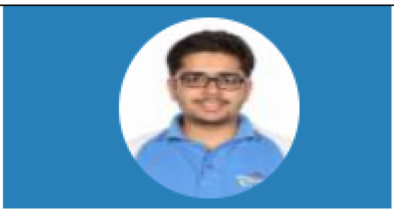
TASK 3

School of Electronics Engineering
VIT, Vellore

Registration No.	17BEC0656		
Student Name	SPARSH ARYA		
Course Code	ECE3003	**Slot & Semester**	L49+L50, FALL SEM 2019-20
Course Name	Microcontroller and its applications		
Program Title	TASK 3		
Date of Exp.	20-09-19	**Date of Submission**	
Faculty	A.Karthikeyan		

Aim:

To write an 8051 ALP to generate a square wave of frequency 1kHz at pin 0 of port 1

Tools Required:

Algorithm:

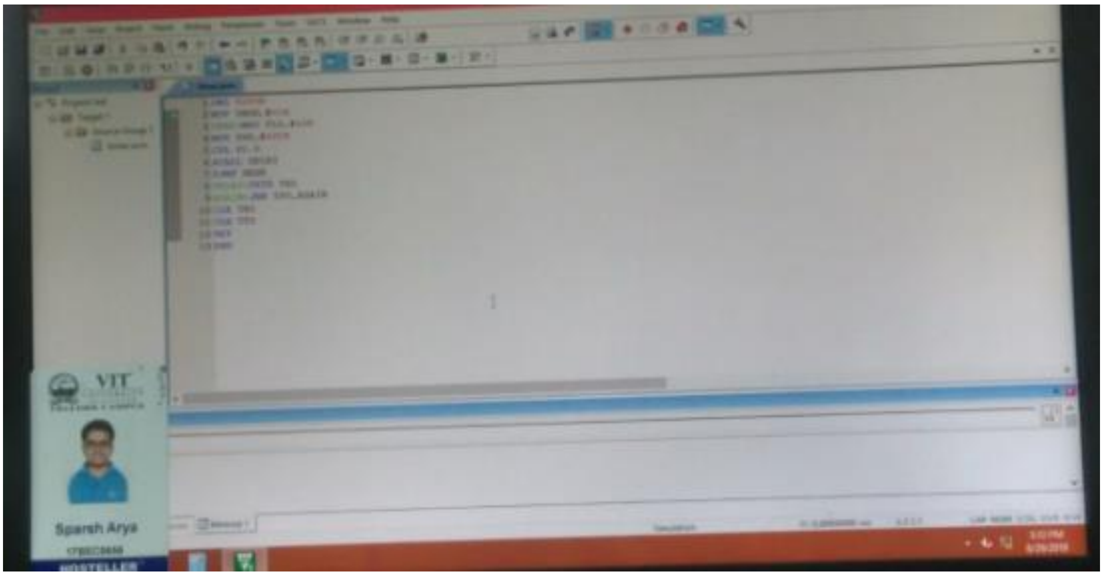
1. The timer mode is configured by transferring appropriate data to the TMOD register.
2. The initial count is transferred to the TL and TH registers
3. The port pins are complemented for generating the square wave and a delay sub routine is called
4. The delay subroutine is written with the help of timer.
5. The timer output is viewed using the Keil software.
6. The frequency of the output wave is measured by measuring the time period.
7. The inverse of the time period is calculated and compared to 1KHz.

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	-	-	-	-	PSEUDO INSTRUC-TION		
0000H		MOV	TMOD,#10H	IMMEDIATE	1	2	DATA TRANSFER	TIMER MODE CONFIGUARTION	
0002H	HERE	MOV	TL1,#33H	IMMEDIATE	1	2	DATA TRANSFER	INITIAL COUNT LOAD	
0004H		MOV	TH1,#0FEH	IMMEDIATE	1	2	DATA TRANSFER	INITIAL COUNT LOAD	
0006H		CPL	P1.0	-	1	1	BIT MANIPULATION	PORT COMPLIMENT TO GENERATE WAVE	
0007H		ACALL	DELAY	-	2	2	BRANCH	DEALY TO GENERATE TIME PERIOD	
0009H		SJMP	HERE	-	2	2	BRANCH	LOOPING STATEMENT	
000CH	DELAY	SETB	TR1	-	1	1	BIT MANIPULATION	STARTS COUNTER OPERATION	
000EH	AGAIN	JNB	TF1,AGAIN	-	3	2	BRANCH	LOOPING STATEMENT	
0011H		CLR	TR1	-	1	1	BIT MANIPULATION	STOP TIMER ACTION	

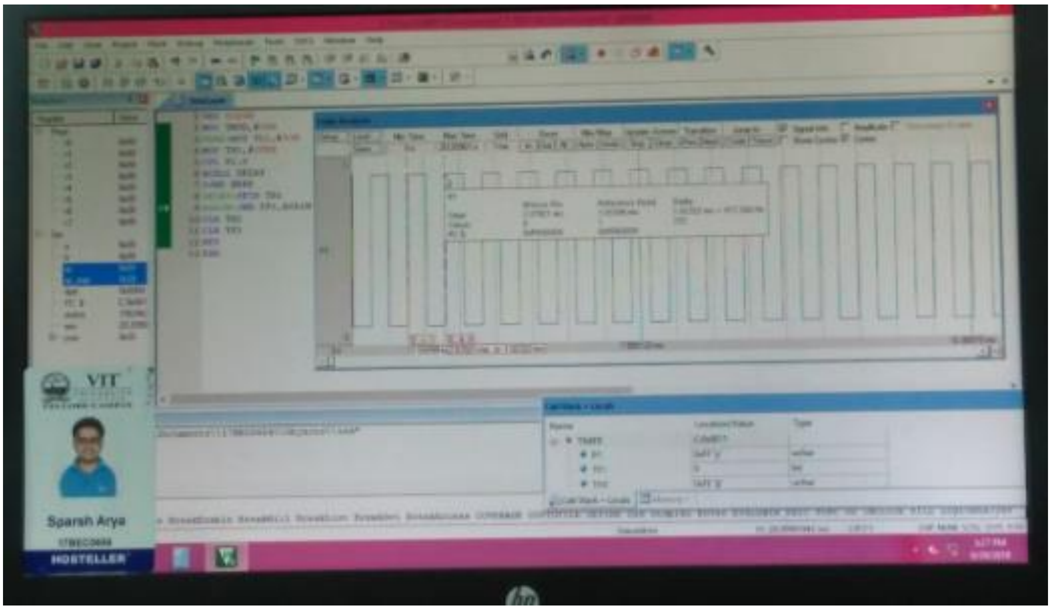
0012H		CLR	TF1	-	1	1	BIT MANIPULATION	CLEAR TIMER FLAG	
RET									
END									

Results and Observations

OUTPUT BEFORE RUNNING THE CODE



OUTPUT AFTER RUNNING THE CODE.



Result:

The 8051 has successfully performed the operation of generating a timer of 1KHz. The waveform has been plotted by Keil software.

Aim:

To write an 8051 ALP to generate a square wave of frequency 0.5kHz at pin 0 of port 1

Tools Required:

Algorithm:

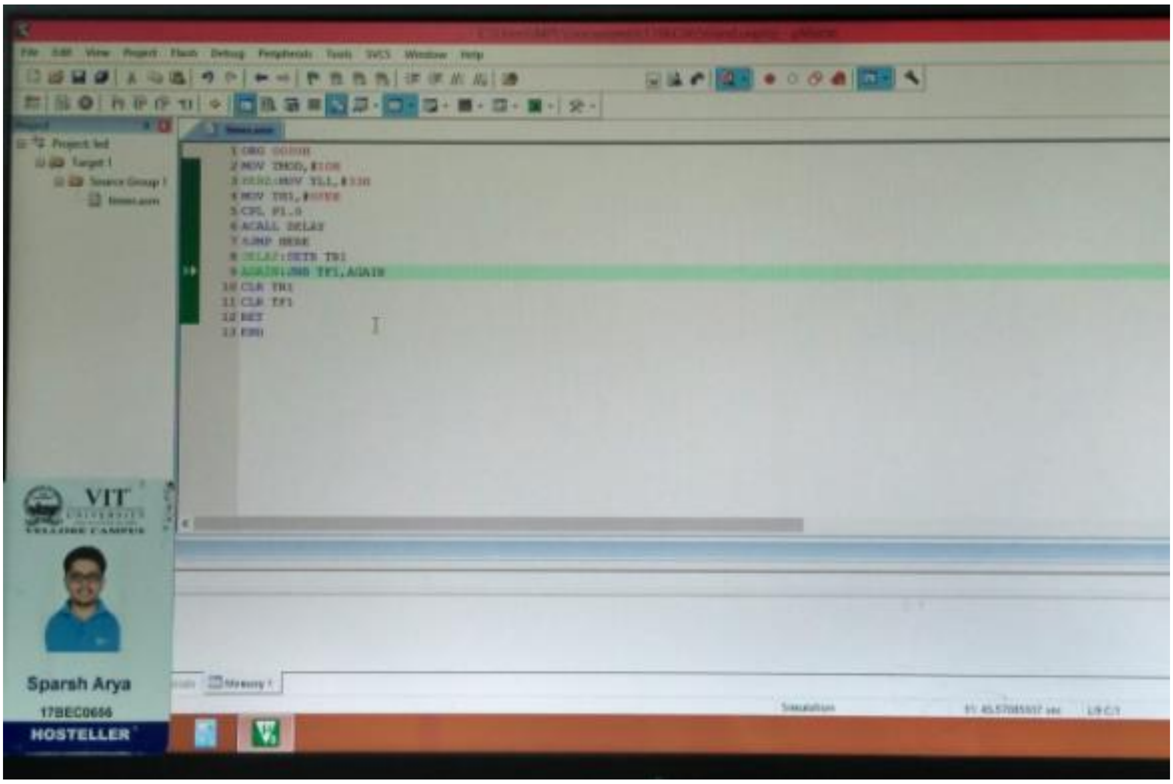
1. The timer mode is configured by transferring appropriate data to the TMOD register.
2. The initial count is transferred to the TL and TH registers.
3. The port pins are complemented for generating the square wave and a delay sub routine is called
4. The delay subroutine is written with the help of timer.
5. The timer output is viewed using the Keil software.
6. The frequency of the output wave is measured by measuring the time period.
7. The inverse of the time period is calculated and compared to 1KHz.

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	-	-	-	-	PSEUDO INSTRUC-TION		
0000H		MOV	TMOD,#01H	IMMEDIATE	1	2	DATA TRANSFER	TIMER MODE CONFIGUARTION	
0002H	HERE	MOV	TL0,#66H	IMMEDIATE	1	2	DATA TRANSFER	INITIAL COUNT LOAD	
0004H		MOV	TH0,#0FCH	IMMEDIATE	1	2	DATA TRANSFER	INITIAL COUNT LOAD	
0006H		CPL	P1.0	-	1	1	BIT MANIPULATION	PORT COMPLIMENT TO GENERATE WAVE	
0007H		ACALL	DELAY	-	2	2	BRANCH	DEALY TO GENERATE TIME PERIOD	
0009H		SJMP	HERE	-	2	2	BRANCH	LOOPING STATEMENT	
000CH	DELAY	SETB	TR0	-	1	1	BIT MANIPULATION	STARTS COUNTER OPERATION	
000EH	AGAIN	JNB	TF0,AGAIN	-	3	2	BRANCH	LOOPING STATEMENT	
0011H		CLR	TR0	-	1	1	BIT MANIPULATION	STOP TIMER ACTION	

0012H		CLR	TF0	-	1	1	BIT MANIPULATION	CLEAR TIMER FLAG	
RET									
END									

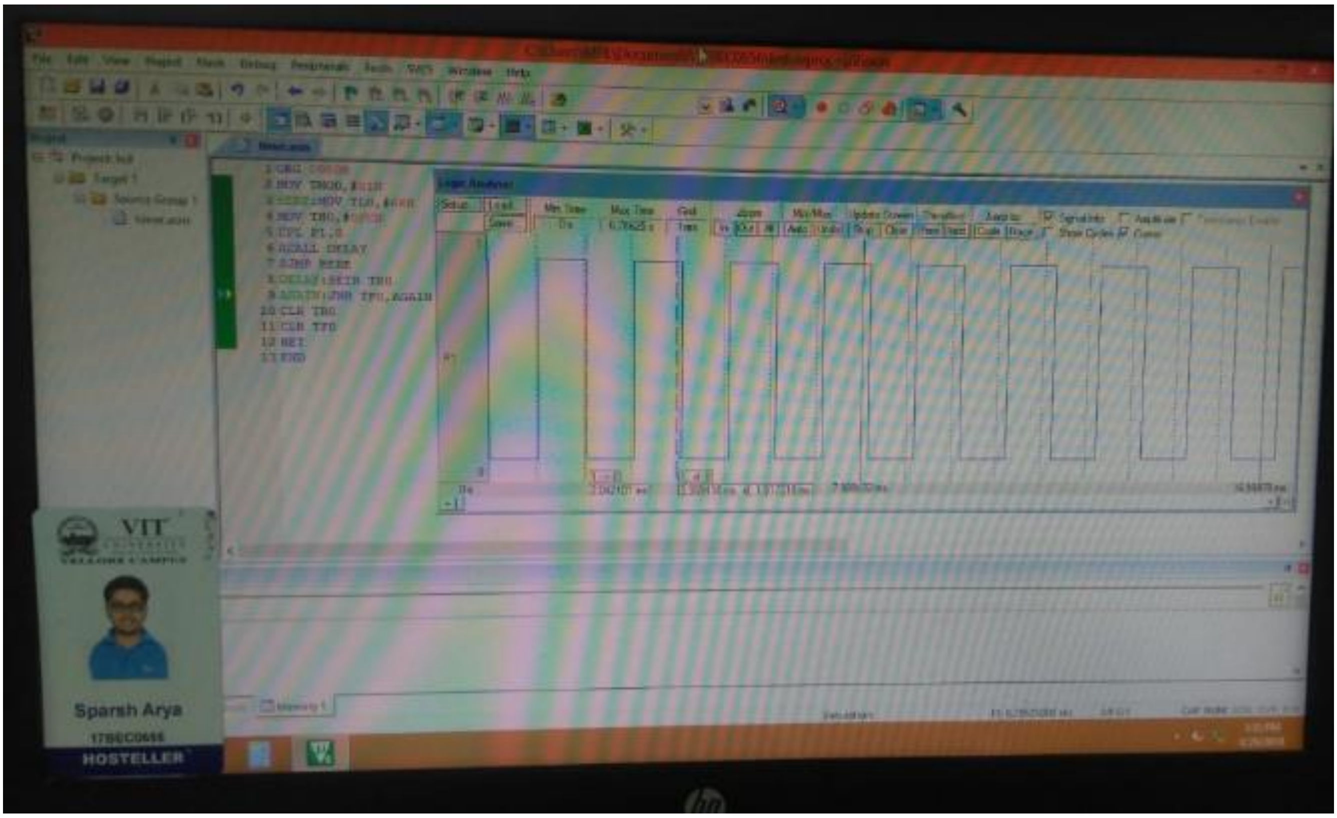
Results and Observations

OUTPUT BEFORE RUNNING THE CODE



```
ORG [illegible]
MOV TMOD,#[illegible]
HERE:MOV TL1,#[illegible]
MOV TH1,#[illegible]
CPL P1.0
ACALL DELAY
SJMP HERE
DELAY:SETB TR1
AGAIN:JNB TF1,AGAIN
CLR TR1
CLR TF1
RET
END
```

OUTPUT AFTER RUNNING THE CODE.



```
ORG [illegible]
MOV TMOD,#[illegible]
HERE:MOV TL0,#[illegible]
MOV TH0,#[illegible]
CPL P1.0
ACALL DELAY
SJMP HERE
DELAY:SETB TR0
AGAIN:JNB TF0,AGAIN
CLR TR0
CLR TF0
RET
END
```

Result:

The 8051 has successfully performed the operation of generating a timer of 1KHz. The waveform has been plotted by Keil software.

Aim:

To write an 8051 ALP to toggle the bits of port 0,1 and 2.

Tools Required:

Algorithm:

1. Transfer high bits to the ports port 0, port 1 and port 2.
2. Call the delay sub routine using ACALL function.
3. Transfer low bits to the ports 0 1 and 2.
4. Call the delay sub routine using ACALL function in order to enable toggle.
5. Using SJMP instruction repeat the above steps iteratively
6. Using the nested loop fashion write a delay sub routine.
7. Observe the output at the respective ports.
8. Observe the waveform and its characteristics.

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	-	-	-	-	PSEUDO INSTRUC-TION		
0000H	HERE	MOV	P0,#55H	-	1	2	DATA TRANSFER		
0002H		MOV	P1,#55H	-	1	2	DATA TRANSFER		
0004H		MOV	P2,#55H	-	1	2	DATA TRANSFER		
0006H		ACALL	DELAY	-	2	2	BRANCH		
0008H		MOV	P0,#0AAH	-	1	2	DATA TRANSFER		
000AH		MOV	P1,#0AAH	-	1	2	DATA TRANSFER		
000CH		MOV	P2,#0AAH	-	1	2	DATA TRANSFER		
000EH		SJMP	HERE	-	2	2	BRANCH		
0010H	DELAY	MOV	R1,#200H	IMMEDIATE	1	2	DATA TRANSFER		
0012H	BACK	MOV	R2,#200H	IMMEDIATE	1	2	DATA TRANSFER		
0014H	AGAIN	DJNZ	R2,AGAIN	-	2	2	BRANCH		
0016H	-	DJNZ	R1,BACK	-	2	2	BRANCH		
RET									

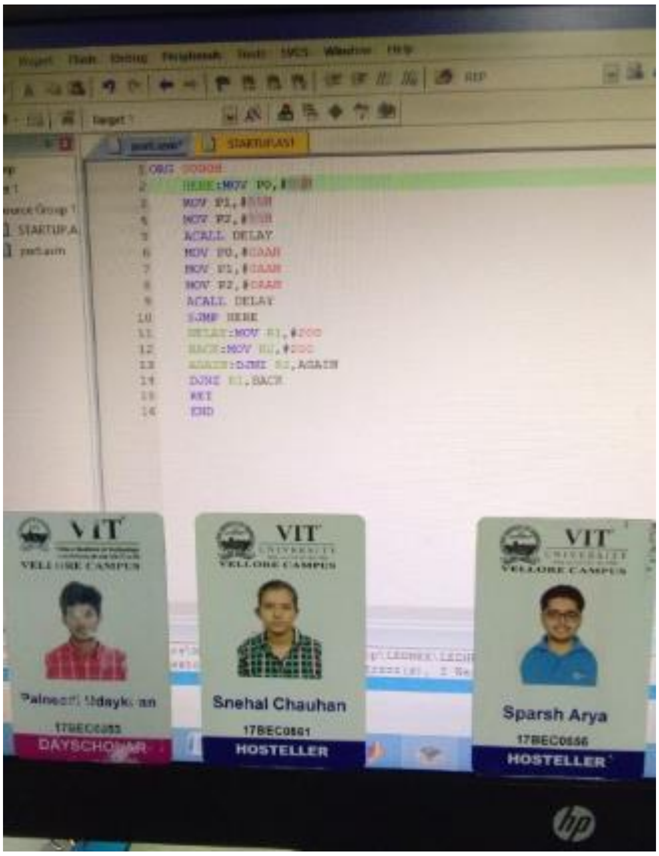
END									

OUTPUT

R1=71H

R0=0AH

Results and Observations



```
ORG 0000H
    HERE:MOV P0,#55H
    MOV P1,#55H
    MOV P2,#55H
    ACALL DELAY
    MOV P0,#0AAH
    MOV P1,#0AAH
    MOV P2,#0AAH
    ACALL DELAY
    SJMP HERE
    DELAY:MOV R1,#200
    BACK:MOV R2,#200
    AGAIN:DJNZ R2,AGAIN
    DJNZ R1,BACK
    RET
    END
```

Result:

The 8051 ALP to perform toggling of bits at the ports has been successfully verified with proper waveforms.

Aim:

To write an 8051 ALP to transfer data from port 1 to port 2

Tools Required:

Algorithm:

1. Data to be handled is initially stored in the Accumulator register.
2. The data is then transferred to the Port 1.
3. The data is transferred from Port 1 to Accumulator
4. Then, Data transfer from Accumulator to Port 2 occurs.
5. Finally, the output at the ports are observed and the waveform is plotted.

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	-	-	-	-	PSEUDO INSTRUC-TION		
0000H		MOV	A,#0FFH	IMMEDIATE	1	2	DATA TRANSFER	Data to be transferred is copied into CPU register	
0002H		MOV	P1,A	REGISTER DIRECT	1	1	DATA TRANSFER	Data first sent to port 1	
0003H	HERE	MOV	A,P1	REGISTER DIRECT	1	1	DATA TRANSFER	Intended data transfer from port	

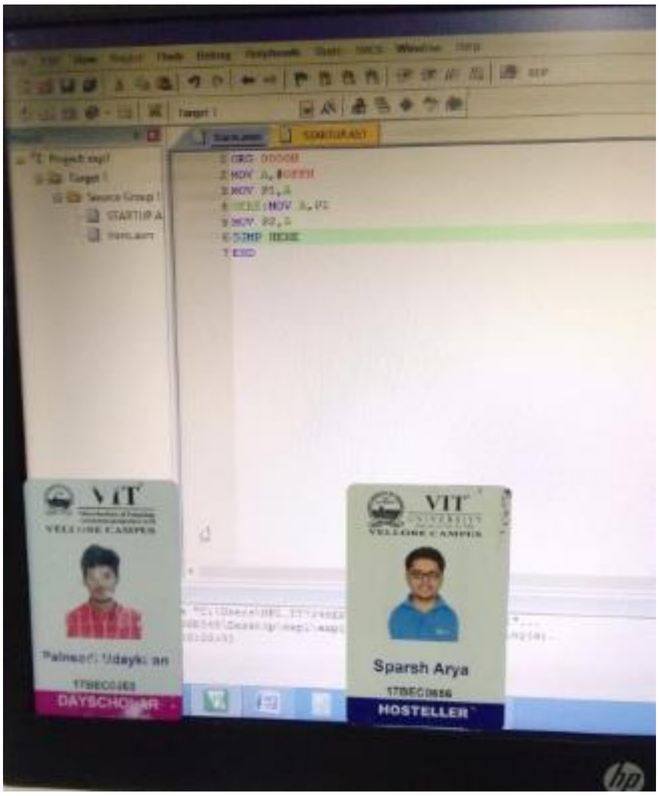
								to CPU first	
0004H		MOV	P2,A	REGISTER DIRECT	1	1	DATA TRANSFER	Data transfer from CPU to another PORT	
0005H		SJMP	HERE	-	2	2	BRANCH	Jump statement to execute a loop	
		END	-	-	-	-	PSEUDO		

Program:

OUTPUT

A = 0FFH

Results and Observations



```
ORG 0000H
MOV A,#0FFH
MOV P1,A
HERE:MOV A,P1
MOV P2,A
SJMP HERE
END
```

Result:

The 8051 ALP to perform transfer of data between ports via clock pulses has been successfully executed.

Aim:

Hardware implementation of port pin output toggling visualized as blinking of LED

Tools Required:

Algorithm:

1. Transfer high bits to the ports 0,1 and 2 .
2. Call the delay sub routine using ACALL
3. Transfer low bits to the ports p0,p1,p2.
4. Call the delay sub routine using ACALL
5. Using SJMP instruction repeat the above steps iteratively
6. Using the nested loop fashion write a delay sub routine.

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	-	-	-	-	PSEUDO INSTRUC-TION		
0000H	HERE	MOV	P0,#55H	-	1	2	DATA TRANSFER		
0002H		MOV	P1,#55H	-	1	2	DATA TRANSFER		
0004H		MOV	P2,#55H	-	1	2	DATA TRANSFER		
0006H		ACALL	DELAY	-	2	2	BRANCH		
0008H		MOV	P0,#0AAH	-	1	2	DATA TRANSFER		
000AH		MOV	P1,#0AAH	-	1	2	DATA TRANSFER		
000CH		MOV	P2,#0AAH	-	1	2	DATA TRANSFER		
000EH		SJMP	HERE	-	2	2	BRANCH		
0010H	DELAY	MOV	R1,#200H	IMMEDIATE	1	2	DATA TRANSFER		
0012H	BACK	MOV	R2,#200H	IMMEDIATE	1	2	DATA TRANSFER		
0014H	AGAIN	DJNZ	R2,AGAIN	-	2	2	BRANCH		
0016H	-	DJNZ	R1,BACK	-	2	2	BRANCH		
RET									

END									

OUTPUT

R1=71H

R0=0AH

Results and Observations

DISPLAY OF HARDWARE OUTPUT

Result:

The 8051 ALP to perform blinking of LED has been successfully executed and outputs have been verified.

Aim:

Hardware implementation of data transfer from port 1 to port 2

Tools Required:

Algorithm:

1. Data to be handled is initially stored in the Accumulator register.
2. The data is then transferred to the Port 1
3. Actual data transfer from Port 1 to Accumulator
4. Data transfer from Accumulator to Port 2.
5. The output is observed at the hardware.

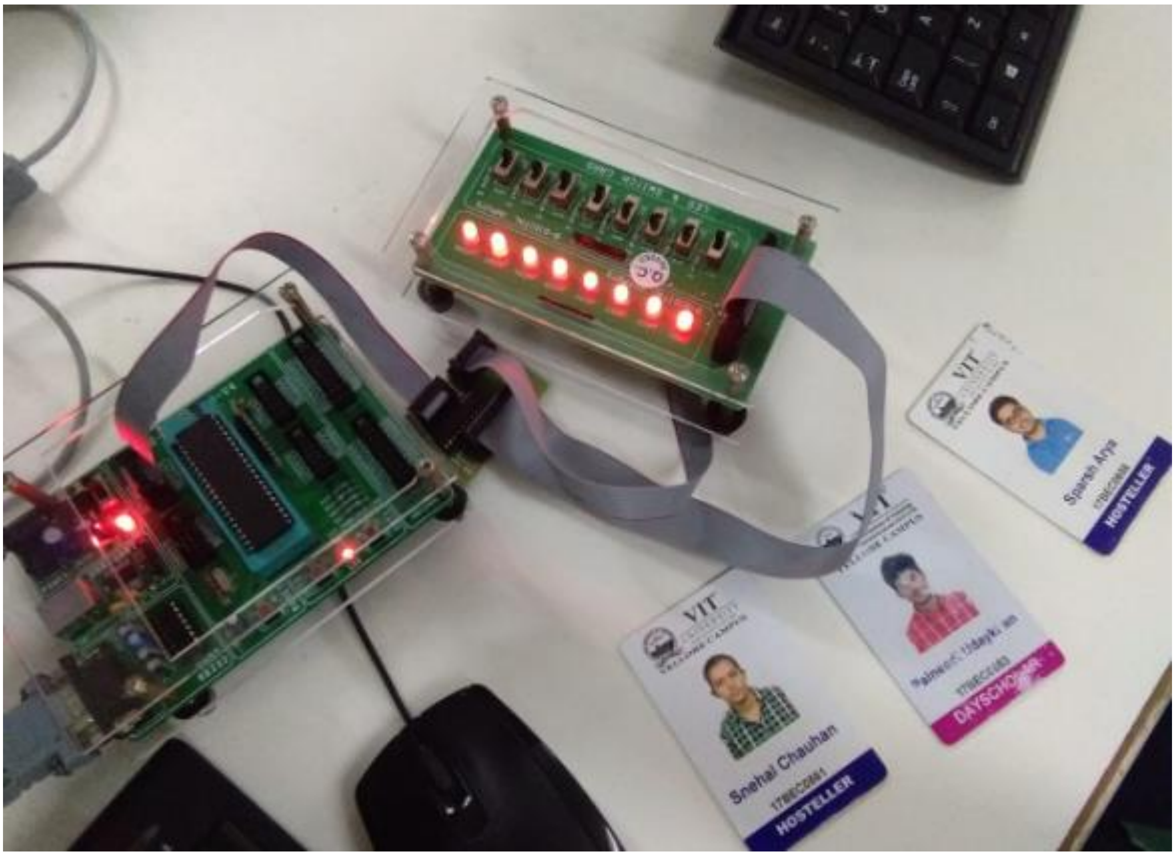
Memory Address	**Label**	**Mnemonics**	**Operands**	**addressing mode used**	**Machine cycle Required**	**Memory Byte Required**	**Type of Instruction**	**Comments**	**Flags getting affected by the Instruction.**
		ORG	-	-	-	-	PSEUDO INSTRUC-TION		
0000H		MOV	A,#0FFH	IMMEDIATE	1	2	DATA TRANSFER	Data to be transferred is copied into CPU register	
0002H		MOV	P1,A	REGISTER DIRECT	1	1	DATA TRANSFER	Data first sent to port 1	
0003H	HERE	MOV	A,P1	REGISTER DIRECT	1	1	DATA TRANSFER	Intended data transfer from port to CPU first	
0004H		MOV	P2,A	REGISTER DIRECT	1	1	DATA TRANSFER	Data transfer from CPU to	

								another PORT	
0005H		SJMP	HERE	-	2	2	BRANCH	Jump statement to execute a loop	
		END	-	-	-	-	PSEUDO		

Output :

A = 0FFH

Results and Observations



HARDWARE DISPLAY OF OUTPUT

Result:

The 8051 ALP to perform transfer of data from port 1 to 2 has been successfully executed and outputs have been verified.