Project Report

Name : Saleh Sadid Mir
Roll : 2207024
Department : Computer Science and Engineering
Group : A1
Course No : CSE-2114
Course Title : Computer Architecture Laboratory
Project Title : Designing a minimal Computer System (27 Bit) using Logisim
Date :
Submitted to :

Dr. M.M.A Hashem
Professor
Department of
Computer Science and Engineering,
KUET.

Md. Shakhawat Hossain
Lecturer
Department of
Computer Science and Engineering,
KUET.

Title: Design of a 27-bit minimal computer.

Objectives:

i. To design and implement a 27-bit minimal computer using Logisim software.
ii. To develop a control unit capable of executing 11 instructions, including AND, ADD, STO, ISZ, BSB, BUN, LOAD, HALT, XOR, and DECREMENT.
iii. To understand and apply the principles of computer architecture by designing a functional minimal computer.
iv. To learn about instruction execution, memory management, and control logic.

Introduction:

A computer's Central Processing Unit (CPU) is the main part that serves as its "Control Centre." The CPU, also called the "central" or "main" processor, is a sophisticated electrical circuit that powers the operating system and applications on the computer. A CPU is made of some main components.

They are:

(a) Control Unit

(b) Arithmetic and Logical Unit (ALU)

(c) Registers

(d) Main Memory

Control Unit:

A part of the central processor unit (CPU) that oversees the functioning of a computer. The control unit retrieves instructions in the form of bit-by-bit representations from the CPU's memory and converts them into control signals, which take the shape of light or electrical pulses.

Arithmetic and Logical Unit (ALU):

In computing, an arithmetic logic unit (ALU) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.

Registers:

MAR (Memory Address Register): Holds the memory address of data to be accessed. Total 6 bits are allocated for MAR.

MBR (Memory Buffer Register): Temporarily stores data being transferred to or from memory. Total 27 bits are allocated for MBR.

IR (Instruction Register): Holds the current instruction being executed. Total 4 bits are allocated for IR.

PC (Program Counter): Keeps track of the memory address of the next instruction. Total 6 bits are allocated for MAR.

AC (Accumulator): Temporarily stores data during processing. Total 27 bits are allocated for Accumulator.

Main Memory:

The main memory is implemented using a built-in RAM module in Logisim, with a 6-bit address bit width and 27-bit data bit width. This allows for 64 addressable memory locations, each capable of storing 27-bit data. Machine code for instructions and data is loaded into the RAM. The memory is accessed using the 6-bit address code, and data is transferred between the memory and the CPU.

The CPU is designed using the Harvard Architecture. The ALU serves as the primary operation execution unit and supports four basic operations: addition, subtraction, bitwise AND and bitwise OR. The CPU can load and store data, perform unconditional or conditional jumps, execute arithmetic operations, and halt after executing a sequence of instructions, enabling it to perform more complex tasks such as branching, multiplication, and nested loops. The data bit width of the instruction RAM is 27 bits, with 4 bits (20^{th}, 21^{st}, 22^{nd},23^{rd}) of an instruction serving as the OP Code and the higher 6 bits holding the address of the operand.

Component	Description
Word size	27 Bits
OP-code	4 Bits
Address Bit Width	6 Bits
No. of memory locations	64

This is the main computer circuit:

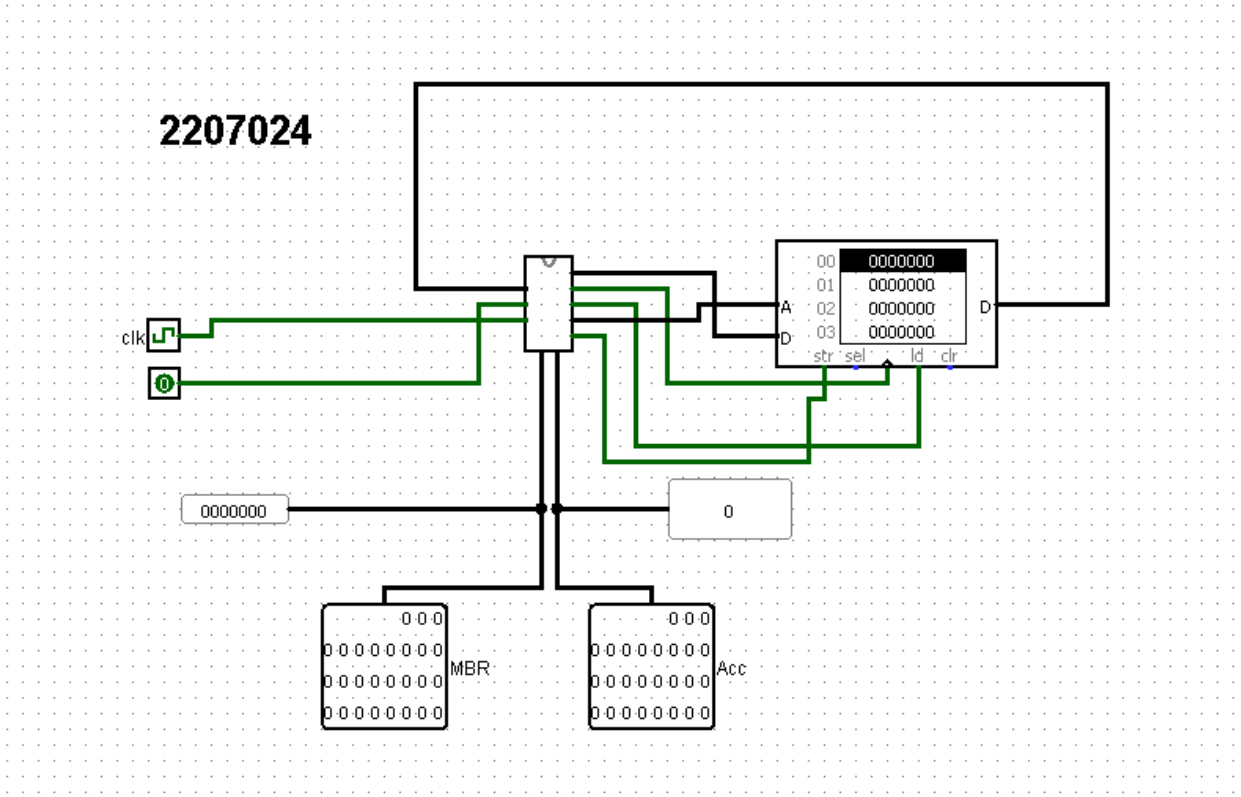


Fig: 27 – bit CPU

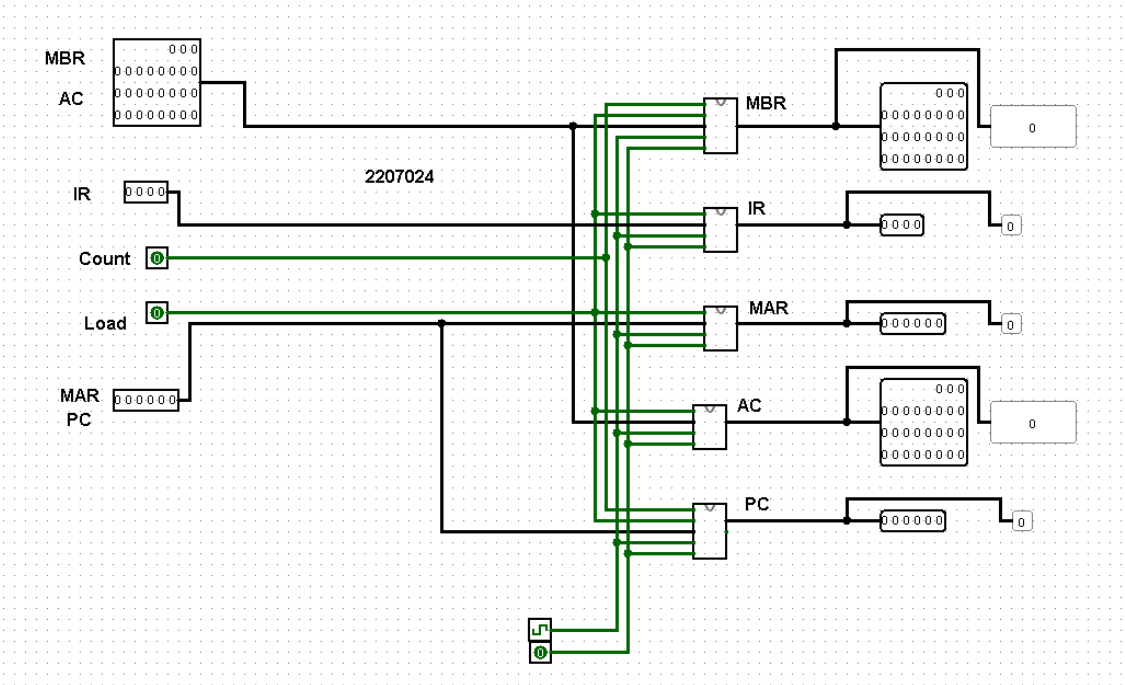


Fig: All registers

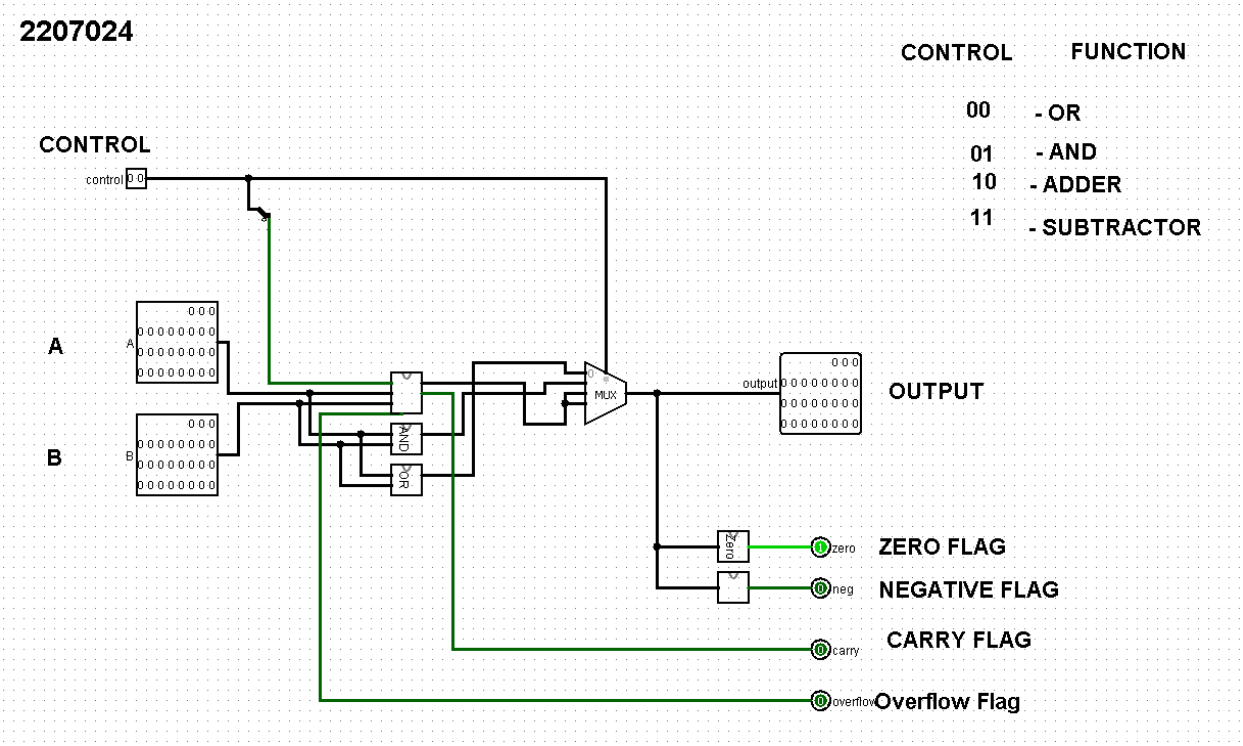


Fig: 27 bit ALU Circuit

Instruction Set and Implementation:

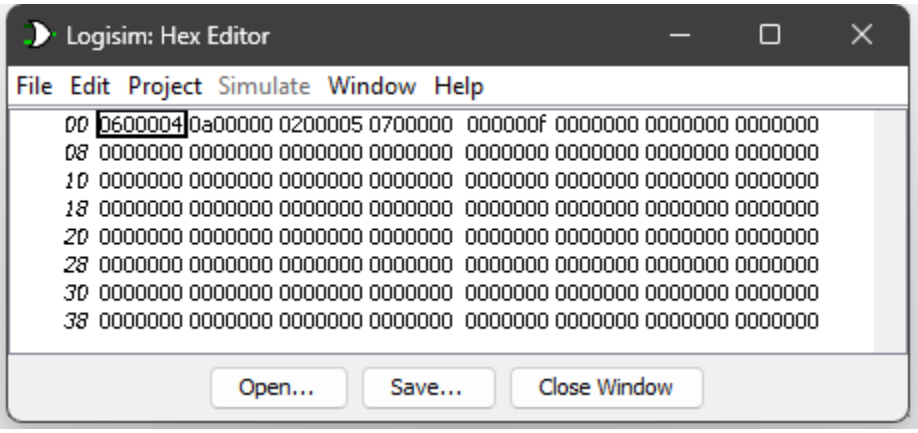
The computer supports a total of 11 instructions, each encoded with a 4-bit opcode.
Instruction Format:(27 Bits)
The instructions are as follows:

1. AND: Performs a bitwise AND operation between the accumulator and a specified memory location.
2. ADD: Adds the contents of a specified memory location to the accumulator.
3. STO: Stores the contents of the accumulator into a specified memory location.
4. ISZ: Increments the contents of a specified memory location and skips the next instruction if the result is zero.
5. BSB: Branches to a subroutine by saving the return address and jumping to a specified memory location.
6. BUN: Branches unconditionally to a specified memory location.
7. LOAD: Loads the contents of a specified memory location into the accumulator.
8. HLT: Stops the execution of the program.
9. INC2: Increments the contents of the accumulator by two.
10. SHR: Shifts the binary value of the accumulator right by one and stores in accumulator .
11. 2CMP: 2's complement the content of accumulator and stores in accumulator .

Supported Op-codes:

INSTRUCTION	OPCODE(HEX)	OPCODE(BINARY)
AND	0	0000
ADD	1	0001
STO	2	0010
ISZ	3	0011
BSB	4	0100
BUN	5	0101
LOAD	6	0110
HLT	7	0111
INC2	8	1000
SHR	9	1001
2CMP	A	1010

A sample program:.



here each block represents a command. Starting from first.
0600004 – tells to load the contents present in '4' cell of memory (fetch cycle).
0a00000 – 'a' is the instruction for 2CMP (2's compliment)
0200005 – '2' This will now store the value of Accumulator in '5' cell of memory. Replacing the previous one (if any).
0700000 – '7' is hault command. After this, the program will stop executing on a further clock cycle.
000000f – This is the content of cell no 4 .

Discussion:

The development of the 27-bit minimal computer presented several challenges, particularly in designing the control unit and integrating the ALU for decrement operations. The control unit managed the fetch-decode-execute cycle for 10 distinct instructions, requiring precise control signals for seamless operation. Memory constraints, with a 6-bit address space allowing only 64 locations, necessitated efficient allocation strategies. Synchronization across components was crucial, with the master clock ensuring proper coordination of the ALU, registers, and memory. The ALU, supporting multiple arithmetic and logical operations, required careful data path and control logic planning. Logisim's built-in RAM streamlined instruction storage and execution. Overall, this project provided valuable insights into CPU design, control logic, and memory management using digital logic simulation tools.

Conclusion:

The project has achieved its objectives by creating a simple computer system using Logisim software. This endeavor has provided valuable insights into computer architecture and digital circuit design in an accessible manner. Through the implementation of basic arithmetic and logical operations, the project has laid a solid foundation for understanding complex CPU designs and the principles of computer architecture.

By exploring the functionalities of the CPU, including instruction execution and control flow management, participants have gained a deeper understanding of how computers process data. Logisim's user-friendly interface has made it easy for users to experiment with digital logic concepts, fostering hands-on learning and exploration.

Looking ahead, there's potential to enhance the performance and functionality of the system for more advanced applications. This could involve optimizing the design to improve speed and efficiency, as well as adding support for additional instructions or features. Overall, the project has provided an engaging introduction to computer architecture, paving the way for further exploration and experimentation in the field.