WIX1003 Computer Systems and Organization

Tutorial: Occ 9

Group 1

Lab Assignment – Report

Cross Junction Traffic Light Control System

Group Member:

1) DENNIS AIMIN OON BIN JEFFREY OON
   22001610
2) AHMAD AMIRUL HAKIMI BIN MOHAMAD
   22001840

Instructor: PROFESOR MADYA DR. TEY KOK SOON

Table of Content

01: Introduction

This lab assignment is part of the coursework requirement for the course WIX 1003 Computer Systems and Organization. This assignment is mini project where we worked in a group to solve a given problem. The details of the problem can be found in the following sections. At the end of the project, we will present our solution to a panel of assessors.

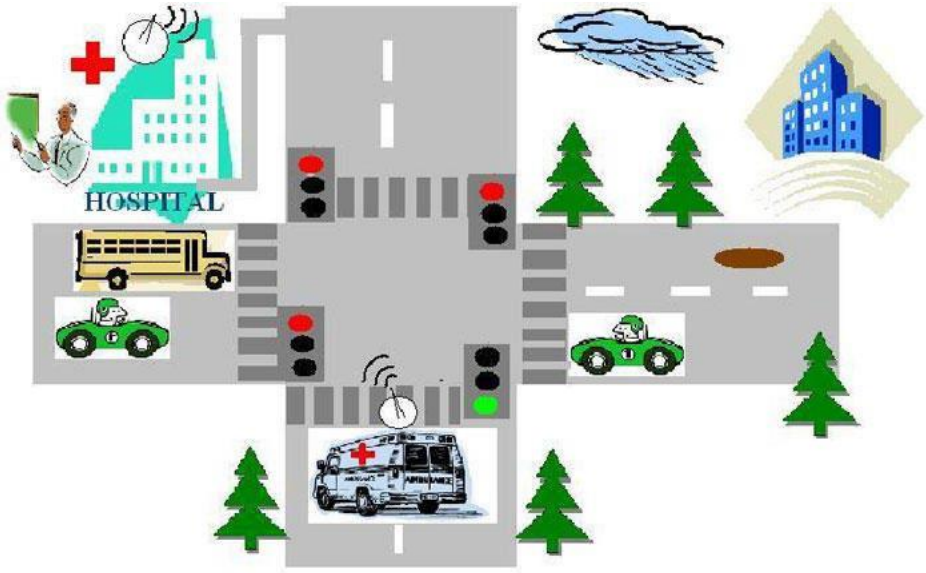
02: Objective

The objective of this lab assignment are as follows:
- To stimulate creativity and innovation among us
- To allow us to have better understanding of the studied subject

03: Background and Requirement

Status of Encounter

My team is assigned with the task to develop a prototype of new computer system that capable to control a cross-junction traffic. My team is required to product a working demo of a traffic light control system for this cross junction. The cross junction and traffic flow that are required to be controlled are given in the figure below.



Minimal Requirement

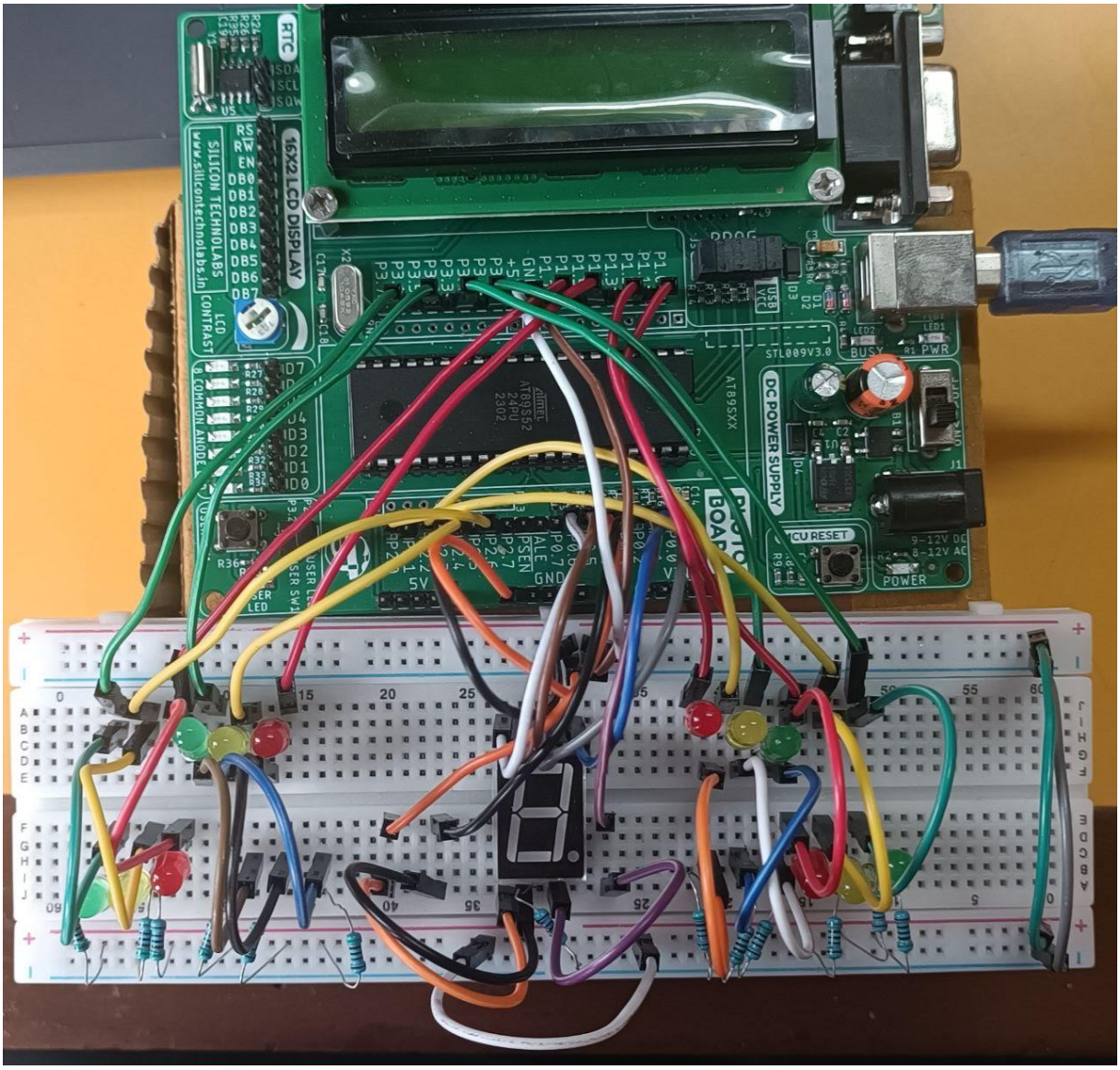
A microprocessor or microcontroller system is required to be used for this system. The system will allow for equal chance for users to exit and cross the junction. Each of the junction is to be controlled by a tri-colour traffic light indicator (green, yellow, red). The controller system will control the lighting of this tri-colour light indicator granting each junction the permission to exit and cross the junction without causing any accident or havoc. The indication of each of the colour is as follows:

Green : Permission to go **Yellow** : Prepare to stop Red : Stop and wait

Additional Feature Suggestion

A count down indicator will be helpful for user to know how long they will need to wait before permission to go is granted.

04: Component Connection Diagram



P0 = LED DISPAY (TIMER)

P1 = RED LIGHT

P2 = YELLOW LIGH

P3 = GREEN LIGHT

05: Complete Code of the System

```
; WIX1003 LAB ASSIGNMENT

; TRAFFIC LIGHT CONTROL SYSTEM

; P1 = LED DISPLAY (TIMER)

; P1 = RED LIGHT

; P2 = YELLOW LIGHT

; P3 = GREEN LIGHT


        ORG   00H

        AJMP  MAIN


MAIN:

        MOV   A, #0FFH      ; SET P3 IN OFF CONDITION

        MOV   P3, A

        MOV   A, #0FFH      ; SET P2 IN OFF CONDITION

        MOV   P2, A

        MOV   A, #0AAH      ; SET P1 IN ON CONDITION (SPECIFIC TO BIT 0,2,4,6)

        MOV   P1, A

        MOV   A, #00H       ; SET P0 IN OFF CONDITION

        MOV   P0, A

        ACALL         DELAYB      ; DELAY 3 SECONDS
```

```
START:

; 1ST GREEN LIGHTS UP

        MOV   P1, #0ABH

        MOV   P3, #0FEH

        ACALL         DELAYA       ; DELAY 9 SECONDS


; 1ST YELLOW LIGHTS UP

        MOV   P3, #0FFH

        MOV   P2, #0FEH

        ACALL         DELAYB       ; DELAY 3 SECONDS


; 1ST RETURN TO RED LIGHT

        MOV   P2, #0FFH

        MOV   P1, #0AAH

        ACALL         DELAYB       ; DELAY 3 SECONDS


; 2ND GREEN LIGHTS UP

        MOV   P1, #0AEH

        MOV   P3, #0FBH

        ACALL         DELAYA       ; DELAY 9 SECONDS


; 2ND YELLOW LIGHTS UP
```

```
        MOV   P3, #0FFH
        MOV   P2, #0FBH
        ACALL         DELAYB       ; DELAY 3 SECONDS

; 2ND RETURN TO RED LIGHT
        MOV   P2, #0FFH
        MOV   P1, #0AAH
        ACALL         DELAYB       ; DELAY 3 SECONDS

; 3RD GREEN LIGHTS UP
        MOV   P1, #0BAH
        MOV   P3, #0EFH
        ACALL         DELAYA       ; DELAY 9 SECONDS

; 3RD YELLOW LIGHTS UP
        MOV   P3, #0FFH
        MOV   P2, #0EFH
        ACALL         DELAYB       ; DELAY 3 SECONDS

; 3RD RETURN TO RED LIGHT
        MOV   P2, #0FFH
        MOV   P1, #0AAH
```

```
        ACALL       DELAYB      ; DELAY 3 SECONDS


; 4TH GREEN LIGHTS UP
        MOV   P1, #0EAH
        MOV   P3, #0BFH
        ACALL       DELAYA      ; DELAY 9 SECONDS


; 4TH YELLOW LIGHTS UP
        MOV   P3, #0FFH
        MOV   P2, #0BFH
        ACALL       DELAYB      ; DELAY 3 SECONDS


; 4TH RETURN TO RED LIGHT
        MOV   P2, #0FFH
        MOV   P1, #0AAH
        ACALL       DELAYB      ; DELAY 3 SECONDS
        AJMP  START         ; ABSOLUTE JUMP TO START


; DELAY SBUROUTINE
DELAY:
        MOV   R1, #009H
        MOV   R2, #007H
```

```
        MOV   R3, #005H
DELAY1:
        DJNZ  R3, DELAY1
        DJNZ  R2, DELAY1
        DJNZ  R1, DELAY1
        RET

; DELAY/COUNTDOWN 9 SECONDS
DELAYA:
        MOV   P0, #67H
        ACALL         DELAY
        MOV   P0, #7FH
        ACALL         DELAY
        MOV   P0, #7H
        ACALL         DELAY
        MOV   P0, #7DH
        ACALL         DELAY
        MOV   P0, #6DH
        ACALL         DELAY
        MOV   P0, #66H
        ACALL         DELAY
        MOV   P0, #4FH
```

```
        ACALL   DELAY
        MOV   P0, #5BH
        ACALL   DELAY
        MOV   P0, #6H
        ACALL   DELAY
        RET

; DELAY/COUNTDOWN 3 SECONDS
DELAYB:
        MOV   P0, #4FH
        ACALL   DELAY
        MOV   P0, #5BH
        ACALL   DELAY
        MOV   P0, #6H
        ACALL   DELAY
        RET

        END
```

06: Explanation on Operation of the System

Documentation:

- Set port 0 as LED display (timer).
- Set port 1 as red light.
- Set port 2 as yellow light.
- Set port 3 as green light.

Initialization:

- Set up the initial program execution.
- The ORG 00H directive specifies the starting address.
- The AJMP MAIN instruction performs a jump to the MAIN label, indicating where the main part of the program begins.

MAIN:

- Initially, all the yellow and green lights are turned off and all the red lights are turned on.
- Assigned #0AAH to P0 because I want to turn on only bit 0,2,4 and 6. By doing this way, I can avoid any misconnection of the bit.
- Add on a short delay of 3 seconds for more smoothing flow of the program.

START:

First Green Light

- The first red light turns off.
- The first green light turns on.
- The program delays for 9 seconds by calling the DELAYA subroutine.

First Yellow Light

- It's a transition to yellow light.
- The first green light turns off.
- The first yellow light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure traffic user have enough time to slow down.

First Red Light

- It will light on back the red light.
- The first yellow light turns off.
- The first red light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure all traffic user stopped.

Second Green Light

- The second red light turns off.
- The second green light turns on.
- The program delays for 9 seconds by calling the DELAYA subroutine.

Second Yellow Light

- It's a transition to yellow light.
- The second green light turns off.
- The second yellow light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure traffic user have enough time to slow down.

Second Red Light

- It will light on back the red light.
- The second yellow light turns off.
- The second red light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure all traffic user stopped.

Third Green Light

- The third red light turns off.
- The third green light turns on.
- The program delays for 9 seconds by calling the DELAYA subroutine.

Third Yellow Light

- It's a transition to yellow light.
- The third green light turns off.
- The third yellow light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure traffic user have enough time to slow down.

Third Red Light

- It will light on back the red light.
- The third yellow light turns off.
- The third red light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure all traffic user stopped.

Fourth Green Light

- The fourth red light turns off.
- The fourth green light turns on.
- The program delays for 9 seconds by calling the DELAYA subroutine.

Fourth Yellow Light

- It's a transition to yellow light.
- The fourth green light turns off.
- The fourth yellow light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure traffic user have enough time to slow down.

Fourth Red Light

- It will light on back the red light.
- The fourth yellow light turns off.
- The fourth red light turns on.
- The program delays for 3 seconds by calling the DELAYB subroutine.
- The main reason we decided to add on a delay for 3 seconds is to make sure all traffic user stopped.
- The program jumps back to the START label, repeating the sequence.

DELAY:

- Introduces a delay by executing nested decrement loops. The specific duration of the delay depends on the initial values assigned to R1, R2, and R3.
- The first instruction MOV R1, #009H moves the immediate value 009H into register R1. This sets the initial value of R1 to 009H.
- The second instruction MOV R2, #007H moves the immediate value 007H into register R2. This sets the initial value of R2 to 007H.
- The third instruction MOV R3, #005H moves the immediate value 005H into register R3. This sets the initial value of R3 to 005H.
- The code then enters a loop labeled as DELAY1.
- Used in DELAYA and DELAYB.

DELAY1:

- The DJNZ instruction stands for "Decrement and Jump if Not Zero." It decrements the value of the specified register and jumps to the specified label if the result is not zero.
- The first DJNZ R3, DELAY1 instruction decrements the value in R3 and checks if it is zero. If R3 is not zero, it jumps back to the DELAY1 label, repeating the loop.
- The second DJNZ R2, DELAY1 instruction does the same with R2.
- The third DJNZ R1, DELAY1 instruction does the same with R1.
- This creates nested loops where R3 is decremented first, followed by R2, and then R1. The loops continue until all three registers reach zero.
- Once the loops are completed, the program execution returns to the point where the subroutine was called.
- Used in DELAYA and DELAYB.

DELAYA (Countdown 9 seconds):

- The DELAYA subroutine introduces a delay or countdown of approximately 9 seconds and display it in LED Display (Timer).
- Used in green light turns on.

DELAYB (Countdown 3 seconds):

- The DELAYB subroutine introduces a delay or countdown of approximately 3 seconds and display it in LED Display (Timer).
- Used in transition to yellow light and red light.

The program continues to repeat the sequence indefinitely, controlling the traffic lights according to the specified timings.

07: Design Consideration and System Limitation

Design Considerations:

- Traffic Light Sequence:
  - The code implements a sequential control for a traffic light system. This is to ensuring that the sequence adheres to traffic regulations and safety standards.

- Timing Accuracy:
  - The delays are implemented using loops and the timing may vary. This is to ensuring accurate timing for each phase of the traffic light sequence.

- 7-segment LED Display and RED, YELLOW, GREEN LEDs:
  - P0 is used as an LED display for the timer. P1 is used as Red LEDs. P2 is used as Yellow LEDs. P3 is used as Green LEDs. This is to avoiding misconnections between them.

- Delay Subroutines:
  - The delay subroutines (DELAY, DELAYA, DELAYB) play a crucial role in achieving the desired timing for the traffic light sequence. This is to ensuring that these delays are sufficient and accurate for the intended application.

- Jump Instructions:
  - The use of jump instructions (AJMP) helps control the flow of the program. Verify that the jumps are correctly implemented to avoid unintended behavior.

- Code Efficiency:
  - Consider optimizing the code for efficiency and readability. This includes using descriptive labels and comments to enhance code understanding and maintenance.

System Limitations:

- Fixed Timing:
  - The code implements fixed delays for each phase of the traffic light sequence. This may limit flexibility in adjusting the timing dynamically based on external factors such as traffic conditions.

- Single-Threaded Execution:

  - The code is single-threaded and the traffic light sequence runs sequentially. For more complex systems, consideration may be needed for multi-threading or interrupt-based designs.

- Clock Dependency:
  - The timing of delays is dependent on the clock frequency of the microcontroller. If the clock frequency changes, the timing of the traffic light sequence may also change.

- No Fault Tolerance:
  - The code does not incorporate fault tolerance mechanisms. In a real-world scenario, the system may need to handle faults or unexpected events, such as sensor malfunctions or power interruptions.

- Limited State Control:
  - The code defines a fixed sequence of traffic light states. Additional features, such as pedestrian crossing signals or adapting to varying traffic loads, would require further enhancements.

08: Recorded presentation

Explain the complete code of your system

https://365umedumy-my.sharepoint.com/:v:/g/personal/22001610_siswa365_um_edu_my/ESK62OHD579IrTl33uvPjW8BxSWYA2-5pQKQDOxCqyBQKA?e=59965B

Explain the circuit and demonstration of the working system

WIX1003_LAB ASSIGNMENT_Explain the circuit and demonstration of tHe working system.mp4

Design consideration and system limitation

https://365umedumy-my.sharepoint.com/:v:/g/personal/22001610_siswa365_um_edu_my/ERq7Hatj_S1Ln8lAkkA4VmYBhkosadMlAhd9v8dfqEZiXQ?e=KHU4lW

09: Conclusion

In conclusion, while the project provides a foundation for understanding a simple implementation traffic light control using assembly language for the AT89S52 microcontroller, further refinement and expansion are necessary to meet the demands of real-world traffic management systems. Consideration for fault tolerance, adaptability and advanced features would be crucial for a more comprehensive and practical implementation.