Communication and Computer Engineering department

Shoubra faculty of engineering

Benha university

Single Cycle MIPS CPU

Supervised by

Dr. Lamiaa Elrefaai

Eng. Shimaa Yosry

By

Ahmed khaled Mohamed

Ahmed waleed Ahmed

Hazem Ahmed Abdelfattah

Saied Khaled Awad

Table of Contents

List of figures

List of tables

1.Design and implementation

1.1. Data path

The CPU is based on a simplified MIPS architecture implementing a subset of the instruction set, totaling 24 instructions: 14 R-type, 9 I-type, and 1 J-type. The data path begins with the Program Counter (PC), which provides the address to the instruction memory to fetch the current instruction. The register file then processes this instruction, which reads the necessary source registers. The subsequent operations depend on the instruction type. For R-type instructions, both operands are sourced from the registers and passed through the ALU to operate, with the result written back to the destination register. For I-type instructions, one operand is an immediate value rather than a register value, and the result is also typically written back to a register, except in the case of the SW (store word) instruction, which writes data to memory instead. The single J-type instruction involves computing a new PC address through the ALU, which is fed back into the PC. The PC control logic ensures that the correct path is taken by selecting the appropriate address via a multiplexer, based on the instruction type and control signals generated accordingly.

1.2. Components

The components for the whole CPU include the Register File, the PC module, the Instruction memory, the data memory, the ALU, the ALU Control, the PC control, and the Main control unit.

1.3. Control

The CPU's control logic is divided into three main units: the Main Control Unit, the ALU Control Unit, and the PC Control Unit. The Main Control Unit is responsible for decoding the opcode from the current instruction and generating the necessary control signals to coordinate the behavior of the processor according to the MIPS specification. It outputs signals such as Jump, BEQ, BNE, ALUSrc, and ALUOp, which are passed to the PC Control Unit and the ALU Control Unit to guide their actions.

The PC Control Unit determines the address that the Program Counter (PC) should hold for the next instruction cycle. This unit uses control signals like Jump, BEQ, and BNE, along with the Zero flag from the ALU, to decide the appropriate next instruction address. Based on this decision, it sets a selection value for a multiplexer that determines whether the PC should proceed to the next sequential instruction, take a branch, or jump to a different address.

The ALU Control Unit receives the ALUSrc and ALUOp signals from the Main Control Unit. If the instruction uses an immediate value, it also considers the function bits from the instruction itself. It then selects two input values for the ALU: A, which is always from a register, and B, which is chosen from either another register or an immediate constant based on the ALUSrc signal. Using the ALUOp, the unit selects the appropriate ALU operation to perform. The result of this operation may also generate additional outputs, such as the Zero signal, which plays a role in branch decisions

1.4. component circuits

1.4.1. Register file

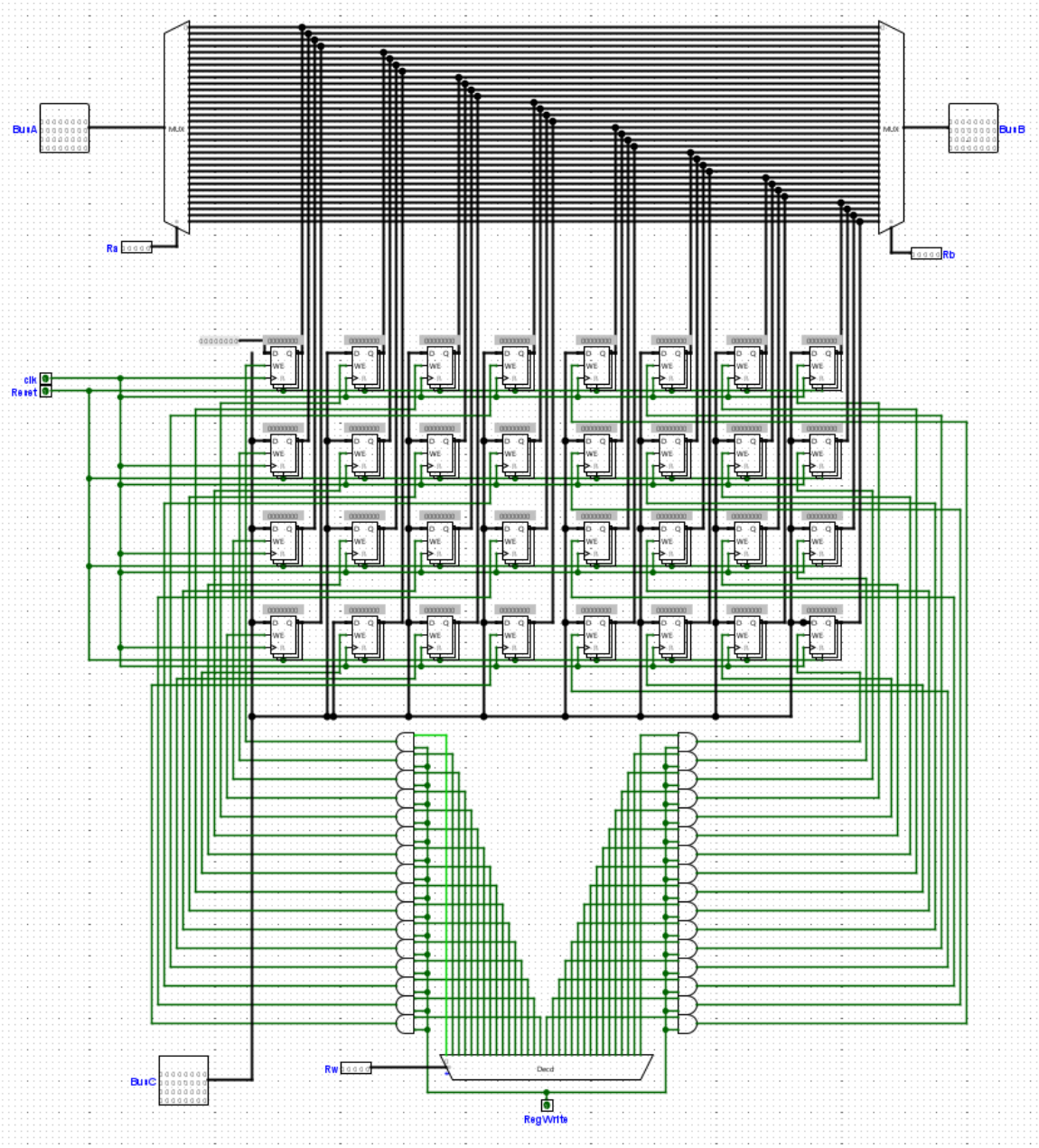

Figure 1: Register file circuit

Description

The circuit diagram shown represents the register file of a MIPS processor, a fundamental component used for temporary data storage and retrieval during instruction execution. This register file follows the MIPS architecture standard, which typically includes 32 general-purpose registers, each 32 bits wide.

Key Components and Their Functions:

1. **Register Bank (32 Registers)**

   - The core of the circuit is an array of 32 registers (labelled from R0 to R31), each capable of storing a 32-bit word.

- These are implemented using D flip-flops arranged in a matrix configuration.

2. **Read Ports**

    - The circuit supports two independent read ports. Each port includes a 5-bit address input (Ra and Rb) that selects one of the 32 registers.
    - Two multiplexers (MUX) are used to access the data from the selected registers and route them to the outputs BusA and BusB.

3. **Write Port**

    - The write operation is controlled by a 5-bit destination address input (Rw), a 32-bit data input (BusC), and a RegWrite control signal.
    - When RegWrite is enabled, the selected register is updated with the value from BusC on the rising edge of the clock signal (Clk), unless the reset (Reset) signal is active.

4. **Decoder Logic**

    - A 5-to-32 decoder is used to select the appropriate register for writing, based on the address Rw.
    - The output of the decoder is ANDed with the RegWrite control signal to generate write-enable signals for the individual registers.

5. **Control Signals**

    - Clk: The global clock input for synchronous writing.
    - Reset: Used to clear all registers (optional, depending on circuit design).
    - RegWrite: Enables writing to the register specified by Rw.

Application in MIPS Pipeline

In the MIPS datapath, this register file is used in the Instruction Decode (ID) and Execution (EX) stages to read operands for ALU operations and write results back after computation. The dual-read and single-write structure matches the format of most MIPS instructions (e.g., add $t0, $t1, $t2).

1.4.2. ALU

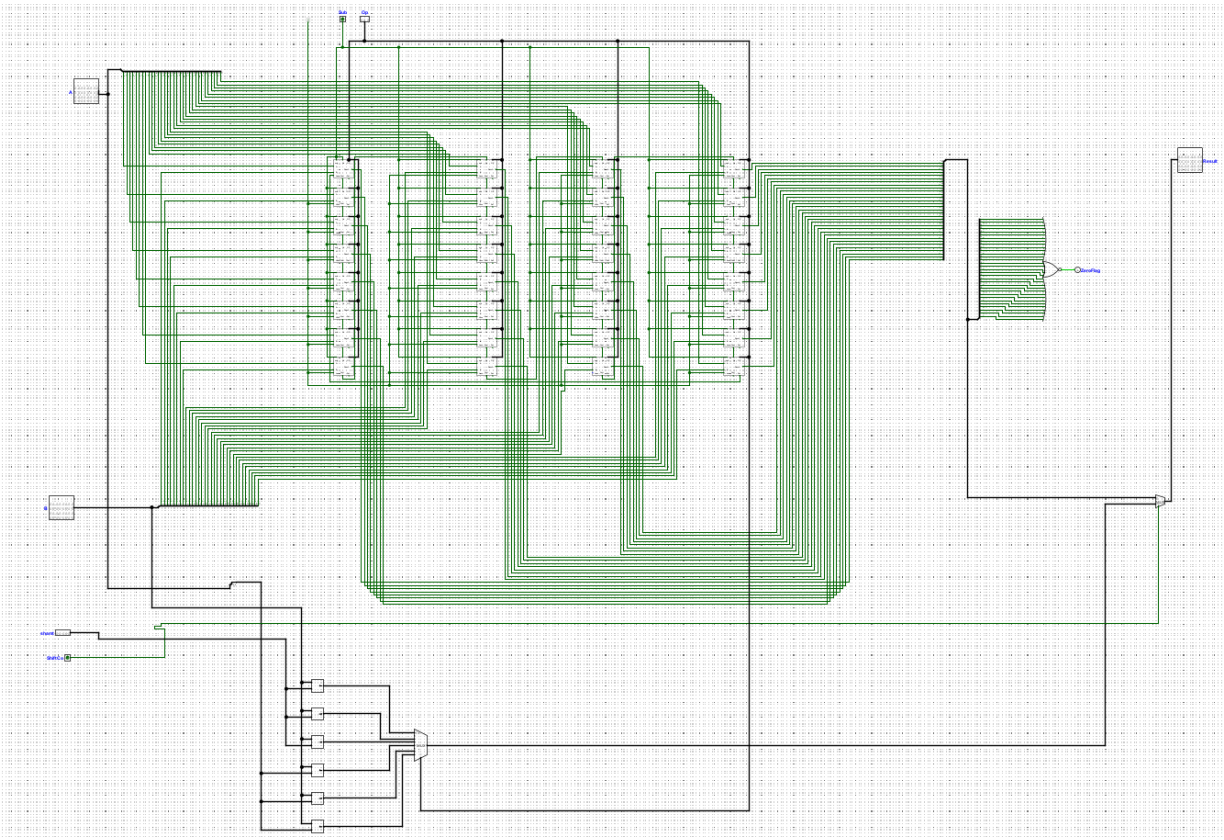

Figure 2 :Arithmetic logic unit circuit

Description

The diagram illustrates the Arithmetic Logic Unit (ALU) used within a MIPS processor. This ALU is responsible for executing arithmetic and logic operations based on control signals received from the control unit.

Main Features

Input Operands: The ALU receives two 32-bit operands through input buses.

Operation Control: A multiplexer and a set of control signals select the specific operation to perform, such as addition, subtraction, AND, OR, and set-on-less-than (SLT).

Bitwise Processing: The ALU is implemented using an array of 1-bit ALU slices, arranged to handle 32-bit operations by cascading the carry and result signals.

Result Output: The final 32-bit result is routed to the output bus.

Zero Detection: A logic gate at the output checks if the result is zero, providing a flag for conditional branching instructions.

Application in MIPS Pipeline

In the MIPS datapath, this ALU operates primarily during the Execution (EX) stage to perform operations specified by the instruction type (R-type or I-type). It plays a critical role in arithmetic calculations, address computation, and comparison operations.

1.4.3 ALU control

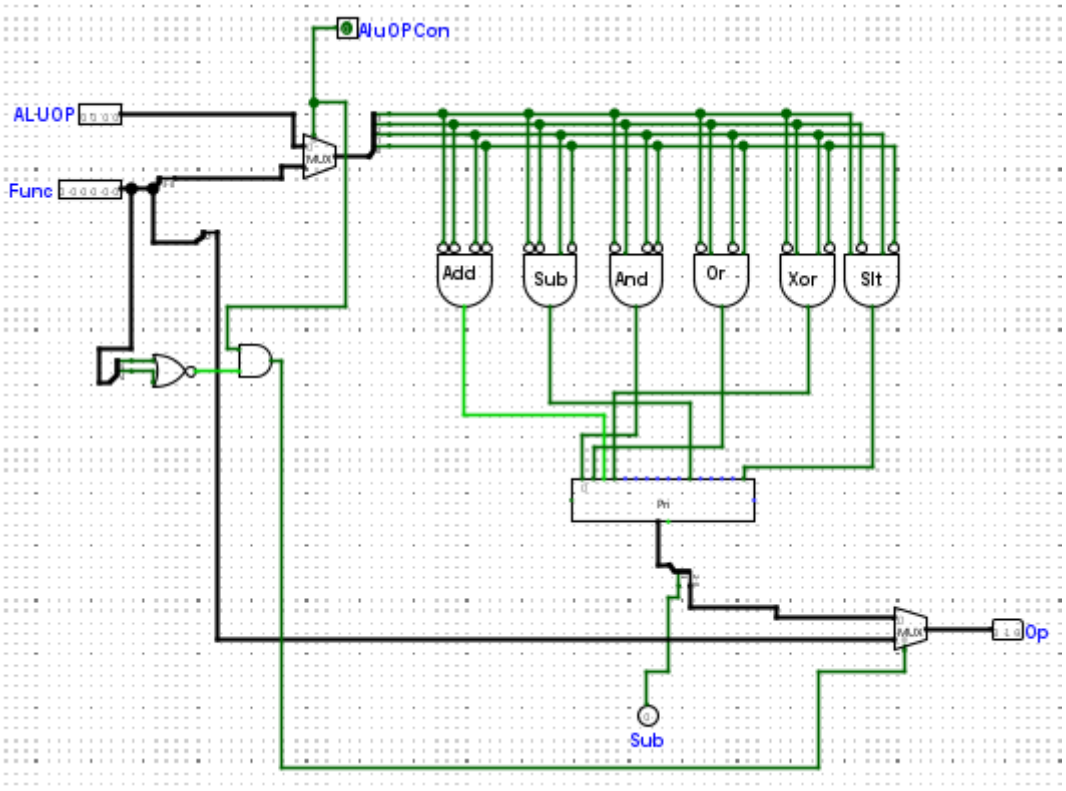

Figure 3: ALU control circuit

Description

This circuit implements the ALU Control Unit, which determines the specific operation the ALU should perform based on the instruction type. It takes a 4-bit ALUOp signal from the main control unit and a 6-bit Funct field from R-type instructions to generate a control signal for the ALU. The circuit uses logic gates and multiplexers to decode the inputs and output the appropriate operation code (e.g., add, sub, and, or, slt).

Application in MIPS Pipeline

The ALU Control Unit is active in the Execution (EX) stage of the MIPS pipeline. It ensures the ALU performs the correct function based on the instruction—arithmetic for R-type, address calculation for load/store, and comparison for branches.

1.4.4. PC control

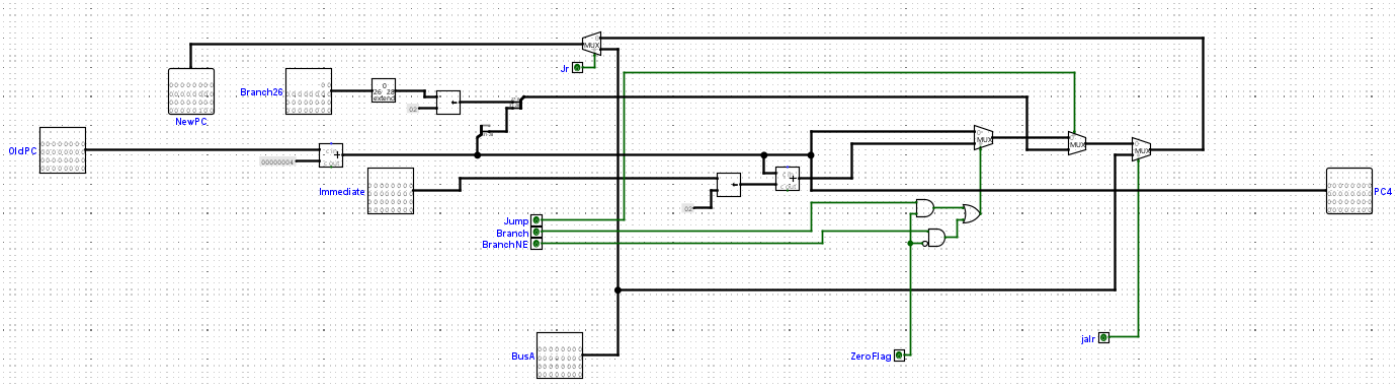

Figure 4: program counter circuit

Description

This circuit represents the Program Counter (PC) update logic in a MIPS processor. It determines the next value of the PC (PC4), based on different control signals and instruction types such as

- Jump (Jump) and Jump Register (jr): used for unconditional jumps.
- Branch (Branch) and Branch Not Equal (BranchNE): used for conditional branching based on the ZeroFlag.
- Immediate and shifted addresses: for forming target addresses in branch and jump instructions.
- OldPC and PC + 4 (implicit in NewPC): typical sequential PC increment.

Application in MIPS pipeline

The circuit uses

- Adders for calculating branch target addresses.
- MUXes (multiplexers) to select among different PC sources (sequential, jump, branch, jr).
- AND/OR gates to evaluate branch conditions (e.g., Branch AND ZeroFlag, BranchNE AND NOT ZeroFlag).

Overall, this logic determines the correct next instruction address based on the instruction type and execution results.

1.4.5. Main control unit

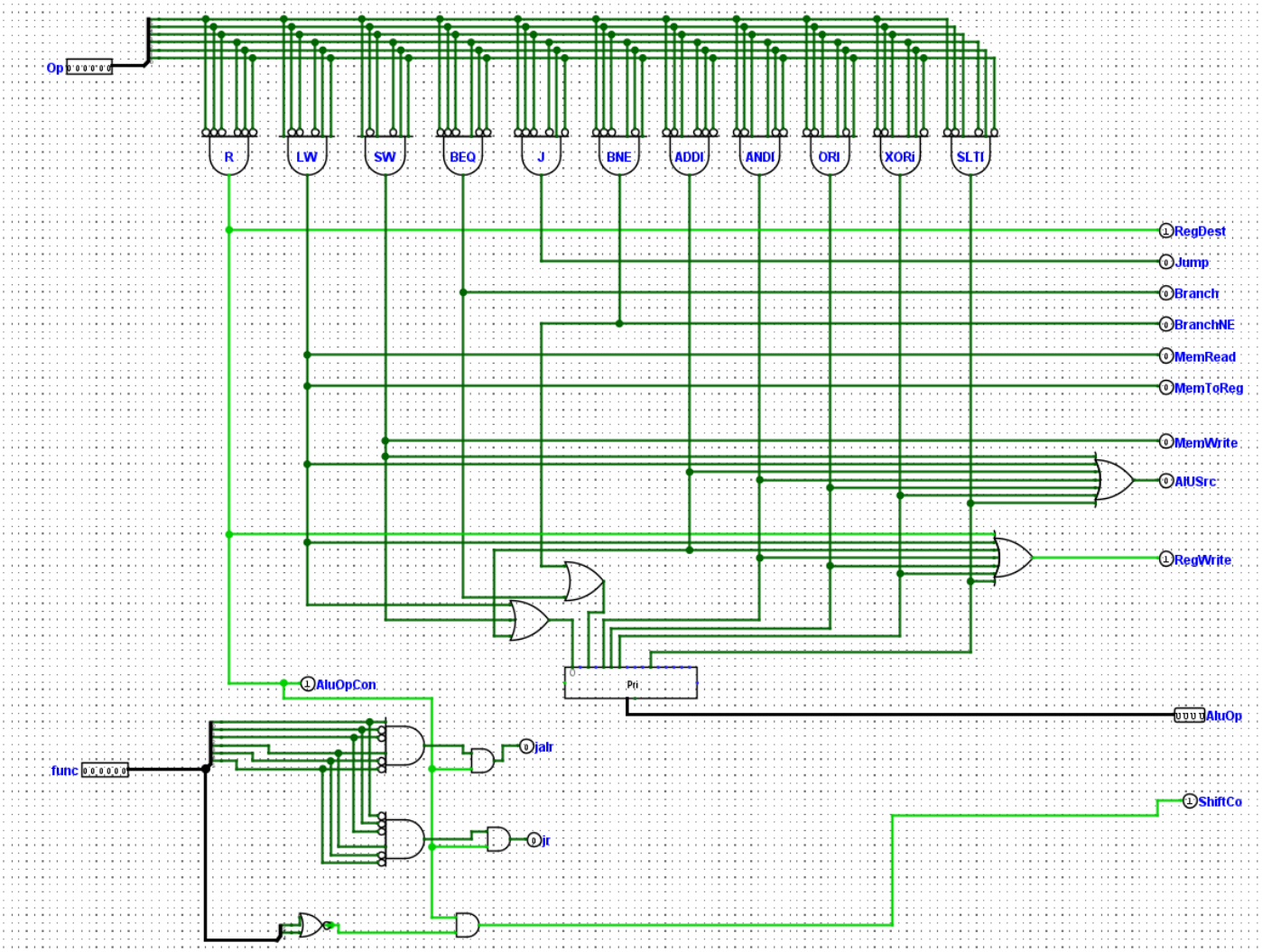

Figure 5: control unit circuit

Description

This circuit is the main control unit of a MIPS processor. It decodes the instruction's opcode (and function field for R-type) to generate control signals like:

- RegDst, ALUSrc, MemToReg, RegWrite
- MemRead, MemWrite, Branch, Jump, etc.

These signals guide the datapath operations (e.g., register writes, ALU control, memory access). It's used in the ID (Instruction Decode) stage of a MIPS pipeline to control how each instruction is executed

Application in MIPS pipeline

In the MIPS pipeline, this unit:

1. **Identifies** the instruction type.
2. **Selects** control signals that are passed along the pipeline.
3. **Ensures** that each stage executes the correct operation, especially ALU and memory access.

Without this unit, the datapath wouldn’t know what operation to perform on which data it's essential for correct instruction execution.

1.5. Main control table

Refer to Table 1 for a detailed view. For simplicity, some abbreviations are applied to the control signals:

- RD stands for RegDest (Register Destination)
- MW stands for MemWrite (Memory Write)
- MR stands for MemRead (Memory Read)
- MTR stands for MemToReg (Memory to Register)
- RW stands for RegWrite (Register Write)

Table I: main control table

Function	jump	RD	BNE	BEQ	MW	MR	MTR	RW	ALUSrc	ExtOp	ALUOp
add	x	1	x	x	x	x	x	x	x	0	0000
sub	x	1	x	x	x	x	x	1	1	0	0010
and	x	1	x	x	x	x	x	1	1	0	0100
or	x	1	x	x	x	x	x	1	1	0	0101
xor	x	1	x	x	x	x	x	1	1	0	0110
slt	x	1	x	x	x	x	x	1	1	0	0101
addi	x	x	x	x	x	x	x	1	0	1	0000
slti	x	x	x	x	x	x	x	1	0	1	1010
andi	x	x	x	x	x	x	x	1	0	0	0100
ori	x	x	x	x	x	x	x	1	0	0	0101
xori	x	x	x	x	x	x	x	1	0	0	0110
lw	x	x	x	x	x	1	1	1	0	0	0000
sw	x	x	x	x	1	x	x	x	0	0	0000
beq	x	x	x	1	x	x	x	x	1	0	0010
bne	x	x	1	x	x	x	x	x	1	0	0010
j	1	x	x	x	x	x	x	x	0	0	x
jr	1	0	0	0	0	0	0	0	0	0	0000
jalr	1`	1	0	0	0	0	0	1	0	0	0000
sll	x	1	x	x	x	x	x	1	0	0	1000
srl	x	1	x	x	x	x	x	1	0	0	1001
sra	x	1	x	x	x	x	x	1	0	0	1010
sllv	x	1	x	x	x	x	x	1	0	0	1011
srlv	x	1	x	x	x	x	x	1	0	0	1100
srav	x	1	x	x	x	x	x	1	0	0	1101

1.6. ALU control signals table

Table II: ALU control table

Function	ALU Op Code	Sub Flag (fed in form ALU Control)
and	000	0
or	001	0
xor	010	0
add	100	0
sub	100	1
slt	101	1

2.Simulation and testin

2.1Programs used

Simulate the processor developed using Logisim. All codes on mars.

2.2Implementation of simulation

The simulation was implemented by first writing the assembly code using the MARS (MIPS Assembler and Runtime Simulator) environment. Once the code was assembled, the resulting hexadecimal (hex) output was extracted and loaded into the instruction memory component within the Logisim software. This enabled accurate simulation of the system's behavior through digital circuit visualization in Logisim.

2.3.The final circuit

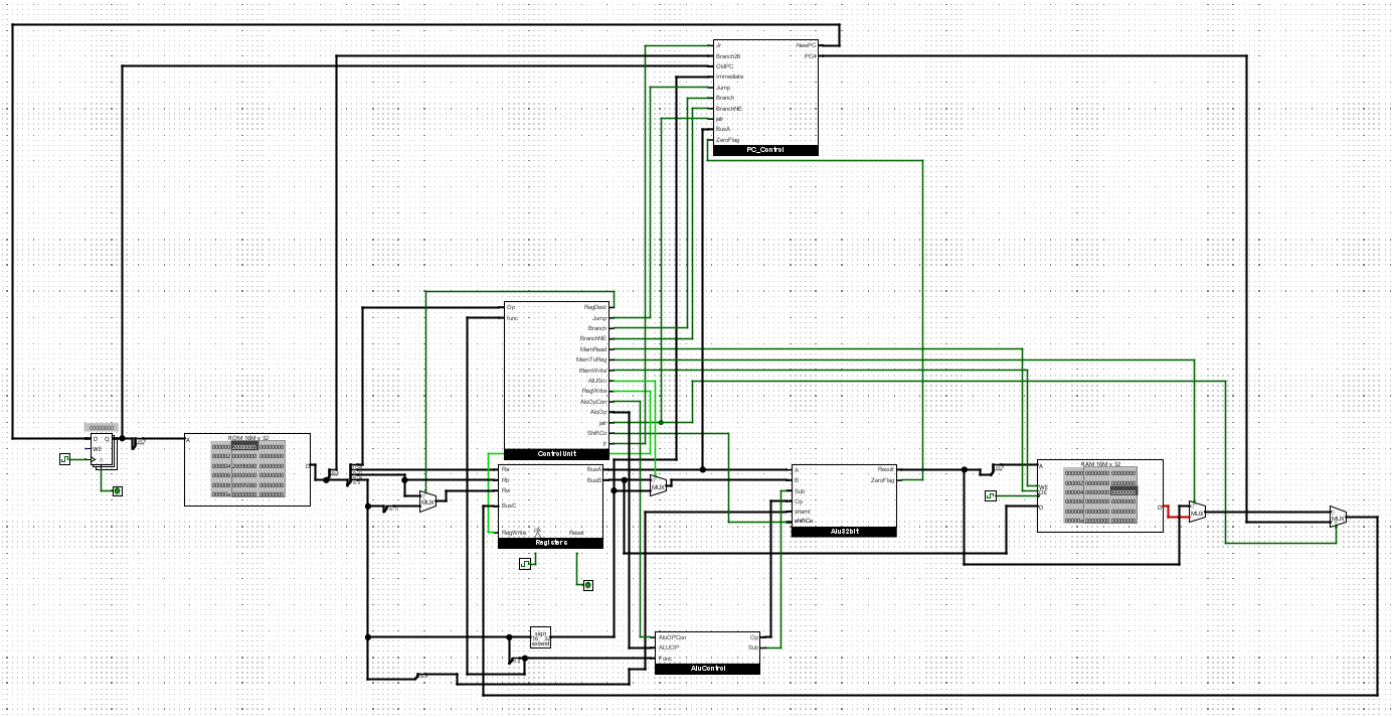

Figure 6: final ciruit diagram on logicim