignoring the number of cycles per instruction. So complex instructions are directly made into hardware making the processor complex and slower in operation.

This architecture is actually designed to reduce the cost of memory by reducing the program length.

- Complex instruction set
- Smaller program
- Less number of registers
- Complex processor circuitry (more number of transistors)
- Little RAM usage
- Variable length instructions
- Variety of addressing modes
- Variable number of clock cycles for each instruction

Special Purpose Processors

There are some processors which are designed to handle some specific functions.

- DSP – Digital Signal Processors
- Coprocessors – processors used along with a main processor (8087 math-coprocessor used with 8086)
- Input/output processors
- Transputer – Transistor Computer: Microprocessor with its own local memory

Examples

- Intel 4004 – The First Microprocessor
- Intel 8085
- Intel 8086
- Intel Pentium 4
- Intel Core i7

- AMD Athlon

8085 MICROPROCESSOR

- 8085 is an 8-bit microprocessor as it operates on 8 bits at a time designed by Intel in 1977 using NMOS technology.
- This microprocessor exhibits some unique characteristics and this is the reason it still holds popularity among the microprocessors.
- Basically, 8085 was the first commercially successful microprocessor by Intel.
- As some of the architectural drawbacks associated with 8080 was also eliminated by 8085.
- 8085 is pronounced as "eighty-eighty-five" microprocessor. It has the following configuration −
- 8-bit data bus
- 16-bit address bus, which can address up to 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock
- It is used in washing machines, microwave ovens, mobile phones, etc.

FEATURES OF 8085

1. It is an 8-bit microprocessor i.e. it can accept, process, or provide 8-bit data simultaneously.
2. It operates on a single +5V power supply connected at Vcc; power supply ground is connected to Vss.
3. It operates on clock cycle with 50% duty cycle.
4. It has on chip clock generator. This internal clock generator requires tuned circuit like LC, RC or crystal. The internal clock generator divides oscillator frequency by 2 and generates clock signal, which can be used for synchronizing external devices.
5. It can operate with a 3 MHz clock frequency. The 8085A-2 version can operate at the maximum frequency of 5 MHz
6. It has 16 address lines, hence it can access (216) 64 Kbytes of memory.
7. It provides 8 bit I/O addresses to access (2^8) 256 I/O ports.
8. In 8085, the lower 8-bit address bus (A0 – A7) and data bus (D0 – D7) are Multiplexed to reduce number of external pins. But due to this, external hardware (latch) is required to separate address lines and data lines.
9. It supports 74 instructions with the following addressing modes:

- **Immediate**
- **Register**
- **Direct**
- **Indirect**
- **Implied**

10. The Arithmetic Logic Unit (ALU) of 8085 performs:

- **8 bit binary addition with or without carry**
- **16 bit binary addition**
- **2 digit BCD addition.**
- **8-bit binary subtraction with or without borrow**
- **8-bit logical AND, OR, EX-OR, complement (NOT), and bit shift operations.**

11. It has 8-bit accumulator, flag register, instruction register, six 8-bit general purpose registers (B, C, D, E, H and L) and two 16-bit registers. (SP and PC). Getting the operand from the general purpose registers is faster than from memory. Hence skilled programmers always prefer general purpose registers to store program variables than memory.

12. It provides five hardware interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

13. It has serial I/O control which allows serial communication.

14. It provides control signals (IO/M, RD, WR) to control the bus cycles, and hence external bus controller is not required.

15. The external hardware (another microprocessor or equivalent master) can detect which machine cycle microprocessor is executing using status signals (IO/M, S_0, S_1). This feature of 8085 Microprocessor is very useful when more than one processor is using common system resources (memory and I/O devices).

16. It has a mechanism by which it is possible to increase its interrupt handling capacity.

17. The 8085 has an ability to share system bus with Direct Memory Access controller. This Features of 8085 Microprocessor allows to transfer large amount of data from I/O device to memory or from memory to I/O device with high speeds.

18. Features of 8085 Microprocessor can be used to implement three chip microcomputer with supporting I/O devices like IC 8155 and IC 8355.

ARCHITECTURE OF 8085

8085 is an 8-bit, general-purpose microprocessor. It consists of the following functional units:

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops −

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)
- Its bit position is shown in the following table –

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits −

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Serial Input/output control

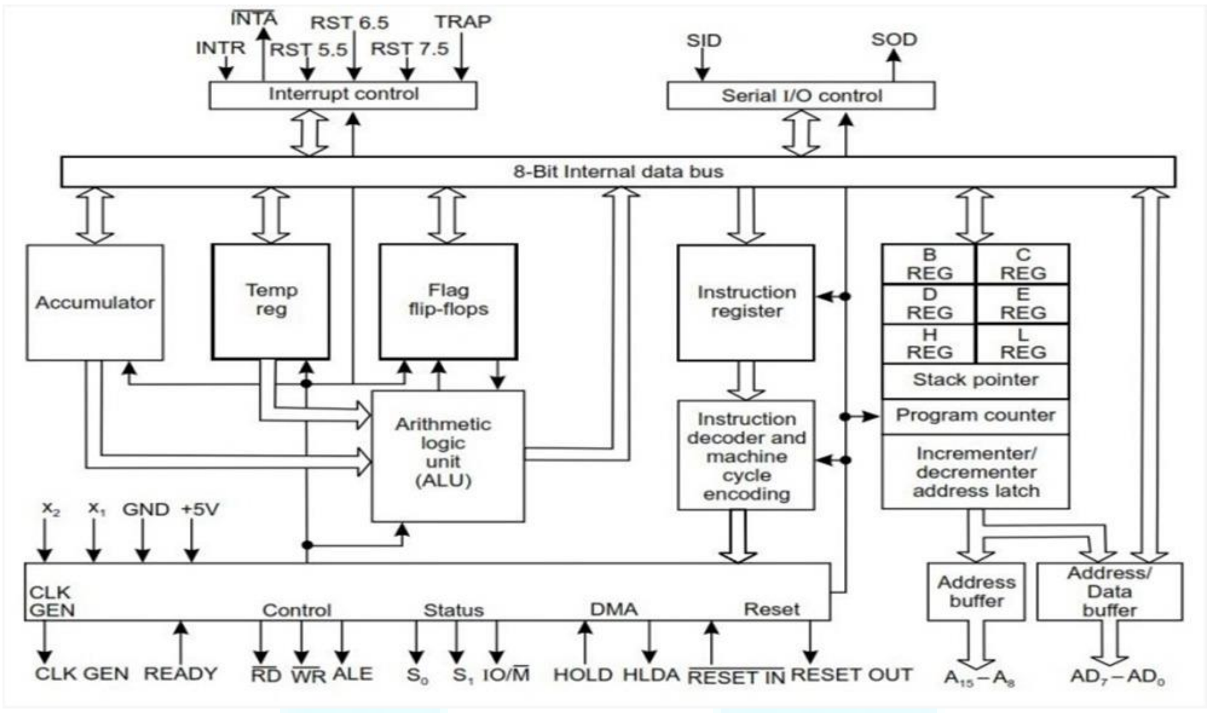
It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address buffer and address-data buffer

The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

Address bus and data bus

Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.



ARCHITECTURE OF 8085

INSTRUCTION SET

- An instruction is a set of codes that the computer processor can understand. The code is usually in 1s and 0s, or machine language. It contains instructions or tasks that control the movement of bits and bytes within the processor.
- An instruction set is a group of commands for a central processing unit (CPU) in machine language. The term can refer to all possible instructions for a CPU or a subset of instructions to enhance its performance in certain situations.

Opcodes and operands

- Each assembly language statement is split into an opcode and an operand.
- The opcode is the instruction that is executed by the CPU.
- The operand is the data or memory location used to execute that instruction.

EXAMPLE:

$$\begin{tikzcd} \text{MOV} \arrow[d] & \text{A, B} \arrow[d] \\ \text{OPCODE} & \text{OPERANDS} \end{tikzcd}$$

CLASSIFICATION OF INSTRUCTION SET

1. Data-transfer instructions
2. Arithmetic instructions
3. Logical instructions
4. Branching instructions
5. Control instructions

1. DATA TRANSFER INSTRUCTION SET

• Data transfer instructions are the instructions which transfer data in the microprocessor. They are also called copy instructions.

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. Example − MOV K, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. Example − MVI K, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. Example − LDA 2034K

LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. Example − LDAX K
LXI	Reg. pair , 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory. Example − LXI K, 3225L
LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. Example − LHLD 3225K
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.

			Example − STA 325K
STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. Example − STAX K
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example − SHLD 3225K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example − XCHG
SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H

			register provide the high-order address and the contents of the L register provide the low-order address. Example − SPHL
XTHL	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1). Example − XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example − PUSH K

POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. Example − POPK
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example − OUT K9L
IN	8-bit port address	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example − IN5KL

2. ARITHMETIC INSTRUCTIONS

Arithmetic Instructions are the instructions which perform basic arithmetic operations such as addition, subtraction and a few more.

Opcode	**Operand**	**Meaning**	**Explanation**
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADD K.
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADC K
ADI	8-bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. Example − ADI 55K
ACI	8-bit data	Add the immediate to the accumulator with carry	The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ACI 55K
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16-bit data into the register pair

			designated in the operand. Example − LXI K, 3025M
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. Example − DAD K
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. Example − SUB K
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. Example − SBB K
SUI	8-bit data	Subtract the immediate from the accumulator	The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator.

			Example − SUI 55K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example − XCHG
INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. Example − INR K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. Example − INX K
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. Example − DCR K
DCX	R	Decrement the register pair by 1	The contents of the designated register pair are decremented by 1 and their result is

			stored at the same place. Example − DCX K
DAA	None	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit BCD digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. Example − DAA

3. LOGICAL INSTRUCTIONS

- Logical instructions are the instructions that perform basic logical operations such as AND, OR, etc.
- In the 8085 microprocessor, the destination operand is always the accumulator. Here logical operation works on a bitwise level.

le	**ınd**	**ing**	**nation**

CMP	R M	are the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	ata	are immediate with the accumulator	The second byte data is compared with the contents of the accumulator.
ANA	R M	l AND register or memory with the accumulator	contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
	ata	l AND immediate with the accumulator	contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
XRA	R M	sive OR register or memory with the accumulator	contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
	8-bit data with the accumulator	sive OR immediate with	contents of the accumulator are Exclusive OR with the 8-bit data and

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

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

4. BRANCHING INSTRUCTION

- Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction.

Branch instructions are classified into the following three categories.

(i) JUMP INSTRUCTIONS.

-Jump instructions are classified into two categories: Unconditional and conditional jump instructions.

- Unconditional jump instructions

Unconditional jump instructions allow the programmer to set up continuous loops.

- Conditional Jump Instructions

Conditional jump instructions allow the microprocessor to make decisions based on certain conditions indicated by various flags. They check the flag condition and decide to change or not change the sequence of the program.

(ii) CALL AND RETURN INSTRUCTIONS.

-The Call and Return instructions in 8085 are associated with the subroutine technique.

-When the CALL instruction is executed, the 8085 stores the contents of the Program Counter on the top of the stack and transfers the program to the location of the subroutine. The Return instruction inserts the element from the top of the stack two the Program Counter.

(iii) RESTART INSTRUCTIONS.

The Restart instruction in 8085 is associated with the interrupt technique. They are executed the same way as Call instruction.

Opcode	**Operand**	**Meaning**	**Explanation**
JMP	16-bit address	Jump unconditionally	The program sequence is transferred to the memory address given in the operand.
	16-bit address	Jump conditionally	The program sequence is transferred to the memory address given in the operand based on the specified flag of the PSW.

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			
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	16-bit address	Unconditional subroutine call	The program sequence is transferred to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.
RET	None	Return from subroutine unconditionally	The program sequence is transferred from the subroutine to the calling program.
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	None	Return from subroutine conditionally	The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW and the program execution begins at the new address.

RZ — Return on zero — Z=1 RNZ — Return on no zero — Z=0 RPE — Return on parity even — P=1			
PCHL	None	Load the program counter with HL contents	The contents of registers H & 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.
RST	0-7	Restart	The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations. 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 The 8085 has additionally 4

			interrupts, which can generate RST instructions internally and doesn’t require any external hardware. Following are those instructions and their Restart addresses – Interrupt — Restart Address TRAP — 0024H RST 5.5 — 002CH RST 6.5 — 0034H RST 7.5 — 003CH

5. **CONTROL INSTRUCTION**

- These types of instructions control machine functions such as Halt, Interrupt, or do nothing. This type of instructions alters the different type of operations executed in the processor.

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are

			disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

INSTRUCTION CYCLE

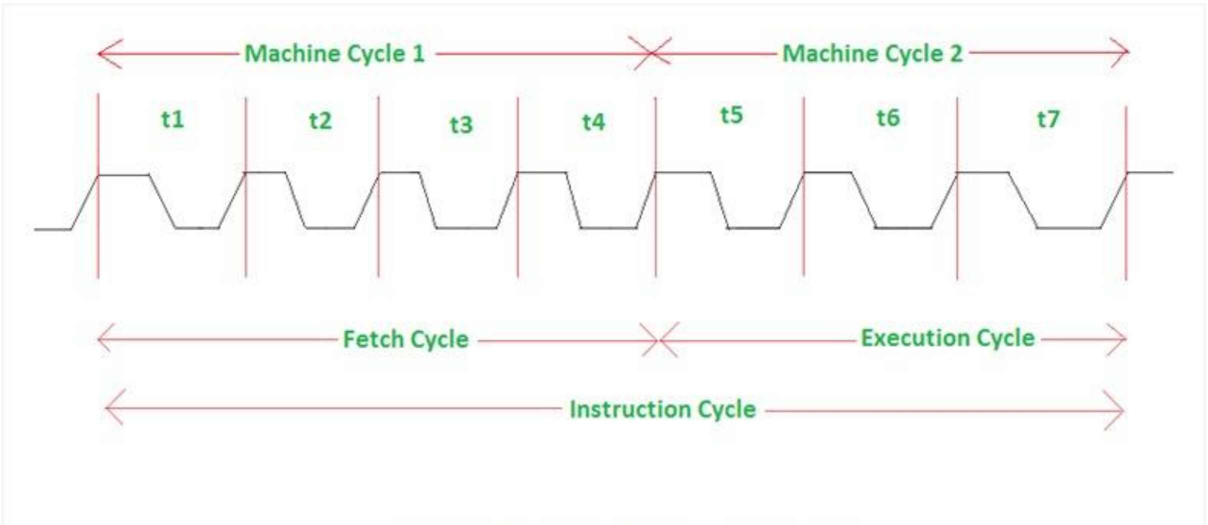
Time required to execute and fetch an entire instruction is called instruction cycle.

It consists:

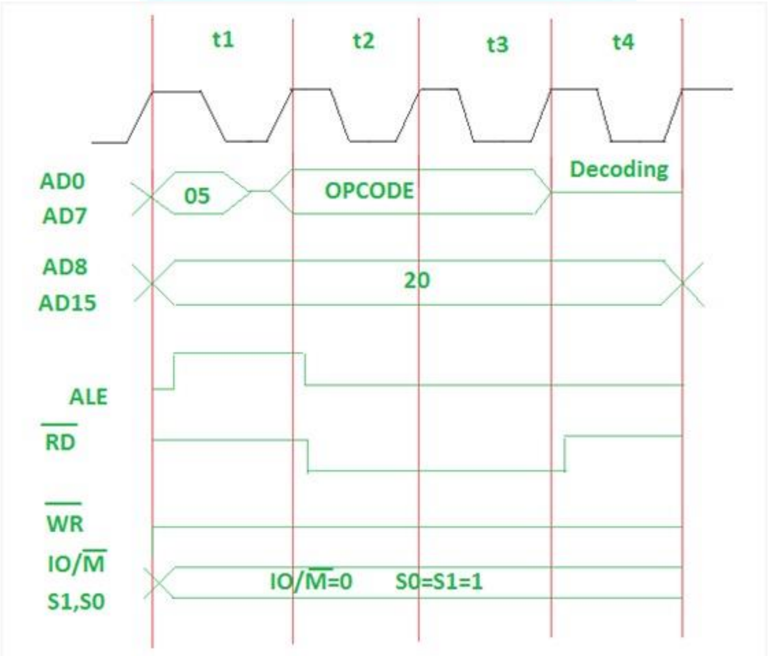
- **Fetch cycle** – The next instruction is fetched by the address stored in program counter (PC) and then stored in the instruction register.
- **Decode instruction** – Decoder interprets the encoded instruction from instruction register.
- **Reading effective address** – The address given in instruction is read from main memory and required data is fetched. The effective address depends on direct addressing mode or indirect addressing mode.
- **Execution cycle** – consists memory read (MR), memory write (MW), input output read (IOR) and input output write (IOW)

MACHINE CYCLE

- The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called machine cycle.
- One time period of frequency of microprocessor is called t-state.
- A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.
- Fetch cycle takes four t-states and execution cycle takes three t-states.



Instruction cycle in 8085 microprocessor



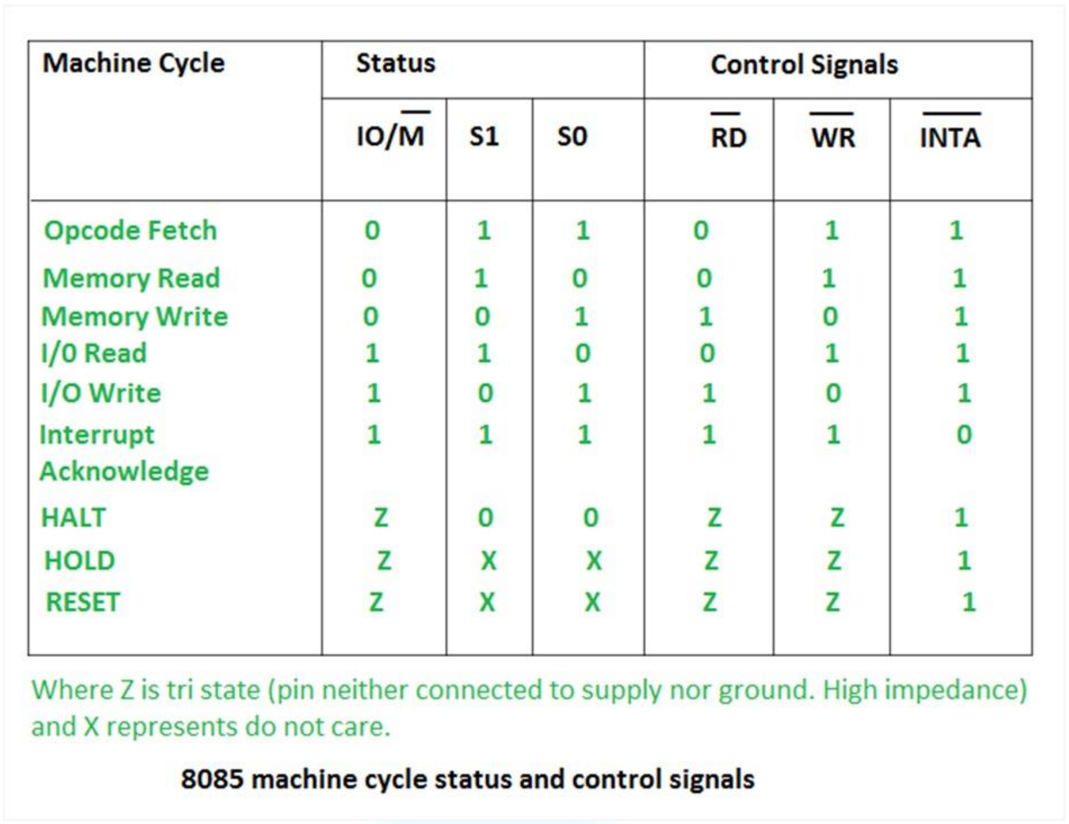
Timing diagram for opcode fetch

Above diagram represents:

- 05 – Lower bit of address where opcode is stored. Multiplexed address and data bus AD0-AD7 are used.
- 20 – Higher bit of address where opcode is stored. Multiplexed address and data bus AD8-AD15 are used.
- ALE – Provides signal for multiplexed address and data bus. If signal is high or 1, multiplexed address and data bus will be used as address bus. To fetch lower bit of

address, signal is 1 so that multiplexed bus can act as address bus. If signal is low or 0, multiplexed bus will be used as data bus. When lower bit of address is fetched then it will act as data bus as the signal is low.

- RD (low active) – If signal is high or 1, no data is read by microprocessor. If signal is low or 0, data is read by microprocessor.
- WR (low active) – If signal is high or 1, no data is written by microprocessor. If signal is low or 0, data is written by microprocessor.
- IO/M (low active) and S1, S0 – If signal is high or 1, operation is performing on input output. If signal is low or 0, operation is performing on memory



Machine Cycle	Status			Control Signals		
	IO/$\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
I/0 Read	1	1	0	0	1	1
I/O Write	1	0	1	1	0	1
Interrupt Acknowledge	1	1	1	1	1	0
HALT	Z	0	0	Z	Z	1
HOLD	Z	X	X	Z	Z	1
RESET	Z	X	X	Z	Z	1

Where Z is tri state (pin neither connected to supply nor ground. High impedance) and X represents do not care.

8085 machine cycle status and control signals

- The following are the various machine cycles of 8085 microprocessor.

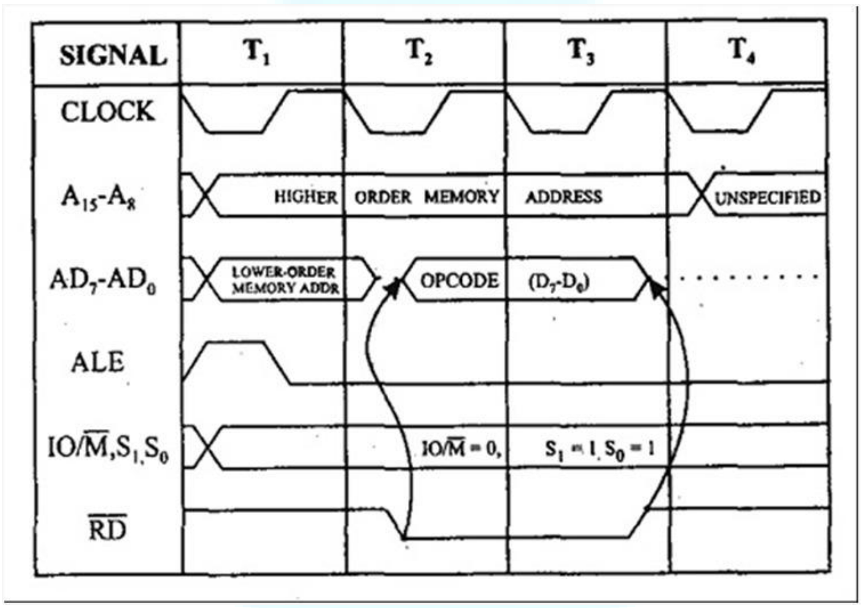
1. Opcode Fetch (OF)
2. Memory Read (MR)
3. Memory Write (MW)
4. I/O Read (IOR)
5. I/O Write (IOW)
6. Interrupt Acknowledge (IA)
7. Bus Idle (BI)

- 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.

Opcode Fetch (OF) machine cycle of 8085:

- Each instruction of the microprocessor has one byte Opcode.
- The Opcode is stored in memory.
- In order to fetch the Opcode from memory, processor executes the Opcode Fetch machine cycle.
- So, every instruction starts with Opcode Fetch machine (OFM) cycle.
- The time taken by the microprocessor to execute the Opcode Fetch cycle is4T (T-states).
- In order to fetch the Opcode from memory, the first 3 T-states are used.
- 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	T1	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

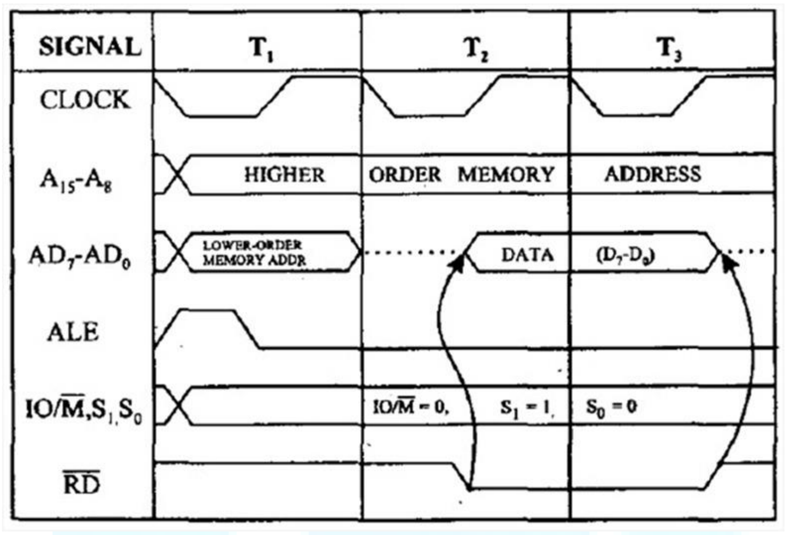
		the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/M' = 0, S1 =1 and S0 = 1. These status signals do not change throughout the OF machine cycle.
4	T2	The microprocessor makes the RD' line LOW to enable memory read and increments the Program Counter.
5		The contents on D7 – D0 (i.e. the Opcode) are placed on the address / data bus.
6	T3	The microprocessor transfers the Opcode on the address / data bus to Instruction Register (IR).
7		The microprocessor makes the RD' line HIGH to disable memory read.
8	T4	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.



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

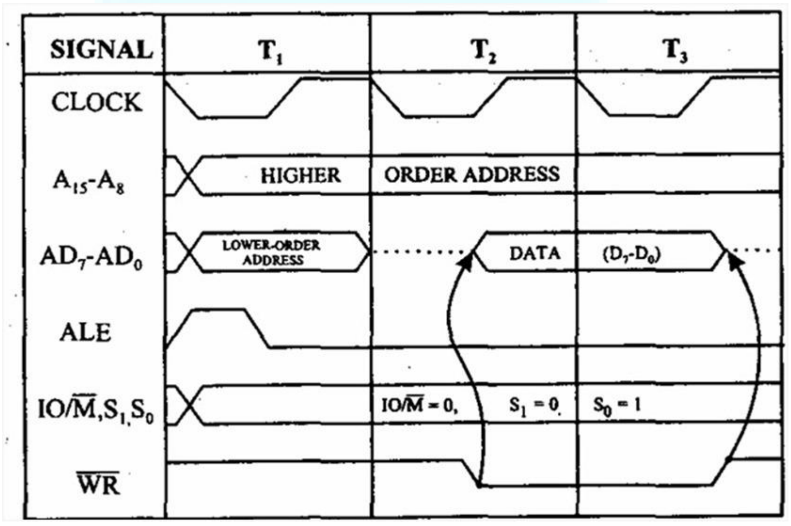
S. No	T state	Operation
1	T1	The microprocessor places the address of the I/O port specified in the instruction on A15 – A8 address bus and also on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/*M*’ = 0, S1 =1 and S0 = 0. These status signals do not change throughout the I/O read machine cycle.

4	T2	The microprocessor makes the 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	T3	The data loaded on the address / data bus is moved to the microprocessor i.e., to the accumulator.
7 .		The microprocessor makes the RD' line HIGH to disable the I/O 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 to disable the memory write machine cycle are given in table.

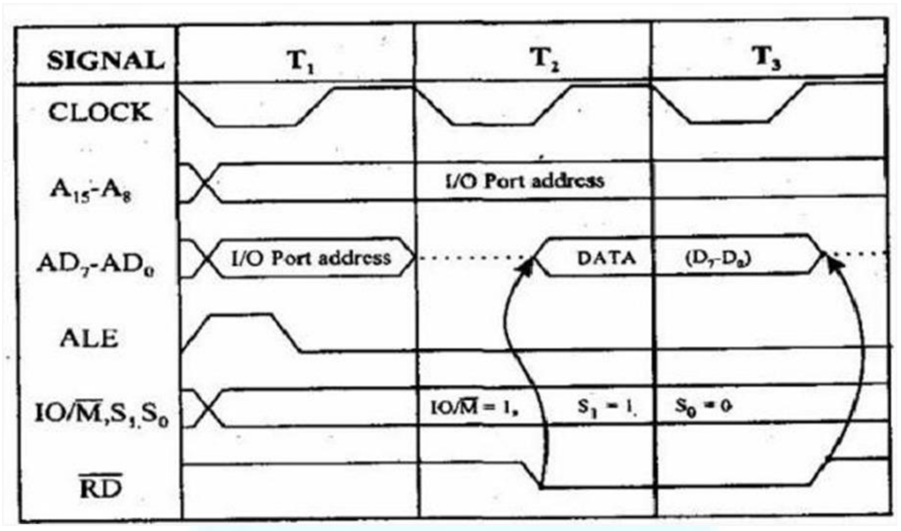
S. No	T state	Operation
1	T1	The microprocessor places the higher order 8-bits of the memory address on A15 – A8 address bus and the lower order 8-bits of the memory address on AD7 – AD0 address / data bus.
2		The microprocessor makes the ALE signal HIGH and at the middle of T1 state, ALE signal goes LOW.
3		The status signals are changed as IO/M’ = 0, S1 =0 and S0 = 1. These status signals do not change throughout the memory write machine cycle.
4	T2	The microprocessor makes the WR’ lines LOW to enable memory write.
5		The contents of the specified register are placed on the address / data bus.
6	T3	The data placed on the address / data bus is transferred to the specified memory location.
7 .		The microprocessor makes the 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	T1	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/*M*’ = 0, S1 =1 and S0 = 0. These status signals do not change throughout the I/O read machine cycle.
4	T2	The microprocessor makes the *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	T3	The data loaded on the address / data bus is moved to the microprocessor ie., to the accumulator.
7		The microprocessor makes the RD’ line HIGH to disable the I/O read operation.

ADDRESSING MODE

- The way of specifying data to be operated by an instruction is called addressing mode.
- The addressing modes in 8085 microprocessors are instructions used to transfer data and perform operations on them.

Addressing Modes for 8085 microprocessor are:

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Indirect Addressing
- Implied Addressing

Immediate Addressing Mode:

- In immediate addressing mode, the data (8 / 16 bit) is specified in the instruction itself.
- The immediate addressing instructions are either 2 bytes or 3 bytes long.
- In 2 byte instruction, the first byte is OPCODE, and the second byte is the 8-bit data.
- In 3 byte instruction, the first byte is OPCODE, second and third bytes are 16-bit data.
- The instruction containing the letter “I” indicate immediate addressing mode.

Examples:

1. MVI A, A0 H: This instruction transfers immediate data (A0 H) to A register.

2. LXI H, C200 H: This instruction transfer 16-bit immediate data C200 to HL register pair. Lower order data (00H) to L register and high order data (C2 H) to H register.

Register Addressing Mode:

- In register addressing mode the source and destination operands are general-purpose registers.
- The register addressing instructions are generally of 1 byte i.e. OPCODE only.
- The OPCODE specifies the operation and registers to be used to perform the operation.

Examples:

1. MOV D, B: This instruction copies the contents of register B to the D register. The source and destination operands are both registers.

2. ADD B: This instruction adds the content of B register and A register. The data is present in both B and A registers. The result is stored in the accumulator.

3. PCHL: This instruction will transfer the content of register pair HL to the PC (Program Counter)

Direct Addressing Mode:

- In direct addressing mode, the 16-bit address of the operand is given within the instruction itself.
- The instruction in the direct addressing mode is 3-byte instructions. The first byte is OPCODE, the second slower order address mode and the third is the higher-order address mode.
- For I/O instruction that uses direct addressing mode are 2-byte as the address if I/O is one byte.

Examples:

1. LDA C200 H: Load accumulator directly from the memory location. In this instruction, the contents of C200 memory location are transferred to the accumulator.

2. STA C200 H: Store accumulator directly to memory location. In this instruction, the content of the accumulator are store at memory location C200 H.

Indirect Addressing Mode:

- In indirect addressing mode the instruction reference the memory through a register pair.

- The memory address where the operand is located is specified by the content of a register pair.

Examples:

1. MOV A, M: In this case, M is a memory pointer specifying the HL register pair where the address is stored. The contents of the HL pair are used as addresses and the content of that memory location is transferred to the accumulator.

2. LDAX B: In this case, the BC register pair is used as an address and the content of the memory location specified by the BC register pair is copied to the accumulator.

Implied Addressing or Implicit Addressing Mode:

- The implied mode of addressing does not require any operand.

The data is specified within OPCODE itself.

- Generally, the implied addressing mode instruction is a 1-byte instruction.

- The data is supposed to be present generally in the accumulator.

Examples:

- RAL: Rotate accumulator left, it operates on the data in accumulator only. So whenever RAL is used it is implied that the data to be operated on is available in the accumulator only.

- CMC: Complement carry flag.

INTERRUPTS

- Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

- The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

Interrupts can be classified into various categories based on different parameters:

- **HARDWARE AND SOFTWARE INTERRUPT**

HARDWARE INTERRUPT

-When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as Hardware Interrupts.

-There are 5 Hardware Interrupts in 8085 microprocessor.

-They are – INTR, RST 7.5, RST 6.5, RST 5.5,TRAP

SOFTWARE INTERRUPT

-Software Interrupts are those which are inserted in between the program which means these are mnemonics of microprocessor.

- There are 8 software interrupts in 8085 microprocessor.

- They are – RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7.

- **VECTORED AND NON-VECTORED INTERRUPT**

VECTORED INTERRUPTS

-Vectored Interrupts are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address.

-Vector Addresses are calculated by the formula 8 * TYPE

INTERRUPT	**VECTOR ADDRESS**
TRAP (RST 4.5)	24 H
RST 5.5	2C H
RST 6.5	34 H
RST 7.5	3C H

- For Software interrupts vector addresses are given by:

INTERRUPT	VECTOR ADDRESS
RST 0	00 H
RST 1	08 H
RST 2	10 H
RST 3	18 H
RST 4	20 H

RST 5	28 H
RST 6	30 H
RST 7	38 H

NON VECTORED INTERRUPTS

-Non-Vectored Interrupts are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts. INTR is the only non-vectored interrupt in 8085 microprocessor.

- **MASKABLE AND NON-MASKABLE INTERRUPTS**

MASKABLE INTERRUPTS

-Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled.

-INTR, RST 7.5, RST 6.5, RST 5.5 is maskable interrupts in 8085 microprocessor.

NON MASKABLE INTERRUPT

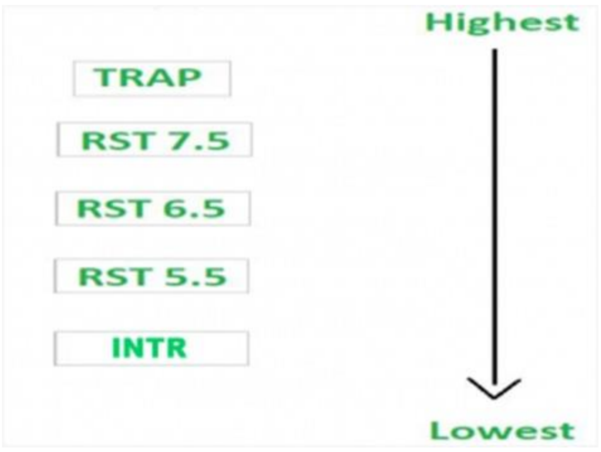
-Non-Maskable Interrupts are those which cannot be disabled or ignored by microprocessor.

-TRAP is a non-maskable interrupt.

-It consists of both level as well as edge triggering and is used in critical power failure conditions.

Priority of Interrupts

-When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.

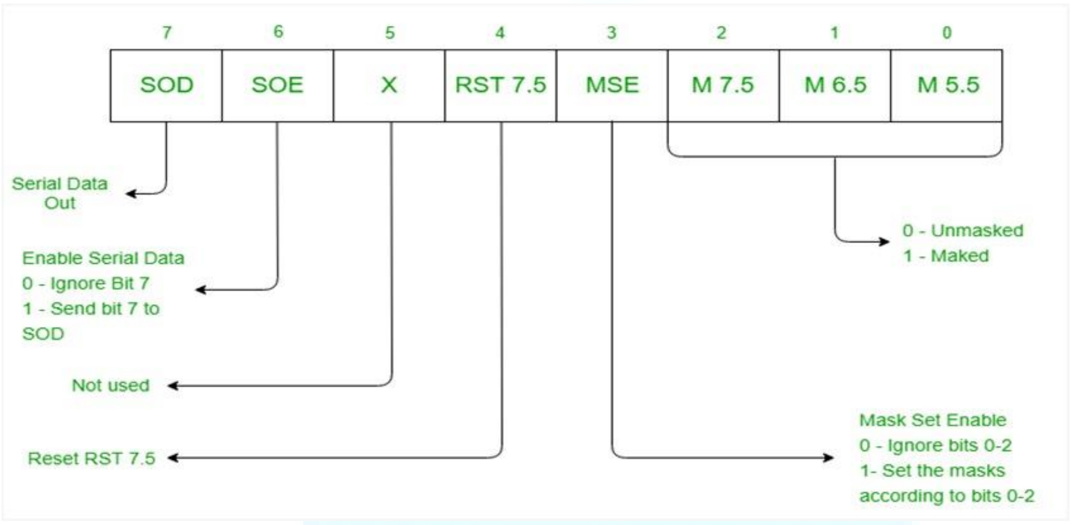


Instruction for Interrupts –

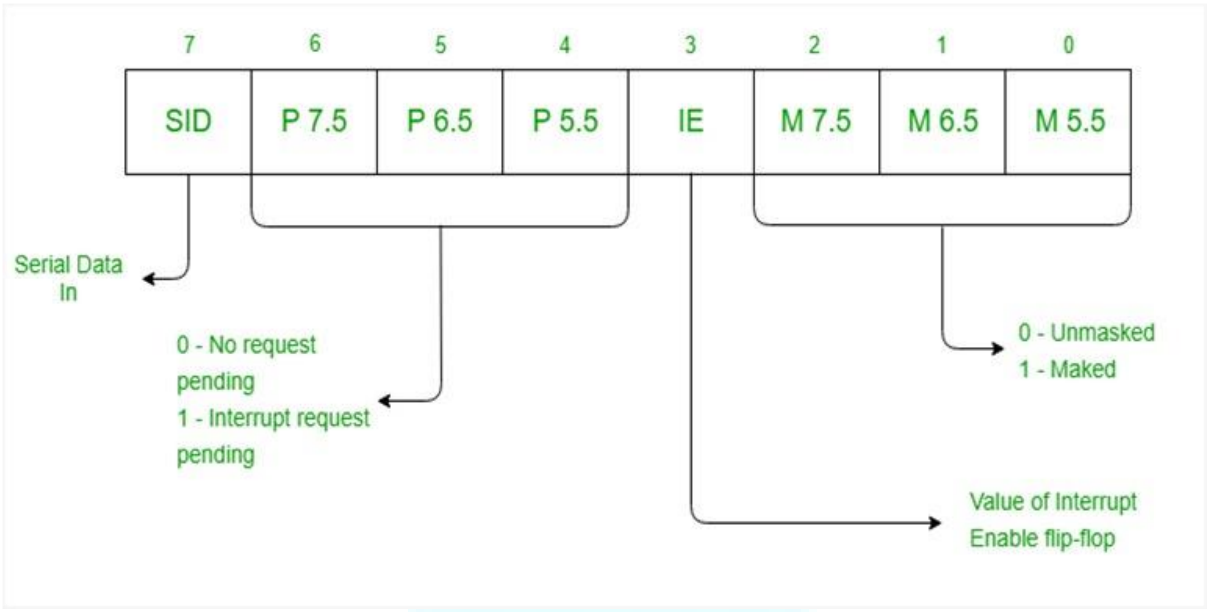
1. Enable Interrupt (EI) – The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flags are affected. After a system reset, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again (except TRAP).

2. Disable Interrupt (DI) – This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts. No flags are affected by this instruction.

3. Set Interrupt Mask (SIM) – It is used to implement the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by setting various bits to form masks or generate output data via the Serial Output Data (SOD) line. First the required value is loaded in accumulator then SIM will take the bit pattern from it.



4.Read Interrupt Mask (RIM) – This instruction is used to read the status of the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by loading into the A register a byte which defines the condition of the mask bits for the interrupts. It also reads the condition of SID (Serial Input Data) bit on the microprocessor.


7
6
5
4
3
2
1
0
SID
P 7.5
P 6.5
P 5.5
IE
M 7.5
M 6.5
M 5.5
Serial Data In
0 - No request pending
1 - Interrupt request pending
0 - Unmasked
1 - Maked
Value of Interrupt Enable flip-flop