PROJECT REPORT

Course name: computer architecture laboratory

Course number: CSE 2114

Project title: Building a minimal computer system

Submission date: 12th February, 2025

Name: Fateh kabir

Roll: 2207079

Section: B

Year: 2nd

Term: 1st

Objectives:

* To learn how to design a CPU of 32 bits and implement basic operations in it.
* To provide a concise overview of the design choices and architecture of the minimal computer system, emphasizing simplicity and functionality.

Introduction:

The CPU, or central processing unit, is the heart of a computer's architecture. It serves as the primary component responsible for executing instructions and performing calculations. Within the CPU, various units such as the arithmetic logic unit (ALU) and control unit work in harmony to interpret and execute instructions fetched from memory. The CPU's architecture defines its instruction set, data-path and control mechanisms, which dictate its capabilities and performance. Understanding the CPU's role and structure is crucial for comprehending how computers process data and execute programs efficiently. CPU comprises several essential parts:

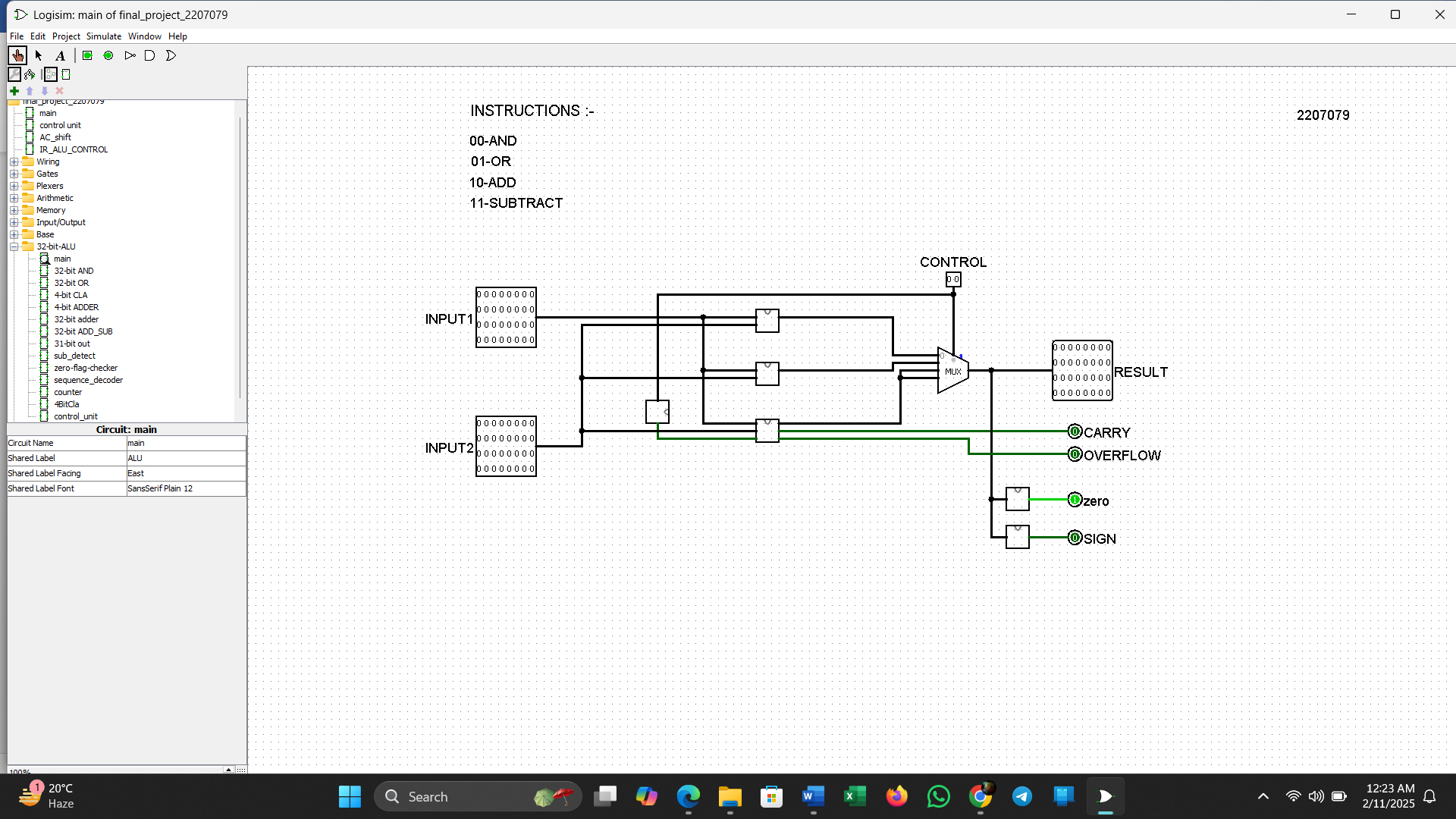
Arithmetic logic unit (ALU): ALU is responsible for performing arithmetic and logical operations like AND, OR, addition, subtraction.

In this project we made a 32bit ALU, where we perform operations of two 32bit operands. Our ALU can performs only those 4 operations mentioned above. We only need 2bit commands to select the required operations. This is done by 2x1 multiplexer. Each operation is done in separate circuit. Then we connected all by a multiplexer. The 2bit control is for selecting the operations.

This the control bit for every operation:

|  |  |
| --- | --- |
| AND | 00 |
| OR | 01 |
| ADDITION | 10 |
| SUBTRACTION | 11 |

For addition and subtraction, we use carry look ahead for reducing the carry delay.

Here is the picture of my ALU circuit:

Here the result is stored in the ‘RESULT’ output. We also calculate some flags too. When we find a carry in the operation the carry flag will set to 1. Similarly, if the result is 0 the zero flag will set to 1 and if the result is a negative number, the sign bit will set to 1.

Registers: Small, high-speed memory units within the CPU used to store data temporarily during processing. Common types include the program counter (PC), instruction register (IR), and general purpose registers (e.g., accumulator). For storing the data from the memory, it has memory buffer register (MBR) and for addressing the memory CPU has memory address register (MAR).

As my CPU is 32bit, my MBR is also 32bit, because it holds the data from the memory and the data bit is 32bits. In our project, we use 32word size memory RAM, there for our MAR is only 5bit, because we need 5bits to represent 32bit memory address. That’s why our PC is also 5bits. Here PC register also holds the address of the memory, but the main difference of PC and MAR is, PC holds the next instruction address of the memory. than we use IR register, it refers instruction register. Our project gives 4bit IR. It means we can have 16 instructions in our CPU.

All of the registers are built with 4bit parallel load register and 4bit parallel load with counter register.

Control unit: Coordinates the activities of the CPU by directing the flow of data and instructions between various components. It decodes instructions fetched from memory and generates control signals to execute them. It activates various microprogram to perform a single task. Control unit handles the flow of the program.

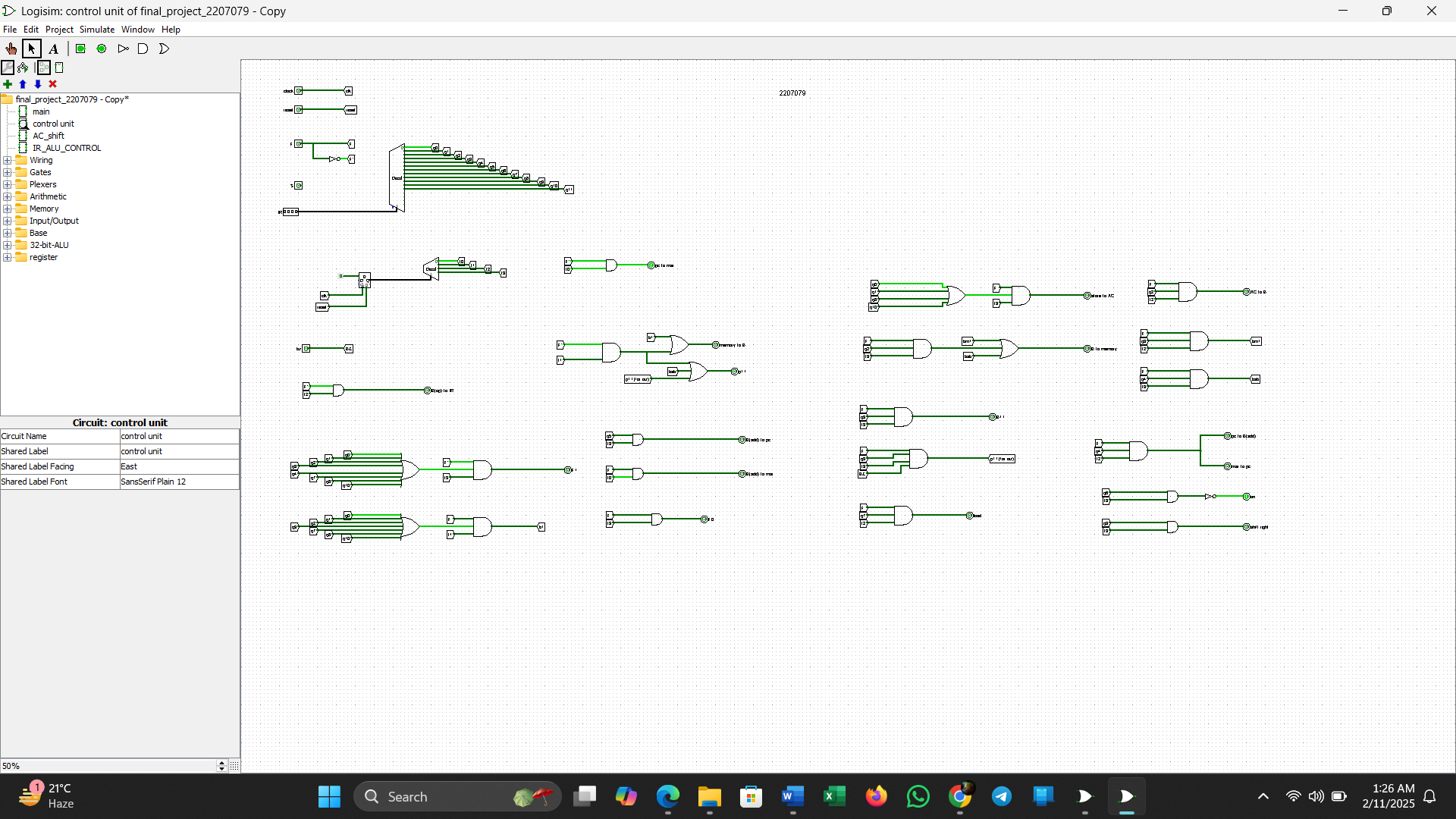
Here first we decode the time with timing sequence (representing as t0, t1, t2, t3). We provide the 4bit op code to the control unit and it decode it with the help of decoder. The F is represented whether it is in the fetch cycle (when F=0) or execution cycle (when F=1).

Here are the op codes of my 32bit CPU instructions:

|  |  |
| --- | --- |
| AND | 0000 |
| ADD | 0001 |
| STORE | 0010 |
| ISZ | 0011 |
| BSB | 0100 |
| BUN | 0101 |
| HALT | 0110 |
| LOAD | 0111 |
| OR | 1000 |
| SHR | 1001 |
| SUB | 1010 |

In my control unit, the IC holds each microprogram for all the instructions above. Every instruction performs multiple programs during their execution time. Control unit decides with action will be perform at that moment to execute the instruction.

The following picture is the complete circuit of my control unit:



Once the control unit is done, now its time to create the main CPU circuit. The CPU has all 5 registers, RAM, control unit, clock, flipflops in it. Each clock pulse performs each task.

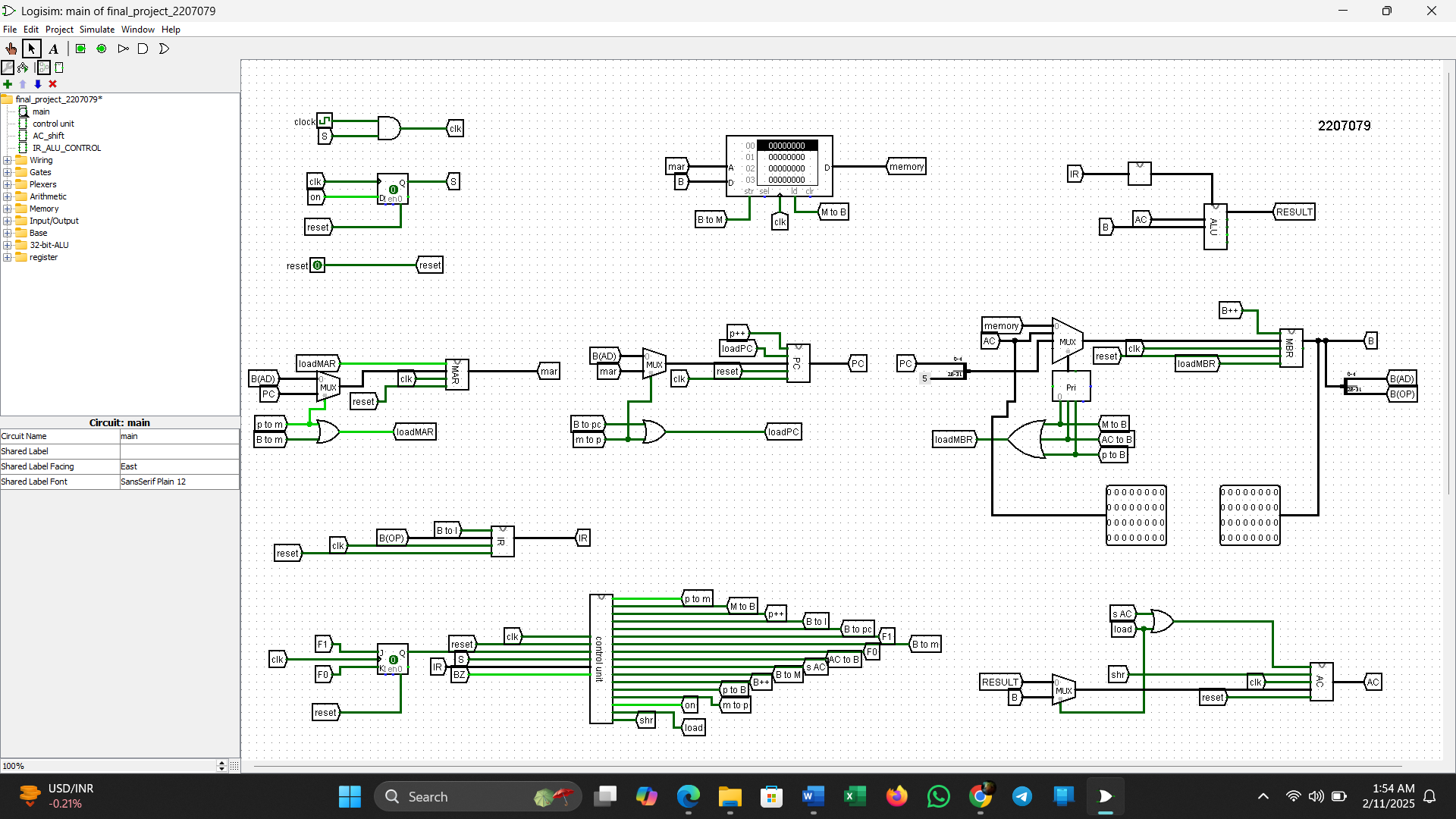
In my circuit, I take MSB 4bit as op code and LSB 5bit as address.

|  |  |  |
| --- | --- | --- |
| Op-code |  | Address |
| 31-28 bit | 27-5 bit | 4-0 bit |

MBR stores 32bit data from memory and gives op code to IR register and address to MAR, when it fetches instruction. Then IR register decode the op code in control unit and finds out which instruction is to be preform now. The microprogram in control unit generates the signal which needs to be executed.

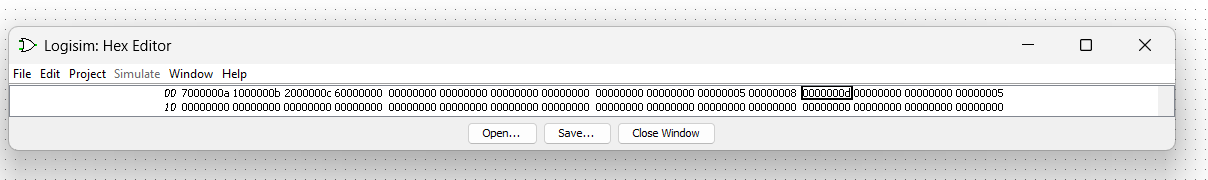
Now CPU has all 5 registers, and each register is connected with control unit by multiplexing. Suppose when the CPU needs to load data in the MBR, control unit generates a signal for load data to buffer, then the MBR loads data from memory. Sometimes MBR stores data from AC and PC. This problem is solved by multiplexing, when MBR needs to store data from memory, it loads from memory, when it needs to store from AC, control signal generates signal to store data of AC and via multiplexing, MBR will store the data. All other registers work similarly.

The following is the 32bit CPU circuit:

In this picture, we can see, I multiplexing each register with required input it needed at program demand.

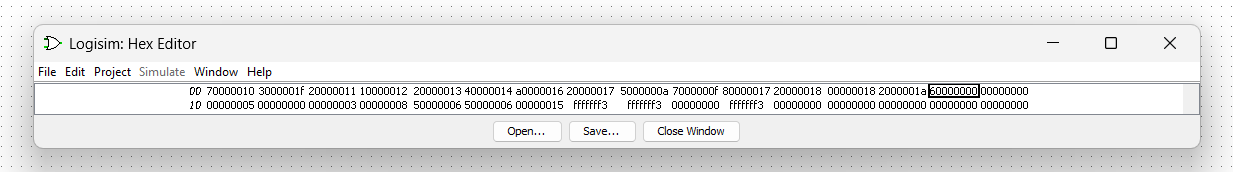
Sample:

Now, I want to execute some program in my CPU. Let’s say, I want to execute addition. For addition, the op code is 0001 and I want to store the value at 1100 address location. Here is the hex editor for my add operation:



Here at first address location, I perform load operation, the program then fetches the instruction and go to the ‘a’ location. Then, it loads the data 5 (which is the value store in ‘a’ address) to the AC register. Meanwhile PC increment and point to the 2nd address. At address 2, add instruction is given. Similarly, the CPU fetch the data from memory and decode the instruction and location (which is ‘b’ now), and perform the operation. Lastly, the value in the AC will store to the ‘c’ location by store operation instruction. Here we can see, in c address value d is stored, which is the addition of 8+5. Finally, the halt (op code-0110) operation executes.

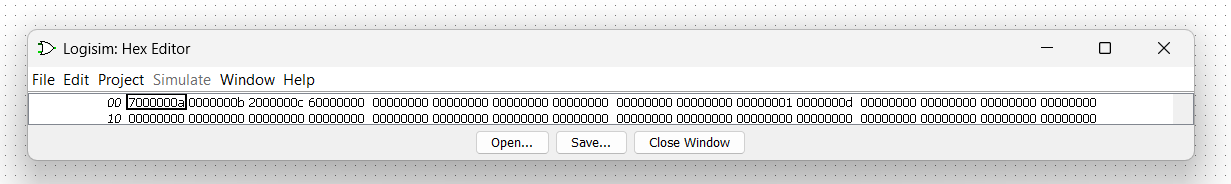
Take another exemple, where I perform various instructions in one program. Here is the picture of the hex editor



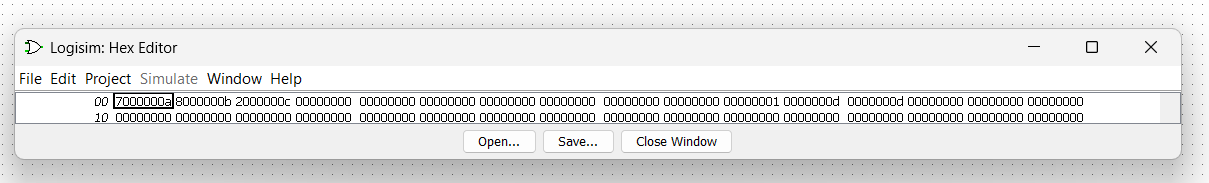
Here I perform all instructions in one program and performs all tasks carefully. We can see all the executions are performed logically and correctly.

Single instruction exectution editor is given below for every instructions I have,

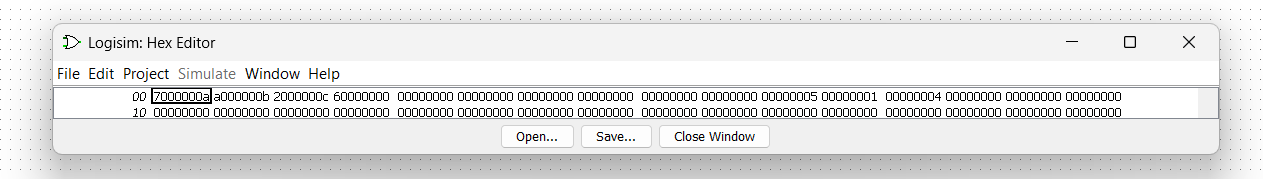
For AND operation:



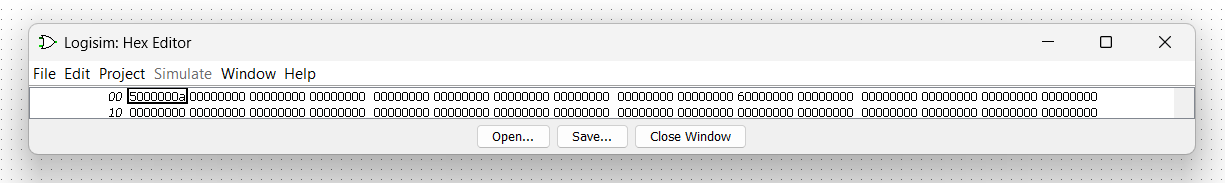
For OR operation:



For SUB operation:



For BUN operation:



conclusion: In this project, we reflect on the valuable insights acquired through the implementation of a simple CPU within the von neuman architecture. This endeavor has provided us with a understanding of CPU architecture and design principles, elucidating how computers execute instructions and process data. Through this process, we have encountered and evaluates various trade-offs inherent in CPU design, notably balancing simplicity with performance and emphasizing the significance of efficient memory access. Despite encountering some inefficiencies during implementation, our successful construction of the CPU offers us a multitude of operational choices, underscoring its potential for further exploration and enhancement.