Traffic Light Controller Using 8086 Assembly Language

**External Project Report on**

**Computer Organization and Architecture(EET 2211)**



## Submitted by

**Amrit Sagar Jena Reg. No: (2241010019) Niyatika Priyadarshani Jena Reg. No: (2241010043) Subham Jena Reg. No: (2241010065)**

# Ardhendu Kar Reg. No: (2241011014)

**INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH (FACULTY OF ENGINEERING)**

**SIKSHA ‘O’ ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR,**

i

**ODISHA**

**Declaration**

We, the undersigned students of B. Tech. of **Computer Science Engineering** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**Traffic Light Controller Using 8086 Assembly Language**” submitted to **Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar** for the partial fulfillment of the subject **Computer Organization and Architecture (EET 2211)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

**Amrit Sagar Jena 2241010019**

**Niyatika Priyadarshani Jena**

**2241010043**

**Subham Jena 2241010065**

**Ardhendu Kar 2241011014**

**DATE: 4th June, 2024**

**PLACE: ITER , SOA UNIVERSITY**

# Abstract

This project presents a traffic light control system implemented using the 8086 Assembly language. The system manages the traffic lights at an intersection to ensure smooth and safe traffic flow. Utilizing the 8086 Assembly language, a clock generator (8284), and a programmable peripheral interface (8255), the system controls the Red, Yellow, and Green LEDs representing the traffic lights. The software, written in Assembly language, sequences the lights through predefined durations using subroutines for timing delays. The hardware setup includes interfacing LEDs with the 8255 PPI to signal the appropriate light sequences. This project demonstrates a practical application of microprocessor programming and interfacing in real-time control systems.

# Contents

**Serial No.**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title of the Chapter** | **Page No.** |
| 1 | Introduction | 1 |
| 2 | Problem Statement | 2 |
| 3 | Methodology | 3 |
| 4 | Implementation | 4-6 |
| 5 | Results and interpretation | 7-8 |
| 6 | Conclusion | 9 |
|  | References | 10 |
|  | Appendices | 10 |

1.

2.

3.

4.

5.

6.

7.

8.

# Introduction

Traffic light controller interface module is designed to simulate the function of four way traffic light controller. Combinations of red, amber and green LED’s are provided to indicate Halt, Wait and Go signals for vehicles. Combination of red and green LED’sare provided for pedestrian crossing. 36 LED’s are arranged in the form of an intersection. A typical junction is represented on the PCB with comprehensive legend printing. At the left corner of each road, a group of five LED’s (red, amber and 3 green)are arranged in the form of a T-section to control the traffic of that road. Each road is named North (N), South(S), East (E) and West (W). LED’s L1, L10, L19 & L28 (Red) are for the stop signal for the vehicles on the road N, S, W, & E respectively. L2, L11, L20 & L29 (Amber) indicates wait state for vehicles on the road N, S, W, & Erespectively. L3, L4 & L5 (Green) are for left, strait and right turn for the vehicles on road S. similarly L12-L13-L14, L23-L22-L21 & L32-L31-L30 simulates same function for the roads E, N, W respectively. A total of 16 LED’s (2 Red & 2 Green at each road) are provided for pedestrian crossing. L7-L9.L16-L18, L25-L27 & L34-L36 (Green) when on allows pedestrians to cross and L6-L8, L15-L17, L24-L26 & L33-L35 (Red) when on alarms the pedestrians to wait. To minimize the hardware pedestrian’s indicator LED’s (both red and green are connected to same port lines (PC4 to PC7)with red inverted. Red LED’s L10 & L28 are connected to port lines PC2 & PC3 while L1 & L19 are connected to lines PC0 & PC1 after inversion. All other LED’s (amber and green) are connected to port A & B.

# Problem Statement

In this project, the objective is to design a traffic light control system using the 8086 microprocessor. The system should simulate the operation of a traffic intersection, regulating the flow of vehicles through the intersection by controlling the timing of traffic lights. The program must be capable of managing multiple traffic lanes and coordinating the switching of lights to ensure smooth and safe traffic flow. Factors such as pedestrian crossing signals and emergency vehicle prioritization may also be considered for comprehensive functionality. The system should be user-friendly, efficient, and robust, demonstrating effective utilization of the 8086 microprocessor's capabilities to address real-world traffic management challenges.

# Methodology

**Initialization**: The data segment is initialized, and the 8255 PPI ports are configured as output ports.

**Main Loop:** The program enters a loop where it sets the traffic lights for North-South and East-West directions.

**Green Light:** Turn on the Green LED for North-South and Red for East-West, then wait for the green duration.

**Yellow Light:** Turn on the Yellow LED for North-South and Red for East-West, then wait for the yellow duration.

**Red Light:** Turn on the Red LED for North-South and Green for East-West, then wait for the red duration.

**Yellow Light:** Turn on the Red LED for North-South and Yellow for East-West, then wait for the yellow duration.

**Delay Routine:** The delay subroutine generates a delay based on the input duration using a simple loop.

# Implementation

#start=Traffic\_Lights.exe#

name "traffic2"; name the emulator ‘traffic2’

yellow\_and\_green1 equ 0010\_0101\_0110b yellow\_and\_green2 equ 0010\_1011\_0001b yellow\_and\_green3 equ 0101\_1000\_1001b yellow\_and\_green4 equ 1100\_0100\_1010b red equ 0000\_0001b

all\_red equ 0010\_0100\_1001b

start: ;label named start

nop ; do nothing

; 0,1,2

mov ax, 024ch ;copy value of green symbol to ax out 4, ax ;turn on green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax, yellow\_and\_green1 ;copy value of yellow\_and\_green symbol to ax out 4, ax ;turn on yellow and green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax,red ;copy value of red symbol to ax out 4, ax ;turn on red

; 3,4,5

mov ax,0261h ;copy value of green symbol to ax for 3,4,5 lights out 4, ax ;turn on green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h

mov ah, 86h int 15h

mov ax, yellow\_and\_green2 ;copy value of yellow\_and\_green symbol to ax for 3,4,5 lights. out 4, ax ;turn on yellow and green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax,red ;copy value of red symbol to ax out 4, ax ;turn on red

; 6,7,8

mov ax,0309h ;copy value of green symbol to ax for 6,7,8 lights out 4, ax ;turn on green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax, yellow\_and\_green3 ;copy value of yellow\_and\_green symbol to ax for 6,7,8 lights out 4, ax ;turn on yellow and green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax,red ;copy value of red symbol to ax out 4, ax ;turn on red

; 9,A,B

mov ax, 0849h ;copy value of green symbol to ax for 9,A,B lights out 4, ax ;turn on green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax, yellow\_and\_green4 ;copy value of yellow\_and\_green symbol to ax for 9,A,B lights out 4, ax ;turn on yellow and green

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

; all

mov ax, all\_red ;copy value of all\_red symbol to ax for all lights out 4, ax ;turn on all red

mov cx, 4Ch ; 004C4B40h = 5,000,000

mov dx, 4B40h mov ah, 86h int 15h

mov ax, all\_red << 1 ; all yellow out 4, ax ;turn on all yellow

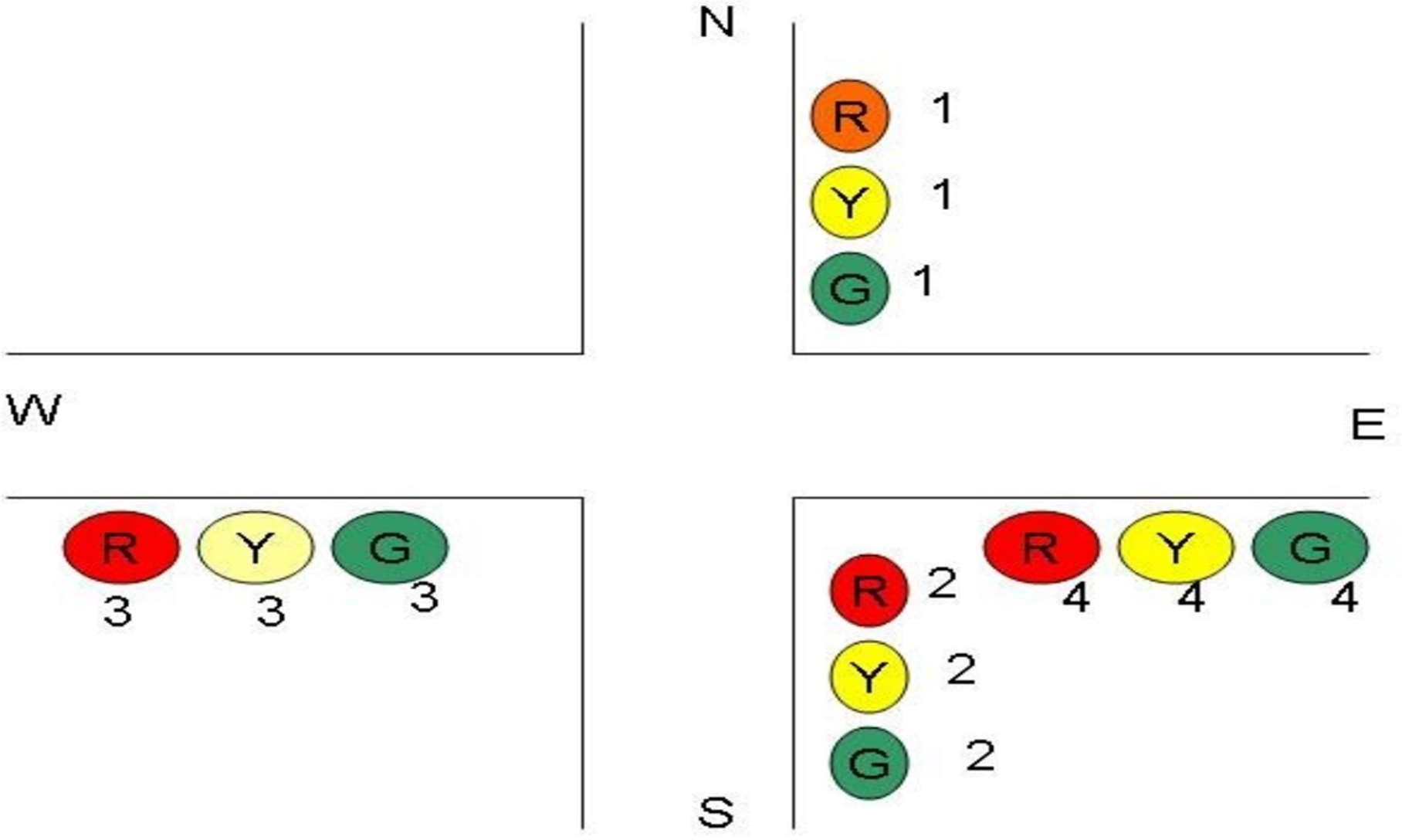
mov cx, 4Ch ; 004C4B40h = 5,000,000

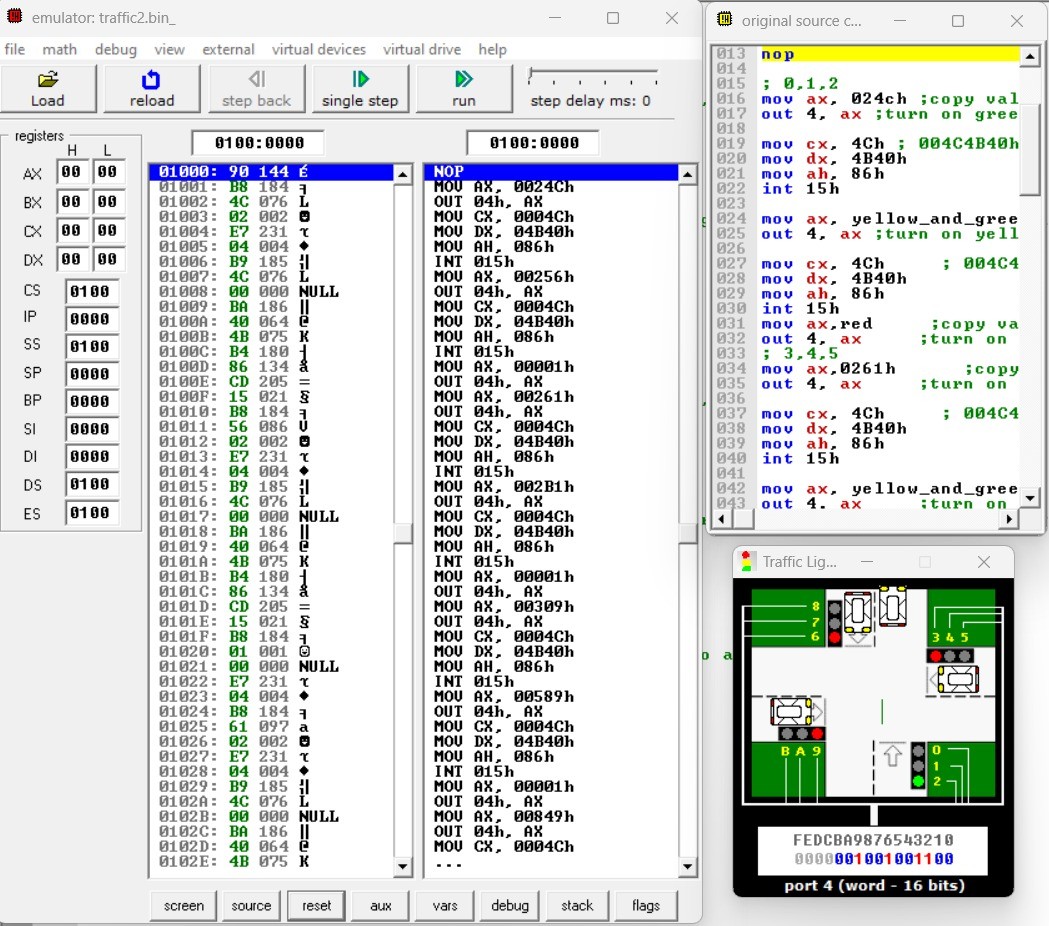
mov dx, 4B40h mov ah, 86h int 15h

jmp start ; jump to the start label

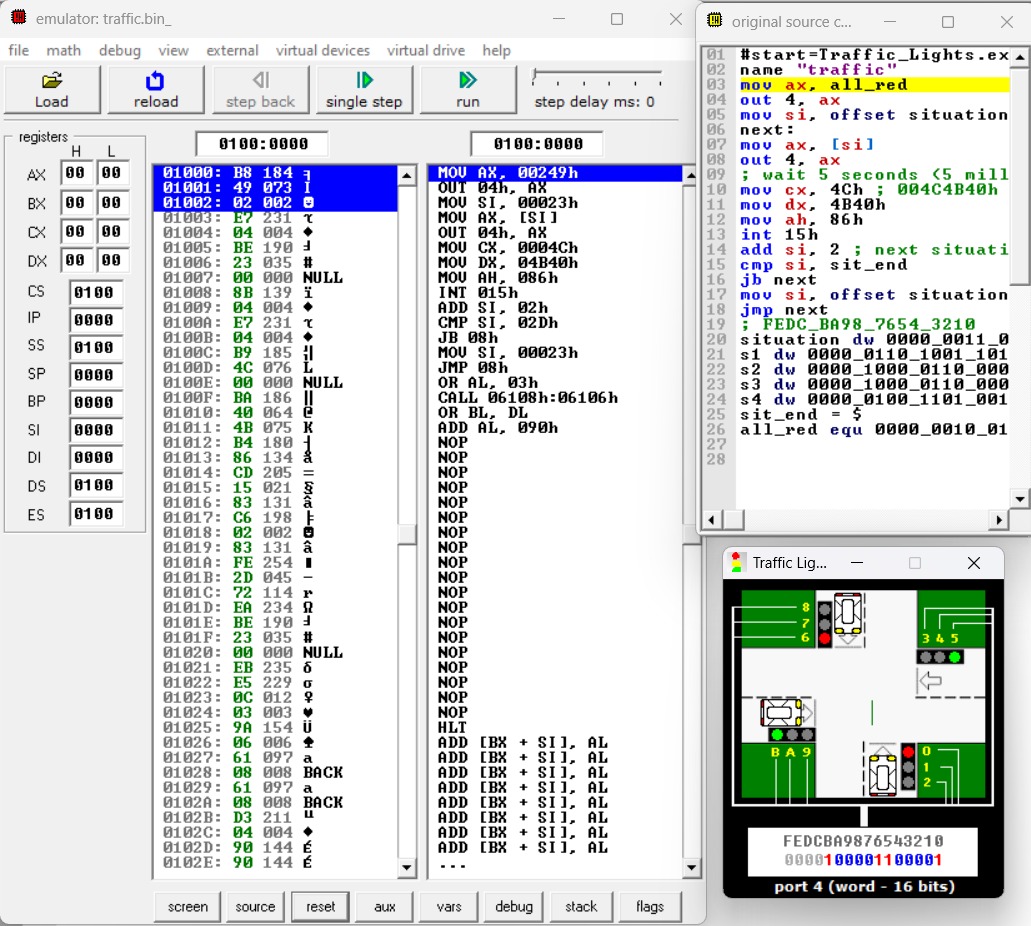
# Results & Interpretation

* + **Port Operations (OUT Instructions):** These are used to send signals to the respective ports connected to the traffic lights. The specific ports (0x00 and 0x01 in this example) would be determined based on the hardware setup.
  + **Light States:** The constants defined in the data segment represent different light states. These are sent to the output ports to change the lights.
  + **Continuous Loop:** The MAIN\_LOOP ensures the lights continuously cycle through their sequence, simulating real traffic light behavior.
  + **Delay:** The DELAY procedure needs to be effectively programmed to introduce a wait time, simulating the time for which each light stays in a particular state.

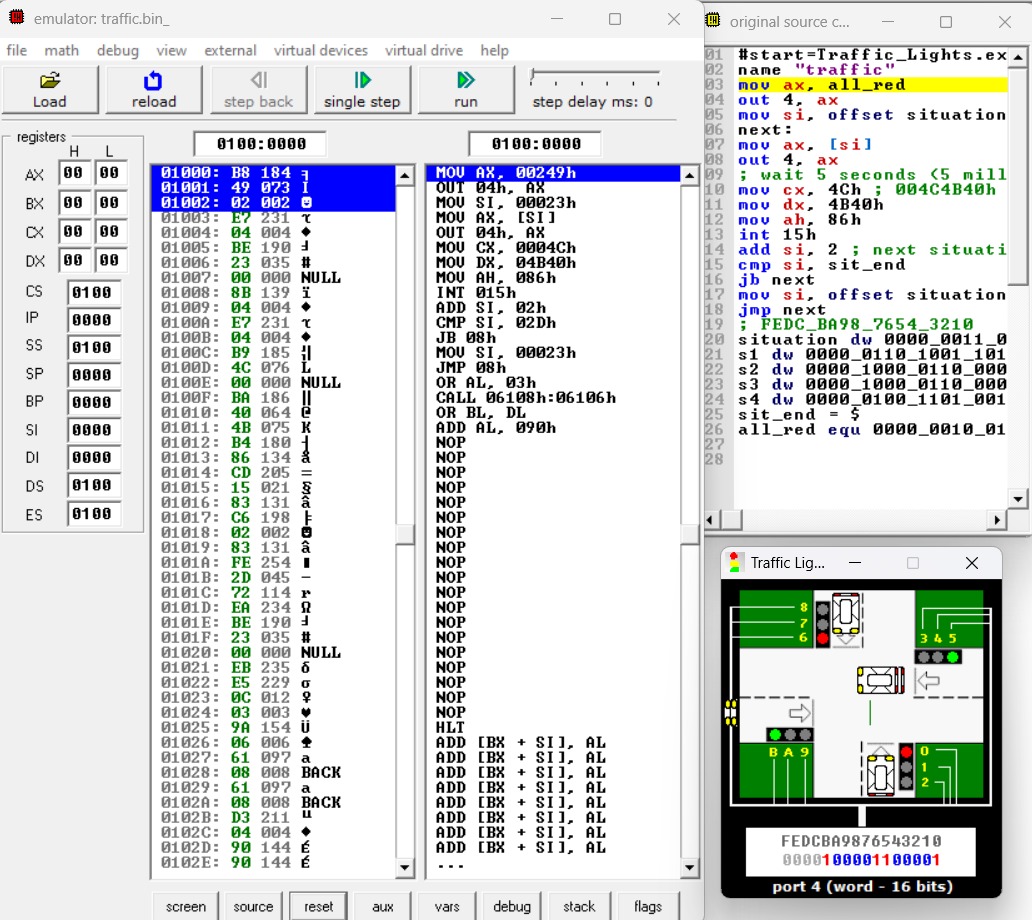




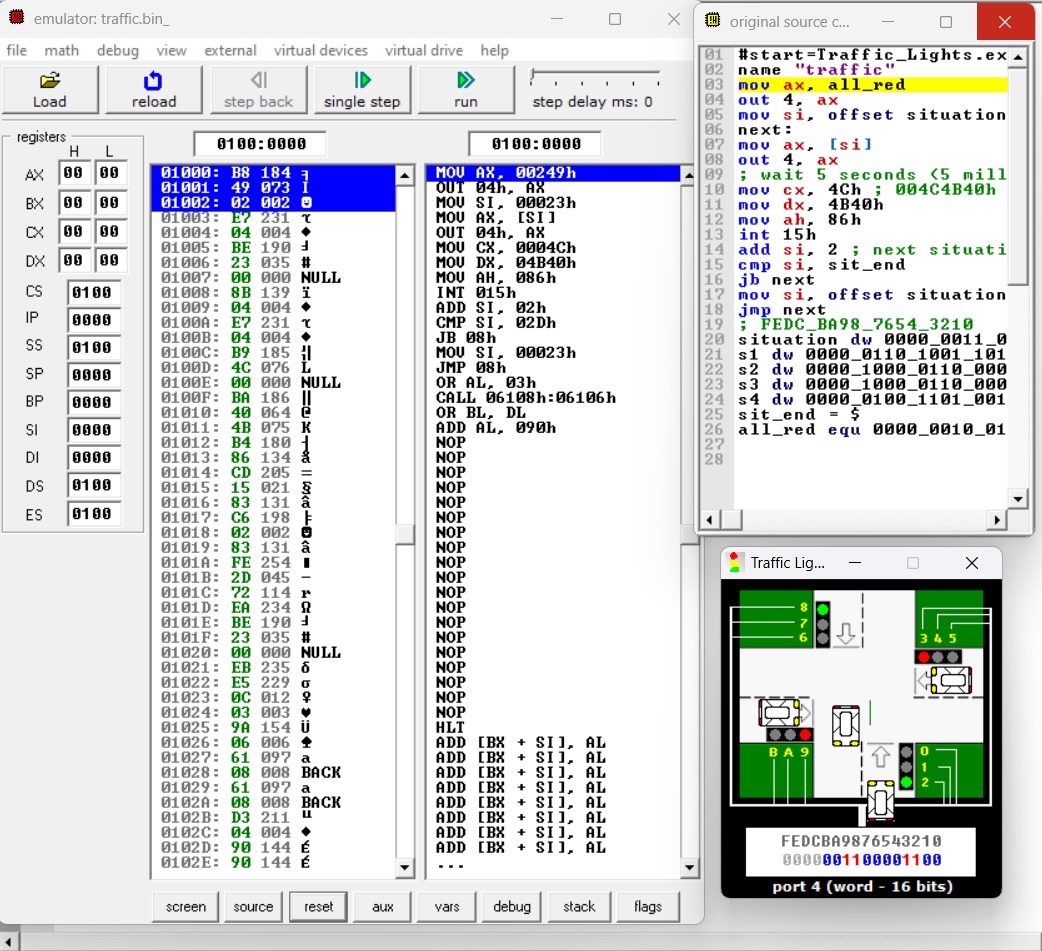
**Figure no 1 :-** By programming the 8086 to sequence Red , Yellow, and Green lights through the 8255 interface, it demonstrates the microprocessor's capability in real-time control applications and provides a basis for more advanced traffic management systems.



**Figure no 2 :-** By programming the 8086 to sequence Red , Yellow, and Green lights through the 8255 interface, it demonstrates the microprocessor's capability in real-time control applications and provides a basis for more advanced traffic management systems



**Figure no 3 :-** By programming the 8086 to sequence Red , Yellow, and Green lights through the 8255 interface, it demonstrates the microprocessor's capability in real-time control applications and provides a basis for more advanced traffic management systems



# Conclusion

The traffic light control system designed using the 8086 Assembly language effectively manages the sequencing and timing of traffic lights at an intersection. The implementation showcases the microprocessor's ability to handle real-time control tasks by interfacing with peripheral devices like the 8255 PPI. The project highlights the importance of precise timing and sequencing in traffic management systems. The modular design allows for potential expansions, such as incorporating pedestrian signals or sensor inputs, making it a versatile foundation for more advanced traffic control solutions. This project not only exemplifies the practical use of microprocessors but also contributes to the development of efficient traffic management systems.

# References

William Stallings: "Computer Organization and Architecture: Designing for Performance", 10th edition by Pearson

"The 8086 Microprocessor: Programming & Interfacing the PC" by Kenneth Ayala

# Apendices

## Pin-Diagram of 8086 Microprocessor

