LAB ASSIGNMENT-4

Registration No.	17BEC0656		
Name	Sparsh Arya		
Course Code	ECE3003	Lab Slot	L49-50
Course Name	Microcontrollers and its Applications		
Project Title	SERIAL TRANSMISSION		
Date of Experiment	18-10-19	Date of Submission	18-10-19
Faculty	Karthikeyan A.		

Aim:

Write an 8051 assembly program to transfer data serially at baud rate 9600 with 8 bit data, one stop bit and observe the transmitted data in the serial window of the simulator.

Algorithm:

1. Store the required data in MYDATA labelled location.
2. Move TH1 to be -3, in order to set the required baud rate of 9600.
3. Use an Accumulator to access the contents stored at MYDATA and store it in SBUF.
4. Display the contents onto the UART window.
5. Observe the outputs.
6. Verify the analysis, with data stored in MYDATA labelled location.

Program

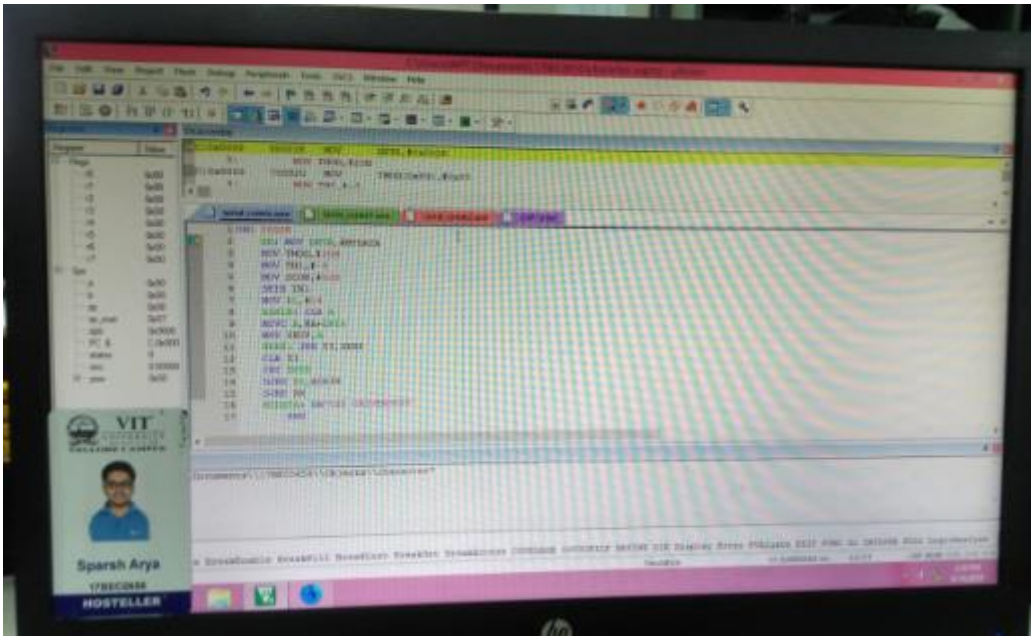
Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	ORG 0000H	-	-	-	-	-	A←0	-
-	-	XX: MOV DPTR,#MYDATA	DPTR	Immediate	2	3	Data Transfer	DPTR←SP	-
-	-	MOV TMOD,#20H	TMOD	Immediate	1	2	TIMER	TIMER 1 MODE 2	M1←1 M0←0
-	-	MOV TH1,#-3	TH1	Immediate	1	2	Baud Rate	Baud rate←9600	-
-	-	MOV SCON,#50H	SCON	Immediate	1	2	Data Transfer	SCON←10 010000B	-
-	-	SETB TR1	TR1	-	1	1	TIMER	Start Timer 1	-
-	-	MOV R1,#14	R1	Immediate	1	2	Data Transfer	R1←14	-
-	-	AGAIN: CLR A	A	-	1	2	Bit Manipulation	A←0	-
-	-	MOVC A,@A+DPTR	A	Index	2	1	Data Transfer	A←A+ DPTR	-
-	-	MOV SBUF,A	SBUF,A	Indirect	1	1	Data Transfer	SBUF←A	-
-	-	HERE: JNB TI,HERE	TI	-	2	3	Bit Manipulation	Jump if no bit in TI	TI←1
-	-	CLR TI	TI	-	1	1	Bit Manipulation	TI←0	-
-	-	INC DPTR	DPTR	-	1	1	Arithmetic	DPTR← DPTR+1	-

-	-	DJNZ R1,AGAIN	R1	-	2	3	Branch	Decrement R1 and jump if R1 not zero	-
-	-	SJMP XX	-	-	2	2	Branch	Short jump to loop XX	-
-	-	MYDATA: DB 'VIT UNIVERSITY'	-	-	1	1	-	“VIT UNIVERSITY” is saved in DB	-
-	-	END	-	-	-	-	-	END	-

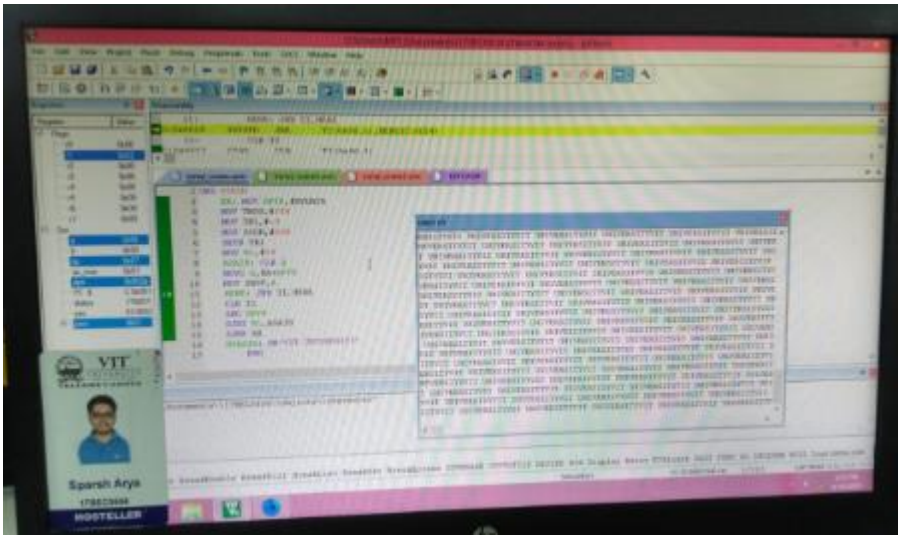
OUTPUTS:

OBSERVATIONS:

- BEFORE EXECUTION



- AFTER EXECUTION



INFERENCE

VIT UNIVERSITY is displayed in the UART window serially with a baud rate of 9800.

RESULTS:

The 8051 ALP to transfer data serially was performed and the output was verified.

Aim:

Write a 8051 Assembly Language program to get data from the PC and display it on P1. Assume 8051 is connrdected to PC and observe the incoming characters. As you press a key on the PC's keyboa, the character is sent to the 8051 serially at 4800 baud rate and is displayed on LEDs. The characters displayed on LEDs are in ASCII (binary).

Assume that the 8051 serial port is connected to the COM port of IBM PC, P1 and P2 of the 8051 are connected to LEDs and switches, respectively.

Algorithm:

1. Input is taken from the UART WINDOW.
2. The Baud rate is set to 4800 by assigning the value of -6 in TH1
3. Use Timer 1 mode 2 (auto reload).
4. SCON is set as 50H to have 8 bit data, one stop bit and REN enabled.
5. Data is received serially from UART window and sent to respective ports.

Program:

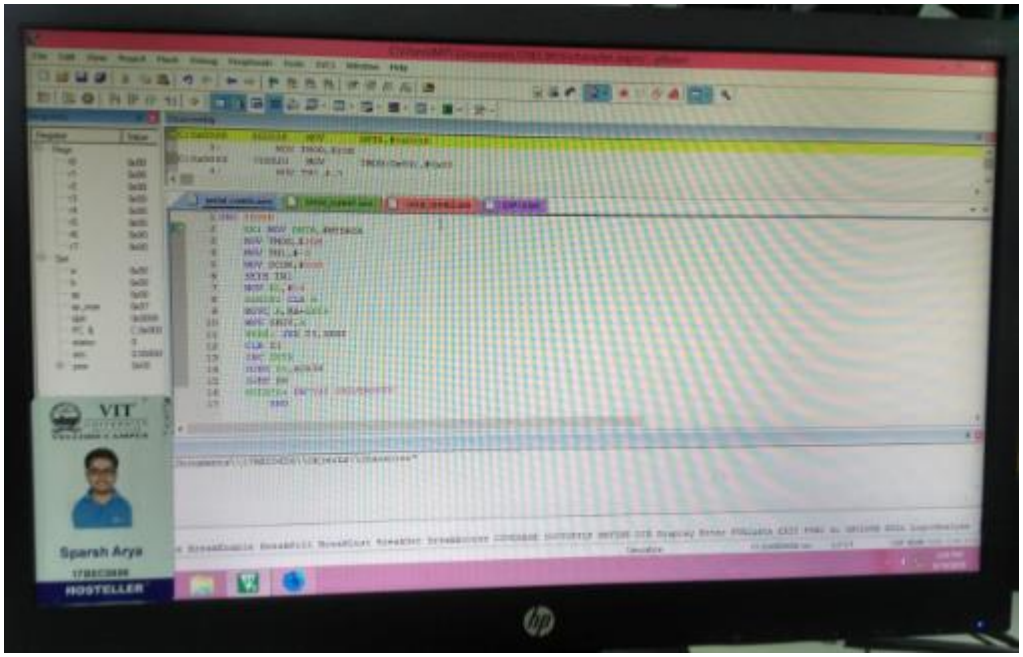
Memory Address	**Label**	**Mnemonics**	**Operands**	**addressing mode used**	**Machine cycle Required**	**Memory Byte Required**	**Type of Instruction**	**Comments**	**Flags getting affected by the Instruction**
-	-	ORG 0000H	-	-	-	-	-	A←0	-
-	-	MOV TMOD,#20H	TMOD	Immediate	1	2	TIMER	TIMER 1 MODE 2	M1←1 M0←0
-	-	MOV TH1,#-6	TH1	Immediate	1	2	Baud Rate	Baud rate←4800	-
-	-	MOV SCON,#50H	SCON	Immediate	1	2	Data Transfer	8 bit data, 1 stop bit and REN enabled	-
-	-	SETB TR1	TR1	-	1	1	TIMER	Start Timer 1	-
-	-	HERE: JNB RI,HERE	RI	-	2	3	Bit Manipulation	Jump if no bit in TI	TI←1

-	-	MOV A,SBUF	A,SBUF	Indirect	1	1	Data Transfer	A←SBUF	-
-	-	MOV P1,A	P1,A	Immediate	1	2	Data Transfer	P1←A	-
-	-	CLR RI	RI	-	1	2	Bit Manipulation	RI←0	-
-	-	SJMP HERE	-	-	2	2	Branch	Short jump to loop HERE	-
-	-	END	-	-	-	-	-	END	-

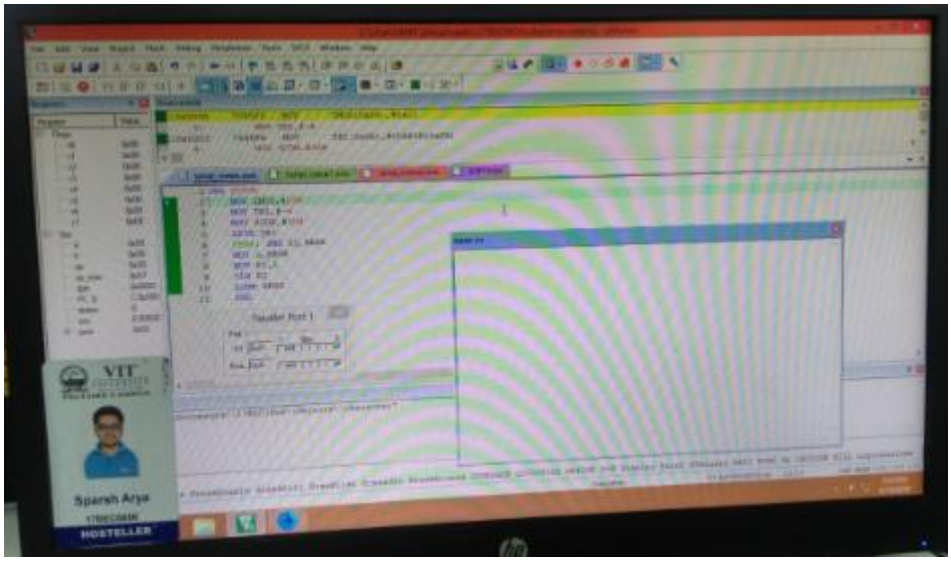
OUTPUTS:

OBSERVATIONS:

- BEFORE EXECUTION



- AFTER EXECUTION



INFERENCE

Input is taken from the UART window and sent to port p1.

RESULTS:

The 8051 ALP to recieve data serially was performed and the output was verified.

Aim:

Write an 8051 assembly program to

(a) Send to PC the message “We Are Ready”

(b) Receive any data send by PC and put it on LEDs connected to P1

(c) Get data on switches connected to P2 and send it to PC serially.

Algorithm:

1. The Baud Rate is set to be 9600(default value).
2. Use Timer 1 mode 2, auto reload is used to avoid assigning after every cycle of transmitting.
3. SCON is set as 50H.It will enable data to have 8 bits, one stop bit and a start bit. REN is also enabled.
4. The message “We Are Ready” is sent to UART window.
5. This data is then sent to port 1 and observed at UART window.
6. Changes are made to port 2 and the changes are reflected onto port 1.

Program:

Memory Address	**Label**	**Mnemonics**	**Operands**	**addressing mode used**	**Machine cycle Required**	**Memory Byte Required**	**Type of Instruction**	**Comments**	**Flags getting affected by the Instruction**
-	-	ORG 0000H	-	-	-	-	-	A←0	-
-	-	MOV P2,#0FFH	P2	Immediate	1	2	Data Transfer	P2 as input port	P2←FF
-	-	MOV TMOD,#20H	TMOD	Immediate	1	2	Timer	TIMER 1 MODE 2	M1←1 M0←0
-	-	MOV TH1,#0FAH	TH1	Immediate	1	2	Data Transfer	TH1←9600 Baud Rate	-
-	-	MOV SCON,#50H	SCON	Immediate	1	2	Data Transfer	8 bit data, 1 stop bit	-

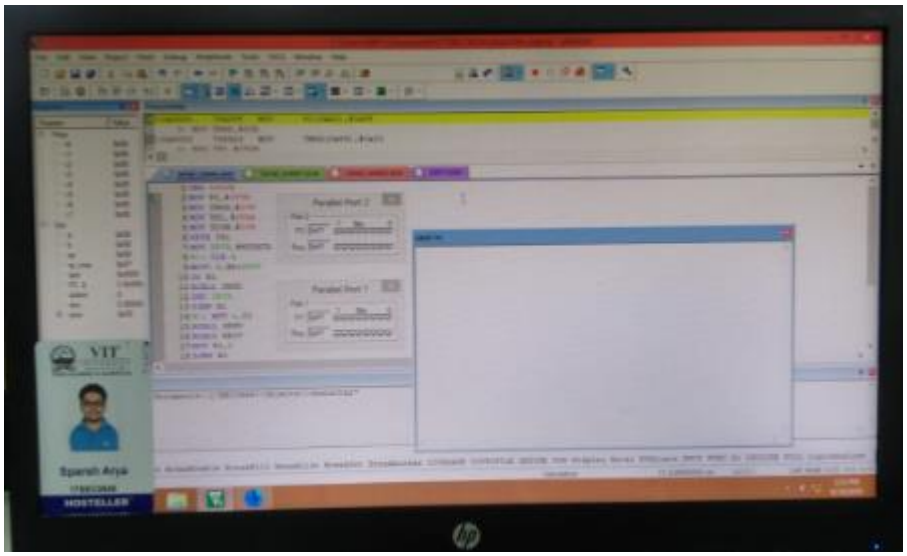
								and REN enabled	
-	-	SETB TR1	TR1	-	1	1	TIMER	Start Timer 1	-
-	-	MOV DPTR,#MYDATA	DPTR	Immediate	2	3	Data Transfer	DPTR←SP	-
-	-	H_1: CLR A	A	-	1	2	Bit Manipulation	A←0	-
-	-	MOVC A,@A+DPTR	A,DPTR	Index	2	1	Data Transfer	A←A+ DPTR	-
-	-	JZ B_1	-	-	2	2	Branch	Jump if bit zero	-
-	-	ACALL SEND	-	-	2	2	Branch	Absolute call with 2K page	-
-	-	INC DPTR	DPTR	-	2	1	Arithmetic	Increment data pointer	-
-	-	SJMP H_1	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	B_1: MOV A,P2	A,P1	Indirect	1	1	Data Transfer	A←P2	-
-	-	ACALL SEND	-	-	2	2	Branch	Absolute call with 2K page	-
-	-	ACALL RECV	-	-	2	2	Branch	Absolute call with 2K page	-
-	-	MOV P1,A	P1,A	Indirect	1	1	Data Transfer	P1←A	-
-	-	SJMP B_1	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	SEND: MOV SBUF,A	SBUF,A	Indirect	1	1	Data Transfer	SBUF←A	-

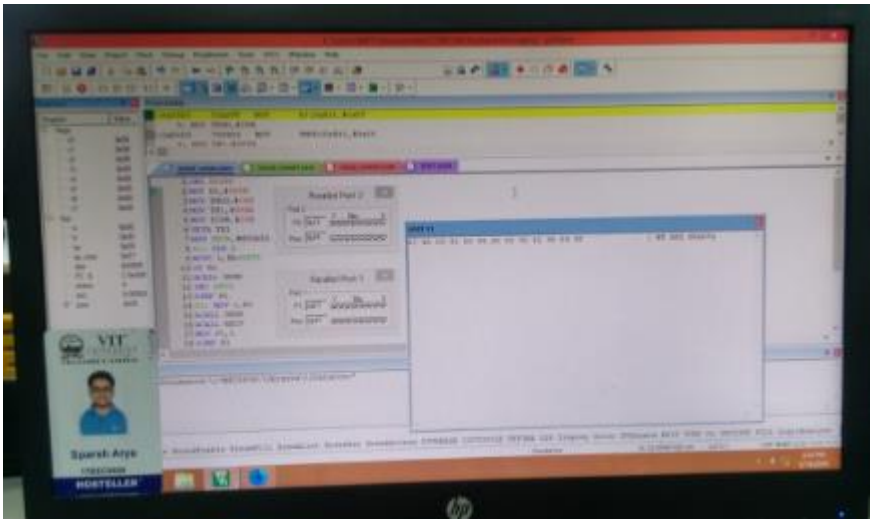
-	-	H_2: JNB TI,H_2	TI	-	2	3	Bit Manipulation	Jump if no bit in TI	TI←1
-	-	CLR TI	TI	-	1	2	Bit Manipulation	Clear TI	TI←0
-	-	RET	-	-	2	1	Branch	Return	-
-	-	RECV: JNB RI,RECV	RI	-	2	3	Bit Manipulation	Jump if no bit in RI	RI←1
-	-	MOV A,SBUF	A,SBUF	Direct	1	1	Data Transfer	A←SBUF	-
-	-	CLR RI	RI	-	1	2	Bit Manipulation	RI←0	-
-	-	RET	-	-	2	1	Branch	Clear RI	RI←0
-	-	MYDATA: DB 'We Are Ready'	-	-	-	-	-	“We Are Ready” is saved in DB	-
-	-	END	-	-	-	-	-	END	-

OUTPUTS:

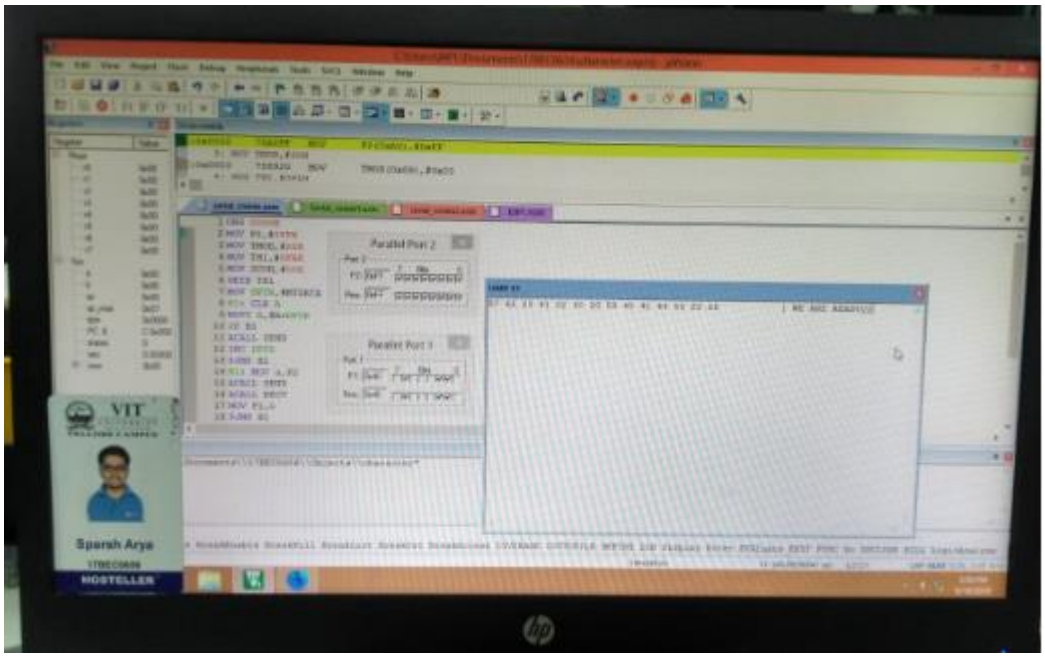
OBSERVATIONS:

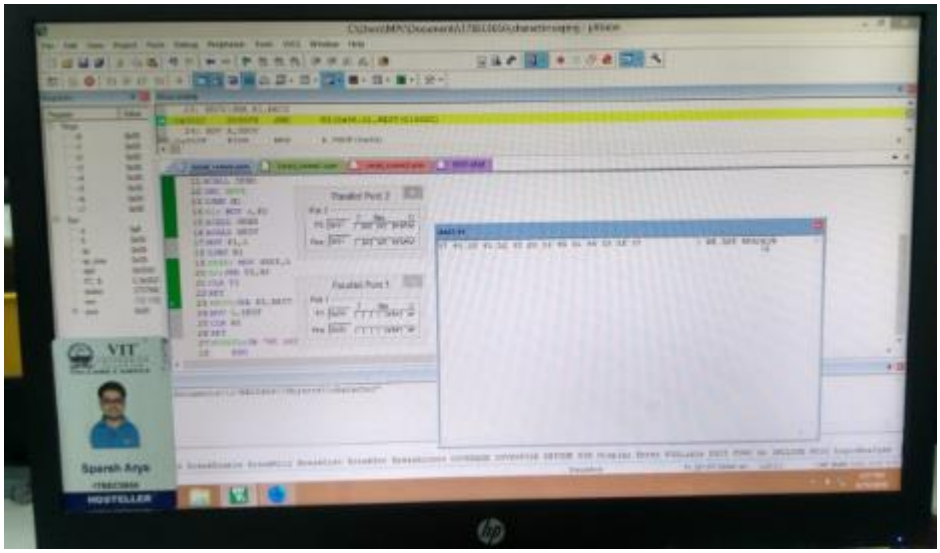
- BEFORE EXECUTION





- AFTER EXECUTION





INFERENCE

The data is framed and deframed to transfer and receive as both operations were performed.

RESULTS:

The 8051 ALP to transfer and receive data serially was performed and the output was verified. Modifications on port 2 are reflected directly on port 1.

Aim:

Write an 8051 program to get data from port P0 and send it to port P1 continuously while an interrupt will do the following: Timer 0 will toggle the P2.1 bit every 100 microseconds.

Algorithm:

1. Interrupt for Timer 0 is used i.e. 000B.
2. Toggle P2.1 using CPL function using a interrupt program.
3. Data is sent serially from port Port 0 to port Port 1.
4. The delay is 100us using timer TH0= -92.
5. The graph is observed in the UART window along with port output.
6. Results are verified.

Program:

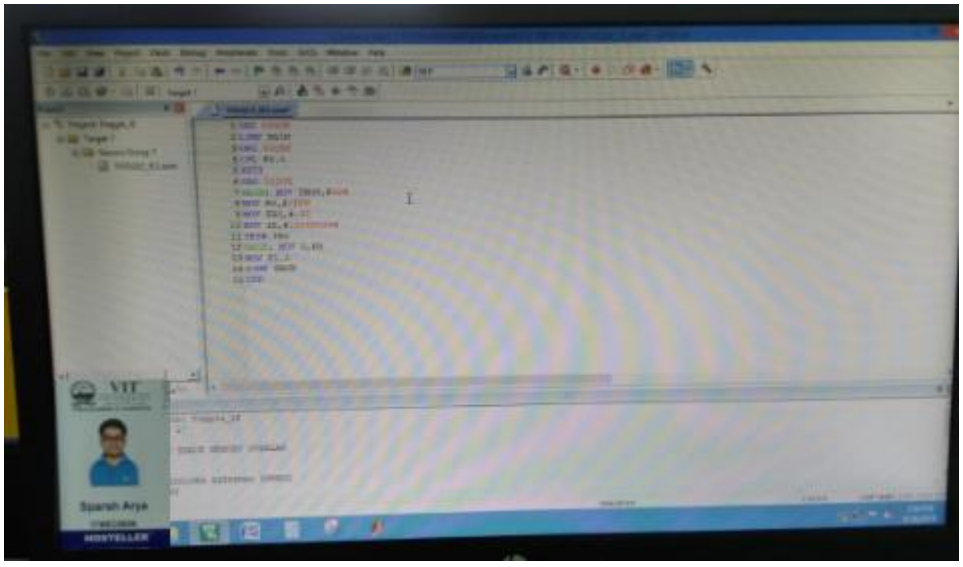
Memory Address	**Label**	**Mnemonics**	**Operands**	**addressing mode used**	**Machine cycle Required**	**Memory Byte Required**	**Type of Instruction**	**Comments**	**Flags getting affected by the Instruction**
-	-	ORG 0000H	-	-	-	-	-	A←0	-
-	-	LJMP MAIN	-	-	2	3	Branch	Absolute long jump	-
-	-	ORG 000BH	-	-	-	-	-	Timer 0 Interrupt	-
-	-	CPL P2.1	P2.1	-	1	1	Bit Manipulation	Complement P2.1 bit	-
-	-	RETI	-	-	2	1	-	Return from Interrupt	-
-	-	ORG 0030H	-	-	-	-	-	Return to Main loop	-
-	-	MAIN: MOV TMOD,#02H	TMOD	Direct	1	2	Data Transfer	Timer 1 mode 2	M1←1 M0←0

-	-	MOV P0,#0FFH	P0	Direct	1	2	Data Transfer	Enable port P0	P0←00H
-	-	MOV TH0,#-92	TH0	Direct	1	2	Data Transfer	For 100us	-
-	-	MOV IE,#10000010B	IE	Direct	1	2	Data Transfer	For Timer 0 Interrupt	-
-	-	SETB TR0	TR0	-	2	2	Bit Manipulation	Start Timer 0	-
-	-	BACK: MOV A,P0	A,P0	Immediate	1	2	Data Transfer	A←P0	-
-	-	MOV P1,A	P1,A	Immediate	1	2	Data Transfer	P1←A	-
-	-	SJMP BACK	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	END	-	-	-	-	-	END	-

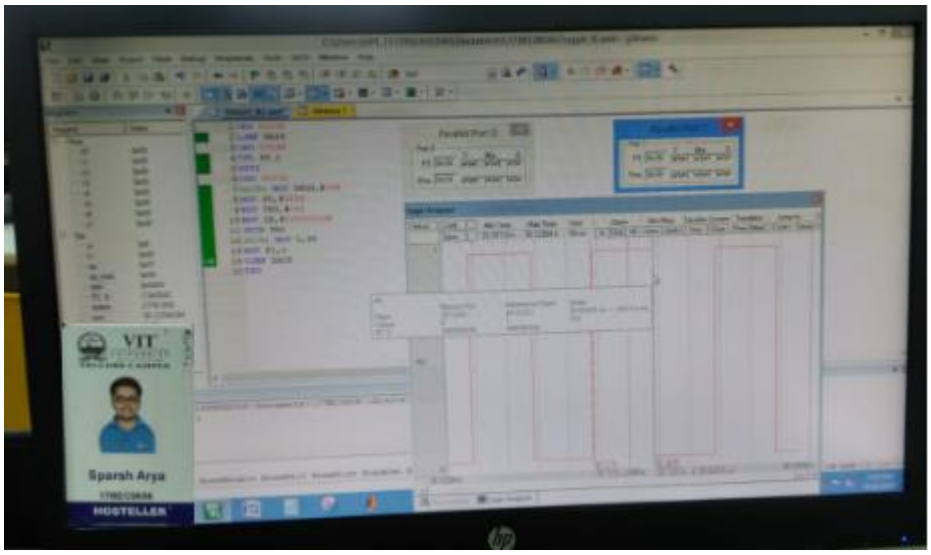
OUTPUTS:

OBSERVATIONS:

- BEFORE EXECUTION



- AFTER EXECUTION



INFERENCE

The IE register helps to perform the interrupt program. Also there are different IVT values for everything. As soon interrupt is called it goes to the ISR to perform that task then return back to main program.

RESULTS:

The 8051 ALP for interrupt programming was performed and the output was verified.

Aim:

Write an 8051 program to get data from a single bit of P1.2 and send it to P1.7 continuously while an interrupt will do the following: A serial interrupt service routine will receive data from a PC and display it on P2 ports.

Algorithm:

1. Get the input bit from p1.2 and move it to p1.7 via C.
2. Serial interrupt Routine is used using 0023H.
3. UART window will take input and transmit serially to port P2.
4. In port P1, bits at P1.2 are changed and the port pin P1.7 is changed serially.
5. Thus, interrupt will manage 2 tasks simultaneously to avoid wastage of machine cycles.

Program:

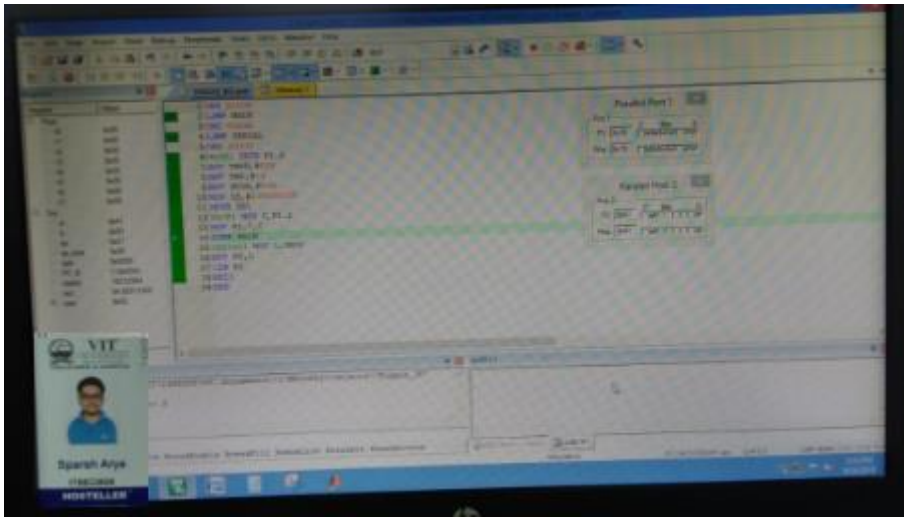
Memory Address	**Label**	**Mnemonics**	**Operands**	**addressing mode used**	**Machine cycle Required**	**Memory Byte Required**	**Type of Instruction**	**Comments**	**Flags getting affected by the Instruction**
-	-	ORG 0000H	-	-	-	-	-	START	-
-	-	LJMP MAIN	-	-	2	3	Branch	Absolute Long Jump	-
-	-	ORG 0023H	-	-	-	-	-	Serial Interrupt	-
-	-	LJMP SERIAL	-	-	2	3	Branch	Absolute Long Jump	-
-	-	ORG 0030H	-	-	-	-	-	Main Loop	-
-	-	MAIN: SETB P1.2	P1.2	-	2	2	Bit Manipulation	P1.2←ON	-
-	-	MOV TMOD,#20H	TMOD	Immediate	1	2	Data Transfer	Timer 1 Mode 2	M1←1 M0←0

-	-	MOV TH1,#-3	TH1	Immediate	1	2	Data Transfer	TH1←-3 Baud rate=9600	-
-	-	MOV SCON,#50H	SCON	Immediate	1	2	Data Transfer	8 bit data, 1 stop bit and REN enabled	-
-	-	MOV IE,#10010000B	IE	Immediate	1	2	Data Transfer	IE for serial interrupt	-
-	-	SETB TR1	TR1	-	2	2	Bit Manipulation	Start timer 1	-
-	-	BACK: MOV C,P1.2	C,P1.2	Immediate	1	2	Data Transfer	C←P1.2	-
-	-	MOV P1.7,C	C,P1.7	Immediate	1	2	Data Transfer	P1.7←C	-
-	-	SJMP BACK	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	SERIAL: MOV A,SBUF	A,SBUF	Immediate	1	2	Data Transfer	A←SBUF	-
-	-	MOV P2,A	P2,A	Immediate	1	2	Data Transfer	P2←A	-
-	-	CLR RI	RI	-	1	2	Bit Manipulation	RI←0	-
-	-	RETI	-	-	2	1	Branch	Return Interrupt	-
-	-	END	-	-	-	-	-	END	-

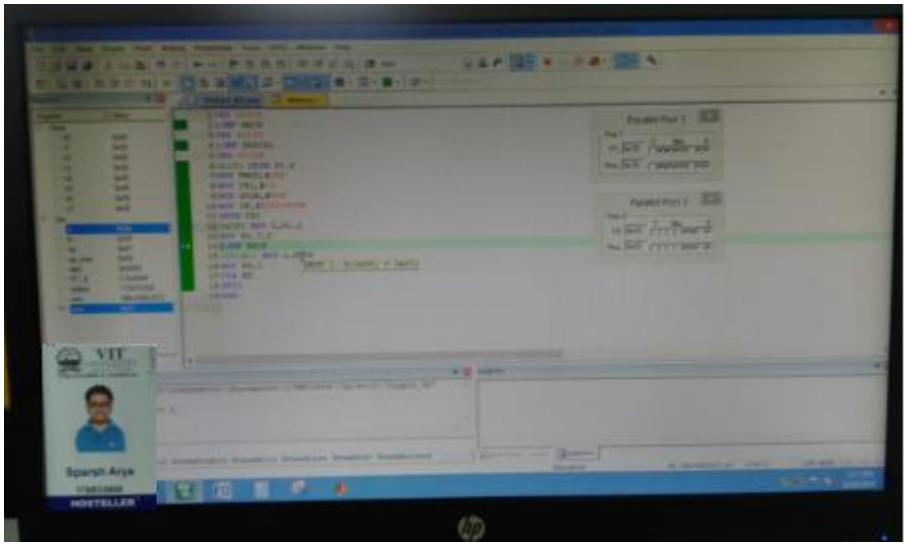
OUTPUTS:

OBSERVATIONS:

- BEFORE EXECUTION



- AFTER EXECUTION



INFERENCE

The IE register helps to perform the interrupt program. Also there are different IVT values for everything. As soon interrupt is called it goes to the ISR to perform that task then return back to main program.

RESULTS:

The 8051 ALP for interrupt programming was performed and the output was verified.

Aim:

Write a program using interrupts to do the following:

(a) Receive data serially and sent it to P0,

(b) Have P1 port read and transmitted serially, and a copy given to P2,

(c) Make timer 0 generate a square wave of 5 KHz frequency on P3.2.

Assume that XTAL-11.0592. Set the baud rate at 4800.

Algorithm:

1.First we write our main program that is to have port 1 read as well as transmit data and give a copy to port 2.

2. Then we give the IVT of timer 0 which is 000b and compliment port 3.2 to generate 5 KHz square wave.

3. Then we give IVT of serial programming 0023h to perform ISR of receiving data serially and sending to port 0.

4. Then we return back from the ISR program.

5. We write the IE register value to perform the interrupt program.

Program:

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	ORG 0000H	-	-	-	-	-	START	-
-	-	LJMP MAIN	-	-	2	3	Branch	Absolute Long Jump	-
-	-	ORG 000BH	-	-	-	-	-	Timer 0 Interrupt	-
-	-	CPL P3.2	P3.2	-	1	2	Bit Manipulation	Complement bit P3.2	-

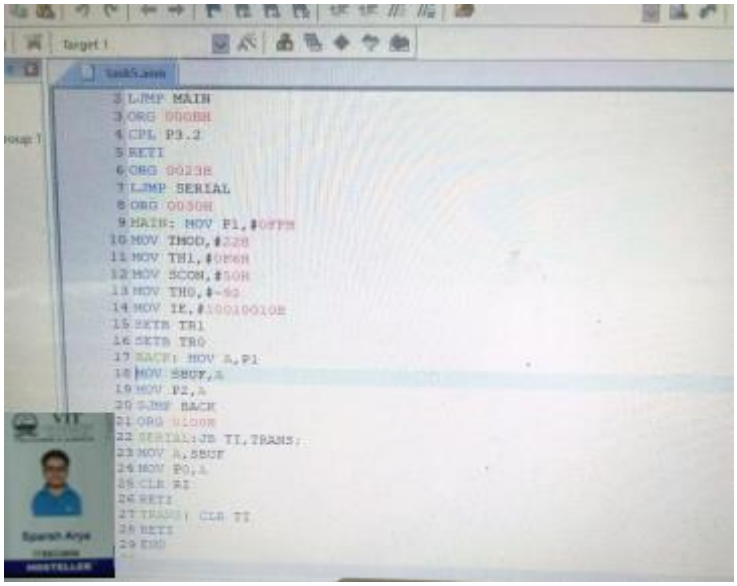
-	-	RETI	-	-	2	1	Branch	Return Interrupt	-
-	-	ORG 0023H	-	-	-	-	-	Serial Interrupt	-
-	-	LJMP SERIAL	-	-	2	3	Branch	Absolute Long Jump	-
-	-	ORG 0030H	-	-	-	-	-	Main Loop	-
-	-	MAIN: MOV P1,#0FFH	P1	Immediate	1	2	Data Transfer	P1←ON	-
-	-	MOV TMOD,#22H	TMOD	Immediate	1	2	Data Transfer	Timer 1 and Timer2 are set	M1←1 M0←0
-	-	MOV TH1,#-6	TH1	Immediate	1	2	Data Transfer	Baud Rate=4800	-
-	-	MOV SCON,#50H	SCON	Immediate	1	2	Data Transfer	8 bit data, 1 stop bit and REN enabled	-
-	-	MOV TH0,#0A4H	TH0	Immediate	1	2	Data Transfer	5KHz frequency	-
-	-	MOV IE,#10010010B	IE	Immediate	1	2	Data Transfer	Serial and Timer Interrupt	-
-	-	SETB TR1	TR1	-	2	2	-	Start timer 1	-
-	-	SETB TR0	TR0	-	2	2	-	Start Timer 0	-
-	-	BACK: SJMP BACK	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	SERIAL:JB TI,TRANS	TI	-	2	3	Bit Manipulation	Jump if TI←1	TI←1
-	-	MOV A,SBUF	A,SBUF	Direct	1	1	Data Transfer	A←SBUF	-

-	-	MOV P0,A	P0,A	Direct	1	1	Data Transfer	P0←A	-
-	-	CLR RI	RI	-	1	2	Bit Manipulation	RI←0	RI←0
-	-	RETI	-	-	2	1	-	Return Interrupt	-
-	-	TRANS: CLR TI	TI	-	1	2	Bit Manipulation	TI←0	TI←0
-	-	RETI	-	-	2	1	-	Return Interrupt	-
-	-	END	-	-	-	-	-	END	-

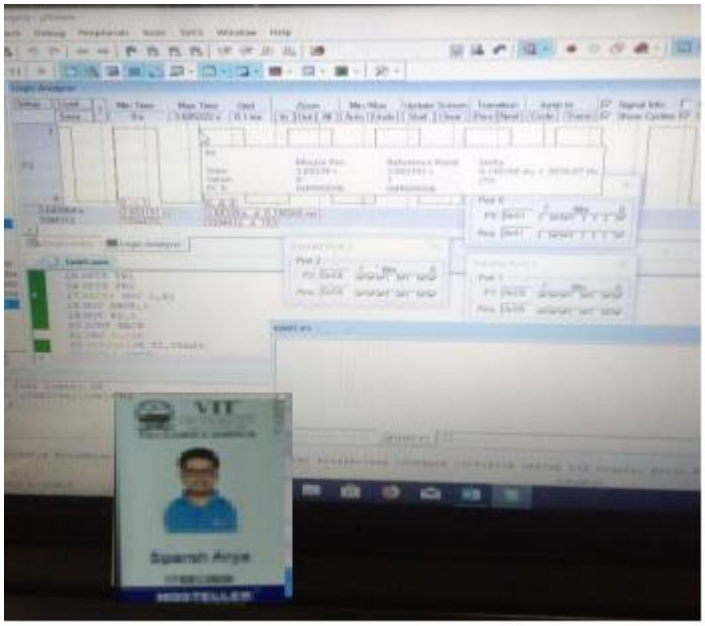
OUTPUTS:

OBSERVATIONS:

- BEFORE EXECUTION



- AFTER EXECUTION



- INFERENCE
- The IE register helps to perform the interrupt program. Also there are different IVT values for everything. As soon interrupt is called it goes to the ISR to perform that task then return back to main program.

-
- RESULTS:
- The 8051 ALP for interrupt programming was performed and the output was verified.

KEY PAD CODING