



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING

WINTER SEMESTER 2022-2023

LAB ASSIGNMENT - 5

Slot: L11 – L12

Class: VL2022230504038

Programme Name & Branch: B. Tech CSE

Course code & Title: BECE204P – Microprocessors and Microcontrollers Lab

Faculty Name: Venu Allapakam

EXPERIMENT – 5 : Interrupt Programming

Aim: To write an ALP in which the interrupt service routine P1.2 is complemented continuously

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  LJMP MAIN
3  ORG 000BH
4  CPL P1.2
5  MOV TLO,#0F0H
6  MOV TH0,#0FFH
7  RETI
8  ORG 0030H
9  MAIN: MOV TMOD,#0000001B
10 MOV TLO,#00H;
11 MOV TH0,#0FFH;
12 MOV IE,#82H
13 SETB TR0
14 HERE: SJMP HERE
15 END

```

Output:

Before Execution & After Execution:

The screenshot displays the Keil IDE interface during the execution of the assembly program. The left pane shows the 'Registers' window with the following values:

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

The right pane shows the 'Disassembly' window with the following code:

```

2: LJMP MAIN
3: ORG 000BH
C:0x0000 020030 LJMP MAIN (C:0030)
C:0x0003 00 NOP
C:0x0004 00 NOP
C:0x0005 00 NOP
C:0x0006 00 NOP
C:0x0007 00 NOP
C:0x0008 00 NOP
C:0x0009 00 NOP
C:0x000A 00 NOP
4: CPL P1.2

```

A 'Parallel Port 1' dialog box is open, showing the port configuration:

Port 1	7	Bits	0
P1: 0xFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Pins: 0xFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The bottom pane shows the 'INTERRUPT.asm' file with the following code:

```

1  ORG 0000H
2  LJMP MAIN
3  ORG 000BH
4  CPL P1.2
5  MOV TLO,#0F0H
6  MOV TH0,#0FFH
7  RETI
8  ORG 0030H
9  MAIN: MOV TMOD,#0000001B

```

The screenshot displays the Keil IDE interface during program execution. The **Registers** window on the left shows the status of various registers, with **sp** (stack pointer) at 0x07 and **sp_max** at 0x09. The **Disassembly** window on the right shows the current instruction at address 0x003E: **SJMP HERE (C:003E)**. A **Parallel Port 1** configuration dialog is open, showing **P1: 0xFB** and **Pins: 0xFB**. The **INTERRUPT.asm** source file is also visible at the bottom.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sp	0x07
sp_max	0x09
PC	C:0x003E
auxr1	0x00
dptr	0x0000
states	13686394
sec	4.97687055
psw	0x00

Address	Disassembly
14: HERE: SJMP HERE	
C:0x003E	80FE SJMP HERE (C:003E)
C:0x0040	00 NOP
C:0x0041	00 NOP
C:0x0042	00 NOP
C:0x0043	00 NOP
C:0x0044	00 NOP
C:0x0045	00 NOP
C:0x0046	00 NOP
C:0x0047	00 NOP
C:0x0048	00 NOP
C:0x0049	00 NOP

```

7  RETI
8  ORG 0030H
9  MAIN: MOV TMOD,#0000001B
10 MOV TLO,#00H;
11 MOV TH0,#0FFH;
12 MOV IE,#82H
13 SETB TR0
14 HERE: SJMP HERE
15 END
  
```

Result:

Hence the program with interrupt service routine that continuously complements the bit P1.2 has been executed successfully.

Program 2:

Aim: To write an 8051 program to get data from P0 and send it to P1 continuously while an interrupt will make timer 0 toggle the P2.1 bit every 100 microseconds

Software Requirement: Keil Software

Program:

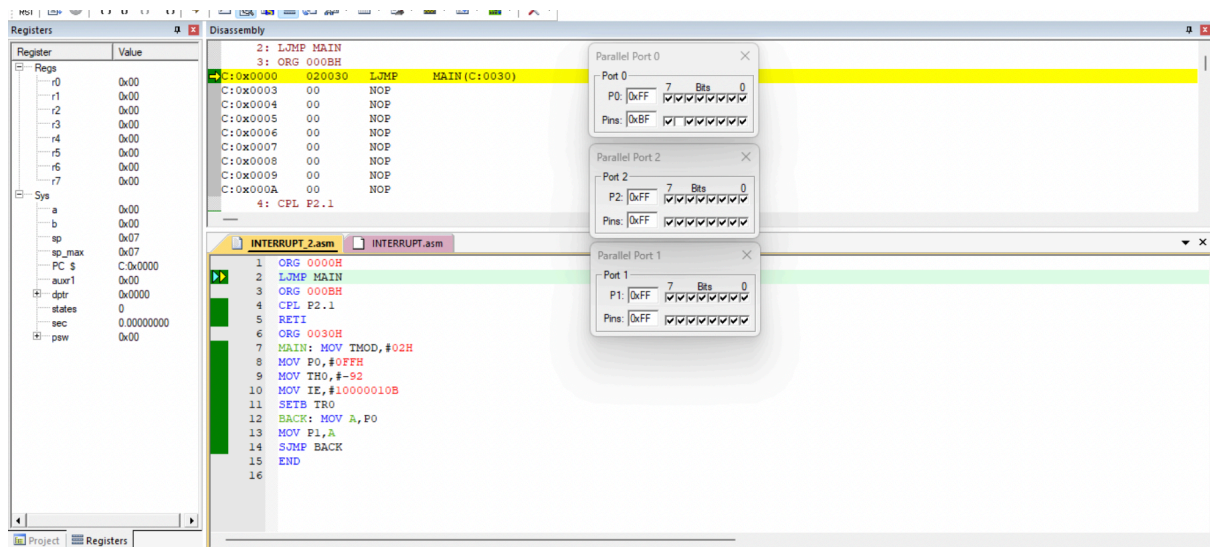
```

1  ORG 0000H
2  LJMP MAIN
3  ORG 000BH
4  CPL P2.1
5  RETI
6  ORG 0030H
7  MAIN: MOV TMOD,#02H
8  MOV P0,#0FFH
9  MOV TH0,#-92
10 MOV IE,#10000010B
11 SETB TR0
12 BACK: MOV A,P0
13 MOV P1,A
14 SJMP BACK
15 END

```

Output:

Before Execution &After Execution:



The screenshot displays a microcontroller development environment with three main panels:

- Registers Panel:** Shows the state of registers. General registers r0-r7 are at 0x00. Special function registers include 'a' at 0xe7, 'sp' at 0x07, 'sp_max' at 0x09, 'PC' at 0x0042, 'auxr1' at 0x00, 'dptr' at 0x0000, 'states' at 24319630, 'sec' at 8.84350182, and 'psw' at 0x00.
- Disassembly Panel:** Shows the disassembled code for the current address range. Key instructions include:
 - 12: BACK: MOV A, P0
 - 13: MOV P1, A
 - 14: SJMP BACK
 - 15: NOP
 - 16: NOP
- Assembly Panel (INTERRUPT_2.asm):** Shows the source assembly code:


```

1  ORG 0000H
2  LJMP MAIN
3  ORG 000BH
4  CPL P2.1
5  RETI
6  ORG 0030H
7  MAIN: MOV TMOD, #02H
8  MOV P0, #0FFH
9  MOV TH0, #-92
10 MOV IE, #10000010B
11 SETB TR0
12 BACK: MOV A, P0
13 MOV P1, A
14 SJMP BACK
15 END
16

```

Three floating windows show the configuration for parallel ports:

- Parallel Port 2:** Port 2 value is 0xFD (bits 7-0: 11111101). Pins are 0xFD (11111101).
- Parallel Port 1:** Port 1 value is 0xE7 (bits 7-0: 11100111). Pins are 0xE7 (11100111).
- Parallel Port 0:** Port 0 value is 0xE7 (bits 7-0: 11100111). Pins are 0xE7 (11100111).

Result:

Hence, the data from P0 was sent to P1 continuously while the interrupt is making timer 0 toggle the P2.1 bit every 100 microseconds.

Program 3:

Aim: To write an ALP to get data from a single bit of P1.2 and send it to P1.7 continuously while a serial service interrupt routine receives data from PC and display it on P2.

Software Requirement: Keil software

Program:

```
1  ORG 0000H
2  LJMP MAIN
3  ORG 0023H
4  LJMP SERIAL
5  ORG 0030H
6  MAIN: SETB P1.2
7  MOV TMOD,#20H
8  MOV TH1,#-3
9  MOV SCON,#50H
10 MOV IE,#10010000B
11 SETB TR1
12 BACK: MOV C,P1.2
13 MOV P1.7,C
14 SJMP BACK
15 SERIAL: JB T1, TRANSMITSG
16 MOV A, SBUF
17 MOV P2,A
18 CLR RI
19 RETI
20 TRANSMITSG: CLR TI
21 RETI
22 END
```

Output:

Before Execution &After Execution:

The screenshot shows the Proteus IDE with the assembly code for the Interrupt 3 routine. The code is displayed in the 'Interrupt_3.asm' window, and the assembly window shows the corresponding machine code. The assembly code includes instructions for setting up the interrupt, clearing the interrupt flag, and returning from the interrupt routine. The machine code window shows the corresponding hex values and assembly instructions.

Assembly Code (Interrupt_3.asm):

```

1  ORG 0000H
2  LJMP MAIN
3  ORG 0023H
4  LJMP SERIAL
5  ORG 0030H
6  MAIN: SETB P1.2
7  MOV TMOD, #20H
8  MOV TH1, #-3
9  MOV SCON, #50H
10 MOV IE, #10100000B
11 SETB TR1

```

Machine Code:

```

14: SJMP BACK
C:0x0044 80FA SJMP BACK(C:0040)
15: SERIAL: JB T1, TRANSMIT3
C:0x0046 209907 JB T1(0x9907) TRANSMIT3(C:004A)
16: MOV A, SBUF
C:0x0049 E599 MOV A, SBUF
17: MOV P2, A
C:0x004B F5A0 MOV P2(0x00), A
18: CLR RI
C:0x004D C298 CLR RI(0x00)
19: RETI
C:0x004F 32 RETI

```

Hence, the data is being continuously sent from P1.2 to P1.7 and during the interrupt, the data is sent from PC to P2.