**Documentation Prepared By:** Group 8

**Name:** Jeffrey John Owusu and Shadrack Agyei Nti  
**CS 331 Computer Organization and Architecture**  
**Instructor:** Dr. Robert Sowah  
**December 9, 2024**

**1. Overview**

This documentation presents the design, simulation, and analysis of a 16-bit Central Processing Unit (CPU). As the culmination of theoretical and practical coursework, this project emphasizes core components such as the Arithmetic Logic Unit (ALU), registers, memory, and the control unit, integrated within a functional datapath. Using Logisim Evolution, the CPU was developed to execute basic instructions, enabling hands-on understanding of hardware-software interaction.

**2. Design and Simulation Process**

**2.1 Arithmetic Logic Unit (ALU)**

**Design Overview**

The ALU serves as the CPU's computational core, executing a variety of arithmetic and logical operations with a **4-bit selector** for arithmetic and a **3-bit selector** for relational instructions.

**Supported Operations**

* **Arithmetic:** Addition, subtraction, multiplication, division, modulo, shifts, and 2’s complement.
* **Relational:** Greater than, less than, equal to, not equal to, and comparisons.

**Inputs and Outputs**

* **Inputs:** A and B (16-bit), AUSel (4-bit), RUSel (3-bit).
* **Outputs:** AUResult (16-bit), RUResult (1-bit).

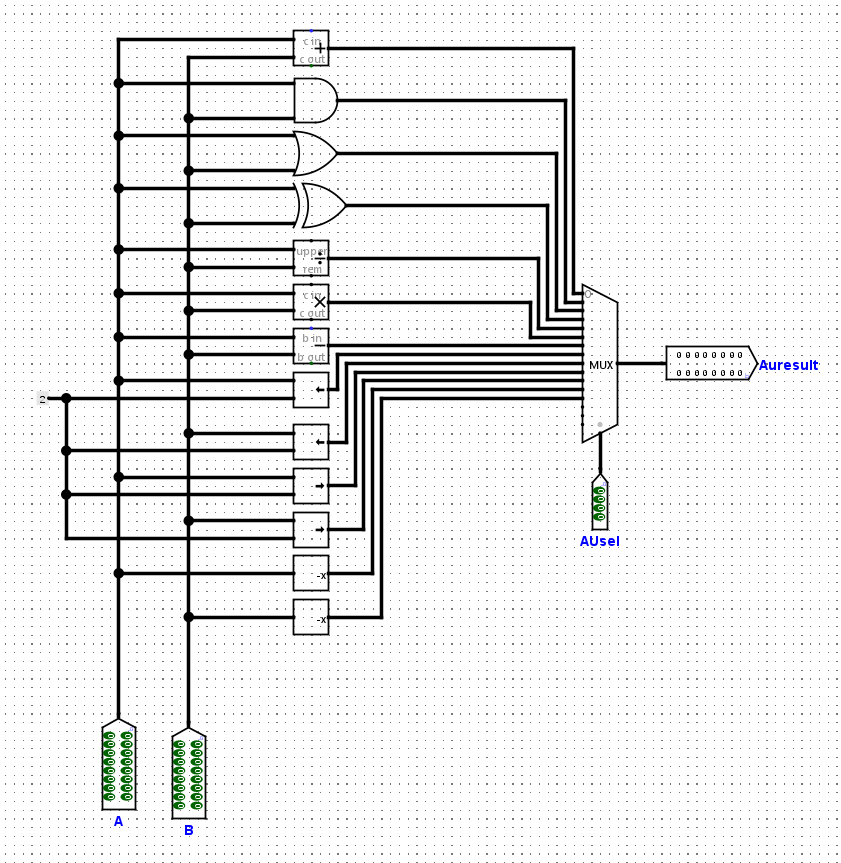
**Implementation**

Utilizing Logisim’s built-in components such as multiplexers, shifters, and negators, the ALU was tested for both individual and integrated functionality.

**Arithmetic Operations**

The ALU supports the following arithmetic operations, each associated with a specific selector value:

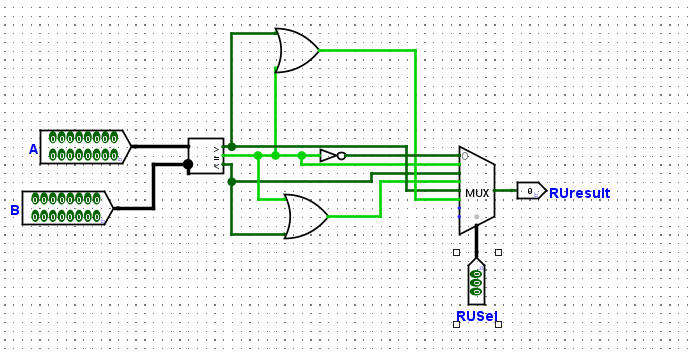
* **Add (0):** Computes A+B .
* **Bitwise AND (1):** Computes A&B .
* **Bitwise OR (2):** Computes A∣B .
* **XOR (3):** Computes A⊕B.
* **Division (4):** Computes A÷B.
* **Remainder (5):** Computes A%B.
* **Multiplication (6):** Computes A×B.
* **Subtract (7):** Computes A−B.
* **Shift Left Logical A by 2 (8):** Computes A≪2.
* **Shift Left Logical B by 2 (9):** Computes B≪2.
* **Shift Right Logical A by 2 (10):** Computes A≫2.
* **Shift Right Logical B by 2 (11):** Computes B≫2.
* **2’s Complement A (12):** Computes −A.
* **2’s Complement B (13):** Computes −B



**Relational Operations**

The ALU also performs relational comparisons, determined by the relational selector value:

* **Greater Than (0):** Checks if A>B.
* **Equal (1):** Checks if A==B.
* **Greater or Equal (2):** Checks if A≥B.
* **Less Than (3):** Checks if A<B.
* **Less or Equal (4):** Checks if A≤B.
* **Not Equal (5):** Checks if A≠B.

**Inputs and Outputs**

* **Inputs:**
  + A and B (16-bit): Operands for computation.
  + AUSel (4-bit): Selector for arithmetic operations.
  + RUSel (3-bit): Selector for relational operations.
* **Outputs:**
  + AUResult (16-bit): Result of the arithmetic operation.
  + RUResult (1-bit): Result of the relational operation (true or false).

A computer screen shot of a diagram

Description automatically generated

**Implementation**

The ALU was implemented using a combination of Logisim’s built-in components, including:

* **Adder:** Performs addition.
* **Subtractor:** Handles subtraction operations.
* **Multiplier:** Supports multiplication, considering carry-out bits.
* **Divider:** Performs division and modulo calculations.
* **Shifter:** Executes logical left and right shifts.
* **Negator:** Computes 2's complement values.
* **MUX (Multiplexer):** Selects the desired operation output based on the selector inputs.

**2.2 Registers**

Registers provide temporary data storage and facilitate efficient CPU operations. Key registers include:

1. **Instruction Register (IR):** Holds the current instruction.
2. **Registers A and B:** Store operands for the ALU.
3. **Memory Address Register (MAR):** Specifies the memory address for operations.
4. **Memory Buffer Register (MBR):** Handles memory data transfers.
5. **Output Register:** Stores results from ALU operations.

**2.3 Datapath Integration**

The datapath connects the CPU’s core components via buses:

* **Internal Buses:** Facilitate communication within the CPU.
* **External Buses:** Enable memory interaction.  
  This integration ensures seamless data flow and proper instruction execution.

A diagram of a machine

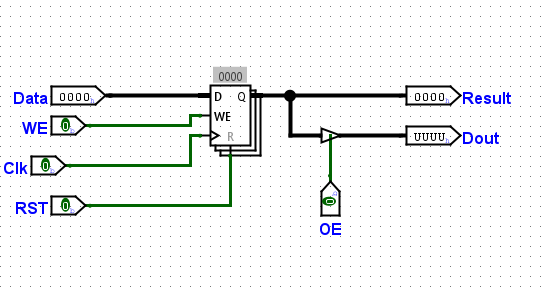
Description automatically generated

**3. CPU Design**

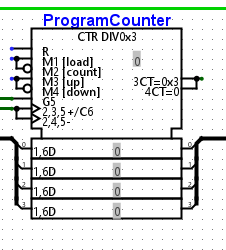
**3.1 CPU Architecture**

The CPU combines the following components into a cohesive system:

1. **ALU:** Performs computations and logic operations.
2. **Control Unit (CU):** Manages instruction decoding and signal generation.
3. **Registers:** Provide high-speed data storage during execution.



1. **Program Counter (PC):** Tracks the sequence of instructions.



1. **RAM Interface:** Connects the CPU to memory for instruction and data retrieval.

**3.2 Instruction Execution**

The CPU processes instructions in four main stages:

1. **Fetch:** The instruction is retrieved from memory and loaded into the IR.
2. **Decode:** The CU interprets the opcode to generate control signals.
3. **Execute:** The ALU performs the specified operation on operands from registers.
4. **Write-back:** The result is stored in the output register or written to memory.

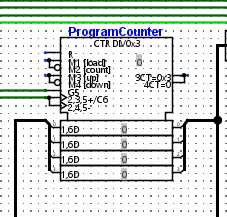
**3.3 Key Features**

* **Modularity:** Individual components like the ALU and registers can be independently tested.
* **Scalability:** The CPU’s design supports future expansion, such as adding more instructions or pipelining.
* **Integration with Peripherals:** The CPU interfaces seamlessly with RAM and display modules.

**4. Program Counter, RAM, and Display Integration**

**4.1 Program Counter (PC)**

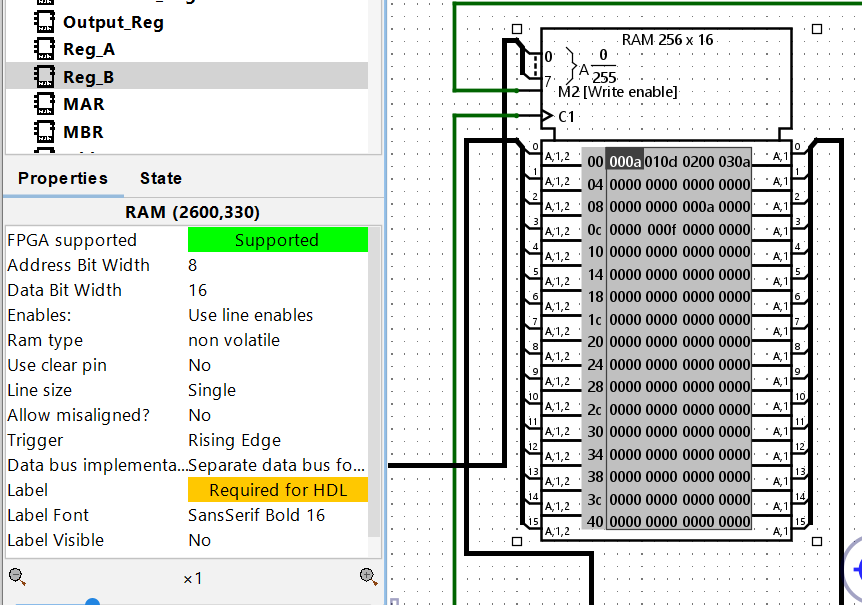
The PC manages sequential execution by incrementing after each instruction. Its value is displayed on a **7-segment display** for real-time monitoring.



**4.2 RAM**

A **16-bit wide, 8-bit addressable RAM** supports instruction and data storage:

* **Instruction Segment:** Holds machine instructions.
* **Data Segment:** Stores operands and results.



**4.3 Display Integration**

Using a **decimal decoder**, the CPU displays values for the program counter, registers, and ALU output.

**5. Opcode Assignments and RAM Content**

**5.1 Opcode Assignments**

The CPU supports the following instructions:

* **LoadA:** 00000000 – Load operand A into Register A.
* **LoadB:** 00000001 – Load operand B into Register B.
* **Arithmetic:** 00000010 – Perform an ALU operation (e.g., addition).
* **Store:** 00000011 – Write the result to memory.

**5.2 RAM Configuration**

RAM was preloaded for testing with the following setup:

* **Instruction Segment:**
  + Address 0: LoadA (Load operand A from memory).
  + Address 1: LoadB (Load operand B from memory).
  + Address 2: Arithmetic (Add operands).
  + Address 3: Store (Write result to memory).
* **Data Segment:**
  + Address 10: Operand A = 10.
  + Address 13: Operand B = 15.

**6. Control Unit and Step Counter**

**6.1 Control Unit Design**

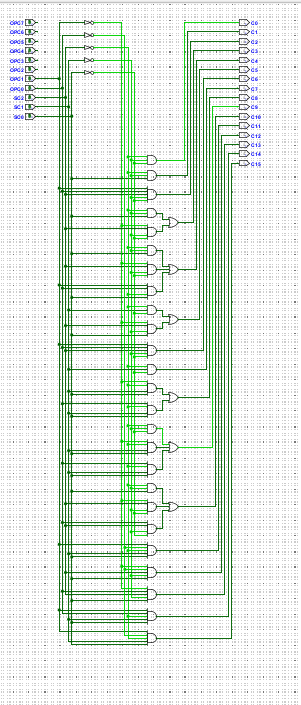
The CU directs CPU operations through a combination of:

* **Opcode Interpretation:** Determines the operation based on instruction.
* **Signal Generation:** Produces control signals to synchronize data flow and execution.

**Table**

**A table of numbers

Description automatically generated with medium confidence**



**6.2 Step Counter**

The step counter coordinates the execution sequence by tracking micro-operations during each instruction cycle.

**7. Challenges and Insights**

**7.1 Challenges**

1. **Timing Synchronization:** Achieving harmony between components to avoid incorrect results.
2. **Debugging:** Identifying issues in the ALU outputs and memory interactions.

**7.2 Insights**

1. **Importance of Testing:** Individual component testing simplifies debugging.
2. **Logisim Features:** Tools like combinational analysis enhance design efficiency.

**8. Conclusion**

The design and simulation of this 16-bit CPU highlight the complexities of hardware development and the elegance of computer architecture. From modular testing to full-system integration, this project exemplifies the principles underpinning modern processors. While basic in functionality, this CPU lays a strong foundation for future enhancements, such as pipelining or expanded instruction sets.

Our CPU journey reaffirms that every digital marvel starts with a single transistor, progressing toward a unified system capable of shaping the digital age.

Answers To Reflection Questions