Name: Bhargavi Kamble

Student ID: 202051048

Practical-1

Aim (i):-

Introduction to Keil software and basics steps of project create for AT89C51 microcontroller.

Theory/ Calculation:-

Introduction:

Keil MDK is the complete software development environment for a range of Arm Cortex-M based microcontroller devices. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and essential middleware components. It supports all silicon vendors with more than 9,000 devices.

Steps:

1: Open the Keil software.

2: Click on Project menu and click on new Project

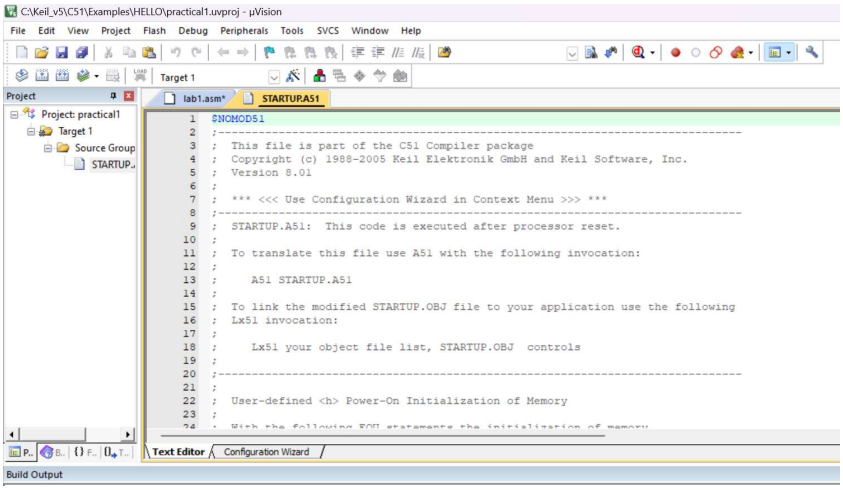
3: In the dialog box, in devices select on AT89C51, then click on Ok.

4: Click on File Menu, and create a new file and save it by using lab1.asm

5: Click on Source package on Project panel on left side and then click on add existing file

6: After clicking on OK a new file will be automatically created.

Result:- (Images/ Sentence's)



Conclusion:-

We learnt about Keil software and about how we can create projects for different microcontrollers.

Practical-1

Aim (ii):-

Introduction to Data Transfer, Logical and Arithmetic instructions. Explain all addressing mode for all the instruction with example for data transfer and logical e of Instruction.

Theory/ Calculation:-

Data transfer:

The **data transfer instructions** are used to transfer data from one location to another. This transfer of data can be either from register to register, register to memory or memory to register. e.g.: MOV, PUSH, POP.

Logical Instructions: The processor instruction set provides the instructions AND, OR, XOR, TEST, and NOT Boolean logic, which tests, sets, and clears the bits according to the need of the program.

Arithmetic Instruction:

The arithmetic instructions define the set of operations performed by the processor Arithmetic Logic Unit (ALU). The arithmetic instructions are further classified into binary, decimal, logical, shift/rotate, and bit/byte manipulation instructions.

Circuit Diagram :- (if required)

None.

Assembly / C Code :-

1. Immediate addressing mode:

mov a,#22H

2. Register addressing mode:

mov a,r0

3. Direct Addressing mode:

mov r0, 22H

4. Register indirect addressing mode

mov a, @r0

Other instructions like push, pop etc. can be written in similar way too.

AND:

Immediate: ANL A, #data

Register: ANL A, Rn

Direct: ANL A, Ri

Register Indirect : ANL A, @Ri

Other instructions like OR, XOR, etc. can be written in similar way too.

Conclusion:-

In this lab we learnt about Data Transfer, Logical and Arithmetic instructions, and different types of addressing modes.

Practical-1

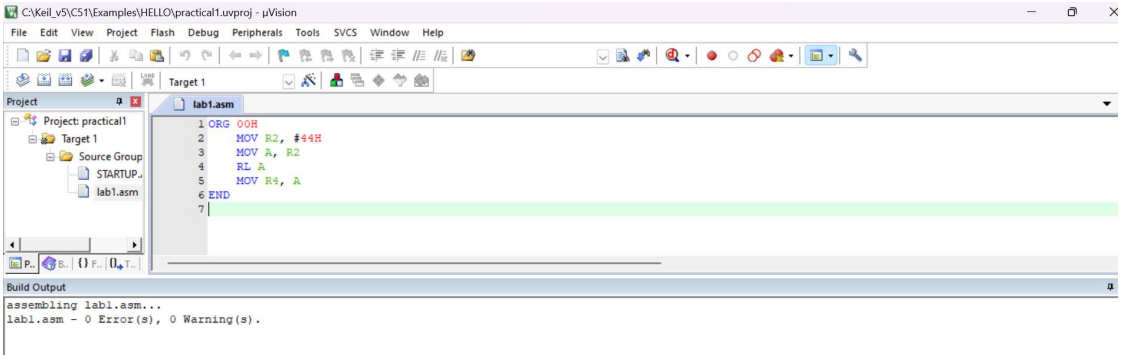
Aim 1 (iii):-

Write a program to double the 8-bit number Stored in R2. Store the result in to the R4.

Assembly / C Code :-

```
ORG 00H
        MOV R2, #44H
        MOV A, R2
        RL A
        MOV R4, A
END
```

Result:- (Images/ Sentence's)



Practical-1

Aim 1 (iv):-

Write a program to half the 8-bit number stored in R6. Store the result into the R6.

Theory/ Calculation:-

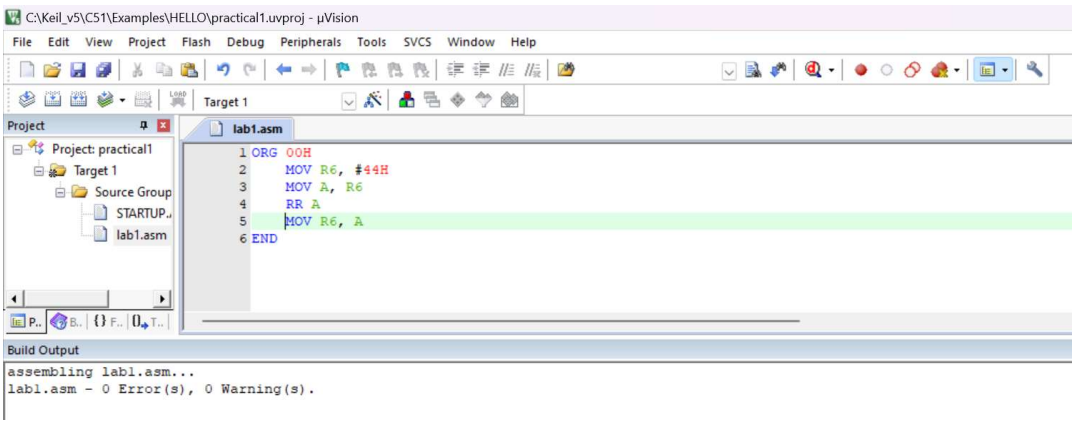
Circuit Diagram :- (if required)

None.

Assembly / C Code :-

```
ORG 00H
        MOV R6, #44H
        MOV A, R6
        RR A
        MOV R6, A
END
```

Result:- (Images/ Sentence's)



Practical-2

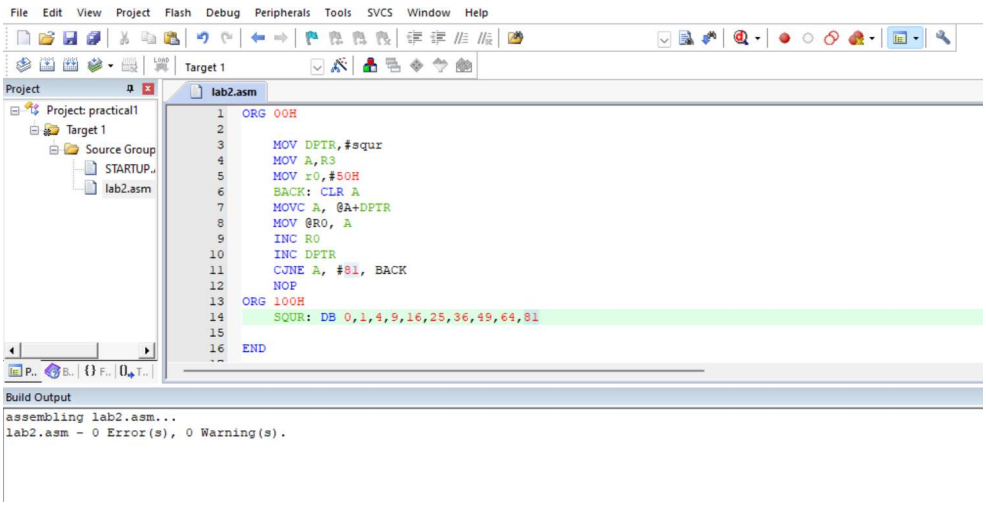
Aim (i):-

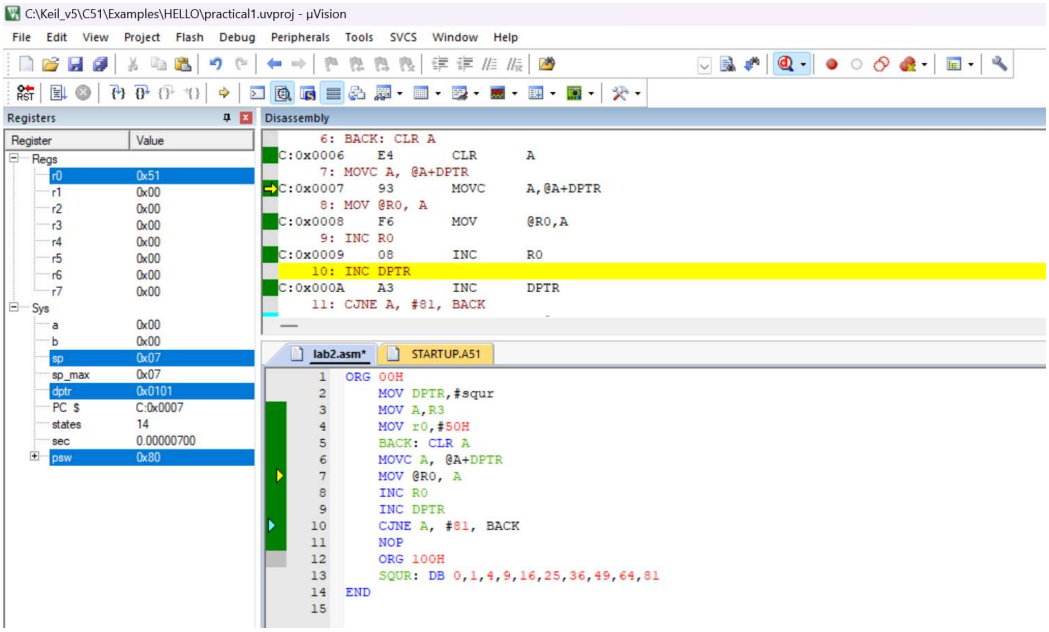
Write an assemble language code to get square of decimal number 0 to 9 using MOVC instruction. Square of each number is store in code memory starting from 100H. Store the result at memory location starting from 50H.

Assembly / C Code :-

```
ORG 00H
        MOV DPTR,#squr
        MOV A,R3
        MOV r0,#50H
        BACK: CLR A
        MOVC A, @A+DPTR
        MOV @R0, A
        INC R0
        INC DPTR
        CJNE A, #81, BACK
        NOP
        ORG 100H
        SQUR: DB 0,1,4,9,16,25,36,49,64,81
END
```

Result:- (Images/ Sentence's)




C:\Keil_v5\C51\Examples\HELLO\practical1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register Value
Regs
r0 0x51
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x00
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0101
PC $ C:0x0007
states 14
sec 0.00000700
psw 0x80
Disassembly
6: BACK: CLR A
C:0x0006 E4 CLR A
7: MOVC A, @A+DPTR
C:0x0007 93 MOVC A,@A+DPTR
8: MOV @R0, A
C:0x0008 F6 MOV @R0,A
9: INC R0
C:0x0009 08 INC R0
10: INC DPTR
C:0x000A A3 INC DPTR
11: CJNE A, #81, BACK
lab2.asm*
STARTUP.A51
1 ORG 00H
2 MOV DPTR,#squr
3 MOV A,R3
4 MOV r0,#50H
5 BACK: CLR A
6 MOVC A, @A+DPTR
7 MOV @R0, A
8 INC R0
9 INC DPTR
10 CJNE A, #81, BACK
11 NOP
12 ORG 100H
13 SQUR: DB 0,1,4,9,16,25,36,49,64,81
14 END
15

Practical-2

Aim (ii) :-

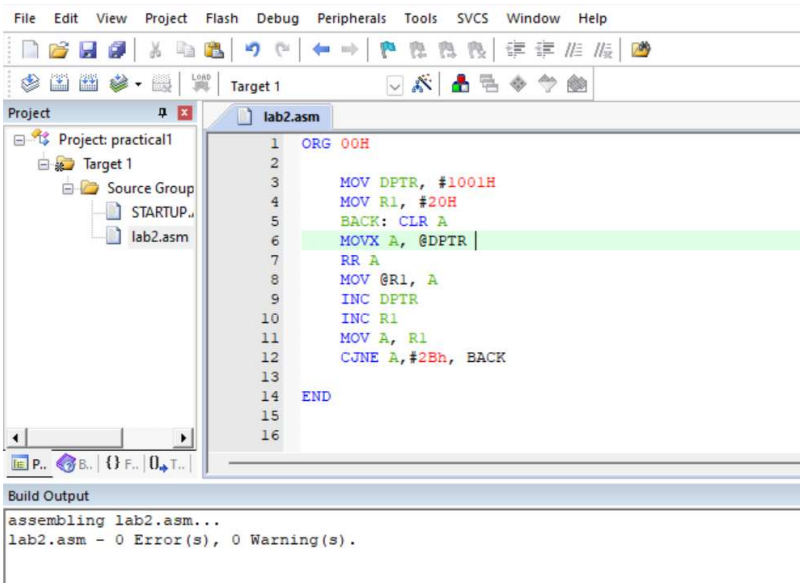
Write an assemble language code to get the 10 different 8 bit data from the external memory location and store the half of the each data to the direct memory location. consider External memory location starting at 1001H and direct memory location starting at 20H.

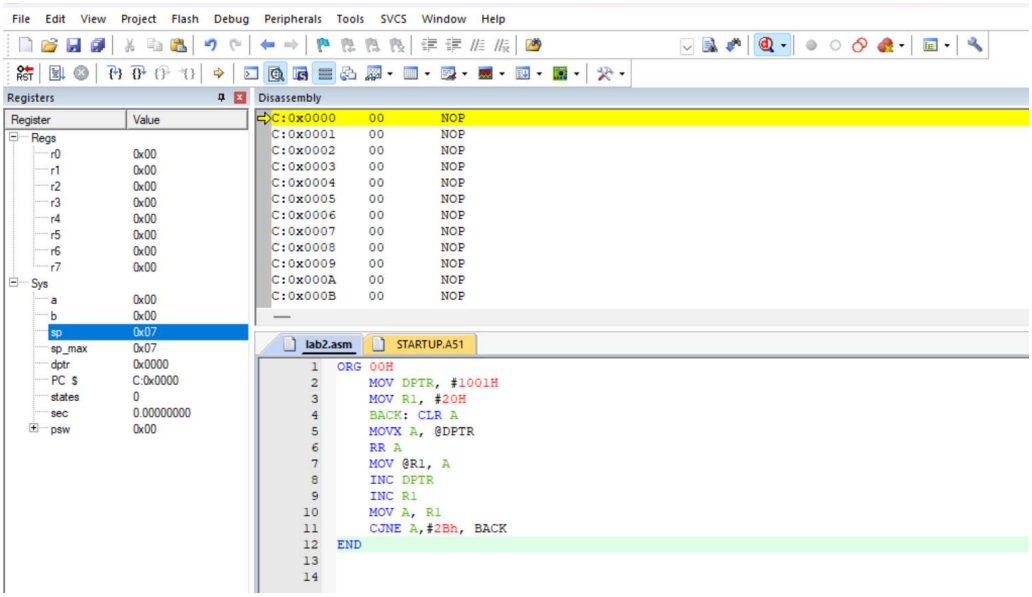
Assembly / C Code :-

```
ORG 00H
   MOV DPTR, #1001H
        MOV R1, #20H
        BACK: CLR A
        MOVX A, @DPTR
        RR A
        MOV @R1, A
        INC DPTR
        INC R1
        MOV A, R1
        CJNE A,#2Bh, BACK

END
```

Result:- (Images/ Sentence's)




File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register
Value
Regs
r0 0x00
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x00
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0000
PC $ C:0x0000
states 0
sec 0.00000000
psw 0x00
Disassembly
C:0x0000 00 NOP
C:0x0001 00 NOP
C:0x0002 00 NOP
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
C:0x000B 00 NOP
lab2.asm
STARTUP.A51
1 ORG 00H
2 MOV DPTR, #1001H
3 MOV R1, #20H
4 BACK: CLR A
5 MOVX A, @DPTR
6 RR A
7 MOV @R1, A
8 INC DPTR
9 INC R1
10 MOV A, R1
11 CJNE A,#2Bh, BACK
12 END
13
14

Practical-2

Aim (iii) :-

Invert the number stored at the R5 and save the inverted number into the R4.

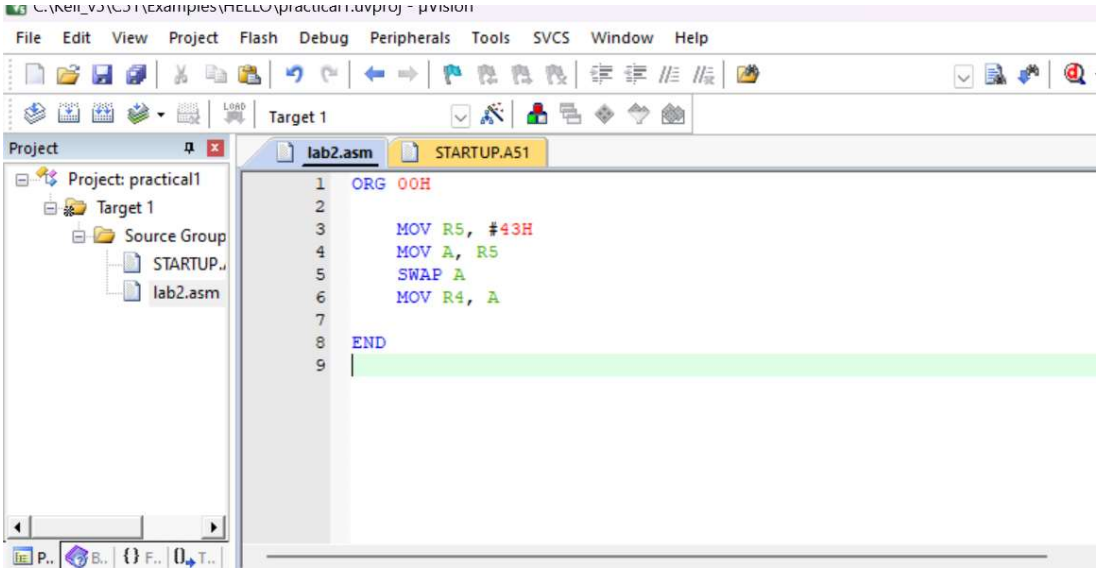
Assembly / C Code :-

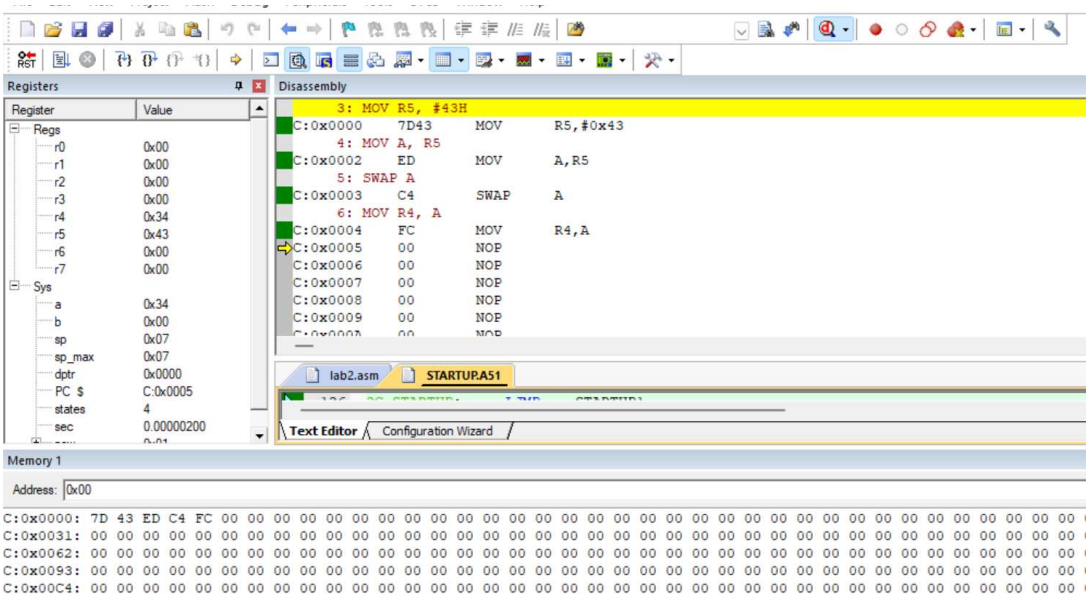
```
ORG 00H

    MOV R5, #43H
    MOV A, R5
    SWAP A
    MOV R4, A

END
```

Result:- (Images/ Sentence's)




Registers
Register
Value
Regs
r0 0x00
r1 0x00
r2 0x00
r3 0x00
r4 0x34
r5 0x43
r6 0x00
r7 0x00
Sys
a 0x34
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0000
PC $ C:0x0005
states 4
sec 0.00000200
Disassembly
3: MOV R5, #43H
C:0x0000 7D43 MOV R5,#0x43
4: MOV A, R5
C:0x0002 ED MOV A,R5
5: SWAP A
C:0x0003 C4 SWAP A
6: MOV R4, A
C:0x0004 FC MOV R4,A
C:0x0005 00 NOP
C:0x0006 00 NOP
C:0x0007 00 NOP
C:0x0008 00 NOP
C:0x0009 00 NOP
lab2.asm
STARTUP.A51
Text Editor
Configuration Wizard
Memory 1
Address: 0x00
C:0x0000: 7D 43 ED C4 FC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0031: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0062: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0093: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x00C4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Practical-3

Aim (i):-

Write a program to do Addition of 4 Different 8-bit unsigned number as given below. (Use Assembly Language) Store the result on R4 of Bank 0 and Carry to the R5 of Bank 0.

Registers Bank	Register	Value
0	R0	23H
1	R0	A2H
2	R0	ABH
3	R0	1AH

Theory/ Calculation:-

Circuit Diagram :- (if required)

None.

Assembly / C Code :-

```
ORG 00H
        MOV R0, #23H
        SETB PSW.3
        MOV R0, #0A2H
        SETB PSW.4
        CLR PSW.3
        MOV R0, #0ABH
        SETB PSW.4
        SETB PSW.3
        MOV R0, #1AH
        MOV A, R0
        CLR PSW.3
        ADD A,R0
        INC 05H
        SK1: CLR PSW.4
        SETB PSW.3
        ADD A,R0
        JNC SK2
        INC 05H
```

SK2: CLR PSW.3

ADD A,R0

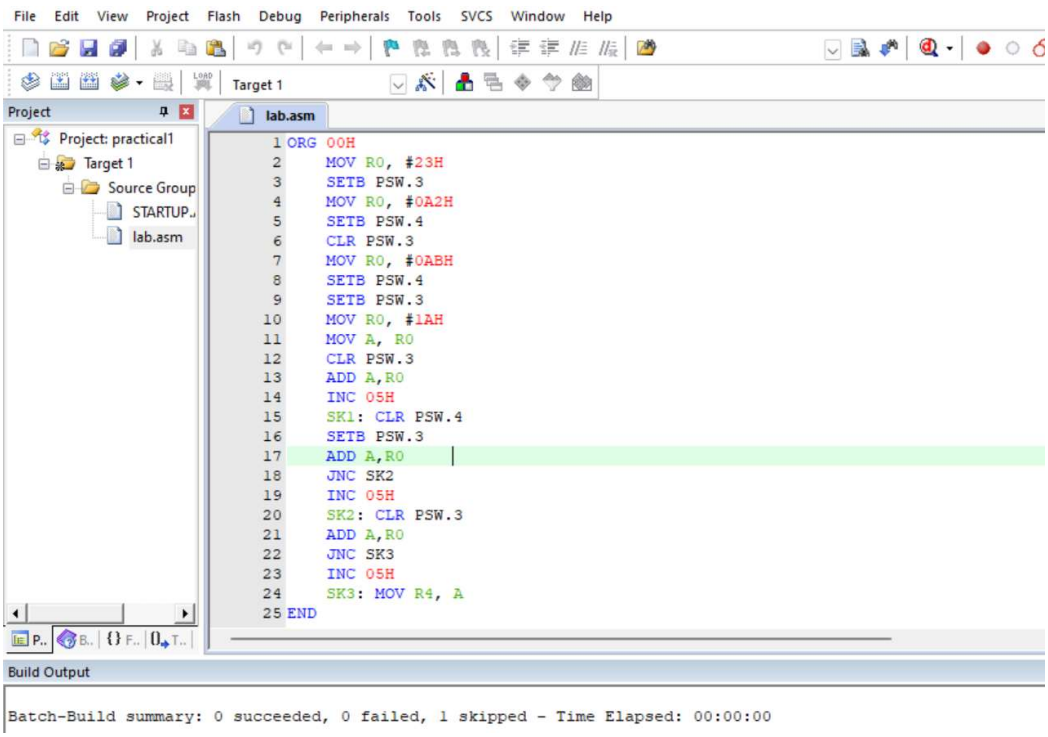
JNC SK3

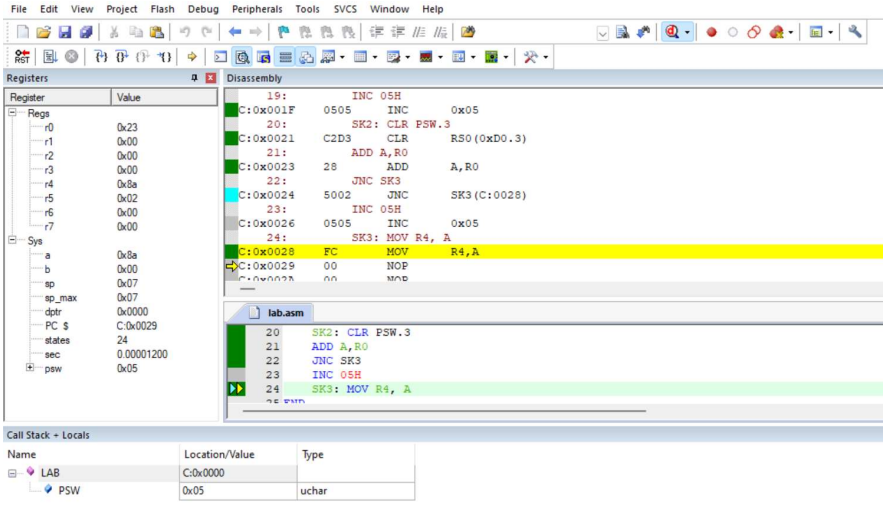
INC 05H

SK3: MOV R4, A

END

Result:- (Images/ Sentence's)




File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register
Value
Regs
r0 0x23
r1 0x00
r2 0x00
r3 0x00
r4 0x8a
r5 0x02
r6 0x00
r7 0x00
Sys
a 0x8a
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0000
PC $ C:0x0029
states 24
sec 0.00001200
psw 0x05
Disassembly
19: INC 05H
C:0x001F 0505 INC 0x05
20: SK2: CLR PSW.3
C:0x0021 C2D3 CLR RS0(0xD0.3)
21: ADD A,R0
C:0x0023 28 ADD A,R0
22: JNC SK3
C:0x0024 5002 JNC SK3(C:0028)
23: INC 05H
C:0x0026 0505 INC 0x05
24: SK3: MOV R4, A
C:0x0028 FC MOV R4,A
C:0x0029 00 NOP
lab.asm
20 SK2: CLR PSW.3
21 ADD A,R0
22 JNC SK3
23 INC 05H
24 SK3: MOV R4, A
Call Stack + Locals
Name
Location/Value
Type
LAB
C:0x0000
PSW
0x05
uchar

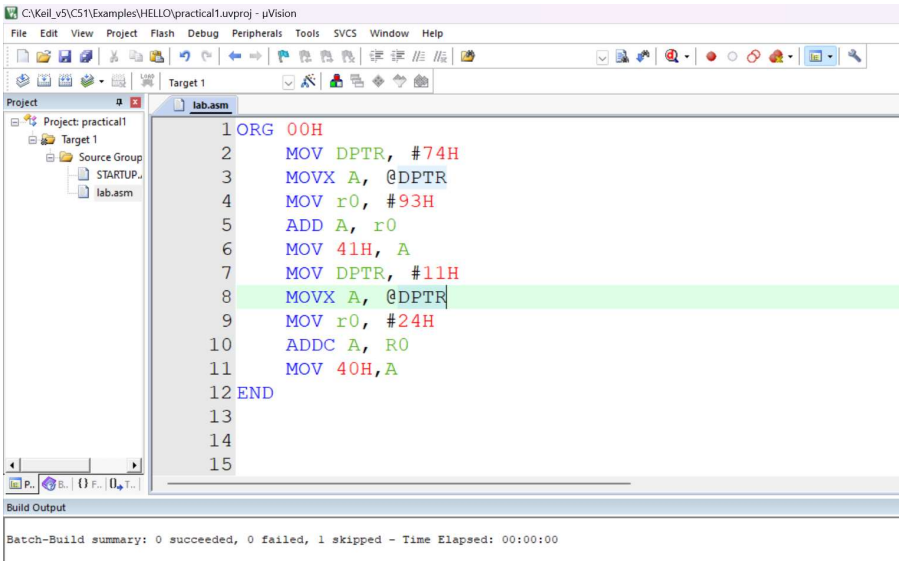
Practical 3

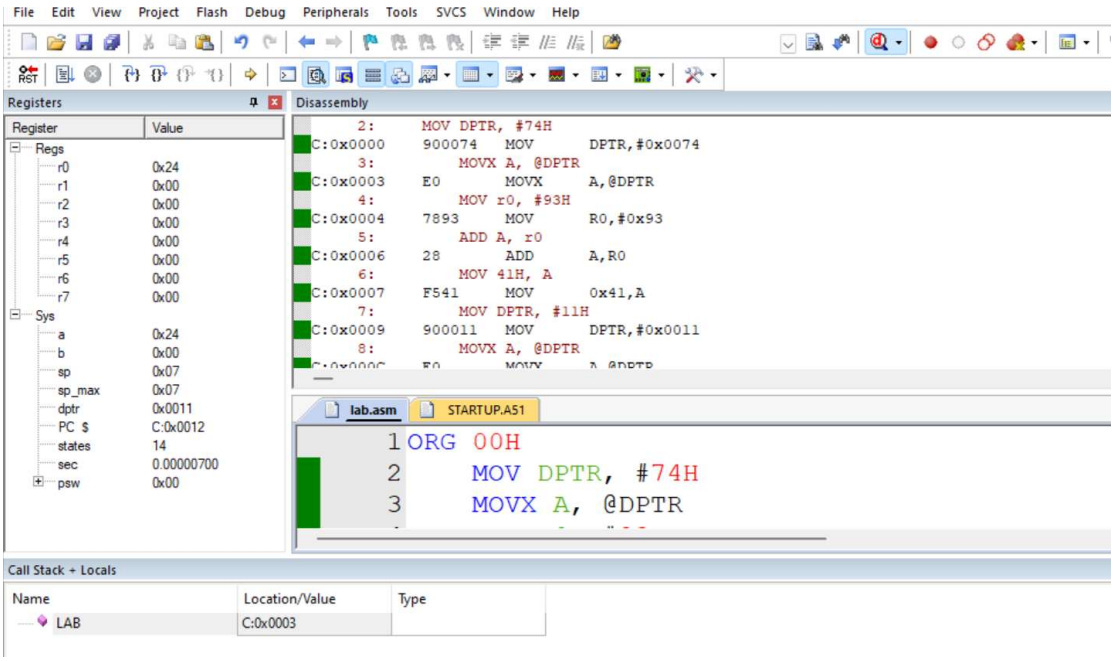
Aim (ii):-
Write an assembly language program to find 16 bit addition from two 16 bit numbers stored at internal and external memory.

Assembly / C Code:-

```
ORG 00H
   MOV DPTR, #74H
        MOVX A, @DPTR
        MOV r0, #93H
        ADD A, r0
        MOV 41H, A
        MOV DPTR, #11H
        MOVX A, @DPTR
        MOV r0, #24H
        ADDC A, R0
        MOV 40H,A
END
```

Result:-




File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register Value
Regs
r0 0x24
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x24
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0011
PC $ C:0x0012
states 14
sec 0.00000700
psw 0x00
Disassembly
2: MOV DPTR, #74H
C:0x0000 900074 MOV DPTR,#0x0074
3: MOVX A, @DPTR
C:0x0003 E0 MOVX A,@DPTR
4: MOV r0, #93H
C:0x0004 7893 MOV R0,#0x93
5: ADD A, r0
C:0x0006 28 ADD A,R0
6: MOV 41H, A
C:0x0007 F541 MOV 0x41,A
7: MOV DPTR, #11H
C:0x0009 900011 MOV DPTR,#0x0011
8: MOVX A, @DPTR
lab.asm
STARTUP.A51
1 ORG 00H
2 MOV DPTR, #74H
3 MOVX A, @DPTR
Call Stack + Locals
Name Location/Value Type
LAB C:0x0003

Practical 3

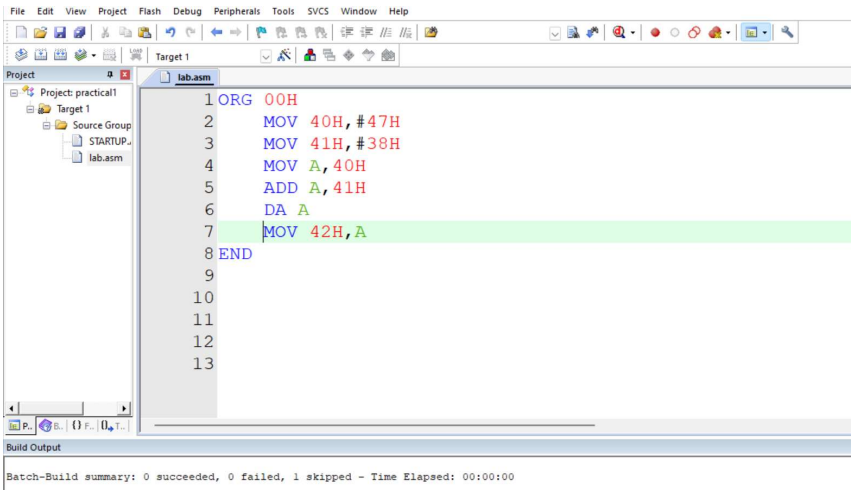
Aim (iii):-

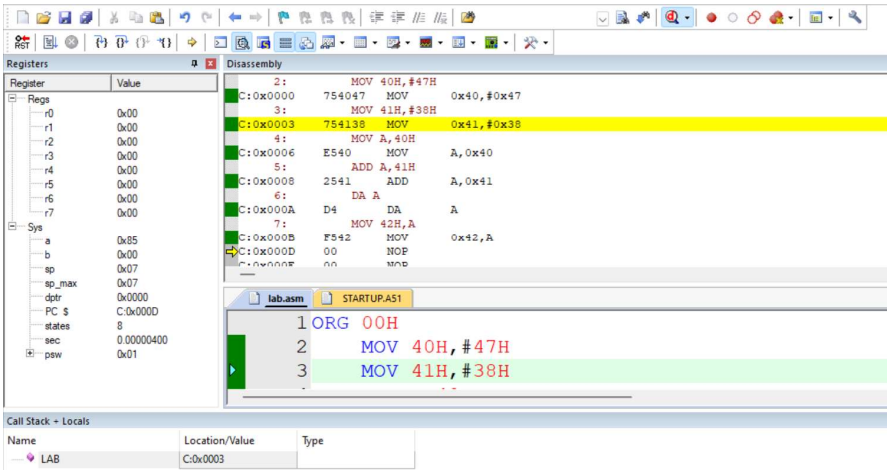
Write an assembly language program to add two different BCD number stored at memory location 40H and 41H. Stored the result at 42H after performing Decimal Adjustment on the result.

Assembly / C Code:-

```
ORG 00H
        MOV 40H,#47H
        MOV 41H,#38H
        MOV A,40H
        ADD A,41H
        DA A
        MOV 42H,A
END
```

Result:-





Practical 3

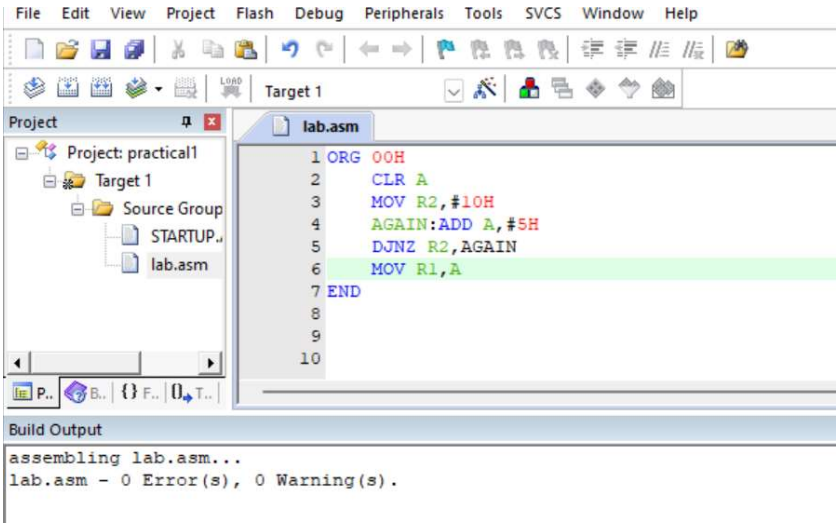
Aim (iv):-

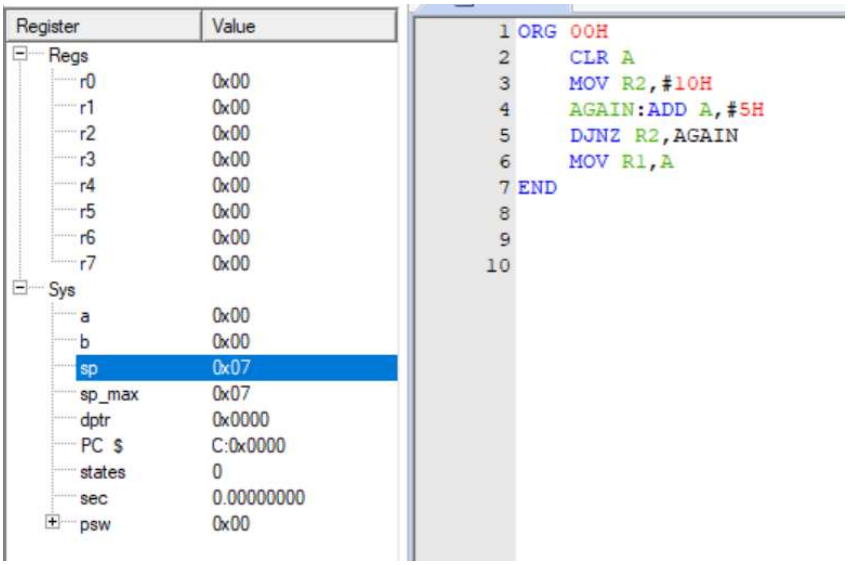
Write an assembly language code to multiply two 8 bit number using repeated addition method.

Assembly / C Code:-

```
ORG 00H
        CLR A
        MOV R2,#10H
        AGAIN:ADD A,#5H
        DJNZ R2,AGAIN
        MOV R1,A
END
```

Result:-





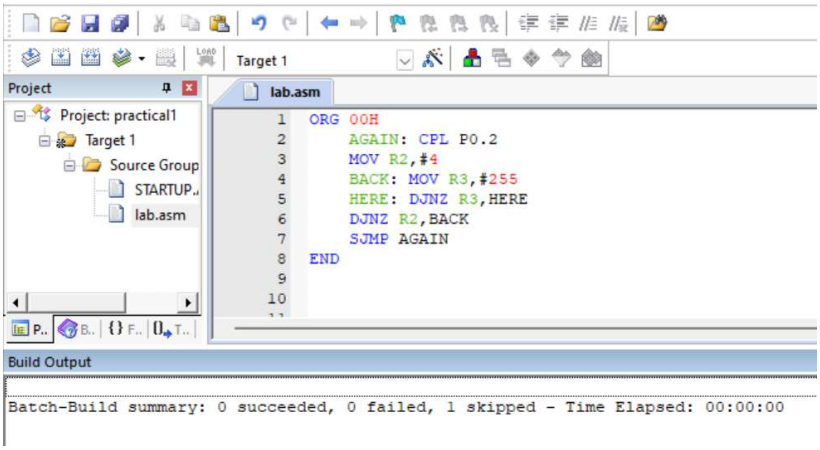
Practical 4

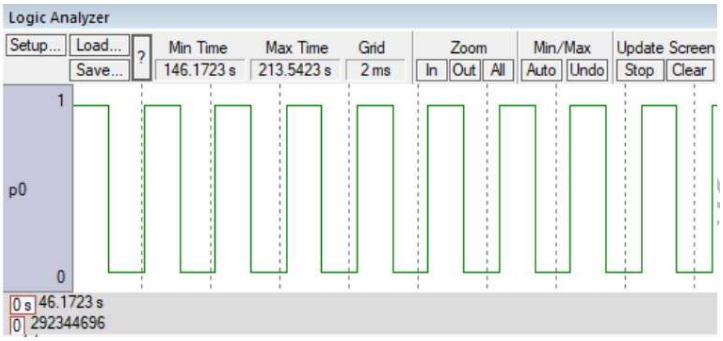
Aim (i):-
Write an ALP to toggle port pin 2 of port 0 with interval of 2 msec. (Use MOV and DJNZ to generate delay).(Assume that XTAL = 11.0592 MHz)

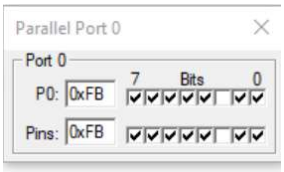
Assembly / C Code:-

```
ORG 00H
        AGAIN: CPL P0.2
        MOV R2,#4
        BACK: MOV R3,#255
        HERE: DJNZ R3,HERE
        DJNZ R2,BACK
        SJMP AGAIN
END
```

Result:-







Practical 4

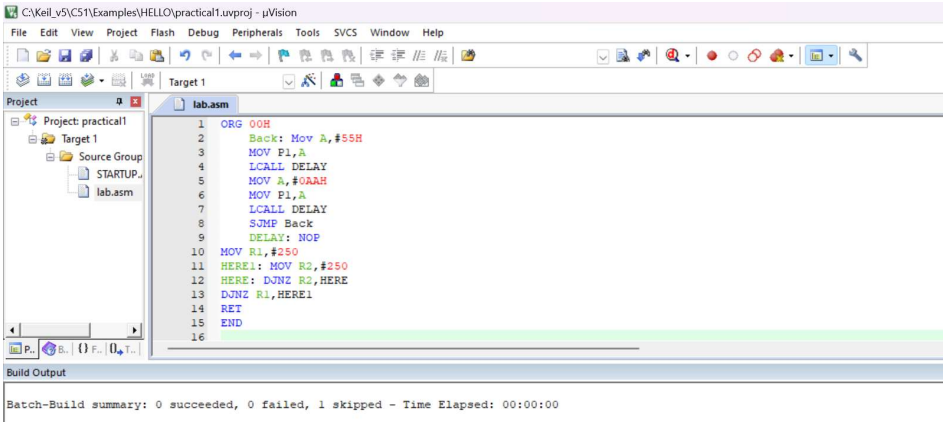
Aim (ii):-

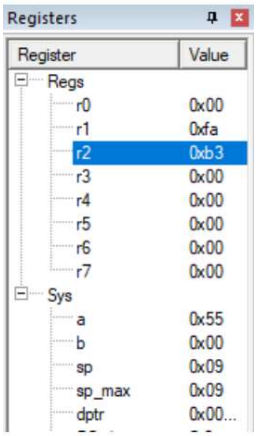
Write a program to toggle all the bits of the port 1 by sending to it values 55H and AAH continuously. Put a time delay in between each issuing of data to port 1. Use LCALL instruction to introduce time delay and SJMP for continuous operation.

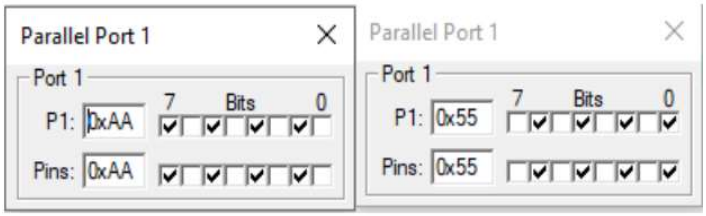
Assembly / C Code:-

```
ORG 00H
        Back: Mov A,#55H
        MOV P1,A
        LCALL DELAY
        MOV A,#0AAH
        MOV P1,A
        LCALL DELAY
        SJMP Back
        DELAY: NOP
MOV R1,#250
HERE1: MOV R2,#250
HERE: DJNZ R2,HERE
DJNZ R1,HERE1
RET
END
```

Result:-




Registers
Register
Value
Regs
r0 0x00
r1 0xfa
r2 0xb3
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x55
b 0x00
sp 0x09
sp_max 0x09
dptr 0x00...


Parallel Port 1
Port 1
P1: 0xAA
7 Bits 0
Pins: 0xAA
Parallel Port 1
Port 1
P1: 0x55
7 Bits 0
Pins: 0x55

Practical 4

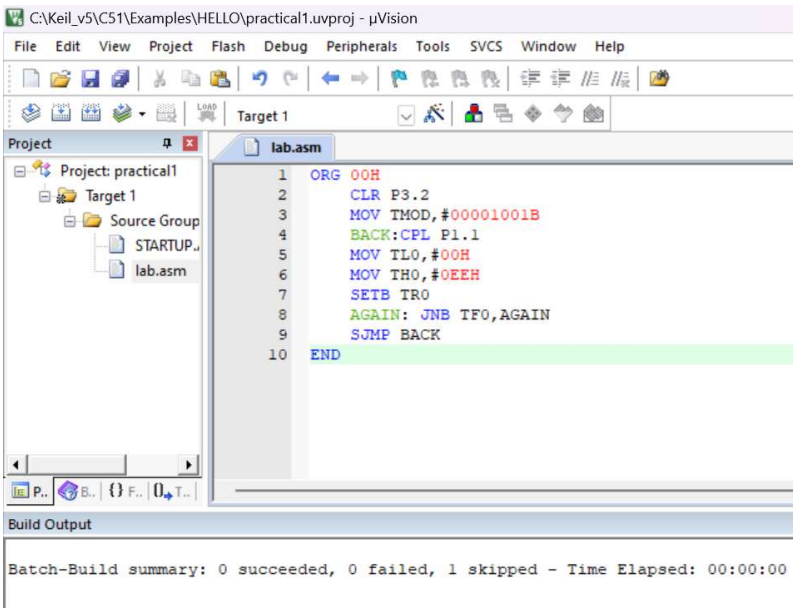
Aim (iii):-

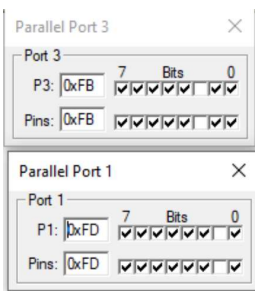
Write an assembly language program to generate square wave of 100Hz frequency on port pin 1 of port 1. (Use timer 0 in mode 1 with external hardware control) (Assume that XTAL = 11.0592 MHz)

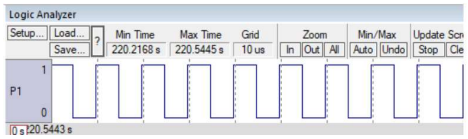
Assembly / C Code:-

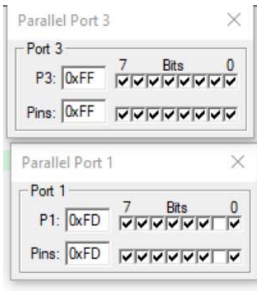
```
ORG 00H
        CLR P3.2
        MOV TMOD,#00001001B
        BACK:CPL P1.1
        MOV TL0,#00H
        MOV TH0,#0EEH
        SETB TR0
        AGAIN: JNB TF0,AGAIN
        SJMP BACK
END
```

Result:-









Practical 4

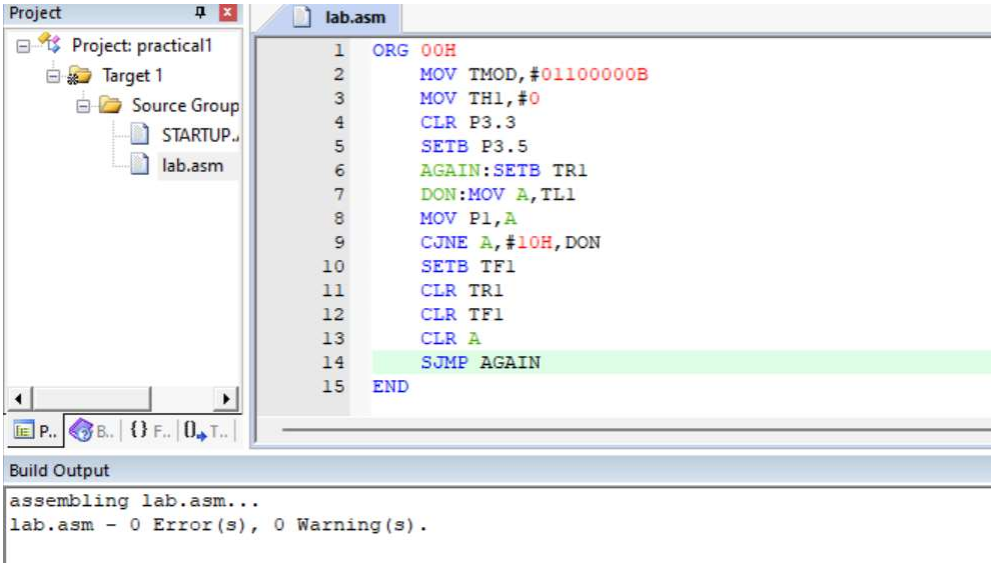
Aim (iv):-

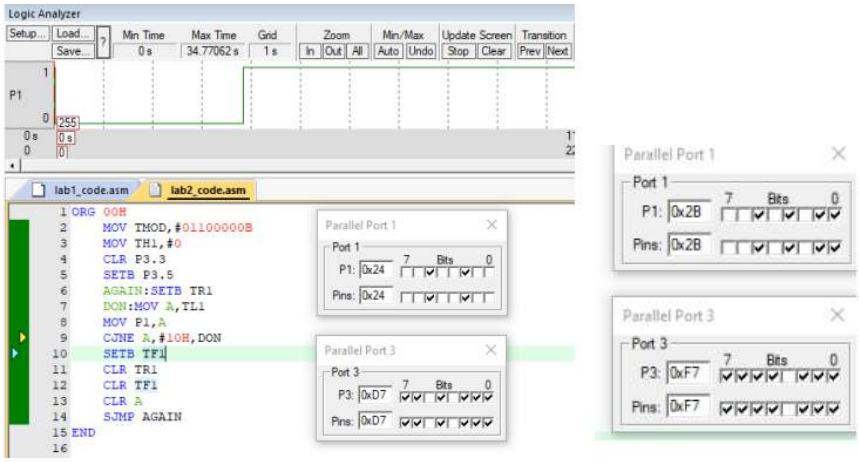
Write an assembly language program to generate square wave of 100Hz frequency on port pin 1 of port 1. (Use timer 0 in mode 1 with external hardware control) (Assume that XTAL = 11.0592 MHz)

Assembly / C Code:-

```
ORG 00H
        MOV TMOD,#01100000B
        MOV TH1,#0
        CLR P3.3
        SETB P3.5
        AGAIN:SETB TR1
        DON:MOV A,TL1
        MOV P1,A
        CJNE A,#10H,DON
        SETB TF1
        CLR TR1
        CLR TF1
        CLR A
        SJMP AGAIN
END
```

Result:-




Logic Analyzer
Setup...
Load...
Save...
Min Time
0 s
Max Time
34.77062 s
Grid
1 s
Zoom
In
Out
All
Min/Max
Auto
Undo
Update Screen
Stop
Clear
Transition
Prev
Next
P1
255
lab1_code.asm
lab2_code.asm
1 ORG 00H
2 MOV TMOD,#01100000B
3 MOV TH1,#0
4 CLR P3.3
5 SETB P3.5
6 AGAIN:SETB TR1
7 DON:MOV A,TL1
8 MOV P1,A
9 CJNE A,#10H,DON
10 SETB TF1
11 CLR TR1
12 CLR TF1
13 CLR A
14 SJMP AGAIN
15 END
16
Parallel Port 1
Port 1
P1: 0x24
Pins: 0x24
Parallel Port 3
Port 3
P3: 0xD7
Pins: 0xD7
Parallel Port 1
Port 1
P1: 0x2B
Pins: 0x2B
Parallel Port 3
Port 3
P3: 0xF7
Pins: 0xF7
7 Bits 0

Practical 4

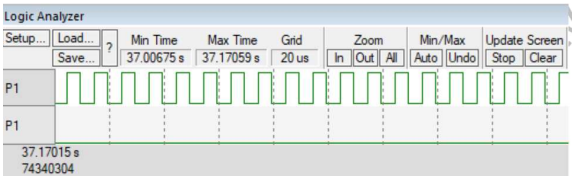
Aim (v):-

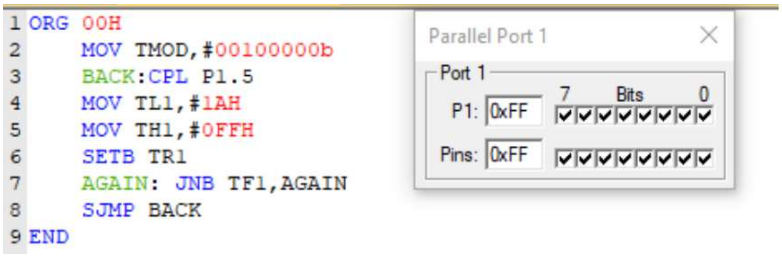
Write an assembly language program to generate a square wave of 2 kHz frequency on pin P1.5.

Assembly / C Code:-

```
ORG 00H
MOV TMOD,#00100000b
BACK:CPL P1.5
MOV TL1,#1AH
MOV TH1,#0FFH
SETB TR1
AGAIN: JNB TF1,AGAIN
SJMP BACK
END
```

Result:-





Practical 5

Aim (i):-

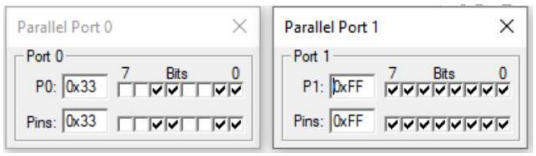
Write a C program to read the P1.0 and P1.1 bits and issue as ASCII character to P0 according to the following table.

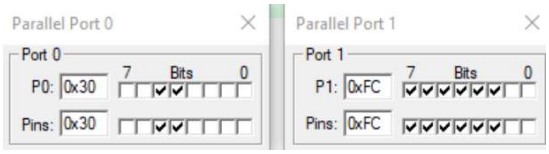
P1.1	P1.0	Data to be send on Port 0
0	0	Send "0"
0	1	Send "1"
1	0	Send "2"
1	1	Send "3"

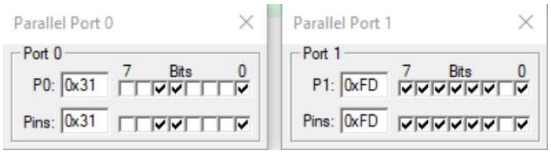
Assembly / C Code:-

```
#include <reg51.h>
void main(void){
        unsigned char z;
        z = P1;
        z= z & 0x3;
        switch(z){
                case(0):{
                        P0 = '0';
                        break;
                }
                case(1):{
                        P0 = '1';
                        break;
                }
                case(2):{
                        P0 = '2';
                        break;
                }
                case(3):{
                        P0 = '3';
                        break;
                }
        }
}
```

Result:-


Parallel Port 0
Port 0
P0: 0x33
7 Bits 0
Pins: 0x33
Parallel Port 1
Port 1
P1: 0xFF
7 Bits 0
Pins: 0xFF


Parallel Port 0
Port 0
P0: 0x30
7 Bits 0
Pins: 0x30
Parallel Port 1
Port 1
P1: 0xFC
7 Bits 0
Pins: 0xFC


Parallel Port 0
Port 0
P0: 0x31
7 Bits 0
Pins: 0x31
Parallel Port 1
Port 1
P1: 0xFD
7 Bits 0
Pins: 0xFD

Practical 5

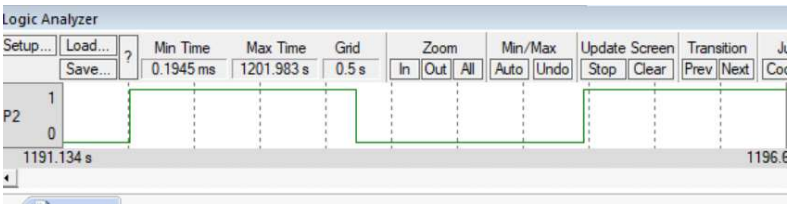
Aim (ii):-

Write a 8051 C code to generate Square wave with period of 500ms on Port Pin P2.0

Assembly / C Code:-

```
#include <reg51.h>
void MSDelay(unsigned int);
sbit MYBIT=P2^0;
void main(void)
{
while(1){
                MYBIT=0;
                MSDelay(500);
                MYBIT=1;
                MSDelay(500);
        }
}
void MSDelay(unsigned int time)
{
                unsigned int i,j;
                for (i=0;i<time;i++)
                for (j=0;j<1275;j++);
}
```

Result:-



Practical 5

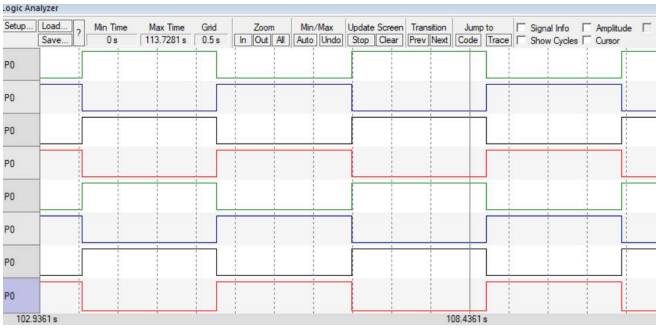
Aim (iii):-

Write a C program to toggle all the bit's of the Port P0 continuously with a 500 ms delay. Use function (without using timer) to generate the require delay.

Assembly / C Code:-

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while (1)
{
P0=0x55;
MSDelay(500);
P0=0xAA;
MSDelay(500);
}
}
void MSDelay(unsigned int time)
{
                unsigned int i,j;
                for (i=0;i<time;i++)
                for (j=0;j<1275;j++);
}
```

Result:-



Practical 5

Aim (iv):-

Write an 8051 C program to generate square wave of 100 ms on P1.5. Use timer 1, mode-1 to create the Delay (without using timer Interrupt).

Assembly / C Code:-

```
#include <reg51.h>
void MSDelay(void);
sbit mybit=P1^5;
void main(void){
while (1)
{
mybit=~mybit;
        MSDelay();
}
}
void MSDelay(void){
TMOD=0x10;
TL1=0xFD;
TH1=0x4B;
TR1=1;
while (TF1==0);
TR1=0;
TF1=0;
}
```

Result:-

Practical 5

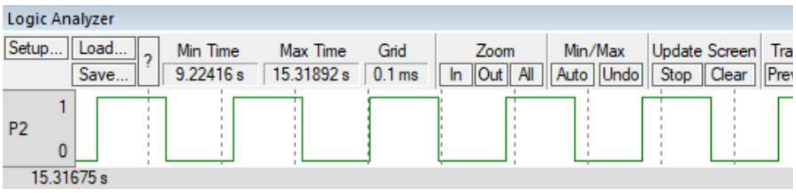
Aim (v):-

Write a C Program using timer interrupt to generate 10KHz Frequency on P2.1 using timer 0 in auto-reload mode.

Assembly / C Code:-

```
#include <reg51.h>
sbit mybit = P2 ^ 1;
void timer0() interrupt 1
{
   WAVE = ~WAVE
}
void main()
{
   TMOD = 0x42;
   TH0 = 0x46;
   IE = 0x86;
   TR0 = 1;
   while (1);
}
```

Result:-



Practical 6

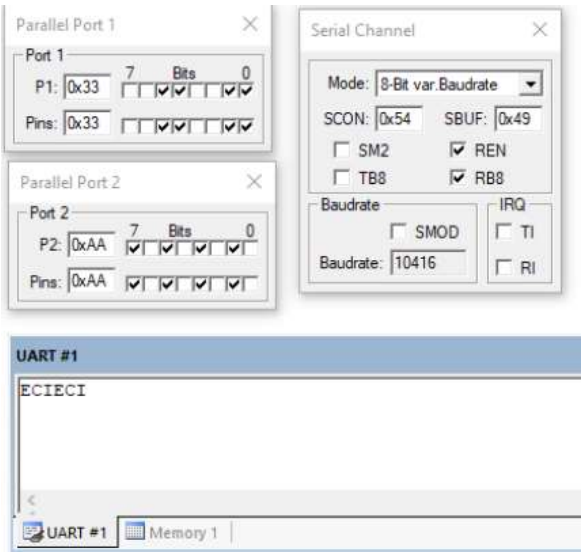
Aim (i):-

Write a C code to send data serially based on the UART received data. (XTAL= 11.0592 MHZ) i. If the received data is “1” then serially send “E”. ii. If the received data is “2” then Serially send “C”. iii. If the received data is “3” then Serially send “I”.

Assembly / C Code:-

```
#include<reg51.h>
void SerTx (unsigned char);
void main(void){
unsigned char mybyte;
TMOD=0x20;
        TH1=0xFA;
SCON=0x50;
TR1=1;
P1=0x00;
while(1){
while(RI==0);
mybyte=SBUF;
P1=mybyte;
RI=0;
if(mybyte=='1'){
P2=0xAA;
SerTx('E');
}
if(mybyte=='2'){
P2=0xAA;
SerTx('C');
}
if(mybyte=='3'){
P2=0xAA;
SerTx('I');
}
}
}
void SerTx(unsigned char x){
SBUF=x;
while(TI==0);
TI=0;
}
```

Result:-


Parallel Port 1
Port 1
P1: 0x33
7 Bits 0
Pins: 0x33
Parallel Port 2
Port 2
P2: 0xAA
7 Bits 0
Pins: 0xAA
Serial Channel
Mode: 8-Bit var.Baudrate
SCON: 0x54
SBUF: 0x49
SM2
REN
TB8
RB8
Baudrate
SMOD
Baudrate: 10416
IRQ
TI
RI
UART #1
ECIECI
UART #1
Memory 1

Practical 6

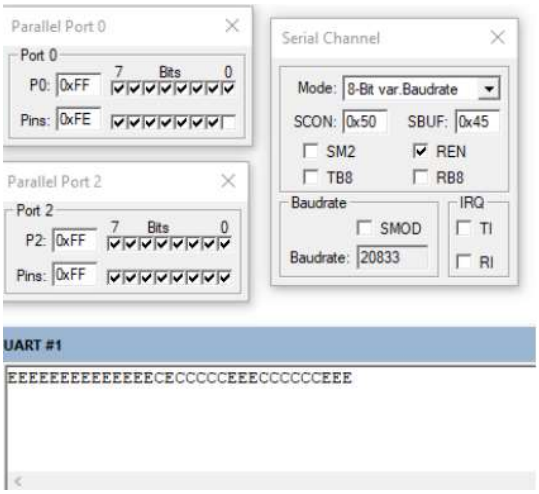
Aim (ii):-
Write a 8051 C code to send data to IO Port with below conditions. (XTAL= 11.0592 MHZ)

i. If P2.0 is one (HIGH) then send data “E” to Port 1 else send data “C” to Port Also generate square wave with 200 µsec of period on pin Port 0.0.Note: Use timer 0 interrupt in 8 bit mode to generate square.

Assembly / C Code:-

```
#include<reg51.h>
sbit wave=P0^0;
sbit sw=P2^0;
void MSDelay(unsigned int);
void timer0(void) interrupt 1{
wave=~wave;
}
void serial0(void) interrupt 4{
if(TI==1){
        TI=0;
}
}
void main(){
TMOD=0x22;
TH1=-3;
TH0=0xA1;
IE=0x92;
SCON=0x50;
TR1=1;
TR0=1;
while(1){
if(sw==1){
SBUF='E';
MSDelay(10);
}
else{
SBUF='C';
MSDelay(10);
}
MSDelay(1000);
}
}
void MSDelay(unsigned int time){
unsigned int i,j;
for(i=0;i<time;i++){
for(j=0;j<1275;j++);
}
}
```

Result:-


Parallel Port 0
Port 0
P0: 0xFF
7 Bits 0
Pins: 0xFE
Parallel Port 2
Port 2
P2: 0xFF
7 Bits 0
Pins: 0xFF
Serial Channel
Mode: 8-Bit var.Baudrate
SCON: 0x50
SBUF: 0x45
SM2
REN
TB8
RB8
Baudrate
SMOD
Baudrate: 20833
IRQ
TI
RI
UART #1
EEEEEEEEEEEEEECECCCCCEEECCCCCCEEE

Practical 6

Aim (iii):-

Write a C program using timer 0 as interrupt for Square wave generation and Serial interrupt for Serial communication to do following conditions. a. When Switch 1 is presses it generate 5 KHz square wave on Port Pin P1.1 continuously. Also send '1" on Serial Port. b. When Switch 2 is presses it generate 10 KHz square ware on Port Pin P1.1 continuously. Also send '2" on Serial Port. c. When both Switches Press then "N" is send serially and if both switches are open send "O" Serially. Assume switch 1 is connected on P0.1 and switch 2 is connected to P0.2. Provide Delay 1000ms in between switch scanning and also gives delay of 10ms after each condition check.(XTAL = 11.0592 MHz, Set the baud rate at 9600)

Assembly / C Code:-

```
#include<reg51.h>
sbit wave=P1^1;
sbit sw1=P0^1;
sbit sw2=P0^2;
void MSDelay(unsigned int);
void timer0(void) interrupt 1{
wave=~wave;
}
void serial0(void) interrupt 4{
if(TI==1){
TI=0;
        }
}
void main(){
TMOD=0x22;
SCON=0x50;
TH1=-3;
TH0=36;
IE=0x92;
TR0=1;
while(1){
if(sw1==0 && sw2==0){
SBUF='0';
TR1=1;
MSDelay(10);
}
else if(sw1==1 && sw2==0){
SBUF='1';
TR1=1;
MSDelay(10);
}else if(sw1==1 && sw2==0){
SBUF='1';
TH0=36; TR1=1;
        MSDelay(10);
}
else if(sw1==0 && sw2==1){
SBUF='2';
```

```
TH0=72;
TR1=1;
MSDelay(10);
}
else if(sw1==1 && sw2==1){
SBUF='N';
TR1=1;
MSDelay(10);
}
MSDelay(1000);
}
}
void MSDelay(unsigned int time){
unsigned int i,j;
for(i=0;i<time;i++){
for(j=0;j<1275;j++);
}
}
```

Result:-