BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 (July 2023)

Microprocessor and Embedded Systems Laboratory

Final Project Report

Section: C2 Group: 03

A 415 Art

Course Instructors:

Dr. Sajid Muhaimin Choudhury, Assistant Professor
Ayan Biswas Pranta, Lecturer (Part time)

Signature of Instructor: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Kaniz Fatema Zarin** **Student ID: 1906172**	**Signature:** _____ _____ **Full Name: Abrar Jahin Niloy** **Student ID: 1906180**
Signature: _____ **Full Name: Md. Mahir Ashif** **Student ID: 1906182**	**Signature:** _____ **Full Name: Zahin Tazwar** **Student ID:1906183**

Table of Contents

1 Abstract

A single-cycle processor completes the execution of each instruction within a single clock cycle, simplifying its design. While this architecture is straightforward, it may result in longer clock cycles and is often employed in educational settings and simpler embedded systems. It is less suitable for high-performance computing due to potential inefficiencies. So illustrating a single cycle processor will improve the knowledge of student.

2 Introduction

Now a day almost every electrical machines use microchip in their control unit. It is imperative for Electrical engineering students to understand basic processor workflow.

But the workflow is complex to understand. So illustrative approach is better for a student to comprehend single cycle processor workflow.

3 Design

3.1 Problem Formulation (PO(b))

3.1.1 Identification of Scope

A single cycle processor is a complex architecture to understand from a text book even with pictures. So we thought of a way to solve it.

3.1.2 Literature Review

A visual working cycle with indications will help the students a lot to understand single cycle processor easily.

3.1.3 Formulation of Problem

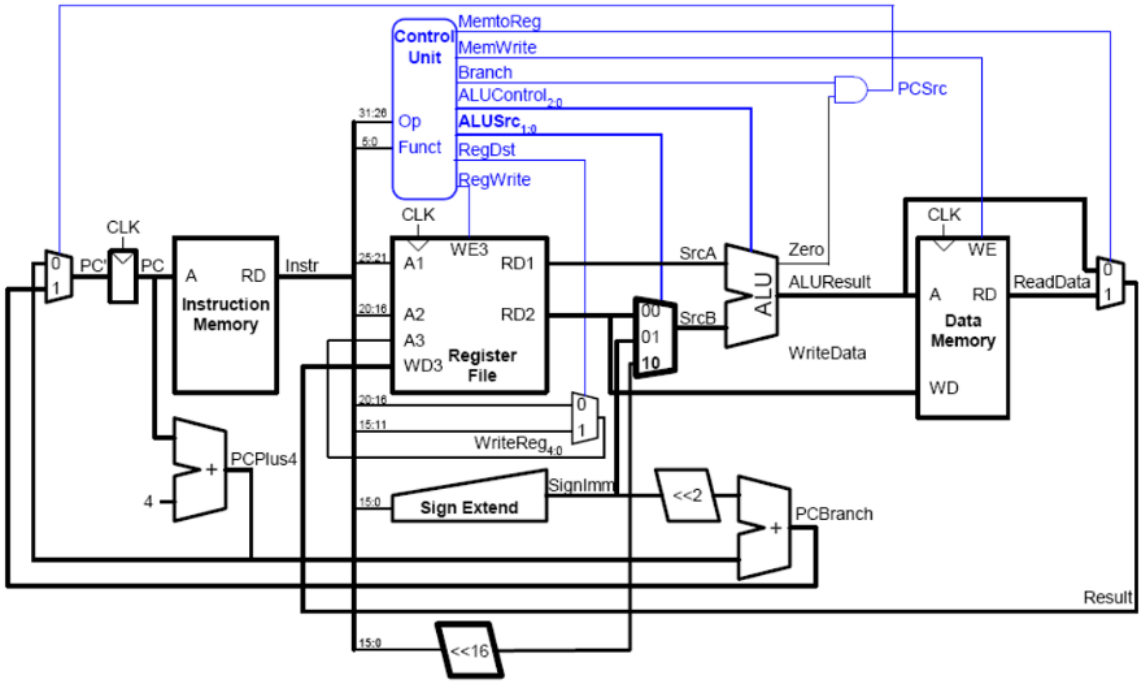
We thought of a LED matrix to demonstrate our project on.

3.1.4 Analysis

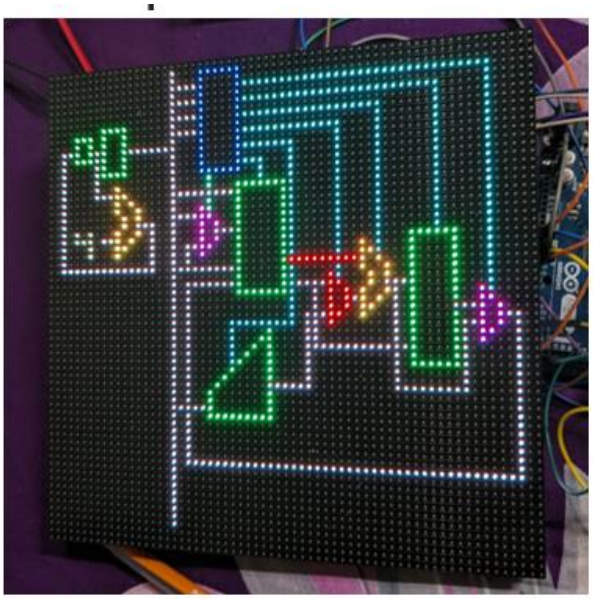
This project will be a good teaching equipment for undergraduate electrical engineering students.

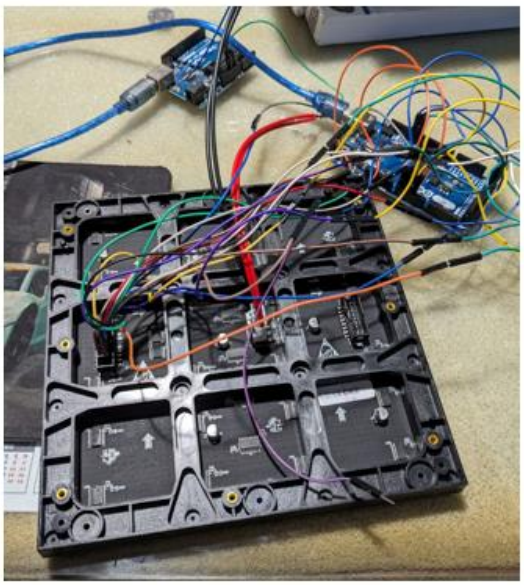
3.2 Design Method (PO(a)).

3.3 Circuit Diagram



3.4 CAD/Hardware Design





3.5 Full Source Code of Firmware

```
#include "RGBmatrixPanel.h"

#include "bit_bmp.h"
#include "fonts.h"
#include <string.h>
#include <stdlib.h>

// Most of the signal pins are configurable, but the
CLK pin has some
// special constraints.  On 8-bit AVR boards it must be
on PORTB...
// Pin 11 works on the Arduino Mega.  On 32-bit SAMD
boards it must be
// on the same PORT as the RGB data pins (D2-D7)...
// Pin 8 works on the Adafruit Metro M0 or Arduino
Zero,
// Pin A4 works on the Adafruit Metro M4 (if using the
Adafruit RGB
// Matrix Shield, cut trace between CLK pads and run a
wire to A4).

//#define CLK  8   // USE THIS ON ADAFRUIT METRO M0,
etc.
//#define CLK A4 // USE THIS ON METRO M4 (not M0)
#define CLK 11 // USE THIS ON ARDUINO MEGA
#define OE   9
#define LAT 10
#define A   A0
#define B   A1
#define C   A2
#define D   A3
#define E   A4

RGBmatrixPanel matrix(A, B, C, D, E, CLK, LAT, OE,
false, 64);
//Configure the serial port to use the standard printf
function
void setup()
{
  Serial.begin(115200);
  matrix.begin();

  //Control signal Block
  matrix.drawRect(22, 1, 6, 14, matrix.Color333(0, 0,
3.5));

  matrix.drawLine(18, 3, 21, 3, matrix.Color333(2.5,
2.5, 2.5));///cond
  matrix.drawLine(18, 5, 21, 5, matrix.Color333(2.5,
2.5, 2.5));//op
  matrix.drawLine(18, 7, 21, 7, matrix.Color333(2.5,
2.5, 2.5));//func
  matrix.drawLine(18, 9, 21, 9, matrix.Color333(2.5,
2.5, 2.5));//rd

  //Bus
 matrix.drawLine(18, 1, 18, 60, matrix.Color333(2.5,
2.5, 2.5));

//Control signal regwrite
  matrix.drawLine(28, 13, 30, 13,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(30, 13, 30, 15,
matrix.Color333(0,4.5,3.5));
//Control signal Path immsrc
  matrix.drawLine(28, 11, 34, 11,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(34, 11, 34, 34,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(34, 34, 26, 34,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(26, 34, 26, 39,
matrix.Color333(0,4.5,3.5));
matrix.drawLine(28, 9, 40, 9,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(40, 9, 40, 28,
matrix.Color333(0,4.5,3.5));

  //Control signal Path alu control
  matrix.drawLine(28, 7, 45, 7,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(45, 7, 45, 25,
matrix.Color333(0,4.5,3.5));

  //Control Signal path mem write
  matrix.drawLine(28, 5, 52, 5,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(52, 5, 52, 22,
matrix.Color333(0,4.5,3.5));

  // Control signal mem to reg
  matrix.drawLine(28, 3, 58, 3,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(58, 3, 58, 30,
matrix.Color333(0,4.5,3.5));

  //pc block
  matrix.drawRect(4, 9, 3, 4, matrix.Color333(0, 7,
0));

  //ins memory
  matrix.drawRect(8, 8, 4, 7, matrix.Color333(0, 7,
0));//ins memory

  //pc block to alu


  matrix.drawLine(7, 11, 7, 11, matrix.Color333(2.5,
2.5, 2.5)); // Pc block to IM
  matrix.drawLine(7, 11, 7, 17, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(7, 17, 10, 17, matrix.Color333(2.5,
2.5, 2.5));

  //alu 1
  matrix.drawLine(10, 16, 13, 19, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 16, 10, 19, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 19, 11, 20, matrix.Color333(7, 7,
0));
  matrix.drawLine(11, 20, 11, 21, matrix.Color333(7, 7,
0));
  matrix.drawLine(11, 21, 10, 22, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 22, 10, 25, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 25, 13, 22, matrix.Color333(7, 7,
0));
  matrix.drawLine(13, 22, 13, 19, matrix.Color333(7, 7,
0));
// Alu plus adding 4

  matrix.drawLine(8, 23, 9, 23, matrix.Color333(2.5,
2.5, 2.5));
  //4
  matrix.drawLine(6, 23, 4, 23, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(4, 23, 4, 22, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(6, 22, 6, 25, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(12, 11, 18, 11, matrix.Color333(2.5,
2.5, 2.5));


  //Registerfile
  matrix.drawRect(27, 16, 7, 15, matrix.Color333(0, 7,
0));
```

Table: Source Code for the main program

```
//mux 1
  matrix.drawLine(22, 19, 24, 21, matrix.Color333(7,
0,7));
  matrix.drawLine(22, 19, 22, 25, matrix.Color333(7,
0,7));
  matrix.drawLine(22, 25, 24, 23, matrix.Color333(7,
0,7));
  matrix.drawLine(24, 23, 24, 21, matrix.Color333(7,
0,7));
  //mux

  //bus to reg file 19:16
  matrix.drawLine(18, 17, 26, 17, matrix.Color333(2.5,
2.5, 2.5));
  //bus to mux 3:0
  matrix.drawLine(18, 20, 21, 20, matrix.Color333(2.5,
2.5, 2.5));

  //bus to reg file 15:12
  matrix.drawLine(18, 27, 27, 27, matrix.Color333(2.5,
2.5, 2.5));

  //bus to mux 15:12
  matrix.drawLine(20 ,24, 21, 24, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(20 ,24, 20, 27, matrix.Color333(2.5,
2.5, 2.5));

  //mux to reg file
  matrix.drawLine(25 ,22, 26, 22, matrix.Color333(2.5,
2.5, 2.5));


  //extend
  matrix.drawLine(23, 43, 31, 35, matrix.Color333(0, 7,
0));
  matrix.drawLine(23, 43, 23, 46, matrix.Color333(0, 7,
0));
  matrix.drawLine(23, 46, 31, 46, matrix.Color333(0, 7,
0));
  matrix.drawLine(31, 46, 31, 35, matrix.Color333(0, 7,
0));



  //mux 2
  matrix.drawLine(39, 28, 39, 34, matrix.Color333(7,
0,7));
  matrix.drawLine(39, 28, 41, 30, matrix.Color333(7,
0,7));
  matrix.drawLine(39, 34, 41, 32, matrix.Color333(7,
0,7));
  matrix.drawLine(41, 30, 41, 32, matrix.Color333(7,
0,7));

  // line reg to mux
  matrix.drawLine(34, 29, 38, 29, matrix.Color333(2.5,
2.5, 2.5));

  // line extender to mux
  matrix.drawLine(31, 42, 36, 42, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(36, 42, 36, 33, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(36, 33, 38, 33, matrix.Color333(2.5,
2.5, 2.5));

  //alu2
  matrix.drawLine(43, 24, 46, 27, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 24, 43, 27, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 27, 44, 28, matrix.Color333(7, 7,
0));
  matrix.drawLine(44, 28, 44, 29, matrix.Color333(7, 7,
0));
matrix.drawLine(28, 9, 40, 9, matrix.drawLine(44, 29,
43, 30, matrix.Color333(7, 7, 0));
  matrix.drawLine(43, 30, 43, 33, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 33, 46, 30, matrix.Color333(7, 7,
0));
  matrix.drawLine(46, 30, 46, 27, matrix.Color333(7, 7,
0));

  //Data Memory

  matrix.drawRect(49, 23, 6, 17, matrix.Color333(0, 7,
0));//reg file

  //mux 3
  matrix.drawLine(57, 30, 59, 32, matrix.Color333(7,
0,7));
  matrix.drawLine(57, 30, 57, 36, matrix.Color333(7,
0,7));
  matrix.drawLine(57, 36, 59, 34, matrix.Color333(7,
0,7));
  matrix.drawLine(59, 34, 59, 32, matrix.Color333(7,
0,7));

  //mux2 to alu
  matrix.drawLine(42, 31, 42, 31, matrix.Color333(2.5,
2.5, 2.5));

  //alu to data memory
  matrix.drawLine(46, 29, 48, 29, matrix.Color333(2.5,
2.5, 2.5));

  //rd2 to data memory
  matrix.drawLine(37, 29, 37, 37, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(37, 37, 48, 37, matrix.Color333(2.5,
2.5, 2.5));

  //rd1 to alu
  matrix.drawLine(34, 26, 42, 26, matrix.Color333(2.5,
2.5, 2.5));

  //data mem to mux
  matrix.drawLine(54, 32, 57, 32, matrix.Color333(2.5,
2.5, 2.5));

  //alu result to mux
  matrix.drawLine(47, 29, 47, 42, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(47, 42, 55, 42, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(55, 42, 55, 34, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(55, 34, 56, 34, matrix.Color333(2.5,
2.5, 2.5));

  //result to wd3
  matrix.drawLine(27, 29, 20, 29, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(20, 29, 20, 52, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(20, 52, 61, 52, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(61,52, 61, 33, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(61, 33, 60, 33, matrix.Color333(2.5,
2.5, 2.5));


}
#include "Fonts/FreeSerif9pt7b.h"
#include "Fonts/FreeSerifBoldItalic9pt7b.h"
#include "Fonts/RobotoMono_Thin7pt7b.h"
#include "Fonts/FreeSans9pt7b.h"

void loop()
{


  //LDR_R1_R2_imm();
  matrix.setCursor(40, 54);
  matrix.println("LDR");
  LDR();
  screen_clear();
```

```
matrix.setCursor(40, 54);
  matrix.println("ADD");
  ADD();
  screen_clear();

  //SUB_R1_R2_imm
  matrix.setCursor(40, 54);
  matrix.println("SUB");
  SUB();
  screen_clear();
}

//Clear screen
void screen_clear()
{
  matrix.fillRect(40, 54, 20, 10, matrix.Color333(0, 0,
0));
}



void LDR()
{
  matrix.drawLine(2, 11, 3, 11, matrix.Color333(7, 0,
0));
  delay(2000);
  //pc block Red
  matrix.drawRect(4, 9, 3, 4, matrix.Color333(7, 0,
0));
  matrix.drawLine(2, 11, 3, 11, matrix.Color333(2.5,
2.5, 2.5));
  delay(2000);
  //Pc block Normal
  matrix.drawRect(4, 9, 3, 4, matrix.Color333(0, 7,
0));
  delay(2000);

  // Pc block to IM
  matrix.drawLine(7, 11, 7, 11, matrix.Color333(7, 0,
0));
  delay(2000);
  // Pc block to IM
  matrix.drawRect(8, 8, 4, 7, matrix.Color333(7, 0,
0));//ins memory
  matrix.drawLine(7, 11, 7, 11, matrix.Color333(2.5,
2.5, 2.5)); // Pc block to IM
  delay(2000);

  //inst memory tto bus
  matrix.drawLine(12, 11, 17, 11, matrix.Color333(7, 0,
0));
  matrix.drawRect(8, 8, 4, 7, matrix.Color333(0, 7,
0));//ins memory
  delay(2000);

  //Bus
  matrix.drawLine(18, 1, 18, 60, matrix.Color333(7, 0,
0));
  //inst memory tto bus
  matrix.drawLine(12, 11, 17, 11, matrix.Color333(2.5,
2.5, 2.5));
  delay(2000);


  //Bus Normal
  matrix.drawLine(18, 1, 18, 60, matrix.Color333(2.5,
2.5, 2.5));

  //bus to Control, Rg and extend
  //Control
  //bus to extend
  matrix.drawLine(23, 45, 19, 45, matrix.Color333(7, 0,
0));

  matrix.drawLine(18, 3, 21, 3, matrix.Color333(7, 0,
0));///cond
  matrix.drawLine(18, 5, 21, 5, matrix.Color333(7, 0,
0));//op
  matrix.drawLine(18, 7, 21, 7, matrix.Color333(7, 0,
0));//func
  matrix.drawLine(18, 9, 21, 9, matrix.Color333(7, 0,
0));//rd
```

```
matrix.drawLine(28, 9, 40, 9,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(40, 9, 40, 28,
matrix.Color333(0,4.5,3.5));

  //Control signal Path alu control
  matrix.drawLine(28, 7, 45, 7,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(45, 7, 45, 25,
matrix.Color333(0,4.5,3.5));

  //Control Signal path mem write
  matrix.drawLine(28, 5, 52, 5,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(52, 5, 52, 22,
matrix.Color333(0,4.5,3.5));

  // Control signal mem to reg
  matrix.drawLine(28, 3, 58, 3,
matrix.Color333(0,4.5,3.5));
  matrix.drawLine(58, 3, 58, 30,
matrix.Color333(0,4.5,3.5));

  //pc block
  matrix.drawRect(4, 9, 3, 4, matrix.Color333(0, 7,
0));

  //ins memory
  matrix.drawRect(8, 8, 4, 7, matrix.Color333(0, 7,
0));//ins memory

  //pc block to alu


  matrix.drawLine(7, 11, 7, 11, matrix.Color333(2.5,
2.5, 2.5)); // Pc block to IM
  matrix.drawLine(7, 11, 7, 17, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(7, 17, 10, 17, matrix.Color333(2.5,
2.5, 2.5));

  //alu 1
  matrix.drawLine(10, 16, 13, 19, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 16, 10, 19, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 19, 11, 20, matrix.Color333(7, 7,
0));
  matrix.drawLine(11, 20, 11, 21, matrix.Color333(7, 7,
0));
  matrix.drawLine(11, 21, 10, 22, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 22, 10, 25, matrix.Color333(7, 7,
0));
  matrix.drawLine(10, 25, 13, 22, matrix.Color333(7, 7,
0));
  matrix.drawLine(13, 22, 13, 19, matrix.Color333(7, 7,
0));
// Alu plus adding 4

  matrix.drawLine(8, 23, 9, 23, matrix.Color333(2.5,
2.5, 2.5));
  //4
  matrix.drawLine(6, 23, 4, 23, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(4, 23, 4, 22, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(6, 22, 6, 25, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(12, 11, 18, 11, matrix.Color333(2.5,
2.5, 2.5));


  //Registerfile
  matrix.drawRect(27, 16, 7, 15, matrix.Color333(0, 7,
0));
```

```
//data mem to mux
  matrix.drawLine(54, 32, 57, 32, matrix.Color333(7, 0,
0));
  delay(2000);

//data mem to mux
  matrix.drawLine(54, 32, 57, 32, matrix.Color333(2.5,
2.5, 2.5));

  //mux 3 Red
  matrix.drawLine(57, 30, 59, 32, matrix.Color333(7,
0,0));
  matrix.drawLine(57, 30, 57, 36, matrix.Color333(7,
0,0));
  matrix.drawLine(57, 36, 59, 34, matrix.Color333(7,
0,0));
  matrix.drawLine(59, 34, 59, 32, matrix.Color333(7,
0,0));
  delay(2000);

  // Control signal mem to reg Red
  matrix.drawLine(28, 3, 58, 3, matrix.Color333(7, 0,
0));
  delay(400);
  matrix.drawLine(58, 3, 58, 30, matrix.Color333(7, 0,
0));
  delay(2000);




  //mux 3 Normal
  matrix.drawLine(57, 30, 59, 32, matrix.Color333(7,
0,7));
  matrix.drawLine(57, 30, 57, 36, matrix.Color333(7,
0,7));
  matrix.drawLine(57, 36, 59, 34, matrix.Color333(7,
0,7));
  matrix.drawLine(59, 34, 59, 32, matrix.Color333(7,
0,7));
  // Control signal mem to reg Normal
  matrix.drawLine(28, 3, 58, 3,
matrix.Color333(0,3.5,3.5));
  matrix.drawLine(58, 3, 58, 30,
matrix.Color333(0,3.5,3.5));

  delay(2000);

   //result to wd3
   matrix.drawLine(61, 33, 60, 33,
matrix.Color333(7,0,0));
  delay(2000);
   matrix.drawLine(61,52, 61, 33,
matrix.Color333(7,0,0));
  delay(400);
   matrix.drawLine(20, 52, 61, 52,
matrix.Color333(7,0,0));
  delay(400);
   matrix.drawLine(20, 29, 20, 52,
matrix.Color333(7,0,0));
  delay(400);
  matrix.drawLine(27, 29, 20, 29,
matrix.Color333(7,0,0));
  delay(400);
```

```
//Registerfile
  matrix.drawRect(27, 16, 7, 15, matrix.Color333(0, 7,
0));
  delay(2000);

  // Alu plus adding 4

  //4 Red
  matrix.drawLine(6, 23, 4, 23, matrix.Color333(7, 0,
0));
  matrix.drawLine(4, 23, 4, 22, matrix.Color333(7, 0,
0));
  matrix.drawLine(6, 22, 6, 25, matrix.Color333(7, 0,
0));
  delay(2000);


  // Alu plus adding 4 Red

  matrix.drawLine(8, 23, 9, 23, matrix.Color333(7, 0,
0));
  // pc to alu red
  matrix.drawLine(7, 11, 7, 17, matrix.Color333(7, 0,
0));
  matrix.drawLine(7, 17, 10, 17, matrix.Color333(7, 0,
0));
  delay(2000);



  //alu 1 Red
  matrix.drawLine(10, 16, 13, 19, matrix.Color333(7, 0,
0));
  matrix.drawLine(10, 16, 10, 19, matrix.Color333(7, 0,
0));
  matrix.drawLine(10, 19, 11, 20, matrix.Color333(7, 0,
0));
  matrix.drawLine(11, 20, 11, 21, matrix.Color333(7, 0,
0));
  matrix.drawLine(11, 21, 10, 22, matrix.Color333(7, 0,
0));
  matrix.drawLine(10, 22, 10, 25, matrix.Color333(7, 0,
0));
  matrix.drawLine(10, 25, 13, 22, matrix.Color333(7, 0,
0));
  matrix.drawLine(13, 22, 13, 19, matrix.Color333(7, 0,
0));



  //4 Normal
  matrix.drawLine(6, 23, 4, 23, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(4, 23, 4, 22, matrix.Color333(2.76,
3.88, 1.82));
  matrix.drawLine(6, 22, 6, 25, matrix.Color333(2.76,
3.88, 1.82));

//
  matrix.drawLine(8, 23, 9, 23, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(7, 11, 7, 17, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(7, 17, 10, 17, matrix.Color333(2.5,
2.5, 2.5));

  delay(2000);
```

```
//extend Normal
  matrix.drawLine(23, 43, 31, 35, matrix.Color333(0, 7,
0));
  matrix.drawLine(23, 43, 23, 46, matrix.Color333(0, 7,
0));
  matrix.drawLine(23, 46, 31, 46, matrix.Color333(0, 7,
0));
  matrix.drawLine(31, 46, 31, 35, matrix.Color333(0, 7,
0));

//Registerfile Normal
  matrix.drawRect(27, 16, 7, 15, matrix.Color333(0, 7,
0));
  delay(2000);

//Line rd1 to alu Red
  matrix.drawLine(34, 26, 42, 26,
matrix.Color333(7,0,0));

// line extender to mux
  matrix.drawLine(31, 42, 36, 42,
matrix.Color333(7,0,0));
  matrix.drawLine(36, 42, 36, 33,
matrix.Color333(7,0,0));
  matrix.drawLine(36, 33, 38, 33,
matrix.Color333(7,0,0));

  delay(2000);

//mux 2 Red
  matrix.drawLine(39, 28, 39, 34,
matrix.Color333(7,0,0));
  matrix.drawLine(39, 28, 41, 30,
matrix.Color333(7,0,0));
  matrix.drawLine(39, 34, 41, 32,
matrix.Color333(7,0,0));
  matrix.drawLine(41, 30, 41, 32,
matrix.Color333(7,0,0));
// line extender to mux normal
  matrix.drawLine(31, 42, 36, 42, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(36, 42, 36, 33, matrix.Color333(2.5,
2.5, 2.5));
  matrix.drawLine(36, 33, 38, 33, matrix.Color333(2.5,
2.5, 2.5));
  delay(2000);

  //Control signal Path alu source Red
  matrix.drawLine(28, 9, 40, 9,
matrix.Color333(7,0,0));
  delay(200);
  matrix.drawLine(40, 9, 40, 28,
matrix.Color333(7,0,0));
  delay(2000);

  //Control signal Path alu source Red
  matrix.drawLine(28, 9, 40, 9,
matrix.Color333(0,3.5,3.5));
  matrix.drawLine(40, 9, 40, 28,
matrix.Color333(0,3.5,3.5));
  delay(2000);

  //mux 2 Normal
  matrix.drawLine(39, 28, 39, 34, matrix.Color333(7,
0,7));
  matrix.drawLine(39, 28, 41, 30, matrix.Color333(7,
0,7));
  matrix.drawLine(39, 34, 41, 32, matrix.Color333(7,
0,7));
  matrix.drawLine(41, 30, 41, 32, matrix.Color333(7,
0,7));
```

```
//Line rd1 to alu Normal
  matrix.drawLine(34, 26, 42, 26,
matrix.Color333(2.5,2.5,2.5));

  //mux2 to alu
  matrix.drawLine(42, 31, 42, 31, matrix.Color333(2.5,
2.5, 2.5));

  delay(2000);

    //Control signal Path alu control Red
  matrix.drawLine(28, 7, 45, 7,
matrix.Color333(7,0,0));
  delay(400);
  matrix.drawLine(45, 7, 45, 25,
matrix.Color333(7,0,0));
  delay(2000);


  //Control signal Path alu control
  matrix.drawLine(28, 7, 45, 7,
matrix.Color333(0,3.5,3.5));
  matrix.drawLine(45, 7, 45, 25,
matrix.Color333(0,3.5,3.5));
 //alu2
  matrix.drawLine(43, 24, 46, 27, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 24, 43, 27, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 27, 44, 28, matrix.Color333(7, 7,
0));
  matrix.drawLine(44, 28, 44, 29, matrix.Color333(7, 7,
0));
  matrix.drawLine(44, 29, 43, 30, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 30, 43, 33, matrix.Color333(7, 7,
0));
  matrix.drawLine(43, 33, 46, 30, matrix.Color333(7, 7,
0));
  matrix.drawLine(46, 30, 46, 27, matrix.Color333(7, 7,
0));
  delay(2000);

//alu to data memory Red
  matrix.drawLine(46, 29, 48, 29, matrix.Color333(7, 0,
0));
  delay(2000);

//Data Memory Red
  matrix.drawRect(49, 23, 6, 17, matrix.Color333(7, 0,
0));//reg file
//alu to data memory
  matrix.drawLine(46, 29, 48, 29, matrix.Color333(2.5,
2.5, 2.5));
  delay(2000);

//Control Signal path mem write
  matrix.drawLine(28, 5, 52, 5, matrix.Color333(7, 0,
0));
  delay(400);
  matrix.drawLine(52, 5, 52, 22, matrix.Color333(7, 0,
0));
  delay(2000);

//Control Signal path mem write
  matrix.drawLine(28, 5, 52, 5,
matrix.Color333(0,3.5,3.5));
  matrix.drawLine(52, 5, 52, 22,
matrix.Color333(0,3.5,3.5));
//Data Memory Red
  matrix.drawRect(49, 23, 6, 17, matrix.Color333(0, 7,
0));//reg file
  delay(2000);
```

4 Implementation

4.1 Description

This is the description for the design

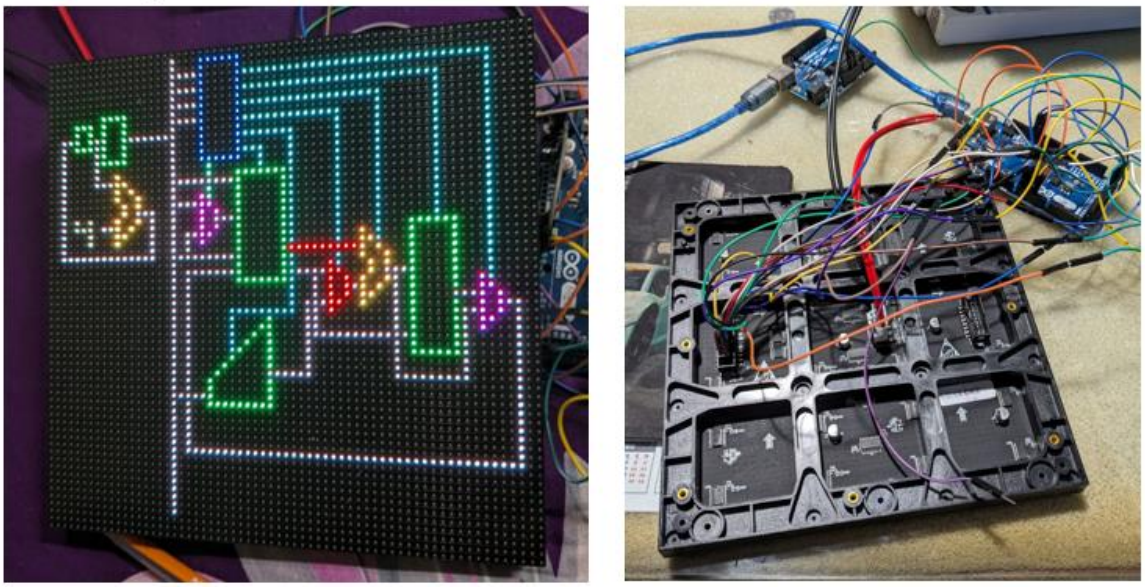


Figure 2: (Left)Front Layout and (Right) Back layout

5 Design Analysis and Evaluation

5.1 Novelty

We are implementing this project for the students to get an in hand experience of Single Cycle Processor working cycle. It will be a good visual approach for learning of students. We haven't found much resources about it in the Internet.

5.2 Design Considerations (PO(c))

5.2.1 Considerations to public health and safety

This project is low carbon emitting and safe for the public health.

5.2.2 Considerations to environment

Single Cycle Processor must be less energy consuming and less carbon emitting.

5.2.3 Considerations to cultural and societal needs

The illustration must be easy to understand and not clumsy at all.

5.3 Investigations (PO(d))

5.3.1 Design of Experiment

The design (circuit diagram) of the project is taken from our textbook "Digital Design and Computer Architecture. ARM Edition" written by Sarah Harris and David Harris.

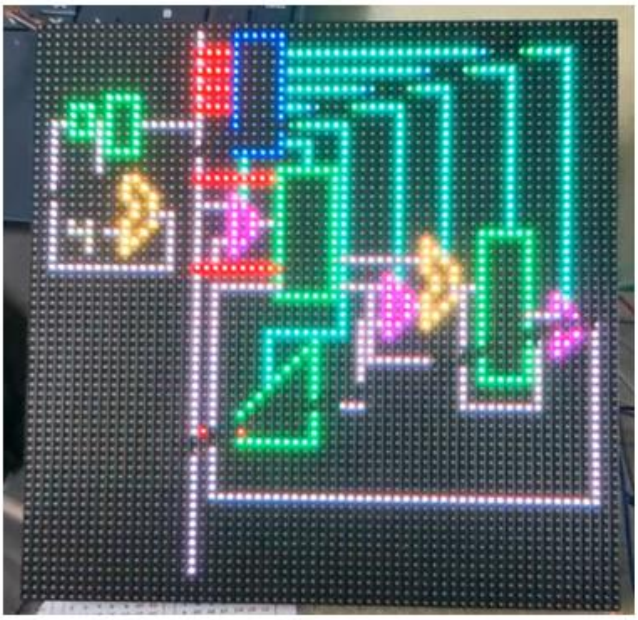
5.3.2 Data Collection

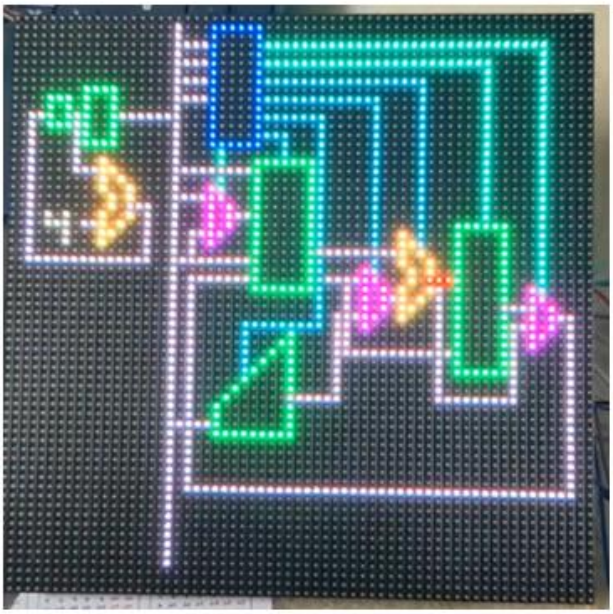
We took our resources from textbook and various internet resources.

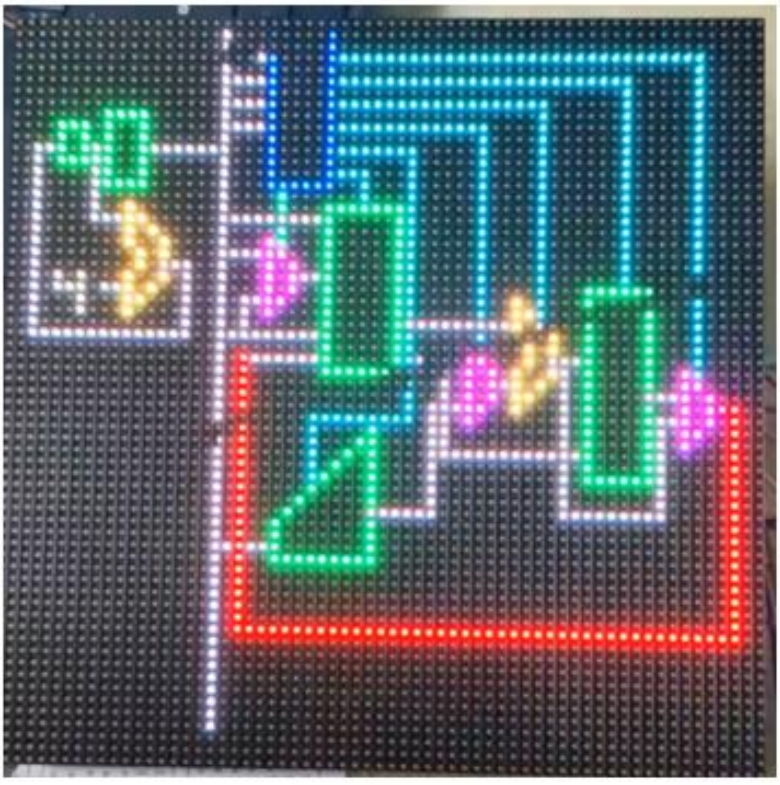
5.3.3 Results and Analysis

Our model runs smoothly and we can show 3 operation of single cycle processor with it.

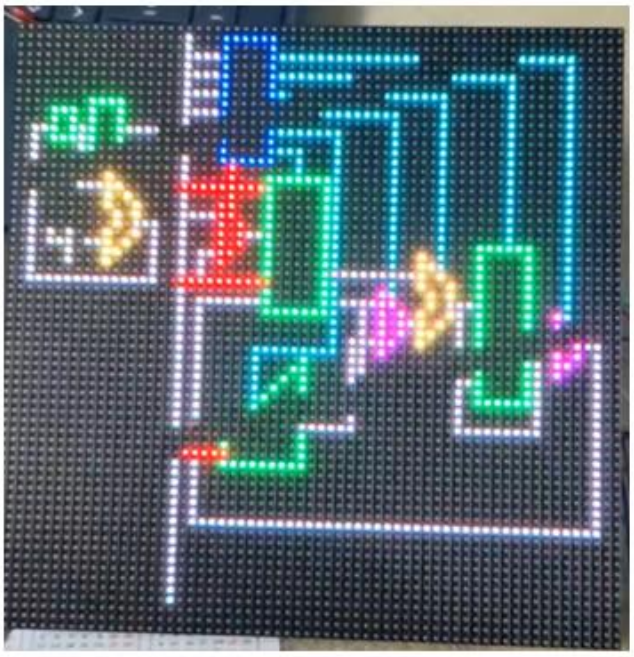
LDR

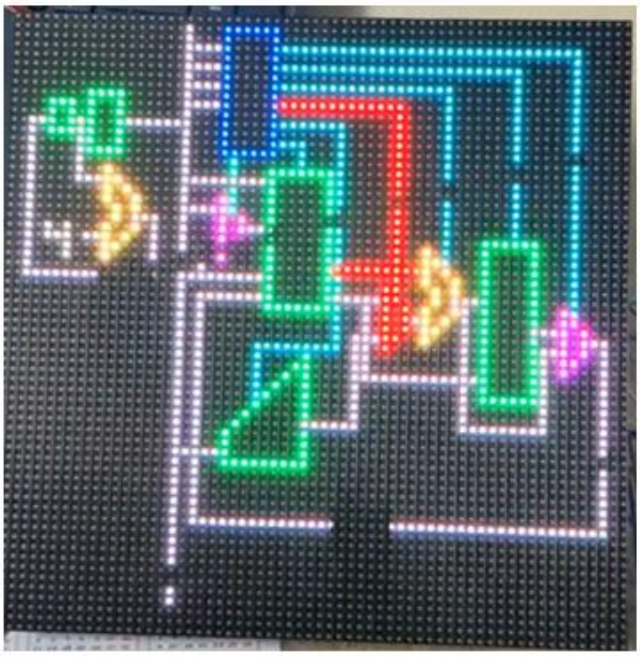


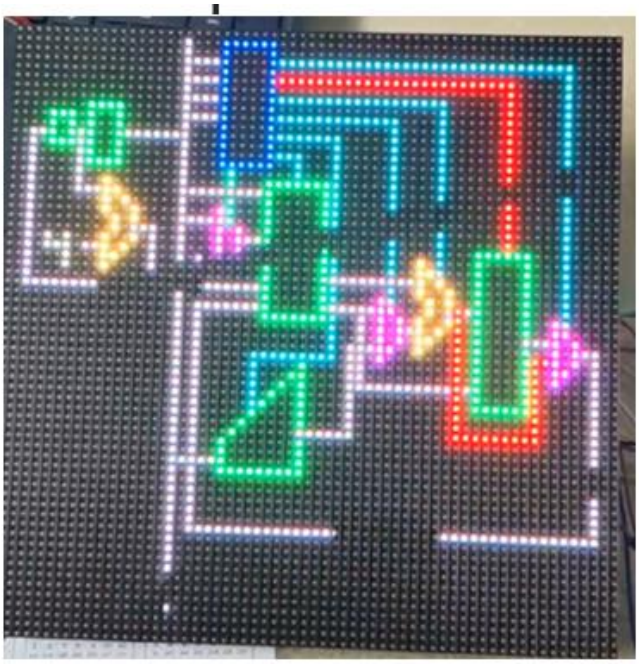


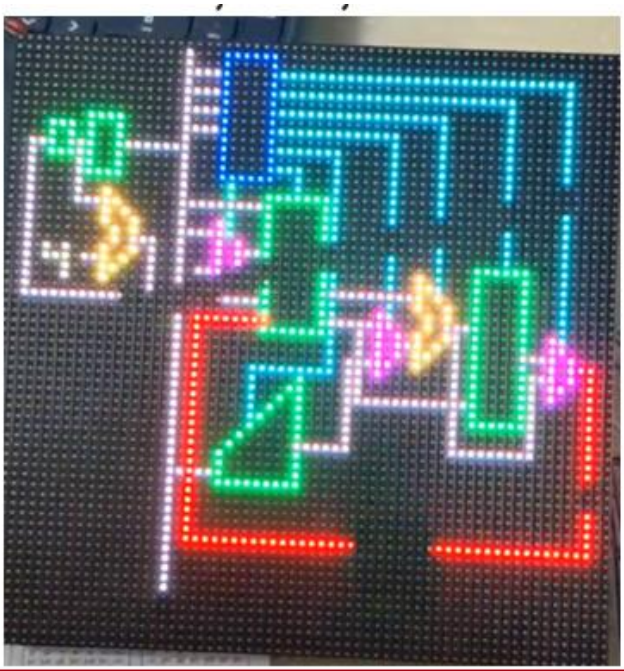


ADD

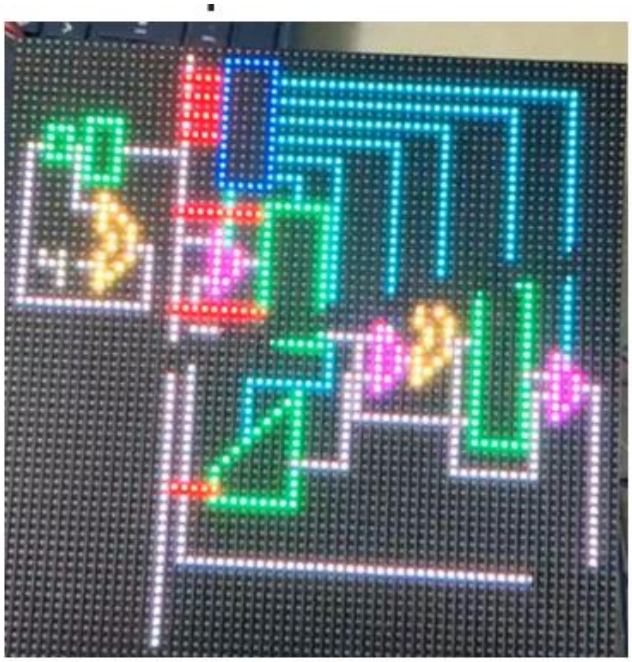


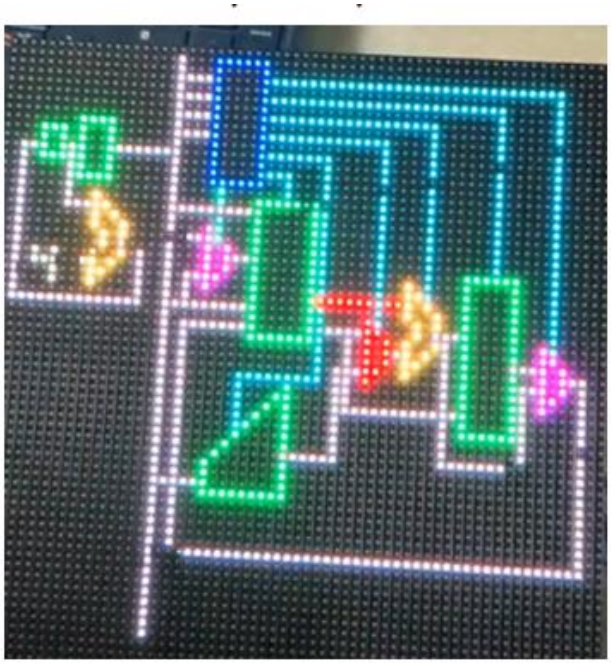


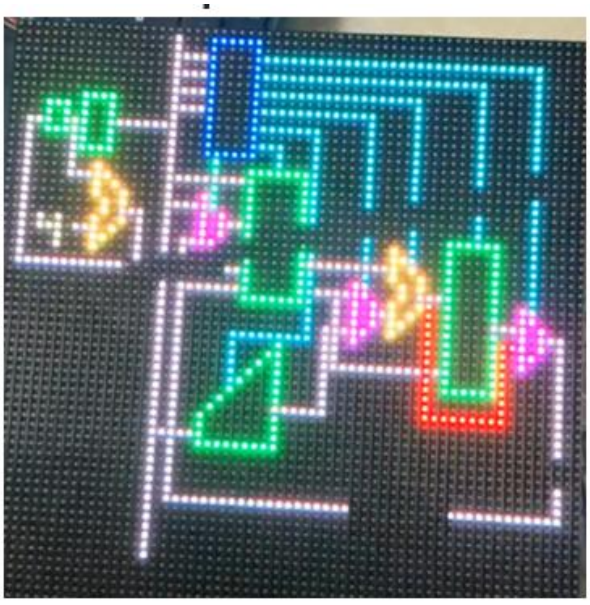


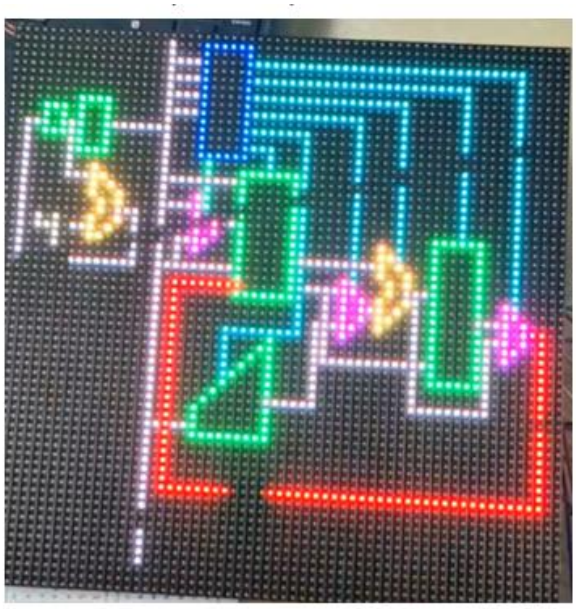


SUB









5.4 Limitations of Tools (PO(e))

We could not merge or manage to implement a bigger matrix so we could not show the branching.

5.5Impact Assessment (PO(f))

5.5.1 Assessment of Societal and Cultural Issues

The knowledge gained from this project can be used to build more advanced processors which can be used to make more advanced control systems as well as advanced devices.

5.5.2 Assessment of Health and Safety Issues

This project is low carbon emitting and less power consuming. So it is safe for the public to use.

5.5.3 Assessment of Legal Issues

This project is totally plagiarism free and all credits are given.

5.6 Sustainability Evaluation (PO(g))

This project is very much sustainable as it operates on very low power.

5.7Ethical Issues (PO(h))

Ethically this project is totally ideal. No social, cultural and public sentiments are hurt by this.

6 Reflection on Individual and Team work (PO(i))

6.1Individual Contribution of Each Member

1906172 - Kaniz Fatema Zarin	- LED Strip utilization
1906180 - Abrar Jahin Niloy	- Coding
1906182 - Md. Mahir Ashif	- Hardware Implementation
1906183 - Zahin Tazwar	- Single Cycle Processor Datapath Management

6.2 Mode of TeamWork and diversity

We worked both online and offline to complete the project. All the team members were enthusiastic to do the project work. Each member gave their best effort for the project. We really enjoyed working together.

7 Communication to External Stakeholders (PO(j))

7.1 Github Link

https://github.com/Abrar-Niloy/single-cycle_processor_LED_Matrix.git

7.2 YouTube Link

https://youtu.be/vXuFqwPZ0sk

8 Project Management and Cost Analysis (PO(k))

8.1 Bill of Materials

	A	B	C	D
1	Component	Per Unit Price	Unit count	Price
2	64*64 RGB LED Matrix	2000	2	4000
3	Arduino Mega	1590	1	1590
4	Power Supply	500	1	500
5	ESP32	440	1	440
6	Jumper Wires	30	2	60
7	LED Strip	450	1	450
8	Transportation	400		400
9				7440
10				

9 Future Work (PO(l))

1) Expand the led matrix to implement branches.

2) Labeling each line and add instructions around the matrix to make it more comprehensible

10 References

https://www.waveshare.com/wiki/RGB-Matrix-P3-64x64?fbclid=IwAR26jVtsXXGJiy-9WncQMiu-R42p7zqwcKAQrFhLlblqjcTRpCjMmaHK8_Q