8-Bit Processor

25.04.2024

IMT2022 Batch

Aryan Mishra - IMT2022502

Harshavardhan R - IMT2022515

Nupur Patil - IMT2022520

Sreyas Janamanchi - IMT2022554

Pradyumna G - IMT2022555

Ishan Jha - IMT2022562

Overview

The aim of this document is to explain the working of the 8-bit processor designed by our team for the analog circuits lab project for the academic year 2023/24. The processor consists of multiple individual elements that run together simultaneously in order for the processor to work. The different individual parts are as follows :

1. Clock
2. Random Access Memory (RAM)
3. Program Counter (PC)
4. Arithmetic Logical Unit (ALU)
5. Control Signals

In this document, each individual component and its working shall be explained, along with corresponding videos doing the same.

The link of the video explaining each component is attached below (please copy the link and paste it in your browser) :

https://iiitbac-my.sharepoint.com/:v:/g/personal/nupur_patil_iiitb_ac_in/EQMi3wg4ZO1OktfjWhwaSDABxe7RM6yFdi1LDFtFQ3p6VA?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=22hJ0a

Clock

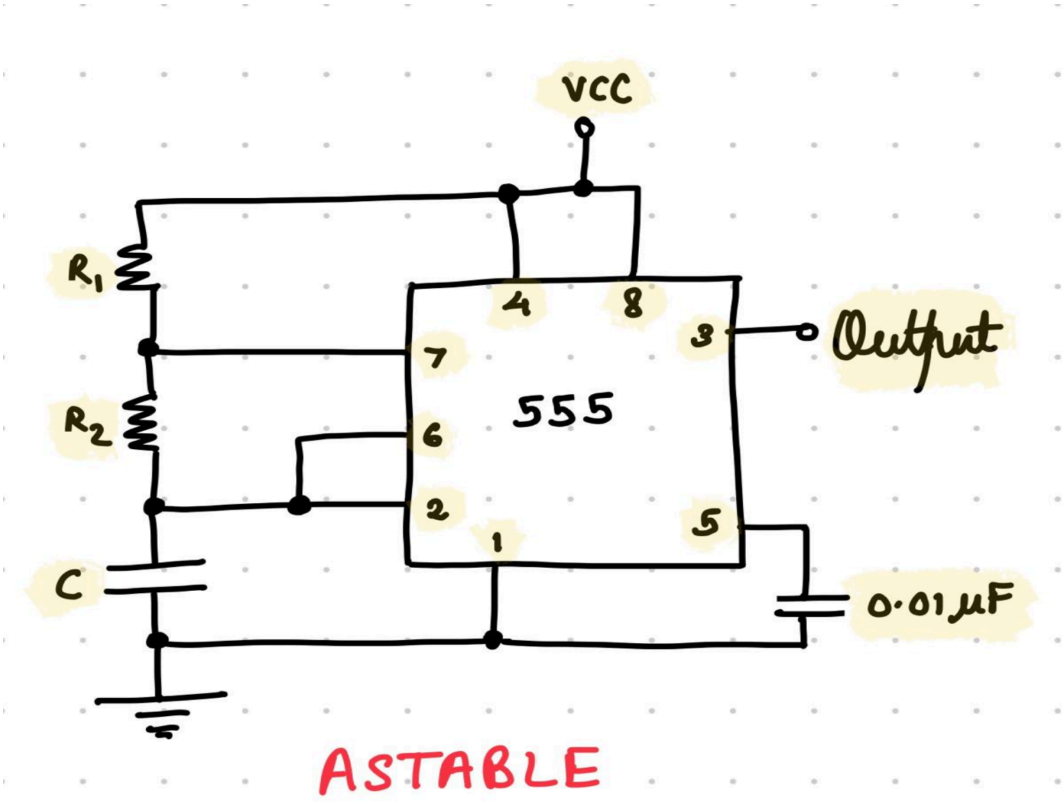
In this project, we use the 555-timer IC in order to design a clock which runs all the components of the project. We have used 2 555-time ICs to design two types of clocks. An “Astable” and a “Monostable” clock. The difference between the two clocks are as follows:

1. Astable Clock - This type of clock automatically keeps going to the next cycle without any human intervention. It is used to run the entire processor automatically as it keeps going to the next cycle by itself.

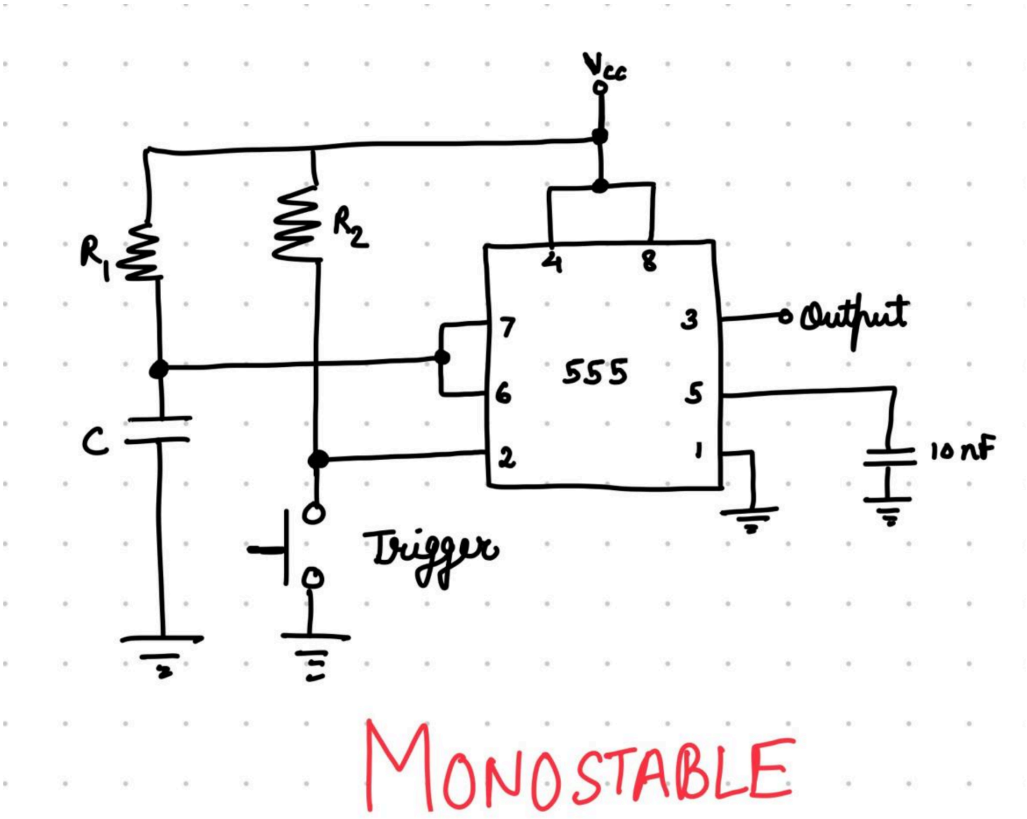
2. Monostable Clock - This type of clock only goes to the next cycle on human intervention, i.e, when a button is pressed only then does the clock enter into the next cycle.

The circuit diagrams of the clock are as follows :

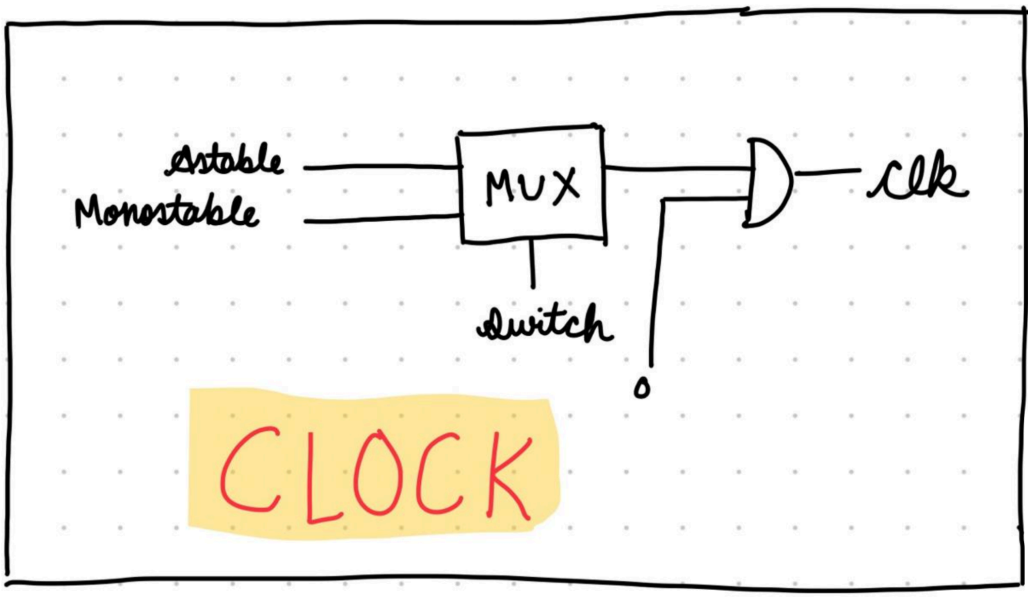
1. Astable Clock :



2. Monostable Clock :



3. Complete Clock :



Program Counter (PC)

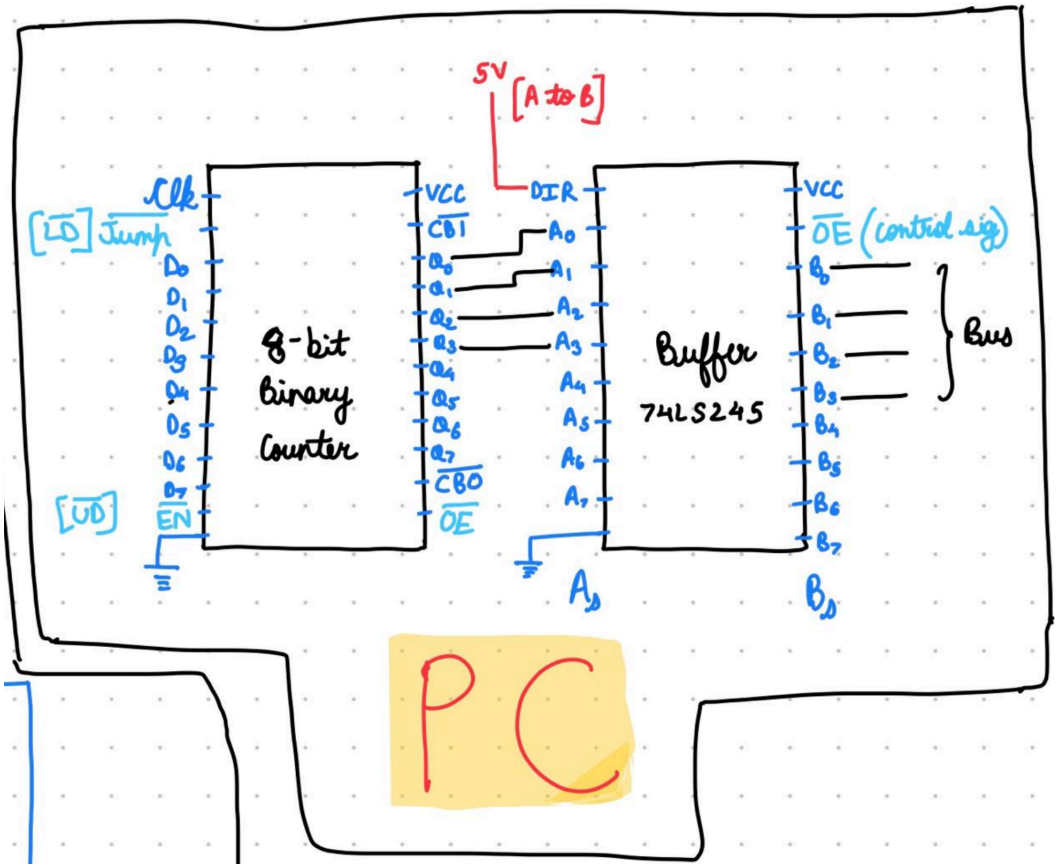
The program counter keeps a track of the next instruction which is to be executed. It stores the address of the next instruction which is to be executed and which is stored in the instruction register.

1. Whenever the clock enters into the next cycle and the count enable (CE) control signal is high, the value of the PC is incremented.

2. When the counter out (CO) control signal is high, the updated PC value is loaded onto the bus.

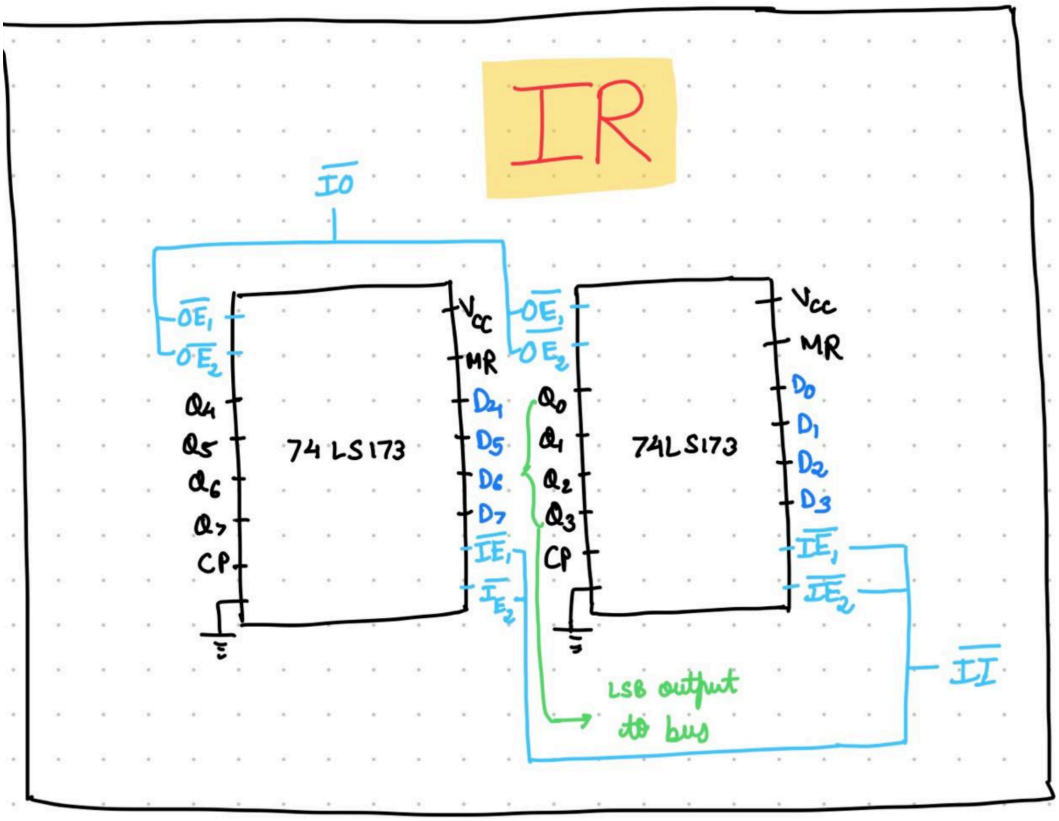
3. The address is picked up from the bus when the control signal named “instruction register in” (II) is high.

The circuit diagram of the program counter is as follows :



Instruction Register (IR)

The instruction register is used to store the instruction which is being currently executed. It loads the 4 least significant bits onto the bus when the control signal named “Instruction Register Output” (IO) is high. The circuit diagram of the instruction register is as follows :



Keep in mind, the pin labeled as Q3 in the IC 74LS173 on the right hand side acts as the signal RA (register address) in the ALU.

Random Access Memory (RAM)

The RAM (Random Access Memory) is used to store different values at different memory locations stored in the IC. The RAM loads data/values onto the bus. The RAM consists of two parts, MAR (Memory Access Register) and the memory.

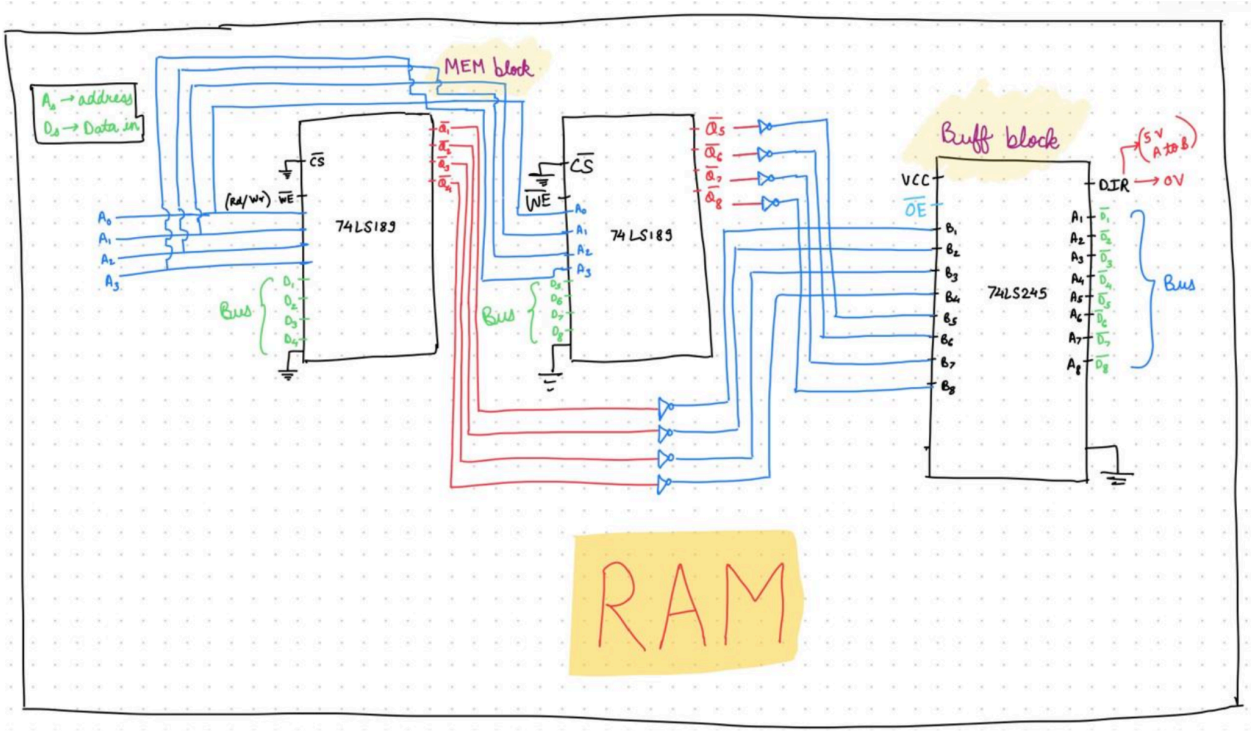
1. The MAR, loads the desired address from the bus when the control signal MARIN (MI) is high. This data then serves as the input for the memory address for the RAM chip.

2. The RAM chip loads data from the bus when the control signal RAMIN (RI) is high.

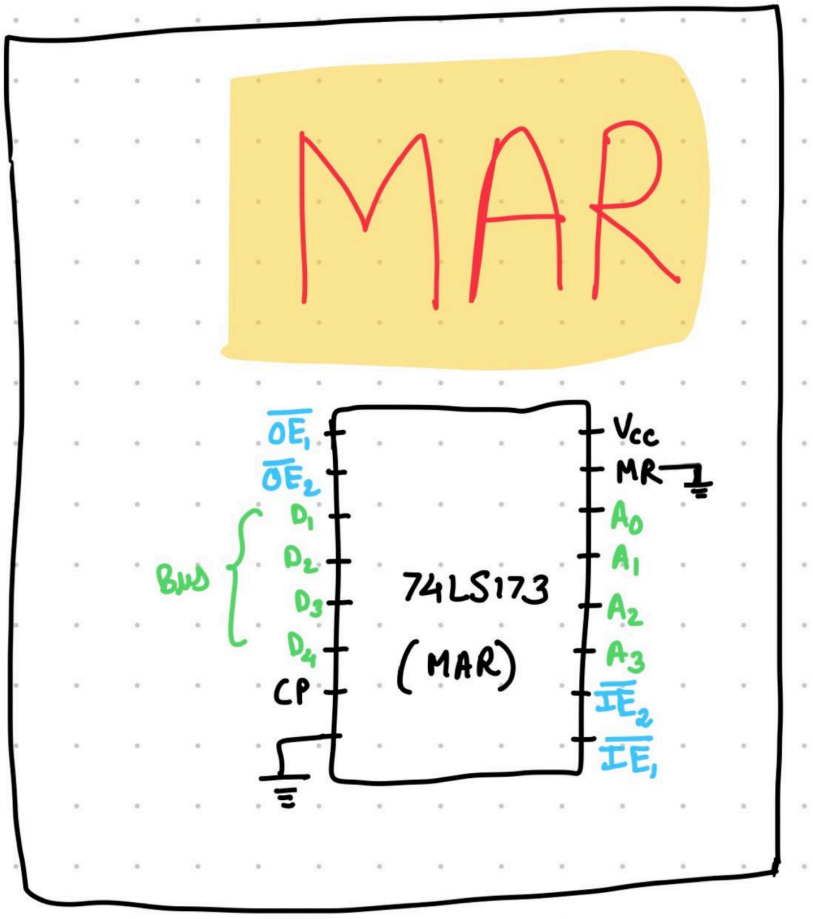
3. The memory loads data into the bus only when the control signal RAMOUT (RO) is high.

The circuit diagram of the RAM section are as follows:

1. RAM Chip :



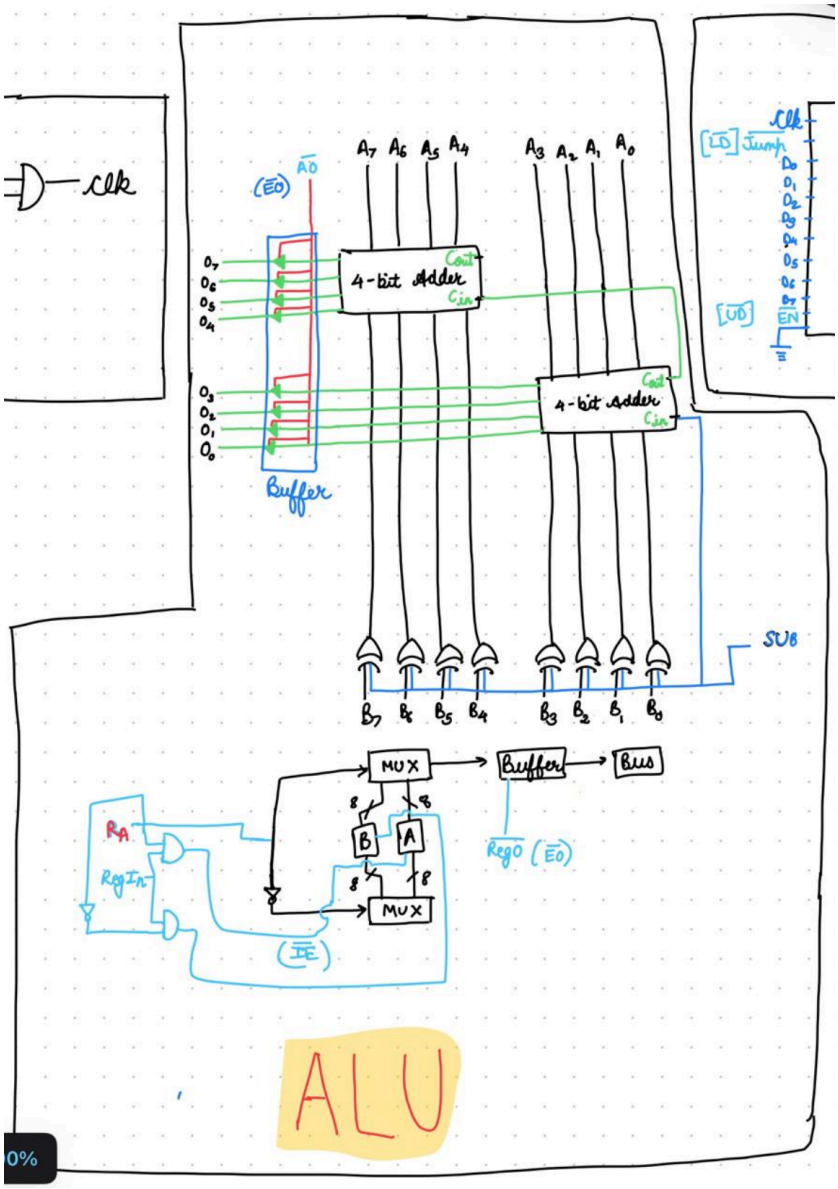
2. MAR (Memory Address Register) :



Arithmetic Logical Unit (ALU)

The arithmetic logical unit, or the ALU, is used to perform the different kinds of operations and generate outputs. These outputs are loaded onto the bus whenever the control signal named “ALU Output” (AO) is high. The input values which are to be processed are stored in registers and this data is loaded into the register when the control signal named “Register In” (RegIN) is high.

There are ***ONLY 2 REGISTERS*** (labeled ‘A’ and ‘B’ in following diagram) in the ALU.



Control Signals

Control signals are signals which are sent to different output interfaces to produce desired outputs.

Sl No	Control Signals	Description	Component Affected
1	MAR IN	Causes the MAR to take address on the bus as input	MAR
2	RI (RAM IN)	Loads data from the bus into a particular address in the RAM	RAM
3	RO (RAM OUT)	Sends data onto the bus from a particular address in the RAM	RAM
4	II (IR IN)	Loads an instruction from the bus onto the IR	IR
5	IO (IR OUT)	Loads the 4 least significant bits of the instruction onto the bus	IR
6	AO (ALU OUT)	Loads the output of an operation from the ALU onto the bus	ALU
7	RegIN (Register In)	Loads data from the bus onto one of the registers in the ALU	ALU
8	RegOUT (Register Out)	Loads data from one of the registers in the ALU onto the bus	ALU
9	S (subtraction)	Helps to generate the 2's complement for subtraction operation	ALU
10	CE (Count Enable)	It enables the binary counter (count enable pin) of the program counter.	PC
11	CO (Count Out)	Loads the address of the current instruction being executed onto the bus	PC
12	J (Jump)	Loads the data available on the bus onto the binary counter of the program counter	PC

The different control signals are pre-programmed into an EEPROM (Electrically Erasable Programmable Read-only Memory). At each address of the EEPROM we store an array of all the control signals being used by the instruction which has the same op-code as the first three bits from the left of that particular address in the EEPROM, while the least significant bit of the address is used to represent the step.

To track which step number of the particular instruction is being executed we use an 8-bit counter in the control unit which counts frm 0-4

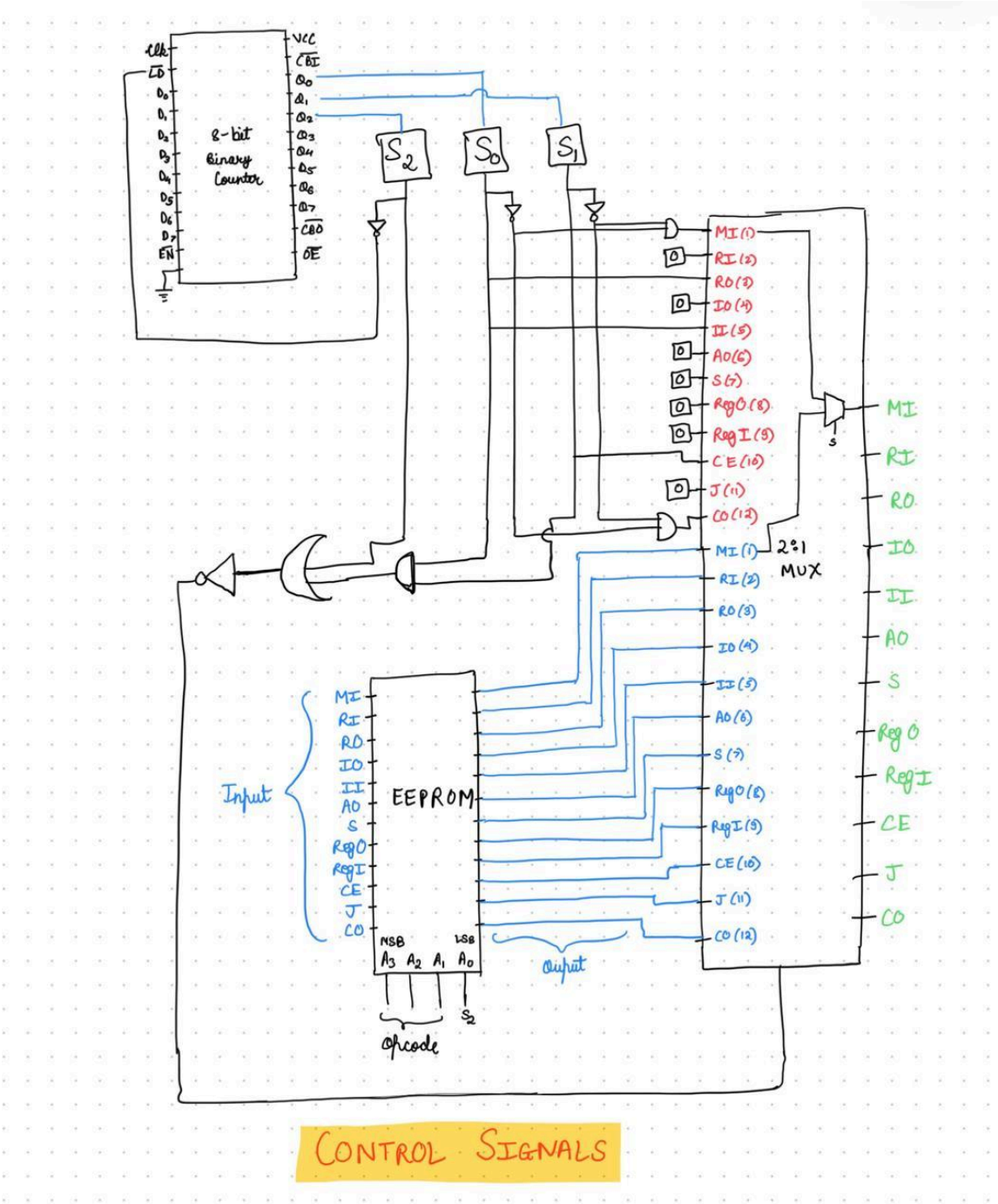
Operation	Step No	MI	RI	RO	IO	II	AO	S	Reg O	Reg I	CE	J	CO
Common	0	1	0	0	0	0	0	0	0	0	0	0	1
	1	0	0	1	0	1	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	1	0	0
Add	3	1	0	0	1	0	0	0	0	0	0	0	0
	4	0	1	0	0	0	1	0	0	0	0	0	0
Subtract	3	1	0	0	1	0	0	1	0	0	0	0	0
	4	0	1	0	0	0	1	0	0	0	0	0	0
Load Imm	3	0	0	0	1	0	0	0	0	1	0	0	0
Load Word	3	1	0	0	1	0	0	0	0	0	0	0	0
	4	0	0	1	0	0	0	0	0	1	0	0	0
Store Word	3	1	0	0	1	0	0	0	0	0	0	0	0
	4	0	1	0	0	0	0	0	1	0	0	0	0
Conditional Jump	3	0	0	0	1	0	0	1	0	0	0	1	0

Since the EEPROM can only store 16 words, we simplify the process of generating control signals by making a combinational circuit of all the array of signals under the “Common” section (first section in the above table).

Two arrays of 12 control signals are generated while executing an instruction, one coming from the combinational circuits for the “Common” section and one from the rest of the sections in the above table. There is a multiplexer, whose select line input

comes from another combinational circuit, and this multiplexer selects one of the two arrays of 12 control signals.

The following is the circuit diagram responsible for generating all the control signals :



CONTROL SIGNALS

Working of the Processor

In our 8-bit processor, each instruction is subdivided into microinstructions. Microinstructions are those instructions which can be completely executed within one clock cycle. Every instruction starts with the same possible 3 microinstructions. These are as follows :

1. MI and CO : These control signals ensure that the output of the program counter (PC) enters into the memory address register (MAR).

2. RO and II : These control signals ensure that the instruction is loaded from the RAM to the instruction register (IR).

3. CE : This control signal updates the program counter (PC) to the next instruction.

The above 3 steps are common for all operations and ensure that the instruction has been loaded into the instruction register(IR) and the program counter is updated to the next instruction. The following are the different operations that our 8-bit processor can perform:

Addition and Subtraction :

The first three bits represent the op-code for addition operation (***000***). The fourth bit is the principal register bit, and the last four bits are the memory address bits. This instruction is used to add the contents of the other register to the contents of the ***principal register*** (register given in the instruction) and then store the result in the memory address numbered by the last 4 bits.

Similarly for subtraction, the first three bits represent the op-code for the subtraction operation(***001***). The fourth bit is the principal register bit, and the last four bits are the memory address bits. This instruction is used to subtract the contents of the other register from the contents of the ***principal register*** (register given in the instruction) and then store the result in the memory address numbered by the last 4 bits.

The following are the steps involved in the execution of addition/subtraction operation after the execution of the 3 common microinstructions :

- MI and IO :

  These control signals load the address numbered by the last 4 bits of the instruction into the memory address register (MAR).

- RI , AO and S :

  The control signal AO allows the output of the ALU after the operation is completed to be loaded onto the bus, while the control signal RI allows this value to be loaded into the RAM chip from the bus.

  During this step, in the case of addition, the value stored in the registers in the ALU directly enters into the adders.

  While for subtraction, an additional control signal, the control signal 'S' (stands for subtraction) ensures to generate the 2's complement of the value of one of the registers and then adds this 2's complement to the value stored on the other register(***i.e the principal register***).

Load Immediate :

The first three bits represent the op-code for load immediate operation (***010***). The fourth bit is the principal register bit, and the last four bits are the value which is to be stored in the given register. This instruction is used to load a value directly into the ***principal register*** (keep in mind this value can only be of length 4 bits).

- IO and RegI:

  The IO control signal is used to output the 4 least significant bits of the instruction(which is the value to be stored in the principal register). onto the bus.

  While RegI control signal is used to load the value from the bus onto the principal register

Load Word and Store Word :

In the case of "load word" operation, the first three bits represent the op-code for load memory operation(***011***). The fourth bit is the register bit, and the last four bits represent the memory address bits. This instruction is used to load

the contents of a memory address numbered by the address bits into a register, obtained by the register bit.

In the case of “store word” operation, the first three bits represent the op-code for store memory operation(***100***). The fourth bit is the register bit, and the last four bits represent the memory address bits. This instruction is used to store the contents of a register obtained by the register bit into a memory location in the RAM numbered by the address bits.

- MI and IO :

  These control signals load the address numbered by the last 4 bits of the instruction into the memory address register (MAR).

- RO / RI and RegI / RegO :

  Load word instruction uses the control signals named RO and RegI. The RO control signal is used to load a value from a particular address from the RAM onto the bus while the control signal RegI takes this value from the bus and stores it in the register.

  Store word instruction uses the control signals named RI and RegO. The control signal RegO loads the value stored in the register onto the bus while the control signal RI causes the RAM to pick up this value from the bus and store it.

Conditional Jump :

The first three bits of the instruction are used to signify the op-code for the conditional jump instruction (***101***). The bit number 4 is used to represent the ***principal register***, while the last 4 bits of the instruction are used to signify the address where the user wants to transfer control to.

If the value stored in the principal register is greater than that of the value stored in the other register, then the control jumps to another location as desired by the user. However if the value of the principal register is smaller than that of the other register, no jumping takes place.

- IO, S and J :

  The IO control signal is used to output the 4 least significant bits of the instruction(which is the address which the user wants to jump to). onto the bus.

While the control signal S is used to subtract the two values present in the principal register and the secondary register. If the pin labeled “Cout” of the 4-bit adder, to which the values “A7” to “A4” are attached to in the ALU circuit diagram provided above, is zero, then it implies that the value of the primary register is greater than that of the secondary register and jumping does take place.

This “Cout” signal is negated (inverse logic) and then enters an “AND” gate with the second input being the control signal named “J” (stands for jump) and its output goes to the pin labeled as (LD) in the 8-bit binary counter in the Program Counter (PC) circuit diagram as shown previously above.