Objective:

In this experiment, the main objective is to learn how to write an assembly program for a blink LED program in a micro controller.

Apparatus:

1) Arduino Uno
2) Arduino IDE
3) one Led
4) one 220 ohm resistor
5) pc having intel microprocessor

Theory:

In this experiment the main objective is to understand the working principle of MTS-86c and MDA 8086 and familiarize emulator EMU 8086 by using a simple program to test its different uses and introduction to segmented memory technology used by Microprocessor 8086.

Procedure:

1) First we have to create led.ino and led.s files using code given above.

2) Then we need to create a folder named led and place the above two files in the led folder.

3) After that we open led.ino using Arduino IDE

4) Then we compile the program and upload it to be the hardware.

Source code:

.ino file:

```
//-----------------------------------
// C Code: RGB LED ON/OFF via Buttons
//-----------------------------------
extern "C"
{
void start();
void btnLED();
}
//----------------------
void setup()
{
start();
}
//----------------------
void loop()
{
btnLED();
}
```

.S file:

```
;-----------------------------------------
; Assembly Code: RGB LED ON/OFF via Buttons
;-----------------------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----------------------
.global start
.global btnLED
;=============================================================================
=======
start:
SBI DDRB, 4 ;set PB4 (pin D12 as o/p - red LED)
SBI DDRB, 3 ;set PB3 (pin D11 as o/p - green LED)
SBI DDRB, 2 ;set PB2 (pin D10 as o/p - blue LED)
CBI DDRD, 2 ;clear PD2 (pin D02 as i/p - red button)
CBI DDRD, 3 ;clear PD3 (pin D03 as i/p - green button)
```

Simulation:

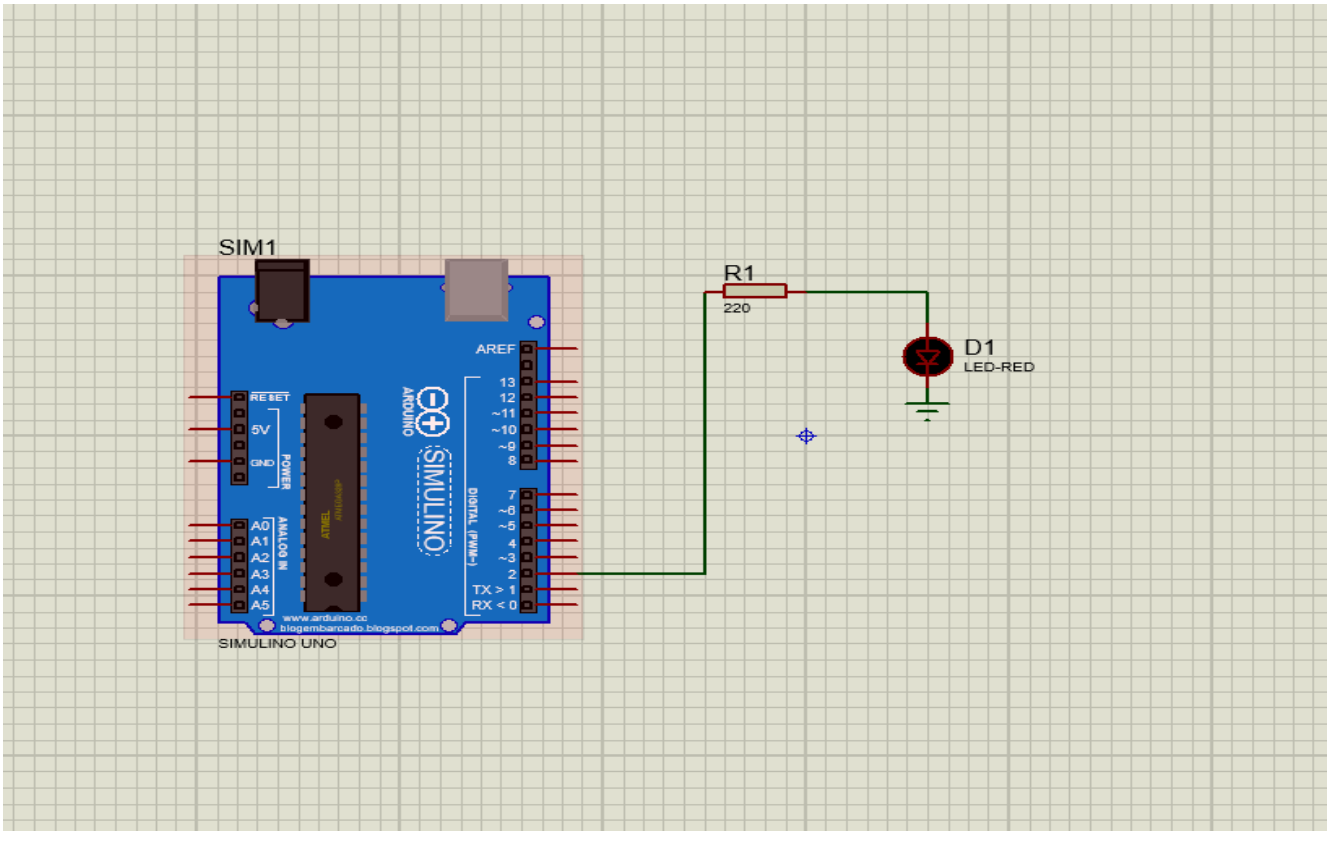

Fig: LED is in off state

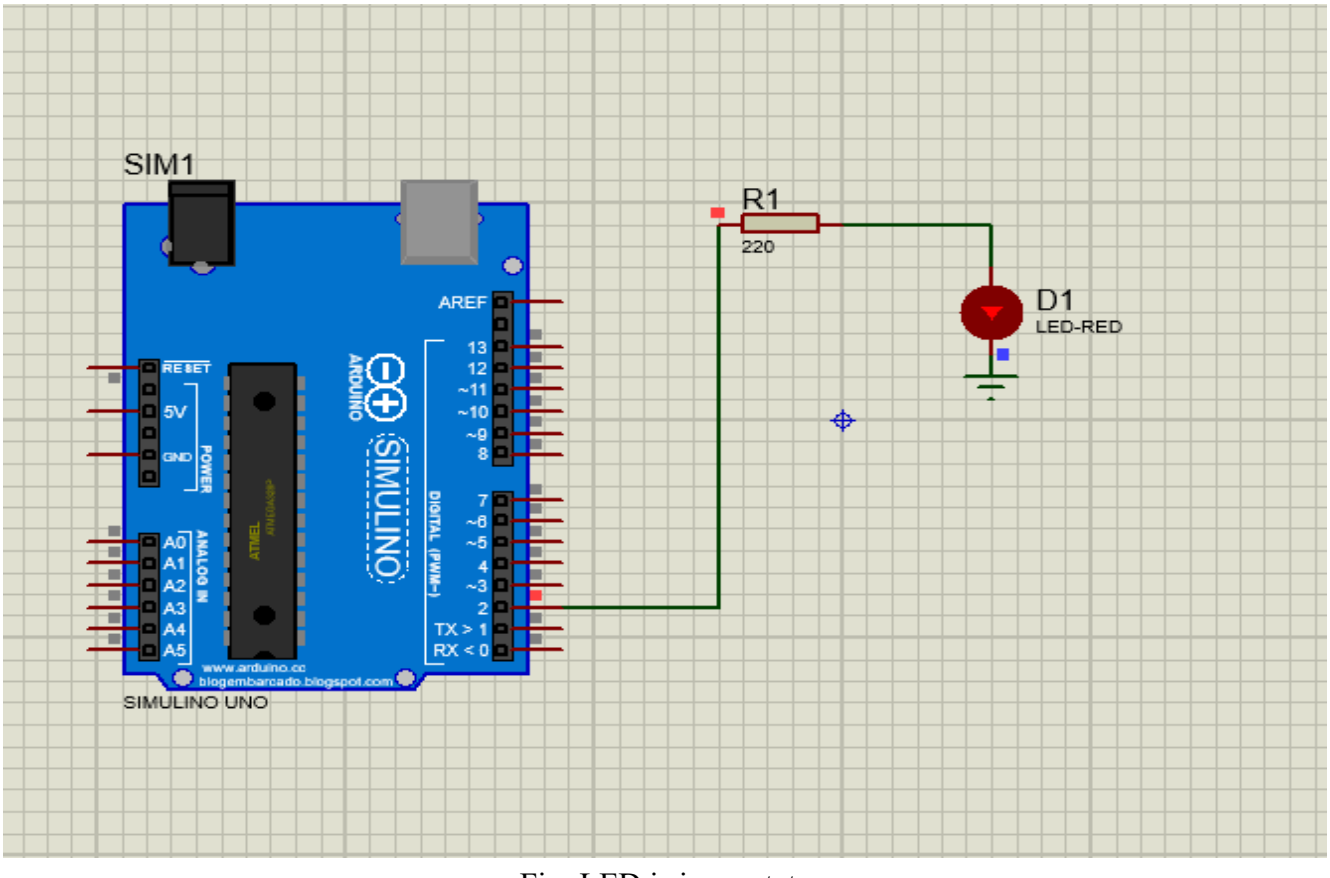

Fig: LED is in on state

Hardware Implementation:

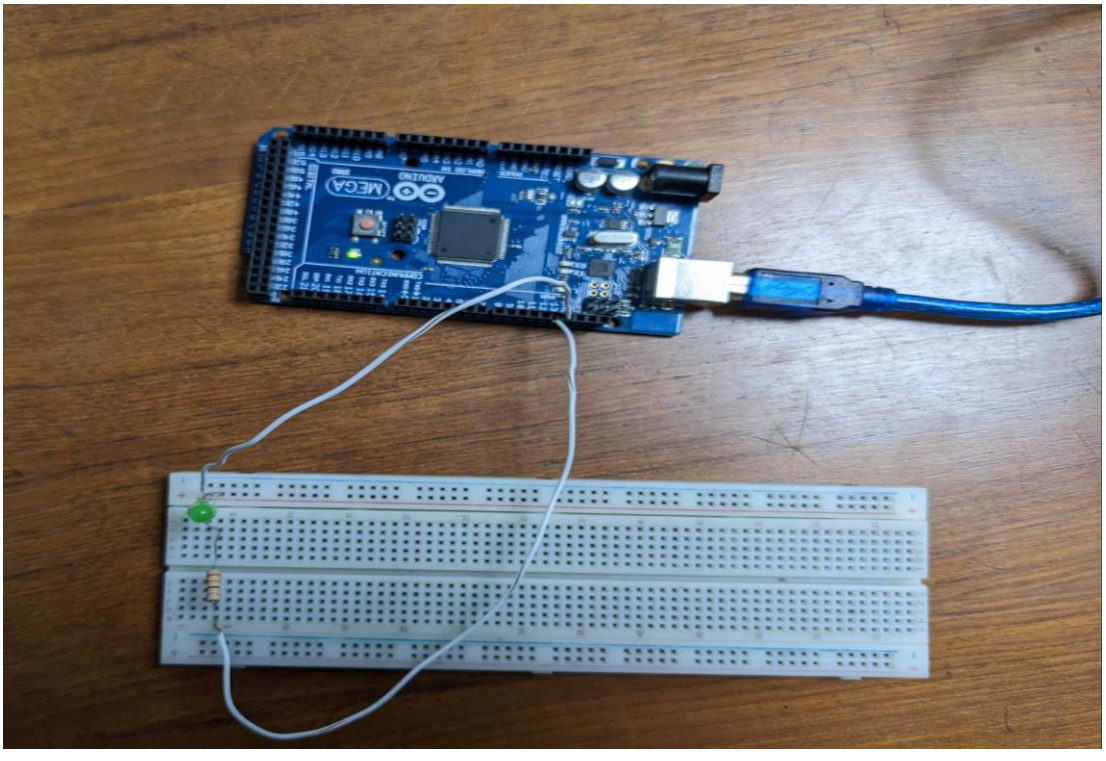

Fig: LED is in off state

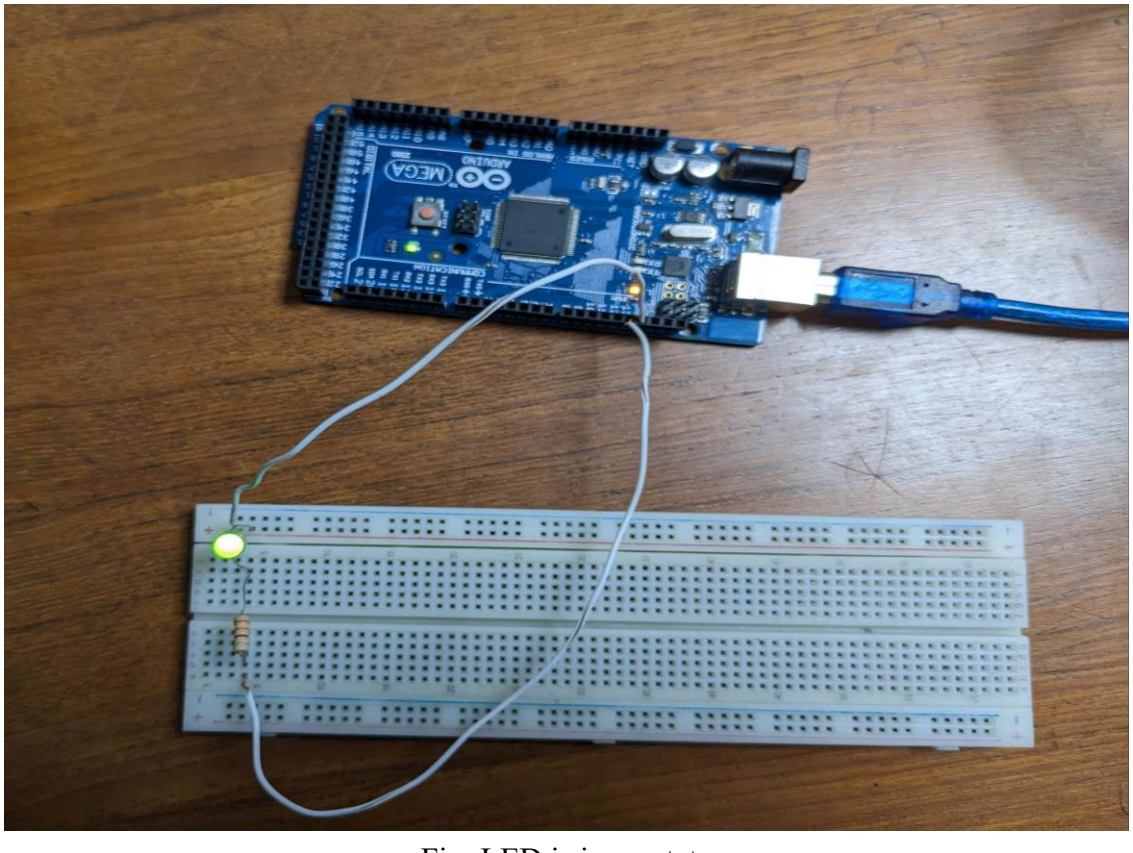

Fig: LED is in on state

Report:

All codes, scripts and proteus simulation of the blink program and traffic light system is attached above.

Discussion:

We will learn about the 8086 microprocessor with this activity. It is possible to observe how the 8086 chip exchanges values and performs additions and subtraction. Specific instructions and arrays can be seen in this laboratory activity. Now that we have a better understanding of the 8086 microprocessor, we can look at more variations of it. Typing mistakes made when entering the codes into the MTS-86c or on the computers are the only thing that could go wrong in this experiment. The success of the experiment depended heavily on the ability to create and decipher 8086 microprocessor code.

Conclusion:

It in crucial to comprehend the rationale for employing each function and the values it in related with before beginning to write any program. It in crucial to comprehend every line of code in order to be able to use the common functions discovered via this experiment to complete a task or solve an issue. A low-level programming language called an assembly language in designed to interface directly with a computer's hardware. Assembly languages are intended to be readable by humans, in contrast to machine language, which uses binary and hexadecimal characters.