**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

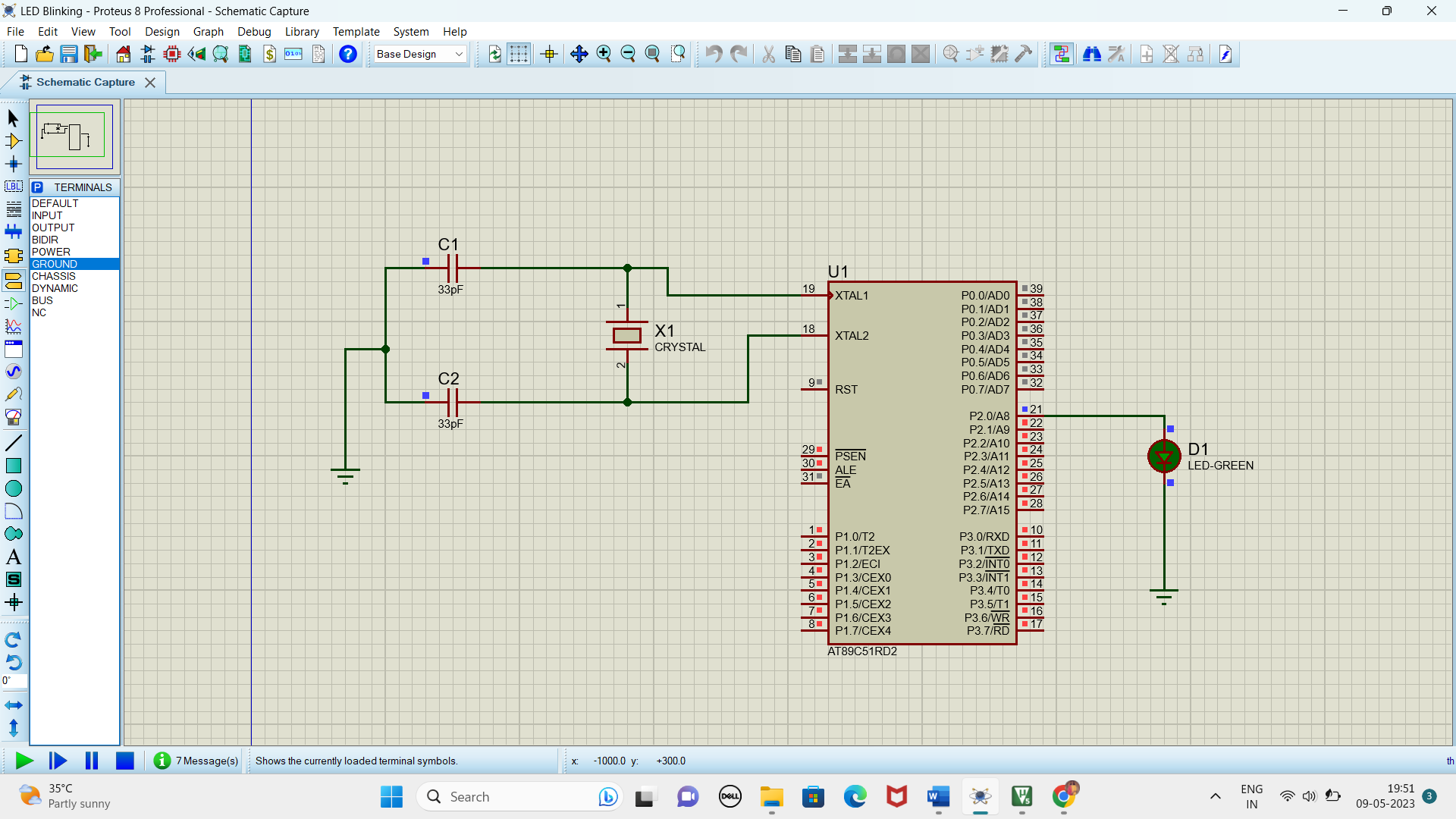
H2:DJNZ R3,H2

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

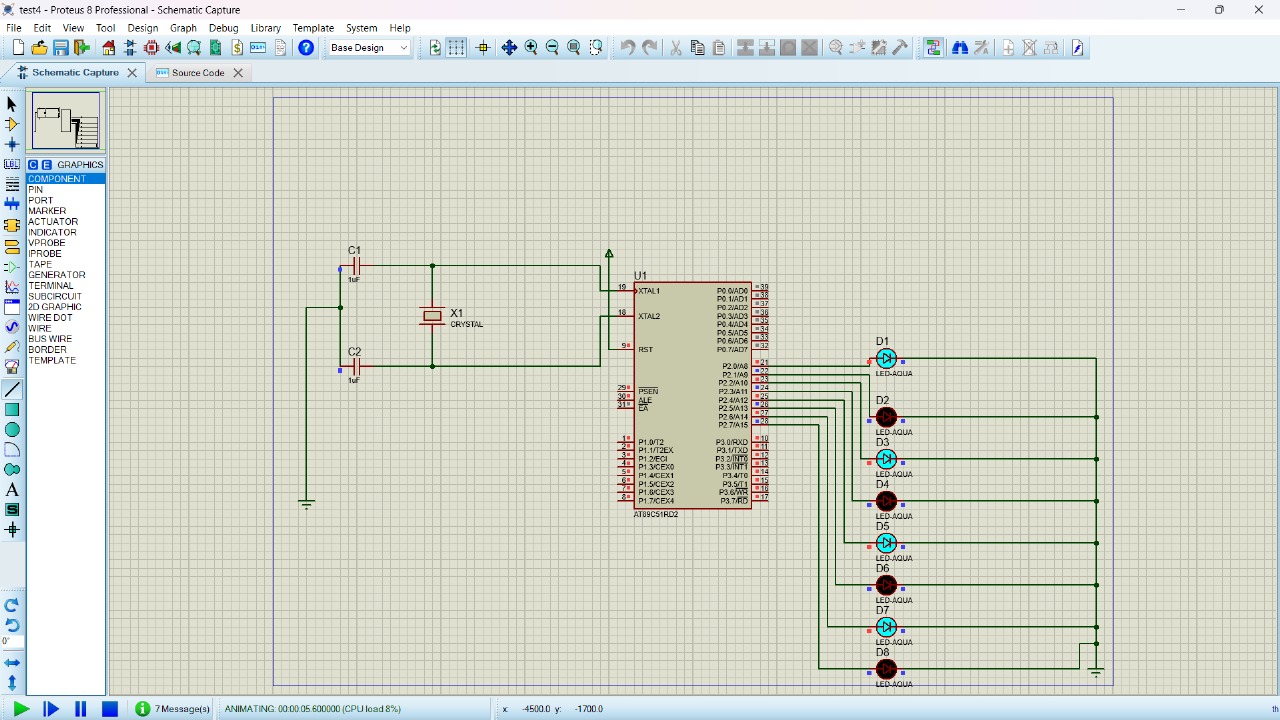
H2:DJNZ R3,H2

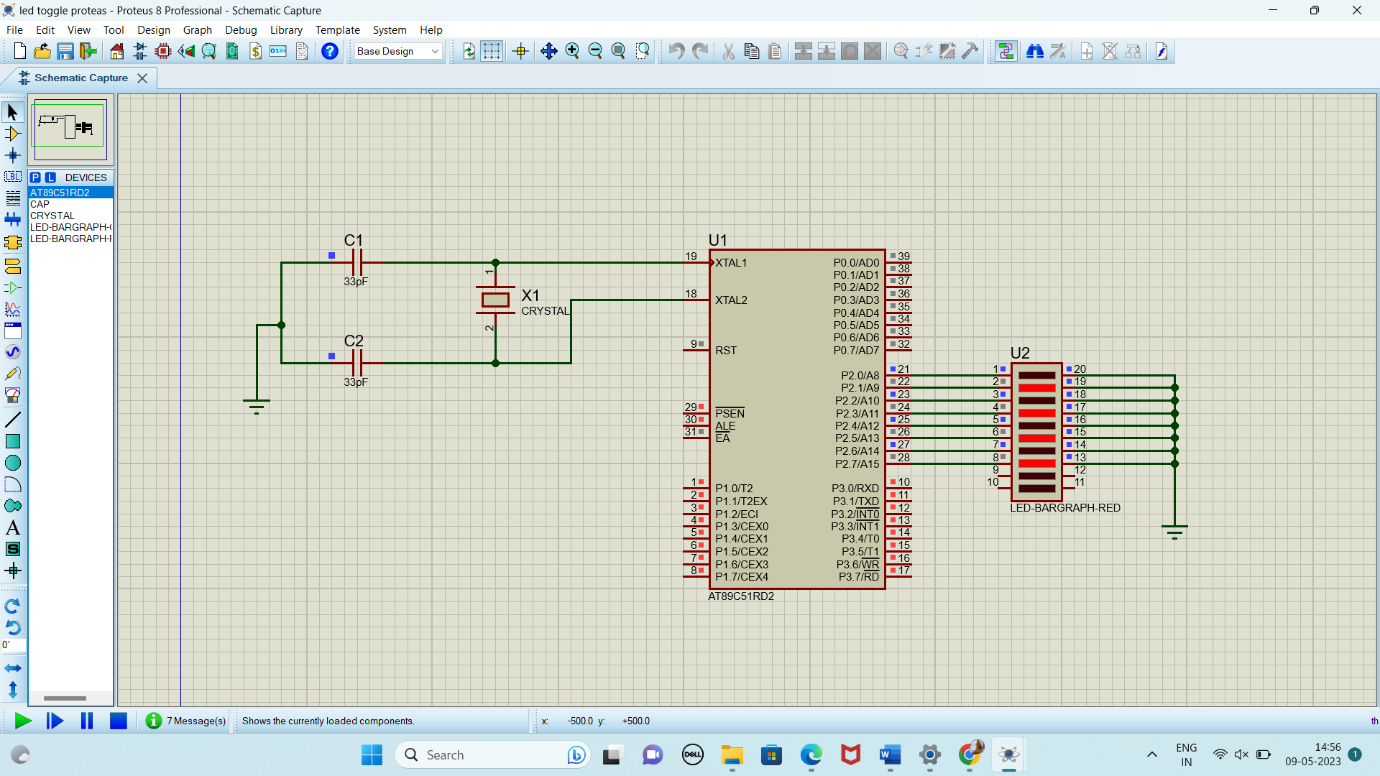
DJNZ R4,H1

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

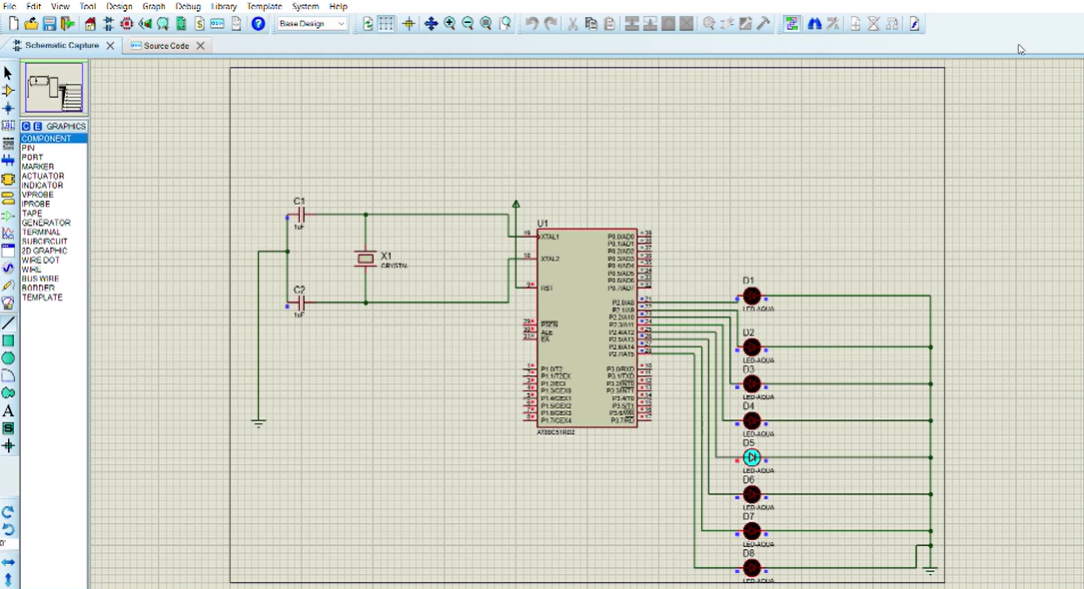
DELAY: MOV R4,#255

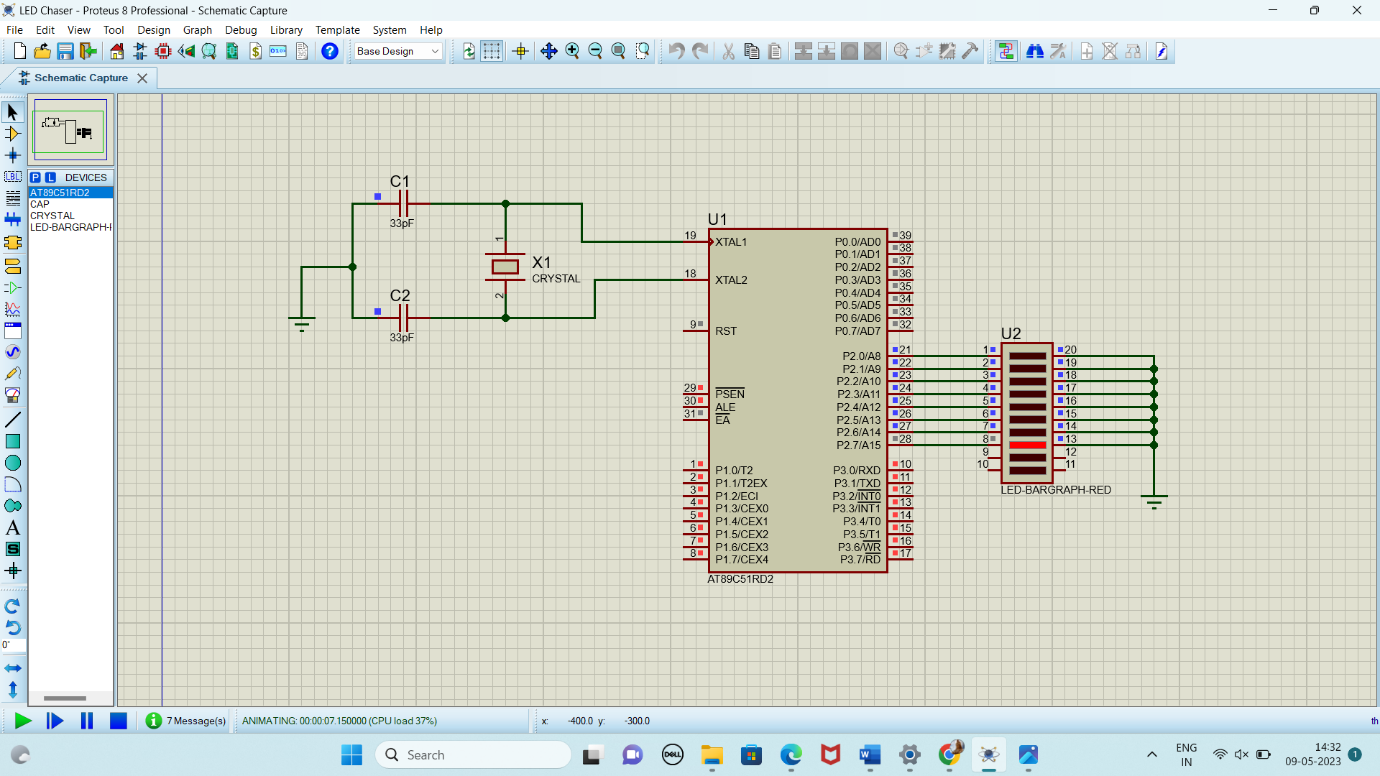
H1: DJNZ R4,H1

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 ; Start program at address 00H

MAIN: MOV P2, #00H ; Initialize Port 2 (LED off)

ACALL FADE\_IN ; Call Fade In subroutine

ACALL FADE\_OUT ; Call Fade Out subroutine

SJMP MAIN ; Repeat forever

; Subroutine to Fade In the LED

FADE\_IN:

MOV R0, #00H ; Start with 0% duty cycle (LED off)

FADE\_IN\_LOOP:

ACALL PWM ; Call the PWM subroutine with the current duty cycle

INC R0 ; Increase the duty cycle

CJNE R0, #FFH, FADE\_IN\_LOOP ; Repeat until max brightness (100% duty cycle)

RET

; Subroutine to Fade Out the LED

FADE\_OUT:

MOV R0, #FFH ; Start with 100% duty cycle (LED on)

FADE\_OUT\_LOOP:

ACALL PWM ; Call the PWM subroutine with the current duty cycle

DEC R0 ; Decrease the duty cycle

CJNE R0, #00H, FADE\_OUT\_LOOP ; Repeat until min brightness (0% duty cycle)

RET

; PWM subroutine

PWM:

MOV A, R0 ; Load duty cycle value

MOV B, #FFH ; Set maximum period

MOV P1, #00H ; LED ON (active-low, so writing 0 turns on the LED)

PWM\_ON\_LOOP:

DJNZ A, PWM\_ON\_LOOP ; Delay based on duty cycle (LED ON time)

MOV P1, #01H ; LED OFF

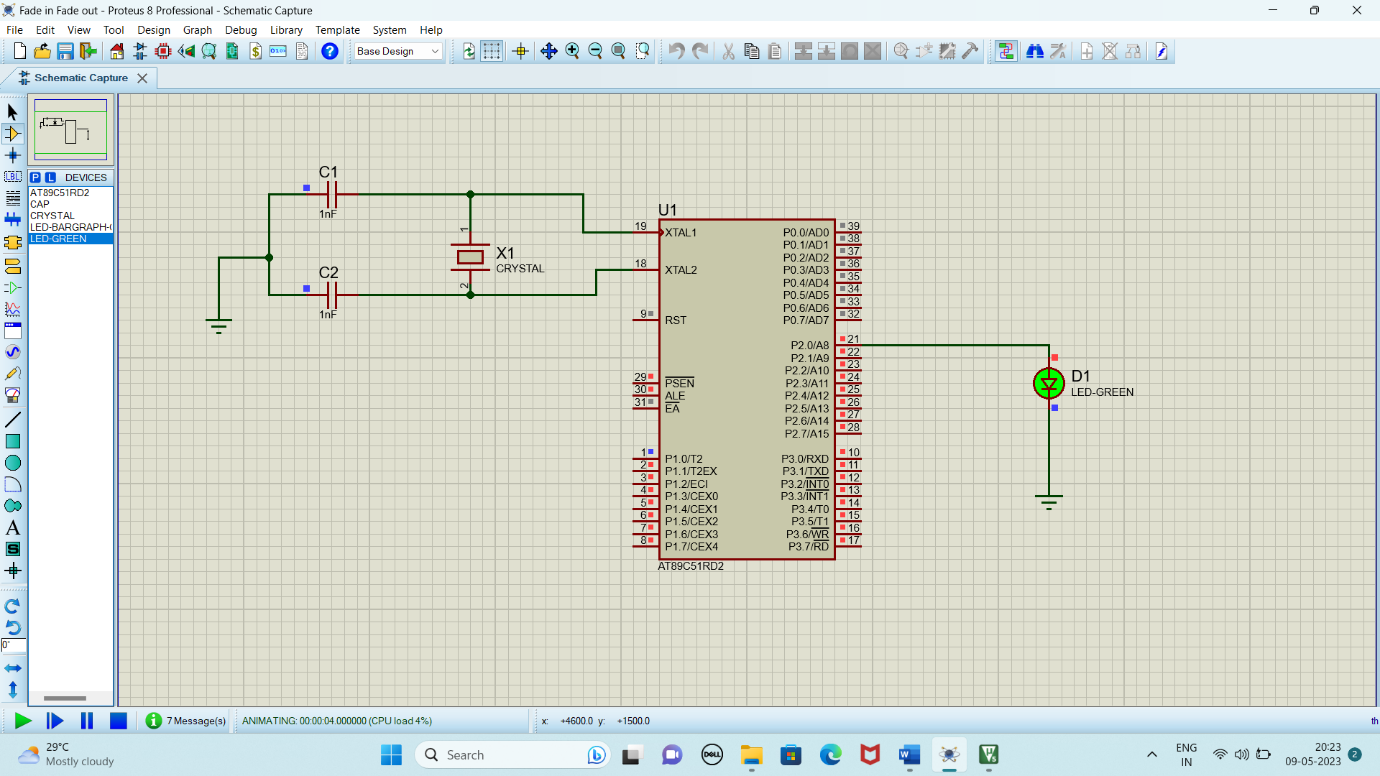
PWM\_OFF\_LOOP:

DJNZ B, PWM\_OFF\_LOOP ; Delay for the rest of the period (LED OFF time)

RET ; Return from subroutine

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

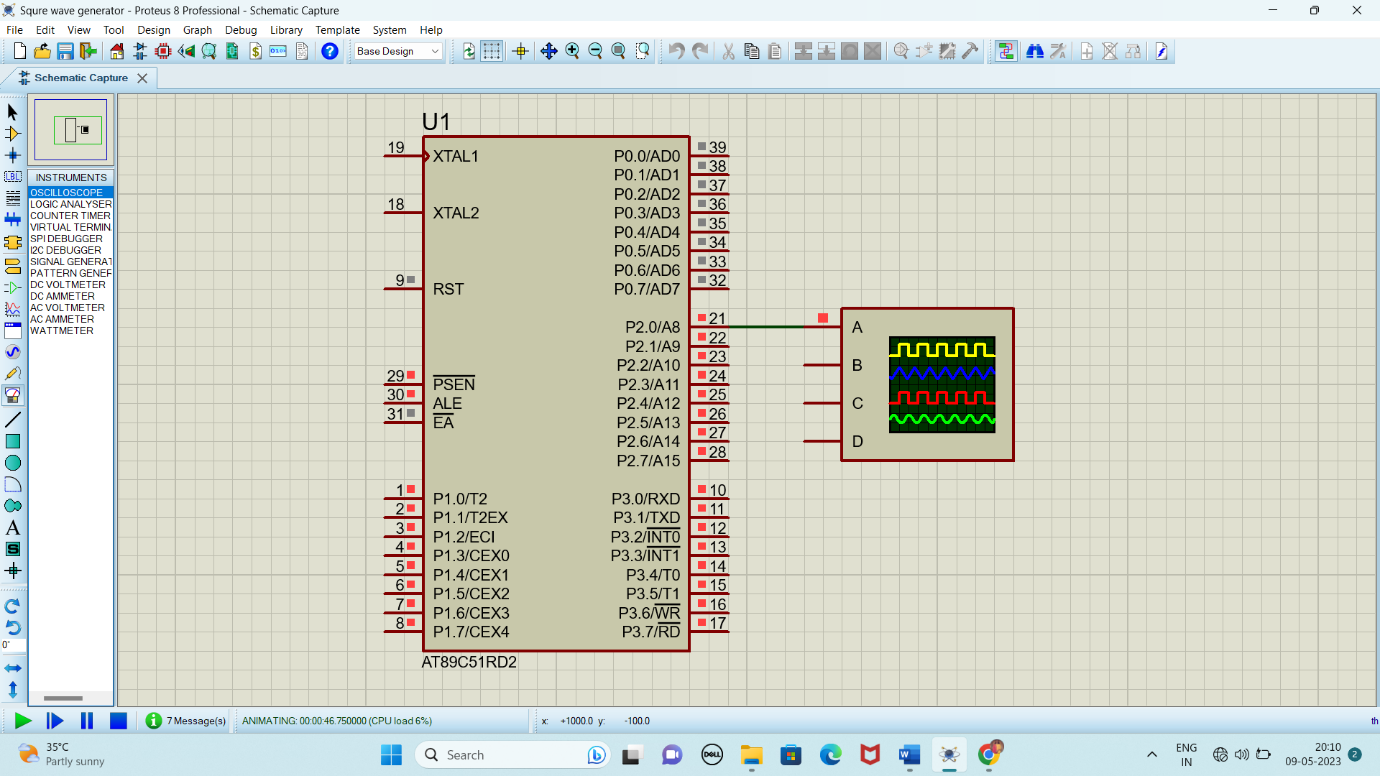
H2: DJNZ R3,H2

DJNZ R4,H1

RET

END

**CIRCUIT DIAGRAM:**



**GENERATION OF TRIANGULAR WAVE USING PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Proteus 8 software.

**PROGRAM**

ORG 00H ; Start of the program

MOV P2.0, #00H ; Clear Port 1 (connected to DAC0808)

MOV A, #00H ; Initialize accumulator to 0 (starting value)

MOV R0, #00H ; Initialize R0 for increment step

UPWARD:

INC A ; Increment the value in the accumulator (rising edge of triangle)

MOV P1, A ; Send the incremented value to Port 1 (connected to DAC)

ACALL DELAY ; Call delay for waveform frequency control

CJNE A, #0FFH, UPWARD ; Continue incrementing until the maximum value (0xFF)

DOWNWARD:

DEC A ; Decrement the value in the accumulator (falling edge of triangle)

MOV P1, A ; Send the decremented value to Port 1

ACALL DELAY ; Delay for waveform frequency control

CJNE A, #00H, DOWNWARD ; Continue decrementing until it reaches 0

SJMP UPWARD ; Repeat the process indefinitely to generate a continuous waveform

; Delay Subroutine

DELAY:

MOV R1, #255 ; Outer loop for delay

DELAY\_LOOP1:

MOV R2, #255 ; Inner loop for delay

DELAY\_LOOP2:

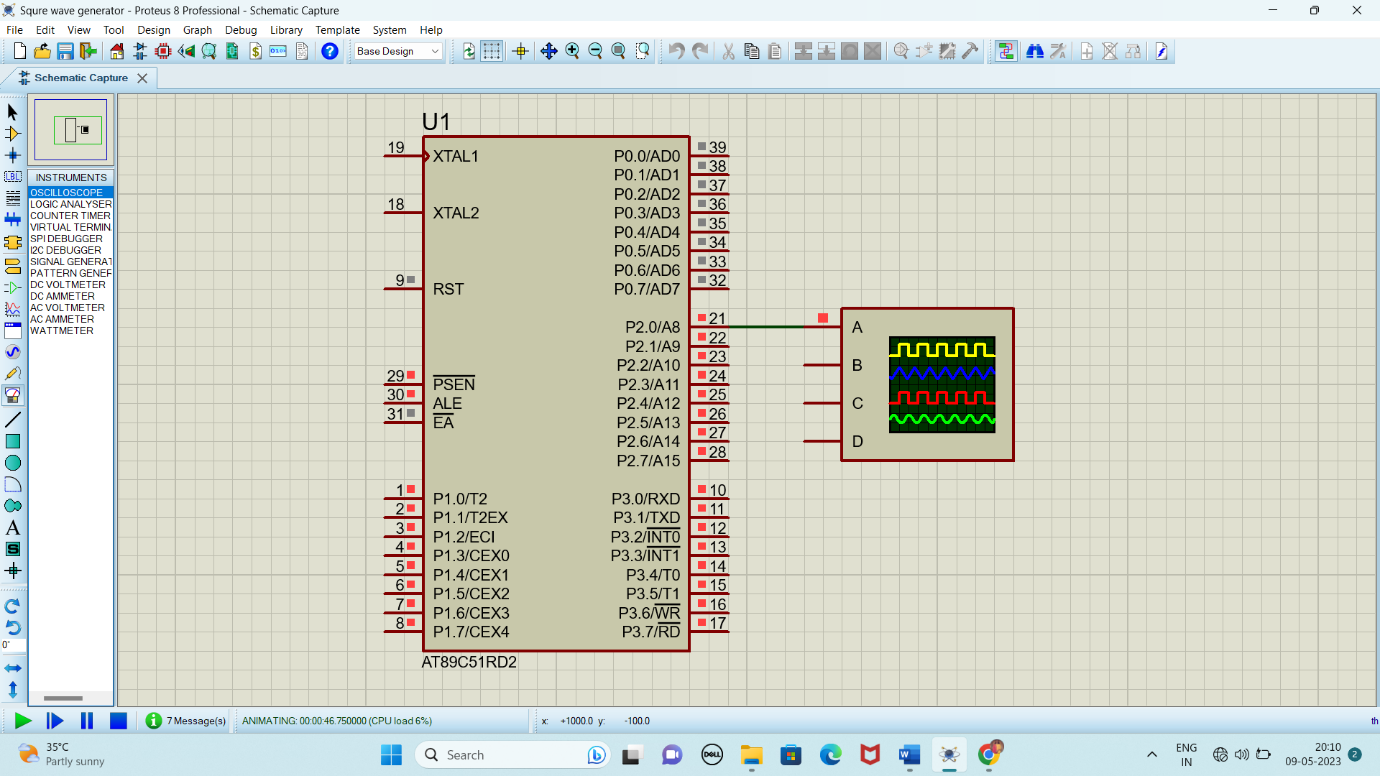
DJNZ R2, DELAY\_LOOP2 ; Decrement inner loop

DJNZ R1, DELAY\_LOOP1 ; Decrement outer loop

RET ; Return from delay

END

**CIRCUIT DIAGRAM:**



**ANTICLOCKWISE ROTATION OF STEPPER MOTOR USING 8051 USING PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Proteus 8 software.

**PROGRAM:**

ORG 00H ; Start program at address 0x00

MAIN: MOV P2, #0F0H ; Initialize Port 2 as output (upper nibble)

ACALL COUNTERCLOCKWISE ; Rotate stepper motor in counterclockwise direction

ACALL DELAY ; Call delay

SJMP MAIN ; Repeat forever

; Subroutine to rotate stepper motor counterclockwise

COUNTERCLOCKWISE:

MOV A, #08H ; Load step 4 (1000)

MOV P2, A

ACALL DELAY

MOV A, #04H ; Load step 3 (0100)

MOV P2, A

ACALL DELAY

MOV A, #02H ; Load step 2 (0010)

MOV P2, A

ACALL DELAY

MOV A, #01H ; Load step 1 (0001)

MOV P2, A

ACALL DELAY

RET ; Return from subroutine

; Subroutine to generate a delay

DELAY:

MOV R1, #0FFH ; Load delay counter (outer loop)

DELAY\_LOOP1:

MOV R2, #0FFH ; Load delay counter (inner loop)

DELAY\_LOOP2:

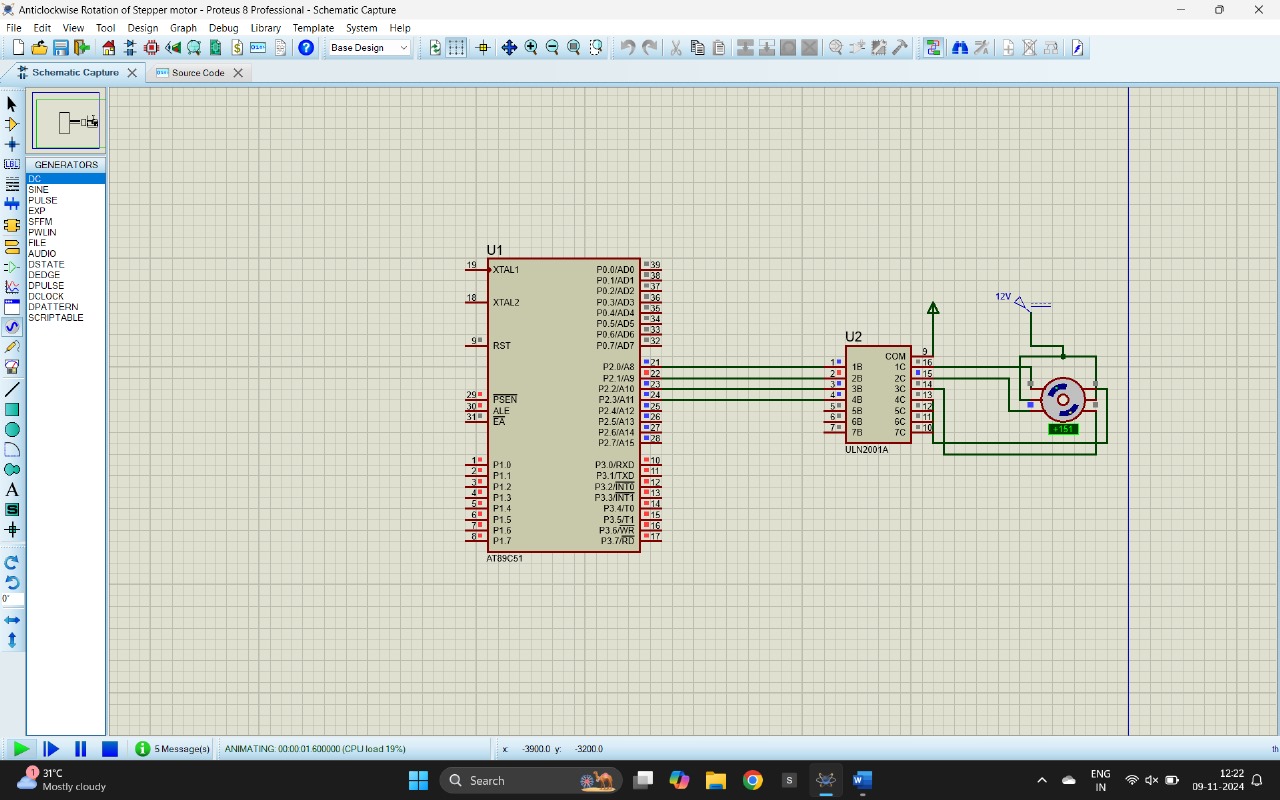
DJNZ R2, DELAY\_LOOP2 ; Decrement inner loop counter

DJNZ R1, DELAY\_LOOP1 ; Decrement outer loop counter

RET ; Return from 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.

**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

UP: MOV P2,#09H

ACALL DELAY

MOV P2,#0CH

ACALL DELAY

MOV P2,#06H

ACALL DELAY

MOV P2,#03H

ACALL DELAY

SJMP UP

DELAY:MOV R4,#18

H1:MOV R3,#255

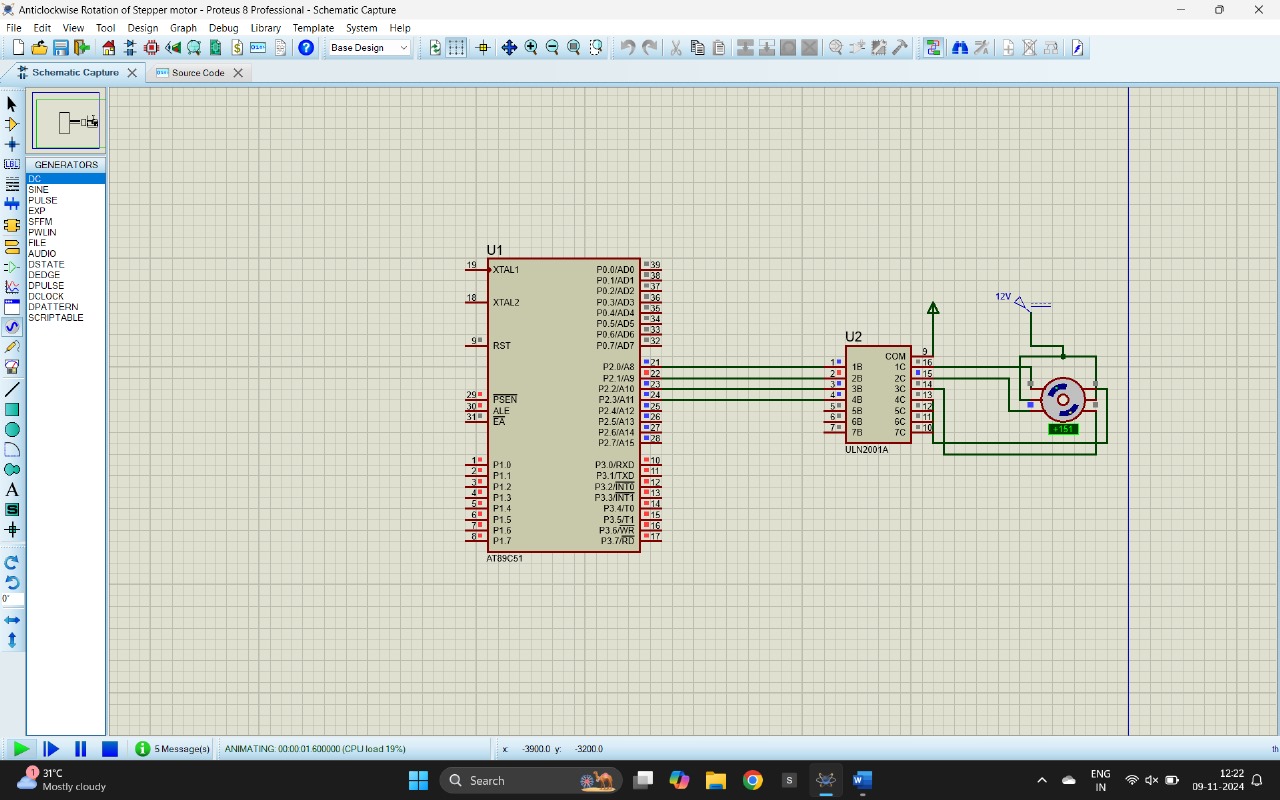
H2:DJNZ R3,H2

DJNZ R4,H1

RET

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 A, #38H

ACALL CMD\_WRITE ; 8-bit mode, 2 lines, 5x7 matrix

ACALL DELAY\_SHORT

MOV A, #0CH

ACALL CMD\_WRITE ; Display ON, Cursor OFF

ACALL DELAY\_SHORT

MOV A, #06H

ACALL CMD\_WRITE ; Auto-increment cursor

ACALL DELAY\_SHORT

MOV A, #01H

ACALL CMD\_WRITE ; Clear display

ACALL DELAY\_SHORT

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 A, #80H

ACALL CMD\_WRITE ; Move cursor to the first line of the LCD

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 DELAY\_SHORT ; Short delay after sending character

RET

; Subroutine to write command to the LCD

CMD\_WRITE:

MOV P2, A ; Send command 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 DELAY\_SHORT ; Short delay after sending command

RET

; Short delay for LCD commands and data

DELAY\_SHORT:

MOV R0, #250 ; Adjust this value for a short delay

DELAY\_SHORT\_LOOP:

DJNZ R0, DELAY\_SHORT\_LOOP

RET

; Subroutine for 1-second delay

DELAY\_1\_SEC:

MOV R3, #50 ; Outer loop for delay

DELAY\_LOOP:

MOV R4, #255 ; Inner loop for delay

DELAY\_LOOP\_INNER:

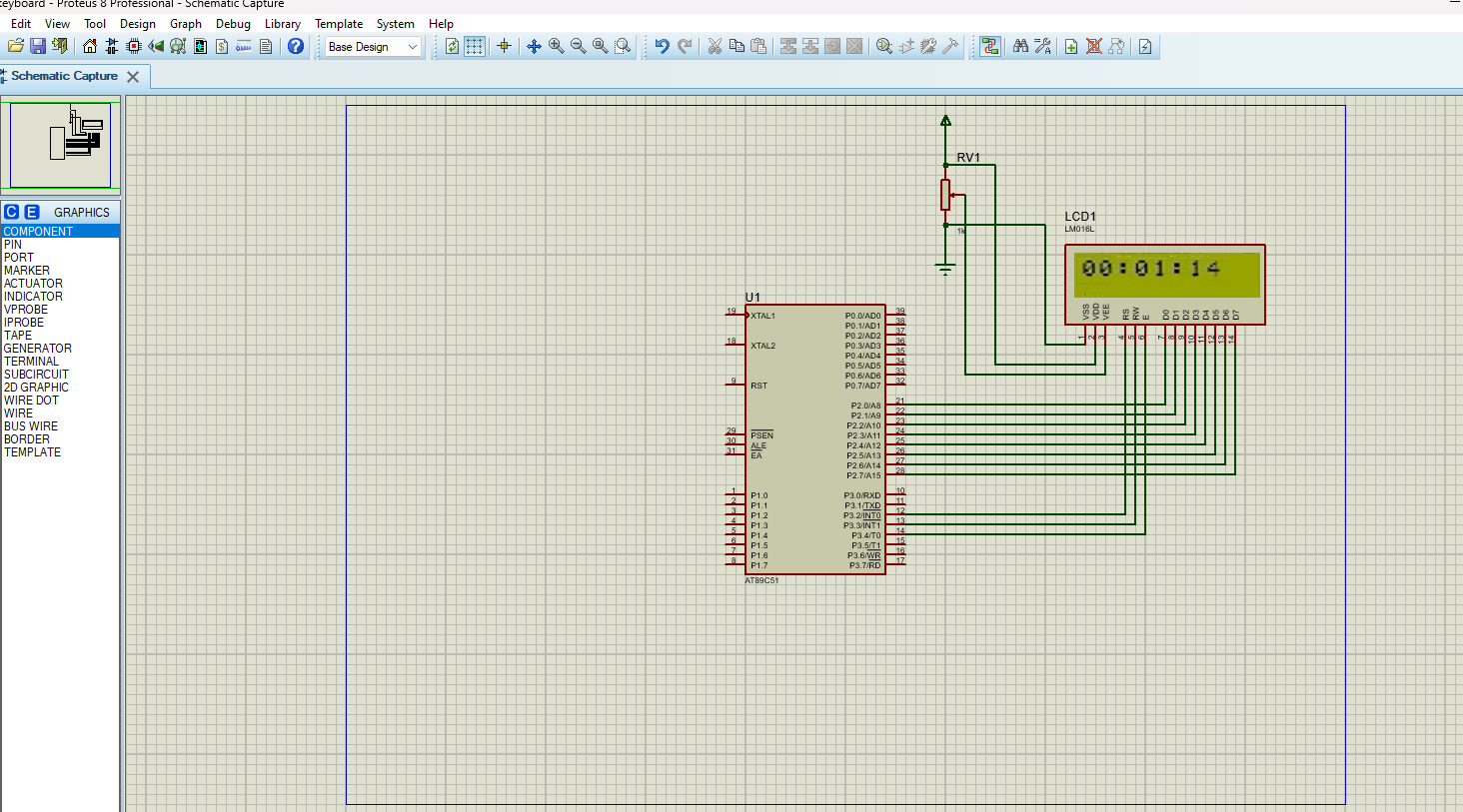
DJNZ R4, DELAY\_LOOP\_INNER

DJNZ R3, 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.

**INTERFACING OF RELAY AND LED WITH 8051 USING PROTEUS**

**AIM:**

To write an assembly language program to interface relay and LED with 8051 using Proteus.

**SOFTWARE REQUIRED:**

* Proteus 8 software.

**PROGRAM:**

ORG 0000H ; Start of program

; Initialize Port 1 as output port for relay control

MOV P1, #00H ; Clear Port 1 (all pins low initially)

MAIN\_LOOP:

SETB P1.0 ; Set P1.0 HIGH (Relay ON, LED ON)

ACALL DELAY ; Call delay to keep the LED ON for some time

CLR P1.0 ; Clear P1.0 (Relay OFF, LED OFF)

ACALL DELAY ; Call delay to keep the LED OFF for some time

SJMP MAIN\_LOOP; Repeat the process

; Delay subroutine for blinking speed

DELAY:

MOV R1, #255 ; Outer loop

DELAY1:

MOV R2, #255 ; Inner loop

DELAY2:

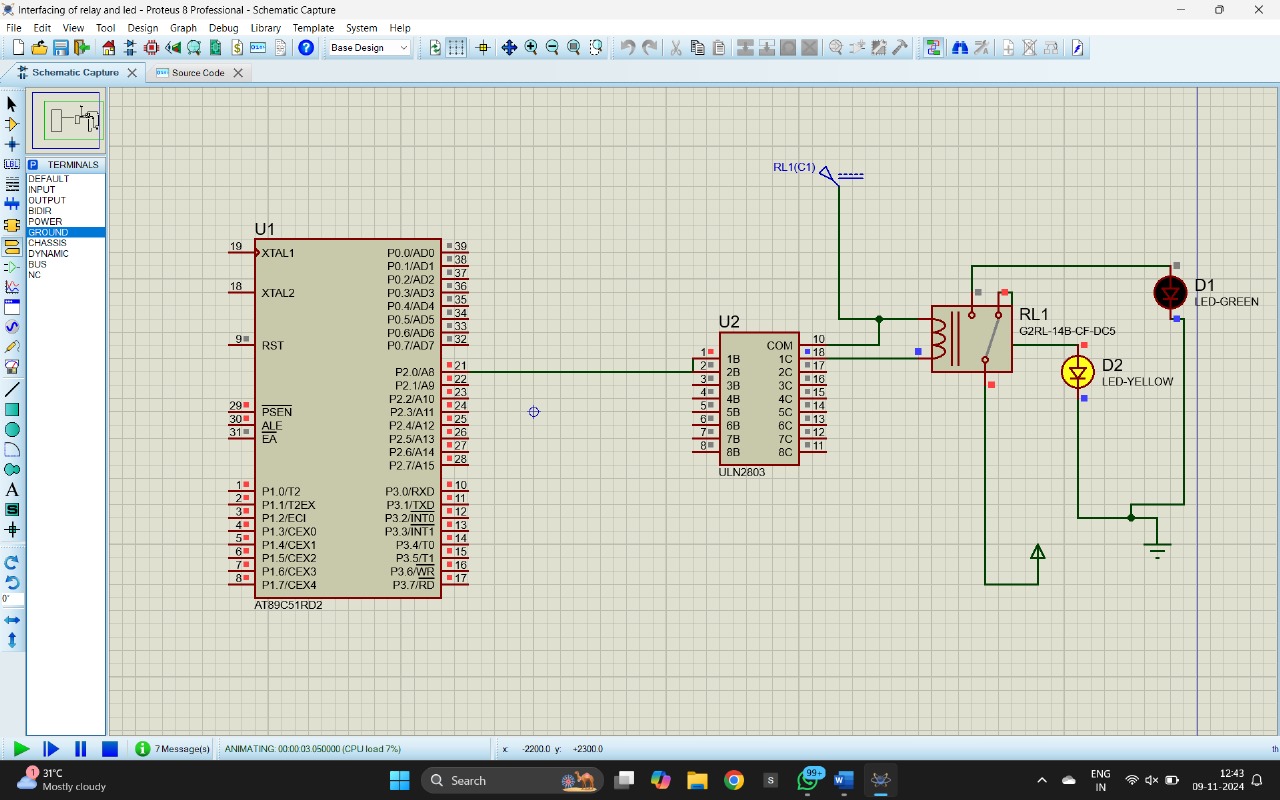
DJNZ R2, DELAY2 ; Decrement inner loop

DJNZ R1, DELAY1 ; Decrement outer loop

RET ; Return to main loop

END ; End of program

**CIRCUIT DIAGRAM:**



**OUTPUT:**

* The LED connected through the relay will blink with a controlled ON and OFF duration.
* The relay acts as a switch controlled by the 8051 microcontroller, turning the LED ON when P1.0 is HIGH and OFF when P1.0 is LOW.
* The blinking rate of the LED can be adjusted by changing the delay subroutine.

**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

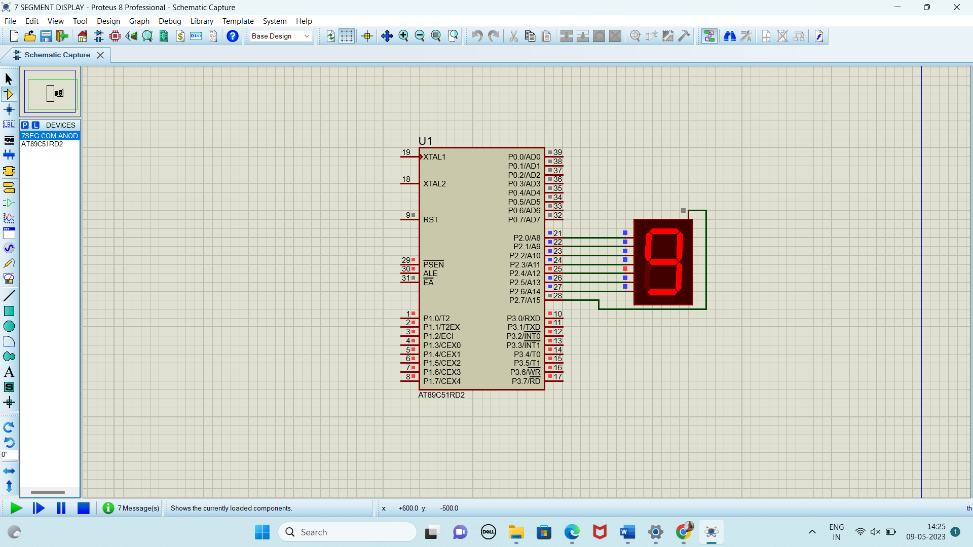
DJNZ R4,H2

DJNZ R5,H1

RET

END

**CIRCUIT DIAGRAM:**



**RESULT:**

Thus the program has been successfully verified and executed.

**TRAFFIC SIGNALS USING 8051 USING PROTEUS**

**AIM:**

To write an assembly language program Traffic Signals Using 8051 using Keil and Proteus.

**SOFTWARE REQUIRED:**

* Proteus 8 software.

**PROGRAM:**

ORG 00H

MOV P2, #00H

MOV P3, #00H

MAIN:

SETB P2.2

SETB P3.2

SETB P2.3

SETB P3.3

ACALL DELAY1

SETB P2.4

SETB P3.4

CLR P2.3

CLR P3.3

ACALL DELAY2

MOV P2, #00H

MOV P3, #00H

SETB P2.5

SETB P3.5

SETB P2.0

SETB P3.0

ACALL DELAY1

SETB P2.1

SETB P3.1

CLR P2.0

CLR P3.0

ACALL DELAY2

MOV P2, #00H

MOV P3, #00H

SJMP MAIN

DELAY1:

MOV R0, #255D

D1\_LOOP1:

MOV R1, #255D

D1\_LOOP2:

MOV R2, #142D

D1\_LOOP3:

DJNZ R2, D1\_LOOP3

DJNZ R1, D1\_LOOP2

DJNZ R0, D1\_LOOP1

RET

DELAY2:

MOV R0, #255D

D2\_LOOP1:

MOV R1, #142D

D2\_LOOP2:

MOV R2, #51D

D2\_LOOP3:

DJNZ R2, D2\_LOOP3

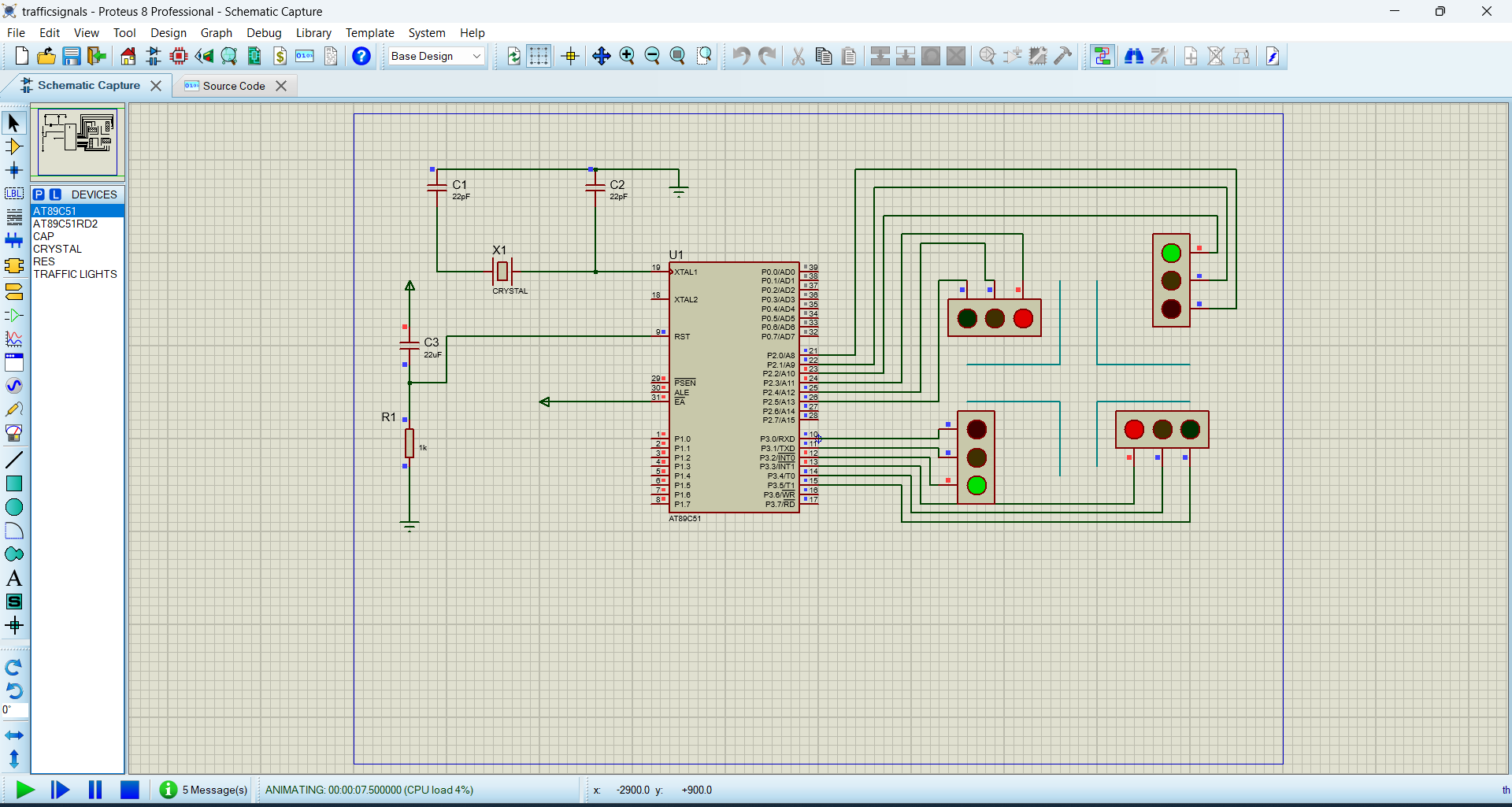
DJNZ R1, D2\_LOOP2

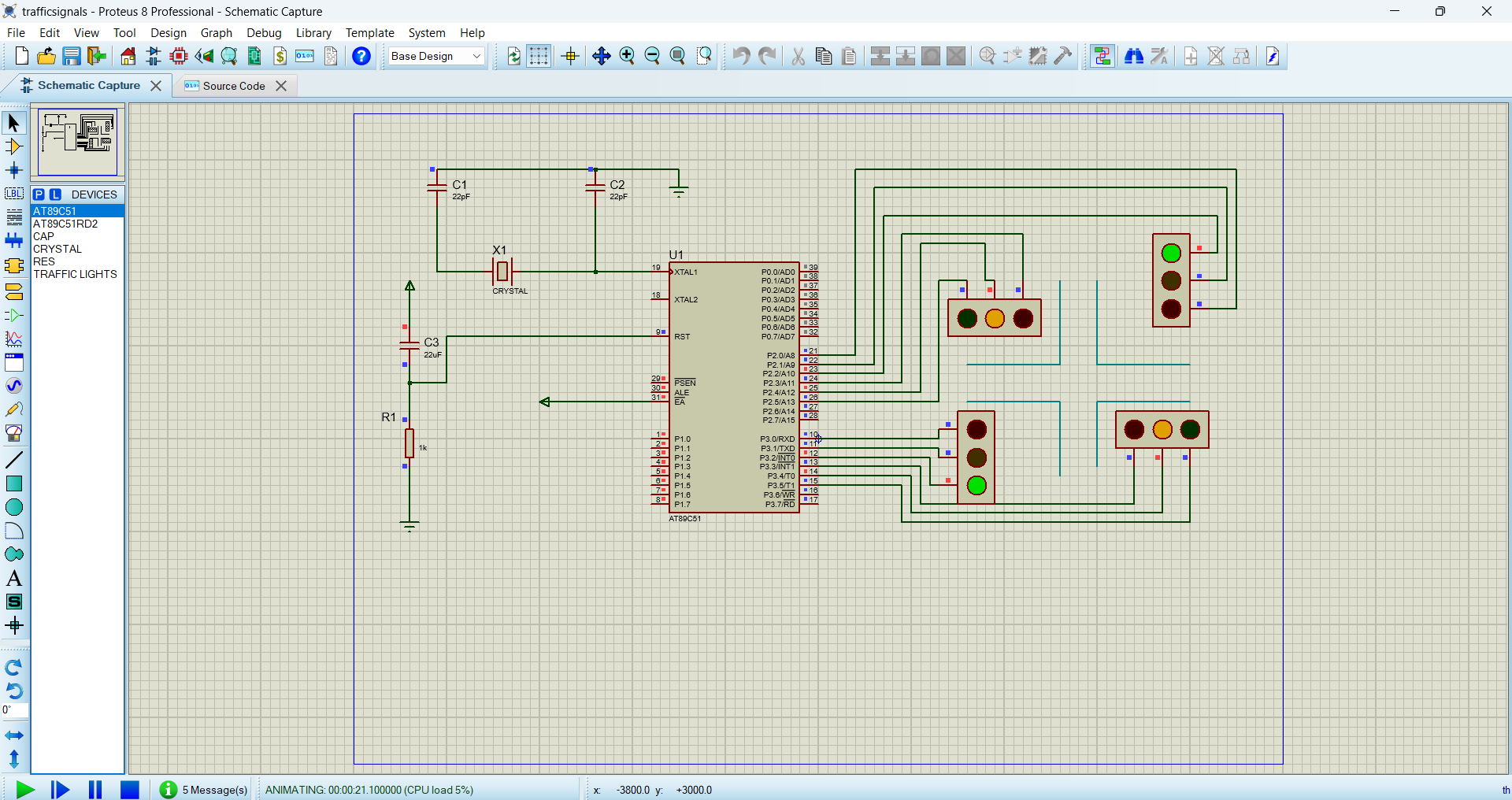
DJNZ R0, D2\_LOOP1

RET

END

**CIRCUIT DIAGRAM:**





**RESULT:**

Thus, the program has been successfully verified and executed