Interrupts

Lecture objectives: at the end of this lecture the student will able to:

1- Define the interrupts.

2- Determine the overall structure of interrupts.

3- Explain the operation of control interrupt instructions.

11.1 Introduction

It is necessary executing one of service routines automatically to serve special task for certain conditions. For example, each time when we type a character on a keyboard, a keyboard services routine is ***called***. It transfers the character you typed from keyboard port into processor and then to a data buffer in memory.

The microprocessor have two ways to response any demand service requested by any one of peripherals devices, these two ways are:

Polling way where the microprocessor's software simply checks each of the I/O devices every so often. During this check, the processor tests to see if any device needs servicing. The second way is the interrupt way which it is requested asynchronously by the external devices using hardware. This request would inform the processor to complete whatever instruction that is currently being executed and then fetch a new routine that will service the requesting devices. Once this servicing is completed, the processor would resume exactly where it left off.

11.2 Definition of interrupts:

It is a request introduced for microprocessor by user or peripheral devices by using hardware or software to serve a task through executing certain service routine. It is classified into two types:

A- Single levels interrupts: In single level interrupts there can be many interrupting devices. But all interrupt requests are made via a single input pin of the microprocessor.

B- Multilevel Interrupts: In multilevel interrupts, the I/O devices are tied to the individual interrupt pins of processor. Thus, the interrupts can be immediately identified by the processor upon receiving an interrupt request from it. When the external asynchronous input (interrupt input) is asserted (a signal is sent to the interrupt input), a special sequence in the control logic begins.

1. The processor completes its current instruction. No instruction is cut-off in the middle of its execution.
2. The program counter's current contents are stored on the stack.
3. The PC is loaded with the address of an interrupt service routine.

4. program execution continuous with the instruction taken from the memory location pointed by the new program counter contents.

5. The interrupt program (service routine) continuous to execute until a return instruction is executed.

11.3 Maskable and Non-maskable interrupt

Masking is preventing the interrupt from disturbing the main program. When an interrupt is masked the processor will not accept the interrupt signal. By setting or resetting particular flip-flops in the processor, interrupt can be masked or unmasked, respectively. Also, some of interrupt can be masked by using software where these interrupts called maskable interrupt, while the interrupt that not masked by software is called non-maskable interrupt. In 8085 microprocessor all interrupts except TRAP are maskable interrupts.

11.4 Vectored and non vectored interrupt

In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt. But in non-vectored interrupts the interrupted device should give the address of the interrupt service routine (ISR). All interrupts except restart interrupt are vectored.

11.5 8085 microprocessor interrupts

11.5.1 Types of interrupts

The 8085 has multilevel interrupt system. It supports two types of interrupts:

A-Hardware: Some pins on the 8085 allow peripheral device to interrupt the main program for I/O operations.

B-Software: In software interrupts, the cause of the interrupt is an execution of the instruction. These are special instructions supported by microprocessor.

11.5.2 Overall Interrupt Structure

1. Hardware interrupt in 8085

The 8085 has five hardware interrupts:

a- TRAP **b-** RST 7.5 **c-** RST 6.5 **d-** RST 5.5 **e-** INTR

When any of these pins, except INTR, is active, the internal control circuit of 8085 produce a CALL to a predetermined memory location (***vector location***) and these interrupts called (***vectored interrupts***). The INTR interrupt is not vectored interrupt. It receives the address of the subroutine from external devices. The Fig. 11.1 shows the interrupt structure of 8085. The figure indicates that, the 8085 is designed to respond to edge triggering, level triggering or both.

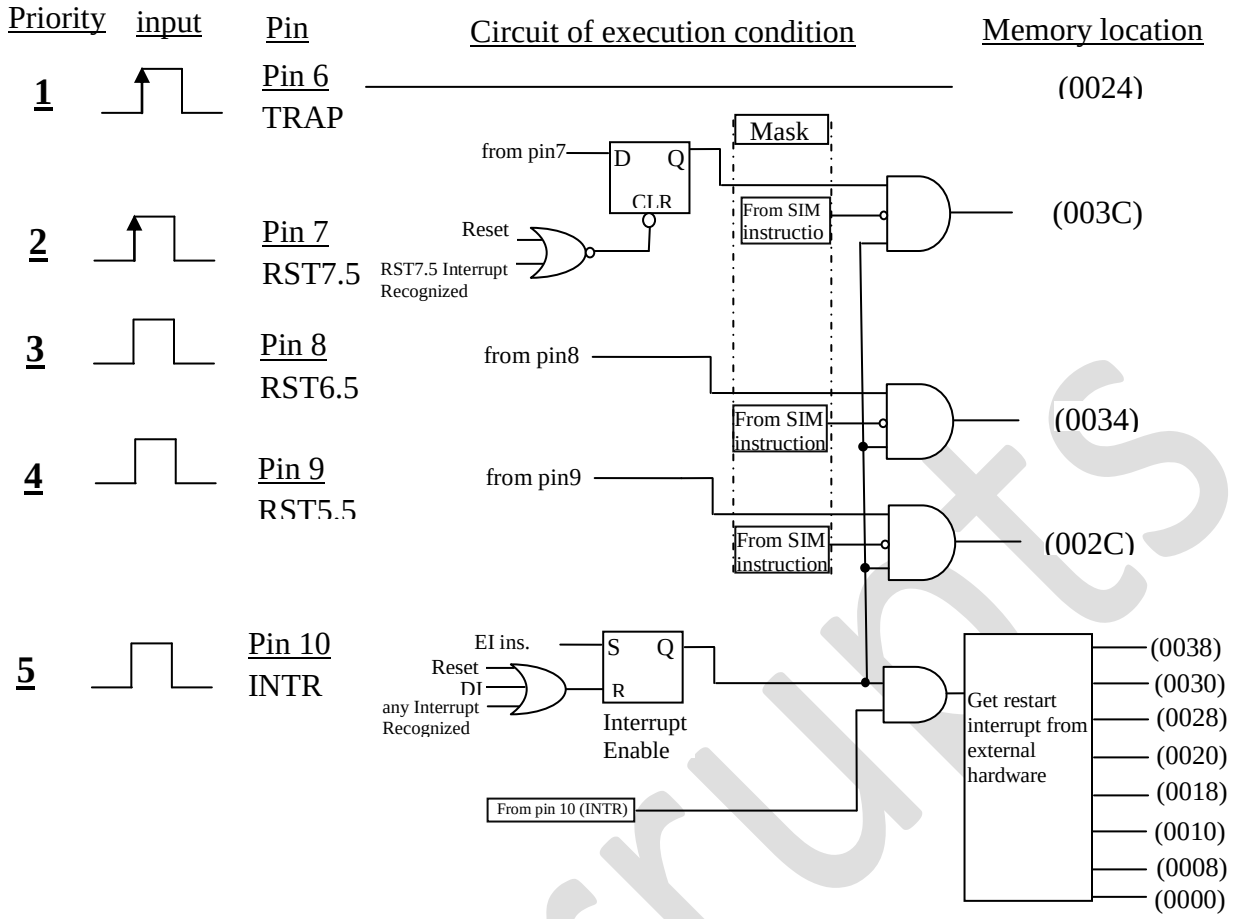


Figure (11.1): 8085 microprocessor interrupt structure

a- TRAP interrupt: it is non masked and vectored interrupt with high priority can execute by supplying ***edge*** and ***level triggered*** on pin 6 of microprocessor, this mean that the TRAP must go high and remain high until it is acknowledged. This avoids false triggering caused by noise and transient (see in Fig. (11-1)). When this interrupt activated, the microprocessor loads PC with vectored address ***0024*** after completes execution of current instruction and store the PC contents in stack memory.

b- RST 7.5 interrupt: it is maskable and vectored interrupt with second priority can executed by satisfying the following conditions, execution **EI**(**E**nable **I**nterrupt) instruction in program ,not mask this interrupt, don't execution **DI**(**D**isable **I**nterrupt) before execution this interrupt, and microprocessor don't execute any other interrupt, before supplying positive edge triggered on pin 7 of microprocessor. When this interrupt activated, the microprocessor loads PC with vectored address ***003C*** after completes execution of current instruction and store the PC contents in stack memory.

c- RST 6.5 interrupt: it is maskable and vectored interrupt with third priority can execute by satisfying the following conditions, execution **EI**(**E**nable **I**nterrupt) instruction in program ,not mask this interrupt, don't execution **DI**(**D**isable **I**nterrupt)

before execution this interrupt, and microprocessor is don't execute any other interrupt, before supplying level triggered signal on pin 8 of microprocessor. When this interrupt activated, the microprocessor loads PC with vectored address ***0034*** after completes execution of current instruction and store the PC contents in stack memory.

d- RST 5.5 interrupt: it is maskable and vectored interrupt with fourth priority can execute by satisfying the following conditions, execution **EI**(**E**nable **I**nterrupt) instruction in program ,not mask this interrupt, don't execution **DI**(**D**isable **I**nterrupt) before execution this interrupt, and microprocessor is don't execute any other interrupt, before supplying level triggered signal on pin 9 of microprocessor. When this interrupt activated, the microprocessor loads PC with vectored address ***002C*** after completes execution of current instruction and store the PC contents in stack memory.

e- INTR: It is a non vectored and maskable interrupt with lowest priority. The following sequence of events occur when INTR signal goes high:

1. The 8085 checks the status of INTR signal during execution of each instruction.
2. if INTR is high, then the processor completes its current instruction and sends an active low interrupt acknowledge signal (***INTA***) if the interrupt is ***enable.***
3. In response to the ***INTA*** signal, external logic circuit places an instruction Opcode on the data bus.
4. On receiving the instruction, the processor transfer (PC) to the stack memory and executes received instruction.

The Fig. 11.2 shows the diagram of external logic that gives the RST 5 instruction Opcode on interrupt acknowledge.

2. Software interrupt in 8085

The 8085 has eight software interrupts from RST0 to RST7. The vector address for these interrupts can be calculated as follow:

Interrupt number ×8=vector address

2×8=16=(10H)

Table 11.1 shows the vectored address of RST interrupt.

Table 11.1: vectors addresses of RST interrupt

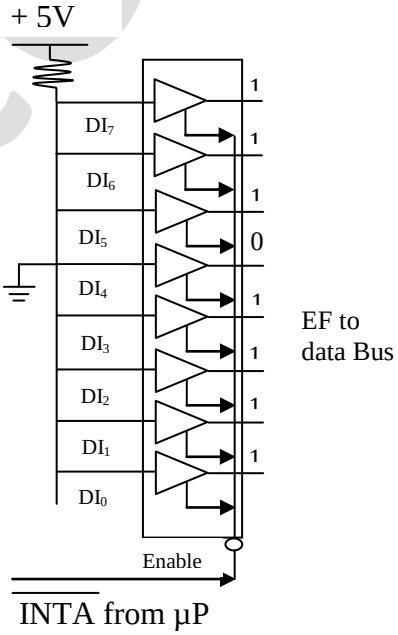
Restart interrupts	Hexadecimal code	Calling location of memory
RST 0	**C7**	**0000**
RST 1	**CF**	**0008**
RST 2	**D7**	**0010**
RST 3	**DF**	**0018**
RST 4	**E7**	**0020**
RST 5	**EF**	**0028**
RST 6	**F7**	**0030**
RST 7	**FF**	**0038**

The RST interrupts can be executed by two methods:

A- By software where can execute any one of RST interrupts by written interrupt as instruction in program, where this instruction will execute in similar way to CALL instruction execution.

B- By external hardware where to execution any one of RST interrupts must insert the code to this RST interrupt to microprocessor by using external hardware and some control signals as shown in Fig.11.2 (6.8 of the book):

The external hardware to RST 5 interrupt is shown in Fig. 11.2, where outputs of try state gate is connect for the data bus. When user wants drive this interrupt, INTR pin must enable with high level signal, then the microprocessor will supply low level signal on INTA pin where this signal supplied to enable of try state gates and this lead to load the states of gats output to the data bus then to microprocessor then execute this interrupt.



Figure(11.2): logic circuit to execution the restart interrupt

By analysis of RST interrupt codes can getting that, three bits (A5, A4, A3) only changed in all codes and other bits are constant as shown in Table 11.2. Therefore, encoder and three tri states gates are used to driven the three changed bits in codes. The priority can getting by encoder where this logic component has priority from high (eighth input) to low (first input).

Table 11.2: RST interrupts codes representation

interrupt	Code (H)	A7	A6	*A5*	*A4*	*A3*	A2	A1	A0
RST0	C7	1	1	***0***	***0***	***0***	1	1	1
RST1	CF	1	1	***0***	***0***	***1***	1	1	1
RST2	D7	1	1	***0***	***1***	***0***	1	1	1
RST3	DF	1	1	***0***	***1***	***1***	1	1	1
RST4	E7	1	1	***1***	***0***	***0***	1	1	1
RST5	EF	1	1	***1***	***0***	***1***	1	1	1
RST6	F7	1	1	***1***	***1***	***0***	1	1	1
RST7	FF	1	1	***1***	***1***	***1***	1	1	1

circuit shown in Fig. 11.3 can used to execution any one from restart interrupt with priority

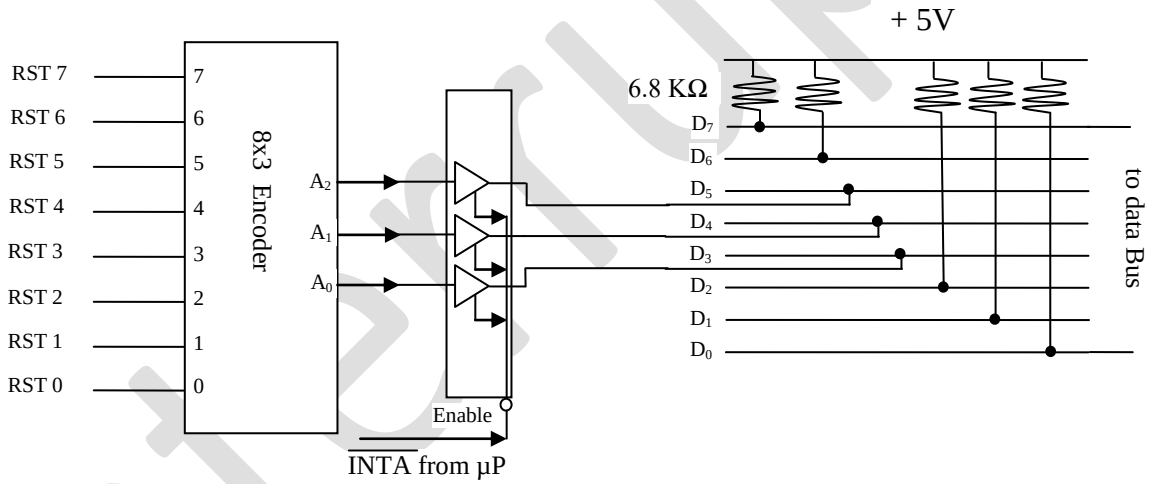


Figure (11.3): Logic circuit to execution the RST interrupt with priority

11.5.3 Masking/Unmasking of Interrupt

Masking and unmasking operations of maskable interrupt using program control are shown in this section. There are four instructions used for control of interrupts:

A. SIM B. RIM C. EI D. DI

A. SIM instruction: (**S**et **I**nterrupt **M**ask) is one byte instruction used to

a- Set mask for RST 7.5, RST 6.5, RST 5.5 interrupts, where this ins. enables or disable the interrupts according to the status of bits D3, D2, D1 and D0 as shown in Fig. (11.4).

b- Reset RST7.5 flip-flop by set D4, see Fig. (11.4).

c- Implement serial I/O of data, see Fig. (11.4).

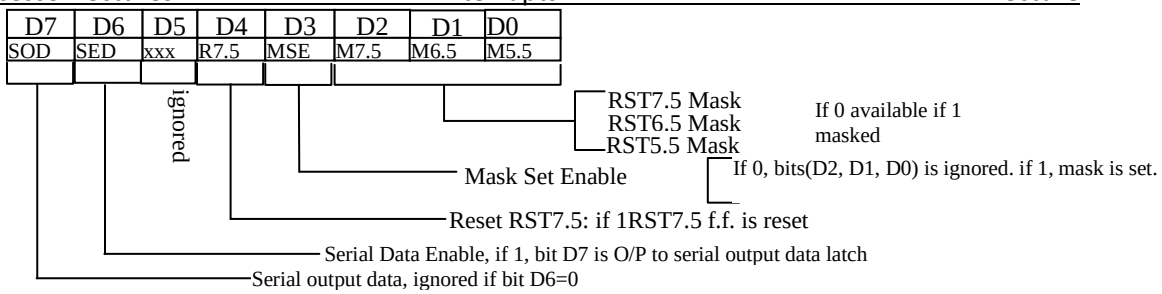


Figure (11.4): operation of SIM instruction

B. Pending interrupts: When one interrupt is being served other interrupts may occur resulting in a pending request. When more than one interrupt occur simultaneously the interrupt with higher priority is served and interrupts with lower priority remain pending. The 8085 has an instruction RIM used by the programmer to know the current status of the pending maskable interrupts.

C. RIM instruction: (**R**ead **I**nterrupt **M**ask) is one byte instruction. RIM instruction is transfer of interrupt control status to accumulator for knowing the interrupts pending. RIM instruction has following functions:

a- Read interrupt mask status and transfer it to accumulator, see Fig.(11.5).

b- Identify the pending interrupts, see fig.(11.5).

c- Receiving data serially, see fig.(11.5).

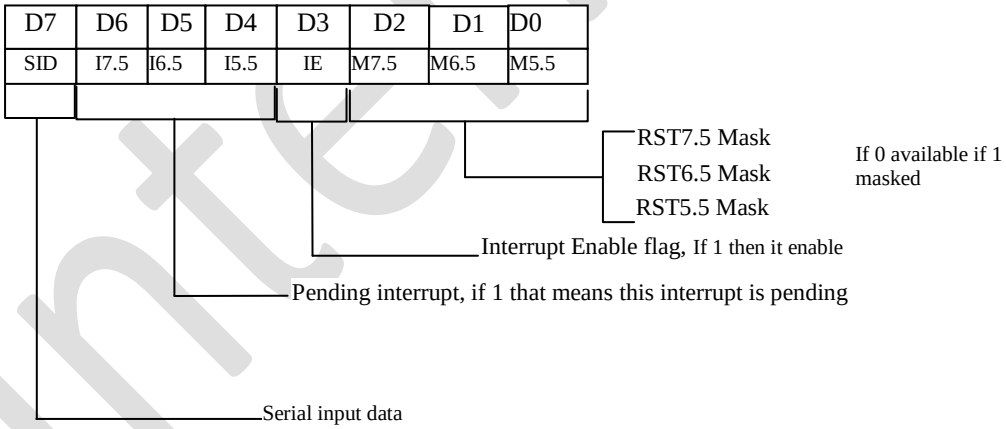


Figure (11.5): Bit Diagram of RIM instruction

All interrupt except TRAP is disabled by resetting the Interrupt Enable F.F., where this F.F. can be reset by one of three methods as below:

a- Execution **DI** (**D**isabled **I**nterrupt) instruction see Fig.(11.1).
b- System reset see Fig.(11.1).
c- Recognition of any interrupt request see Fig.(11.1).

D. EI instruction: (**E**nable **I**nterrupt) is used to enable all types of interrupts where write in beginning of main program. See Fig. (11.5).

E. DI instruction: (**D**isable **I**nterrupt) is used to disable all types of interrupt except TRAP interrupt, This instruction used in beginning of service routine to stopping the execution all interrupt except TRAP interrupt during this service routine. See Fig.(11.5).

Example/ write ALP to display real time clock. Assume that a periodic signal is interrupting RST 7.5 signal after every 0.5 seconds

Solution:

main program

```
        MVI C, 00H
        LXI H. 0000H
        MVI D, 00H
        MVI A, 0BH
        EI
A1: JMP A1
```

ISR - Interrupt Service Routine

```
        INR C
        MOV A,C
        CPI, 02H
        JNZ A2
        MVI C,00H
        MOV A,L
        ADI 01H
        DAA
        MOV L,A
        CPI 3CH
        JNZ A2
        MVI L,00H
        MOV A,H
        ADI 01H
        DAA
        MOV H,A
        CPI 3CH
        JNZ A2
        MVI H, 00H
        MOV A,D
        ADI 01H
        DAA
        MOV D,A
        CPI 18H
        JNZ A2
        MVI D, 00H
A2:     CALL DISPLAY
        EI
        RET
```