Computer Architecture Lab Project Documentation

Name: Mutiur Rahman

Roll: 2207097

Group: B2

Course No: CSE 2114

Course Title: Computer Architecture Laboratory

Project Title: Design and Simulation of an 18-bit CPU Architecture using Logisim

Date of submission: 03/08/2025

Submitted to

Dr. M.M.A. Hashem
Professor,
Department of Computer Science and Engineering,
Khulna University of Engineering & Technology, Khulna.

Md. Sakhawat Hossain
Lecturer,
Department of Computer Science and Engineering,
Khulna University of Engineering & Technology, Khulna.

Objectives:

- To understand and implement the architecture of a basic CPU.
- To simulate an **Arithmetic Logic Unit (ALU)** and its integration into the CPU system.
- To model the functionality of registers like **MAR, MBR, PC, IR, and Accumulator**.
- To visualize the instruction execution cycle in a simulated environment.
- To enhance comprehension of **fetch-decode-execute** cycles.
- To explore hardware-software coordination in digital computer systems.
- To gain hands-on experience in digital circuit design using Logisim.
- To analyze real-world data flow and control signal operations within the CPU.
- To develop a simplified CPU that mimics fundamental processor operations.

Introduction:

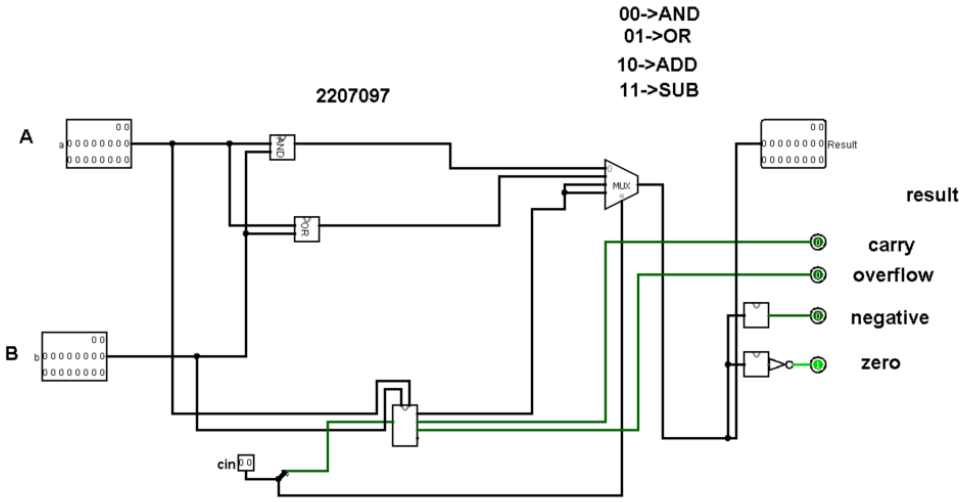
A computer system is composed of several interrelated components that collectively execute instructions and manage data flow. At the core of this system lies the Central Processing Unit (CPU), which drives the execution cycle. This project aims to replicate a simplified 18-bit CPU using Logisim, focusing on the core architectural elements and instruction handling.

The designed CPU incorporates essential modules: an 18-bit ALU, control unit, memory registers, and instruction execution logic. This project allows for an intuitive understanding of low-level CPU operations, instruction formats, and memory access protocols. Here the CPU is designed using a 18-bit RAM with address bit width of 5. So the RAM has total of 2^5 =32 addressable unit. As the word width is 18 bits, so the total size of the RAM is 2^5 x18 bits.

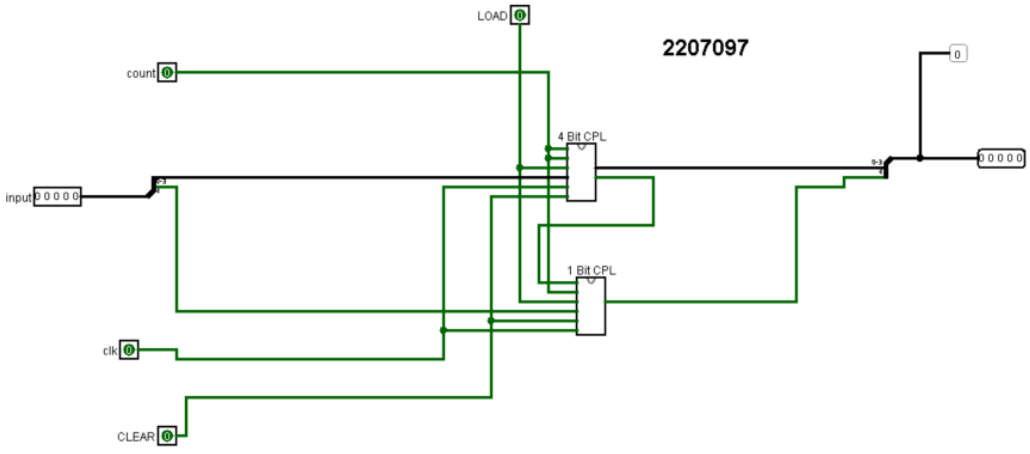
In this system, each instruction is structured to use specific bit ranges for different purposes. Four bits are allocated to represent the operation code (opcode), and five bits are designated for the memory address. Specifically, the least significant five bits (bits 0–4) of the instruction are reserved for the address, while bits 12–15 are used for the opcode. The remaining bits in the instruction are treated as "don't care" bits, meaning they are not assigned a specific function in this context. For data representation, all 18 bits are utilized to store the data value.

RAM: Random-Access Memory (RAM) is a vital type of computer storage that allows data to be accessed or modified in any order with consistent speed. Unlike sequential storage like hard drives, RAM enables quick retrieval and updating of information, regardless of its location in the memory. RAM temporarily holds active data and program instructions, acting as a fast workspace for a computer’s processor. This ensures smooth and efficient task execution. Its uniform access time sets it apart from slower storage systems, making it essential for responsive performance in devices like computers and smartphones.

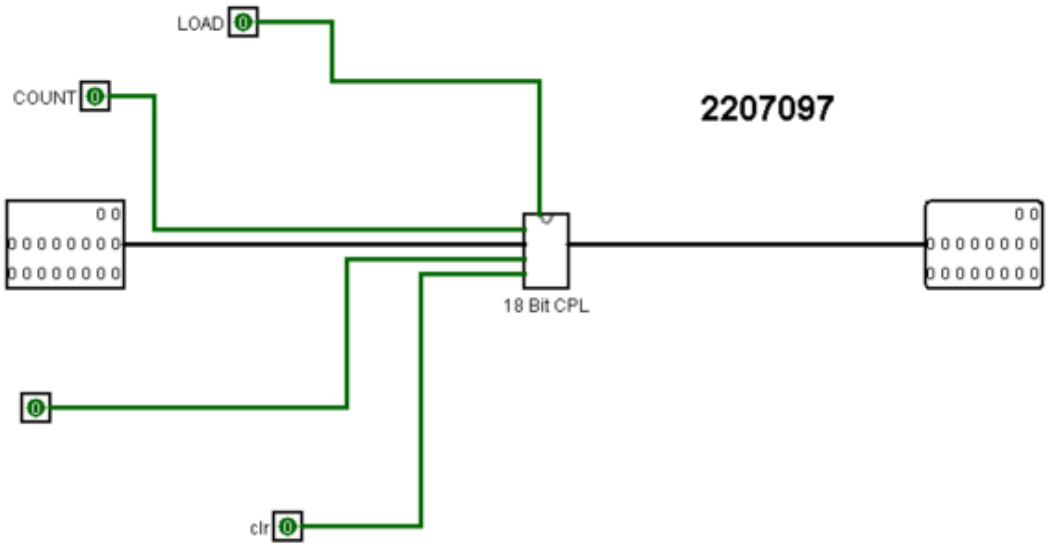
ALU: The ALU performs arithmetic operations (Addition, Subtraction) and logical operations (AND, OR) on 18-bit binary numbers. The implemented operations are ADD, SUB, AND, OR. The ALU takes inputs from memory or registers, processes the operation, and returns results to the accumulator or memory.



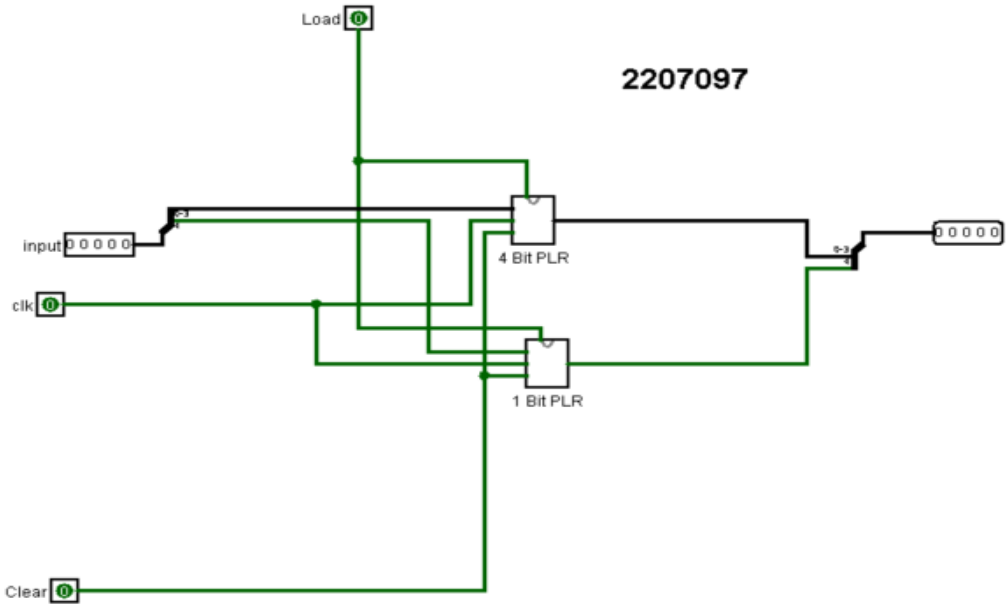
PC: The Program Counter (PC) is a vital CPU register that stores the memory address of the next instruction to be executed in a program. By advancing with each instruction cycle, the PC guarantees the CPU fetches and processes instructions in the proper order, playing a critical role in seamless program execution and system performance.
The Instruction Pointer (IP), a key element of a computer's processor, holds the address of the next command to be retrieved from memory. Incrementing with each cycle, the IP ensures the CPU executes instructions sequentially, making it indispensable for maintaining program flow and operational efficiency.



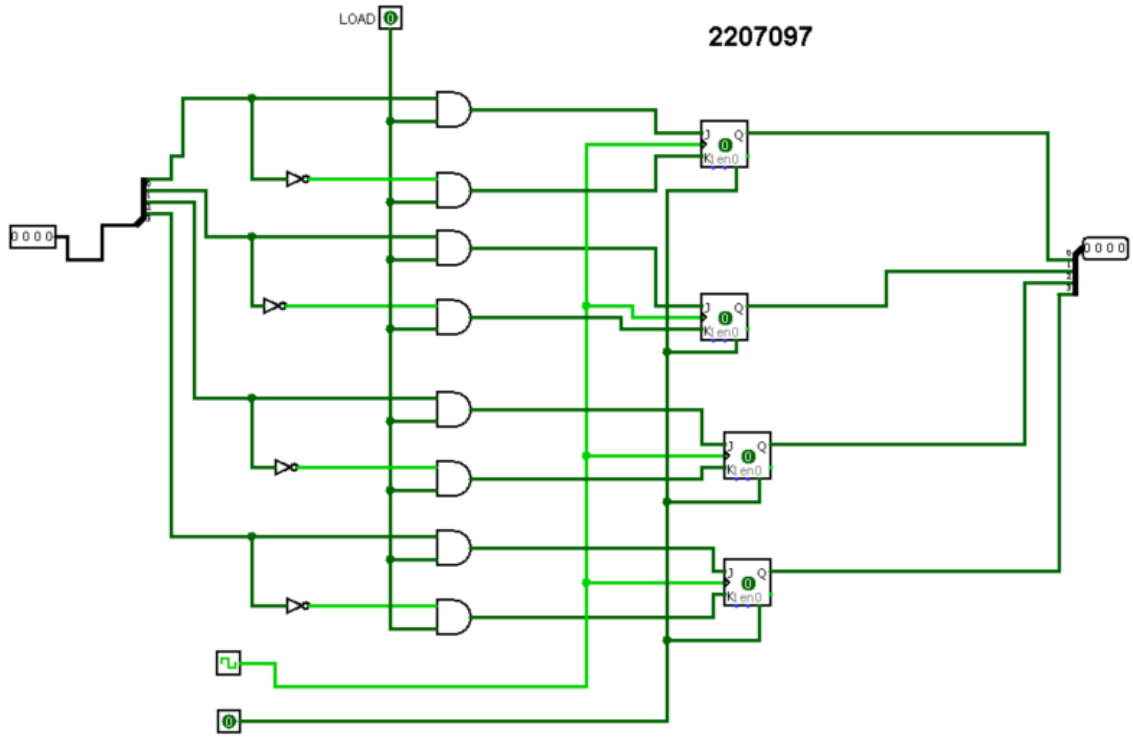
MBR: The Memory Buffer Register (MBR), also known as the Memory Data Register (MDR), is a critical CPU component that temporarily holds data being transferred to or from the computer's main memory. Acting as an intermediary, the MBR stores instructions or data fetched from memory during the fetch phase of the instruction cycle or holds data to be written to memory during the execution phase. By buffering these transfers, the MBR ensures efficient communication between the CPU and memory, enhancing the speed and reliability of data processing in a computer system.



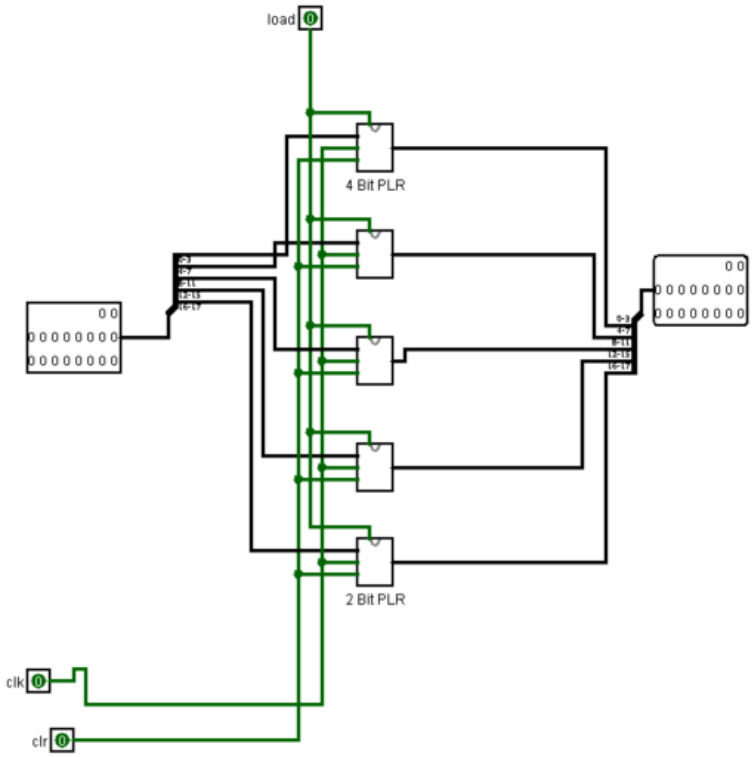
MAR: The Memory Address Register (MAR) is an essential CPU component that stores the memory address of the data or instruction to be accessed in the computer's main memory. During the instruction cycle, the MAR holds the address from which the CPU fetches data or instructions or to which it writes data.



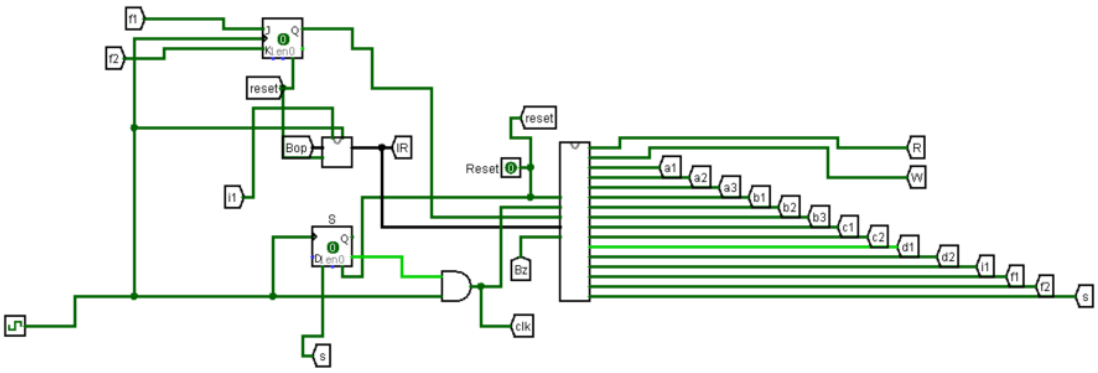
IR: The Instruction Register (IR) is a crucial CPU component that temporarily holds the current instruction fetched from memory during the instruction cycle. Once the Program Counter (PC) provides the address of the next instruction and it is retrieved via the Memory Address Register (MAR) and Memory Buffer Register (MBR), the IR stores this instruction for decoding and execution. By holding the instruction while the CPU interprets and processes it, the IR ensures accurate execution of program commands, playing a pivotal role in maintaining the flow and efficiency of program operations.



Accumulator Register: The Accumulator Register (AC) is a key component of a CPU that serves as a temporary storage location for intermediate results during data processing. It holds data fetched from memory via the Memory Buffer Register (MBR) and stores the outcomes of arithmetic, logical, or data manipulation operations performed by the Arithmetic Logic Unit (ALU). By acting as a central hub for computations, the Accumulator facilitates efficient processing and minimizes memory access, significantly enhancing the CPU's performance and speed in executing program instructions.



Control Unit: The Control Unit (CU) is a fundamental component of a CPU that orchestrates the execution of program instructions. It directs the operation of the processor by coordinating the flow of data between the CPU's registers, such as the Program Counter (PC), Memory Address Register (MAR), Memory Buffer Register (MBR), Instruction Register (IR), and Accumulator (AC), as well as the Arithmetic Logic Unit (ALU) and memory. The CU fetches instructions from memory, decodes them, and generates the necessary control signals to execute each instruction, ensuring proper sequencing and timing.



Opcode List: The following chart describes the opcode of instructions.

Instruction	Opcode	Instruction	Opcode
AND	0	LDA	6
ADD	1	HLT	7
STO	2	OR	8
ISZ	3	SUB	9
BSB	4	INC	A
BUN	5		

Instruction Format in the 18-bit CPU:

NONE	OPCODE	NONE	ADDRESS
17-16	15-12	11-5	4-0

Instruction Set Overview for Minicomputer Operations:

1. **0060c**: (Fetch Data into Accumulator) Retrieves the data stored at memory location 0x1c and loads it into the Accumulator for processing.
2. **0000d**: (Logical AND Operation) Loads the value from memory location 0x1d into the Memory Buffer Register (MBR) and performs a bitwise AND operation with the Accumulator's contents using the ALU.
3. **0020e**: (Store Result) Transfers the contents of the MBR to memory location 0x1e for storage.
4. **00505**: (Branch and Save Return) Jumps program execution to memory location 0x05, storing the return address at that location for later use (Branch and Save Back).
5. **0010c**: (Addition Operation) Adds the value stored at memory location 0x0c to the current value in the Accumulator, with the result stored back in the Accumulator.
6. **00700**: (Halt Execution) Stops all processing operations while keeping the system clock active, effectively pausing the minicomputer.

System Design and Operation

The minicomputer integrates a CPU, memory, and control unit to execute programmed tasks. The CPU fetches instructions from memory, decodes them, and carries out operations. The control unit manages data and instruction flow, ensuring seamless coordination. Instructions are fetched by transferring the address from the Memory Buffer Register (MBR) to the Memory Address Register (MAR). The system executes instructions sequentially, with the Arithmetic Logic Unit (ALU) handling arithmetic (e.g., addition) and logical operations (e.g., AND). The Accumulator temporarily holds intermediate results, while memory stores both instructions and data. Supported operations include loading, storing, jumping, and halting, enabling the system to perform basic computations. By simulating clock cycles and providing input data, the minicomputer processes instructions to deliver the intended output.

Observations & Discussion

Through a series of lab sessions, we progressed from basic logic design to a functional CPU simulation. Starting from designing an 18-bit ALU, we explored complex computation via control signals, and finally integrated our learnings into a complete CPU circuit. By isolating each sub-component (ALU, PC, IR, etc.), we ensured modular testing and easier debugging. The use of Logisim provided a powerful yet intuitive platform to visualize logic flow, making the debugging and testing process efficient.

Conclusion

In conclusion, We successfully built a simple minicomputer using Logisim, diving into the essentials of computer architecture like CPU operations, memory management, and control systems. This hands-on project brought data flow and instruction processing to life, deepening our appreciation for computing fundamentals. Though limited in scope, our minicomputer is a solid starting point for future exploration in computer science and engineering, sparking both skill and excitement for what's next.