BLINKING OF LED USING 8051 MICROCONTROLLER USING PROTEUS

AIM:

To Write an assembly language program to LED blink using 8051

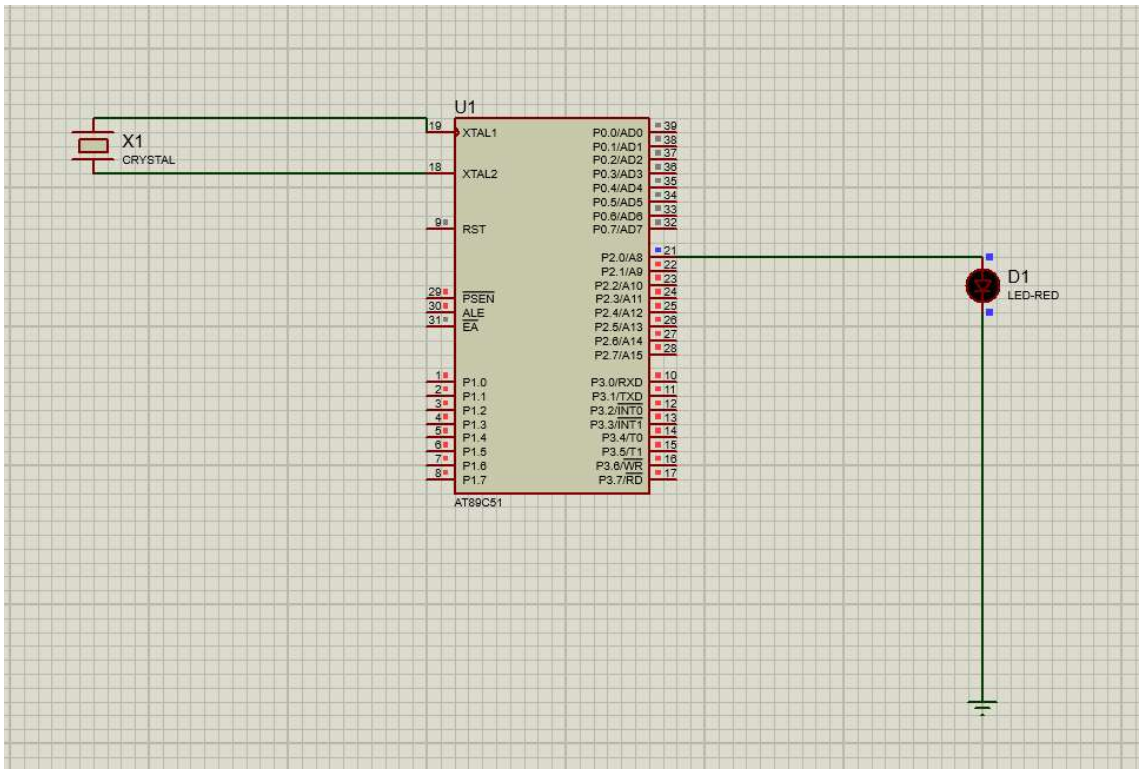
SOFTWARES REQUIRED:

- Proteus software

PROGRAM

```
        ORG 0000H
UP: SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY: MOV R4,#35
H1:MOV R3,#255
H2:DJNZ R3,H2
        DJNZ R4,H1
        RET
        END
```

CIRCUIT DIAGRAM:



RESULT

Thus the program has been successfully verified and executed.

LED TOGGLE USING 8051 USING PROTEUS

AIM:

Write an assembly language program for LED Toggle Using 8051 using Keil and Proteus

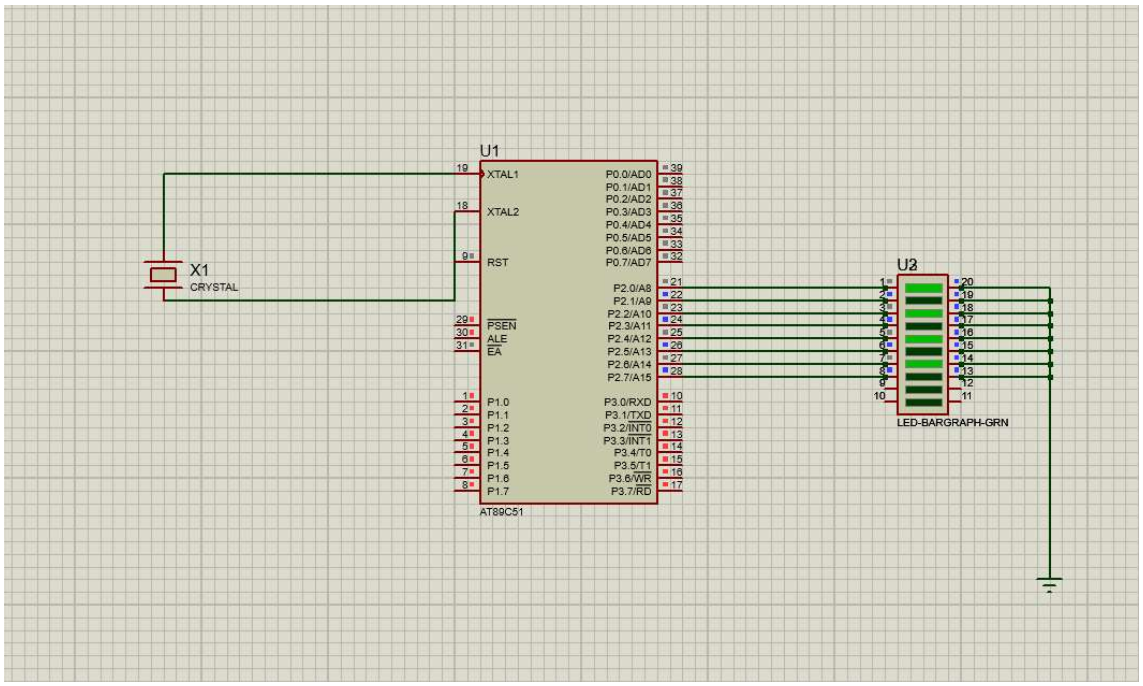
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#55H
ACALL DELAY
MOV P2,#0AAH
ACALL DELAY
SJMP UP

DELAY:MOV R4,#10
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

LED CHASER USING 8051 USING PROTEUS

AIM:

Write an assembly language program for LED Chaser Using 8051 using Keil and Proteus

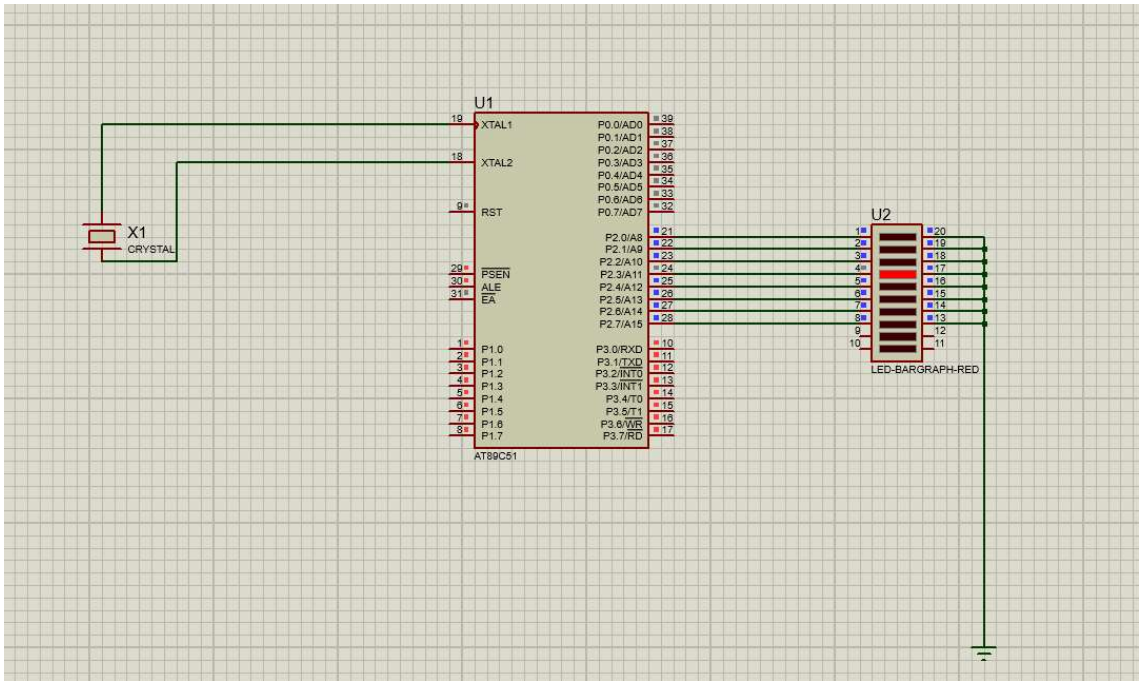
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#01H
ACALL DELAY
MOV P2,#02H
ACALL DELAY
MOV P2,#04H
ACALL DELAY
MOV P2,#08H
ACALL DELAY
MOV P2,#10H
ACALL DELAY
MOV P2,#20H
ACALL DELAY
MOV P2,#40H
ACALL DELAY
MOV P2,#80H
ACALL DELAY
SJMP UP

DELAY: MOV R4,#255
H1: DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

FADE IN FADE OUT OF LED USING 8051 USING PROTEUS

AIM:

To write an assembly language program for Fade in Fade out of LED Using 8051 using Keil and Proteus.

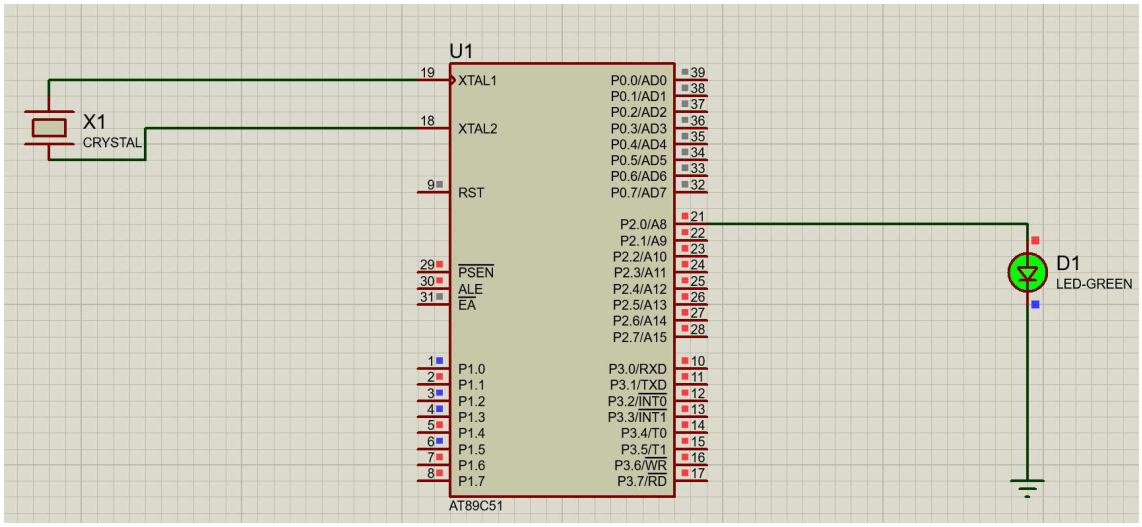
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 00H

MOV P1, #00H ; Initialize P1 to 0 (LED off)

FADING_LOOP:
   MOV R0, #0FFH ; Initialize R0 to 0xFF (highest intensity)

FADING_DOWN:
   MOV P1, R0
   ACALL DELAY ; Call delay subroutine
   DJNZ R0, FADING_DOWN ; Decrement R0 and loop until 0

   MOV R0, #00H ; Initialize R0 to 0 (lowest intensity)

FADING_UP:
   MOV P1, R0
   ACALL DELAY ; Call delay subroutine
   INC R0
   CJNE R0, #0FFH, FADING_UP ; Loop until R0 reaches 0xFF

SJMP FADING_LOOP ; Repeat the fading process

DELAY:
   MOV R1, #0FFH
   LOOP1:
      MOV R2, #0FFH
      LOOP2:
         NOP
      DJNZ R2, LOOP2
   DJNZ R1, LOOP1
   RET
    END
```

CIRCUIT DIAGRAM:



OUTPUT:

The brightness of the LED is gradually increasing and decreasing with 1000ms delay.

RESULT:

Thus, the program has been successfully verified and executed

GENERATION OF SQUARE WAVE USING PROTEUS

AIM:

To write an assembly language program to generate square wave using 8051.

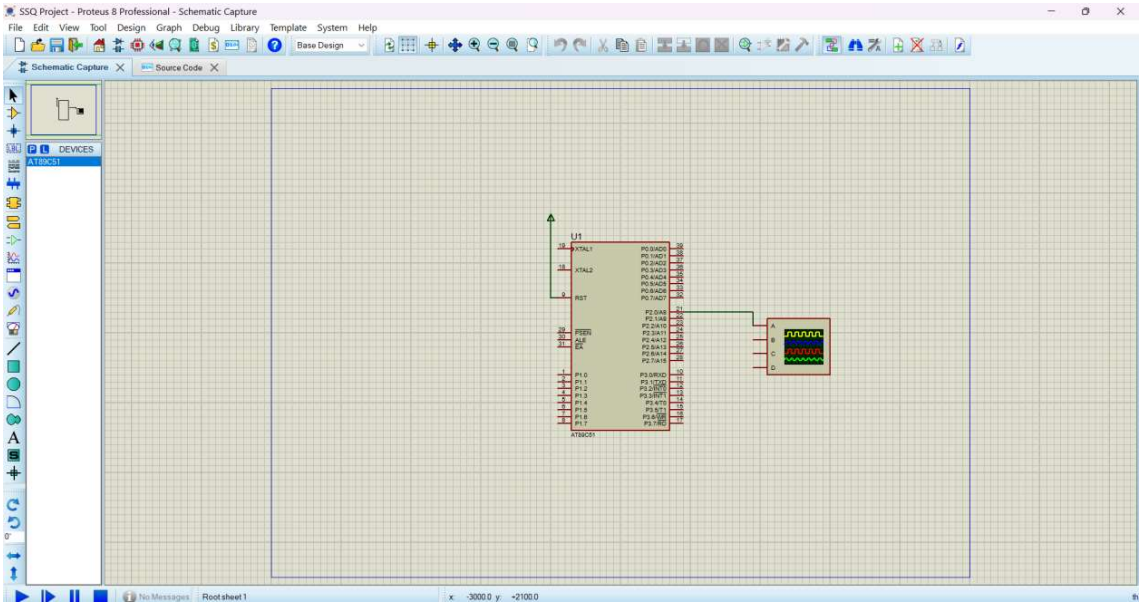
SOFTWARE REQUIRED:

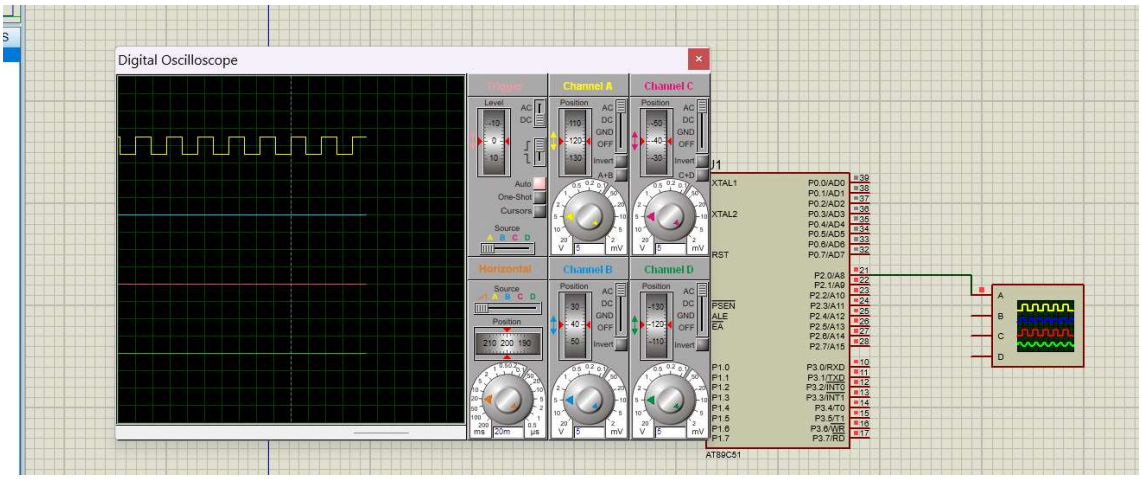
- Proteus 8 software.

PROGRAM

```
        ORG 0000H
UP:     SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY:  MOV R4,#35
H1:     MOV R3,#255
H2:     DJNZ R3,H2
        DJNZ R4,H1
        RET
        END
```

CIRCUIT DIAGRAM:





- Adjust the frequency in horizontal and channel A to Observe the square waveform on the oscilloscope for a consistent, symmetrical shape.

RESULT:

Thus the program has been successfully verified and executed using Proteus 8.10.

INTERFACING OF RELAY AND LED WITH 8051 USING PROTEUS

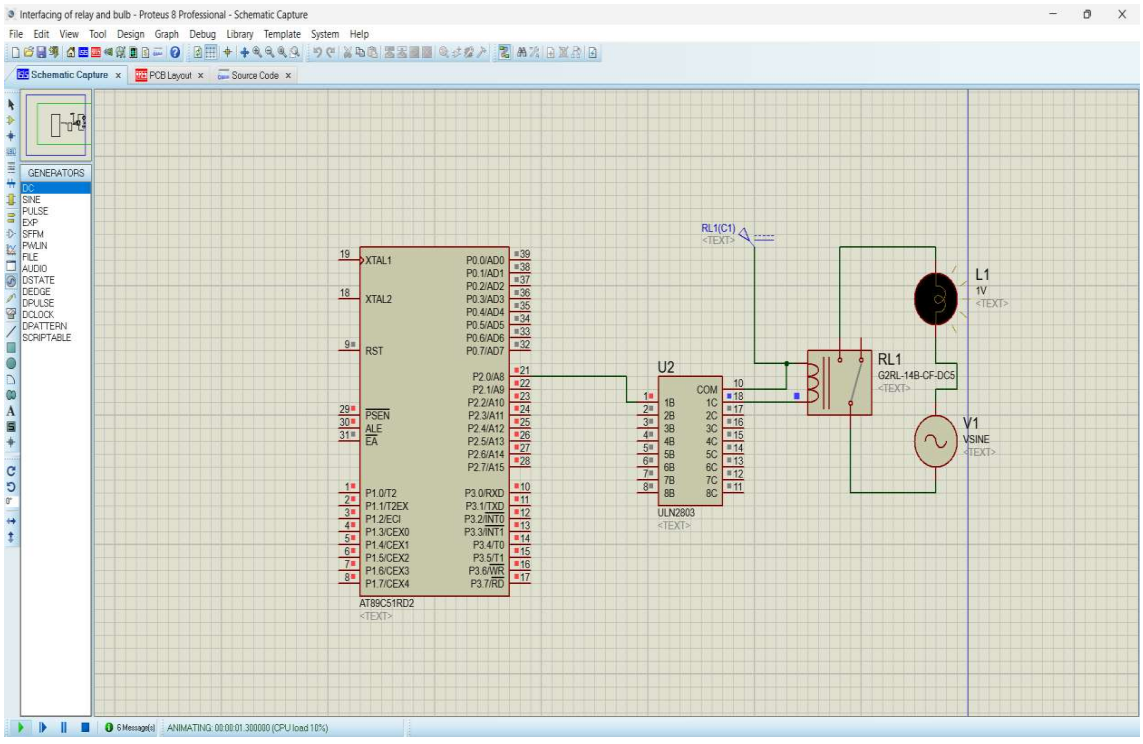
AIM:

To write an assembly language code for interfacing relay and led using Proteus 8

PROGRAM:

```
        ORG 0000H
    UP: SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY: MOV R4,#18
    H1: MOV R3,#255
    H2: DJNZ R3,H2
        DJNZ R4,H1
        RET
        END
```

OUTPUT:



RESULT:

Thus the program has been successfully verified and executed using Proteus 8.10.

GENERATION OF SQUARE WAVE USING PROTEUS

AIM:

To write an assembly language program to generate square wave using 8051.

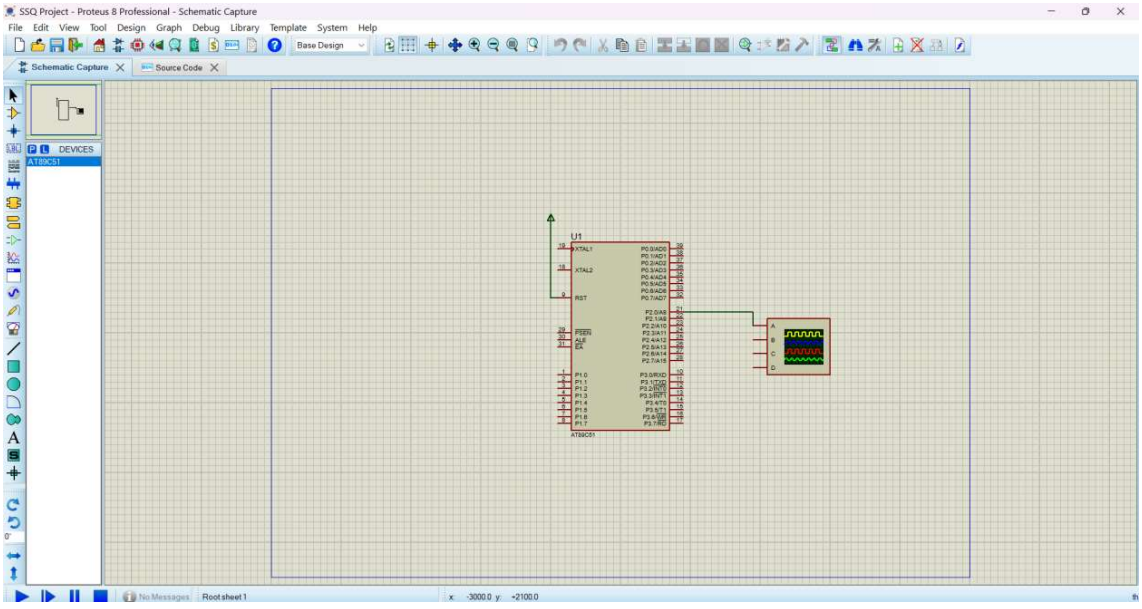
SOFTWARE REQUIRED:

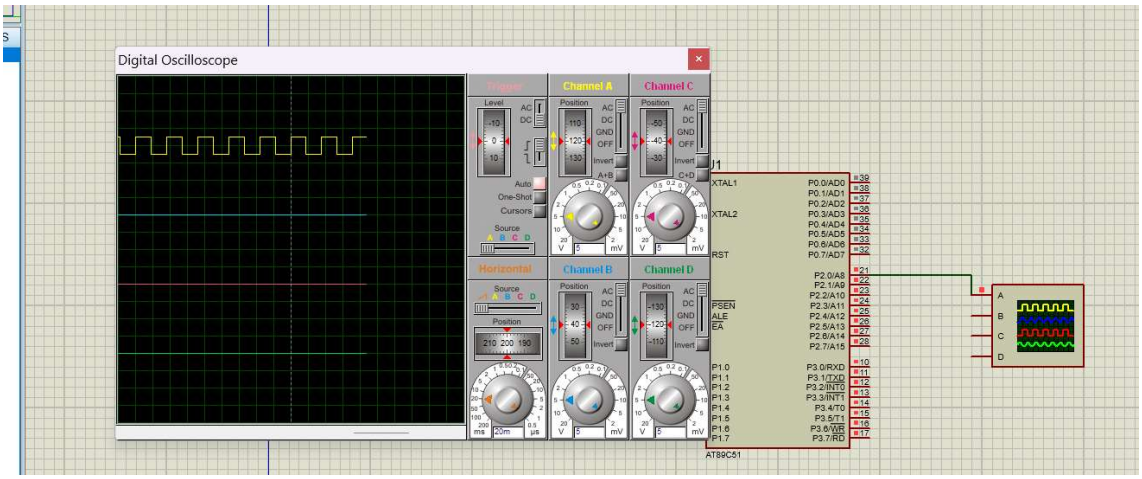
- Proteus 8 software.

PROGRAM

```
        ORG 0000H
UP:     SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY:  MOV R4,#35
H1:     MOV R3,#255
H2:     DJNZ R3,H2
        DJNZ R4,H1
        RET
        END
```

CIRCUIT DIAGRAM:





- Adjust the frequency in horizontal and channel A to Observe the square waveform on the oscilloscope for a consistent, symmetrical shape.

RESULT:

Thus the program has been successfully verified and executed using Proteus 8.10.

GENERATION OF TRIANGULAR WAVE USING PROTEUS

AIM:

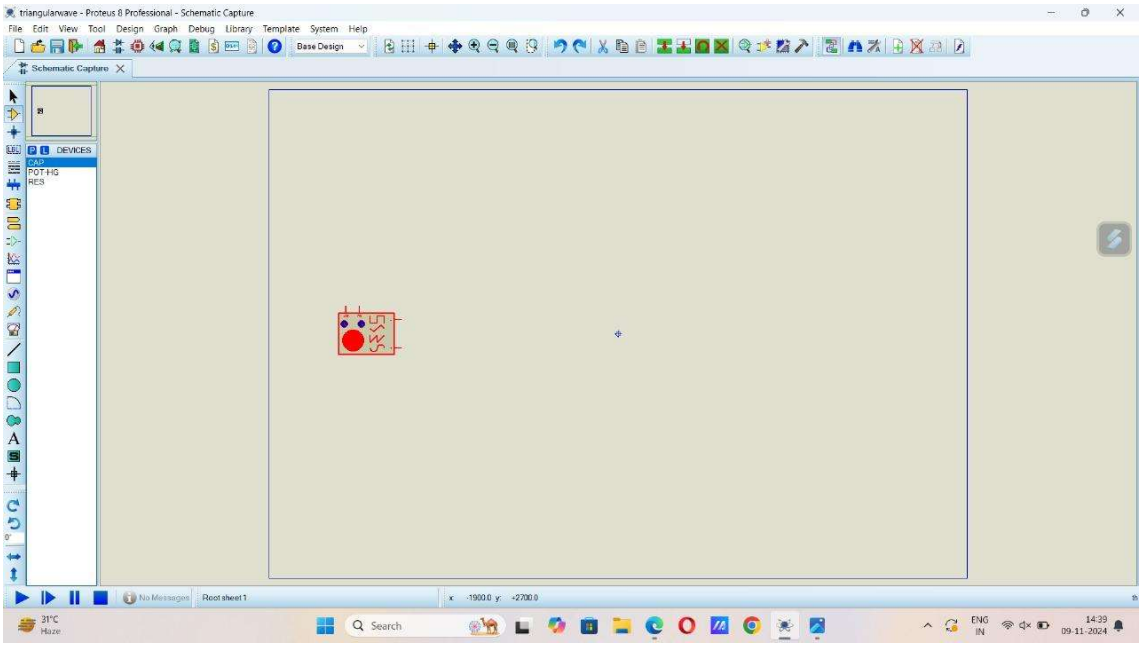
To write an assembly language code for generating triangular wave using Proteus

SOFTWARES:

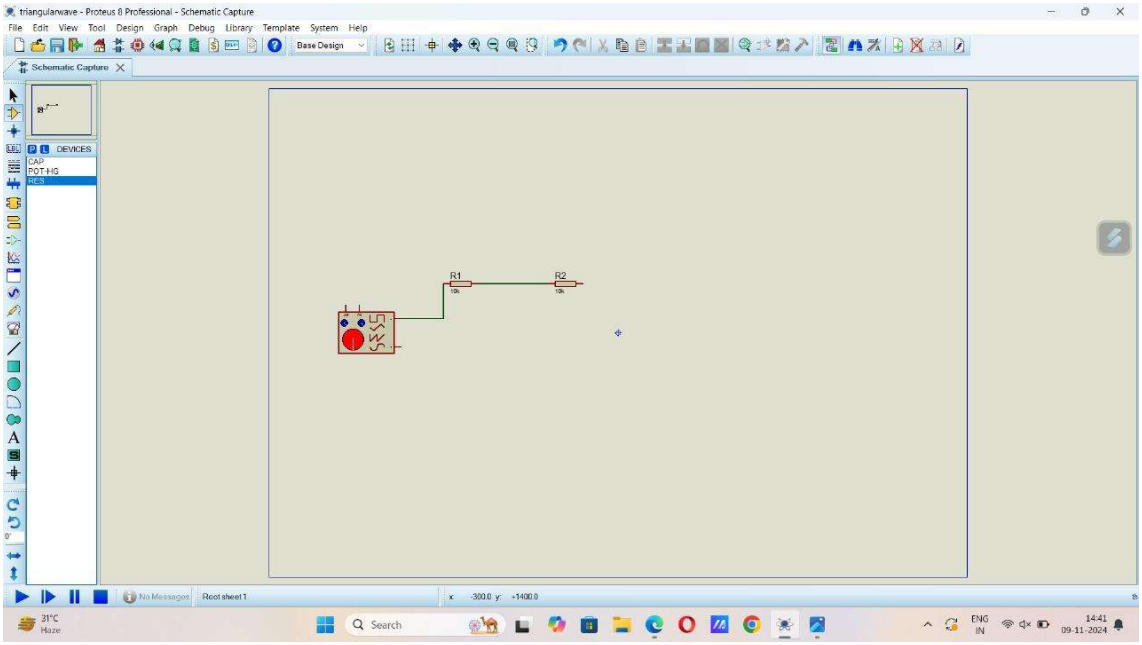
Proteus 8 Professional

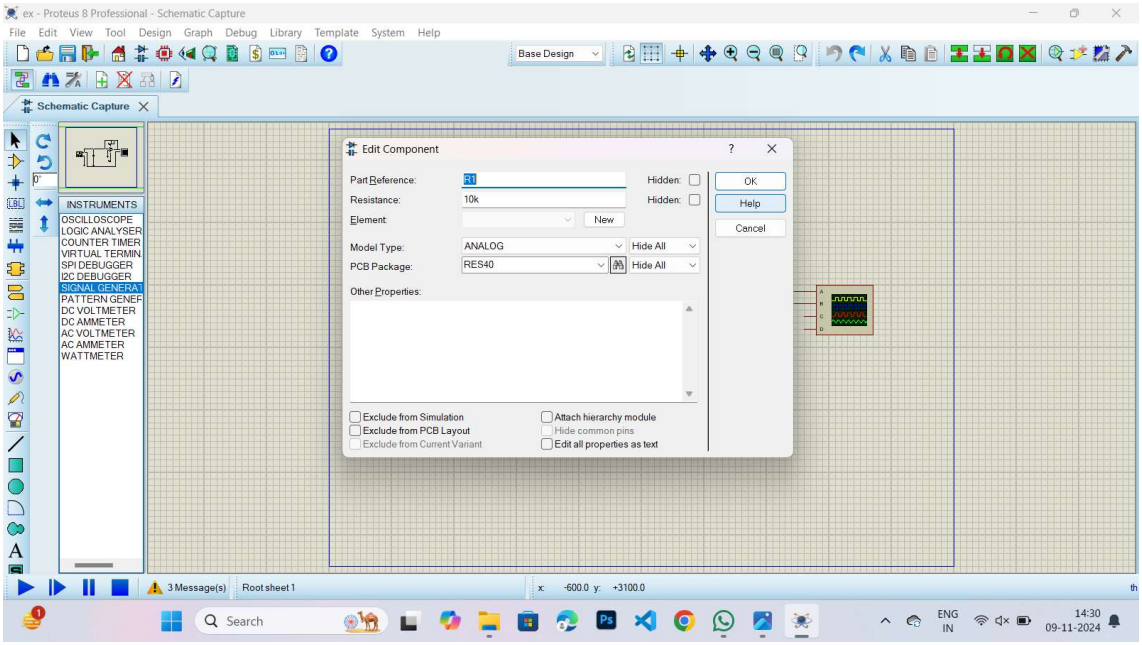
Procedure:

1.Select signal generator.

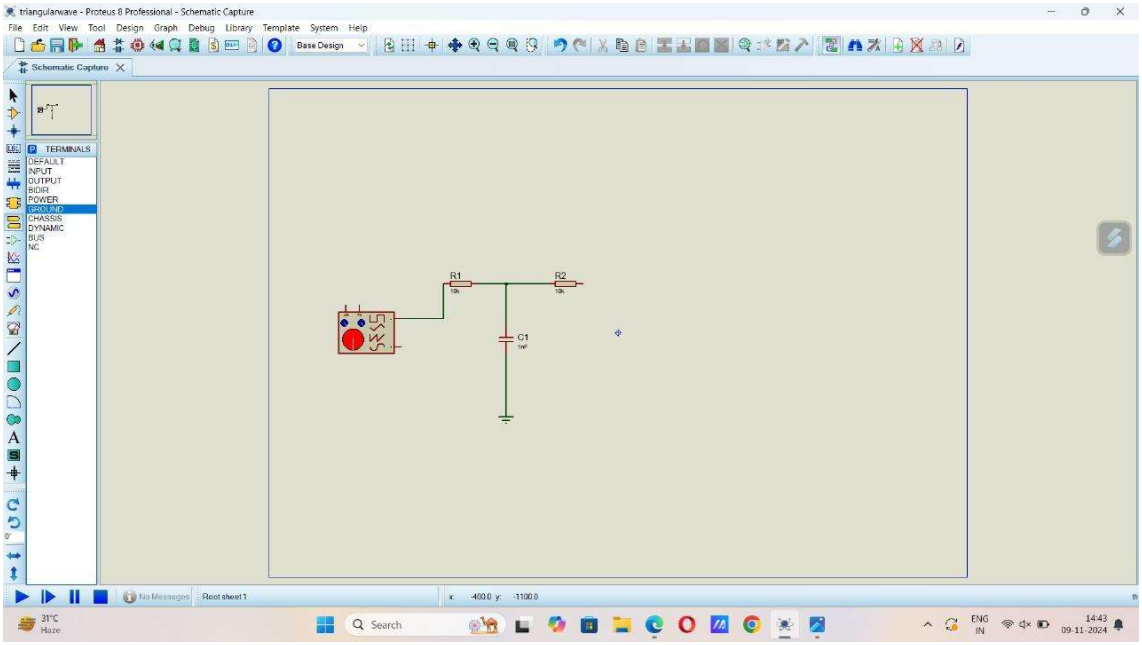


2.Connect 2 resistors R1 and R2 and change the resistance to 10k for both.

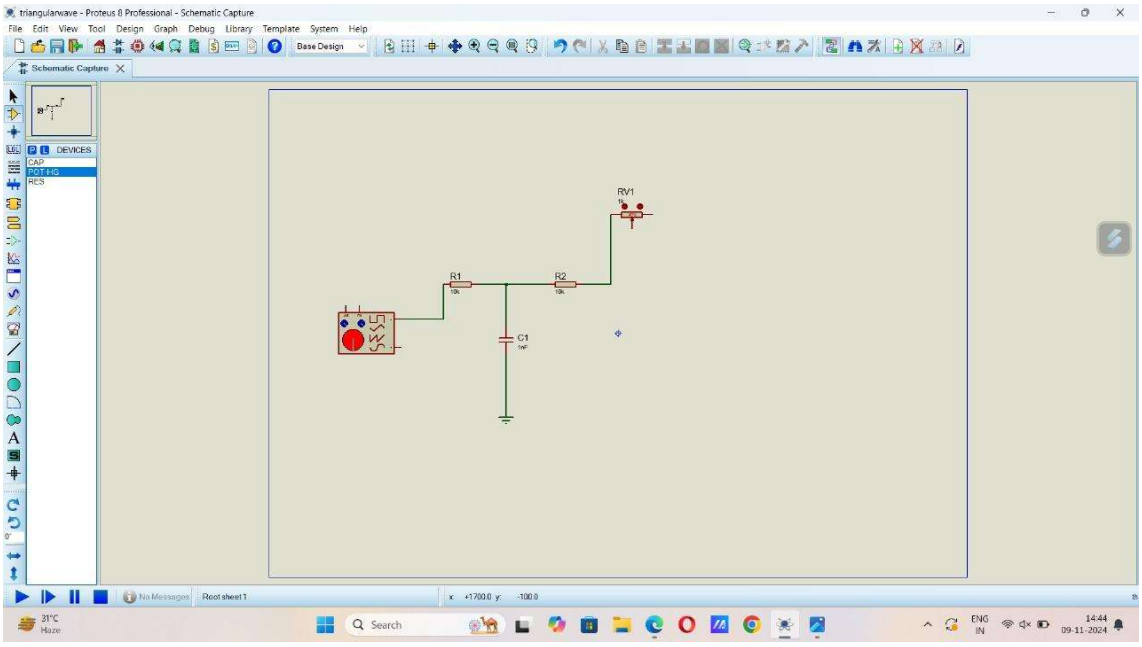




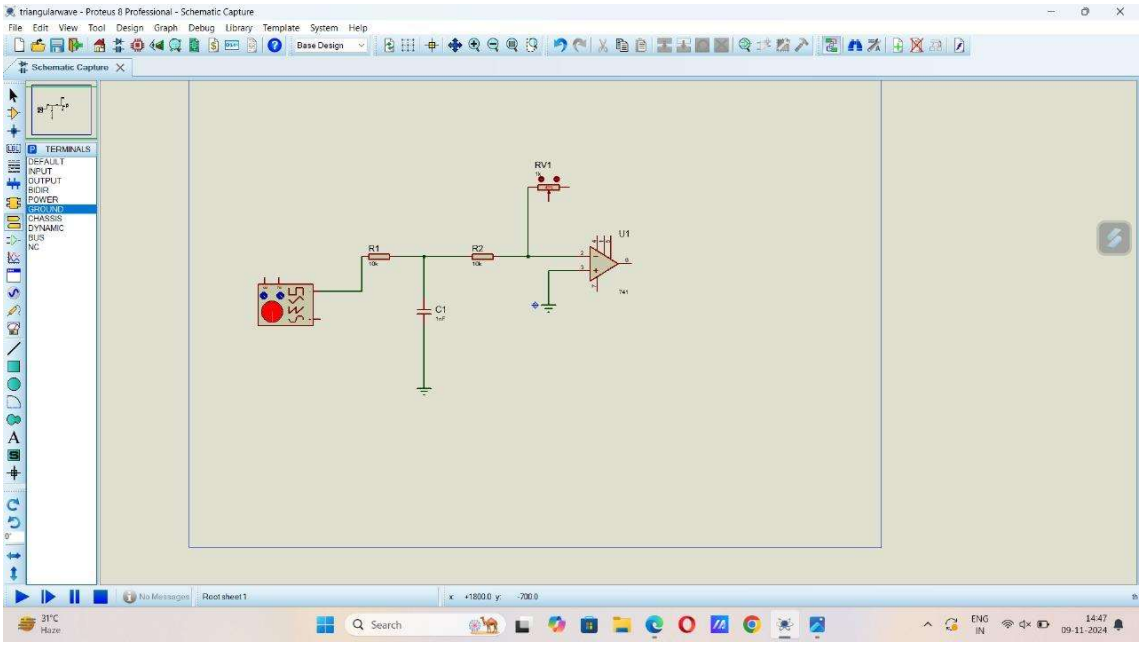
3.Add capacitor C1 with 10nf between the resistors and connect it to the ground.



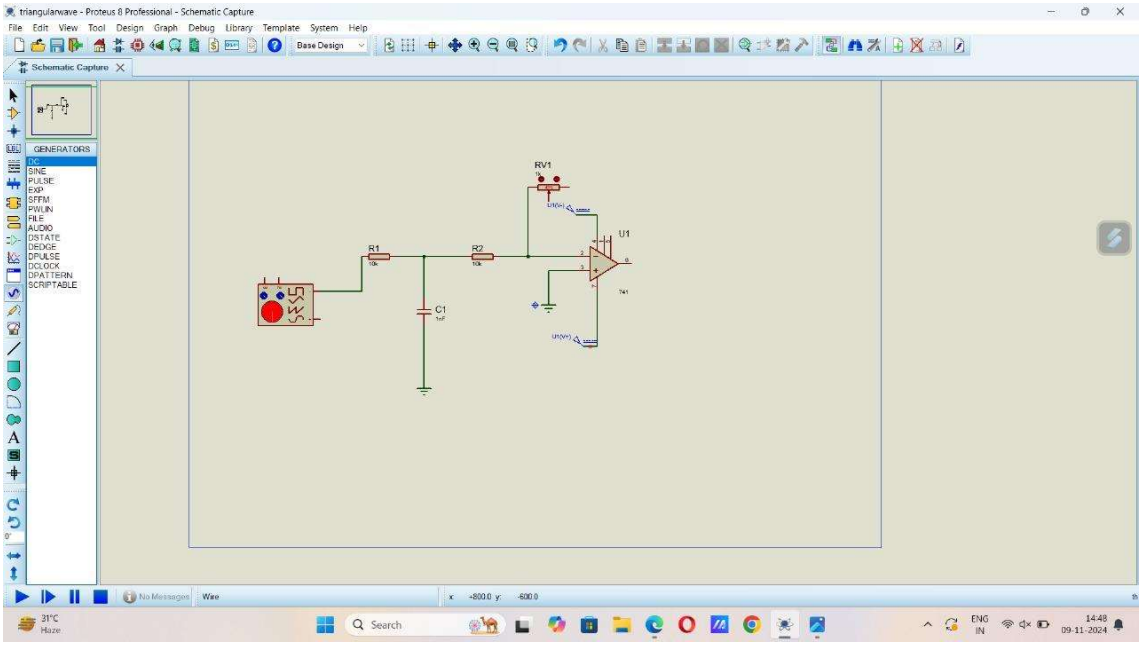
4.Add POT-HG to the from R2.



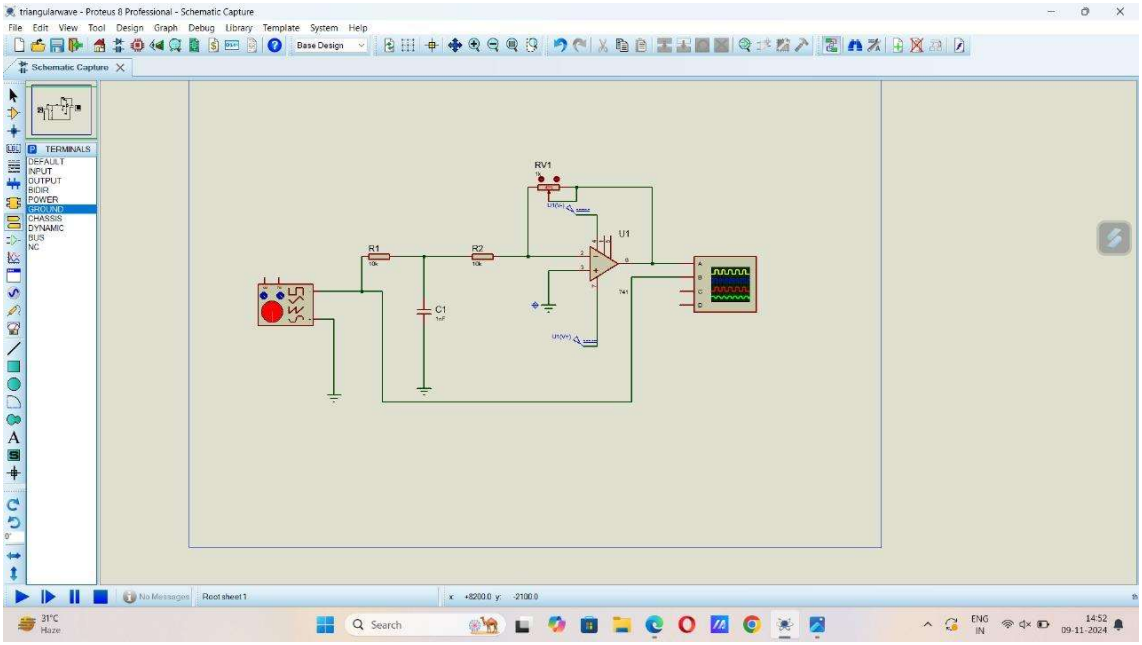
5.Add 741 from R2 and connect to 2 of 741, and connect 3 to the ground.



6.Add DC to 7 and 4 of 741.

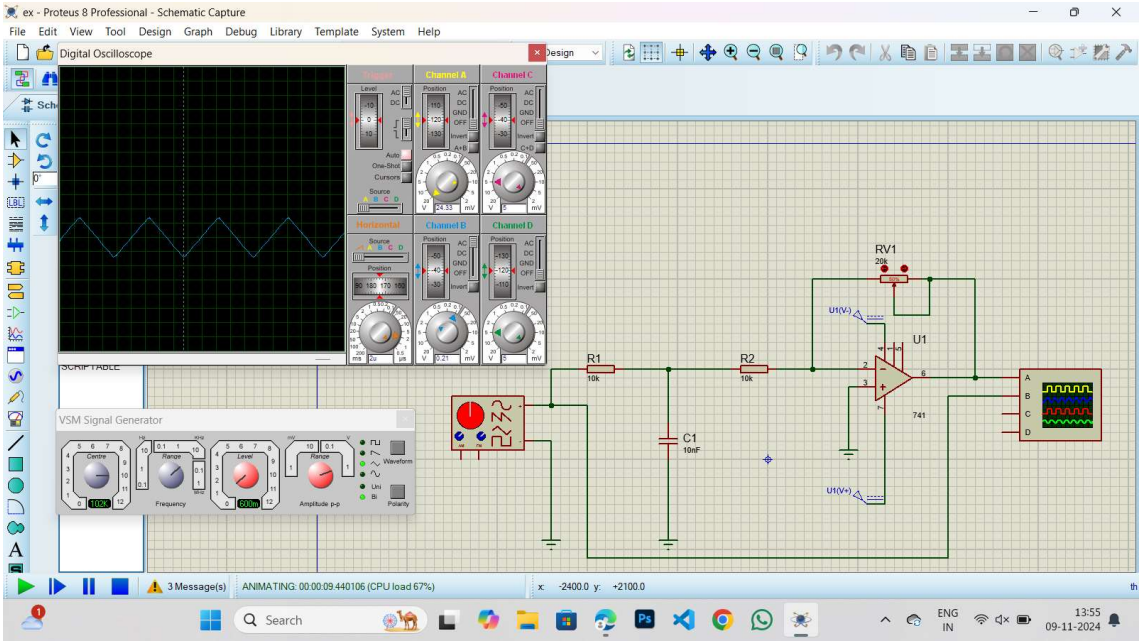


7.connect Oscilloscope as shown in the figure.



Output:

-Adjust the frequency accordingly to get the triangular wave.



RESULT:

Thus the program has been successfully verified and executed.

7 SEGMENT DISPLAY USING 8051 USING PROTEUS

AIM:

Write an assembly language program for 7 Segment Display Using 8051 using Keil and Proteus

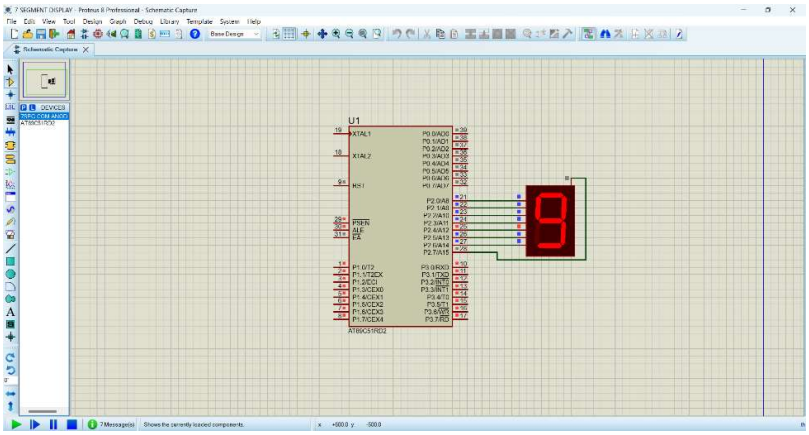
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 000H
UP:MOV P2,#0C0H
ACALL DELAY
MOV P2,#0F9H
ACALL DELAY
MOV P2,#0A4H
ACALL DELAY
MOV P2,#0B0H
ACALL DELAY
MOV P2,#99H
ACALL DELAY
MOV P2,#92H
ACALL DELAY
MOV P2,#82H
ACALL DELAY
MOV P2,#0F8H
ACALL DELAY
MOV P2, #80H
ACALL DELAY
MOV P2,#90H
ACALL DELAY

DELAY: MOV R5,#10
H1:MOV R4,#180
H2:MOV R3,#255
H3:DJNZ R3,H3
DJNZ R4,H2
DJNZ R5,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

ANTICLOCKWISE ROTATION OF STEPPER MOTOR USING 8051 USING PROTEUS

AIM:

Write An Assembly Language Program For Anticlockwise Rotation Of Stepper Motor Using 8051 Using Proteus

SOFTWARE REQUIRED:

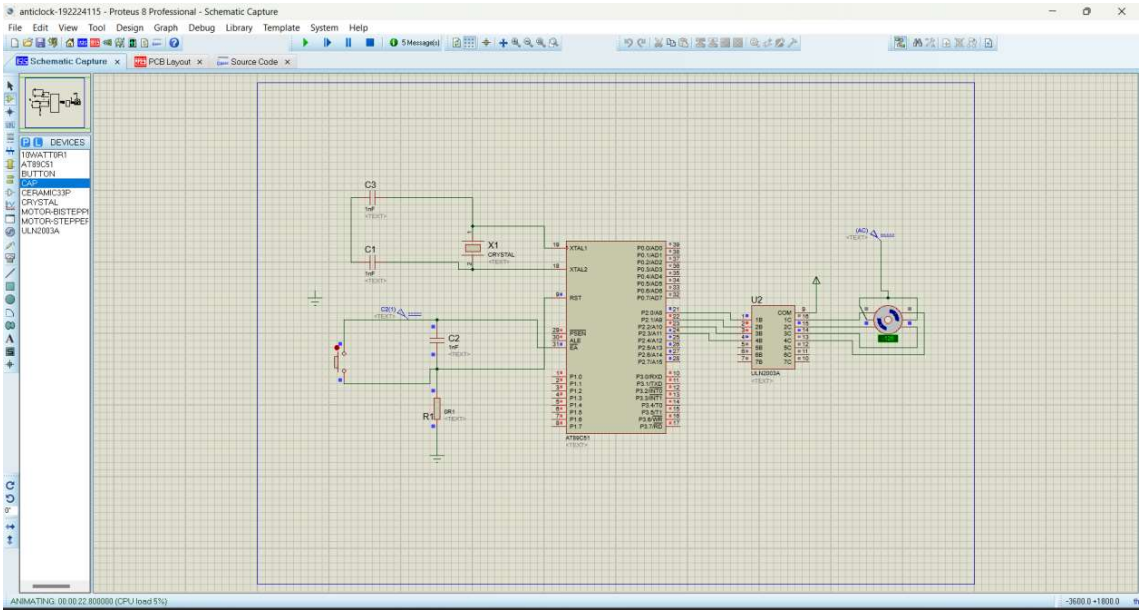
- Proteus 8 software.

PROGRAM:

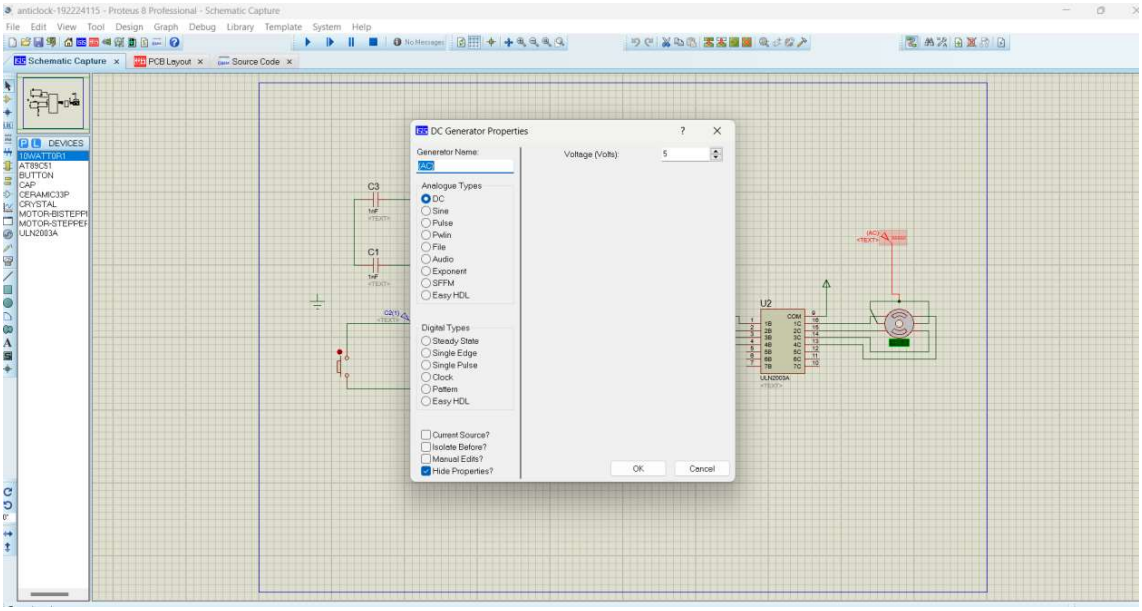
```
ORG 0000H
MOV P2, #0F0H   ; Set P2 as output port

; Rotate motor in anticlockwise direction
ANTICLOCKWISE:
  MOV P2, #09H ; Step 1 (1001)
  ACALL DELAY
  MOV P2, #03H ; Step 2 (0011)
  ACALL DELAY
  MOV P2, #06H ; Step 3 (0110)
  ACALL DELAY
  MOV P2, #0CH ; Step 4 (1100)
  ACALL DELAY
  SJMP ANTICLOCKWISE ; Loop back for continuous rotation

DELAY:
  MOV R0, #255
D1: MOV R1, #255
D2: DJNZ R1, D2
   DJNZ R0, D1
   RET
```

END

OUTPUT:



.Change value of DC from 0 to 5V



RESULT:

Thus the program has been successfully verified and executed.

CLOCKWISE ROTATION OF STEPPER MOTOR USING 8051 USING PROTEUS

AIM:

To write an assembly language program to rotate the Stepper Motor in clockwise direction in 8051 using Proteus

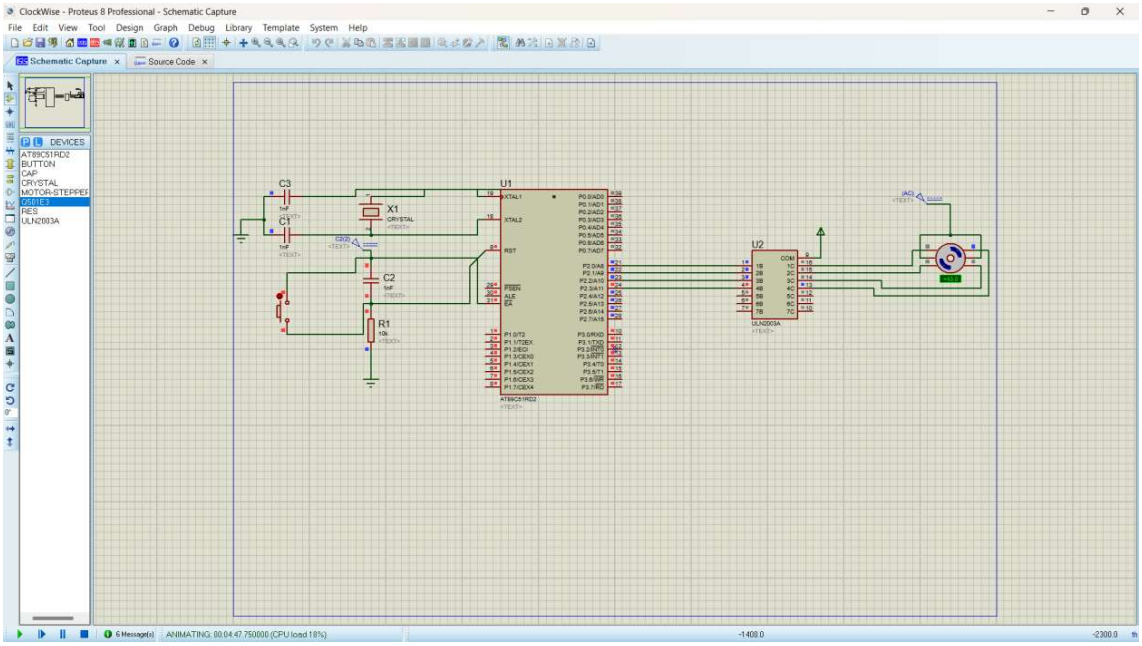
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 0000H         ; Starting address
UP:
   MOV P2, #08H  ; Step 1: Coil 4 = ON, others OFF (Clockwise direction)
   ACALL DELAY   ; Delay for a short period
   MOV P2, #04H  ; Step 2: Coil 3 = ON, others OFF
   ACALL DELAY   ; Delay for a short period
   MOV P2, #02H  ; Step 3: Coil 2 = ON, others OFF
   ACALL DELAY   ; Delay for a short period
   MOV P2, #01H  ; Step 4: Coil 1 = ON, others OFF
   ACALL DELAY   ; Delay for a short period
   SJMP UP       ; Repeat the sequence for continuous rotation

DELAY:
   MOV R4, #18   ; Outer delay loop counter
H1:
   MOV R3, #255  ; Inner delay loop counter
H2:
   DJNZ R3, H2   ; Decrement and loop if not zero
   DJNZ R4, H1   ; Decrement and loop if not zero
   RET           ; Return from delay subroutine

END
```

CIRCUIT DIAGRAM:



OUTPUT:

The stepper motor is rotating in clockwise direction in steps.

RESULT:

Thus, the program has been successfully verified and executed.

DIGITAL CLOCK ON LCD

AIM:

To write an assembly language program to display digital clock on LCD with using Proteus.

SOFTWARES REQUIRED:

- Proteus software

PROGRAM:

```
ORG 0000H           ; Start address of the program


MOV R7, #00H        ; Initialize hours (HH)

MOV R6, #00H        ; Initialize minutes (MM)

MOV R5, #00H        ; Initialize seconds (SS)


ACALL INIT_LCD      ; Initialize the LCD


MAIN_LOOP:

  ACALL UPDATE_LCD      ; Update the time on the LCD

  ACALL DELAY_1_SEC     ; Wait for 1 second

  ACALL INCREMENT_TIME  ; Increment time (HH:MM:SS)

  SJMP MAIN_LOOP        ; Repeat the process


; Subroutine to initialize the LCD

INIT_LCD:

  MOV R1, #38H

  ACALL CMD_WRITE

  ACALL SHORT_DELAY     ; Short delay between commands


  MOV R1, #0CH          ; Display ON, Cursor OFF

  ACALL CMD_WRITE

  ACALL SHORT_DELAY


  MOV R1, #06H          ; Auto-increment cursor
```

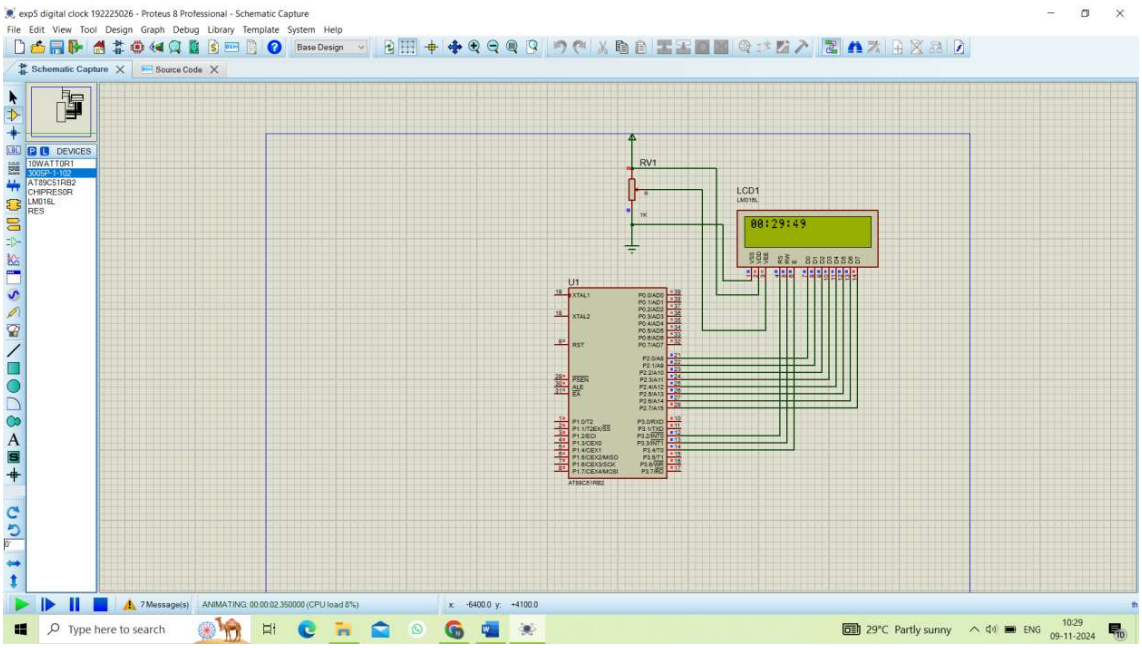
```
    ACALL CMD_WRITE
    ACALL SHORT_DELAY

    MOV R1, #01H          ; Clear display
    ACALL CMD_WRITE
    ACALL DELAY_CLEAR     ; Longer delay after clearing display
    RET

; Subroutine to increment time
INCREMENT_TIME:
    INC R5                ; Increment seconds (SS)
    CJNE R5, #60, DONE_SEC ; If seconds < 60, continue
    MOV R5, #00H          ; Reset seconds to 00
    INC R6                ; Increment minutes (MM)
    CJNE R6, #60, DONE_SEC ; If minutes < 60, continue
    MOV R6, #00H          ; Reset minutes to 00
    INC R7                ; Increment hours (HH)
    CJNE R7, #24, DONE_SEC ; If hours < 24, continue
    MOV R7, #00H          ; Reset hours to 00
DONE_SEC:
    RET

; Subroutine to update the LCD with the current time
UPDATE_LCD:
    MOV R1, #80H          ; Move cursor to the first line of the LCD
    ACALL CMD_WRITE
    ACALL SHORT_DELAY

    MOV A, R7             ; Load hours (HH) into accumulator
    ACALL DISPLAY_TWO_DIGIT ; Display hours (HH)
```

```
    ACALL DISPLAY_COLON     ; Display ':'

    MOV A, R6               ; Load minutes (MM) into accumulator
    ACALL DISPLAY_TWO_DIGIT ; Display minutes (MM)

    ACALL DISPLAY_COLON     ; Display ':'

    MOV A, R5               ; Load seconds (SS) into accumulator
    ACALL DISPLAY_TWO_DIGIT ; Display seconds (SS)

    RET

; Subroutine to display two-digit numbers on the LCD
DISPLAY_TWO_DIGIT:
    MOV B, #10              ; Divide the value in A by 10
    DIV AB                  ; Quotient in A (tens), remainder in B (ones)

    ADD A, #30H             ; Convert tens digit to ASCII
    ACALL DISPLAY_CHAR      ; Display the tens digit

    MOV A, B                ; Move the remainder (ones digit) to A
    ADD A, #30H             ; Convert ones digit to ASCII
    ACALL DISPLAY_CHAR      ; Display the ones digit

    RET

; Subroutine to display colon ':' on the LCD
DISPLAY_COLON:
    MOV A, #3AH             ; ASCII value of ':'
    ACALL DISPLAY_CHAR      ; Display ':'
    RET
```

```
; Subroutine to display a character on the LCD
DISPLAY_CHAR:
    MOV P2, A             ; Send ASCII character to data pins (P2 connected to D0-D7 of LCD)
    SETB P3.2             ; Set RS to 1 (data register)
    CLR P3.3              ; Set RW to 0 (write mode)
    SETB P3.4             ; Set E to 1 (Enable high)
    NOP                   ; Small delay
    CLR P3.4              ; Set E to 0 (Enable low)
    ACALL SHORT_DELAY     ; Short delay after writing character
    RET

; Subroutine to write command to the LCD
CMD_WRITE:
    MOV P2, R1            ; Send command from R1 to data pins (P2 connected to D0-D7 of LCD)
    CLR P3.2              ; Set RS to 0 (command register)
    CLR P3.3              ; Set RW to 0 (write mode)
    SETB P3.4             ; Set E to 1 (Enable high)
    NOP                   ; Small delay
    CLR P3.4              ; Set E to 0 (Enable low)
    ACALL SHORT_DELAY     ; Short delay after writing command
    RET

; Short delay subroutine (for small delays between LCD commands)
SHORT_DELAY:
    MOV R2, #50           ; Adjust for a small delay
DELAY1:
    DJNZ R2, DELAY1
    RET

; Longer delay subroutine for clearing the LCD
```

```
DELAY_CLEAR:
    MOV R2, #200          ; Longer delay after clearing the display
DELAY2:
    DJNZ R2, DELAY2
    RET

; Subroutine for 1-second delay
DELAY_1_SEC:
    MOV R3, #50           ; Adjust this value to create a 1-second delay
DELAY_LOOP:
    DJNZ R3, DELAY_LOOP   ; Simple delay loop
    RET

END
```

CIRCUIT DIAGRAM:



OUTPUT:

- When this program is run, the LCD will display the current time in the format HH:MM.
- Every second, the display will update to increment the seconds value.
- After reaching 59 seconds, the seconds will reset to 00, and the minutes will increment.

- Similarly, when the minutes reach 59 and increment again, they will reset to 00, and the hours will increment.
- The hours will increment from 00 to 23 in a 24-hour format. When the hours reach 23 and the next second occurs, the hours, minutes, and seconds will all reset to 00:00:00.

RESULT:

Thus, the assembly language program to display digital clock on LCD with using Proteus was executed.

DIGITAL THERMOMETER

AIM : To write an assembly language program for Digital Thermometer Using Proteus.

SOFTWARE:

- Proteus 8 Professional
- Keil

PROGRAM:

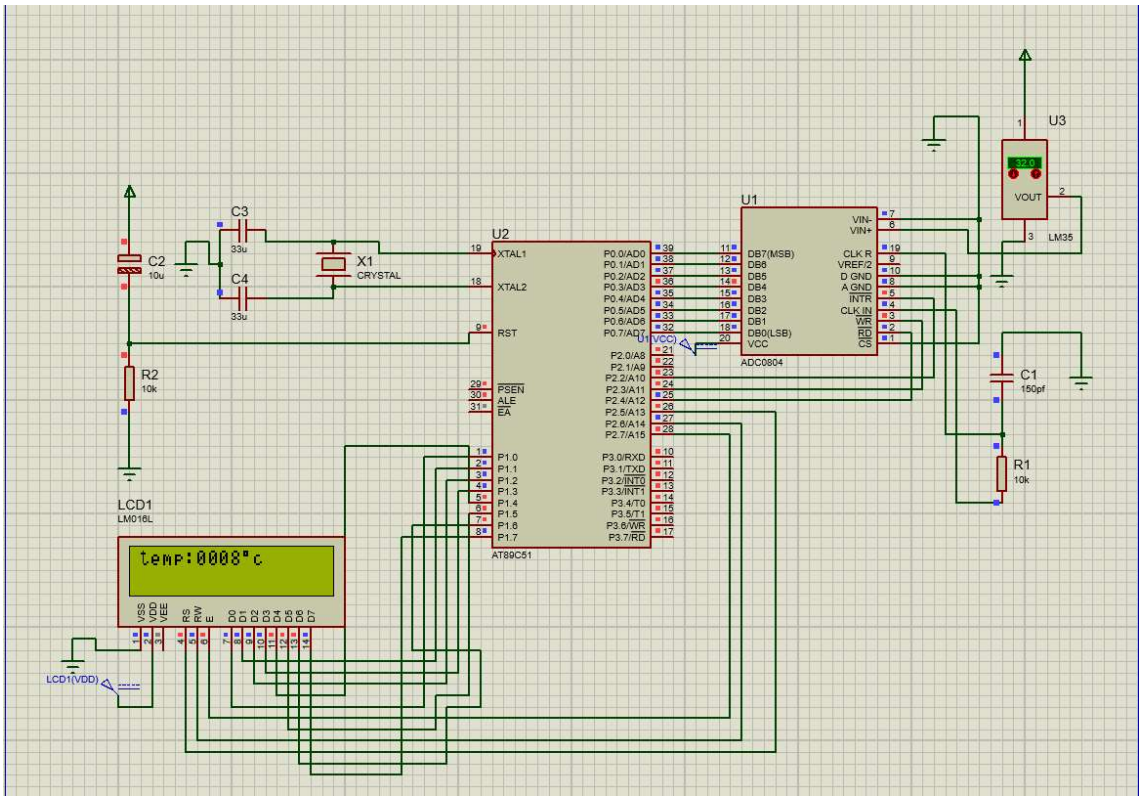
```
#include<reg51.h>

#include<string.h>

#define lcd P1

#define input P0


sbit rd=P2^4;

sbit wr=P2^3;

sbit intr=P2^2;

sbit rs=P2^5;

sbit rw=P2^6;

sbit e=P2^7;

void delay (unsigned int);

void cmd (unsigned char);

void ldata (unsigned char);

unsigned int adc();

void string (char *c);


void delay (unsigned int d)

{

        unsigned int i;

        for(;d>0;d--)
```

```
        {
                for(i=250;i>0;i--);
        }
}
void cmd (unsigned char c)
{
        lcd=c;
        rs=0;
        rw=0;
        e=1;
        delay(5);
        e=0;
}
void ldata (unsigned char c)
{
        lcd=c;
        rs=1;
        rw=0;
        e=1;
        delay(5);
        e=0;
}

unsigned int adc()                                      // Reading values from
ADC and display on the LED's
{
        unsigned int adc=0x00;
        rd=1;
```

```
        wr=0;
        delay(10);
        wr=1;
        while(intr==1);
        rd=0;
        intr=1;
        delay(10);
        adc=input;
        return(adc);
}

void string (char *c)
{
        while(*c)
        {
                ldata(*c++);
        }
}

void main()
{
        int i=0,j =0;
        unsigned char x3;
  unsigned char tmpAdcData;
        cmd(0x38);
        cmd(0x0c);
        cmd(0x01);
```

```
        cmd(0x80);
        while(1)
        {
                delay(10);
                cmd(0x80);
                input=0xff;
                string("temp:");
                cmd(0x85);
                tmpAdcData=adc();
                ldata((tmpAdcData/1000)+(0x30)); // SEPERATE BITS AND
PRINT
   tmpAdcData = tmpAdcData%1000;
   ldata((tmpAdcData/100)+(0x30));
   tmpAdcData = tmpAdcData%100;
   ldata((tmpAdcData/10)+(0x30));
   tmpAdcData = tmpAdcData%10;
   ldata((tmpAdcData)+(0x30));
                x3=0xDF;
                ldata(x3);
                string("c");
        }
}
```

CIRCUIT DIAGRAM



RESULT:

Thus, the assembly language program to digital thermometer on LCD with using Proteus was executed.