8085 Interrupts
Ch-12

Types of Interrupts

- Controlled by the Interrupt Enable flip-flop
  - If flip-flop is enabled and INTR signal to pin-10 goes high, then processor is interrupted
  - Maskable interrupt and can be disabled
  - Non-vectored, address of sub-routine to be supplied manually
- 3 additional vectored-interrupts
  - Address of sub-routine hard-wired
- 1 non-maskable interrupt
- Interrupt process should be enabled and can be disabled
  - Instruction: EI
  - Instruction: DI

Steps

1)Enable interrupts: EI

2)During program execution, it checks INTR line during execution of each instruction

3)If INTR=1, it completes the current instruction, disables interrupts, sends $\overline{\text{INTA}}$. The processor cannot accept any more interrupts until the flip-flop is enabled again

4)$\overline{\text{INTA}}$ is used to insert a RST (or Call instr). It saves return address and transfers control to location on page 00H and restarts execution at that location after executing step-5

5)When it receives RST, it saves address of next instr on stack

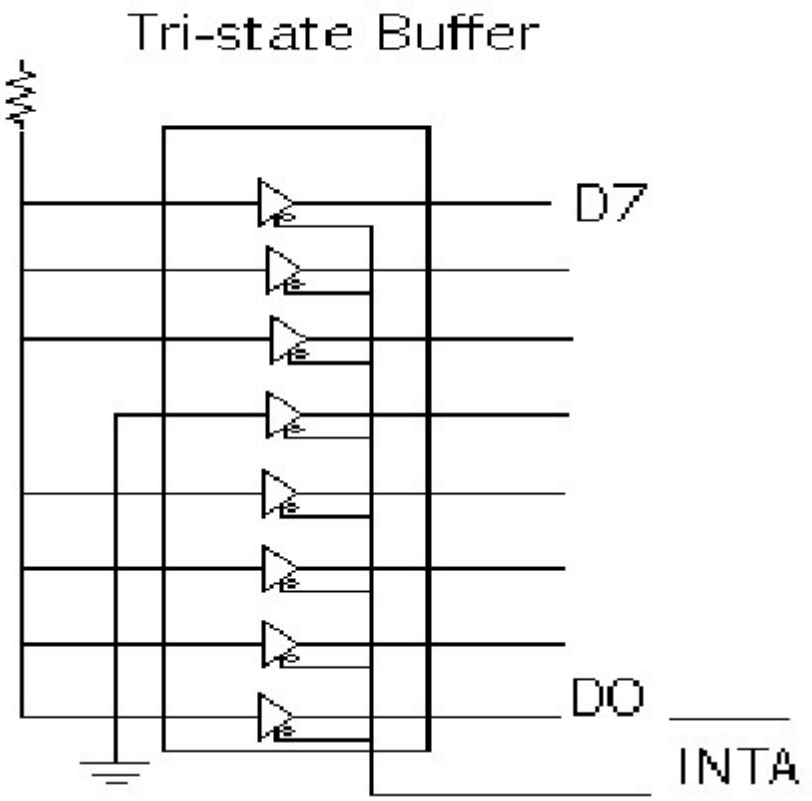
6)Perform the task given in the service routine

7)Service routine should include an EI at the end

8)At end of subroutine, the RET instr returns to the interrupted program and restarts the execution from the proper location

Hardware generation of RST opcode

- Figure shows the generation of RST 5 opcode = EF
- During the interrupt acknowledge machine cycle,(the 1st machine cycle of the RST operation):
  - The Microprocessor activates the INTA signal
  - This signal will enable the Tri-state buffers, which will place the value EFH on the data bus.
  - Therefore, sending the Microprocessor the RST 5 instruction
- The RST 5 instruction is exactly equivalent to CALL 0028H



- RST Mnemonics and call location on page 00H
- The INTA signals enables the tri-state buffer, and the appropriate code for RST comes on the data bus
- This acts as the opcode of the next instruction
- The return location is put on the stack and the program jumps to some location on page 00 H
- At this location another jump must be written to the user defined service routine
- However this is not possible on the kit to modify the content of page 00H

Mnemonics	Hex code	Call location in Hex
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

Vectored interrupts

- TRAP highest priority, non-maskable, always enabled and cannot be disabled
- Level and edge-sensitive
- RST 7.5, 6.5 and 5.5 are enabled under program control using EI and SIM (Set Interrupt Mask) instructions
- SIM
  - Sets mask for RST 7.5, 6.5 and 5.5 using the code word written in the accumulator. D_3=1 means D_{0-2} are effective: 0 = enable and 1 = mask/disable
  - D4=1 means RST 7.5 is reset, overrides D2.
  - D6 = enable for serial I/O and D7=data to transmit in serial I/O

7	6	5	4	3	2	1	0
SOD	SDE	xxx	R7.5	MSE	M7.5	M6.5	M5.5
Serial data	=1 enable serial I/O	Ignored	Reset 7.5	Mask set enable =0: ignore 0-2 bits =1: mask is set		0 = available 1 = masked	

Use of SIM

```
EI            ; enable interrupts
MVI A, 08H    ; load bit pattern to enable 7.5,6.5,5.5
SIM           ; enable 7.5,6.5,5.5
```

```
MVI A, 18H    ; set d4=1
SIM           ; reset 7.5 interrupt flip-flop
```

7	6	5	4	3	2	1	0
SOD	SDE	xxx	R7.5	MSE	M7.5	M6.5	M5.5
Serial data	=1 enable serial I/O	Ignored	Reset 7.5	Mask set enable =0: ignore 0-2 bits =1: mask is set	0 = available 1 = masked		

Pending interrupts

- Several interrupt lines
- While serving one others may come and wait
- To sense these pending interrupts another instruction RIM (Read Interrupt Mask)
  - RIM loads accumulator with 8 bits indicating current status of interrupt masks
  - Bits D4, D5 and D6 identify pending interrupts
  - D7 reserved for serial I/O to receive serial data

7	6	5	4	3	2	1	0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
Serial input data	= 1 pending			Interrupt enable flag 1=enabled	Interrupt masks 1 = masked		

Example

- Assume processor is completing an RST 7.5 request and see if RST 6.5 is pending.
- If pending, enable 6.5 without affecting any other interrupts; else return to main program

```
      RIM        ; read interrupt mask
      MOV B,A    ; save mask info
      ANI 20H    ; check if 6.5 is pending
      JNZ NEXT
      EI
      RET        ; 6.5 is not pending, return to main program
NEXT: MOV A,B    ; get bit pattern, 6.5 is pending
      ANI 0DH    ; enable 6.5 by making D1=0
      ORI 08H    ; enable SIM by making D3=1
      SIM
      JMP SERV   ; jump to service routine for RST 6.5
```

Assignment

- Main program displays 00 in the data field infinitely
- Apply interrupt and write jump (to your service routine) at the given location
- RST 7.5 signal can be driven by either on-board signal KBINT or by off-board signal RST75 via connectors J3 and J4
- When RST 7.5 is recognised, CPU pushes next PC and executes instruction 3CH. The instruction at 3CH is JMP 8FBFH. User has to write another JMP at location 8FBFH to jump to the user service routine. This location may be different for your kits.
- Service routine
  - First occurrence: Start the stop watch (a timer) for period-1
  - Second occurrence:
    - Stop the watch for period-1
    - Compute t1 = time elasped in period-1
    - Display t1
  - Third occurrence: Start the watch for period-2
  - Fourth occurrence:
    - Stop the watch for period-2
    - Compute t2 = time elasped in period-2
    - Display t2
  - Fifth occurence: Display
    - 01 if t1 > t2
    - 02 if t1 <= t2