Final Lab Report

Hanad-Keysse Mohamed Hassan ID:40299566
Jeremy de Passorio PID:40271747

Submitted on June 7th 2024
Section K

• Introduction

This series of lab experiments aimed to build a fundamental understanding of computer systems at the hardware level. The primary goal was to design, construct, and test a basic digital 4 bit computer system from discrete components, exploring concepts such as memory storage, sequential logic, bus communication, and instruction execution.

The first lab focused on the implementation of a timing signal generator, accompanied by a clock signal in order to later create an orderly loop sequence that would be composed of the instruction process of the computer.

The second lab integrated an arithmetic unit and program counter with a shared bus system. Using 74LS173 registers and a 74LS283 adder, a system was created to increment addresses and execute sequences of instructions, showcasing data flow and arithmetic operations.

The third experiment involved wiring and testing the program memory with a microcontroller as ROM and a decoder. An ATTiny2313A microcontroller stored and fetched instructions, which were decoded and executed by the processor, illustrating instruction decoding and execution.

• Memory

Computer memory stores data in bits. This project utilizes ROM (Read-Only Memory), which cannot be modified, only read. The program counter sequences ROM addresses for instruction retrieval. Program memory holds instruction words with an operation code (op-code) and operand addresses. The ATTiny2313A microcontroller acts as a ROM, providing instructions via address inputs (PD0 to PD3) to the Memory Address Register (MAR). These instructions are output through data pins (PB0 to PB3) and decoded by a 7442 decoder to generate control signals for execution, ensuring precise instruction display on an LED pack. Memory fetches instructions by addressing through the MAR, retrieving data from ROM via the data bus. Decoders translate binary codes into actionable signals, facilitating accurate processor operations. The MCU delivers instructions like Increment A (01111), Increment B (10111), Move AB (11011), Move BA (11101), and NOP (11110). Implementation with data registers enhances operational efficiency.

Data registers act as temporary storage within the processor. This experiment implements two 74LS173 4-bit registers, A and B, for operands and results, improving processing speed by minimizing memory access delays. These registers' tri-state outputs enable efficient bus interaction. In this setup, registers A and B connect their inputs and outputs to the bus, with output enable and clock pins tied to VCC to prevent unintended data transmission.

The MAR, implemented with a 74LS173 register, continually provides the RAM address for instruction execution. It captures data from the bus when the program counter outputs, ensuring accurate memory access.

INDEX	BINARY INSTRUCTION	INSTRUCTION
0	**11110**	**NOP**
1	**01111**	**INC A**
2	**01111**	**INC A**
3	**01111**	**INC A**
4	**11011**	**MOV AB**
5	**10111**	**INC B**
6	**10111**	**INC B**
7	**11110**	**NOP**
8	**11101**	**MOV BA**

9	**10111**	**INC B**
10	**01111**	**INC A**
11	**10111**	**INC B**
12	**01111**	**INC A**
13	**11011**	**MOV AB**
14	**11101**	**MOV BA**
15	**01111**	**INC A**

It is possible to find the timing diagram with the 5 instructions below :

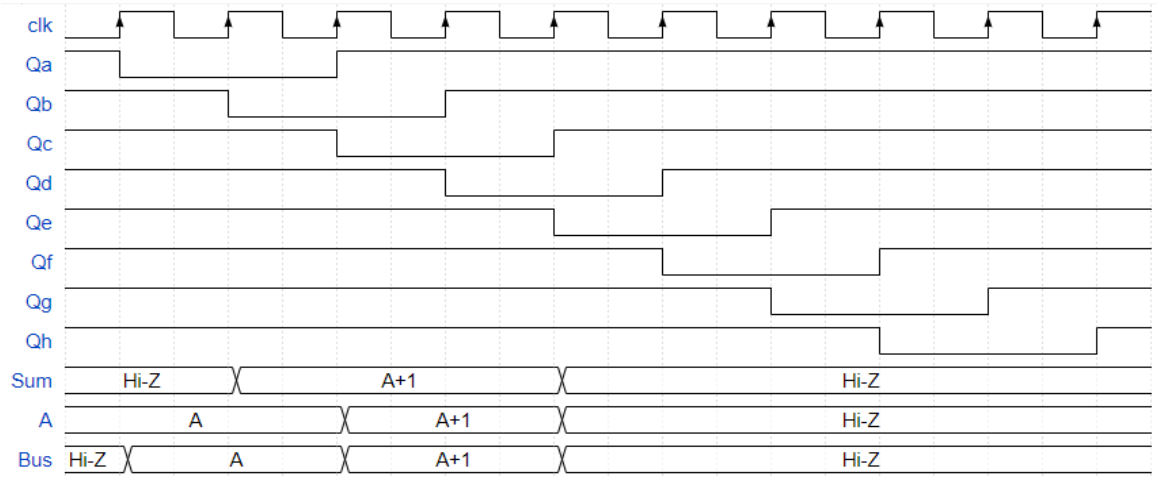

Figure 1 : INC A

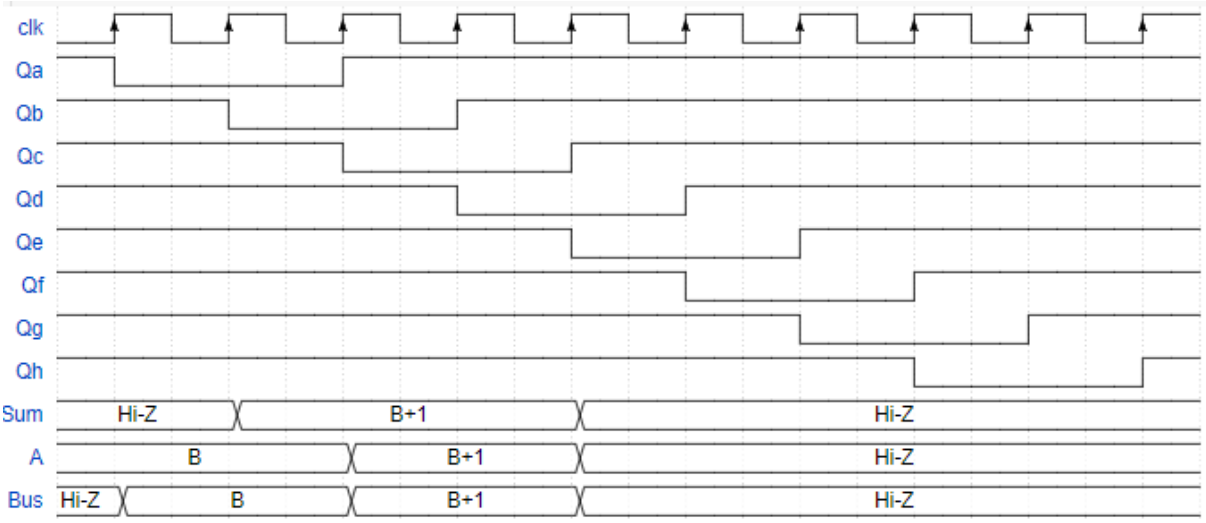

Figure 2 : INC B

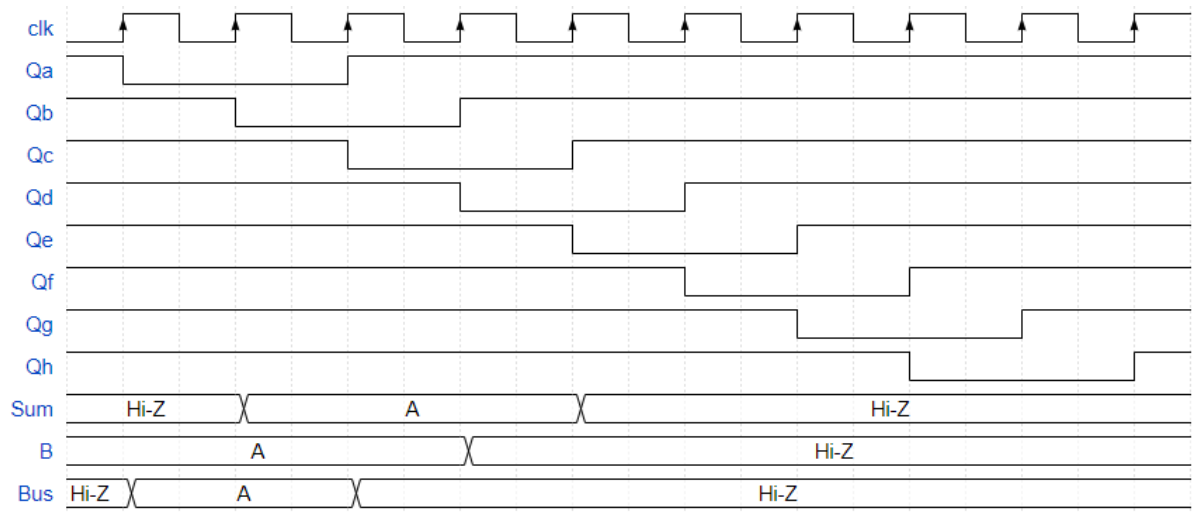

Figure 3 : MOV AB

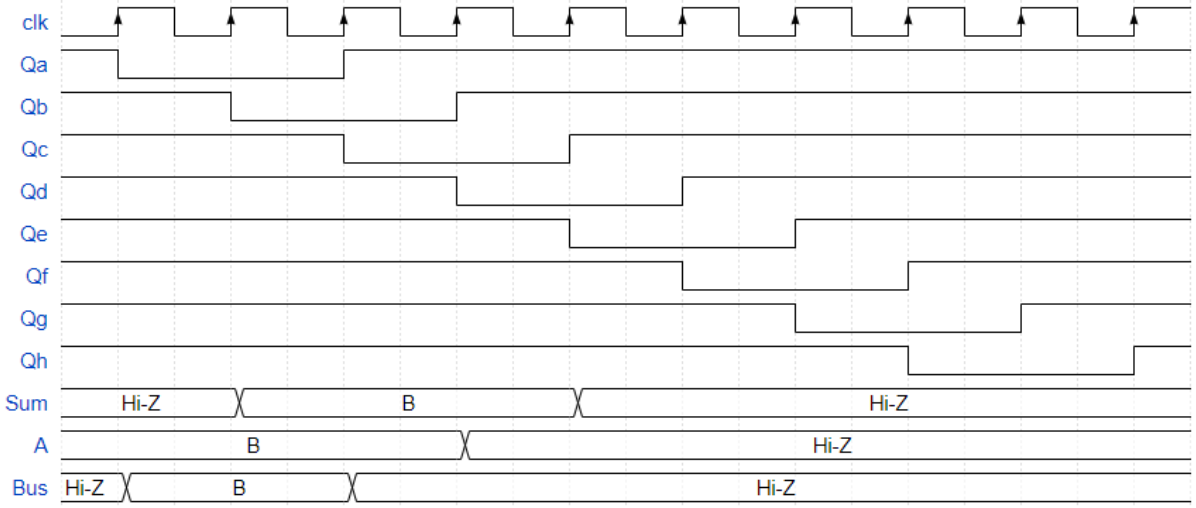

Figure 4 : MOV BA

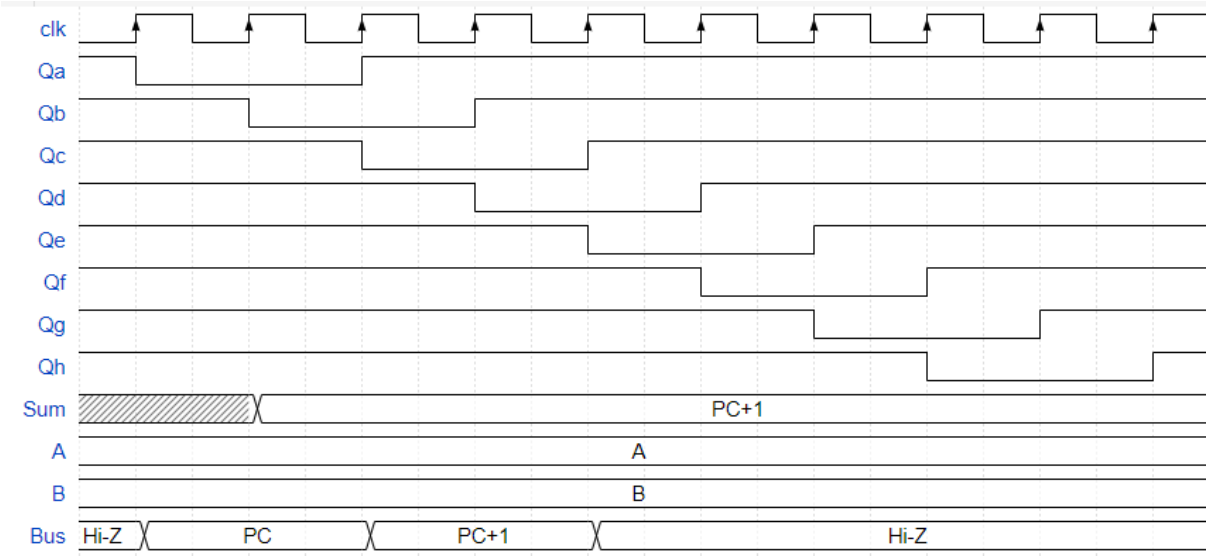

Figure 5 : NOP

• Timing Signal Generator

The 555 timer IC contains an S-R flip flop. It is used in astable mode, which indicates that Its signal alternates between 0 and 1. The flip flop It switches according to the voltage. It is connected to a circuit with resistors and capacitors. The 74LS164 SIPO shift register IC has two inputs but the same one is used for the sake of this experiment. It has 8 outputs : Qa to Qh in order. As the name says, the shift register's job is to shift bits. After each clock cycle, It moves the value (bits) of the input from Qa to Qh. As every output shifts according to a clock, they all act as timing signals. A pulse like effect is required in this project. To do so, a feedback signal has to be made. The chosen pins are Qb, Qd, Qf and Qg. They are all fed into a NAND gate whose output serves as the inputs for the shift register. When all 4 inputs of the NAND gate are 1, the output (input A and B of the register) is 0. After the 8th clock cycles and the 9th clock cycle, QB, QD, QF and QG are all 1s. They are fed into the NAND gate, which generates a 0 and fed back into the register . In these two instances, the shift register's input is 0 and it moves from Qa all the way to Qh. This creates the aforementioned pulse, (wave pattern, 2 0s in a sea of 1s) when all the Timing signals are connected to an LED pack.

Here is the detailed circuit illustrated in Schematic 1:

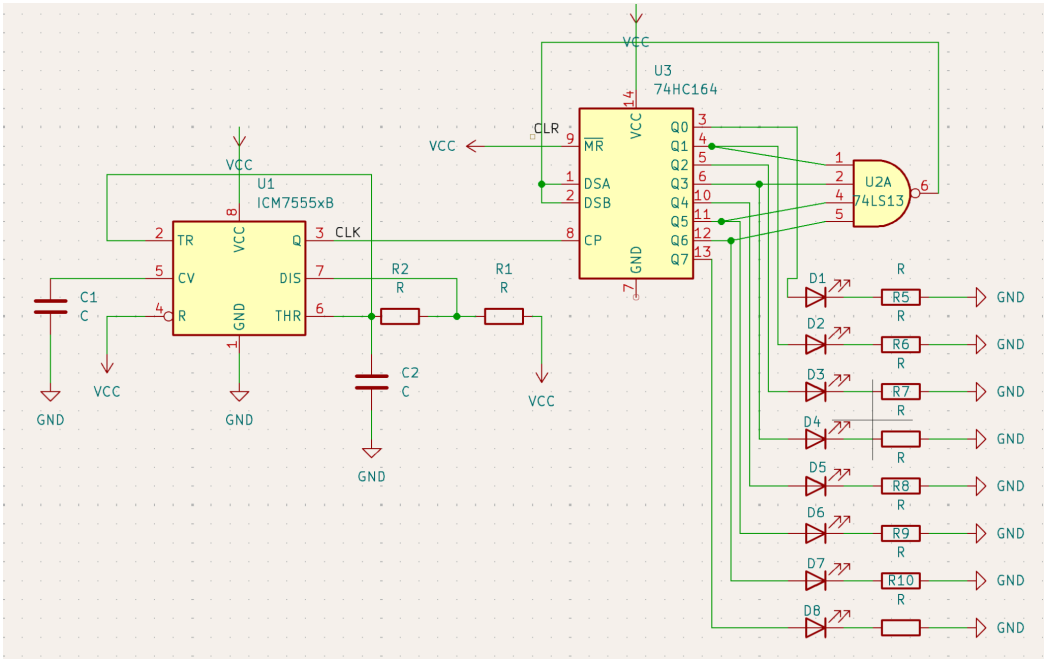

Figure 6 : TSG circuit

• The Data Bus

A bus consists of a bunch of wires that are wired together in order to connect multiple devices. The wires are attached next to each other in order to increase the connection between adjacent rows of the breadboard and utilize the maximum length by going over the gaps in the breadboard. The devices that are connected are 7 ICs : 6 registers (74LS173) and 1 adder (74LS283).

• Program Counter

A program counter holds the address of the instruction to be executed by the processor. Its incrementing is done through the bus during the fetch phase of the instruction in order to prepare the next instruction. The register that serves as the PC is the 74LS173 register, a 4 bit register. Since the PC holds 4 address bits, there can be a total of 16 different instructions.connected to ground as they are write-enable inputs. The master reset pin (pin 15) is tied to GND to prevent the register from resetting. The register has a tri-state buffer. Its purpose is to prevent multiple inputs from being present on the data bus.

• Arithmetic unit

The arithmetic unit's purpose is to help increment the PC. As previously mentioned, it has to cycle through instructions. Therefore, after the PC gets its address from the bus, it holds it and feeds it back onto the bus. It goes into one of the components of the AU, the 74LS283 adder. The adder adds 1 to the 4-bit address and outputs it into the sum registers another 74LS173 register which has a tri-state buffer. The sum register feeds the incremented value back into the bus and it is then taken by the PC. This repeats throughout the whole 16 instructions. There is also a mirror register which quite literally mirrors the sum register, except its output is shown on an LED pack to see the instruction address cycle. In addition to the tri-state buffers, the timing signals are used to order this process so that only 1 register outputs bits onto the bus. T0 triggers the PC (BUS ←PC), T1 triggers the sum registers input (Sum register ← adder), T2 triggers the output of the sum register as well as the input of the mirror register (Bus ← Sum register & mirror register ← BUS), and finally T3 triggers the input of the PC which takes the incremented address (PC ← BUS). The timing signals inputs are inverted to accommodate for the fact that the 74LS173 registers are positive-edge triggered as the computer is negative-edge triggered. Here is the circuit schematic illustrating the previously mentioned components of the 4-bit computer:

:

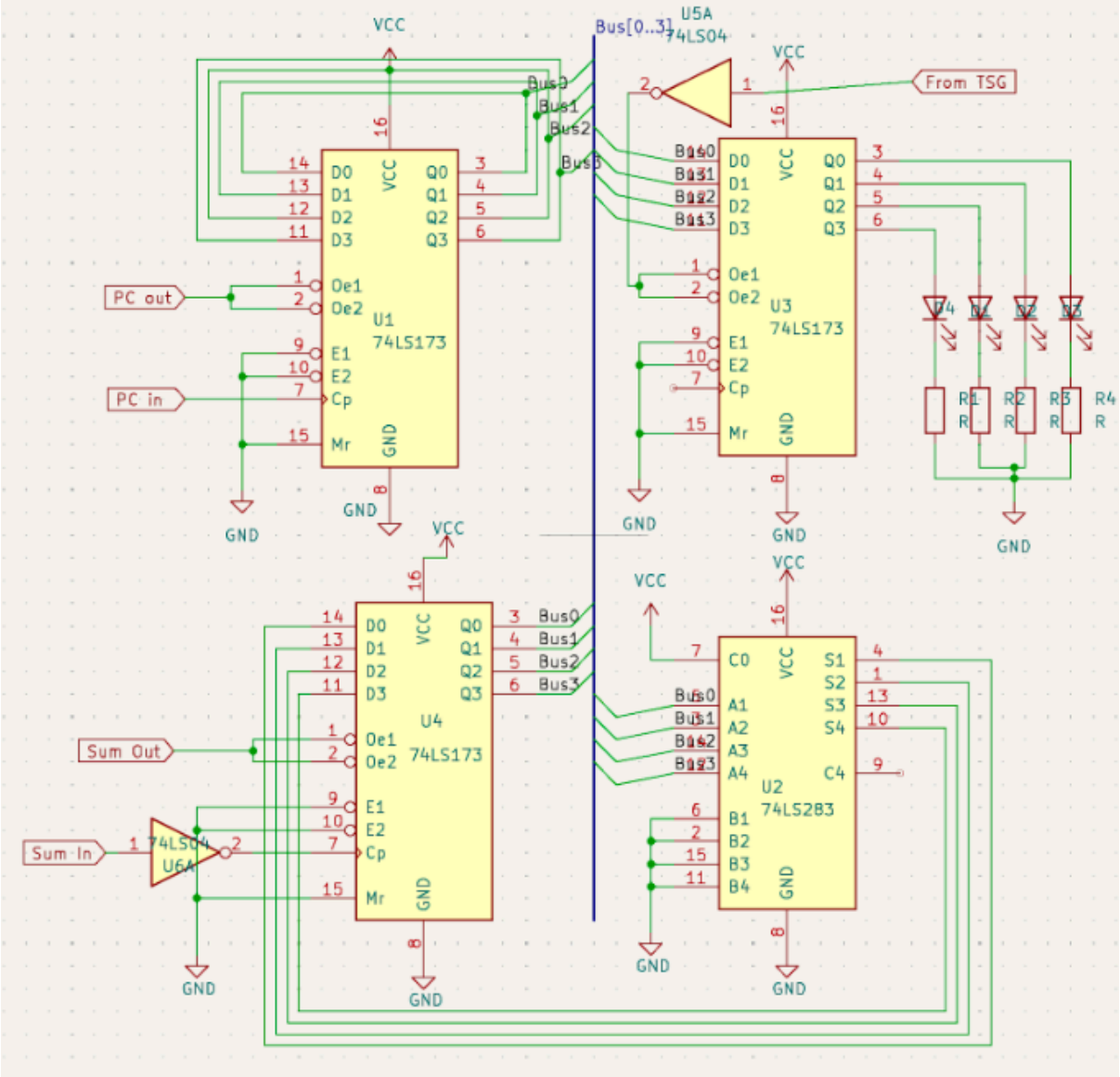

Figure 7 : PC + Arithmetic unit schematic

• Control Signal Generator

Four timing signals were used up to this point to allow the PC to go through the instructions and the instruction addresses. Now, T4 to T7 will be used for the execution part. The instructions still need to be executed. Therefore, a control signal generator will take the instructions of the memory that were decoded and (they are fed into the CSG) which, in turn, gives control signals to the appropriate registers in order to make the execution of the instructions possible.

Here are all of the instructions and how they work :

1- INC A : The data register A feeds its content into the bus, it is incremented by the adder (A=A+1) and then stored in the sum register, and it is fed back into the bus so that data register A picks it back up.
2- INC B : It is the exact same process as the incrementation of A, except this time it is B = B+1, it originates from data register B, and it is stored back into data register B.
3- MOV AB : This instruction's execution only consists of data register A putting its information onto the bus and data register B storing it
4- MOV BA : It is the same process as MOV AB, data register B feeds its data onto the bus and data register A captures it and holds it.
5- NOP : No operation. Acts as a delay between two instruction executions.

As such, combinational logic is required to implement this. AND, OR and inverters are used.
For data register A, its clock input becomes : (T7 + I0) · (T5 + I3), Its output enable becomes : (T4 + I0) · (T4 + I2).
For data register B, the clock becomes : (T7 + I1) · (T5 + I2), The output enable becomes : (T4 + I1) · (T4 + I3).
For the sum register, the clock becomes : (T5 + I0) · (T5 + I1) · T1, The output enable becomes : (T6+I0) · (T6 + I1) · T2. The NOT IC intervenes in the inversion of the clocks which are negative-edged but fed into registers that are positive-edged.

• Overview of the Computer

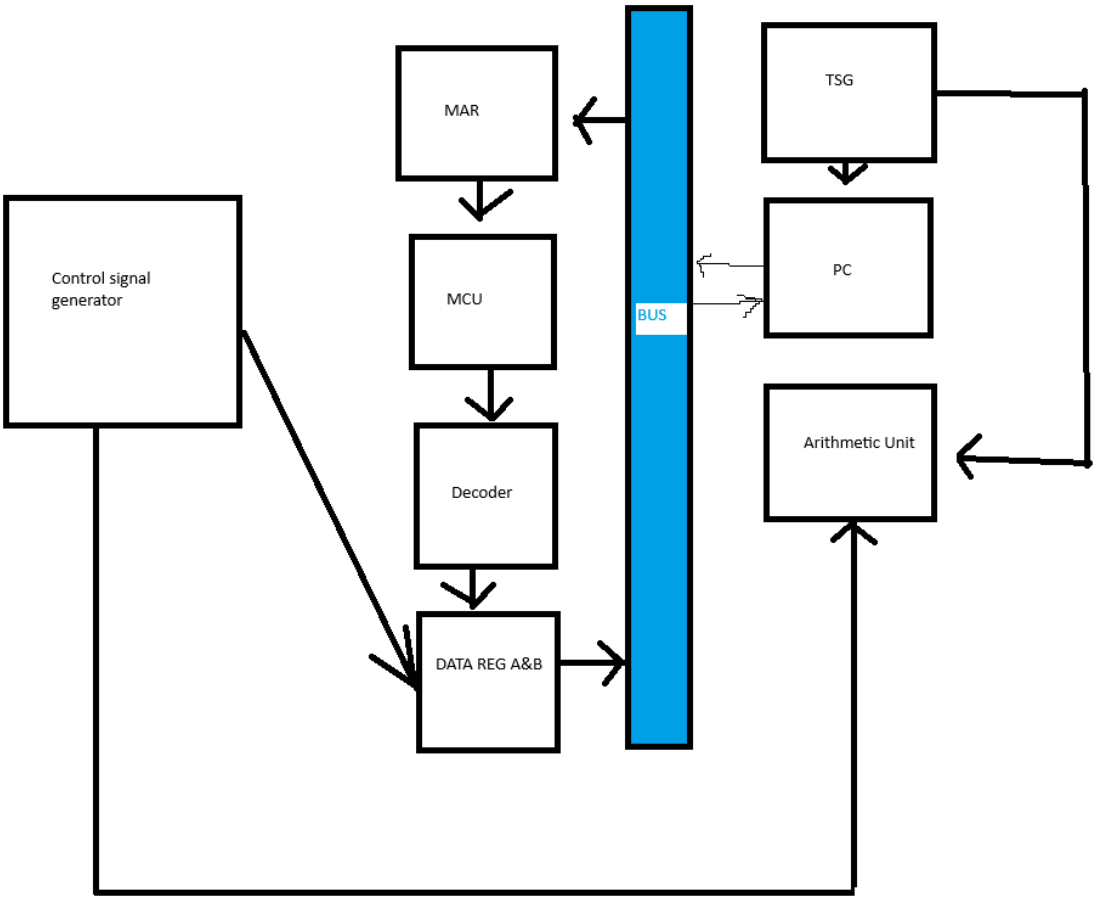

Figure 10 : Block diagram

• Operation of the Computer

The program counter, implemented using a 4-bit register, holds the address of the next instruction to be executed. This address is incremented by the arithmetic unit, consisting of a 4-bit adder and a sum register, and then fed back into the bus. It is also picked up by the MAR which communicates with the ROM (MCU to retrieve the instruction pointed by the program counter). The microcontroller, programmed as ROM, provides the predefined instructions which are decoded and executed by the processor. Two data registers, A&B respectively, will hold the data that is being manipulated during the multiple instruction cycles. A control signal allows the computer to execute its instructions. The timing signals generated by a 555 timer ensure sequential and synchronized operations of the components, enabling the processor to correctly read and execute instructions. The use of tri-state buffers prevents conflicts on the bus by ensuring only one device drives the bus at any time, facilitating smooth data flow and accurate instruction execution. Essentially, the computer cycles through five main operations which are : Increment A (01111), Increment B (10111), Move AB (11011), Move BA (11101), and NOP (11110). The main difference between this computer

and a regular 32-64 bit computer is that it is very limited in what it can do as the amount of instructions is very limited. There were only five instructions coded into the MCU.

Below is the schematic of the 4-bit computer

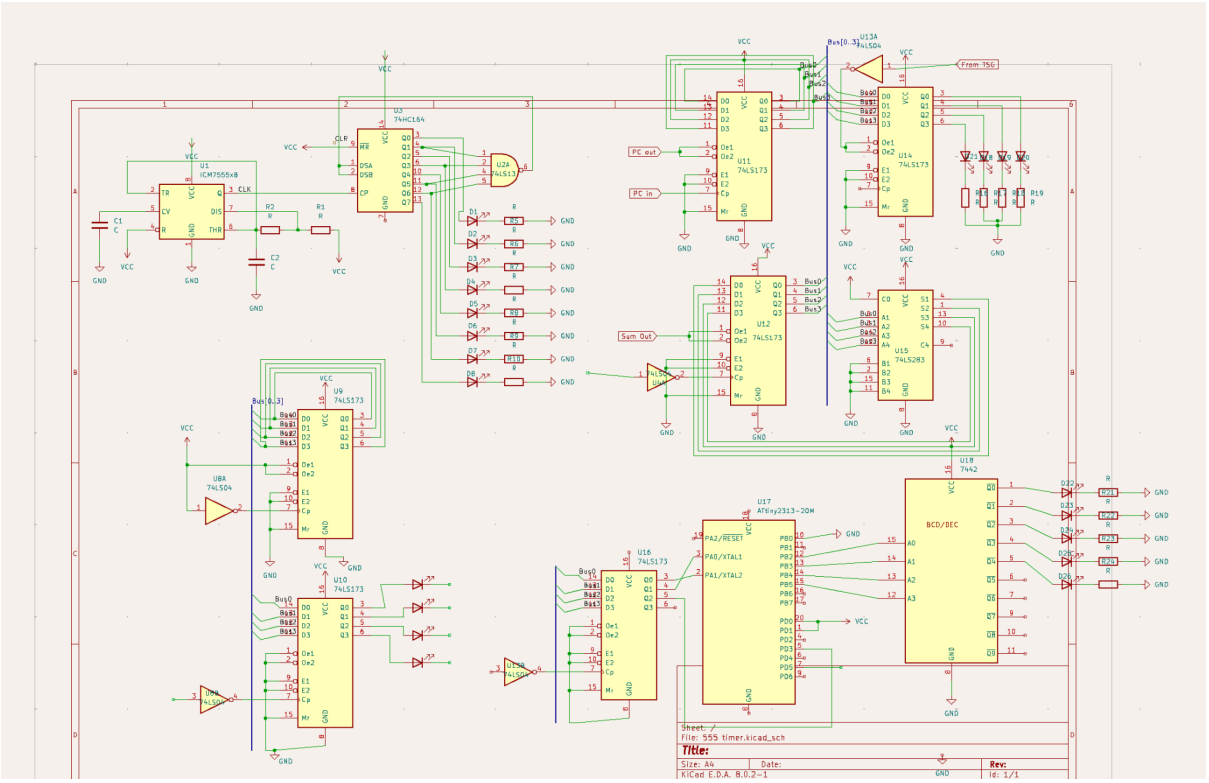

Figure 11: Schematic of computer