Lecture-46

INTERRUPT I/O TRANSFER

In previous lecture, we have discussed how an input device can be interfaced with the processor in polled I/O mode. The status of the device can be checked by inputting one bit. When several input devices are used in a system with programmed I/O, a subroutine checks the ready flag of each device in turn to see which is ready for data transfer or has data available to be read by the microprocessor. This process is known is polling. For example, if 8-input devices are to be interfaced with the processor, their ready flags can be connected to D_7-D_0 bits of the status port as shown in fig.8.1.

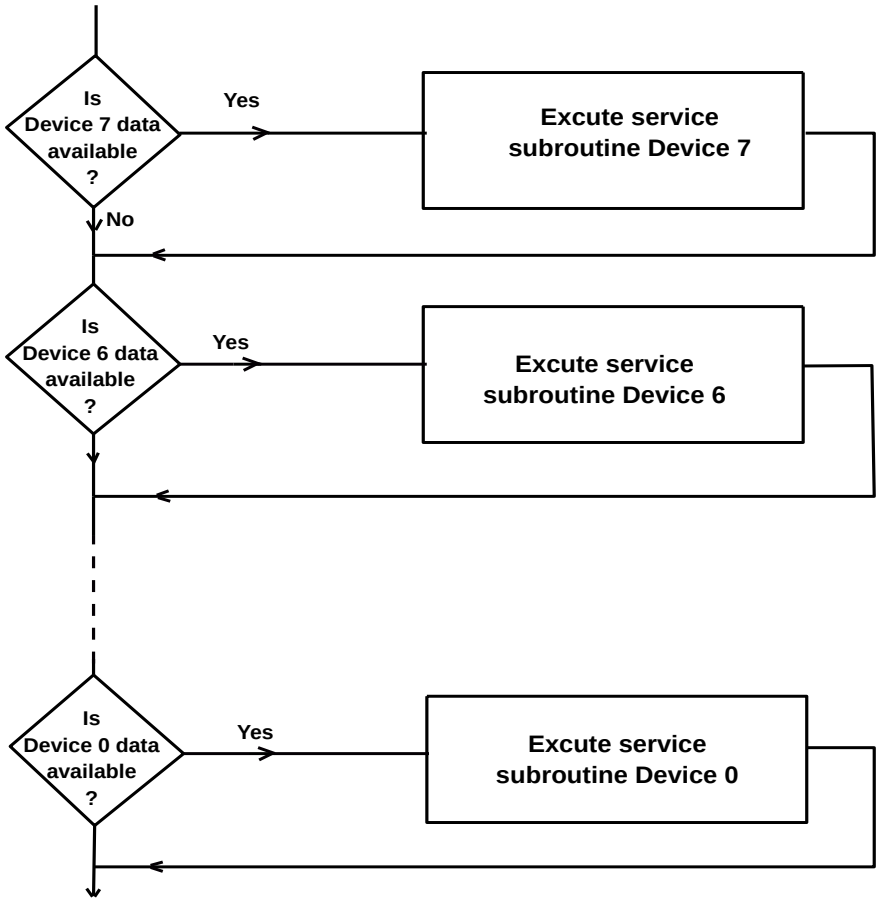

Fig.8.1 Serving 8 Devices in Polled I/O Mode

The polling subroutine checks the status flag of each input device one by one and branches to a service subroutine for the device if its flag is set. It also sets up a priority among the eight input devices by the order in which it tests the service requested bits, with input device ‘7’ having the highest priority. The service subroutine for each input device saves the content of the accumulator (A), then inputs the data byte and stores it in memory or process it.

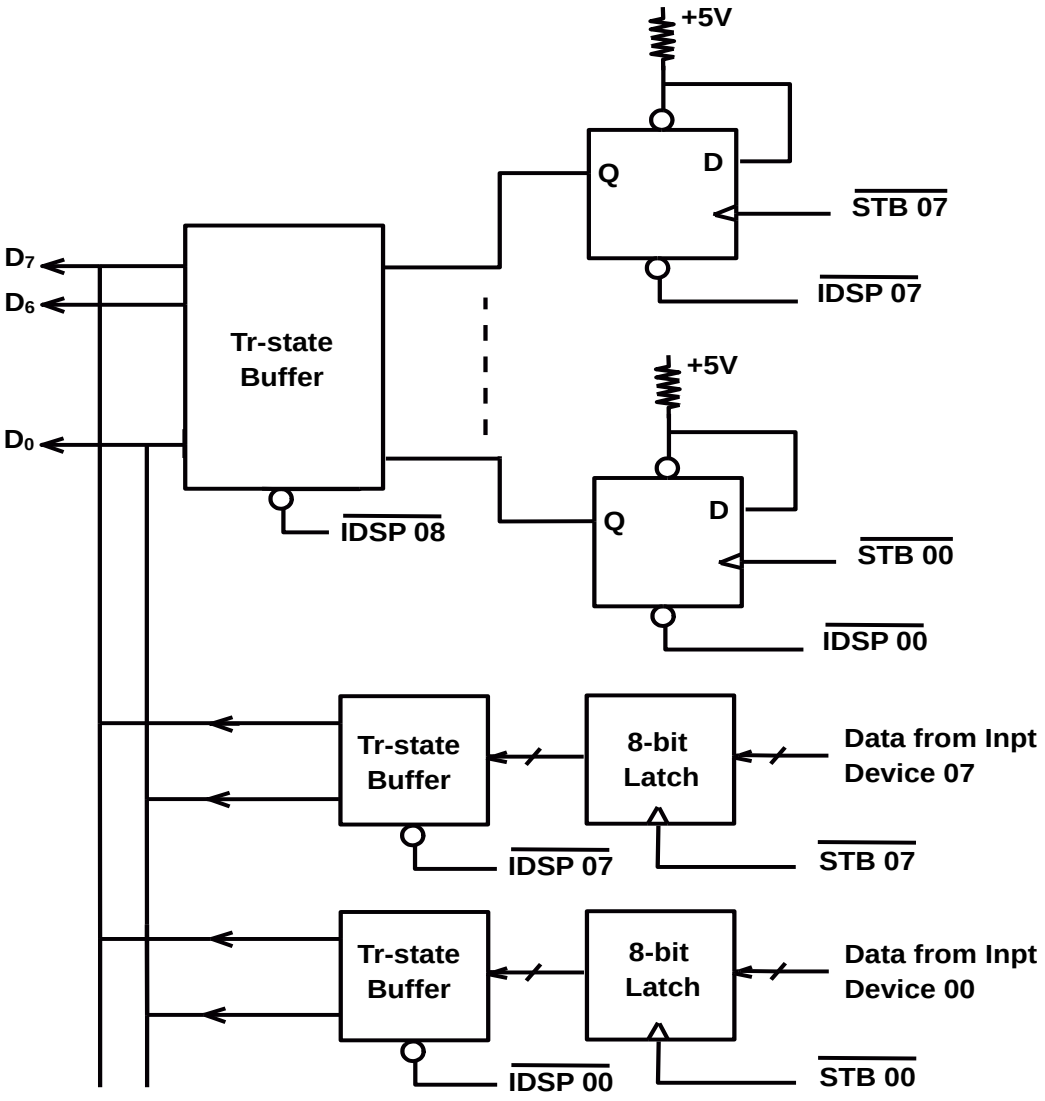

Fig.8.2 Schematic Diagram of Connecting 8 Devices in Polled I/O Mode

Before returning from the service subroutine, accumulator is restored. For output operations, a ready flag in the output device indicates when the device can accept the next data byte. This is necessary when an output device requires time to process the data previously transferred to it before it can accept new data.

```
POLL:   IN    STSTUS
        RAL
        CC    SRV 7
        RAL
        CC    SRV 6
        RAL
        CC    SRV5
        :
        :
        RET
```

If the devices are interfaced as above there are few problems. The processor is all the time busy in checking the readiness of the I/O devices. If the wait away tine is more, the processor is continually busy in checking the readiness of the devices. If there are few time critical operations and need immediate service, they may not be serviced immediately. Further, the sequence in which a device is checked for its readiness assigns the priority to the device and cannot be changed under running condition. To overcome these problems, devices initiated I/O data transfer technique is used.

Interrupt controlled data transfer is a device initiated processor controlled I/O transfer. A μp may be communicating to a number of devices at one or the other time. As discussed above, in case of CPU

initiated polled I/O transfer, polling of I/O service request flags monopolizes a significant amount of a microprocessor time. This reduces system through put the total useful information processed or communicated during a specified time period. Therefore, it is advantageous, in term of increasing throughput as well as reducing program complexity, if an I/O device demands service directly from the microprocessor.

Interrupts provide this capability essentially an interrupt is a subroutine call initiated by external hardware device and is asynchronous, meaning it can be initiated at any time without reference to the system clock. However, the response to an interrupt is directed or controlled by the microprocessor.

A simple structure that allows a single device to interrupt a microprocessor is shown figure.

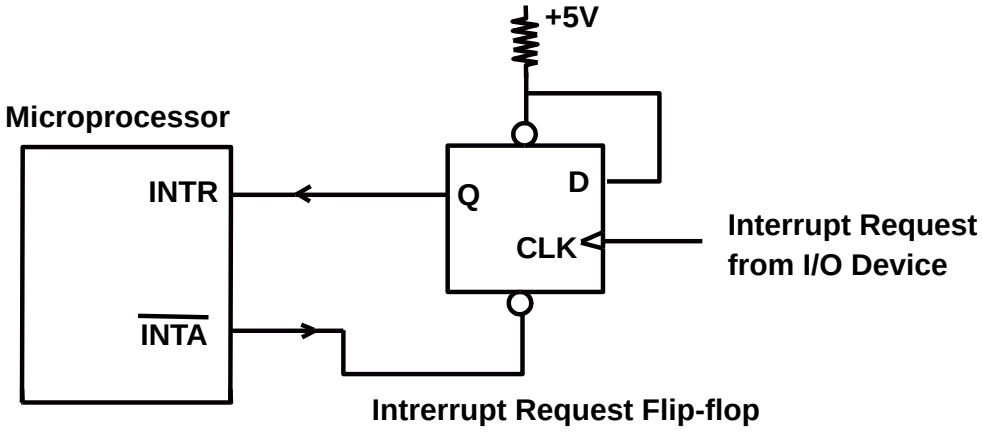

Fig.8.3 Generation of Interrupt by a Device

When an I/O device requires service, it sets its interrupt request flip-flop. This flip-flop is functionally the same as the service request flip-flop except that instead of its output being connected to an input port, it is connected to an interrupt pin of the microprocessor. Thus,

the D-flip-flop stores the I/O device interrupt request until it is acknowledged by the processor.

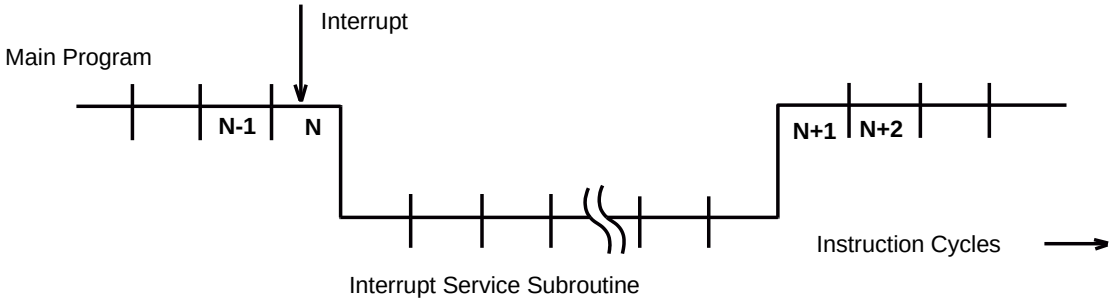

Fig.8.4 Servicing of Interrupt

Since the interrupt request is asynchronous, it may occur at any point in a program's execution. When an interrupt occurs, the execution of the current instruction is completed, the interrupt is acknowledged by the microprocessor and the control is transferred to a subroutine that services the interrupt. When the microprocessor responds to the interrupt; the interrupt request flip-flop is cleared by a signal directly from the processor or by a device select pulse generated by the service subroutine. To resume program execution at the proper point when the I/O service subroutine is finished, the program counter is automatically saved on the top of the stack before control is transferred to the service subroutine. The service subroutine saves the contents of any registers it uses on the stack and restores the register's contents before returning. The contents of the program counter, flag register, accumulator, and general purpose registers together represent the state of the microprocessor.

There are two types of interrupt input - non-maskable and maskable. When a logic signal is applied to a non maskable interrupt input, the microprocessor is immediately interrupted. When a logic

signal is applied to a maskable interrupt input, the microprocessor is interrupted only if that particular interrupt input is enabled. Maskable interrupts are enabled or disabled under program control. If disabled an interrupt request is ignored by the microprocessor.

A non-maskable interrupt input can be masked externally by an interrupt mask signal from an output port. The mask bit from an output port gates the interrupt signal. If an instruction writes a ‘1’ in the mask bit position, the interrupt is enabled; if it writes a ‘0’, it is disabled.

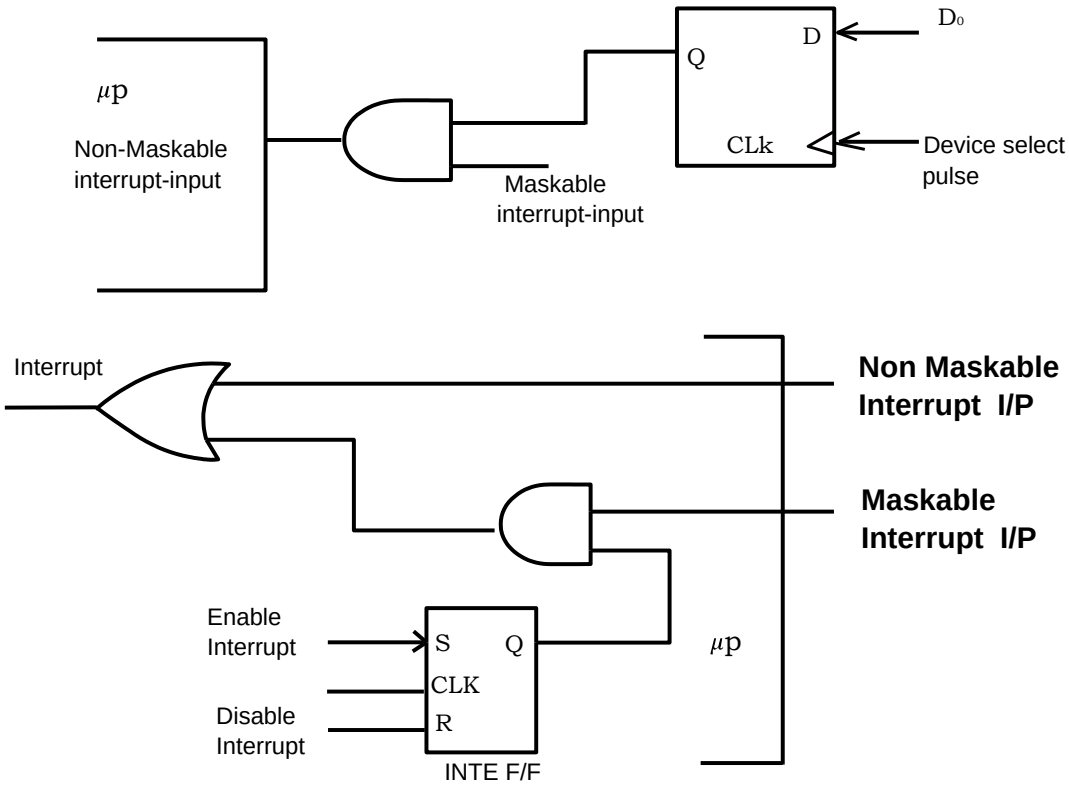

Fig.8.5 Maskable and Non-Maskable Interrupt Inpurs

In response to an interrupt, the following operations occur:

1. The processing of current instruction of the main program is completed.

2. An interrupt instruction cycle is executed during which the program counter is saved and control is transferred to an appropriate memory location.
3. The state of the microprocessor is saved on top of stack.
4. If more than one I/O device is associated with the location transferred to, the highest priority device requesting an interrupt is identified.
5. A subroutine is executed which services the interrupting I/O device. This subroutine clears the interrupt service request flip-flop if it was not cleared in step 2.
6. The saved state of the microprocessor is restored.
7. Control is returned to the instruction that follows the interrupted instruction.

Each step requires a certain amount of time. The combined times for a given microprocessor and external interrupt logic determine how quickly the processor responds to an I/O devices request for service.

Fig.8.6 shows the timing involved in servicing the single interrupt.

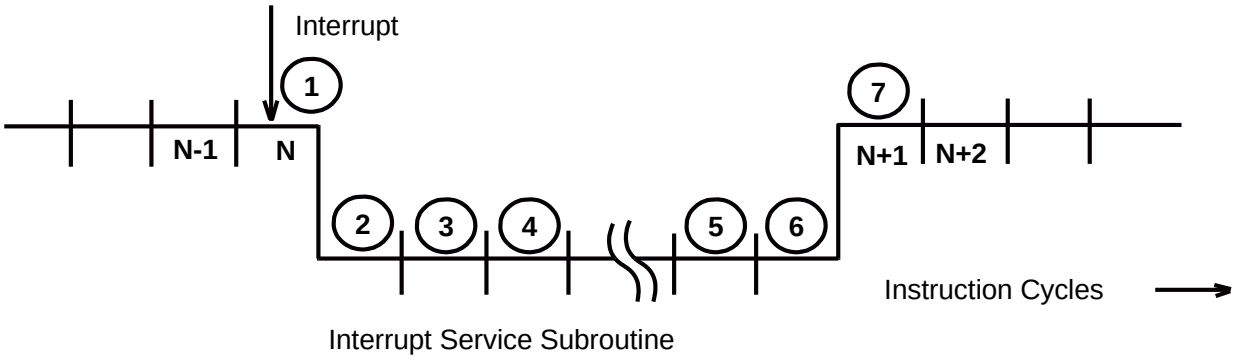

Fig.8.6 Different Operations After Occurance of Interrupt

The time taken in different operations as indicated in the figure are:

(1) Latency time
(2) Interrupt machine cycle time or Bus idle machine cycle time
(3) Saved states
(4) Time taken to identify the device
(5) Actual servicing
(6) Restore states
(7) Return to previous procession of main program.

The time that elapses between the occurrence of the interrupt and the beginning of the execution of the interrupt handling subroutine is the response time, the sum of the times of steps (1) through (4). The difference between the total time that the microprocessor is interrupted and the actual execution time of the service subroutine is referred to as interrupt overhead. Interrupt structures with low overhead allow greater throughput.

Latency time is the time between the occurrence of an interrupt request and the beginning of the interrupt machine cycle.

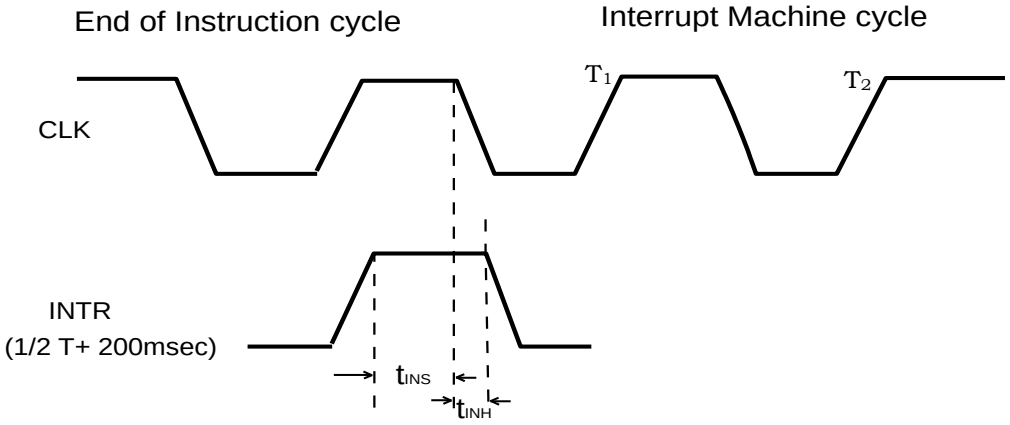

Fig.8.7 Timing Requirement for Acknowledgement of Interrupt

As shown in fig, the interrupt signal must be valid for a time greater or equal to the interrupt set up time t_{INS}, before the falling edge of CLK of the last state of the instruction cycle in order for the next machine cycle to be an interrupt machine cycle. For the Intel 8085A, the minimum value of t_{INS} is 160nsec.

If the interrupt becomes valid precisely, t_{INS} seconds before the beginning of the next machine cycle, then that cycle is an interrupt cycle with a minimum latency time, $t_{LATMIN} = t_{INS}$. If, however, the interrupt signal becomes valid just after this setup time, then it is not responded to until after the next instruction is executed. This provides a worst case latency time of

$$t_{LATMAX} = 160ns + 18T.$$

This relationship assumes there are no WAIT and HOLD states in the instruction cycle during which an interrupt request occurs. A further assumption is, of course, that the interrupt is enabled when the interrupt request occurs.

Lecture-47

INTEL 8085A INTERRUPT STRUCTURE

There are five interrupt inputs TRAP, RST7.5, RST6.5, RST5.5 and INTR. TRAP is a non-maskable interrupt, that is, it cannot be disabled by an instruction. RST7.5, RST6.5, RST5.5 and INTR are maskable interrupts i.e. they can be enabled or disabled through software. The 8085A interrupt structure is shown in fig.8.8.

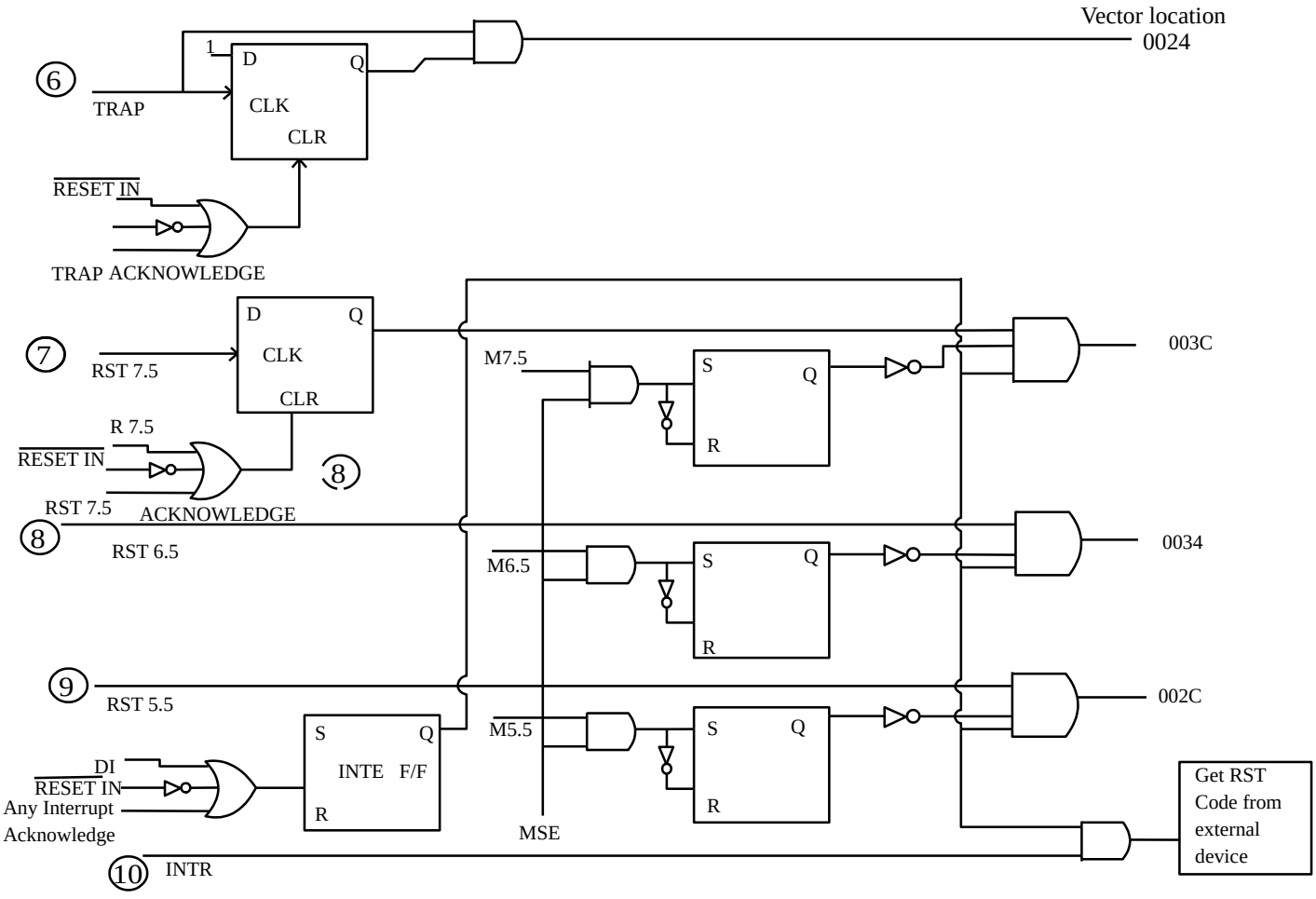

Fig.8.8 Interrupt Structure of Intel 8085A

The following flip-flops are internally provided in the interrupt system of the μp.

1. **R7.5 F/F:** The RST7.5 signal is a LOW to HIGH transition active interrupt control signal input. The LOW to HIGH transition of the signal is registered in R7.5 flip-flop. Thus R7.5 F/F provides a seat for RST7.5 interrupt. It is normally RESET when the power is on. Only LOW to HIGH transition of RST7.5 sets this R7.5 flip-flop to store interrupt. When M7.5 is RESET only then R7.5 signal can interrupt the processor. R7.5 flip-flop can be reset or cleared through the SIM instruction. It is for the user to make use of these facilities.

2. **MSE F/F:** A common chain MASK SET ENABLE (MSE) F/F is provided for all the interrupt masks. This flip-flop must be SET Mask set enable (MSE) flip-flop is also set. This flip-flop can be SET to '1' using SIM instruction for individually enabling or disabling the MASK doors.

3. **INTE F/F:** It is an interrupt enable flip-flop. When the power is turned ON for the first time, $\overline{\text{RESET IN}}$ signal goes LOW. It resets the processor 8085A. It also resets the INTE flip-flop so that the entire interrupt structure is disabled. The INTE F/F can be SET or RESET using instructions. When INTE F/F is RESET, except for TRAP no other interrupt signal can interrupt the processor. When the INTE F/F is SET, the interrupt system is enabled and other interrupt control signals can be selectively enabled or disabled.

4. **INTA F/F**: This is an Interrupt acknowledge flip-flop used only for internal operation by the microprocessor. When first the power is ON this F/F is reset by the $\overline{\text{RESET IN}}$ control signal. Thereafter, whenever a valid interrupt is recognized by the μp it always resets the INTE F/F and then sets the INTA F/F before further action. Thus, further interrupts shall not be recognized, unless, user through instructions in the programme desires further recognition of the interrupt.

5. **MASK F/F (M5.5, M6.5, M7.5):** M7.5, M6.5 and M5.5 are mask flip-flops. These Mask F/Fs are used individually to MASK the interrupts RST5.5, RST6.5 and RST7.5 When these F/Fs are individually SET, then the corresponding interrupt is masked and the interrupt control signal in question can not interrupt the μp. These mask flip-flops are SET to '1' during power ON by $\overline{\text{RESET IN}}$ control signal going LOW. These mask flip-flops can be individually and selectively clear to '0' through SIM instruction (SET INTERRUPT MASK) provided MSE F/F is also SET. MASK SET ENABLE F/F can be SET simultaneously using SIM instruction.

TRAP:

TRAP is a non-maskable vectored interrupt. It can interrupt the μp once the power is ON. Most μp interrupt inputs are level sensitive however, some are edge sensitive and others are both edge and level sensitive. The TRAP input is both edge sensitive and level

sensitive interrupt. It means that TRAP make a low to high transition and remain high until it is acknowledged. The positive edge of the TRAP signal sets the D flip-flop. Because of the AND gate, however the final TRAP also depends on a sustained high level TRAP input. This is why the TRAP is both edge and level sensitive. This also avoids false triggering caused by noise and transients.

For example, suppose the 8085A is midway through an instruction cycle with another 2μsec to completion. If a 300nsec noise spike hits the TRAP input, it will edge triggered but not level trigger the TRAP interrupt because 8085A is still working on the current instruction cycle. Because the TRAP is both edge and level sensitive the 8085A avoids responding to false TRAPs.

Since the TRAP input has the highest priority, it is used for catastrophic events such as power failure, parity errors, and other events that require immediate attention. In the case of brief power failure it may be possible to save critical data. With parity errors, the data may be re-sampled or corrected before going on.

Whenever TRAP comes, μp completes the current instruction, pushes the program counter on the top of the slack and branches to fixed location 0024H. Once the 8085A microprocessor recognizes a TRAP interrupt, it will send a high TRAP ACKNOWLEDGE bit to the TRAP F/F, thus clears the F/F so that even if TRAP remains high it is not recognized again and again. It is further recognized only if it goes low, then high and remains high. The TRAP F/F is also cleared when μp is being reset during which $\overline{\text{RESET IN}}$ goes low and clears the F/F.

RST7.5, RST6.5 & RST5.5:

These are maskable vectored interrupts. These interrupts can be enabled or disabled through software. RST 7.5 has the highest priority among these & RST5.5 has the lowest priority.

RST7.5 control signal input is a rising edge sensitive interrupt. Whenever LOW to HIGH transition occurs, it can interrupt the microprocessor. This LOW to HIGH transition is registered in second D flip-flop. The output of this flip-flop is labeled I7.5. Whenever the other inputs are high the μp recognizes this interrupt. This request is remembered until

1. the processor 8085A responds to the interrupt. When interrupt is acknowledged, it sends a high RST7.5ACKNOWLEDGE bit to the clear input of D flip-flop. This clears it for future interrupts.
2. or until the request is RESET by SIM instruction. R7.5 bit is made high through SIM instruction and the flip-flop can be cleared. It is for the user to make use of these facilities.
3. or until the μp is being reset i.e., $\overline{\text{RESET IN}}$ signal becomes low. Whenever RST7.5 is recognized, control is transferred to 003CH.

RST6.5 & RST5.5 are also vectored and maskable interrupts but are HIGH level sensitive interrupt control signal inputs. These are directly connected to AND gate. The signal at these inputs must be maintained until the interrupt is acknowledged. Whenever RST6.5 is recognized, the control is transferred to 0034H & whenever RST5.5 is recognized, the control is transferred to 002CH.

The internal control signals I7.5, I6.5 & I5.5 are called pending interrupts. The signal IE (output of bottom flip-flop) is called interrupt enable flag. It must be high to activate the AND gates. Also, notice the M7.5, M6.5 & M5.5 signals. They must be low to enable the AND gates. e.g., to activate RST7.5 interrupt, I7.5 must be high, M7.5 must be low and IE must be high.

The interrupt enable flip-flop can be set or reset through software. This flip-flop can be set using EI instruction. EI stands for enable interrupt. Whenever EI is executed, it produces a high EI bit and sets the INTE F/F and produces a high IE output. This flip-flop can be reset in three ways.

1. When the power is ON for the first time or $\overline{\text{RESET IN}}$ signal goes low, it resets the INTE F/F so that that entire interrupt structure is disabled. When INTE F/F is reset except for TRAP no other interrupt can interrupt the μp.
2. The INTE F/F can be reset using DI instruction. DI stands for disable interrupt. When executed it produces a high DI bit & clear the INTE F/F.
3. When the processor 8085A recognizes an interrupt, it produces a high ANY INTERRUPT ACKNOWLEDGE bit. This disables the interrupts.

Because the interrupts are automatically disabled by the ANY INTERRUPT ACKNOWLEDGE bit, programmer usually includes an EI as the next to last instruction in the service subroutine so that the interrupt structure is enabled again. For instance, the last two instructions of interrupt service subroutines typically are

Subroutine: :

:

EI

RET

This subroutine cannot be interrupted (except by a TRAP). After the EI is executed, the processing returns to the main program with the interrupt system enabled.

INTR:

INTR is a maskable interrupt. A high on this pin interrupts the processor. The interrupt signal input INTR is not affected by SIM instruction. Only INTE F/F must be SET to '1' before this interrupt comes.

With the above explanation can write the logic expression for the logic variable, VALID INT

VALID INT = 0 when none of the interrupt control signal input are interrupting the.

VALID INT = 1 when any of the interrupt control signal is active.

Thus, in 8085A, we can write the logical expression for the LOGIC variable VALID INT as below:

VALID INT = TRAP + INTE.[INTR + R75. $\overline{M7.5}$ + RST6.5. $\overline{M6.5}$ + RST5.5. $\overline{M5.5}$]

Lecture-48

Interrupt I/O Transfer

In previous lectures, it was discussed that whenever an interrupt is recognized, the control is transferred to an interrupt service subroutine. If the interrupt acknowledged is a vectored interrupt (TRAP, RST7.5, RST6.5 or RST5.5), the control is transferred to fixed locations depending on the interrupt. If the interrupt acknowledged is non-vectored interrupt (INTR), the address where the control is transferred is provided by the interrupting device. The question is how the control is transferred to these addresses?

Whenever any interrupt is recognized after the execution of the current instruction during which interrupt has occurred, it executes an interrupt machine cycle. If the VALID INT is true due to either TRAP or RST7.5, or RST6.5 or RST5.5 then the interrupt machine cycle executed is BUS IDLE machine cycle of 6 states, during which the μp initially generates the operation code for a restart instruction with the appropriate restart address (e.g. 0024H for TRAP). This OP code is loaded in to the instruction register for execution. The PC is not increment during this BIMC and thus contains the one address of the instruction following the one being executed when the interrupt occurs.

The action of this internally generated instruction is as follows;

RST (internal)

M [(SP)-1] $\leftarrow$ (PCH)

M [(SP)-2] $\leftarrow$ (PCL)

(SP) $\leftarrow$ (SP)-2

(PC) $\leftarrow$ restart address

The Bus Idle machine cycle is identical to OFMC except that the $\overline{RD}$ line remains high during BIMC. The operation code which is generally read in during OFMC is instead generated internally during the BI machine cycle by the microprocessor. Similar to RSTn instruction, after BIMC, two more machine cycles (MWRMC) of 3 states each will be executed to save the content of program counter (PC) on top of stack.

After an RST(internal) is executed, the (PC) contains the address of the starting location for the subroutine that handles the interrupt. This procedure for identifying the interrupting device and directly transferring control to the starting location is called a vectored interrupt. Since only a few memory locations separate the different vector addresses, there is usually a jump instruction at the vector address that transfer control to another memory location where the actual service subroutine begins.

In the case of an INTR interrupt, the interrupt machine cycle entered is an Interrupt Acknowledged machine cycle. INTAMC is also similar to OFMC except that in OFMC processor reads the instruction opcode from program memory and in INTAMC the processor reads the opcode from the interrupting device. Therefore, IO/$\overline{M}$ =1 and instead of $\overline{RD}$ the μp generates an $\overline{INTA}$ strobe during T_2 and T_3 state and the value of the program counter is not incremented during INTAMC. Thus the PC contains the address of the instruction following the one being executed when instruction occurred.

In response to the $\overline{INTA}$ strobe, external logic places an instruction Opcode on the data bus. This opcode may be of either

CALL ADDR or RST-n. CALL ADDR is 3-byte call instruction and RST-n is 1-byte call. CALL ADDR is placed by 8259(PIC). If the CALL ADDR instruction is jammed on the data bus by the external device then three INTA machine cycles will be executed. First one will be of 6 states during which time the operation code. CD_H is placed on the data bus and loaded into the IR. In the subsequent two machine cycles, each of 3-states, the lower order 8-bits of the address and the higher order 8-bits are placed on the data bus and then stored in (Z) and (W) registers, respectively; this address being the starting address of the service subroutine. So to save the return address two more MWRMCs are executed, each of 3 states and the return address from (PC) is saved at the top of stack.

When the OP-code placed on the data bus is RST-n than only one INTAMC is executed of 6 states. The restart instruction RST n has variation from 0 to 7.

RST (internal)

[(SP)-1] ← (PCH)

M [(SP)-2] ← (PCL)

(SP) ← (SP)-2

(PC) ← 8xn

This instruction is essentially the same as the previously mentioned internal restart, except for the restart address, and the fact that it is generated by external hardware. Restart has the following bit pattern, frequently referred to as the restart or interrupt vector.

1	1	N	N	N	1	1	1	N

Where n=NNN is a 3 bit binary number. When this instruction is executed the program counter is saved on the top of stack, thus save the return address. The control is transferred to a location with an address which is 8 times NNN, thus facilitating branch to any one of eight fixed addresses 00H, 08H, 10H, 20H, 28H, 30H, 38H or 3CH depending on the value of NNN these addresses are referred to as restart location, 0, 1, 2.....7.

External logic controls a tri-state buffer with the $\overline{\text{INTA}}$ signal in order to place a restart vector onto the data bus. In fig a single I/O device is connected to the μp interrupt structure. The output of the interrupt request F/F is directly connected to the INTR interrupt pin of the microprocessor. The instruction RST-n can be selected by the 3-toggle switches NNN (000 to 111) whenever the external input device is ready to send the pulse of very short duration. This sets the interrupt request F/F and the INTR signal becomes active. The processor completes the execution of the current instruction and then initiates an interrupt acknowledge machine cycle. During this cycle, the internal INTE F/F is cleared, disabling further interrupt from affecting the μp. The $\overline{\text{INTA}}$ signal that is generated enables the three state buffer and the RST-n instruction opcode is placed on the data bus. $\overline{\text{INTA}}$ also clears the interrupt request flip-flop. The μp inputs the restart vector, saves the program counter and branches to desired memory location. The subroutine that starts at location services the I/O devices.

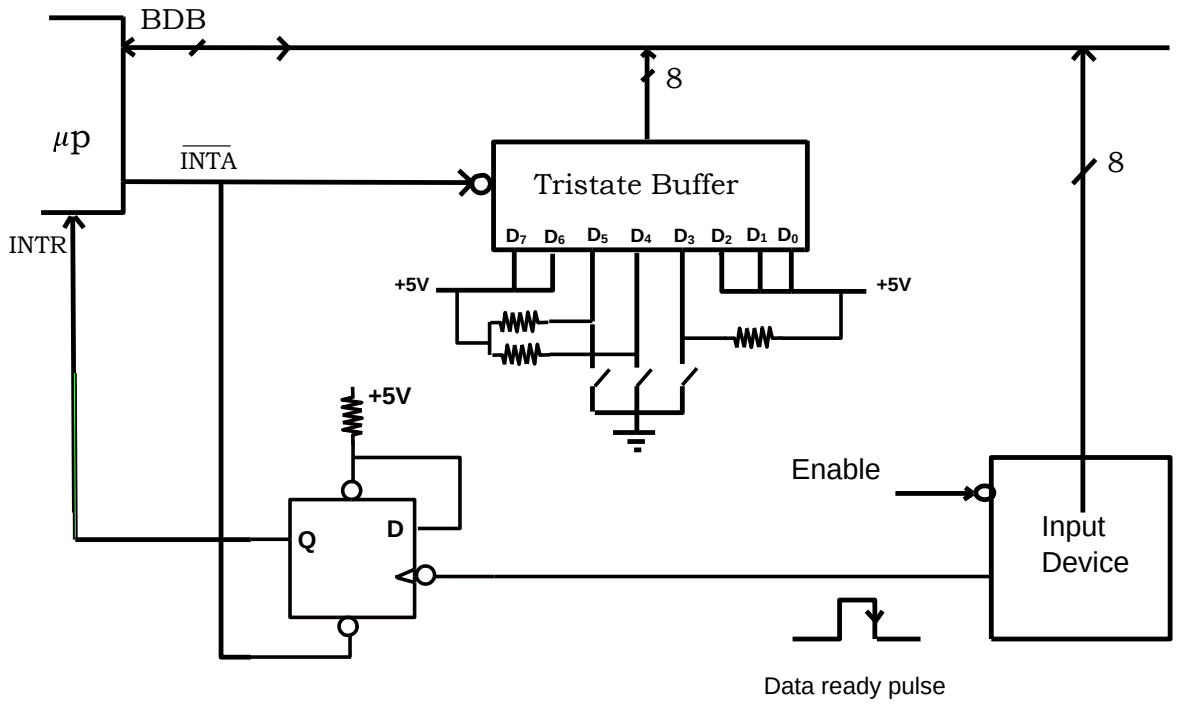

Fig.8.9 Generation of RST-n Instruction Code in Response to Interrupt

Example:

We have already discussed how an ADC can be interfaced with the processor and data transfer takes place in polled I/O mode. Let us interface ADC for data transfer in interrupt mode. The start of conversion pulse (SOC), end of conversion signal (EOC) and outpur enable (OE) signals are as shown below.

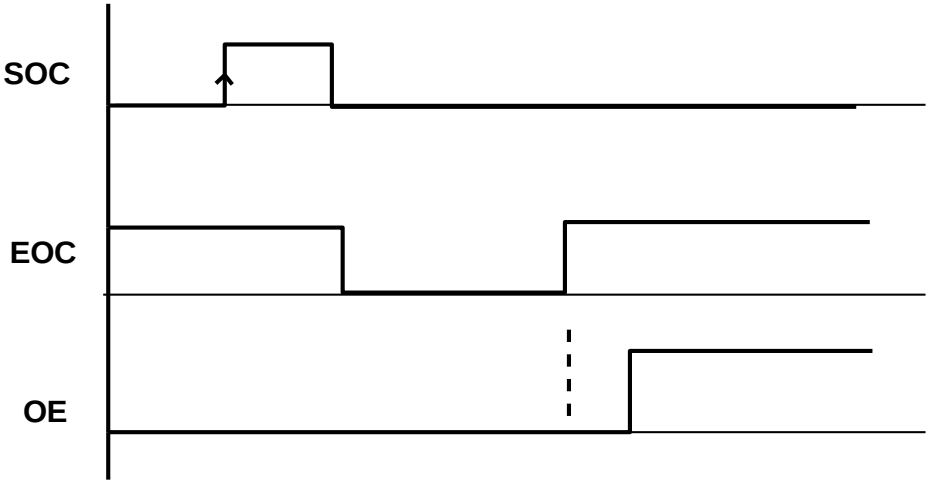

Fig.8.10 Timing Diagram During ADC Operation

The ADC is interfaced to 8085A through port ‘00’ and ‘01’ as shown fig. Port ‘00’ is used to get 8-bit data from ADC and port ‘01’ is used for generating control signals SOC and OE.

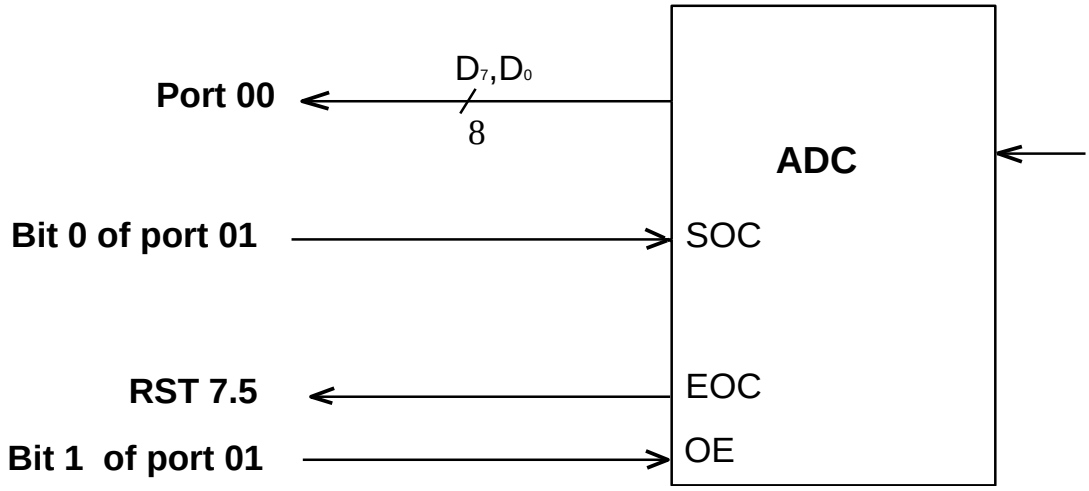

Fig.8.11Interfacing of ADC with Microprocessor Using I/O Ports 00 and 01

End of conversion (EOC) signal output of ADC is connected to RST7.5 interrupt. In the main program start of conversion (SOC) pulse is issued LOW to HIGH to LOW to initiate the data conversion. RST7.5 interrupt is unmasked and interrupt structure is enabled before SOC pulse is issued. The microprocessor enters into the halt state. During conversion EOC is LOW and when the conversion is over, EOC is made HIGH by ADC. This low to high transition of EOC generates RST7.5 interrupt and then microprocessor jumps to interrupt service subroutine (ISS) i.e., (PC) is loaded with 003CH address. In the subroutine the OE signal is issued and data is read via port 00.