Experiment no -01

**Study of Keil Micro Vision**

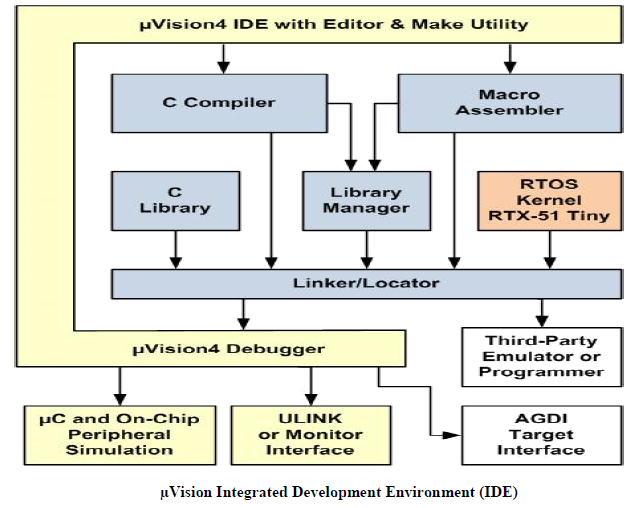
**Introduction to Keil:-**

Embedded system means some combination of computer hardware and programmable software which is specially designed for a particular task like displaying message on LCD. It involves hardware (8051 microcontroller) and software (the code written in assembly language). Some real life examples of embedded systems may involve ticketing machines, vending machines, temperature controlling unit in air conditioners etc. Microcontrollers are nothing without a Program in it.The μVision IDE is the easiest way for most developers to create embedded applications using the Keil development tools. The new Keil μVision4 IDE has been designed to enhance developer's productivity, enabling faster, more efficient program development.

**Keil MicroVision** is a free software which solves many of the main points for an embedded program developer. This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too. μVision4 introduces a flexible window management system, enabling us to drag and drop individual windows anywhere on the visual surface including support for Multiple Monitors.

**Development Tools**

The Keil C51 development tools offer numerous features and advantages that help you to develop embedded applications quickly and successfully. Find out more about the supported devices and the possible tool combinations available for the different 8051 variants. The following block diagram shows the components involved in the build process. The μVision IDE is a window-based software development tool that combines project management and a rich-featured editor with interactive error correction, option setup, make facility, and on-line help. Use μVision to create source files and organize them into a project that defines your target application.



**C Compiler**

The Keil Cx51 Compiler is a full ANSI implementation of the C programming language and supports all standard features of the C language. In addition, numerous extensions have been included to directly support the 8051 and extended 8051 architecture.

**Macro Assembler**

The Keil Ax51 Macro Assembler supports the complete instruction set of the 8051 and all 8051 derivatives.

**Library Manager**

The LIBx51 Library Manager allows you to create the object library from object files created by the compiler and assembler. Libraries are specially formatted, ordered program collections of object modules that may be used by the linker at a later time. When the linker processes a library, only those object modules necessary to create the program are used.

**Linker/Locater**

The Lx51 Linker/Locater creates the final executable 8051 program and combines the object files created by the compiler or assembler, resolves external and public references, and assigns absolute addresses. In addition, it selects and includes the appropriate run-time library modules.

**μVision Debugger :**The μVision Debugger is ideally suited for fast and reliable program debugging. The debugger includes a high-speed simulator capable of simulating an entire 8051 system including on-chip peripherals and external hardware.

**RTOS Kernel**

The RTOS Kernel, describes the advantages of using a real-time kernel like the Keil RTX51 Tiny in embedded systems.

**Creation of HEX File**

Some applications require a HEX file to download the application software into the physical device using a Flash programming utility. μVision creates HEX files with each build process when Create HEX File is enabled in the dialog Options for Target Output.

**Start Debugging**

μVision provides several ways to invoke debugging commands:

* Commands used from the menu Debug or the **Debug Toolbar**.
* Commands entered manually in the Command Window.
* Commands available from the **Context Menu** of the **Editor** or **Disassembly** window.
* Debug Functions executed from an initialization file.

**Start the Debugger**

* Use the **Start/Stop Debug Session** button from the **Debug Toolbar** to start or stop a debugging session.
* The current instruction or high-level statement (the one about to execute) is marked with a yellow arrow. For each step-command, the arrow moves to reflect the new current line or instruction.
* Depending on the **Options for Target — Debug** configuration, μVision loads the application program and runs the startup code (**Run to main ()**).
* μVision saves the editor screen layout and restores the screen layout of the last debug session. When program execution stops, μVision opens an Editor window with the source text or shows MCU instructions in the Disassembly Window.

**Execute Commands**

* + Run the program to the next break point, or type **GO** in the **Command Line**.
  + Halt the program, or press **Esc** while in the **Command Line**
  + Click **Reset** from the **Debug Toolbar** or from the **Debug — Reset CPU Menu** or type **RESET** in the **Command Line** to reset the CPU.

**Single-Stepping Commands**

* To step through the program and into function calls. Alternatively, you can enter **TSTEP** in the **Command Line**, or press **F11**.
* To step over the program and over function calls. Alternatively, you can enter **PSTEP** in the **Command Line**, or press **F10**.
* To step out of the current function. Alternatively, you can enter **OSTEP** in the **Command Line**, or press **Ctrl+F11**.

**On-Chip Peripherals**

There are a number of techniques you must know to create programs that can use the various on-chip peripherals and features of the 8051 family. Use the code examples provided here to get started working with the 8051.

* There is no single standard set of on-chip peripherals for the 8051 family. Instead, 8051 chip vendors use a wide variety of on-chip peripherals to distinguish their parts from each other. The code examples demonstrate how to use the peripherals of a particular chip or family. Be aware that there are more configuration options available than are presented in this text.

Follow the links to the on-chip peripherals:

1. **Header Files** - use the include files to define peripheral registers of the device in use.
2. **Startup Code** - initializes the microcontroller and transfers control to the **main** function.
3. **Special Function Registers** - explains how to use Special Function Registers (SFRs).
4. **Register Bank**s **-** explains how to use Register Banks.
5. **Interrupt Service Routines** - lists the different interrupt variants on 8051 devices.
6. **Interrupt Enable Registers** - shows how to enable the interrupts.
7. **Parallel Port I/O** - explains how to use standard I/O ports.
8. **Timers/Counters** - explains standard timers and counters.
9. **Serial Interface** - explains the implementation of serial UART communication.
10. **Watchdog Timer -** use a watchdog timer to recover from hardware or software failures.
11. **D/A Converter** - convert a digital output voltage to an analog output value.
12. **A/D Converter** - convert an analog input voltage to a digital value.
13. **Power Reduction Modes** - put the device into IDLE or POWER DOWN mode.

Experiment 02

**BLINKING OF LED USING 8051 MICROCONTROLLER USING KEIL AND PROTEUS**

**AIM:**

To Write an assembly language program to LED blink using 8051

**SOFTWARES REQUIRED:**

* Keil software

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* Animated LED(Green)
* CRYSTAL DEVICE
* Cap
* Using terminal click ground and place it two times.
* Connecting pin number 19 and 18 to one end of the the two capacitors
* Now connect both the capacitors with the other end
* Connecting the ground to the capacitor
* Connecting another ground to the led
* Connecting pin number 21 to the led
* Connect the crystal (X1) to the capacitor(C1 & C2)
* Give input in crystal as 16MHz.
* Give input to C1 and C2 as 33pF.
* Give input to AT89C51 as HEX file.
* Start the simulation process

**PROGRAM**

ORG 0000H

UP: SETB P2.0

ACALL DELAY

CLR P2.0

ACALL DELAY

SJMP UP

DELAY: MOV R4,#35

H1:MOV R3,#255

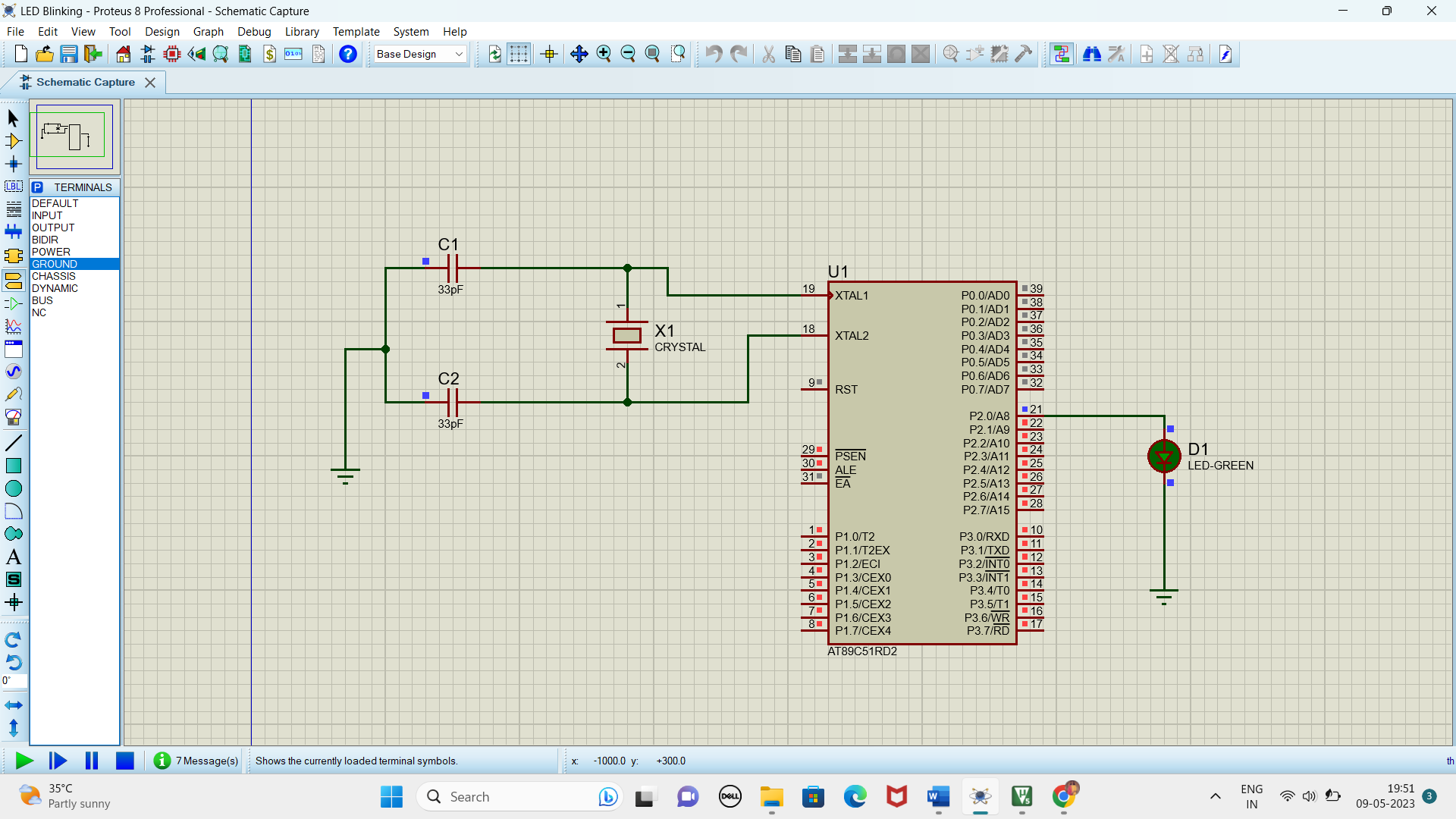
H2:DJNZ R3,H2

DJNZ R4,H1

RET

END

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.

Experiment 3 **GENERATION OF SQUARE WAVE USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program to Generate square wave using 8051.

**SOFTWARE REQUIRED:**

* Keil software.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* Using terminal mode click oscilloscope
* Connecting pin number 21 to A in oscilloscope
* Give input to AT89C51 as HEX file.
* Start the simulation process
* Now change the position to 200 in the Horizontal
* Select 2mV in Channel A
* Drag 180 in Channel D
* Drag -80 in Channel B
* Now adjust to 1mV in Channel A
* Square wave is generated

**PROGRAM**

ORG 0000H

UP:MOV P2,#00H

ACALL DELAY

MOV P2,#OFFH

ACALL DELAY

SJMP UP

DELAY:MOV R4,#100

H1:MOV R3,#255

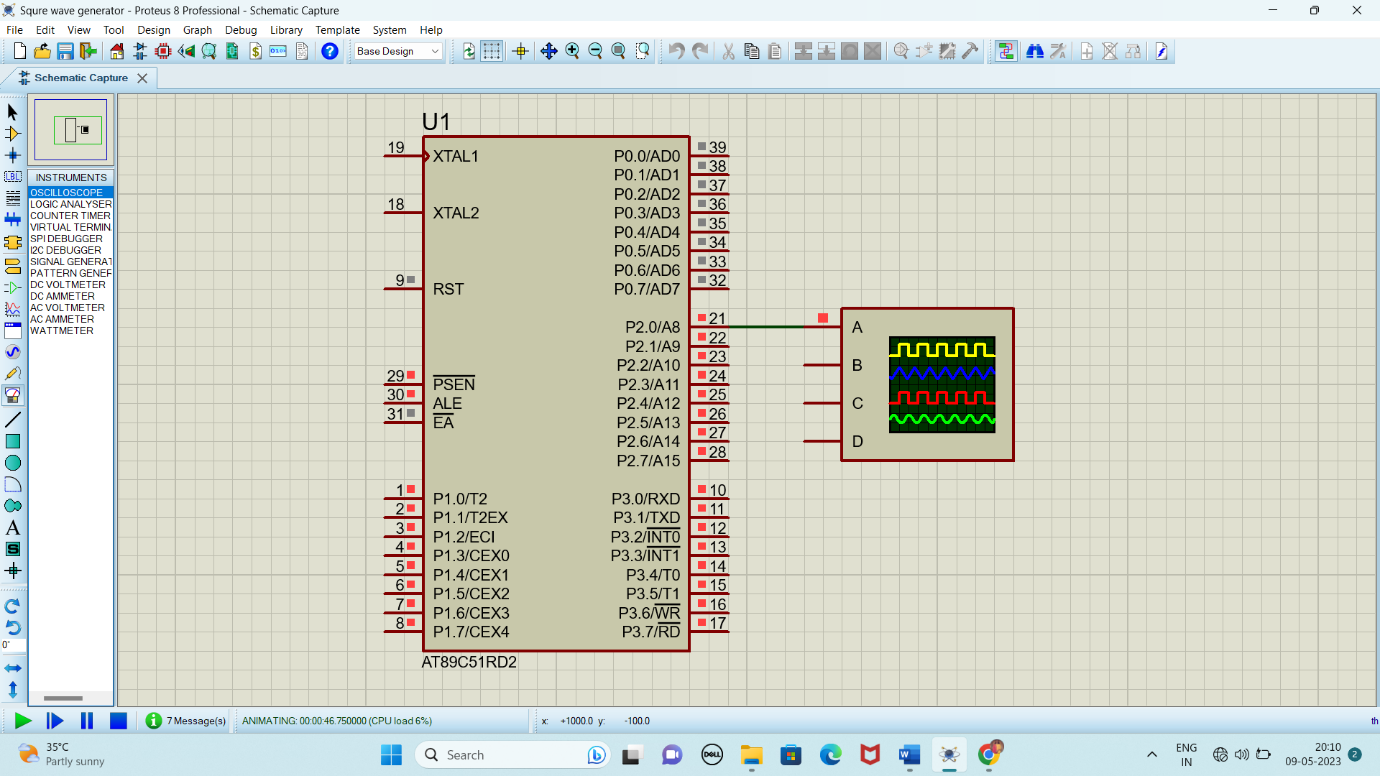
H2:DJNZ R3,H2

DJNZ R4,H1

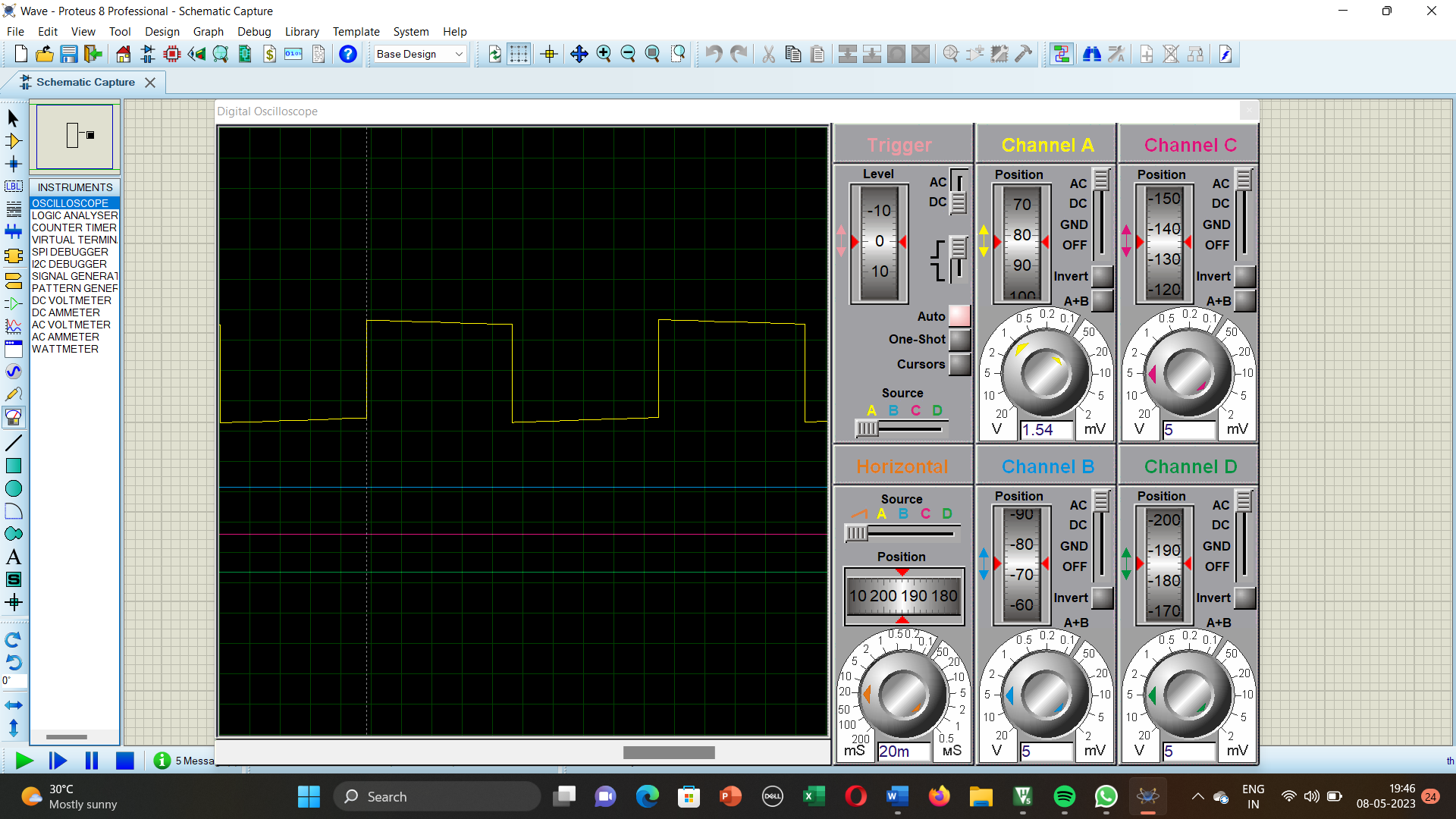
RET

END

**Circuit Diagram:**



**OUTPUT:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 4 **LCD DISPLAY USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for LCD display Using 8051.

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13.Create Hex file

**PROTEUS SOFTWARE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* LM016L
* Connecting pin number 7 from the LCD (LM016L) to pin 1 in the At89c51
* Likewise, connect pin 8, 9, 10, 11, 12, 13 & 14 from the LCD(LM016L) to the pins 2, 3, 4, 5, 6, 7 & 8 of the AT89c51
* Connecting pin 4 from the LCD (LM016L) to the pin 21 in the AT89c51
* Likewise, connect pins 5 & 6 in the LCD (LM016L) to the pins 22 & 23 in the AT89c51
* Select the hex file

Start the simulation process

**PROGRAM:**

ORG 0000H

RS BIT P2.0

RW BIT P2.1

EN BIT P2.2

MOV A,#38H

ACALL CMD

MOV A,#0EH

ACALL CMD

MOV A,#80H

ACALL CMD

MOV A,#06H

ACALL CMD

MOV A,#'S'

ACALL DATA1

MOV A,#'A'

ACALL DATA1

MOV A,#'V'

ACALL DATA1

MOV A,#'E'

ACALL DATA1

MOV A,#'E'

ACALL DATA1

MOV A,#'T'

ACALL DATA1

MOV A,#'H'

ACALL DATA1

MOV A,#'A'

ACALL DATA1

MOV A,#0C0H

ACALL CMD

MOV A,#'U'

ACALL DATA1

MOV A,#'N'

ACALL DATA1

MOV A,#'I'

ACALL DATA1

MOV A,#'V'

ACALL DATA1

MOV A,#'E'

ACALL DATA1

MOV A,#'R'

ACALL DATA1

MOV A,#'S'

ACALL DATA1

MOV A,#'I'

ACALL DATA1

MOV A,#'T'

ACALL DATA1

MOV A,#'Y'

ACALL DATA1

CMD:ACALL READY

MOV P1,A

CLR RS

CLR RW

SETB EN

ACALL DELAY

CLR EN

RET

READY:SETB P1.7

CLR RS

SETB RW

H:CLR EN

ACALL DELAY

SETB EN

JB P1.7,H

RET

DATA1:ACALL READY

MOV P1,A

SETB RS

CLR RW

SETB EN

ACALL DELAY

CLR EN

DELAY:MOV R4,#180

HERE1:MOV R3,#255

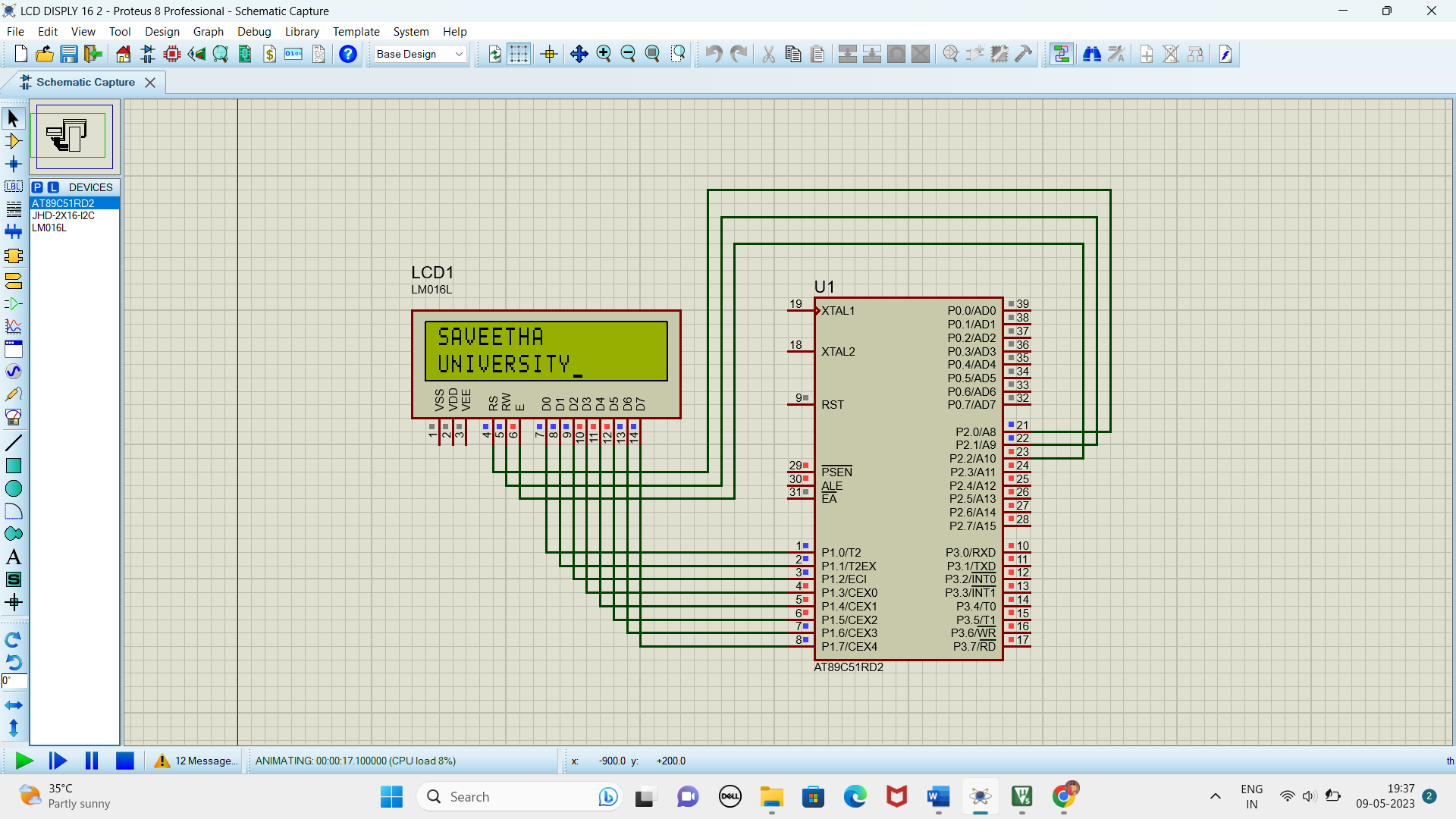
HERE2:DJNZ R3,HERE2

DJNZ R4,HERE1

RET

END

**Circuit Diagram:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 5 **FADE IN FADE OUT OF LED USING 8051 USING KEIL AND PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

13. Create hex file

**PROTEUS PROCEDURE:**

* 1. Open proteus by clicking run as administrator.
  2. Open new project and enter the file name.
  3. Click next, next, next and finish.
  4. Click P symbol and search keyword and place the required components
  5. The components required are:

1. AT89C51
2. Crystal (X1)
3. Led
   * 1. Connecting pin number 18 & 19 from the AT89C51 to the both ends of the Crystal (X1) in the ATC591
     2. Likewise, connect pin 21, 22 & 23 from the AT89C51 to the Led pins
4. Select the hex file
5. Start the simulation process

**PROGRAM:**

#include <REGX52.h>

delay(unsigned int y)

{

unsigned int i,j;

for(i=0;i<y;i++)

{

for(j=0;j<1275;j++){}

}

}

main()

{

while(1)

{

delay(100);

P1\_0 = 0;

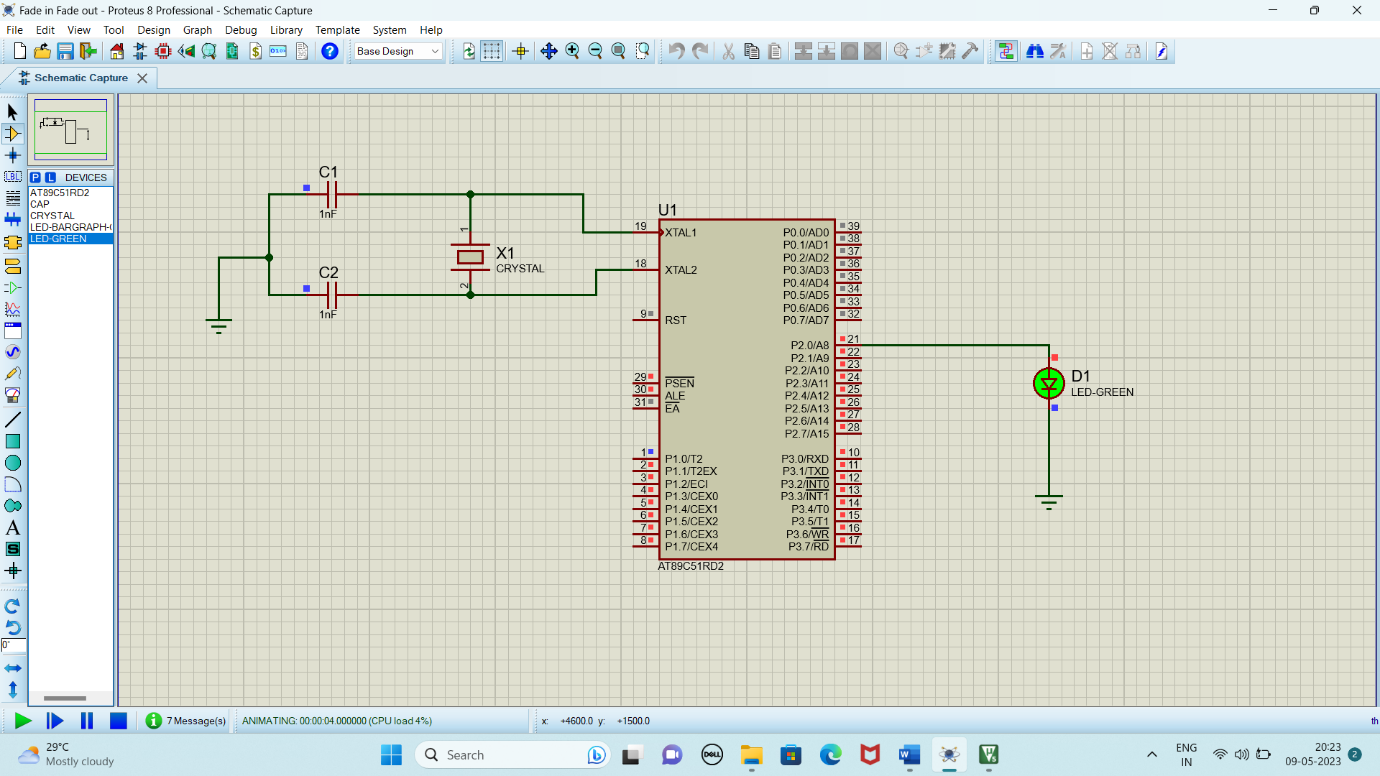
delay(100);

P1\_0 = 1;

}

}

Circuit Diagram:



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 6 **STEPPER MOTOR USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for Stepper Motor Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* Motor Stepper
* ULN2803

* Connecting pin number 21 to 1 in ULN2803
* Likewise, connect pin 22,23 and 24 to the pin 2,3 and 4 in the ULN2803
* Connecting pin 18 from ULN2803 to one end of the stepper motor
* Connect pin 17,16 and 15 to each end of the stepper motor
* Connect the two ends of the stepper motor to each other parallel
* Now, right click and click place and then choose DC to select the Generator
* Connecting the generator to the Stepper motor
* Click on generator and change to 5volts
* Select the hex file
* Start the simulation process

**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

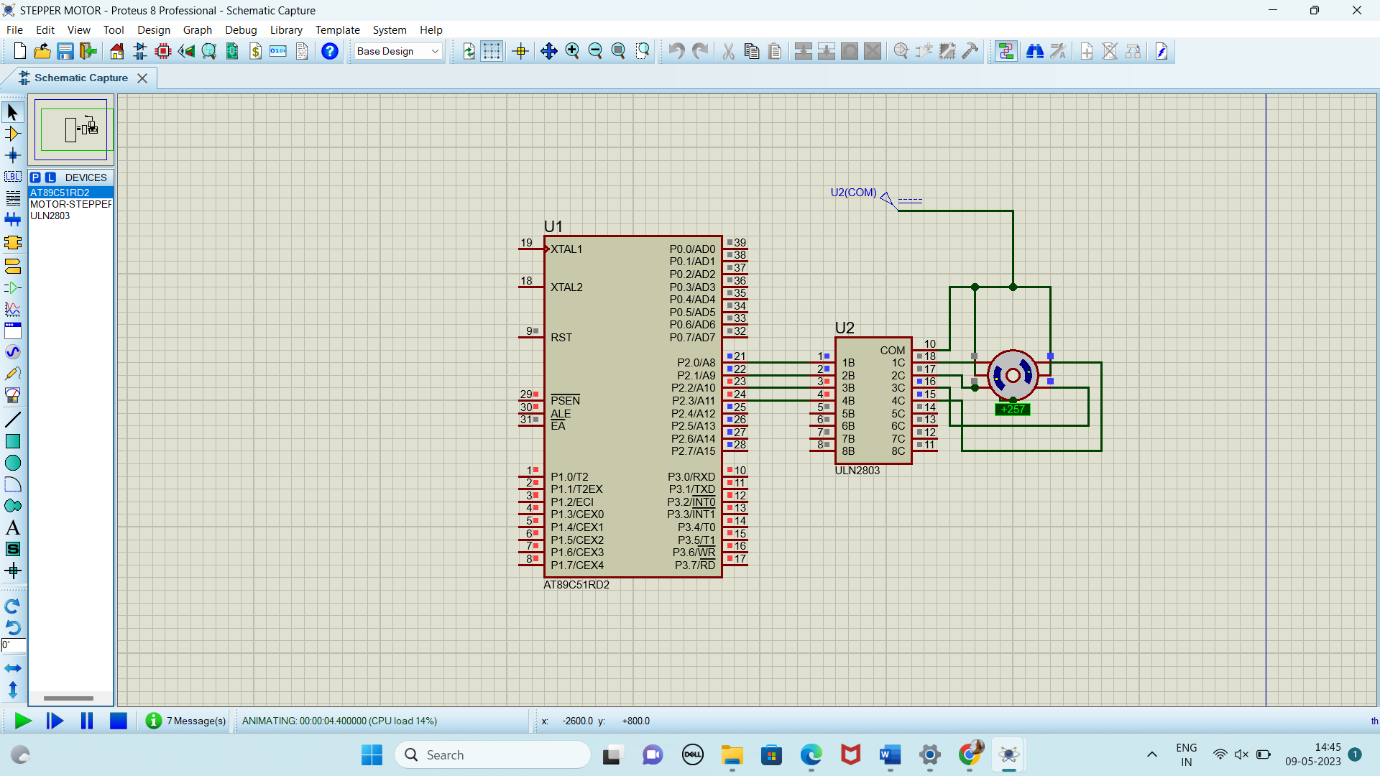
H2:DJNZ R3,H2

DJNZ R4,H1

RET

END

Circuit Diagram:



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 7 **INTERFACING OF RELAY USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for Interfacing of Relay Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* G2RL-14B-CF-DC5
* Lamp
* OZ-SH-105D
* ULN2803
* VSINE

* Connecting pin number 21 to 1 in ULN2803
* Likewise, connect pin 10 to one end of RL-1 and other end to the pin 18 in ULN2803
* Connecting RL-1 to Lamp
* Right click and choose Dc to select Generator
* Connect the generator to the Rl-1
* Click generator and change to 5 volts
* Select the hex file
* Start the simulation process

**PROGRAM:**

ORG 0000H

UP:SETB P2.0

ACALL DELAY

CLR P2.0

ACALL DELAY

SJMP UP

DELAY:MOV R4,#18

H1:MOV R3,#255

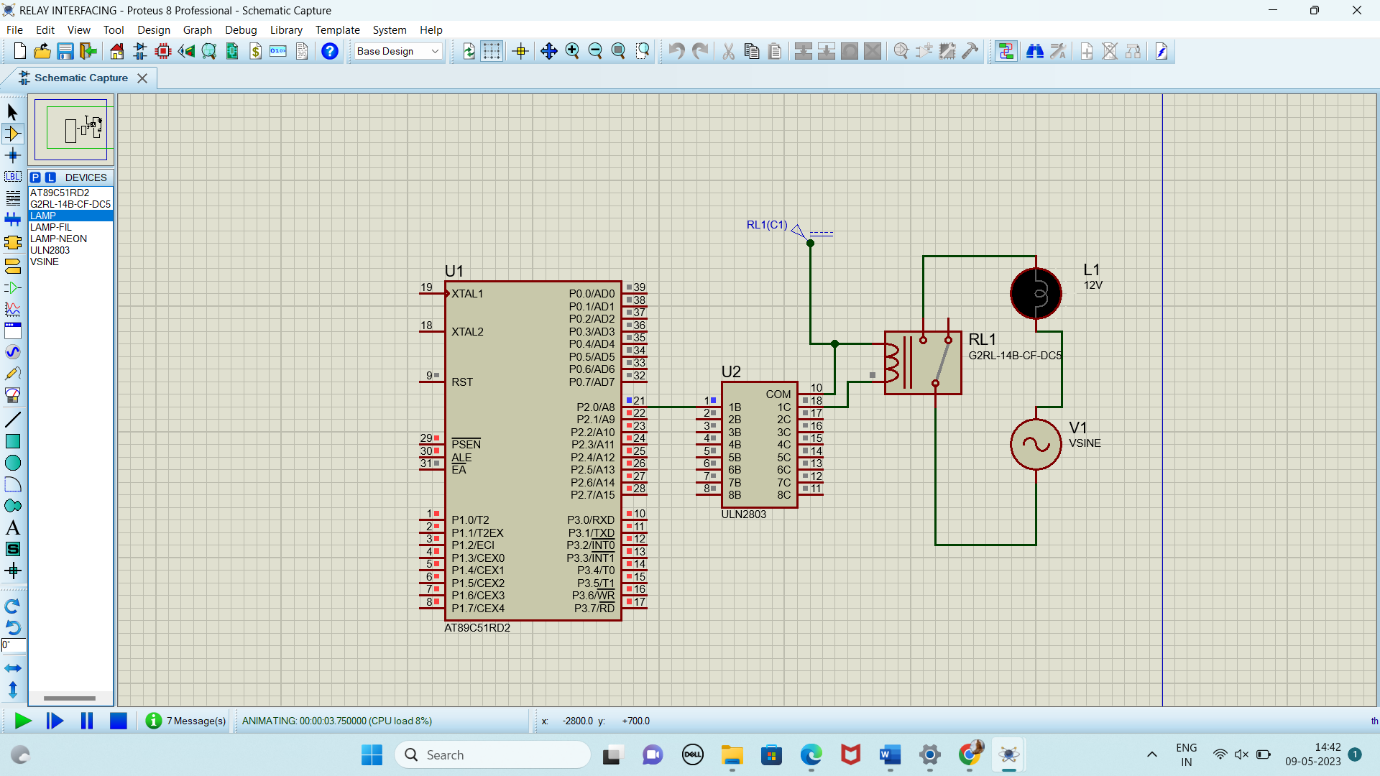
H2:DJNZ R3,H2

DJNZ R4,H1

RET

END

**CIRCUIT DIAGRAM:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 8 **7 SEGMENT DISPLAY USING 8051 USING KEIL AND PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13.Create Hex file

**PROTEUS SOFTWARE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* 7 Segment Anode

* Connecting pin number 21 to the first end of the 7 segment
* Likewise, connect pin 22, 23, 24, 25, 26, 27 & 28 to the ends of the 7 Segment
* Select the hex file
* Start the simulation process

**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

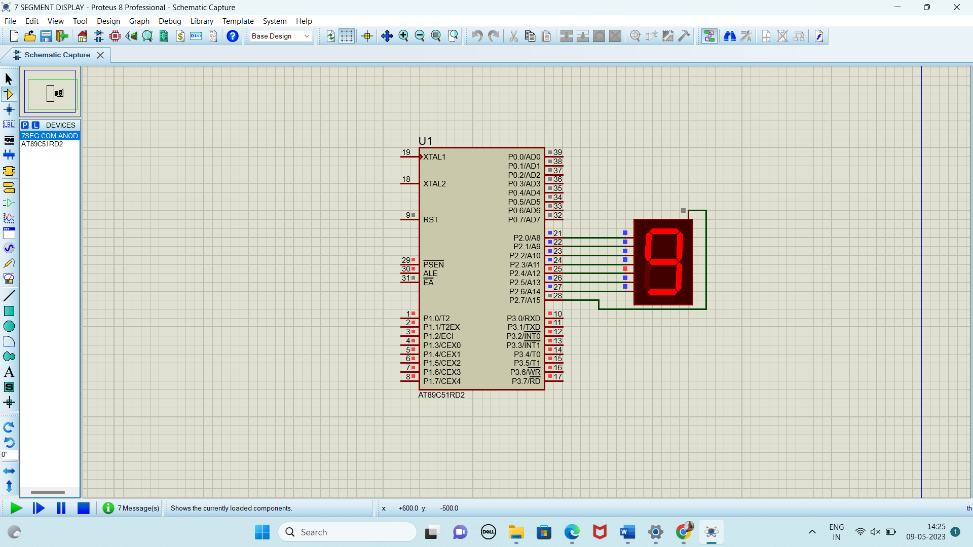
DJNZ R4,H2

DJNZ R5,H1

RET

END

**CIRCUIT DIAGRAM:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 9 **LED TOGGLE USING 8051 USING KEIL AND PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT895C1
* LED-BARAGRAPH-RED
* CRYSTAL
* CAP
* GROUND TERMINAL
* As per declaration so do your connection

5. Connect pin number from 21 to 28 in led baragraphred

6. And then ground to led baragraphred pin you have to connect

7. Left the pin number 11 and 12 in led bargraph and expect this connect all pin to GND

8. U1 19 and 18 pin have to connect in crystal

9. Crystal line you have to connect in GND

10. And then touch the crystal component &change the crystal value and capacitor value

11. Crystal value=16mhz & capacitor value=33pF

12. Give input to AT89C51 as HEX file.

14. Start the simulation process

**PROGRAM:**

ORG 0000H

UP: MOV P2,#55H

ACALL DELAY

MOV P2,#0AAH

ACALL DELAY

SJMP UP

DELAY:MOV R4,#10

H1:MOV R3,#255

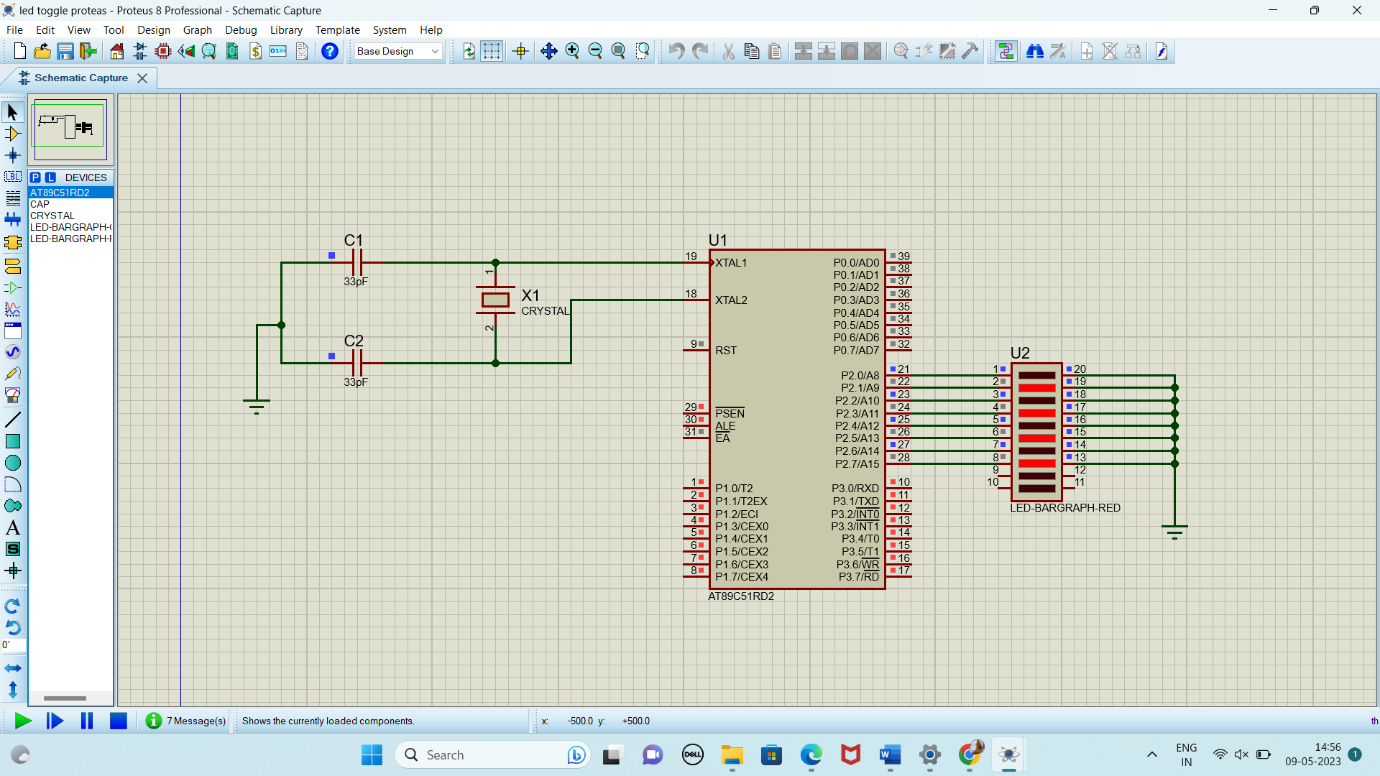
H2:DJNZ R3,H2

DJNZ R4,H1

RET

END

**CIRCUIT DIAGRAM:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 10 **LED CHASER USING 8051 USING KEIL AND PROTEUS**

**AIM:**

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

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**PROCEDURE:**1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51RD2

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

13. Create Hex file

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* Led Bar graph Red
* Crystal
* Choose terminal mode, select ground and place it two times
* Cap

* Connecting pin number 21 to 1 in Led Bargraph
* Likewise, connect pin 22, 23, 24, 25, 26, 27 & 28 to the pins 2, 3, 4, 5, 6, 7 & 8 in the Led Bargraph
* Connecting ground to the Led Bargraph(pin 20)
* Connecting pins 13, 14, 15, 16, 17, 18, 19, 20 to the ground
* Connecting pins 19 & 18 of the AT89C51 to the ends of the two capacitors
* Now, connect both the capacitors together
* Connect the ground to the Capacitors
* Connecting two ends of the Crystal
* Change the frequency of the Crystal to 16MHZ
* Change the Capacitance to 33pF
* Select the Hex file
* Start the simulation process

**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

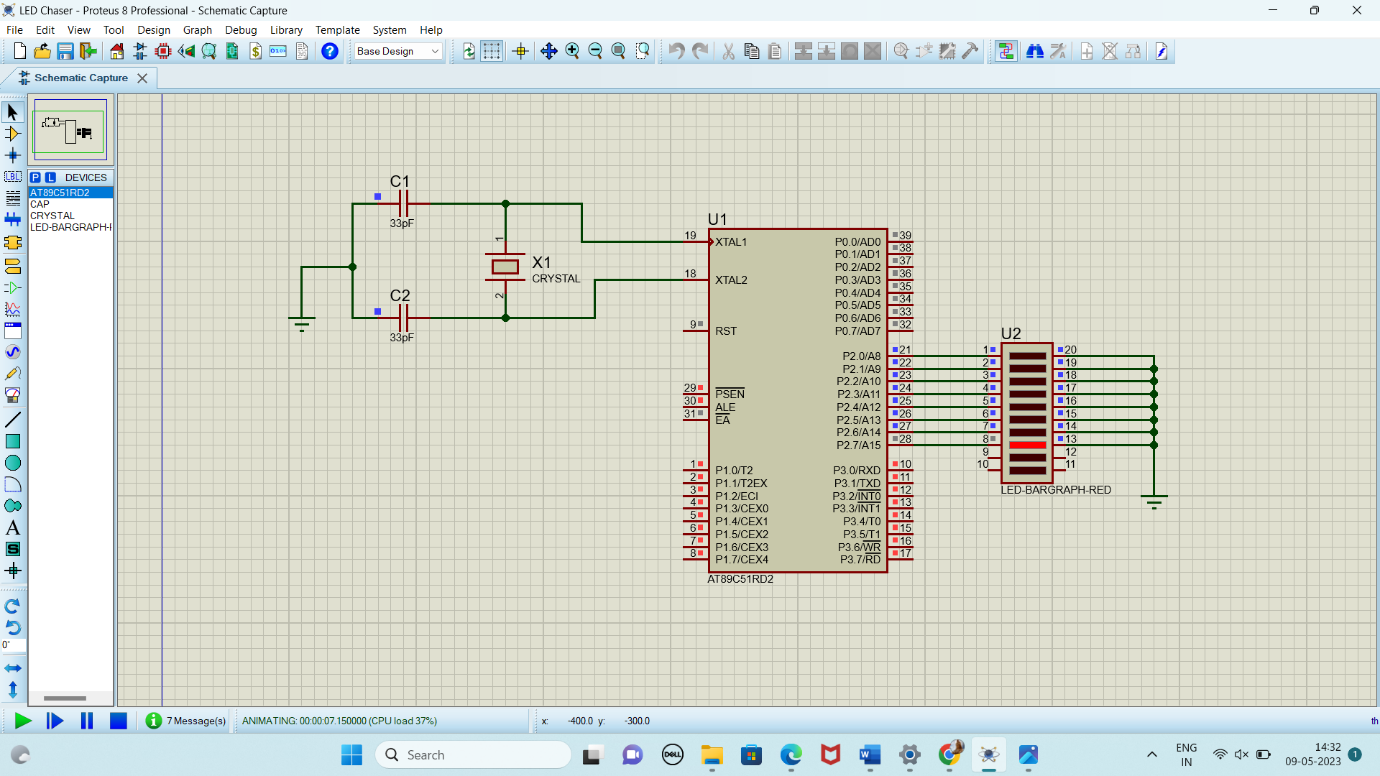
DELAY: MOV R4,#255

H1: DJNZ R4,H1

RET

END

Circuit Diagram:



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 11 **DC MOTOR INTERFACING USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for DC Motor Interface Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**PROCEDURE:**1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

13.Create Hex file

**PROTEUS PROCEDURE:**

1. **1. Open proteus by clicking run as administrator.**
2. **2. Open new project and enter the file name.**
3. **3. Click next, next, next and finish.**
4. **4. click P symbol and search keyword and place it’**
5. **AT89C51**
6. **L293D**
7. **MOTOR**
8. **RESISTOR**
9. **SWITCH**
10. **Using terminal click ground and place it two times.**
11. **Connecting pin number 21-1 up to 28-8**
12. **Connecting the ground to pin 20.**
13. **Connecting the pin no. 19-13 to the wire between pin 20 and GND.**
14. **Connecting pin 19 to the C1 and 18 to C2.**
15. **Connect C1 and C2**

**28.Connect another GND to the wire between C1 and C2.**

**29.Connect the crystal pin no.1 to C1 and pin 2 to C2.**

**30.Give input in crystal as 16MHz.**

**31.Give input to C1 and C2 as 33pF.**

1. **Give input to AT89C51 as HEX file.**

**PROGRAM:**

#include<reg51.h>

sbit sw1 = P1^0;

sbit sw2 = P2^1;

sbit in1 = P2^0;

sbit in2 = P2^1;

void main(void)

{

sw1 = sw2 = 1;

in1 = in2 = 0;

while(1)

{

if(sw1==0 & sw2 ==1)

{

in1 = 1;

in2 = 0;

}

else if(sw1==1 & sw2==0)

{

in1 = 0;

in2 = 1;

}

else

{

in1=0;

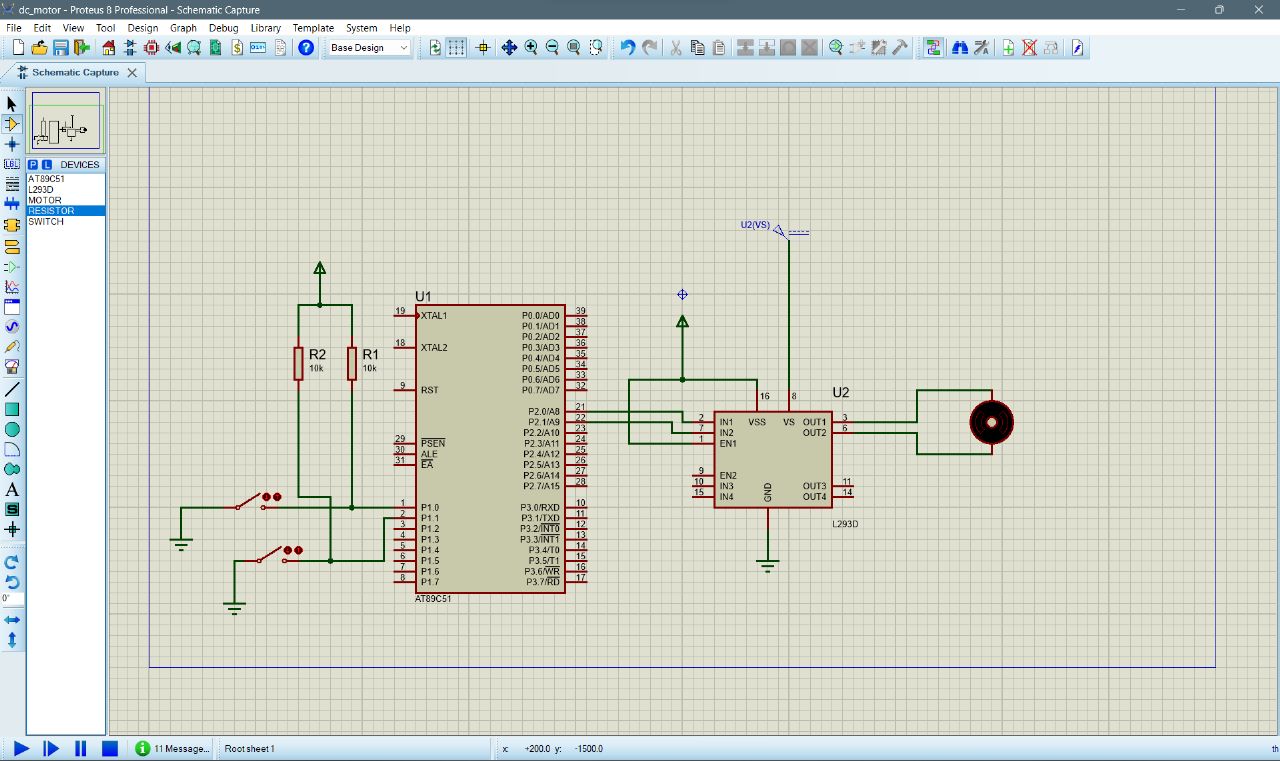
in2=0;

}

}

}

Circuit Diagram:



**RESULT:** Thus the program has been successfully verified and executed.

Experiment 12 **LCD INTERFACING USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for LCD Interfacing Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**PROCEDURE:**1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

13. Create Hex file

**PROTEUS PROCEDURE:**

1. **1. Open proteus by clicking run as administrator.**
2. **2. Open new project and enter the file name.**
3. **3. Click next, next, next and finish.**
4. **4. click P symbol and search keyword and place it’**
5. **AT89C51**
6. **RESPACK-8**
7. **LM016L**
8. **Using terminal click ground and place it two times.**
9. **Connecting pin number 21-1 up to 28-8**
10. **Connecting the ground to pin 20.**
11. **Connecting the pin no. 19-13 to the wire between pin 20 and GND.**
12. **Connecting pin 19 to the C1 and 18 to C2.**
13. **Connect C1 and C2**
14. **Connect another GND to the wire between C1 and C2.**
15. **Connect the crystal pin no.1 to C1 and pin 2 to C2.**
16. **Give input in crystal as 16MHz.**
17. **Give input to C1 and C2 as 33pF.**
18. **Give input to AT89C51 as HEX file.**

**PROGRAM:**

#include<reg51.h>

sbit rs=P1^0;

sbit rw=P1^1;

sbit en=P1^2;

void lcdcmd(unsigned char);

void lcddat(unsigned char);

void delay();

void main()

{

P2=0\*00;//output declaration,data lines d0-d7 connected

while(1)

{

lcdcmd(0x38);//5\*7 matrix crystal

delay();

lcdcmd(0x01);//clear screen

delay();

lcdcmd(0x10);//clear screen

delay();

lcdcmd(0x0c);//display on

delay();

lcddat('S');

delay();

lcddat('S');

delay();

lcddat('I');

delay();

lcddat('M');

delay();

lcddat('A');

delay();

lcddat('I');

delay();

lcddat('S');

delay();

}

}

void lcdcmd(unsigned char val)

{

P2=val;

rs=0;

rw=0;

en=1;

delay();

en=0;

}

void lcddat(unsigned char val)

{

P2=val;

rs=1;

rw=0;

en=1;

delay();

en=0;

}

void delay()

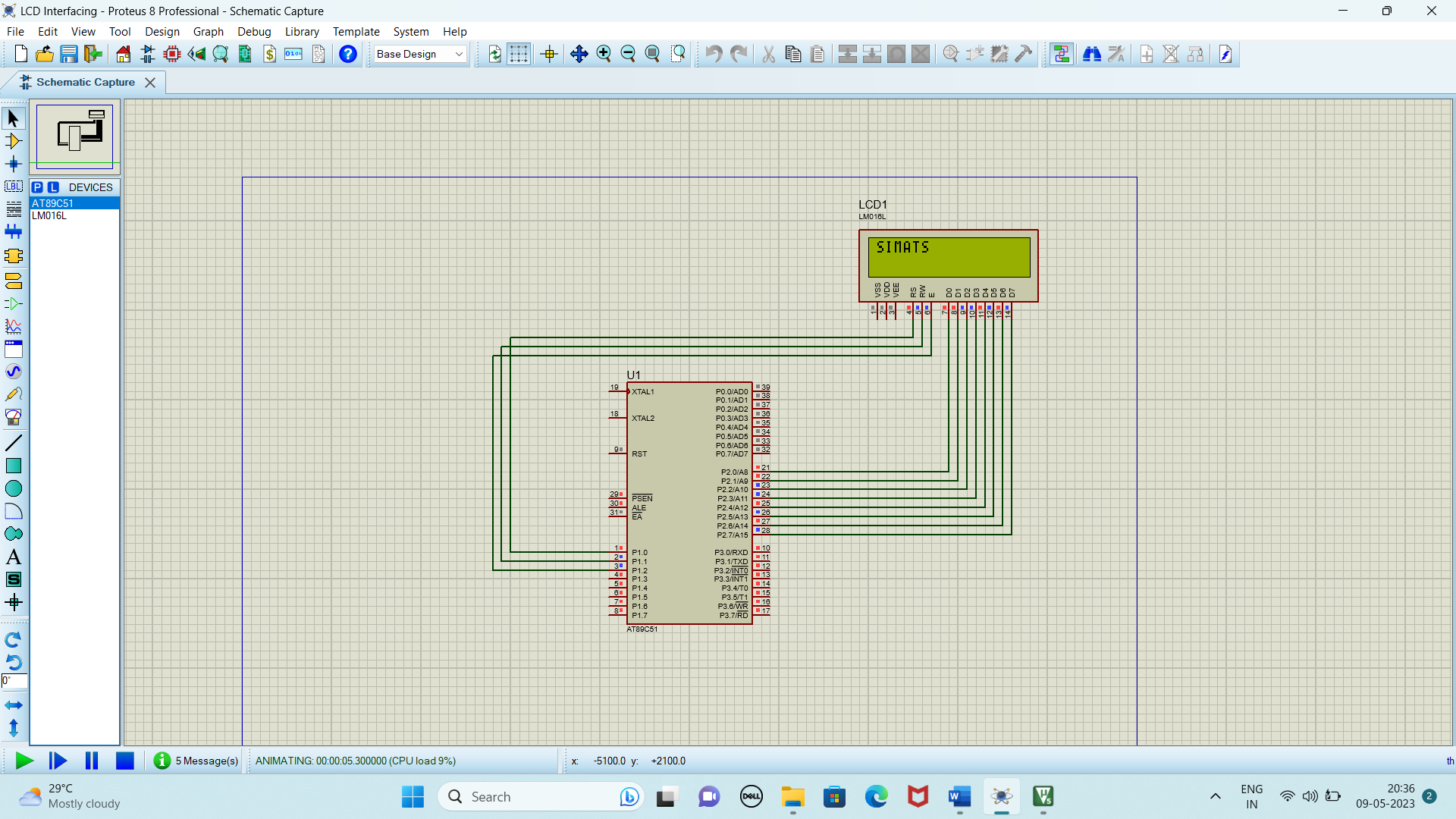
{

unsigned int i;

for(i=0;i<12000;i++);

}

Circuit Diagram:



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 13 **KEYPAD INTERFACING USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for Keypad Interfacing Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

**PROTEUS PROCEDURE:**

**1. Open proteus by clicking run as administrator.**

**2. Open new project and enter the file name.**

**3. Click next, next, next and finish.**

**4. click P symbol and search keyword and place it’**

**AT89C51**

**KEYPAD-PHONE**

**LM016L**

**Using terminal click ground and place it two times.**

**5. Connecting pin number 21-1 up to 28-8**

**6. Connecting the ground to pin 20.**

**7. Connecting the pin no. 19-13 to the wire between pin 20 and GND.**

**8. Connecting pin 19 to the C1 and 18 to C2.**

**9. Connect C1 and C2**

**10. Connect another GND to the wire between C1 and C2.**

**11. Connect the crystal pin no.1 to C1 and pin 2 to C2.**

**12. Give input in crystal as 16MHz.**

**13. Give input to C1 and C2 as 33pF.**

**14. Give input to AT89C51 as HEX file.**

**PROGRAM:**

#include<reg51.h>

#define display\_port P2 //Data pins connected to port 2 on microcontroller

sbit rs = P3^2; //RS pin connected to pin 2 of port 3

sbit rw = P3^3; // RW pin connected to pin 3 of port 3

sbit e = P3^4; //E pin connected to pin 4 of port 3

sbit C4 = P1^0; // Connecting keypad to Port 1

sbit C3 = P1^1;

sbit C2 = P1^2;

sbit C1 = P1^3;

sbit R4 = P1^4;

sbit R3 = P1^5;

sbit R2 = P1^6;

sbit R1 = P1^7;

void msdelay(unsigned int time) // Function for creating delay in milliseconds.

{

unsigned i,j ;

for(i=0;i<time;i++)

for(j=0;j<1275;j++);

}

void lcd\_cmd(unsigned char command) //Function to send command instruction to LCD

{

display\_port = command;

rs= 0;

rw=0;

e=1;

msdelay(1);

e=0;

}

void lcd\_data(unsigned char disp\_data) //Function to send display data to LCD

{

display\_port = disp\_data;

rs= 1;

rw=0;

e=1;

msdelay(1);

e=0;

}

void lcd\_init() //Function to prepare the LCD and get it ready

{

lcd\_cmd(0x38); // for using 2 lines and 5X7 matrix of LCD

msdelay(10);

lcd\_cmd(0x0F); // turn display ON, cursor blinking

msdelay(10);

lcd\_cmd(0x01); //clear screen

msdelay(10);

lcd\_cmd(0x81); // bring cursor to position 1 of line 1

msdelay(10);

}

void row\_finder1() //Function for finding the row for column 1

{

R1=R2=R3=R4=1;

C1=C2=C3=C4=0;

if(R1==0)

lcd\_data('1');

if(R2==0)

lcd\_data('4');

if(R3==0)

lcd\_data('7');

if(R4==0)

lcd\_data('\*');

}

void row\_finder2() //Function for finding the row for column 2

{

R1=R2=R3=R4=1;

C1=C2=C3=C4=0;

if(R1==0)

lcd\_data('2');

if(R2==0)

lcd\_data('5');

if(R3==0)

lcd\_data('8');

if(R4==0)

lcd\_data('0');

}

void row\_finder3() //Function for finding the row for column 3

{

R1=R2=R3=R4=1;

C1=C2=C3=C4=0;

if(R1==0)

lcd\_data('3');

if(R2==0)

lcd\_data('6');

if(R3==0)

lcd\_data('9');

if(R4==0)

lcd\_data('#');

}

void row\_finder4() //Function for finding the row for column 4

{

R1=R2=R3=R4=1;

C1=C2=C3=C4=0;

if(R1==0)

lcd\_data('A');

if(R2==0)

lcd\_data('B');

if(R3==0)

lcd\_data('C');

if(R4==0)

lcd\_data('D');

}

void main()

{

lcd\_init();

while(1)

{

msdelay(30);

C1=C2=C3=C4=1;

R1=R2=R3=R4=0;

if(C1==0)

row\_finder1();

else if(C2==0)

row\_finder2();

else if(C3==0)

row\_finder3();

else if(C4==0)

row\_finder4();

}

}

**Circuit Diagram:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 14 **DIGITAL CLOCK INTERFACING USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for Digital Clock Interfacing Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**1.open the software ,click on project and open new version project.

2.create a new project file

3. enter AT89C51

4.click no

5.click ctrl n and write code

6. open project and click target build

7. open target build and open source file and ADD, CLOSE

8. click target build

9. next debug start and stop

10.open peripherals and select port 2

11. run the program in debug

12. open project and click optional properties and in that give output as hexa file.

**PROTEUS PROCEDURE:**

**1. Open proteus by clicking run as administrator.**

**2. Open new project and enter the file name.**

**3. Click next, next, next and finish.**

**4. click P symbol and search keyword and place it’**

**AT89C51**

**7SEG-MPX4-CA**

**Using terminal click ground and place it two times.**

**5. Connecting pin number 21-1 up to 28-8**

**6. Connecting the ground to pin 20.**

**7. Connecting the pin no. 19-13 to the wire between pin 20 and GND.**

**8. Connecting pin 19 to the C1 and 18 to C2.**

**9. Connect C1 and C2**

**10. Connect another GND to the wire between C1 and C2.**

**11. Connect the crystal pin no.1 to C1 and pin 2 to C2.**

**12. Give input in crystal as 16MHz.**

**13. Give input to C1 and C2 as 33pF.**

**14. Give input to AT89C51 as HEX file.**

**PROGRAM:**

include <reg51.h>

#define msec 1

unsigned int arr[10]={0x40,0xF9,0x24,0x30,0x19,0x12,0x02,0xF8,0x00,0x10};

sbit d4=P1^0;

sbit d3=P1^1;

sbit d2=P1^2;

sbit d1=P1^3;

sbit d0=P1^4;

sbit d= P1^5;

unsigned int v1,v2,v3,v4,v0,v5,v6;

void delay(unsigned int count)

{

unsigned int j,k;

for (j=0;j<=count;j++)

for (k=0;k<=5;k++);

}

void main()

{

v1=v2=v3=v4=v0=v5=v6=0;

while(1)

{

{

v0=v0+1;

if(v0==130)

{

v0=0;

v1=v1+1;

}

P2=0xFF;

d = 1;

d3 = d2 = d4 = d0 = d1= 0;

P2 = arr[v1];

delay(msec);

if(v1==10)

{

v1=0;

v2=v2+1;

}

P2=0xFF;

d0 = 1;

d4 = d3 = d1 =d=d2= 0;

P2 = arr[v2];

delay(msec);

if(v2==6)

{

v2=0;

v3=v3+1;

}

P2=0xFF;

d1 = 1;

d2 = d4 = d3 =d=d0= 0;

P2 = arr[v3];

delay(msec);

if(v3==10)

{

v3=0;

v4=v4+1;

}

P2=0xFF;

d2 = 1;

d3 = d4 = d1 =d=d0= 0;

P2 = arr[v4];

delay(msec);

if(v4==6)

{

v4=0;

v5=v5+1;

}

P2=0xFF;

d3 = 1;

d0 = d2 = d1 =d=d4= 0;

P2 = arr[v5];

delay(msec);

if(v5==10)

{

v5=0;

v6=v6+1;

}

P2=0xFF;

d4 = 1;

d3 = d2 = d1 =d=d0= 0;

P2 = arr[v6];

delay(msec);

if(v6==1&&v5==2)

{

v1=0;

v2=0;

v3=0;

v4=0;

v5=0;

v6=0;

}

delay(msec);

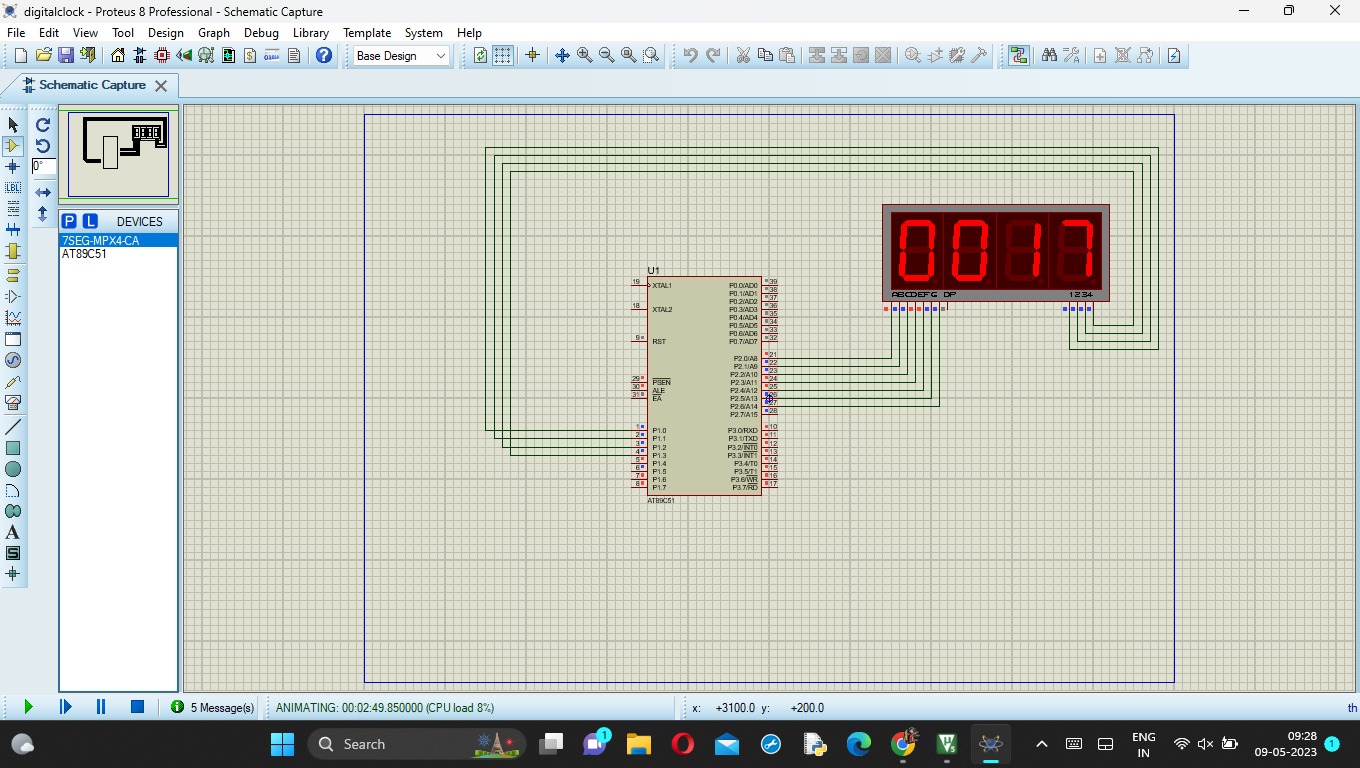
P2=0xFF;

}

}

}

Circuit Diagram:



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 15 **AUTOMATIC DOOR LOCKING SYSTEM USING 8051 USING KEIL AND PROTEUS**

**AIM:**

Write an assembly language program for Automatic Door Locking System Using 8051 using Keil and Proteus

**SOFTWARE REQUIRED:**

* Keil software 5.
* Proteus 8 software.

**KEIL PROCEDURE:**

1. Open the software, Click on project and open new version project.

2. Create a new project file

3. Enter AT89C51

4. Click NO

5. Click [Ctrl +N] and Type the code

6. Open project and click Build target

7. Open Build target and open source file and ADD, CLOSE

8. Click build target

9. Next debug start and stop

10. Open peripherals and select port 2

11. Now run the program in Debug

12. Open project and click optional properties and in that give output as hex file.

13. Create hex file.

**PROTEUS PROCEDURE:**

* Open proteus by clicking run as administrator.
* Open new project and enter the file name.
* Click next, next, next and finish.
* Click P symbol and search keyword and place the required components

The components required are:

* AT89C51
* L293D
* MOTOR DC
* BUTTON
* Using terminal click ground and place it two times.
* As per declaration so do your connection

5. The button which is connected with [1]

6. And the one end is connected with IO and the other end is connected with groud

7. Touch the button and name it in string as Sensor here

8. You need to add driver IC with two inputs these two which as connected with 2&3

9. Motor is connected with correct corresponding output lines

10. The both GND should connect with both lines and enable lines also

11. Give direct power connection and you have to place the control and should connected with power line

12. And at last ground pin connected with GND line

13. Give input to AT89C51 as HEX file.

14. Start the simulation process

**PROGRAM:**

#include<reg52.h>

sbit r0=P2^0; sbit r1=P2^1; sbit r2=P2^2; sbit r3=P2^3; sbit c0=P2^5; sbit c1=P2^6;

sbit c2=P2^7; sbit en=P3^6; sbit rs=P3^5; sbit rw=P3^7; sbit lock=P3^0;

char t1[]="Enter PIN:";

char t2[]="Access Granted";

char t3[]="Access Denied";

char pin[]="1234";

char pinEntered[4];

unsigned int m = 0;

unsigned int flag = 0;

void delay(unsigned int no)

{

unsigned int i,j;

for(j=0;j<=no;j++) for(i=0;i<=10;i++);

}

void lcdcmd(unsigned int command){

P1=command; rw=0; rs=0; en=0; delay(1000); en=1; delay(1000); en=0;

}

void lcddata(char data1)

{

P1=data1; rw=0; rs=1; en=0; delay(1000); en=1; delay(1000); en=0;

}

void lcdint(){

lcdcmd(0x30); delay(1000); lcdcmd(0x30); delay(1000); lcdcmd(0x30); delay(1000);

lcdcmd(0x30); delay(1000); lcdcmd(0x30); delay(1000); lcdcmd(0x38); delay(1000);

lcdcmd(0x01); delay(1000); lcdcmd(0x0F); delay(1000); lcdcmd(0x80); delay(1000);

}

char keypad()

{

char c='a';

while(c!='s'){

r0=0;r1=1;r2=1;r3=1;

if(c0==0){lcddata('1');P0=0xF0;delay(10000);c='s';return '1';}

if(c1==0){lcddata('2');P0=0xF0;delay(10000);c='s';return '2';}

if(c2==0){lcddata('3');P0=0xF0;delay(10000);c='s';return '3';}

r0=1;r1=0;r2=1;r3=1;

if(c0==0){lcddata('4');P0=0xF0;delay(10000);c='s';return '4';}

if(c1==0){lcddata('5');P0=0xF0;delay(10000);c='s';return '5';}

if(c2==0){lcddata('6');P0=0xF0;delay(10000);c='s';return '6';}

r0=1;r1=1;r2=0;r3=1;

if(c0==0){lcddata('7');P0=0xF0;delay(10000);c='s';return '7';}

if(c1==0){lcddata('8');P0=0xF0;delay(10000);c='s';return '8';}

if(c2==0){lcddata('9');P0=0xF0;delay(10000);c='s';return '9';}

r0=1;r1=1;r2=1;r3=0;

//if(c0==0){lcddata('\*');P0=0xF0;delay(10000);c='s';return '1';}

if(c1==0){lcddata('0');P0=0xF0;delay(10000);c='s';return '0';}

// if(c2==0){lcddata('#');P0=0xF0;delay(10000);c='s';return '1';}

}

}

void main()

{

unsigned int i=0;

P1=0x00; P2=0xF0; P3=0x00;

lcdint();

while(1){

i=0;

while(t1[i]!='\0')

{

lcddata(t1[i]);

i++;

}

lock=0;

lcdcmd(0xC0);

for(i=0;i<=3;i++)

pinEntered[i] =keypad();

i=0;

lcdcmd(0x01);

if(pinEntered[0]==pin[0])

{ i++;

if(pinEntered[1]==pin[1])

{

if(pinEntered[2]==pin[2])

{

if(pinEntered[3]==pin[3])

{

lock=1; i=0;

while(t2[i]!='\0')

{

lcddata(t2[i]);

i++;

}i=0;

}

}

}

}else{ i=0;

while(t3[i]!='\0')

{

lcddata(t3[i]);

i++;

} i=0; }

if(i!=0)

{ //Download Full Code For Better Understanding

i=0;

while(t3[i]!='\0')

{

lcddata(t3[i]);

i++;

}

}

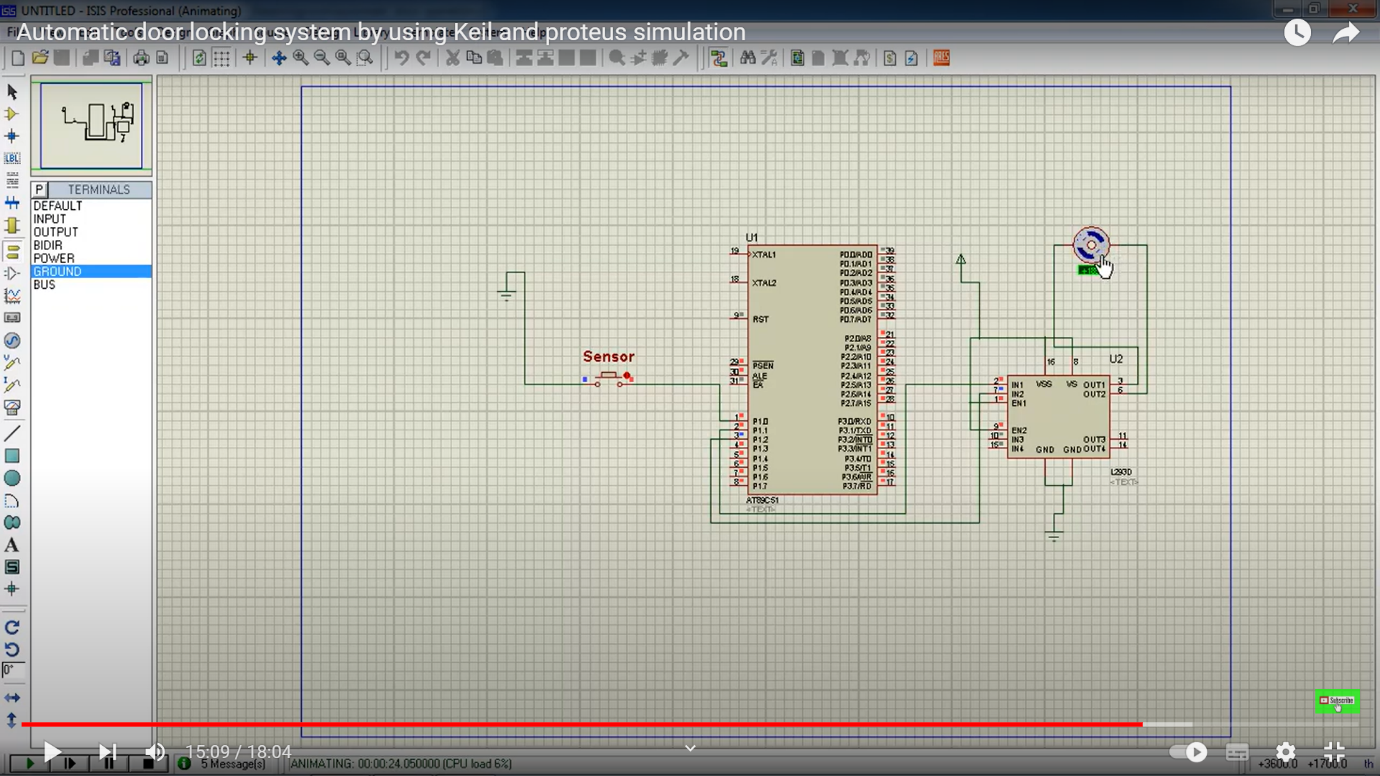
delay(1000000);

lcdcmd(0x01);

lcdcmd(0x80);

i=0;

**CIRCUIT DIAGRAM:**



**RESULT:**

Thus the program has been successfully verified and executed.

Experiment 16

**BLINKING OF LED USING ARDUINO IDE AND PROTEUS**

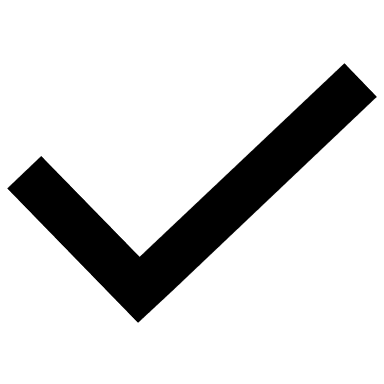
**AIM:**

To Write an assembly language program to LED blink using arduino

**SOFTWARES REQUIRED:**

* Arduino IDE software
* Proteus Software

**ARDUINO PROCEDURE:**

1. Open the software.
2. Click [Ctrl +N] to open new sketch and Type the code.
3. Click Verify. 
4. This will compile the code and check for errors.
5. After compiling, click file and select preferences.
6. Check the box “Compile” and click OK.
7. Compile the code again to generate Hex File.
8. The location of the hex file can be found in the ouput black box in the formal of .hex.

**PROTEUS PROCEDURE:**

1. Open proteus by clicking run as administrator.
2. Open new project and enter the file name.
3. Click next, next, next and finish.
4. Click P symbol and search keyword and place the required components
5. The components required are:

* Arduino UNO R3 V1.0
* Animated LED(Green)
* Using terminal select ground.

1. Connect pin number 8 to one end of the led.
2. Connect another end of the led to the ground.
3. Set the clock frequency of Arduino UNO as 10MHz.
4. Give input to Arduino UNO as HEX file.
5. Start the simulation process

**PROGRAM**

int led=8;

void setup()

{

 pinMode(led, OUTPUT);

}

void loop()

{

digitalWrite(led, HIGH);

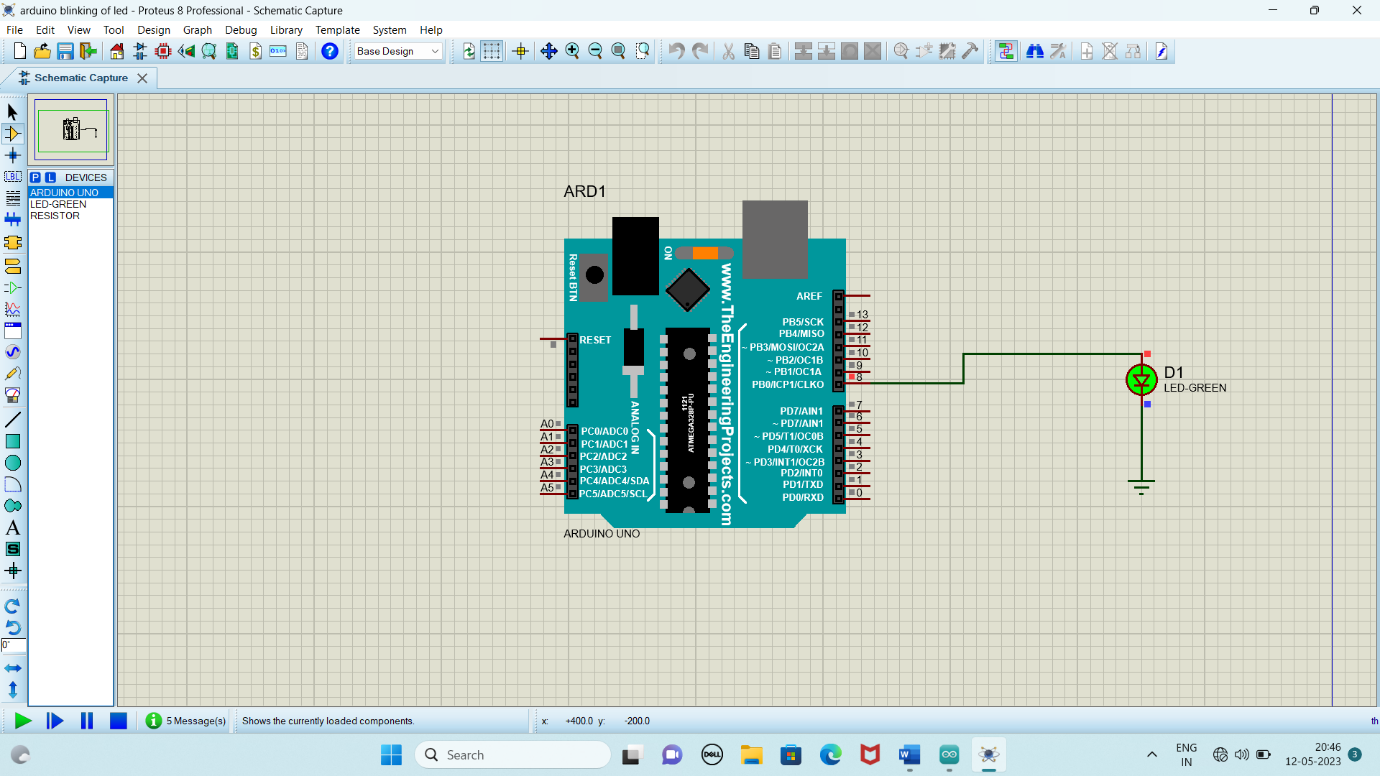
delay(1000);//Milliseconds

digitalWrite(led, LOW);

delay(1000);

}

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.

Experiment 17

**FADE IN FADE OUT OF LED USING ARDUINO IDE AND PROTEUS**

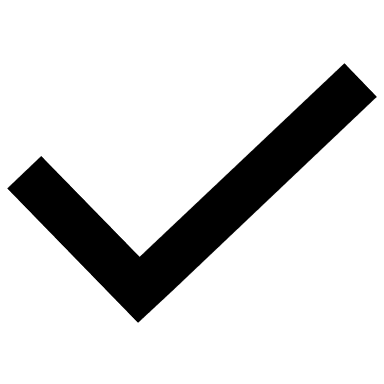
**AIM:**

To Write an assembly language program to Fade in Fade Out of LED using Arduino

**SOFTWARES REQUIRED:**

* Arduino IDE software
* Proteus Software

**ARDUINO PROCEDURE:**

1. Open the software.
2. Click [Ctrl +N] to open new sketch and Type the code.
3. Click Verify. 
4. This will compile the code and check for errors.
5. After compiling, click file and select preferences.
6. Check the box “Compile” and click OK.
7. Compile the code again to generate Hex File.
8. The location of the hex file can be found in the ouput black box in the formal of .hex.

**PROTEUS PROCEDURE:**

1. Open proteus by clicking run as administrator.
2. Open new project and enter the file name.
3. Click next, next, next and finish.
4. Click P symbol and search keyword and place the required components
5. The components required are:
   * Arduino UNO R3 V1.0
   * Animated LED(Green)
   * Using terminal select ground.
6. Connect pin number 9 to one end of the led.
7. Connect another end of the led to the ground.
8. Set the clock frequency of Arduino UNO as 10MHz.
9. Give input to Arduino UNO as HEX file.
10. Start the simulation process

**PROGRAM**

int led = 9;

int brightness = 0;

int fadeAmount = 5;

void setup() {

  pinMode(led, OUTPUT);

}

void loop(){

  analogWrite(led, brightness);

  brightness = brightness + fadeAmount;

  if(brightness <=0 || brightness >= 255 ) {

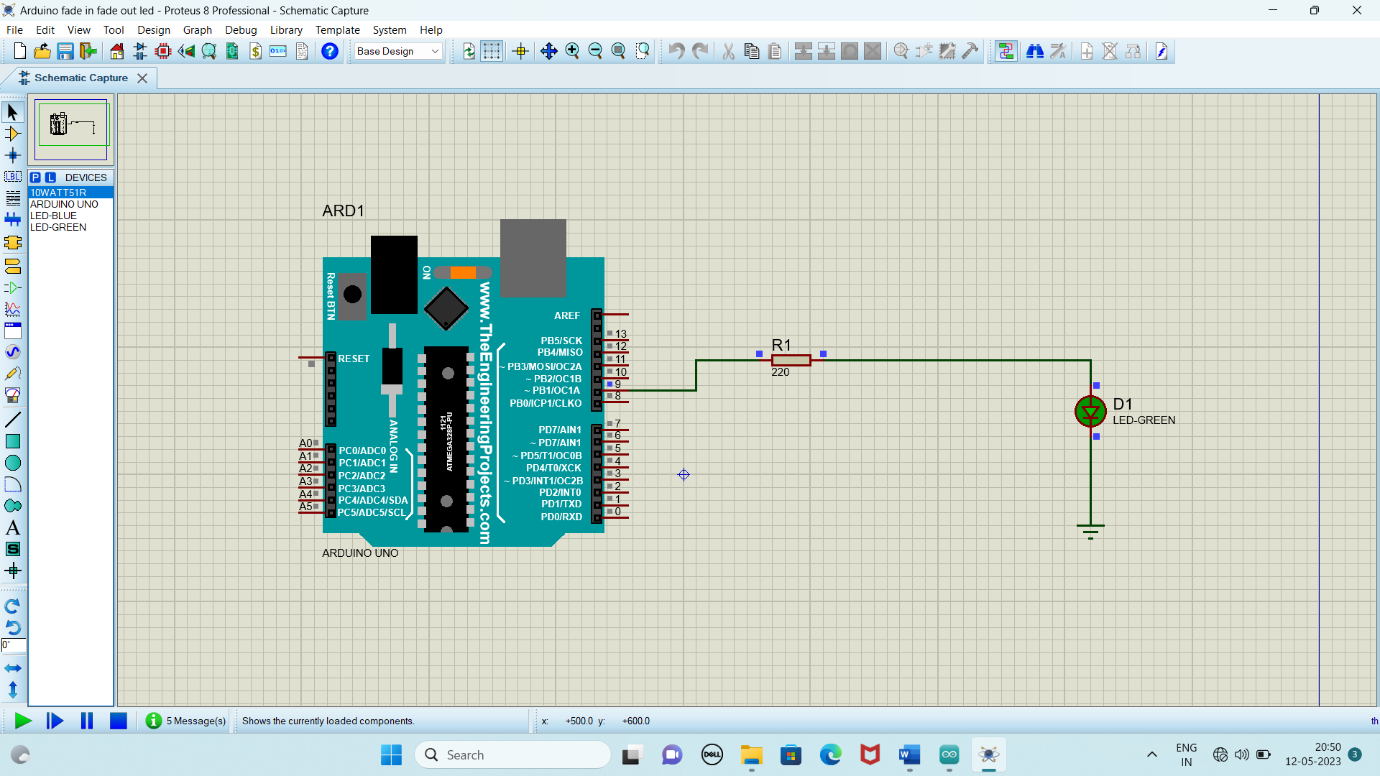
    fadeAmount = -fadeAmount;

  }

  delay(30);

}

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.

Experiment 18

**INTERFACING OF LCD USING ARDUINO IDE AND PROTEUS**

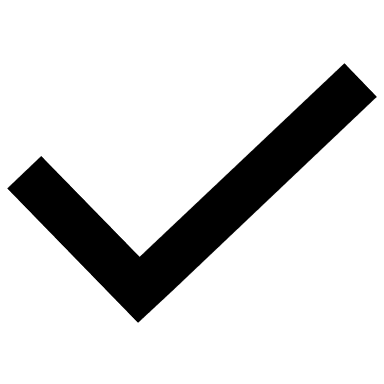
**AIM:**

To Write an assembly language program for interfacing of LCD using Arduino

**SOFTWARES REQUIRED:**

* Arduino IDE software
* Proteus Software

**ARDUINO PROCEDURE:**

1. Open the software.
2. Click [Ctrl +N] to open new sketch and Type the code.
3. Click Verify. 
4. This will compile the code and check for errors.
5. After compiling, click file and select preferences.
6. Check the box “Compile” and click OK.
7. Compile the code again to generate Hex File.
8. The location of the hex file can be found in the ouput black box in the formal of .hex.

**PROTEUS PROCEDURE:**

1. Open proteus by clicking run as administrator.
2. Open new project and enter the file name.
3. Click next, next, next and finish.
4. Click P symbol and search keyword and place the required components
5. The components required are:

* Arduino UNO R3 V1.0
* 16x2 Alphanumeric LCD (LM016L)
* Using terminal select ground.

1. Connect pin number 1,5 of LCD to the ground.
2. Connect pin 2,3,4,5 of Arduino to 14,13,12,11 of LCD.
3. Then connect pin 11,12 of Arduino to 6,4 of LCD
4. Set the clock frequency of Arduino UNO as 10MHz.
5. Give input to Arduino UNO as HEX file.
6. Start the simulation process.

**PROGRAM**

#include<LiquidCrystal.h>

const int rs=12,en=11,d4=5,d5=4,d6=3,d7=2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

  lcd.begin(16,2);

  lcd.print("SIMATS");

  }

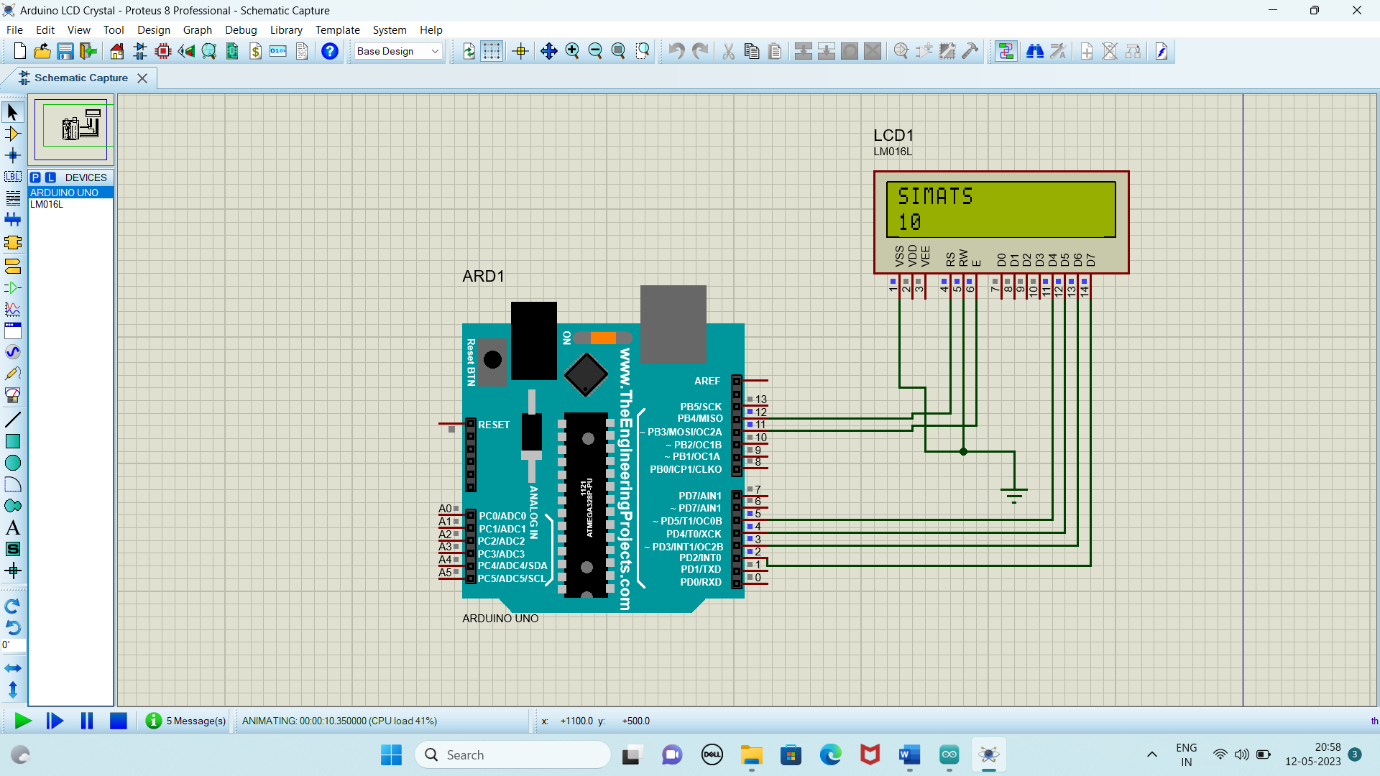
void loop(){

  lcd.setCursor(0,1);

  lcd.print(millis()/1000);

}

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.

Experiment 19

**INTERFACING RFID MODULE USING ARDUINO IDE AND PROTEUS**

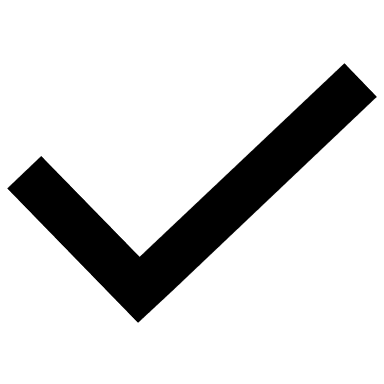
**AIM:**

To Write an assembly language program for interfacing of RFID using Arduino

**SOFTWARES REQUIRED:**

* Arduino IDE software
* Proteus Software

**ARDUINO PROCEDURE:**

1. Open the software.
2. Click [Ctrl +N] to open new sketch and Type the code.
3. Click Verify. 
4. This will compile the code and check for errors.
5. After compiling, click file and select preferences.
6. Check the box “Compile” and click OK.
7. Compile the code again to generate Hex File.
8. The location of the hex file can be found in the ouput black box in the formal of .hex.

**PROTEUS PROCEDURE:**

1. Open proteus by clicking run as administrator.
2. Open new project and enter the file name.
3. Click next, next, next and finish.
4. Click P symbol and search keyword and place the required components
5. The components required are:

* Arduino UNO R3 V1.0
* LED Yellow Animated
* Resistor 10k
* Virtual Terminal - 2
* Using terminal select ground.

1. Connect pin number 13 of arduino to the end of LED.
2. Connect the other end of LED to one end of resistor.
3. Connect the other end of resistor to ground.
4. Connect Pin 1 to RXD of one virtual terminal
5. Connect Pin 0 to TXD of another virtual terminal
6. Set the clock frequency of Arduino UNO as 10MHz.
7. Give input to Arduino UNO as HEX file.
8. Start the simulation process.
9. Give Input in the black box of virtual terminal.

**PROGRAM**

void setup(){

  Serial.begin(9600);

  pinMode(13,OUTPUT);

  Serial.println("please scan your RFID TAG");

}

void loop(){

  while(Serial.available()>0)

  {

    c=Serial.read();

    count++;

    id +=c;

    if(count == 12)

    {

      Serial.print(id);

      //break;

      if(id=="AB123456789A")

      {

        Serial.println("Valid TAG");

        digitalWrite(13,HIGH);

      }

      else

      {

        digitalWrite(13,LOW);

        Serial.println("Invalid TAG");

      }

    }

  }

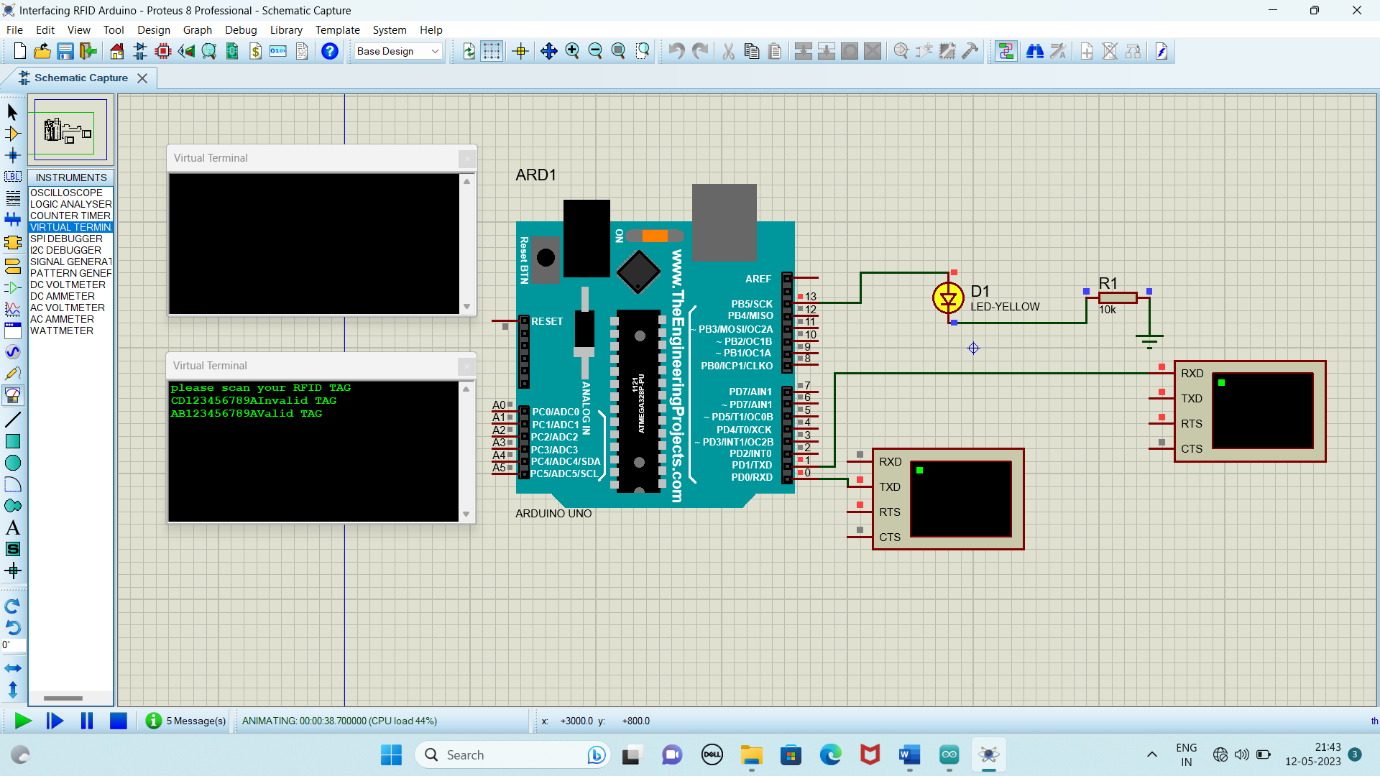
  count = 0;

  id="";

  delay(500);

}

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.

Experiment 20

**INTERFACING OF ULTRASONIC SENSOR USING ARDUINO IDE AND PROTEUS**

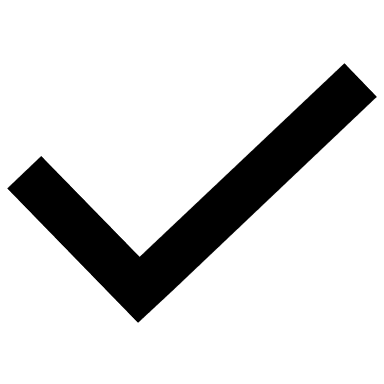
**AIM:**

To Write an assembly language program for interfacing of Ultrasonic Sensor using Arduino

**SOFTWARES REQUIRED:**

* Arduino IDE software
* Proteus Software

**ARDUINO PROCEDURE:**

1. Open the software.
2. Click [Ctrl +N] to open new sketch and Type the code.
3. Click Verify. 
4. This will compile the code and check for errors.
5. After compiling, click file and select preferences.
6. Check the box “Compile” and click OK.
7. Compile the code again to generate Hex File.
8. The location of the hex file can be found in the ouput black box in the formal of .hex.

**PROTEUS PROCEDURE:**

1. Add the Ultrasonic Sensor Library.
2. Open proteus by clicking run as administrator.
3. Open new project and enter the file name.
4. Click next, next, next and finish.
5. Click P symbol and search keyword and place the required components
6. The components required are:

* Arduino UNO R3 V1.0
* Ultrasonic Sensor.
* Resistor
* Input
* Generator -DC
* Virtual Terminal
* Using terminal select ground.

1. Connect pin number 1 of Arduino RXD of Virtual Terminal
2. Connect +5V of Ultrasonic Sensor to input.
3. Connect test pin to resistor and connect the resistor to ground.
4. Connect the DC to resistor.
5. Connect GND pin of Ultrasonic Sensor to Ground
6. Connect Pin 9 of Arduino to trigger pin of Sensor.
7. Connect Pin 8 of Arduino to echo pin of Sensor.
8. Set the clock frequency of Arduino UNO as 10MHz.
9. Give input to Arduino UNO as HEX file.
10. Start the simulation process.

**PROGRAM**

long duracion;

long distancia;

int echo=8;

int trig=9;

void setup()

{

  Serial.begin(9600);

  pinMode(trig,OUTPUT);

  pinMode(echo,INPUT);

}

void loop()

{

  digitalWrite(trig,LOW);

  delayMicroseconds(4);

  digitalWrite(trig,HIGH);

  delayMicroseconds(10);

  digitalWrite(trig,LOW);

  duracion=pulseIn(echo,HIGH);

  distancia=duracion/58.4;

  Serial.print("Distancia:");

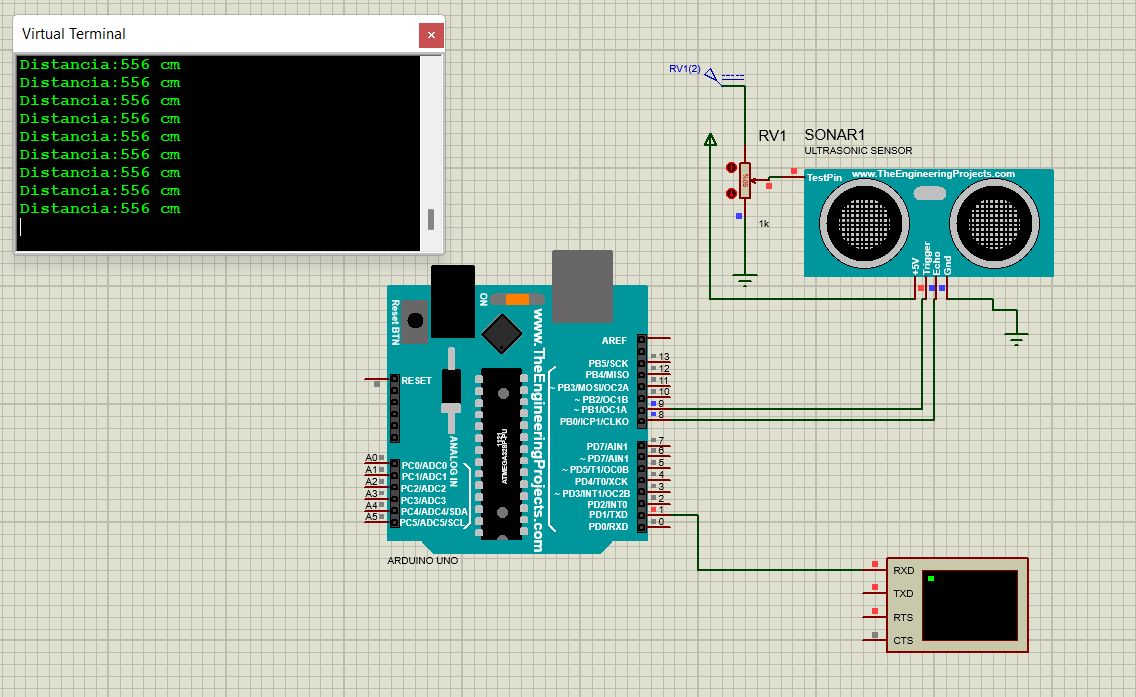
  Serial.print(distancia);

  Serial.println("cm");

  delay(100);

}

**CIRCUIT DIAGRAM:**



**RESULT**

Thus the program has been successfully verified and executed.