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

EEE 304(January 2023)

Digital Electronics Laboratory

Final Project Report

Section: C2 Group: 03

8 BIT COMPUTER

Course Instructors:

Shafin Bin Hamid, Lecturer
Sadat Tahmeed Azad, Part-Time Lecturer

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: Md. Nayem Hasan** **Student ID: 1906175**	**Signature:** _____ **Full Name: Tanvir Rahman** **Student ID:1906176**
Signature: _____ **Full Name: Samiul Hossain** **Student ID:1906179**	**Signature:** _____ **Full Name: Tousif Ansari** **Student ID:1906190**

Table of Contents

1 Abstract

In this project our aim is to implement a programmable 8 Bit computer which can load input data from RAM to perform mathematical calculations and display the output using 7 Segment display. This project can be divided into individual sections for easier understanding of the workflow. The sections are clock module, Registers, Arithmatic Logic Unit, RAM, Program Counter, Output Register, Bringing all the segments together, CPU Control logic. The computer will operate according to the pre-defined instruction set built for the computer. In this project we will build this 8 Bit computer in breadboards using simple logic gates and will perform mathematical operations.

2 Introduction

In this project our intention is to build an 8 Bit computer and do mathematical operations using the computer. For this we have to build all the necessary parts of a simple computer. We have to build Program Counter, Memory Address register, RAM, Arithmatic Logic Unit, Instruction Register, Output Register, CPU Control Logic, Output Display. After building all the individual parts we have to bring all the parts together using a common bus. Any block will transfer data to another block using the bus. CPU Control logic is the heart of this computer. Each of the components has some control signal going into it. We have to build some logics to turn on appropriate control signals in every step of a mathematical operation.

To perform any operation like load data, ADD or Subtract we have to build some instruction set depending on the control signal needed to be turn on to perform a certain operation. In this project building such instruction set is quite challenging for us. Besides the connection set up for each block is very complex because a large number of connections needed. There is a chance to make mistake while making connections. A single mistake in connection or in instruction set will result in erroneous output. These are the complex engineering problem in this project.

3 Design

3.1 Problem Formulation

3.1.1 Identification of Scope

In this project we intend to solve mathematical problems using the 8-bit computer and thus, understand how computers work. We can implement various instructions to increase the operations that can be computed. But as it is a breadboard computer, there are several restrictions too. But overall, this project should provide a better understanding of how computers work in general.

3.1.2 Literature Review

This is the first activity of the project is to read the first nine chapters of Digital Computer Electronics by Malvino (1983). The early chapters cover the digital electronics fundamentals which are recap of the Digital Electronics . Subsequently,various applications of the fundamental logic components such as the adder,subtracter, flip-flops, registers, counters, and memory are demonstrated. Along the descriptions in these latter chapters, sub-modules of SAP-1 have been introduced as appropriate. This has ensured good understanding during literature study of SAP-1. Also , literature study of SAP-1 was done based on chapter 10 of that book. In this chapter, how small digital circuits introduced previously are combined for a simple computer system has been seen. Details of this activity have been discussed in Chapter 2 of this report.

3.1.3 Formulation of Problem

There are various segments in our project like clock, counter, register, ram, control module etc. The idea is to follow the rules of SAP-1 computer and create the computer using 74 series ICs. We followed the proper instructions and used 74LS chips for our computer.

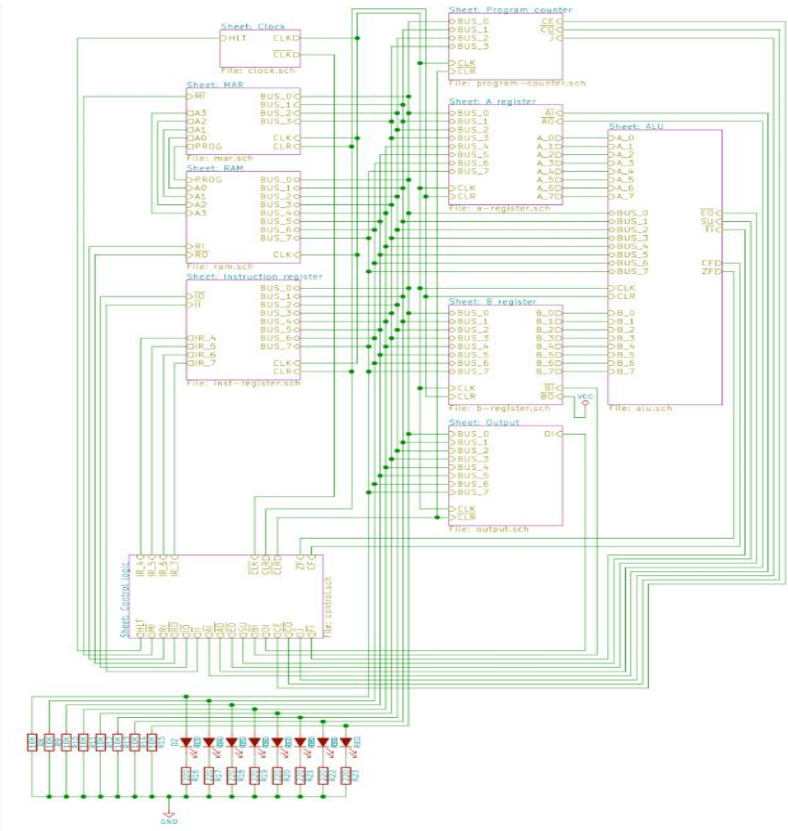
3.1.4 Analysis

The project is dependent upon some clock triggered modules. Only the Alu is always active. A special module we used in this project was the EEPROM and it is used to control the project. This is a vital part of the 8 bit computer. The clock operates the whole system but there is also another clock that is used to compute micro steps in each large instruction.

3.2 Design Method

We had to perform some calculations while doing the project. These are clock speed for manual and auto clock modes, synchronizing all the clock connected modules. For the control circuit we used negative clock so that it didn't interfere with the rest of the positive edge triggered modules.

3.3Circuit Diagram



3.4Full Source Code of Firmware

Controller Code

```
#define SHIFT_DATA 2
#define SHIFT_CLK 3
#define SHIFT_LATCH 4
#define EEPROM_D0 5
#define EEPROM_D7 12
#define WRITE_EN 13

#define HLT 0b1000000000000000  // Halt clock
#define MI  0b0100000000000000  // Memory address register in
#define RI  0b0010000000000000  // RAM data in
#define RO  0b0001000000000000  // RAM data out
#define IO  0b0000100000000000  // Instruction register out
#define II  0b0000010000000000  // Instruction register in
#define AI  0b0000001000000000  // A register in
#define AO  0b0000000100000000  // A register out
```

```
#define EO  0b0000000010000000  // ALU out
#define SU  0b0000000001000000  // ALU subtract
#define BI  0b0000000000100000  // B register in
#define OI  0b0000000000010000  // Output register in
#define CE  0b0000000000001000  // Program counter enable
#define CO  0b0000000000000100  // Program counter out
#define J   0b0000000000000010  // Jump (program counter in)

uint16_t data[] = {
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 0000 - NOP
  MI|CO,  RO|II|CE,  IO|MI,  RO|AI,  0,         0, 0, 0,   // 0001 - LDA
  MI|CO,  RO|II|CE,  IO|MI,  RO|BI,  EO|AI,     0, 0, 0,   // 0010 - ADD
  MI|CO,  RO|II|CE,  IO|MI,  RO|BI,  EO|AI|SU,  0, 0, 0,   // 0011 - SUB
  MI|CO,  RO|II|CE,  IO|MI,  AO|RI,  0,         0, 0, 0,   // 0100 - STA
  MI|CO,  RO|II|CE,  IO|AI,  0,      0,         0, 0, 0,   // 0101 - LDI
  MI|CO,  RO|II|CE,  IO|J,   0,      0,         0, 0, 0,   // 0110 - JMP
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 0111
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1000
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1001
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1010
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1011
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1100
  MI|CO,  RO|II|CE,  0,      0,      0,         0, 0, 0,   // 1101
  MI|CO,  RO|II|CE,  AO|OI,  0,      0,         0, 0, 0,   // 1110 - OUT
  MI|CO,  RO|II|CE,  HLT,    0,      0,         0, 0, 0,   // 1111 - HLT
};


/*
 * Output the address bits and outputEnable signal using shift registers.
 */
void setAddress(int address, bool outputEnable) {
  shiftOut(SHIFT_DATA, SHIFT_CLK, MSBFIRST, (address >> 8) | (outputEnable ? 0x00 : 0x80));
  shiftOut(SHIFT_DATA, SHIFT_CLK, MSBFIRST, address);

  digitalWrite(SHIFT_LATCH, LOW);
  digitalWrite(SHIFT_LATCH, HIGH);
  digitalWrite(SHIFT_LATCH, LOW);
}


/*
 * Read a byte from the EEPROM at the specified address.
 */
byte readEEPROM(int address) {
  for (int pin = EEPROM_D0; pin <= EEPROM_D7; pin += 1) {
    pinMode(pin, INPUT);
  }
  setAddress(address, /*outputEnable*/ true);

  byte data = 0;
  for (int pin = EEPROM_D7; pin >= EEPROM_D0; pin -= 1) {
    data = (data << 1) + digitalRead(pin);
  }
  return data;
}


/*
 * Write a byte to the EEPROM at the specified address.
 */
void writeEEPROM(int address, byte data) {
  setAddress(address, /*outputEnable*/ false);
  for (int pin = EEPROM_D0; pin <= EEPROM_D7; pin += 1) {
    pinMode(pin, OUTPUT);
  }

  for (int pin = EEPROM_D0; pin <= EEPROM_D7; pin += 1) {
    digitalWrite(pin, data & 1);
    data = data >> 1;
  }
  digitalWrite(WRITE_EN, LOW);
  delayMicroseconds(1);
  digitalWrite(WRITE_EN, HIGH);
  delay(10);
```

```
}


/*
 * Read the contents of the EEPROM and print them to the serial monitor.
 */
void printContents() {
  for (int base = 0; base <= 255; base += 16) {
    byte data[16];
    for (int offset = 0; offset <= 15; offset += 1) {
      data[offset] = readEEPROM(base + offset);
    }

    char buf[80];
    sprintf(buf,
"%03x:  %02x %02x %02x %02x %02x %02x %02x %02x   %02x %02x %02x %02x %02x %02x %02x %02x",
            base, data[0], data[1], data[2], data[3], data[4], data[5], data[6], data[7],
            data[8], data[9], data[10], data[11], data[12], data[13], data[14], data[15]);

    Serial.println(buf);
  }
}


void setup() {
  // put your setup code here, to run once:
  pinMode(SHIFT_DATA, OUTPUT);
  pinMode(SHIFT_CLK, OUTPUT);
  pinMode(SHIFT_LATCH, OUTPUT);
  digitalWrite(WRITE_EN, HIGH);
  pinMode(WRITE_EN, OUTPUT);
  Serial.begin(57600);

  // Program data bytes
  Serial.print("Programming EEPROM");

  // Program the 8 high-order bits of microcode into the first 128 bytes of EEPROM
  for (int address = 0; address < sizeof(data)/sizeof(data[0]); address += 1) {
    writeEEPROM(address, data[address] >> 8);

    if (address % 64 == 0) {
      Serial.print(".");
    }
  }

  // Program the 8 low-order bits of microcode into the second 128 bytes of EEPROM
  for (int address = 0; address < sizeof(data)/sizeof(data[0]); address += 1) {
    writeEEPROM(address + 128, data[address]);

    if (address % 64 == 0) {
      Serial.print(".");
    }
  }

  Serial.println(" done");


  // Read and print out the contents of the EERPROM
  Serial.println("Reading EEPROM");
  printContents();
}


void loop() {
  // put your main code here, to run repeatedly:

}
```

Output register

```
#define SHIFT_DATA 2
#define SHIFT_CLK 3
#define SHIFT_LATCH 4
#define EEPROM_D0 5
#define EEPROM_D7 12
#define WRITE_EN 13
```

```
int base = 0;
void setAddress(int address,bool outputEnable){
  shiftOut(SHIFT_DATA,SHIFT_CLK,MSBFIRST,(address>>8)|(outputEnable? 0x00 : 0x80));
  shiftOut(SHIFT_DATA,SHIFT_CLK,MSBFIRST,address);
  digitalWrite(SHIFT_LATCH,LOW);
  digitalWrite(SHIFT_LATCH,HIGH);
  digitalWrite(SHIFT_LATCH,LOW);

}
byte readEEPROM(int address){
  for(int pin = EEPROM_D0; pin<=EEPROM_D7; pin += 1){
    pinMode(pin, INPUT);
  }
  setAddress(address,true);
  byte data_read = 0;
  for (int pin=EEPROM_D7; pin>=EEPROM_D0; pin -= 1){
    data_read = (data_read<<1) + digitalRead(pin);
  }
  return data_read;
}

void writeEEPROM(int address, byte data_write){
  for(int pin = EEPROM_D0; pin<=EEPROM_D7; pin += 1){
    pinMode(pin, OUTPUT);
  }
  setAddress(address,false);
  for(int pin=EEPROM_D0; pin<=EEPROM_D7; pin +=1){
      digitalWrite(pin,data_write & 1);
      data_write = data_write>>1;

  }
  digitalWrite(WRITE_EN,LOW);
  delayMicroseconds(1);
  digitalWrite(WRITE_EN, HIGH);
  delay(10);
}
 void printContents(){
   byte data[16];
   for (int base = 0; base <= 255; base += 16) {
    byte data[16];
    for (int offset = 0; offset <= 15; offset += 1) {
      data[offset] = readEEPROM(base + offset);
    }
    char buf[80];
    sprintf(buf,"%03x: %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x",
base, data[0], data[1], data[2],
    data[3],  data[4],data[5],  data[6],  data[7],  data[8],  data[9],  data[10],  data[11],  data[12],
data[13], data[14], data[15]);

    Serial.println(buf);
  }

 }

//byte data[] = {0x7e, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x70, 0x7f, 0x7b, 0x77, 0x1f, 0x4e, 0x3d,
0x4f, 0x47};

void setup(){
   pinMode(SHIFT_DATA, OUTPUT);
   pinMode(SHIFT_CLK, OUTPUT);
   pinMode(SHIFT_LATCH, OUTPUT);
   digitalWrite(WRITE_EN,HIGH);
   pinMode(WRITE_EN,OUTPUT);
   Serial.begin(57600);

   byte digits[] = {0x7e, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x70, 0x7f, 0x7b};

   Serial.println("Programming Ones place");
   for(int value=0; value<=255; value +=1){
     writeEEPROM(value,digits[(value%10)]);
   }

   Serial.println("Programming tens place");
```

```
  for(int value=0; value<=255; value +=1){
    writeEEPROM(value + 256,digits[(value/10) % 10]);
  }

  Serial.println("Programming hundreds place");

  for(int value=0; value<=255; value +=1){
    writeEEPROM(value + 512,digits[(value/100) % 10]);
  }

  Serial.println("Programming thousands place");

  for(int value=0; value<=255; value +=1){
    writeEEPROM(value + 768, 0);
  }

  Serial.println("Programming ones place(two's compliment)");

  for (int value = -128; value<=127; value +=1){
    writeEEPROM((byte)value + 1024, digits[abs(value)%10]);
  }

  Serial.println("Programming tens place(two's compliment)");

  for (int value = -128; value<=127; value +=1){
    writeEEPROM((byte)value + 1280, digits[abs(value/10)%10]);
  }

  Serial.println("Programming hundreds place(two's compliment)");

  for (int value = -128; value<=127; value +=1){
    writeEEPROM((byte)value + 1536, digits[abs(value/100)%10]);
  }

  Serial.println("Programming sign (two's compliment)");


  for(int value=-128; value<=127; value = value+1){
    if(value<0){
      writeEEPROM((byte)value+1792, 0x01);
    }
    else{
      writeEEPROM((byte)value+1792, 0);
    }
  }

  printContents();




}

void loop() {
  // put your main code here, to run repeatedly:

}
```

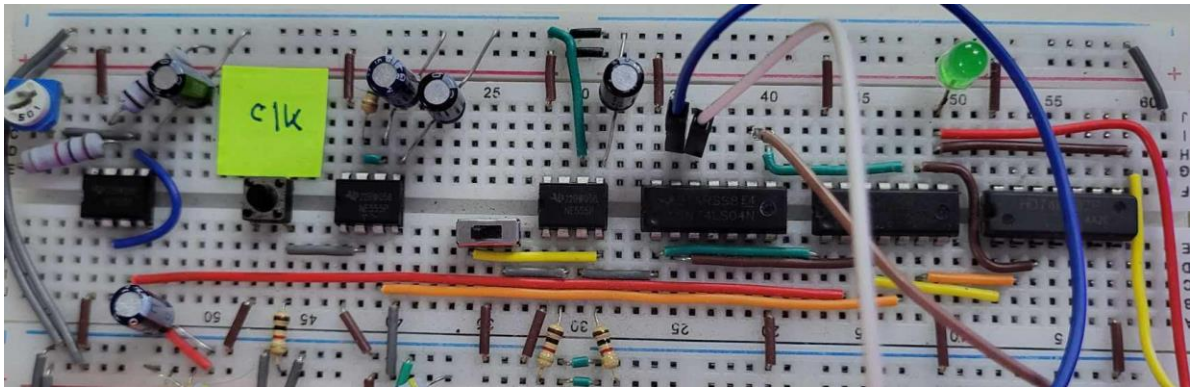
4 Implementation

4.1 Description

Clock:

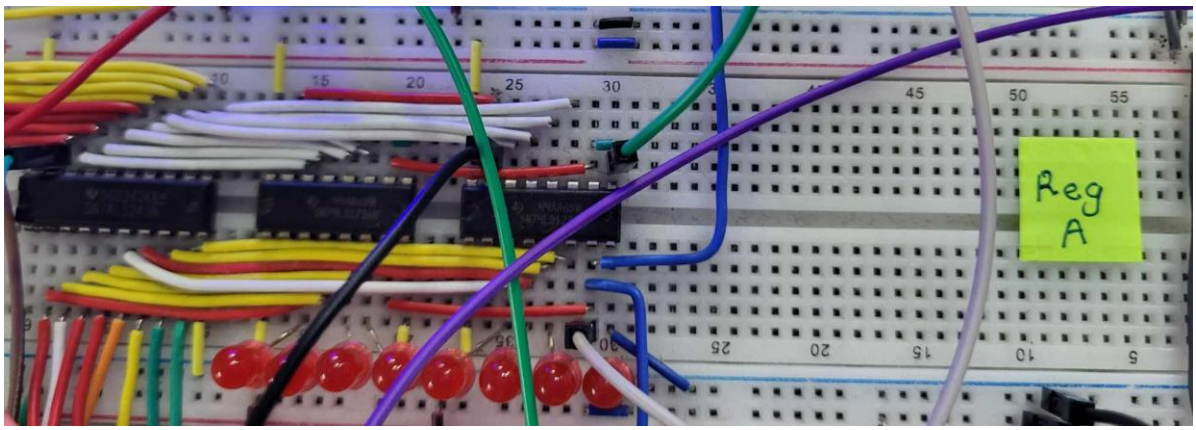
The computer's clock is used to synchronize all operations. The clock we're building is based on the popular 555 timer IC.

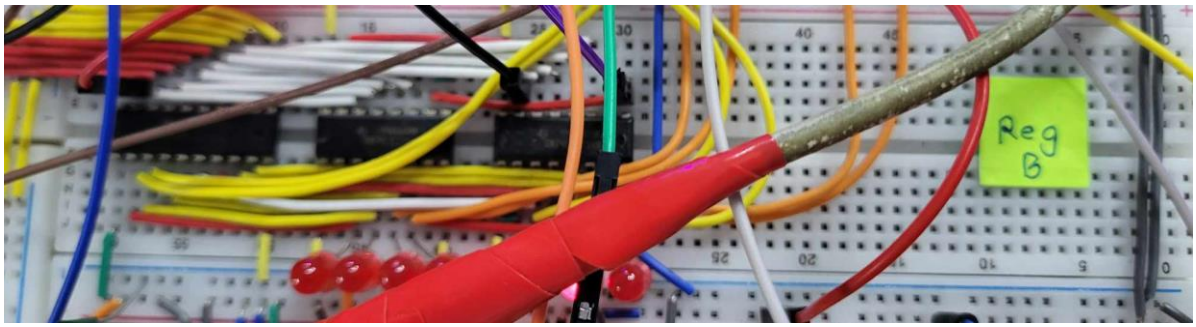
Our clock is adjustable-speed (from less than 1Hz to a few hundred Hz). The clock can also be put into a manual mode where we push a button to advance each clock cycle. This will be a really useful feature for debugging the computer too.

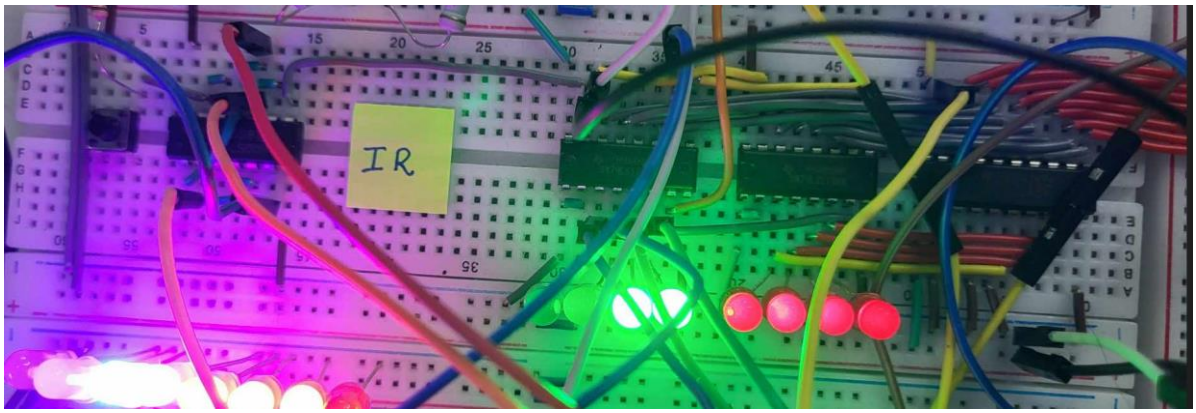


Register:

In our simple breadboard CPU, we'll build three 8-bit registers: A, B and an Instruction Register. The A and B registers are general-purpose registers. IR works similarly, but we'll only use it for storing the current instruction that's being executed. Here 74LS245 is used which is an 8-way in-phase three-state bidirectional bus transceiver that can transmit data in both directions. 74LS245 also has a two-way three-state function, which can output or input data and allows us to control the data flow.







Random Access Memory (RAM) module:

The RAM stores the program the computer is executing as well as any data the program needs. In this project , we have used 4-bit addresses which means it has 16 bytes of RAM, limiting the size and complexity of programs it can run.Each byte(memory address) contains 8 bit data. So total bit is (16*8) or 128 bits. For this ,we have implemented it using two 74LS189 IC.

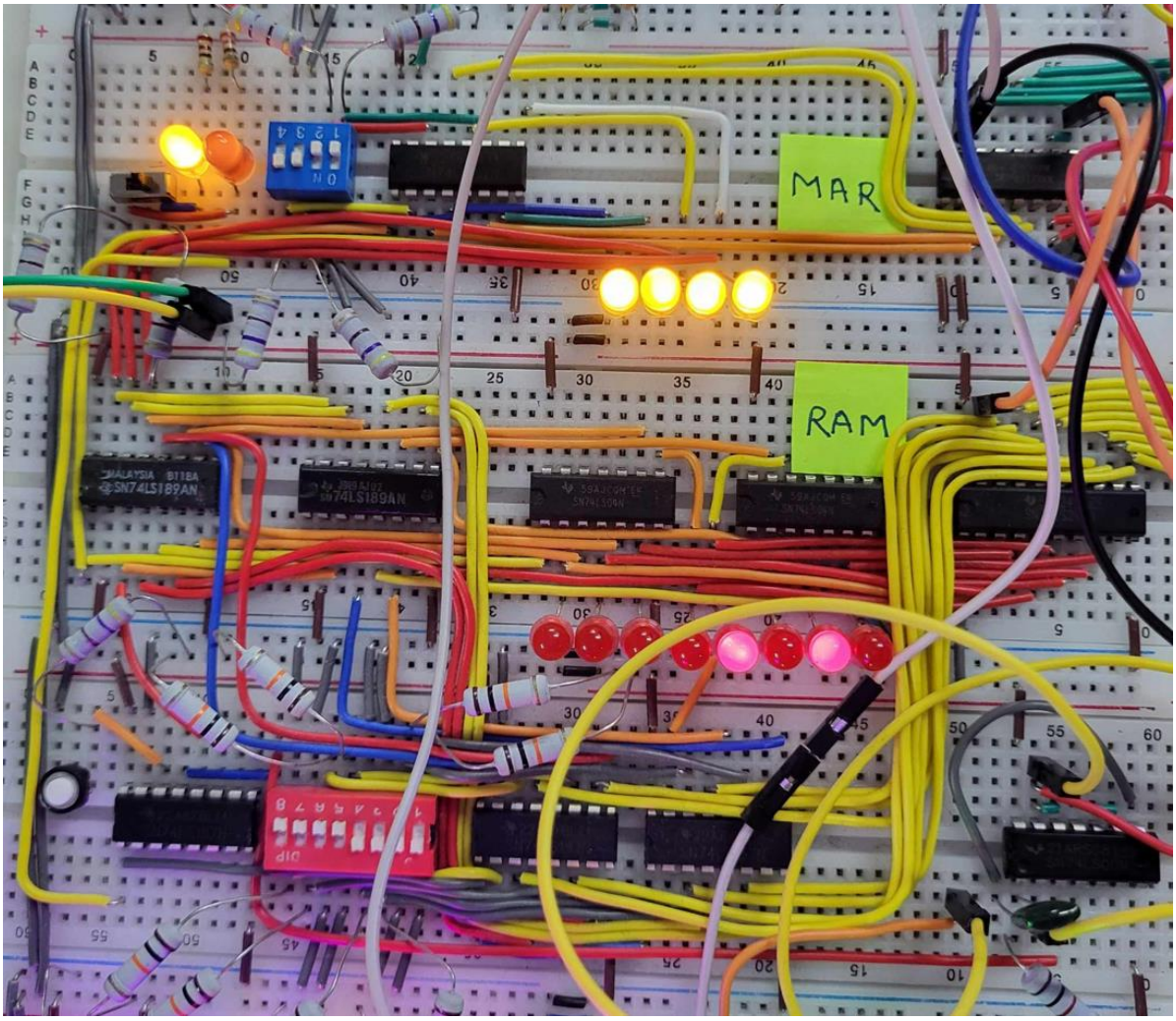
Memory Address Register (MAR) module:

The purpose of MAR is to store the address bit of RAM. It has two modes-

- Program mode : This mode is for controlling the RAM address by manual switching.

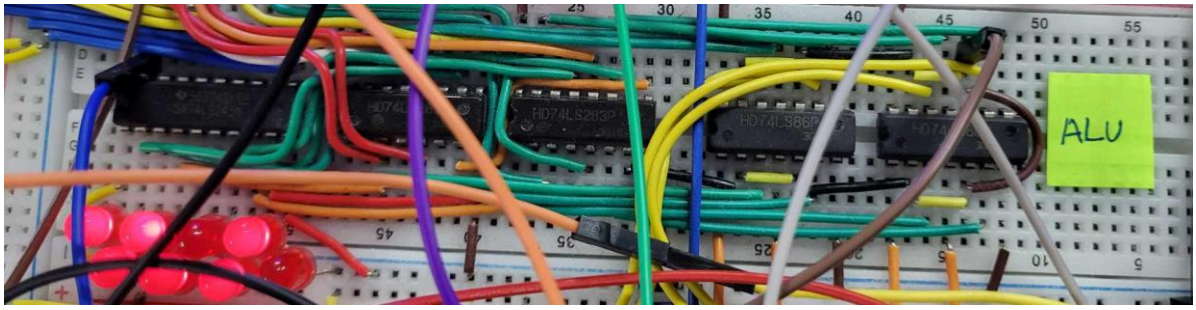
- Run mode : This mode is for addressing RAM what MAR contains and ensuring the connection between RAM and BUS as well.

IC 74LS157 has been used for selecting the mode of MAR & IC74LS173 mux for reading data from BUS.



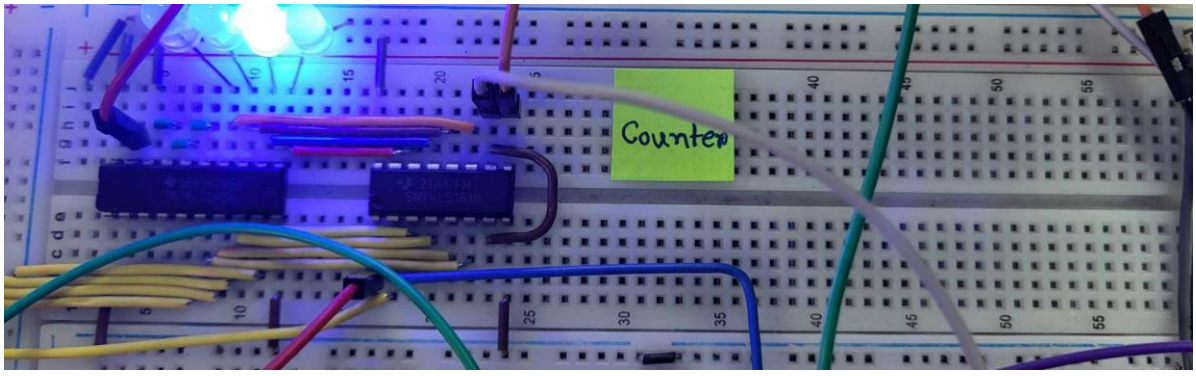
Arithmetic Logic Unit (ALU):

The arithmetic logic unit (ALU) , part of a CPU is usually capable of performing various arithmetic, bitwise, and comparison operations on binary numbers. In our project, the ALU will be able to add and subtract. It will be connected to the A and B registers and outputs will be either the sum of A+B or the difference of A-B.



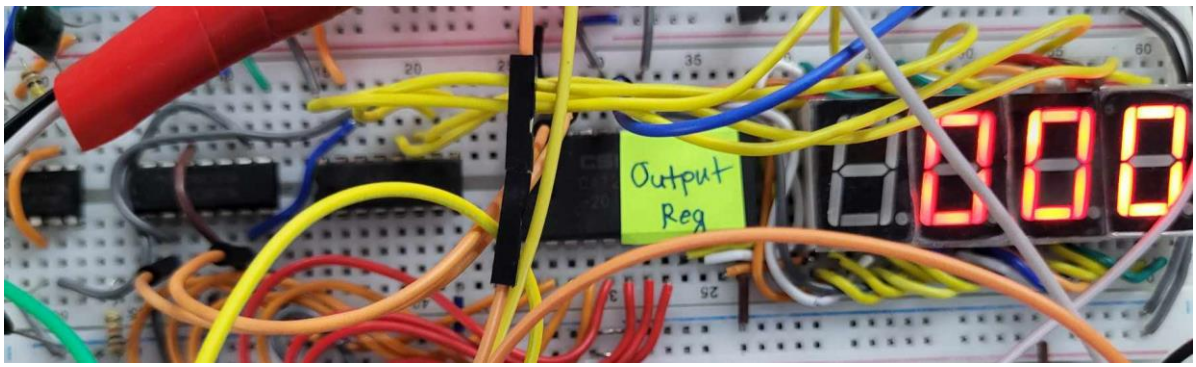
Program Counter:

Programs are stored in the computer as a list of instructions in specific addresses. Programs are sequentially collected and implemented starting from address-01 (Binary 0001). The Program Counter will keep count of this address. We shall be using 4-bit counter since the memory that we have built has 4 bit address. The value stored by the counter indicates the address of the next instruction to be executed.



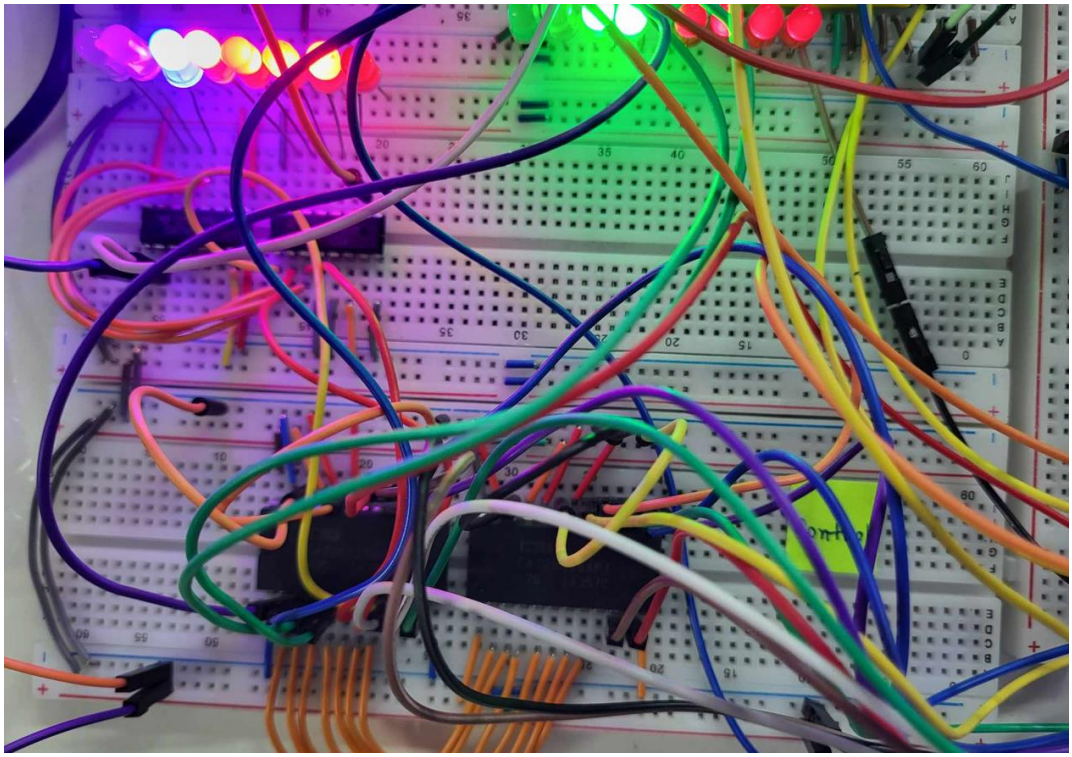
Output Register:

For the purpose of displaying the output of our project , instead of relying on binary outputs displayed by led blinking, we would like to convert them into a decimal output. We will use three seven segment displays to display the output since 8-bit binary number correspond to a maximum a decimal number of the hundredth digit



CPU Control Logic:

The control logic is the heart of the CPU. Each of the components has some control signal going into it. We have to build some logics to turn on appropriate control signals in every step of a mathematical operation. Each of the instruction is made up of several different steps. Each of this step is called micro-instruction. Control logic part consists of mainly coupled 28C16EEPROMs, 74LS161 4-bit binary counter, 74LS138 3to8 line decoder. EEPROMs will take instructions from instruction resister as well as step counter. EEPROMs are programmed with micro-codes that are based on the instructions bit sequence and step counter bit sequence. Then it sets appropriate bits to turn on appropriate control signals. EEPROM can be manually programed. We can also use Arduino based EEPROM programmer which will make it little bit easier for us to add more instruction.



4.2 Data Analysis

Using the computer we build we did some mathematical problems. At first we did addition operation of two numbers

Problem1: 28+14 =?

We write the following microcode to solve the problem and show the result in the output display—

[P. T. O]

Memory Address	Opcode Load
0000	0001 1110 0001(Load) 1110(indicates Value will be loaded at address 14)
0001	0010 1111 0010(ADD) 1111(Value will be loaded at address 14)
0010	1110 0000 1110(OUT)
0011	1111 0000 1111(means Execution)
1110	0001 1100 (Value 28 loaded at A register)
1111	00001110 (Value 14 loaded at B register)

Output display shows output 42

Problem2: 177-205=?

Memory Address	Opcode Load
0000	0001 1110 0001(Load) 1110(indicates Value will be loaded at address 14)
0001	0011 1111 0011(SUB) 1111(Value will be loaded at address 14)
0010	1110 0000 1110(OUT)
0011	1111 0000 1111(means Execution)
1110	1011 0001 (Value 177 loaded at A register)
1111	1100 1101 (Value 205 loaded at B register)

Output display shows output -28

4.3 Results

Our computer gives result of 28+14=42 and 177-205=-28

5 Design Analysis and Evaluation

5.1 Novelty

Our project is novel in the sense it will help a complete beginner in computer science and architecture to have an easy insight to computer architecture and computer assembly language. It also encompasses the entirety of the syllabus on Digital Logic Design and has a wide variation on the use of ICs and chips. So, our project is indeed a great learning and teaching tool.

5.2 Design Considerations

5.2.1 Considerations to public health and safety

This project is quite safe to public health and safety. The project is stable and easy to use so there is negligible chance for any accident to occur.

5.2.2 Considerations to environment

Our project can also be used to render service to the environment. 8-bit microprocessors with low power features are widely researched today due to high demand of green electronics with portable devices.

5.2.3 Considerations to cultural and societal needs

These project can be a great learning and training tool for beginners in computer science and computer architecture.
Small scale household applications like remote, toys and simple sensors do not require lots of processing power and hence they can be functioned using 8-bit microprocessors

5.3 Limitations of Tools

While implementing our project we faced problems as some of the ICs turned out to be damaged. Also, we got unstable outputs due to using combination of HC and LS ICs. We managed to solve this problem by replacing almost all the HC ICs with LS ones. Furthermore, we faced power supply issues. Powering the whole circuit using only one power supplies didn't suffice as there had been considerable amount of voltage drops across the different modules. We managed to solve this problem to some extent by using at least four power supplies. Despite all these, we are still getting erroneous output in some cases. So, more work needs to be done to ensure stability of our project.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

ID 1906175 (Nayem Hasan) prepared the A,B and instruction registers.

ID 1906176 (Tanvir Rahman) worked on the ALU and Program Counter. ID 1906179 (Samiul Hossain) worked on the RAM and Clock.

ID 1906190 (Tousif Ansari) worked on output register and controller circuit

6.2 Mode of TeamWork

We have started our project work from midbreak. We had to sit for minimum 10 times. So mainly our mode was offline based. We also faced some difficulties regarding the IC's. But later , with frequent group sitting, we were able to overcome the issues.

7 Communication

7.1 Executive Summary

A SAP computer is a simple computer architecture designed for educational purposes. It consists of the following components:

- A clock that controls the timing of the operations

- A program counter that keeps track of the current instruction

- A memory address register that holds the address of the instruction or data

- A memory that stores the instructions and data

- A instruction register that holds the instruction to be executed

- A controller-sequencer that generates the control signals for each step

- A accumulator that performs arithmetic and logic operations

- A output register that displays the result

The SAP computer can execute a limited set of instructions, such as load, add, subtract, and output. Each instruction is 8 bits long and consists of a 4-bit opcode and a 4-bit operand. The SAP computer follows a fetch-decode-execute cycle to process each instruction.

User Manual

To do any operation using the computer the user should follow the instruction below. At first user has to select the program mode. Then using the input switchs one should save the instructions and data input in the memory. After that user has to select the run mode and start clock module. Then the given instruction will be implemented in the next 6 clock cycle.

8 Project Management and Cost Analysis

Cost analysis:

- Breadboard (15 pcs) ------ Tk. 2385
- Hookup wire ----------Tk. 487
- Resistors of different values---------Tk. 125
- Variable resistor pot (5 pcs) ---------- Tk. 20
- Capacitors of different values---------- Tk. 100
- NE-555 timer (6 pcs)----------Tk. 60
- 7402 NOR gate (3 pcs)----------Tk. 75
- 7404 PDIP 14 (7 pcs) ---------- Tk. 175
- 7408 AND gate (5 pcs) ---------- Tk. 125
- LEDs of various colors ----------Tk. 95
- Tactile Pushbutton switch ---------- Tk. 20
- 7400 NAND gate (4 pcs) ---------- Tk 114
- 7432 OR gate (2 pcs) ----------Tk. 58
- 74107 Dual JK FlipFlop----------Tk. 46
- 7476 JK Flipflop ---------- Tk. 45
- 7486 XOR Gate (3 pcs) ---------- Tk 70
- 74138 Decoder (2 pcs) ---------- Tk. 63
- 74157 Multiplexer IC (6 pcs) ---------- Tk 180
- 74161 Synchronous Counter (3 pcs) ---------- Tk. 111
- 74273 D Flip Flop (2 pcs) ---------- Tk. 123
- 74283 Adder (3 pcs) ---------- Tk 109
- SPDT Slide Switch (4 pcs)---------- Tk 22
- Dip Switch - 8 Position (2 pcs) ---------- Tk 50
- Dip Switch - 4 Position (2 pcs) ---------- Tk 41
- 7 Segment Display CC - 1 Digit (5 pcs) ---------- Tk 61
- Jumper Wire Set (8 sets) ---------- Tk 1600
- Female To Female Jumper Wire – Single (10 pcs) ------ Tk 30
- Male To Male Jumper Wire – Single (50 pcs) ---------- Tk 150
- Male To Female Jumper Wire – Single (40 pcs) --------- Tk 120
- 74245 bus transceiver ICs (10 pcs) ---------- Tk 540
- 74139 decoders (2 pcs) ---------- Tk 60
- 74173 D Flip Flops (10 pcs) ---------- Tk 450
- 74189 RAM ICs (4 pcs) ---------- Tk 320
- 74595 shift registers (3 pcs) ---------- Tk 60
- EEPROM (5 pcs of two different models) -------- Tk 2100

Total Cost : Tk 10,190

9 Future Work

This project has scope of expansion in the future. We only implemented SAP-1 but we can easily add Flags register and jump command which will allow our 8-bit computer to execute more complex calculations like multiplication, finding Fibonacci numbers etc.

10 References

https://www.youtube.com/@BenEater

https://eater.net/8bit/kits

Ref Book:Digital Computer Electronics
by Albert Paul Malvino (Author)