Sub Code: 18BIT46S

Skill Based Subject – II: MICRO PROCESSOR & ASSEMBLY LANGUAGE PROGRAMMING

UNIT IV: Peripheral devices and their interfacing: Address space partitioning - Memory and I/O interfacing -Data Transfer schemes- Interrupts of Intel 8085.

Prepared by Dr.P.SUMATHI

UNIT IV- PERIPHERAL DEVICES AND THEIR INTERFACING

4.1. Introduction

A microprocessor combined with memory and input/output devices forms a microcomputer. The microprocessor is the heart of a microcomputer. Memories and input/output devices are interfaced to microprocessor to form a microcomputer.

4.1. Address space partitioning

Intel 8085 uses a 16-bit wide address bus or addressing memory and I/O devices. It can access 2^{16}=64k bytes of memory and I/O devices. There are two schemes for the allocation of address to memories or I/O devices.

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

Memory Mapped I/O Scheme

In this scheme there is only one address space. Address space is defined as set of all possible addresses that a microprocessor can generate. Some address are assigned to memories and some address to I/O devices. Suppose memory locations are assigned the address 2000-2500. One address is assigned to each memory location. These addresses cannot be assigned to I/O devices. The addresses assigned to I/O devices are different from address assigned to memory. For example, 2500, 2501, 2502 etc. may be assigned to I/O devices. One address is assigned to each I/O device.

In this scheme all the data transfer instruction of the microprocessor can be used for both memory as well as I/O devices. For example, MOV A, M will be valid for data transfer from the memory location or I/O device whose address is in H-L pair. This scheme is suitable for small system.

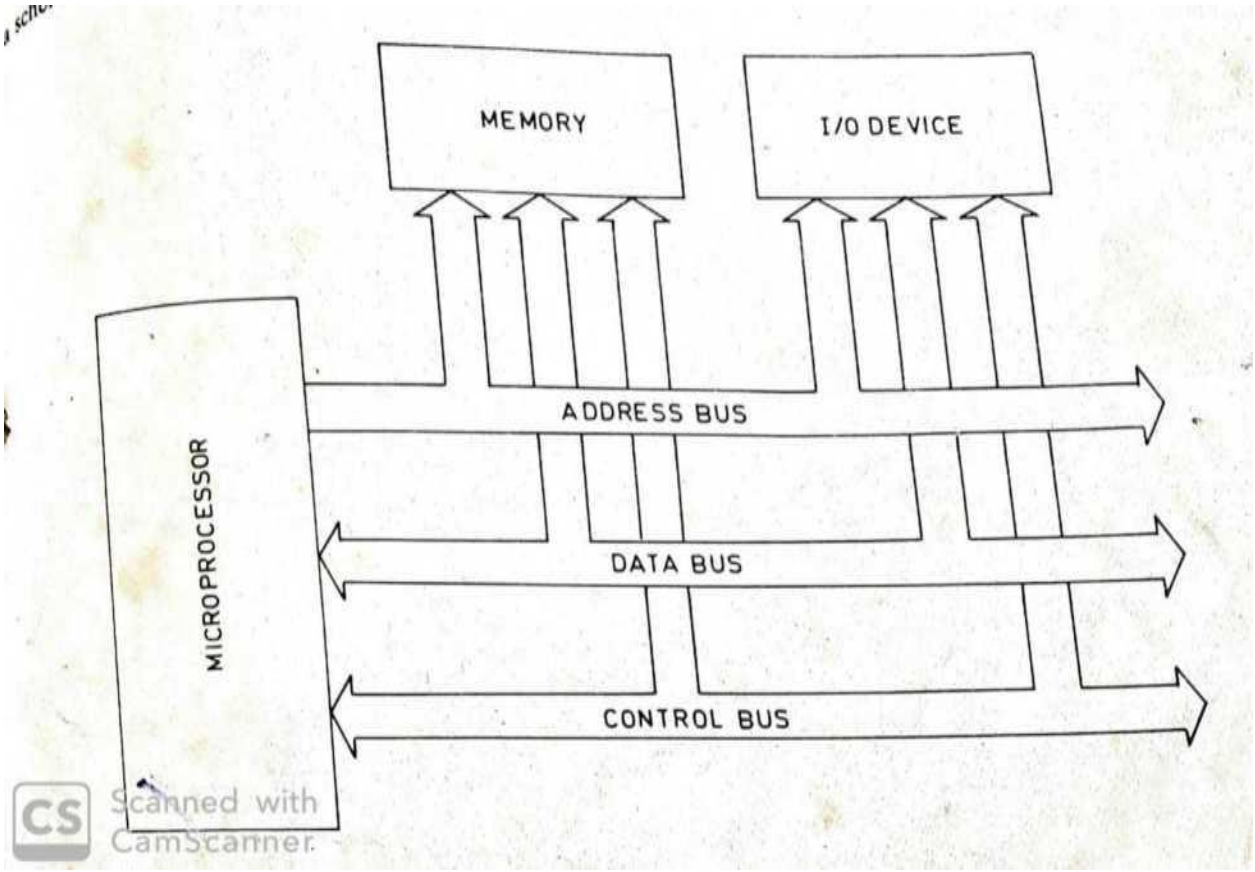
I/O Mapped I/O Scheme

In this scheme the address are assigned to memory locations can also be assigned to I/O devices.

To distinguish whether the address on an address bus is for memory location or I/O devices. The Intel 8085 issues IO/M..... signal for this purpose. When the signal is high the address of an address bus is for I/O device. When low, the address is for a memory location. Two extra instructions IN and OUT are used to address I/O device. The IN instruction is used to read data from an input device. And OUT instruction is used to an output device. This scheme is suitable for large system.

4.2. Memory and I/O Interfacing

An address decoding circuit is employed to select the required I/O device or a memory chip. When IO/M is high, decoder is to active and the require IO device is selected. If IO/M is low, the decoder1 is activated the required memory chip is selected. A few MSB of address line is applied to the decoder to select the memory chip or an I/O device.



4.1. Schematic Diagram for memory and I/O interfacing

4.3. Data Transfer Schemes

The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral. There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed data transfer scheme
2. DMA (Direct Memory Access)

Programmed I/O: It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the

transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.

Example of Programmed I/O: In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility.

i)Synchronous Data Transfer Scheme

ii) Asynchronous Data Transfer Scheme

iii) Interrupt Driven Data Transfer Scheme

Direct Memory Access: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

Types of DMA transfer using DMA controller:

i) Burst Transfer : DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.

ii) Cyclic Stealing :

An alternative method in which DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU delays its operation only for one memory cycle to allow the direct memory I/O transfer to “steal” one memory cycle.

Synchronous Data Transfer

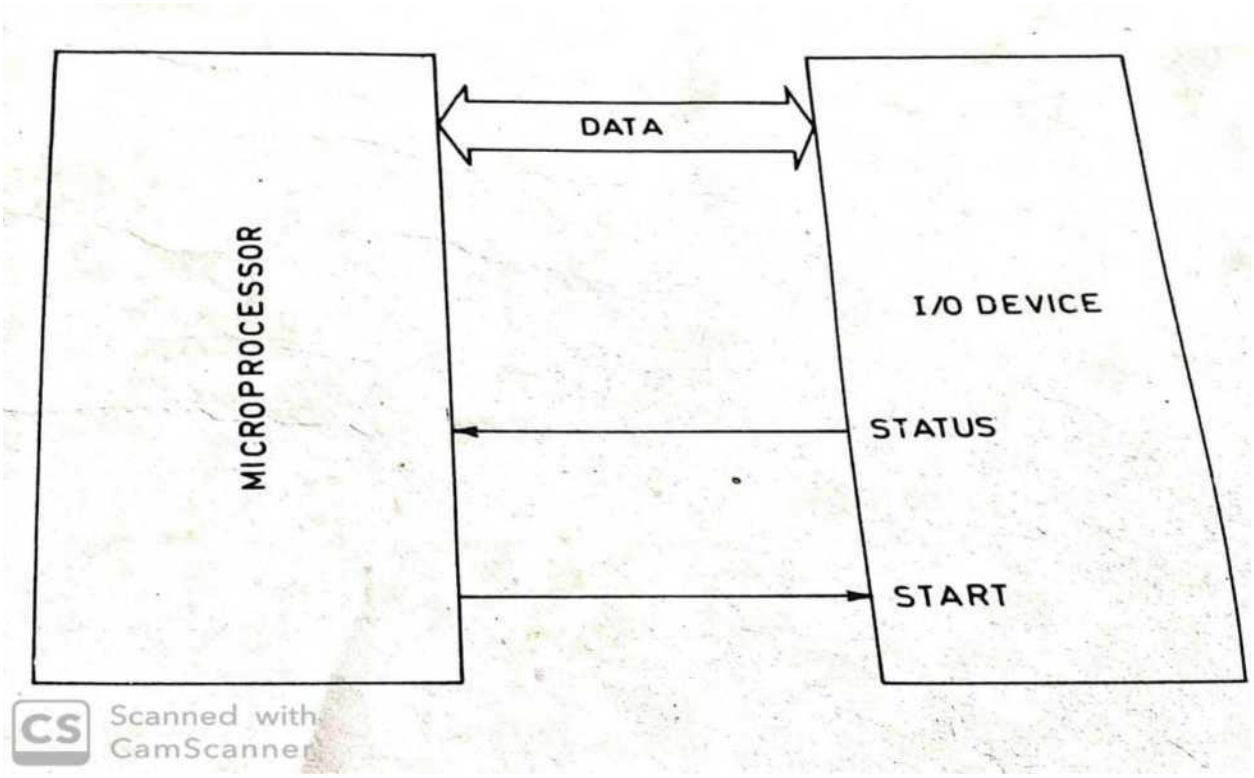
Synchronous means “ at the same time”. The device which spends data and the device which receives data are synchronized with the same clock. When the CPU and I/O devices match in speed this technique of data transfer is employed.

Synchronous transmission is effective, dependable, and often utilised for transmitting a large amount of data. It offers real-time communication between linked devices. An example of synchronous transmission would be the transfer of a large text file. Before the file is transmitted, it is first dissected into blocks of sentences. The blocks are then transferred over the communication link to the target location.

Because there are no beginning and end bits, the data transfer rate is quicker but there’s an increased possibility of errors occurring. Over time, the clocks will get out of sync, and the target device would have the incorrect time, so some bytes could become damaged on account of lost bits. To resolve this issue, it’s necessary to regularly re-synchronise the clocks, as well as to make use of check digits to ensure that the bytes are correctly received and translated.

Asynchronous Data Transfer

In asynchronous transmission, data moves in a half-paired approach, 1 byte or 1 character at a time. It sends the data in a constant current of bytes. The size of a character transmitted is 8 bits, with a parity bit added both at the beginning and at the end, making it a total of 10 bits. It doesn't need a clock for integration—rather, it utilizes the parity bits to tell the receiver how to translate the data.It is straightforward, quick, cost-effective, and doesn't need 2-way communication to function.

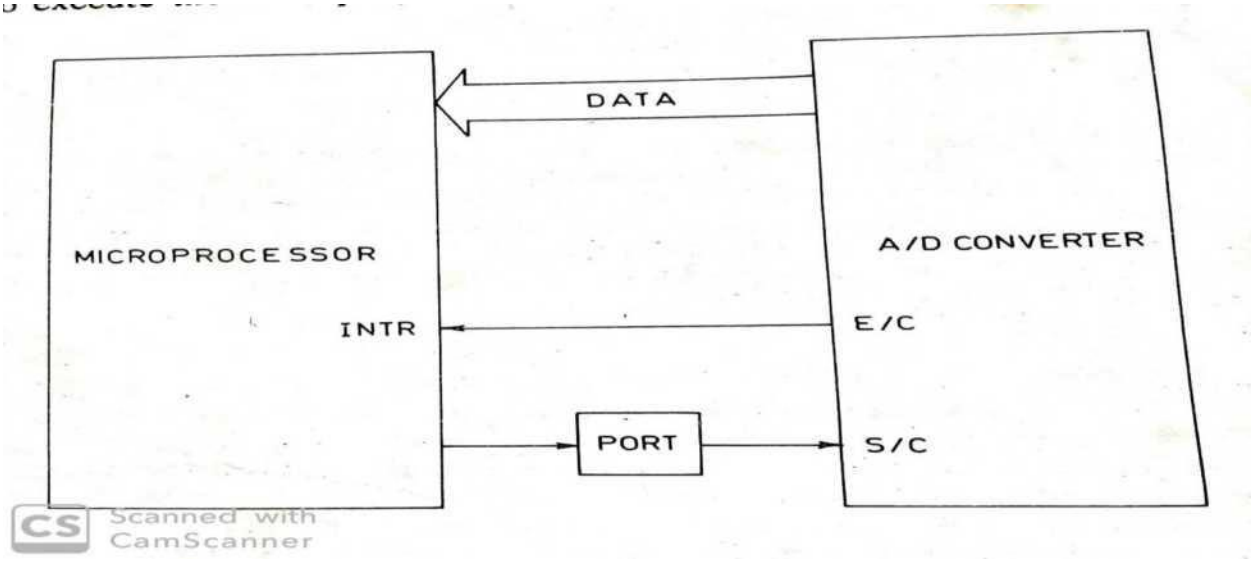


4.2. Asynchronous Data Transfer

Interrupt Driven Data Transfer

In the asynchronous mode of transfer, microprocessor is busy all the time in checking for the availability of data from the slower I/O devices. And it also busy in checking if I/O device is ready for the data transfer or not. In other words in this data transfer scheme, some of the microprocessor time is wasted in waiting while an I/O device is getting ready. To overcome this

problem interrupt driven I/O data transfer introduced. In this interrupt driven I/O data transfer method the I/O device informs the microprocessor for the data transfer whenever the I/O device is ready. This is achieved by interrupting the microprocessor by using the interrupt pins of microprocessor.

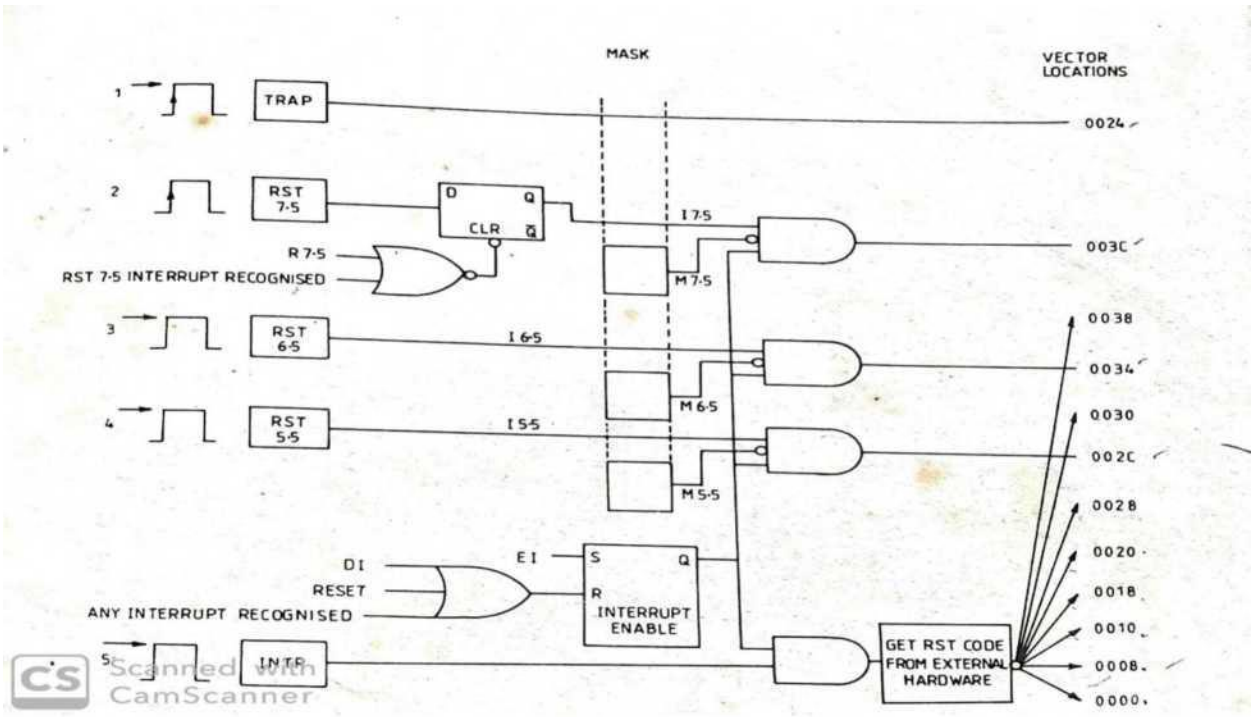


4.3. Interrupt Driven Data Transfer Scheme for an A/D Converter

In the beginning the microprocessor initiates data transfer by requesting the I/O device 'to get ready' and then continue executing its original program rather wasting its time by checking the status of I/O device. Whenever the device is ready to accept or supply data, it informs the processor through a control signal. This control signal known as interrupt (INTR) signal. In response to this interrupt signal, the microprocessor sends back an interrupt acknowledge signal to the I/O device. Interrupts driven data transfer is better from asynchronous mode but it is still not very effective technique when data needs to be transferred in large amounts because it requires an interrupt for every character read or written. This leads us to an another approach called direct memory access(DMA) mode.

4.4. Interrupts of intel 8085

Interrupt is a process where an external device can get the attention of the microprocessor. The process starts from the I/O device. The process is asynchronous, means can occur at any time during execution of program. In order to communicate with pP & I/O devices either Polling or Interrupt method is used. An interrupt is considered to be an emergency signal. The Microprocessor should respond to it as soon as possible.



4.4. Schematic Diagram of 8085 Interrupts

Polling Method in polling, pP polls i.e. ask each device in sequence whether it is ready for communication (data transfer). If device is ready, then data transfer takes place between device & pP. If device is not ready or completed its data transfer, then pP asks the next device in chain. Main disadvantage of this method is that most of the time pP remains busy in polling. So some useful tasks get less time to execute. This method is useful only if pP has contains few I/O devices.

Interrupt is signal send by an external device to the microprocessor to request the processor to perform a particular task or work. It is a simple routine program that keeps a check for the occurrence of the interrupt. Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral (I/O) and the microprocessor. If the pP accept the interrupt and send the INTA (active low) signal to the peripheral. When interrupt is received, pP suspends its current activity and upon completion, it resumes the suspended activity. The processor executes an interrupt service routine (ISR) addressed in program counter. It returned to main program by RET instruction. Advantage is that pP need not waste time in polling the devices.

4.4.1. Interrupt process

When the MPU is executing a program it checks all the interrupt lines during the execution of each instruction.

- If any Interrupt line is enables, the processor completes the current going instruction execution.
- If more than one lines are enabled simultaneously then the processor pick up the request which have the highest priority and all other are discarded.
- After completion of the current instruction execution, processor checks for the respective conditions for the activated interrupt or selected interrupt in case of more than one.
- If condition are not favorable then request is discarded or stored or if the condition are favorable then the processor generates an external INTA or internal acknowledges signal to insert a RST(restart) instruction or the vector location respectively.
- Now the processor save the address of the next instruction (program counter value) on to stack and switch to the related RST location or vector location.
- Service routine written on the location is completed which have RET as its last instruction which returns the program control to the main program by retrieving the return address from the stack.

4.4.2. Types of interrupts

i. S oftware Interrupt
ii. Hardware Interrupt

Software interrupts: It is a instruction based Interrupt which is completely control by software. That means programmer can use this instruction to execute interrupt in main program. There are eight software interrupt available in pP that are RST0 to RST7.

The vector address for these interrupts can be calculate as Interrupt number * 8 = vector address For RST 5 5*8 = 40(in decimal) =28H (in Hexa) Vector address for interrupt RST5 is 0028H. This vector address is stored in Program Counter(PC). These instruction allow transfer of program control from the main program to predefined service routine is also referred to as ISR(Interrupt Service Routine).

Hardware interrupts: This interrupt is caused by sending a signal on one of the interrupt pins of the microprocessor. An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor. If the interrupt is accepted then the process or executes an interrupt service routine (ISR). Hardware interrupt is Asynchronous(it can occur at any time). The 8085 has five hardware interrupts (1)TRAP (2)RST7.5 (3)RST6.5 (4)RST5.5 (5)INTR(address is supplied externally). The hardware interrupts are classified Two types:

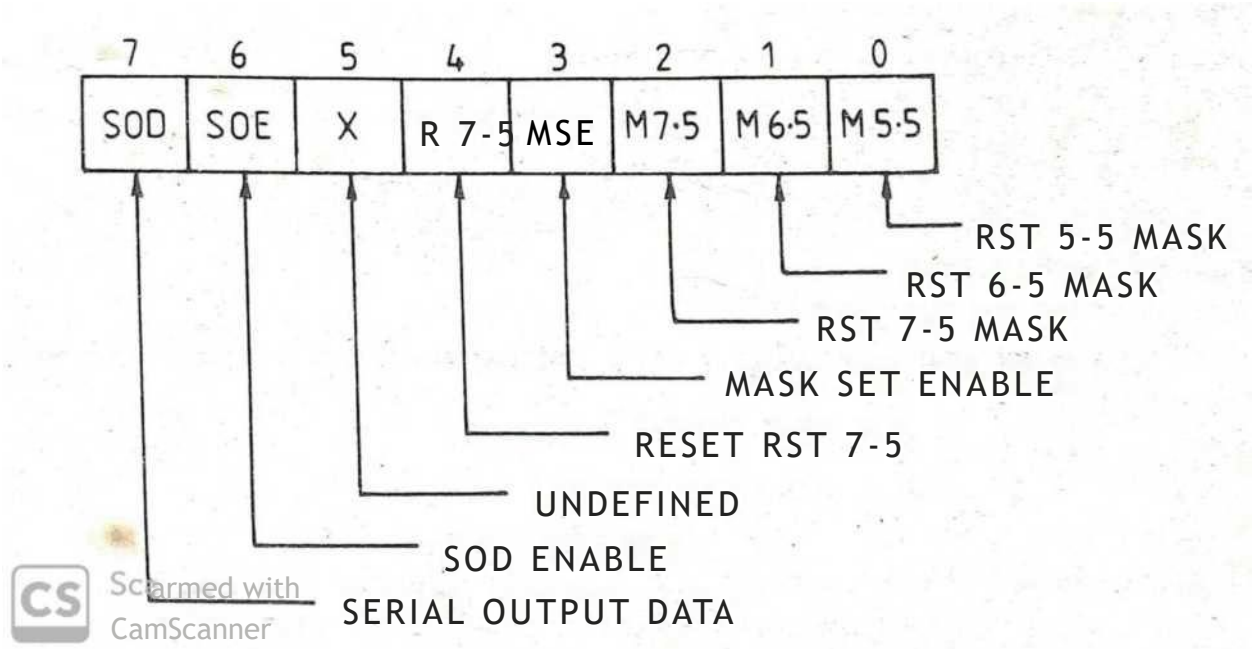
i. Maskable Interrupts
ii. Non-Maskable Interrupts

Maskable interrupts: An interrupt which can be disabled by software that means we can disable the interrupt by sending appropriate instruction, is called a maskable interrupt. RST 7.5, RST 6, RST 5.5 , INT R are the example of Maskable Interrupt.

Non-Maskable interrupts: Cannot disable the interrupt by sending any instruction is called Non Maskable Interrupt. TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognize the interrupt we cannot mask it.

Triggering levels: When a device interrupts, it actually wants the MP to give a service which is equivalent to asking the MP to call a subroutine. This subroutine is called ISR (Interrupt Service Routine). This interrupts can be enable and disable by using EI (enable interrupt) & DI (disable interrupt) instructions. The 'EI' instruction is a one byte instruction and is used to Enable the non-maskable interrupts. The 'DI' instruction is a one byte instruction and is used to Disable the non-maskable interrupts.

Enable Interrupt(EI) The interrupt process is enable by using EI instruction in the main program. It is 1-byte instruction. It enables the interrupt process. Enabling will save the current status and jumps to an interrupt service routine (ISR). After completion it will return back to the main program again.



4.5. Accumulator Content for SIM

Disable Interrupt(DI) This DI instruction is used to disable the interrupt. It is 1-byte instruction. This instruction reset the interrupt enable and disables the interrupt. Both EI & DI are used to enable and disable the interrupts. If the interrupt is masked (disabled), they will not

be recognized by microprocessor. To enable It again they must be unmasked (enabled) by using EI.