INTERRUPT SERVICE SUBROUTINE

BY

SUBATHRA S

INTERRUPT SERVICE SUBROUTINE

OBJECTIVE

To write an assembly language program to interrupt the 8085 microprocessor and to execute the interrupt service subroutine.

APPARARTUS REQUIRED

- Single board Microcomputer.
- Power supply.
- Flat Ribbon Cable.
- Bread board
- Resistors
- LED's.
- Wires.

DESCRIPTION

An Interrupt Service Subroutine (ISS) is similar to a procedure that it may be branched from any other program and return branch is made to that program and return branch after interrupt routine has executed.

The interrupt routine must be written so that except for lapse in time the interrupted program proceeds first as if nothing had happened. This means that PSW and registers used by routine must be saved and restored and return must be made to instruction and that which follows last instruction executed before the interrupt.

ALGORITHM

1.Intialize the stack pointer to store PSW.

2.Intialize 8255 in I/O made with all its port as output ports.

3.Using SIM and EI instructions RESET RST7.5 and mask if available.

4.Main program is obituary and here it is counting 00 to FF in an infinite loop.

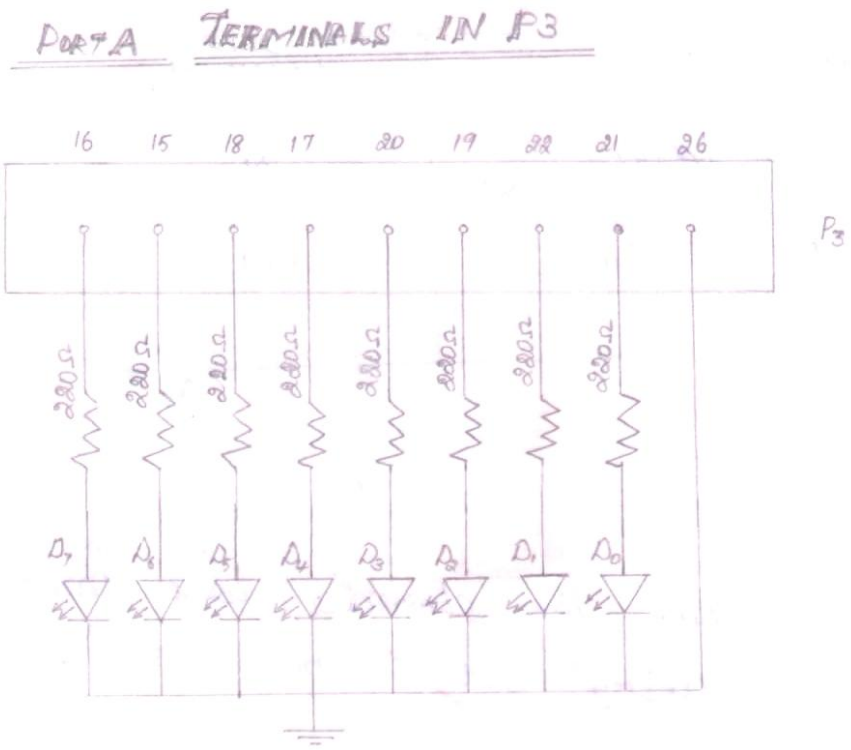
5. Processor states are stored in PC.

6.Suitable delay program is made use or.

7.When interrupted by RST7.5 vector interrupt routine is executed. The PSW is disabled until interrupt routine is completed.

8.Interrupt is enabled and PSW is restored back.

CONNECTION DETAILS



CONNECTION OF FRC TO LED

DATA BITS	**D_7**	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**
PIN NO	16	15	18	17	20	19	22	21

26^{TH} PIN - GROUND

SET INTERRUPT MASKS INSTRUCTION

D_7	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**
SOD	**SDE**	**X**	**R7.5**	**MSE**	**M7.5**	**M6.5**	**M5.5**
Serial O/P Data Ignored If D_6=0	Serial Data Enable 0-disable 1-enable	Ignored	Reset RST7.5 Flip flop 0-not reset 1-reset	Mask Set Enable 0-bits 0 to 2 Ignored 1-mask is set	RST7.5 Mask	RST 6.5 Mask	RST 5.5 Mask

ASSEMBLY LANGUAGE PROGRAM

ADDRESS	LABEL	MNEMONICS	OPCODE/OPERAND
C100		**LXI SP,C500$_H$**	31 00 C5
C103		**MVI A,80$_H$**	3E 80
C105		**OUT CWR**	D3 DB
C107		**MVI A,1B$_H$**	3E 1B
C109		**SIM**	30
C10A		**EI**	FB
C10B		**MVI A,00$_H$**	3E 00
C10D	**DISP**	**OUT PORTA**	D3 D8
C10F		**MOV D,A**	57
C110		**CALL DELAY**	CD 18 C1
C113		**MOV A,D**	7A
C114		**INR A**	3C
C115		**JMP DISP**	C3 0D C1
C118	**DELAY**	**LXI B,FFFF$_H$**	01 FF FF
C11B	**AGAIN**	**DCX B**	0B
C11C		**MOV A,C**	79
C11D		**ORA B**	B0
C11E		**JNZ AGAIN**	C2 1B C1
C121		**RET**	C9
C122	**ISS**	**PUSH PSW**	F5
C123		**DI**	F3
C124		**MVI E,06H**	1E 06
C126		**XRA A**	AF
C127	**NEXT**	**CMA**	2F
C128		**OUT PORTA**	D3 D8
C12A		**MOV M,A**	77
C12B		**CALL DELAY**	CD 18 C1
C12E		**MOV A,M**	7E
C12F		**DCR E**	1D
C130		**JNZ NEXT**	C2 27 C1
C133		**EI**	FB
C134		**POP PSW**	F1
C135		**RET**	C9

PROGRAM TRACE

LABEL	MNEMONICS	DESCRIPTION
	LXI SP,C500$_H$	Initialize the stack pointer. **MEMORY** C4FD XX C4FE XX C4FF XX **C500 XX** ← STACK POINTER C501 XX

MVI A,80_H

Initializing the ports of the PPI 8255 as O/P ports by writing the control word as 80_H.

DATA BITS	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	1	0	0	0	0	0	0	0
COMMENT	I/O mode	Mode0		PortA O/P	PortC Upper O/P	Mode0	PortB O/P	PortC Lower O/P

80_H is moved to accumulator.

REGISTERS

A	**80**	XX	F
B	XX	XX	C
D	XX	XX	E
H	XX	XX	L

OUT CWR

Control word specifies the I/O function for each port of 8255.

MVI A,$1B_H$

$1B_H$ is loaded in to Accumulator.

REGISTERS

A	**1B**	XX	F
B	XX	XX	**C**
D	XX	XX	E
H	XX	XX	L

SIM

Set Interrupt Mask

The execution of the SIM instruction uses the contents of the accumulator (which must be previously loaded) to perform the following function

- Program interrupt mask for the RST 5.5,RST 6.5,RST 7.5 hardware interrupts.
- Reset the edge triggered RST 7.5 input latch.
- Load the SOD output latch.

SIM INSTRUCTION

SOD	**SDE**	**X**	**R7.5**	**MSE**	**M7.5**	**M6.5**	**M5.5**
Serial O/P Data	Serial Data Enable	Undef -ined	Reset RST7.5 Flip Flop	Mask Set Enable	RST7.5 Mask	RST 6.5 Mask	RST 5.5 Mask

When SOD = 0 => data will not be send out from SOD.

SIM INSTRUCTION DESCRIPTION

- Multipurpose instruction.
- To implement the 8085 interrupts (RST 7.5,6.5,5.5).
- Serial data output.

INSTRUCTION BIT			**DESCRIPTION**
D_0	**M5.5**	**1**	RST 5.5 is marked or disabled.

			D_1	**M6.5**	**1**	RST 6.5 is masked or disabled.
			D_2	**M7.5**	**0**	RST 7.5 is enabled.
			D_3	**MSE**	**1**	It enables the function of bits D_2,D_1,D_0 .This is master control over the entire interrupt-masking bit.
			D_4	**R7.5**	**1**	RST 7.5 flip-flop is reset.
			D_5	**X**	**0**	Undefined.
			D_6	**SDE**	**0**	Disable serial output.
			D_7	**SOD**	**0**	Data not latched to peripheral.
	EI	Enable Interrupt The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected.				
	MVI A,00_H	Initialize the accumulator. **REGISTERS** **A** **00** XX F B XX XX C D XX XX E H XX XX L				
DISP	**OUT PORTA**	Output the 00_H configuration through port A.				
	MOV D,A	Move accumulator content to D register. **REGISTERS** A 00 XX F B XX XX C **D** **00** XX E H XX XX L				
	CALL DELAY	Call delay sub program.				
	MOV A,D	D register content 00_H is moved to accumulator. **REGISTERS** **A** **00** XX F B XX XX C D 00 XX E H XX XX L				
	INR A	Increment the accumulator content. **REGISTERS** **A** **01** XX F B XX XX C D 00 XX E H XX XX L				
	JMP DISP	Infinite looping				

DELAY	**LXI B,$FFFF_H$**	Initialize the memory pointer at $FFFF_H$ .i.e. loads the 16-bit data in the register pair designated. **REGISTERS** A 00 XX F **B** **FF** **FF** **C** D 00 XX E H XX XX L

		$C900_H$ is the memory pointer to the starting address of the sine waveform look up table data sequence. **MEMORY** **FFFF** **XX** FFFE XX FFFD XX FFFC XX FFFB XX
AGAIN	**DCX B**	Decrement the BC register pair. **REGISTERS** A 00 XX F **B** **FF** **FE** **C** D 00 XX E H XX XX L
	MOV A,C	The C register content is moved to accumulator. **REGISTERS** **A** **FE** XX F B FF FE C D XX XX E H XX XX L
	ORA B	OR the accumulator content with B register. FE_H => 1111 1110 FF_H => 1111 1111 --- 1111 1111 => FF_H ---
	JNZ AGAIN	Until 00 loop
	RET	Return to main program.

ISS	**PUSH PSW**	Consider that interrupt service is being called at 09 of the accumulator during the main program execution
	DI	The interrupt enable flip-flop is reset and all the interrupt except the TRAP are disabled. No flags are affected. *Comment:* *This instruction is commonly used when the execution of a code sequence cannot be interrupted. For e.g., in critical time delays, the instruction is used at the beginning of the code & the interrupts are enabled at the end of the code. The 8085 TRAP cannot be disabled.*
	MVI E,06_H	E register is initialized. **REGISTERS** A 00 XX F B XX XX C D XX **06** **E** H XX XX L
	XRA A	The content of the operand are exclusive ORed with the content of the accumulator, and the results are placed

		in the accumulator. The content of the operand is not altered. Flags: Z,S,P are altered. CY,AC are reset. **REGISTERS** **A** **00** XX F B XX XX C D XX 06 E H XX XX L XRA A instruction is used for clearing the accumulator to 00 thereby initializing register A.
NEXT	**CMA**	The accumulator content is complemented, no flags are affected. **REGISTERS** **A** **FF** XX F B XX XX C D XX XX E H XX XX L
	OUT PORTA	All LEDs will glow.
	MOV M,A	The accumulator content is moved to memory. **REGISTERS** **A** **FF** XX F B XX XX C D XX 06 E H XX XX L
	CALL DELAY	Make the glowing visible by considerable time delay.
	MOV A,M	The memory content is moved to accumulator. **REGISTERS** **A** **FF** XX F B XX XX C D XX 06 E H XX XX L
	DCR E	The E register content is decremented. **REGISTERS** A FE XX F B XX XX C D XX **05** **E** H XX XX L
	JNZ NEXT	Until 00 the loop continues The LED glows FF & OO alternatively. When E=06 => FF GLOW When E=05 => 00 GLOW When E=04 => FF GLOW When E=03 => 00 GLOW When E=02 => FF GLOW When E=01 => 00 GLOW Thus during ISS, the LEDs glow alternatively when interrupted. Since now the interrupt has been disabled in order to perform this blinking action.

	EI	Enable interrupt
	POP PSW	Now, note that the accumulator is restored with the value 09 , where the interrupt service was called. Hence the main program continues with the next count.
	RET	Return to main program, where vector interrupt was being pressed.

VERIFICATION

During the execution of the program, while the LED's start counting, press the vector interrupt button. Then it goes to ISS that is blinking display. It blinks few times and then starts counting from the previous count.

OBSERVATION

Hardware interrupt = **RST 7.5**
Vector location = **003C $_H$**

In **ROM** at (003C)=**C3 $_H$**
(003D)=**B1 $_H$**
(003E)=**FF $_H$**
=>**JMP FFB1 $_H$**

In **user RAM** at (FFB1)=**C3 $_H$**
(FFB2)=**00 $_H$**
(FFB3)=**C5 $_H$**
=>**JMP C500 $_H$**
In general, **JMP "ISS LOCATION"**

REFERENCE

1. Ramesh S.Gaonkar, Microprocessor Architecture, Programming, and Applications, Fourth Edition, Penram International Publishing (India), 2000.

2. S.Subathra, "Advanced Microprocessor Laboratory", Record work, Adhiparashakthi Engineering College, Melmaruvathur, October 2002

3. S.Subathra, "Programming in 8085 Microprocessor and its applications – An Innovative Analysis", Technical Report, Adhiparashakthi Engineering College, Melmaruvathur, March 2003

4. Micro-85 EB, User Manual, Version – 3.0, CAT #M85 EB-002, VI Microsystems Pvt. Ltd., Chennai.