Unit – 1: Basics of Computer Organization and Processor Evolution

Observe the characteristic of Intel processor from 4 bit (4004) to i7:

1. First Generation Microprocessors:

- The first generation microprocessors were introduced in the year 1971-1972. The instructions of these microprocessors were processed serially, they fetched the instruction, decoded and then executed it. When an instruction of the microprocessor was finished, then the microprocessor updates the instruction pointer & fetched the following instruction, performing this consecutive operation for each instruction in turn.

2. Second Generation Microprocessors:

- In the year 1970, a small number of transistors were available on the integrated circuit in the second-generation microprocessors. Examples of the second-generation microprocessors are 16-bit arithmetic 7 pipelined instruction processing, MC68000 Motorola microprocessor. These processors are introduced in the year 1979, and Intel 8080 processor is another example of the microprocessor. The second generation of the microprocessor is defined by overlapped fetch, decode, and execute the steps. When the first generation is processed in the execution unit, then the second instruction is decoded and the third instruction is fetched.

3. Third Generation Microprocessors:

- The third generation microprocessors were introduced in the year 1978, as denoted by Intel's 8086 and the Zilog Z8000. These were 16-bit processors with a performance like mini computers. These types of microprocessors were different from the previous generations of microprocessors in that all main workstation industrialists began evolving their own ISC based microprocessor architectures.

4. Fourth Generation Microprocessors:

- As many industries converted from commercial microprocessors to in house designs, the fourth generation microprocessors are entered with outstanding design with a million transistors. Leading-edge microprocessors like Motorola's 88100 and Intel's 80960CA could issue & retire more than one instruction per clock cycle.

5. Fifth Generation Microprocessors:

- Fifth-generation microprocessors employed decoupled superscalar processing, and their design soon exceeded 10 million transistors. In the fifth generation, PCs are a low-margin, high volume business conquered by a single microprocessor.

- On Dec 23rd, 1947, the Transistor was invented in Bell lab whereas an integrated circuit was invented in 1958 by J Kilby in Texas Instruments. So, Intel or INTegrated ELectronics has invented the first microprocessor.

NAME	YEAR	TRANSISTORS	DATA WIDTH	CLOCK SPEED
8080	1974	6,000	8 bits	2 MHz
8085	1976	6,500	8 bits	5 MHz
8086	1978	29,000	16 bits	5 MHz
8088	1979	29,000	8 bits	5 MHz
80286	1982	134,000	16 bits	6 MHz
80386	1985	275,000	32 bits	16 MHz
80486	1989	1,200,000	32 bits	25 MHz
PENTIUM	1993	3,100,000	32/64 bits	60 MHz
PENTIUM II	1997	7,500,000	64 bits	233 MHz
PENTIUM III	1999	9,500,000	64 bits	450 MHz
PENTIUM IV	2000	42,000,000	64 bits	1.5 GHz

1. 4-bit Microprocessor:

- The INTEL 4004/4040 was invented in the year 1971 by Stanley Mazor & Ted Hoff. The clock speed of this microprocessor is 740 KHz. The number of transistors used in this microprocessor is 2,300 and instruction per second is 60K. The number of pins of this microprocessor is 16.

2. 8-bit Microprocessor

- The 8008 processor was invented in the year 1972. The clock speed of this microprocessor is 500 KHz and instruction per second is 50K

- The 8080 microprocessor was invented in the year 1974. The clock speed is 2 MHz. The number of transistors used is 60k and instruction per second is 10 times quicker as compared with 8008 processor.

- The 8085 microprocessor was invented in the year 1976. The clock speed is 3 MHz. The number of transistors used is 6,500 and instruction per second is 769230. The number of pins of this microprocessor is 40

3. 16-bit Microprocessor

- The 8086 microprocessor was invented in the year 1978. The clock speed is 4.77, 8 & 10 MHz. The number of transistors used is 29000 and instruction per second is 2.5 Million. The number of pins of this microprocessor is 40.

- The 8088 microprocessor was invented in the year 1979 and instruction per second is 2.5 Million.

- The microprocessors like 80186 or 80188 were invented in the year 1982. The clock speed is 6 MHz.

- The 80286 microprocessor was invented in the year 1982. The clock speed is 8 MHz. The number of transistors used is 134000 and instruction per second is 4 Million. The number of pins of this microprocessor is 68.

4. 32-bit Microprocessor

- The Intel 80386 microprocessor was invented in the year 1986. The clock speed is 16 MHz to 33 MHz. The number of transistors used is 275000.

- The Intel 80486 microprocessor was invented in the year1986. The clock speed is 16MHz to 100 MHz. The number of transistors used is 1.2 Million transistors and instruction per second is 8 KB of cache memory.

- The PENTIUM microprocessor was invented in the year 1993. The clock speed is 66 MHz and instruction per second is Cache memory 8-bit for instructions 8- bit for data. The number of pins of this microprocessor is 237 PGA.

5. 64-bit Microprocessor

- The INTEL core 2 microprocessor was invented in the year 2006. The clock speed is 1.2 GHz to 3 GHz. The number of transistors used is 291 Million and instruction per second is 64 KB of L1 cache for each core 4 MB of L2 cache.

- The i3, i5, i7 microprocessors were invented in the years 2007, 2009, 2010 2. The clock speed is 2GHz to 3.3GHz, 2.4GHz to 3.6GHz & 2.93GHz to t 3.33GHz.

Basic CPU Structure of CU, ALU and MU:

1. Control Unit (CU):

✧ Function:

- The Control Unit is responsible for coordinating and controlling the operations of the entire CPU.

- It fetches instructions from memory and decodes them to determine the necessary actions.

➢ Key Components:

1. Instruction Register (IR):

- Holds the current instruction being executed.

- The CU fetches the instruction from memory and stores it in the IR.

2. Program Counter (PC):

- Keeps track of the memory address of the next instruction to be fetched.

- Gets incremented as each instruction is fetched.

3. Instruction Decoder:

- Decodes the instructions fetched from memory to determine the actions to be performed.

4. Control Lines:

- Transmit control signals to various components of the CPU, coordinating their activities.

2. Arithmetic Logic Unit (ALU):

✧ Function:

- The Arithmetic Logic Unit is responsible for performing arithmetic and logical operations.
- It executes the actual computations specified by the instructions.

➢ Key Components:

1. Arithmetic Unit:

- Performs arithmetic operations such as addition, subtraction, multiplication, and division.

2. Logic Unit:

- Performs logical operations such as AND, OR, NOT, and comparisons.

3. Registers:

- Temporary storage locations within the ALU for operands and results.
- Accumulator, operand registers, and status flags are examples.

4. Memory Unit (MU):

✧ Function:

- The Memory Unit is responsible for storing data and instructions.
- It includes both primary and secondary memory.

➢ Key Components:

1. Registers:

- Fast, small-sized storage locations within the CPU used for quick access.
- Examples include data registers, address registers, and program counter.

2. Cache Memory:

- A small-sized, high-speed memory that stores frequently accessed data and instructions for faster retrieval.

3. Primary Memory (RAM):

- Volatile memory used for temporary storage of data and program instructions.
- Random Access Memory (RAM) is a common type of primary memory.

4. Secondary Memory (Storage):

- Non-volatile memory used for long-term storage of data and programs.
- Examples include Hard Disk Drives (HDDs) and Solid State Drives (SSDs).

5. Interaction:

1. Fetch-Decode-Execute Cycle:

- The Control Unit fetches the next instruction from memory using the Program Counter.
- The instruction is stored in the Instruction Register.
- The Control Unit decodes the instruction and determines the necessary actions.
- The ALU performs the specified arithmetic or logical operation.
- Results may be stored in registers or sent back to memory.

2. Data Flow:

- Data is transferred between the CPU and memory or between different registers during the execution of instructions.

Various Registers used in CPU & its applications AC, DR, AR, PC, MAR, MBR, IR:

1. Accumulator (AC - Accumulator Register):

✧ Application:

- Primary register for arithmetic and logic operations.
- Accumulates results during computations.

➢ Example:

- In an addition operation, the accumulator would store the sum of two operands.

2. Data Register (DR - Data Register):

✧ Application:

- Temporarily holds data during the execution of an instruction.

➢ Example:

- When data is loaded from memory, it may be temporarily stored in the data register.

3. Address Register (AR - Address Register):

✧ Application:

- Holds the address of a memory location being accessed

➢ Example:

- When fetching an instruction or data from memory, the address register holds the memory address.

4. Program Counter (PC - Program Counter):

✧ Application:

- Keeps track of the memory address of the next instruction to be fetched.

➢ Example:

- Incremented after each instruction fetch, pointing to the next instruction in memory.

5. Memory Address Register (MAR - Memory Address Register):

✧ Application:

- Holds the memory address for data read or written.

➢ Example:

- Before accessing memory, the CPU loads the memory address into the MAR.

6. Memory Buffer Register (MBR - Memory Buffer Register):

✧ Application:

- Temporarily holds data to be written to or read from memory.

➢ Example:

- When reading data from memory, the MBR holds the retrieved data.

7. Instruction Register (IR - Instruction Register):

✧ Application:

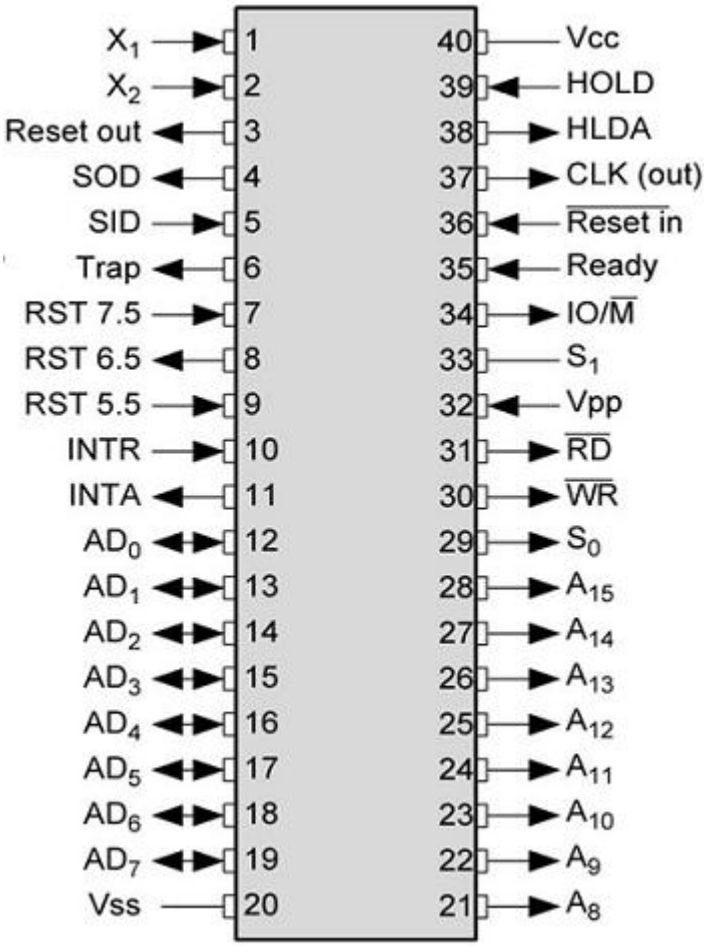
- Holds the current instruction being executed.

➢ Example:

- After fetching an instruction from memory, it gets stored in the instruction register for decoding and execution.

Unit – 2: 8085 Microprocessor

• 8085 Pin Diagram & Pin Functions:-



The pins of a 8085 microprocessor can be classified into seven groups:-

1. Address bus

A15-A8, it carries the most significant 8-bits of memory/IO address.

2. Data bus

AD7-AD0, it carries the least significant 8-bit address and data bus.

3. Control and status signals

These signals are used to identify the nature of operation. There are 3 control signal and 3 status signals.

Three control signals are RD, WR & ALE.

1. **RD** − This signal indicates that the selected IO or memory device is to be read and is ready for accepting data available on the data bus.

2. **WR** − This signal indicates that the data on the data bus is to be written into a selected memory or IO location.

3. **ALE** − It is a positive going pulse generated when a new operation is started by the microprocessor. When the pulse goes high, it indicates address. When the pulse goes down it indicates data.

- ## Three status signals are IO/M, S0 & S1:-

1. IO/M:-

This signal is used to differentiate between IO and Memory operations, i.e. when it is high indicates IO operation and when it is low then it indicates memory operation.

2. S1 & S0:-

These signals are used to identify the type of current operation.

3. Power supply:-

There are 2 power supply signals − VCC & VSS. VCC indicates +5v power supply and VSS indicates ground signal.

- ## Clock signals:-

There are 3 clock signals, i.e. X1, X2, CLK OUT.

1. **X1, X2** − A crystal (RC, LC N/W) is connected at these two pins and is used to set frequency of the internal clock generator. This frequency is internally divided by 2.

2. **CLK OUT** − This signal is used as the system clock for devices connected with the microprocessor.

- ## Interrupts & externally initiated signals:

Interrupts are the signals generated by external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. We will discuss interrupts in detail in interrupts section.

1. **INTA** − It is an interrupt acknowledgment signal.
2. **RESET IN** − This signal is used to reset the microprocessor by setting the program counter to zero.
3. **RESET OUT** − This signal is used to reset all the connected devices when the microprocessor is reset.
4. **READY** − This signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.
5. **HOLD** − This signal indicates that another master is requesting the use of the address and data buses.
6. **HLDA (HOLD Acknowledge)** − It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed.

- ## Serial I/O signals

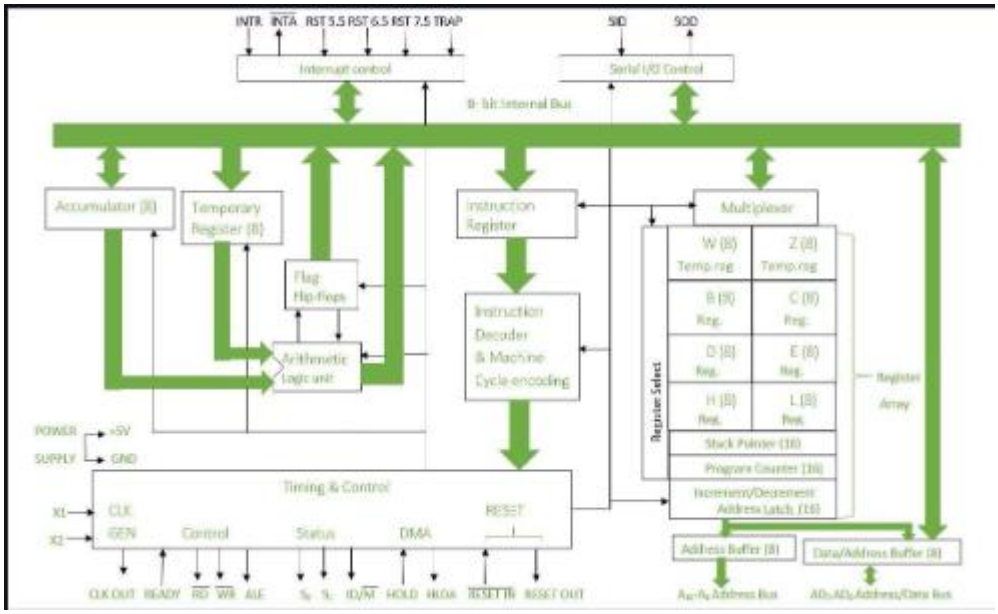
There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.

1. **SOD** (Serial output data line) − The output SOD is set/reset as specified by the SIM instruction.
2. **SID** (Serial input data line) − The data on this line is loaded into accumulator whenever a RIM instruction is executed.

8085 Microprocessor Architecture:-

8085 Microprocessor Architecture Diagram

The 8085 microprocessor architecture diagram is as follows:



The **8085 microprocessor architecture** has the following essential components:

- **Accumulator (A):**

The accumulator is an 8-bit register used for arithmetic and logical operations. It holds one of the operands during calculations and stores the result.

- **General-Purpose Registers (B, C, D, E, H, L):**

The 8085 microprocessor has six general-purpose registers, each of which is 8 bits. These registers can be used for data manipulation and storage.

- **Stack Pointer (SP):**

The stack pointer is a 16-bit register that points to the top of the stack. The stack is used to store return addresses during subroutine calls, as well as local variables and other data.

- **Program Counter (PC):**

Program counter is a 16-bit register that holds the memory address of the following instruction for fetching and executing. It automatically increments after each instruction fetch, allowing programs to execute the next instruction quickly.

- **Flag Register (F):**

The flag register is a 8-bit register that contains several flags that indicate the status of certain conditions after arithmetic and logical operations. The flags are as follows:

Carry Flag (C): Set if there is a carry or borrow during arithmetic operations.

Zero Flag (Z): Set if the result of an operation is zero.

Sign Flag (S): Set if the result of an operation is negative (MSB set).

Parity Flag (P): Set if the result of an operation has even parity.

Auxiliary Carry Flag (AC): Set if there is a carry or borrow from bit 3 to bit 4 during arithmetic operations.

- **Address and Data Buses:**

The **8085 microprocessor** has a 16-bit address bus, allowing it to address up to 64 KB of memory. It also has an 8-bit data bus for transferring data between the microprocessor and memory or I/O devices.

- **Instruction Decoder and Control Unit:**

The instruction decoder interprets the fetched instructions and generates control signals to execute the corresponding operation. The control unit coordinates the timing and sequencing of instructions and manages the flow of data and control signals within the microprocessor.

- **Interrupts:**

The 8085 microprocessor supports 5 hardware interrupts as follows:

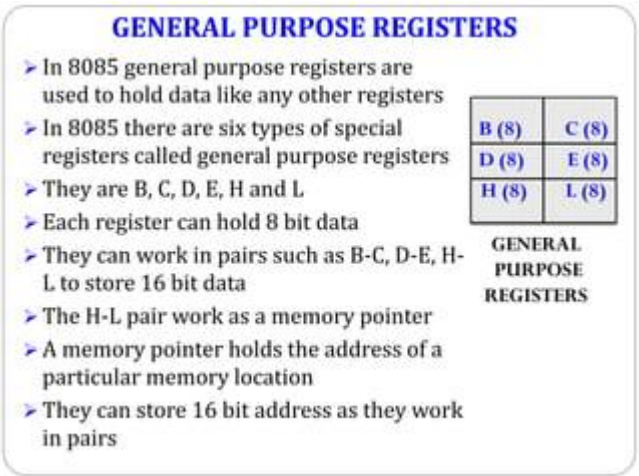
1. INTR
2. RST 5.5
3. RST 6.5
4. RST 7.5
5. TRAP

Interrupts allow the microprocessor to respond to external events and handle them in a prioritized manner.

- **Input/Output (I/O) Ports:**

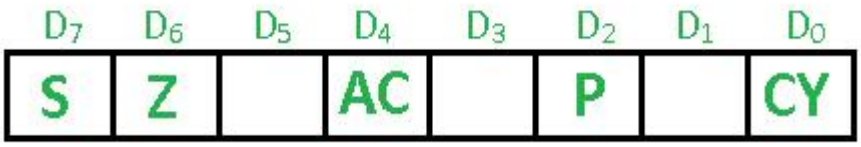
The 8085 microprocessor has a limited number of I/O pins. It uses I/O ports to connect with external devices for input and output operations.

2.1.1. 8085 General Purpose Registers



2.1.2. 8085 Flag Register

The Flag register is a 5-bit register in the 8085 microprocessor that contains information about the status of the arithmetic and logic operations performed by the processor



1. **Sign Flag (S) –** After any operation if the MSB (B(7)) of the result is 1, it indicates the number is negative and the sign flag becomes set, i.e. 1. If the MSB is 0, it indicates the number is positive and the sign flag becomes reset i.e. 0. from 00H to 7F, sign flag is 0 from 80H to FF

Example: MVI A 30 (load 30H in register A) MVI B 40 (load 40H in register B) SUB B (A = A – B) These set of instructions will set the sign flag to 1 as 30 – 40 is a negative number. MVI A 40 (load 40H in register A) MVI B 30 (load 30H in register B) SUB B (A = A – B) These set of instructions will reset the sign flag to 0 as 40 – 30 is a positive number.

2. **Zero Flag (Z) –** After any arithmetical or logical operation if the result is 0 (00)H, the zero flag becomes set i.e. 1, otherwise it becomes reset i.e. 0. 00H zero flags is 1. from 01H to FFH zero flag is 0 1- zero-result 0- non-zero result **Example:** MVI A 10 (load 10H in register A) SUB A (A = A – A) These set of instructions will set the zero flag to 1 as 10H – 10H is 00H

3. **Auxiliary Carry Flag (AC) –** This flag is used in the BCD number system(0-9). If after any arithmetic or logical operation D(3) generates any carry and passes it on to D(4) this flag becomes set i.e. 1, otherwise, it becomes reset i.e. 0. This is the only flag register that is not accessible by the programmer 1-

carry out from bit 3 on addition or borrows into bit 3 on subtraction 0-otherwise
Example: MVI A 2BH (load 2BH in register A) MVI 39H (load 39H in register B) ADD B (A = A + B) These set of instructions will set the auxiliary carry flag to 1

4. **Parity Flag (P) –** If after any arithmetic or logical operation the result has even parity, an even number of 1 bit, the parity register becomes set i.e. 1, otherwise it becomes reset i.e. 0.
Example: MVI A 05 (load 05H in register A) This instruction will set the parity flag to 1 as the BCD code of 05H is 00000101, which contains an even number of ones i.e. 2.

5. **Carry Flag (CY) –** Carry is generated when performing n bit operations and the result is more than n bits, then this flag becomes set i.e. 1, otherwise, it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset
Example: MVI A 30 (load 30H in register A) MVI B 40 (load 40H in register B) SUB B (A = A – B) These set of instructions will set the carry flag to 1 as 30 – 40 generates a carry/borrow. MVI A 40 (load 40H in register A) MVI B 30 (load 30H in register B) SUB B (A = A – B) These set of instructions will reset the carry flag to 0 as 40 – 30 does not generate any carry/borrow.

8085 Instruction Execution

- **Fetch**
- **Decode**
- **Execute operations**

The instruction cycle of the 8085 microprocessor consists of four basic steps, which are:

1. Fetch: In this step, the microprocessor fetches the instruction from the memory location pointed to by the program counter (PC). The PC is incremented by one after the fetch operation.
2. Decode: Once the instruction is fetched, the microprocessor decodes it to determine the operation to be performed and the operands involved.
3. Execute: In this step, the microprocessor performs the operation specified by the instruction on the operands.

4. Store: Finally, the result of the execution is stored in the appropriate memory location or register.

Once the execution of an instruction is complete, the microprocessor returns to the fetch step to fetch the next instruction to be executed. This cycle repeats until the program is complete or interrupted.

◆ Why need the Instruction cycle in 8085 microprocessor ?

The instruction cycle is a fundamental concept in the operation of the 8085 microprocessor because it is the process by which the microprocessor fetches, decodes, and executes instructions. The execution of a program in a microprocessor involves a sequence of instructions, and each instruction is executed using the instruction cycle.

The instruction cycle is necessary in the 8085 microprocessor because it ensures that the instructions are executed in the correct sequence and that the correct operation is performed on the correct data. The fetch step ensures that the correct instruction is obtained from memory, the decode step ensures that the correct operation is determined, and the execute step ensures that the correct operation is performed on the correct data.

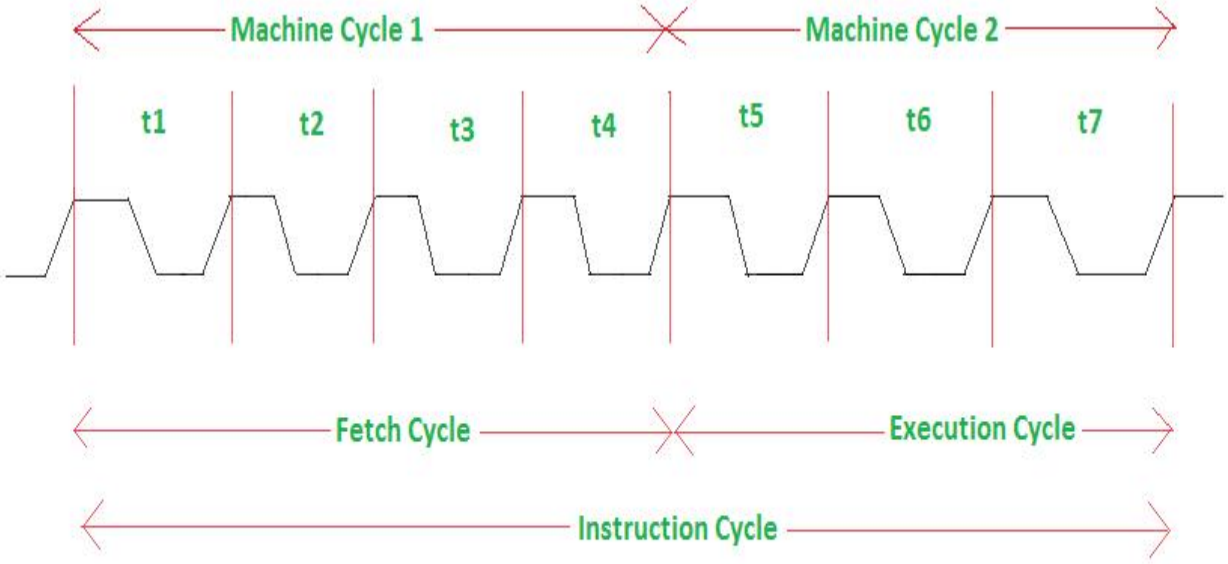
Furthermore, the instruction cycle allows the microprocessor to execute instructions at a very high speed, which is critical in applications where real-time performance is required. By efficiently executing instructions using the instruction cycle, the microprocessor can perform complex tasks and computations quickly and accurately.

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)

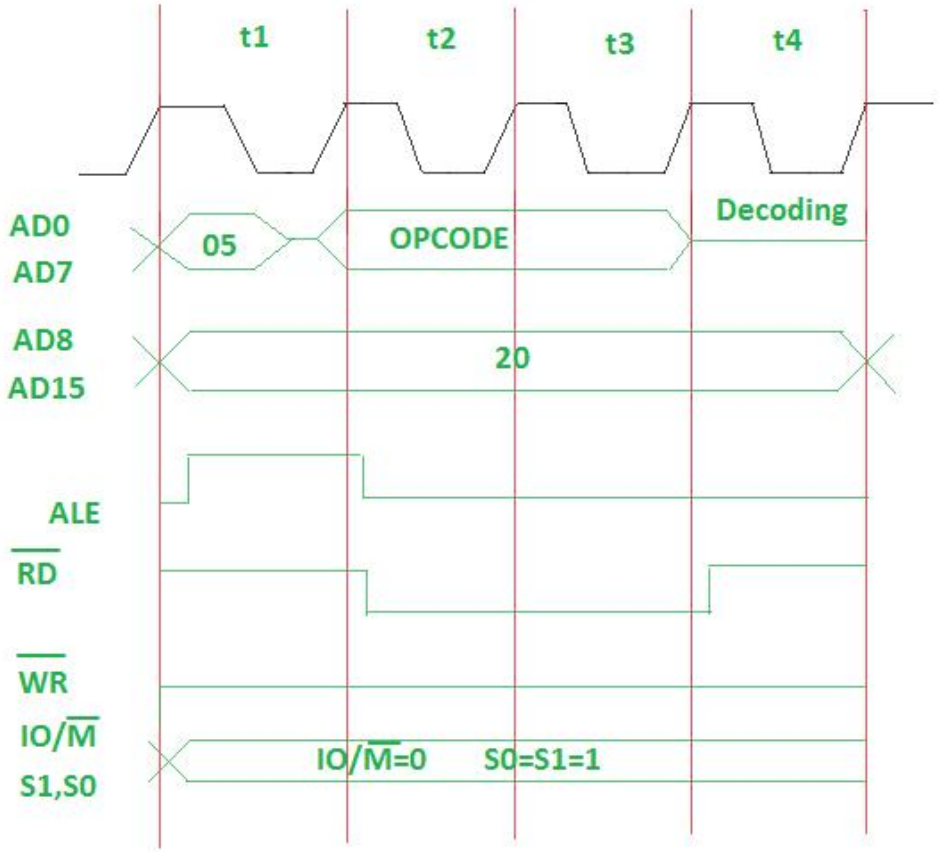
The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called *machine cycle*. One time period of frequency of microprocessor is called *t-state*. A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock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

states.



Instruction cycle in 8085 microprocessor

Timing diagram for fetch cycle or opcode :



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

- **Uses of Instruction cycle in 8085 microprocessor :**

Some of the key uses of the instruction cycle in the 8085 microprocessor include:

1. **Execution of instructions:** The instruction cycle is used to execute the instructions in a program. Each instruction in the program is fetched, decoded, and executed using the instruction cycle.
2. **Control flow:** The instruction cycle is used to control the flow of instructions in a program. Once an instruction is executed, the microprocessor moves on to the next instruction in the program.
3. **Real-time processing:** The instruction cycle allows the 8085 microprocessor to execute instructions quickly and accurately, making it well-suited for real-time processing applications where speed and accuracy are critical.
4. **Resource management:** The instruction cycle is used to manage the resources of the 8085 microprocessor, including the memory and registers. The fetch step retrieves instructions from memory, while the store step writes results back to memory or registers.
5. **Interrupt handling:** The instruction cycle is used to handle interrupts in the 8085 microprocessor. When an interrupt occurs, the current instruction cycle is suspended, and the microprocessor jumps to a separate interrupt routine to handle the interrupt.

Issues of Instruction cycle in 8085 microprocessor :

Some of the key issues of the instruction cycle in the 8085 microprocessor include:

1. Timing: The instruction cycle requires precise timing to ensure that each step is executed correctly. If the timing is off, it can lead to incorrect results or cause the microprocessor to malfunction.
2. Instruction set limitations: The 8085 microprocessor has a limited instruction set, which can make it difficult to perform certain operations or tasks. This can lead to inefficient code and slower execution times.
3. Data transfer: The instruction cycle is used to transfer data between memory and registers, but this process can be slow and inefficient. This can be a problem when working with large amounts of data or when real-time processing is required.
4. Interrupt handling: Although the instruction cycle is used to handle interrupts, it can be challenging to ensure that the microprocessor returns to the correct point in the program after handling an interrupt.
5. Instruction sequencing: The instruction cycle requires instructions to be executed in a specific sequence, which can limit the flexibility of the microprocessor and make it difficult to optimize code for specific tasks.

Unit – 3: 8085 Assembly Language Programming

- **Instruction format opcode & Operands:-**

1. **Opcode (Operation Code):** This is the part of the instruction that indicates the operation or action to be executed by the computer's central processing unit (CPU). The opcode is usually represented by a unique binary code for each operation. Common opcodes include instructions for arithmetic operations (add, subtract, multiply), data movement (load, store), control flow (branch, jump), and logical operations (AND, OR, NOT).
2. **Operands:** These are the data or the addresses of the data on which the operation specified by the opcode will be performed. Operands provide the necessary input for the instruction. The number of operands and their format depend on the specific instruction and the architecture.

- **Zero Operand:** Some instructions don't require explicit operands, as they operate on data implicitly stored in registers or have predefined behavior.
- **One Operand:** Some instructions have a single operand, often representing a register or a memory address.
- **Two Operands:** Many instructions involve operations on two operands, which could be registers, immediate values, or memory addresses.
- **Three Operands:** In some architectures, particularly those with a more complex instruction set, instructions may involve three operands.

Here's a simplified example in assembly language:

```assembly
ADD  R1, R2, R3
```

- **opcode**: ADD
- **Operands:** R1, R2, R3

In this example, the ADD opcode indicates addition, and the operands are registers R1, R2, and R3. The instruction would add the contents of registers R2 and R3 and store the result in register R1.

Machine Language Instruction Format: 1-Byte, 2-Byte & 3-Byte

The 8085 instruction set is classified into 3 categories by considering the length of the instructions. In 8085, the length is measured in terms of "byte" rather than "word" because 8085 microprocessor has 8-bit data bus. Three types of instruction are: 1-byte instruction, 2-byte instruction, and 3-byte instruction.

1. One-byte instructions –
In 1-byte instruction, the opcode and the operand of an instruction are represented in one byte.

Example-1: Task- Copy the contents of accumulator in register B.
Mnemonic- MOV B, A
Opcode- MOV
Operand- B, A
Hex Code- 47H
Binary code- 0100 0111

Note – The length of these instructions is 8-bit; each requires one memory location. The mnemonic is always followed by a letter (or two letters) representing the registers (such as A, B, C, D, E, H, L and SP).

2. Two-byte instructions –
Two-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next 8 bits indicates the operand.

Example-1: Task- Load the hexadecimal data 32H in the accumulator.
Mnemonic- MVI A, 32H
Opcode- MVI
Operand- A, 32H
Hex Code- 3E
32
Binary code- 0011 1110
0011 0010

Note – This type of instructions need two bytes to store the binary codes. The mnemonic is always followed by 8-bit (byte) data.

3. Three-byte instructions –
Three-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next two bytes specify the 16-bit address. The low-order address is represented in second byte and the high-order address is represented in the third byte.

Example-1: Task- Load contents of memory 2050H in the accumulator.
Mnemonic- LDA 2050H
Opcode- LDA
Operand- 2050H
Hex Code- 3A
50

20
Binary code- 0011 1010
0101 0000
0010 0000

8085 Addressing Modes:

The 8085 microprocessor has several addressing modes that are used to access memory locations. Some of the most commonly used addressing modes in the 8085 microprocessor are:

The way of specifying data to be operated by an instruction is called addressing mode.

Types of addressing modes –
In 8085 microprocessor there are 5 types of addressing modes:

Immediate Addressing Mode –
In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

Examples:
MVI B 45 (move the data 45H immediately to register B)
LXI H 3050 (load the H-L pair with the operand 3050H immediately)
JMP address (jump to the operand address immediately)

Register Addressing Mode –
In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor.

Examples:
MOV A, B (move the contents of register B to register A)
ADD B (add contents of registers A and B and store the result in register A)
INR A (increment the contents of register A by one)

Direct Addressing Mode –
In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.

Examples:
LDA 2050 (load the contents of memory location into accumulator A)
LHLD address (load contents of 16-bit memory location into H-L register pair)
IN 35 (read the data from port whose address is 35)

Register Indirect Addressing Mode –
In register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.

Examples:
MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)
LDAX B (move contents of B-C register to the accumulator)
STAX B (store accumulator contents in memory pointed by register pair B-C)

Implied/Implicit Addressing Mode –
In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.

Examples:
CMA (finds and stores the 1's complement of the contents of accumulator A in A)
RRC (rotate accumulator A right by one bit)
RLC (rotate accumulator A left by one bit)

6.Relative Addressing Mode:

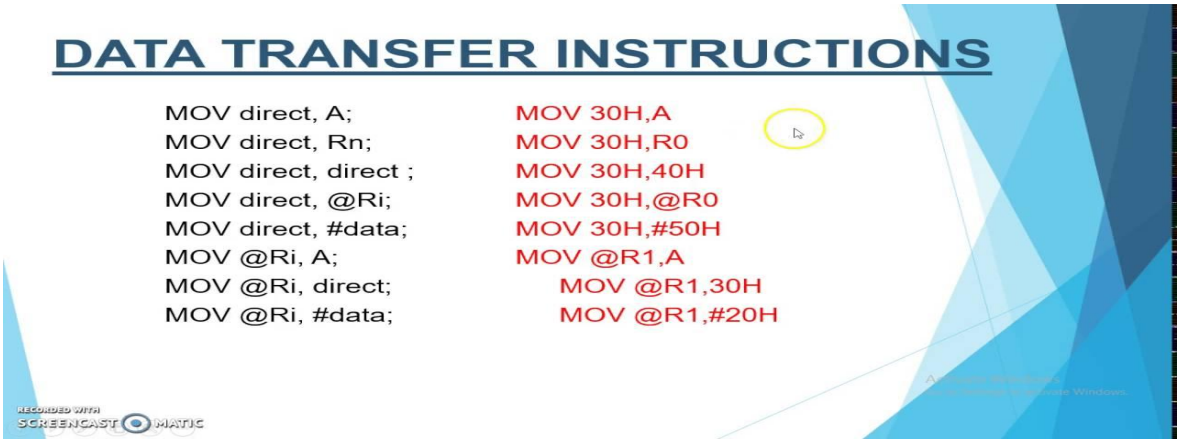
In this mode, the operand is a memory location specified by the contents of the program counter plus a constant value.
example:

```
MOV R0,#05H
AGAIN:
  MVI A,#55H
  ADD A,R0
  JMP AGAIN
```

In this example, the instruction JMP AGAIN uses the Relative Addressing Mode. The instruction jumps to the label AGAIN by adding the contents of the program counter with the specified constant value. The constant value is calculated based on the distance between the current instruction and the label AGAIN.

Data transfer Instructions:

Data transfer instructions are the instructions that transfer data in the microprocessor. They are also called copy instructions. Here is the following is the table showing the list of logical instructions:



Arithmetical Instructions:

Arithmetic Instructions are the instructions which perform basic arithmetic operations such as addition, subtraction and a few more. In 8085 microprocessor, the destination operand is generally the accumulator. Following is the table showing the list of arithmetic instructions:

MAL Arithmetic/Logical Instructions

Instruction		Operation
add	D, S_1, S_2	$D \leftarrow S_1 + S_2$
sub	D, S_1, S_2	$D \leftarrow S_1 - S_2$
mul	D, S_1, S_2	$D \leftarrow S_1 * S_2$
div	D, S_1, S_2	$D \leftarrow S_1$ div S_2
rem	D, S_1, S_2	$D \leftarrow S_1$ rem S_2
and	D, S_1, S_2	$D \leftarrow S_1$ and S_2
or	D, S_1, S_2	$D \leftarrow S_1$ or S_2
xor	D, S_1, S_2	$D \leftarrow S_1$ xor S_2
nor	D, S_1, S_2	$D \leftarrow S_1$ nor S_2
not	D, S_1	$D \leftarrow$ not S_1
move	D, S_1	$D \leftarrow S_1$

1 Add

The content of operand are added to the content of the accumulator and the result is stored in accumulator .

with the addition instruction , the following 3 operations can be done.

1) any 8 bit number can be added to the contents of the accumulator and the result is stored in the accumulator.

2) The contents of a register can be added to the contents of the accumulator and result is stored in the accumulator .

3) The contents of a memory location can be added to the contents of the accumulator and result is stored in accumulator.

this 1 byte instruction.

example – addb c it adds the content of accumulator to the content of the register b

2) SUB

Any 8 bit data or the contents of a register or contents of a memory location can be subtracted from the contents of the accumulator.

with the subtraction instruction the following 3 operator can be done .

1) any 8 bit number can be subtracted from the contents of the accumulator . The result is stored in the accumulator.

In the table, R stands for register M stands for memory Mc stands for memory contents r.p. stands for register pair

Logical Instructions:

Logical instructions in the 8085 microprocessor are a set of instructions that perform logical operations on data in registers and memory. Logical operations are operations that manipulate the bits of data without affecting their numerical value. These operations include AND, OR, XOR, and NOT.

The logical instructions in the 8085 microprocessor include:

ANA – Logical AND: This instruction performs a logical AND operation between the accumulator and a specified register or memory location, and stores the result in the accumulator. For example, the instruction “ANA B” performs a logical AND operation between the contents of the accumulator and the contents of the B register.
ORA – Logical OR: This instruction performs a logical OR operation between the accumulator and a specified register or memory location, and stores the result in the accumulator. For example, the instruction “ORA C” performs a logical OR operation between the contents of the accumulator and the contents of the C register.
XRA – Logical XOR: This instruction performs a logical XOR operation between the accumulator and a specified register or memory location, and stores the result in the accumulator. For example, the instruction “XRA M” performs a logical XOR operation between the contents of the accumulator and the contents of the memory location pointed to by the HL register.
CPL – Logical Complement: This instruction performs a logical complement operation on the contents of the accumulator. This operation flips all the bits of the accumulator, effectively reversing its value.
CMA – Complement Accumulator: This instruction performs a bitwise complement operation on the contents of the accumulator. This operation flips all the bits of the accumulator, effectively reversing its value.

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.

Branching & Looping Instructions:

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

The three types of branching instructions are:

1.Jump (unconditional and conditional)

2.Call (unconditional and conditional)

3.Return (unconditional and conditional)

1. Jump Instructions – The jump instruction transfers the program sequence to the memory address given in the operand based on the specified flag. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Jump Instructions: Transfers the program sequence to the described memory address.
(b) Conditional Jump Instructions: Transfers the program sequence to the described memory address only if the condition in satisfied.

2. Call Instructions – The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

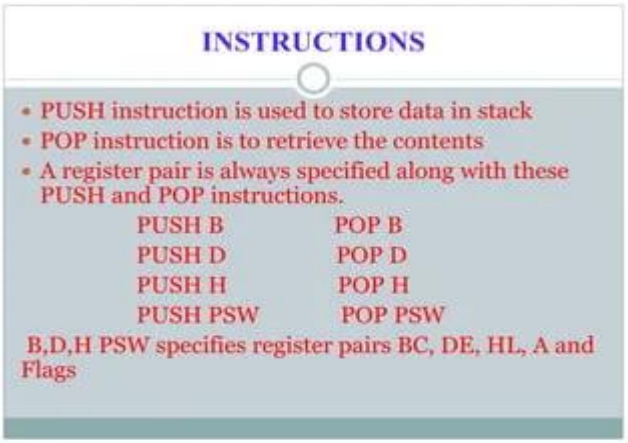
(a) Unconditional Call Instructions: It transfers the program sequence to the memory address given in the operand.
b) Conditional Call Instructions: Only if the condition is satisfied, the instructions executes.

3. Return Instructions – The return instruction transfers the program sequence from the subroutine to the calling program. Return instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program.
(b) Conditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program only is the condition is satisfied.

Stack Instructions:

What is a stack?
A stack is a data structure used to save register contents for later restoration, pass parameters into procedures and save addresses so procedures can return to the right place.



I/O and Machine Control Instructions:

What are the IO instructions in 8085?
In 8085 Instruction set, there are two instructions in 8085 for communication with I/O ports. They are the IN and OUT instructions. The IN or OUT instruction mnemonic should be followed by an 8-bit port address. Thus, we can have 2 8 = 256 input ports and 256 output ports are possible in 8085-based microcomputer.1

Introduction:

Microprocessors are electronic devices that process digital information using instructions stored in memory. Machine control instructions are a type of instruction that control machine functions such as Halt, Interrupt, or do nothing. These instructions alter the different type of operations executed in the processor. In this article, we will discuss the various types of machine control instructions found in microprocessors and their significance in controlling the microprocessor's operations.

Types of Machine Control Instructions:

The following are the types of Machine control instructions:

1. NOP (No operation)
2. HLT (Halt)
3. DI (Disable interrupts)
4. EI (Enable interrupts)
5. SIM (Set interrupt mask)
6. RIM (Reset interrupt mask)

1. NOP (No operation)

Opcode- 00
Operand- None
Length- 1 byte
M-Cycles- 1
T-states- 4
Hex code- 00

2. HLT (Halt)

Opcode- 76
Operand- None
Length- 1 byte
M-Cycles- 2 or more
T-states- 5 or more
Hex code- 76

HLT is used to stop the execution of the program temporarily. The microprocessor finishes executing the current instruction and halts any further execution. The contents of the registers are unaffected during the HLT state. HLT can be used to enter a wait state in which the microprocessor waits for a specific event to occur before resuming execution.

3. DI (Disable interrupts)

Opcode- F3
Operand- None
Length- 1 byte
M-Cycles- 1
T-states- 4
Hex code- F3

DI is used when the execution of a code sequence cannot be interrupted. For example, in critical time delays, this instruction is used at the beginning of the code and the interrupts are enabled at the end of the code. The TRAP interrupt cannot be disabled.

4. EI (Enable interrupts)

Opcode- FB
Operand- None
Length- 1 byte
M-Cycles- 1
T-states- 4
Hex code- FB

EI is used to enable interrupts after a system reset or the acknowledgement of an interrupt. The Interrupt Enable flip-flop is reset, thus disabling the interrupts.

5. SIM (Set interrupt mask)

Opcode- 30
Operand- None
Length- 1 byte
M-Cycles- 1
T-states- 4
Hex code- 30

SIM is used for the implementation of different interrupts of 8085 microprocessor like RST 7.5, 6.5, and 5.5 and also serial data output. It does not affect the TRAP interrupt.

6. RIM (Reset interrupt mask)

Opcode- 20
Operand- None
Length- 1 byte
M-Cycles- 1
T-states- 4
Hex code- 20

RIM is a multipurpose instruction used to read the status of 8085 interrupts 7.5, 6.5, 5.5, and to read serial data input bit.

Classification of 8085 Interrupts and its priorities:

In the 8085 microprocessor, an interrupt is a signal that temporarily suspends the normal execution of a program and redirects the control to a specific interrupt service routine (ISR). Interrupts allow the microprocessor to respond to external events, such as user input, system events, or hardware signals, without the need for constant polling.

Interrupts in 8085. Interrupts are the signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. Vector interrupt − In this type of interrupt, the interrupt address is known to the processor.

8085 Vectored interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5 and RST Instruction :

There are five interrupt signals in the 8085 microprocessor:

1. TRAP: The TRAP interrupt is a non-maskable interrupt that is generated by an external device, such as a power failure or a hardware malfunction. The TRAP interrupt has the highest priority and cannot be disabled.

2. RST 7.5: The RST 7.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the second highest priority.

3. RST 6.5: The RST 6.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the third highest priority.

4. RST 5.5: The RST 5.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the fourth highest priority.

5.INTR: The INTR interrupt is a maskable interrupt that is generated by an external device, such as a keyboard or a mouse. It has the lowest priority and can be disabled.

8085 Non-Vectored Interrupts: INTR:

INTR. It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor. When INTR signal goes high, the following events can occur − The microprocessor checks the status of INTR signal during the execution of each instruction.

Non-Vector interrupt − In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address needs to be sent externally by the device to perform interrupts. For example: INTR.