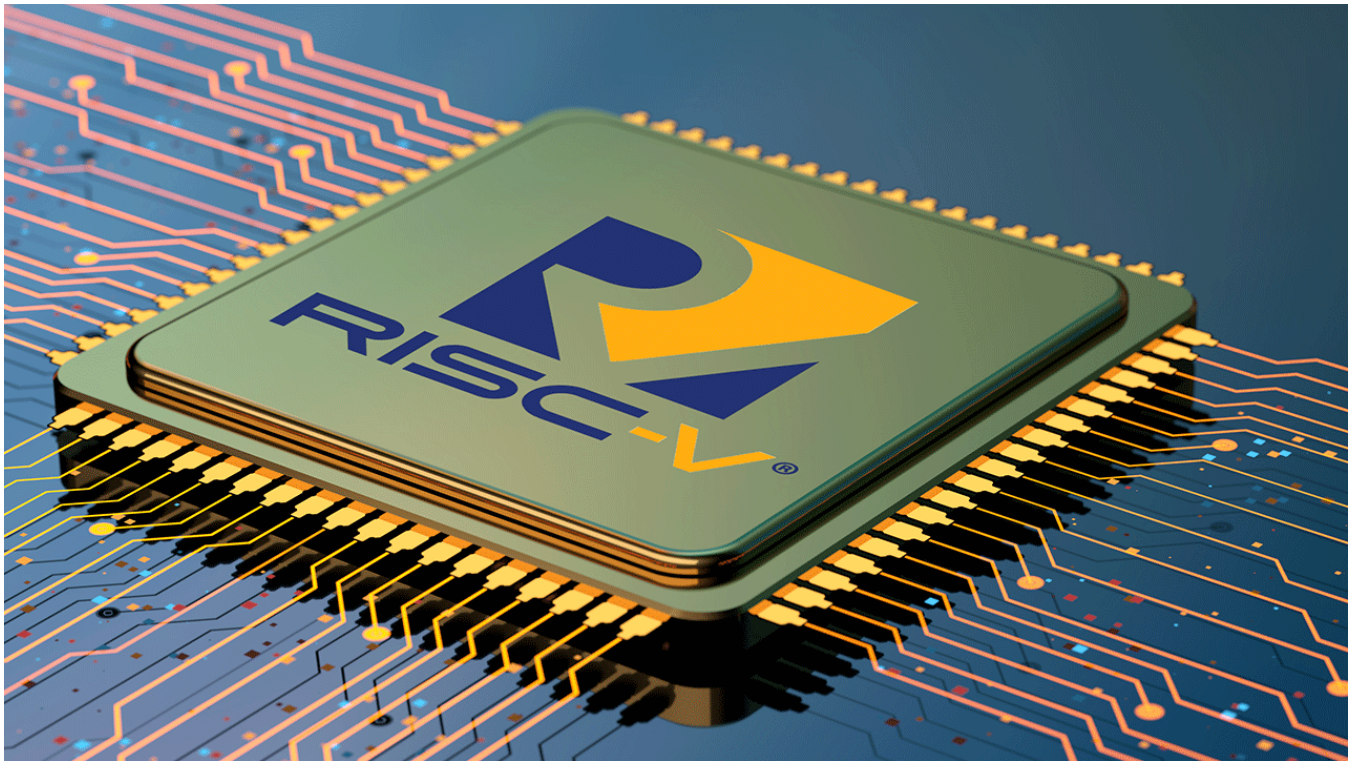
# RISC-V Single Cycle Processor RTL Design

## Using Verilog




**1. Introduction**

This document provides an overview of a single-cycle RISC-V processor implementation based on the provided architecture diagram. The processor supports a subset of the RISC-V instruction set architecture (ISA).

RISC-V (Reduced Instruction Set Computer – V) is an open-source ISA that is designed to be simple and flexible. It supports various extensions (e.g., integer, floating-point, atomic) and has gained popularity for educational purposes, research, and even commercial applications.

**2. Architecture Overview**

The processor architecture consists of the following key components:

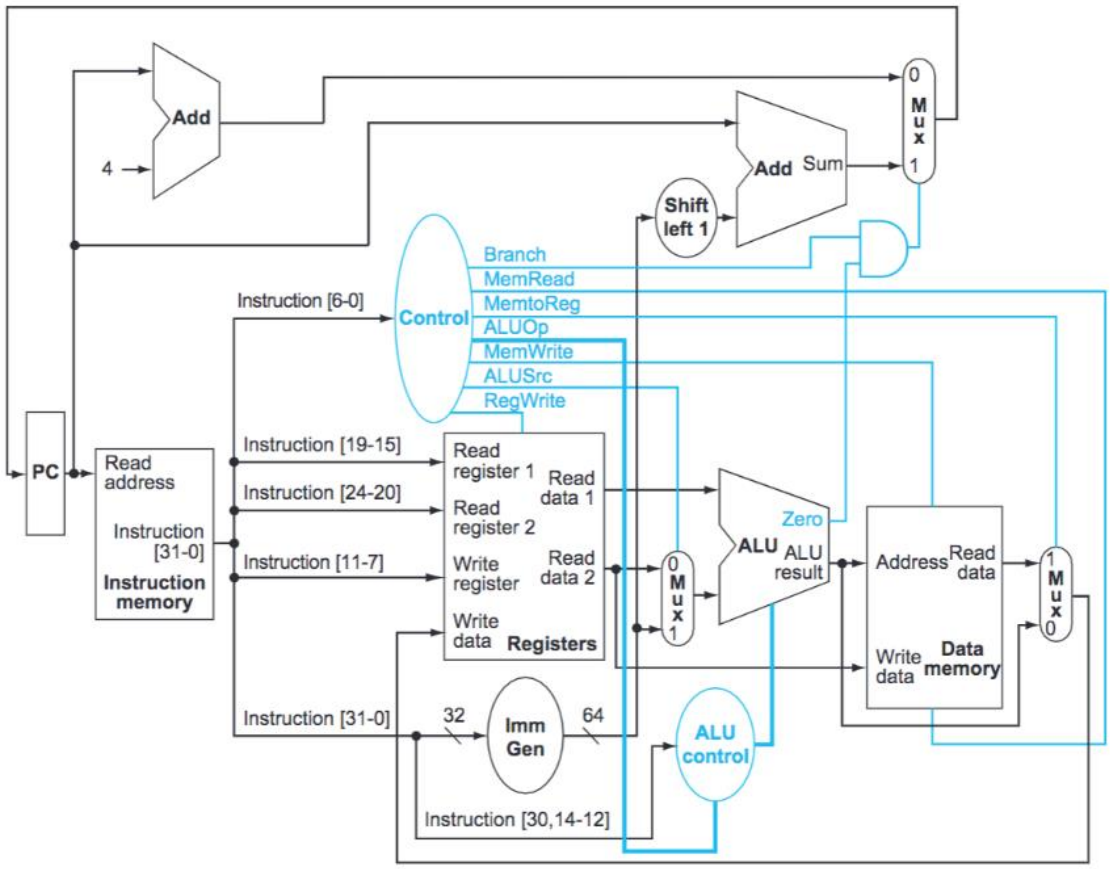
- **Instruction Memory:** Stores the instructions to be executed.
- **Program Counter (PC):** Keeps track of the address of the next instruction to be fetched.
- **Instruction Memory:** Holds the currently fetched instruction.
- **Control Unit:** Generates control signals to coordinate the operation of other components.
- **Register File:** Stores data and program counters.
- **ALU:** Performs arithmetic and logical operations.
- **Data Memory:** Stores data that can be accessed by the processor.
- **Multiplexers:** Select the appropriate data paths based on the control signals.

**3. Instruction Set**

The processor supports the following RISC-V instructions:

- **Load-Store Instructions:** Load data from memory to registers and store data from registers to memory.
- **Arithmetic and Logical Instructions:** Perform arithmetic and logical operations on register data.
- **Branch Instructions:** Branch to a different instruction based on a condition.
- **Jump Instructions:** Jump to a specific instruction address.

The following diagram is a single-cycle implementation of a subset of RISC-V instructions:




- *Single-cycle operation of RISC-V instruction set.*
- *Datapath designed to support data transfers required by instructions.*
- *Controller causes correct transfers to happen.*

**4. Limitations**

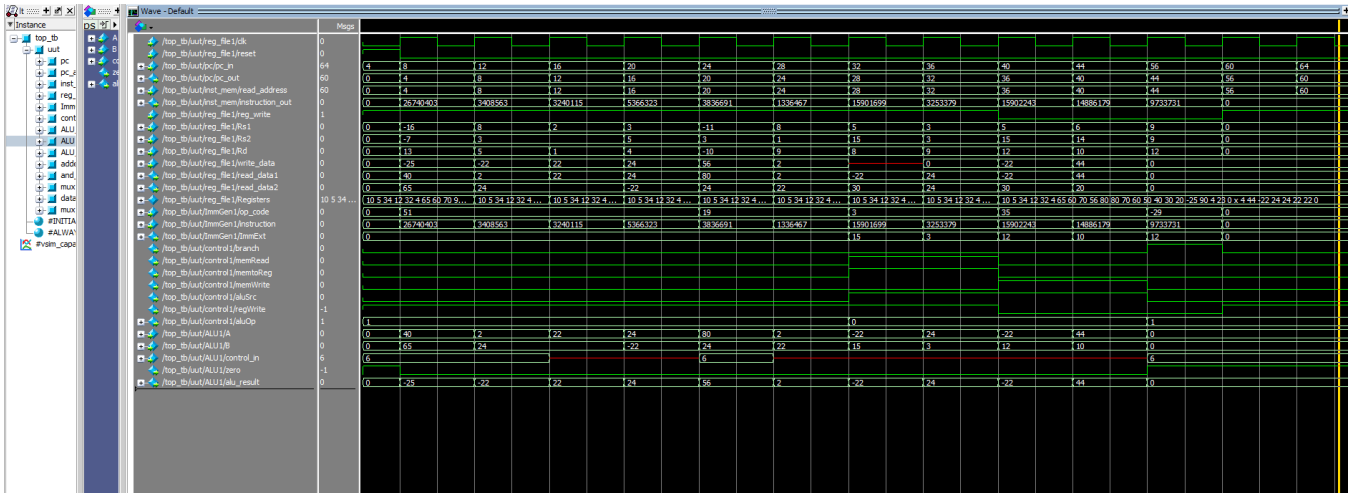
Due to the single-cycle design, the processor's performance is limited by the longest stage in the pipeline. This can lead to performance degradation for instructions that require multiple clock cycles, such as memory access and complex ALU operations.

**********

*For a deeper understanding of RISC-V architecture and design principles, refer to the book 'Computer Organization and Design RISC-V Edition: The Hardware/Software Interface, 2nd Edition' by David Patterson and John Hennessy.*

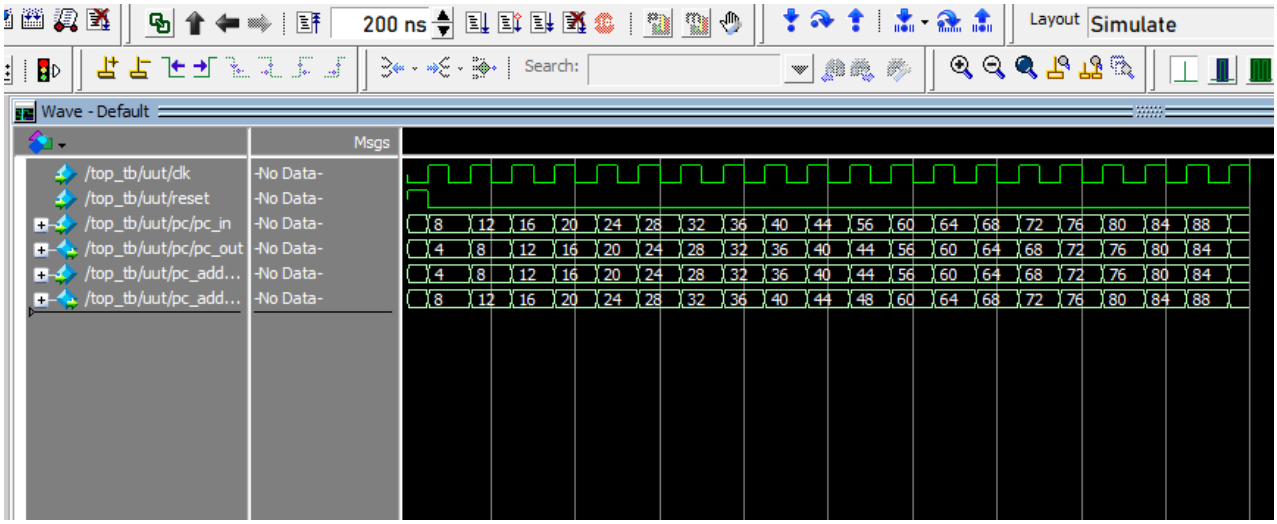
## 5. Simulation

- **Full Operation**

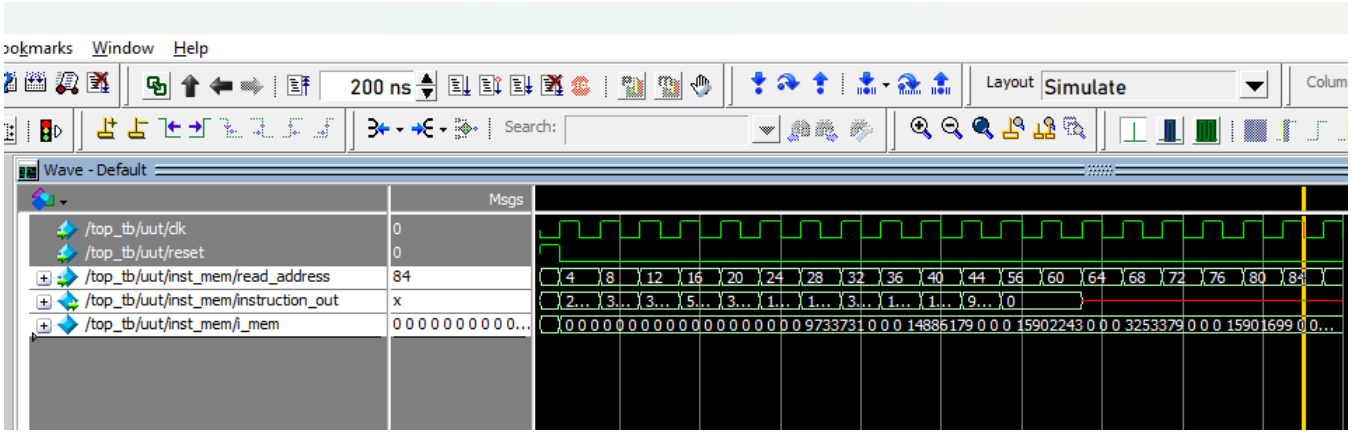


- **Program Counter (PC):**

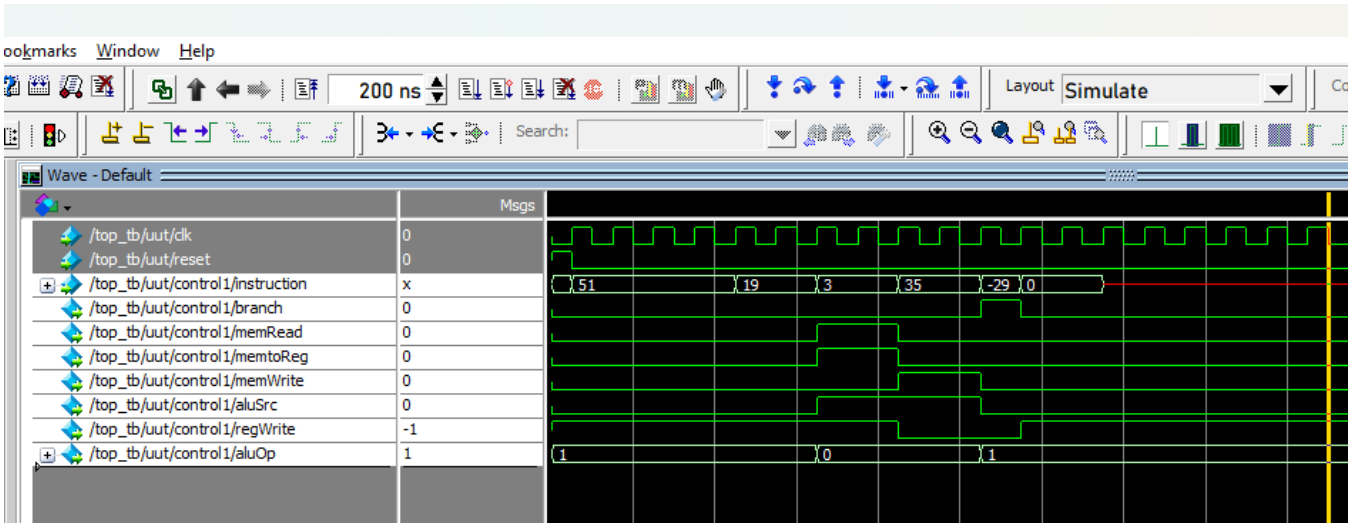
The Program Counter holds the address of the current instruction being executed. After each cycle, it increments to point to the next instruction unless altered by jump or branch instructions.



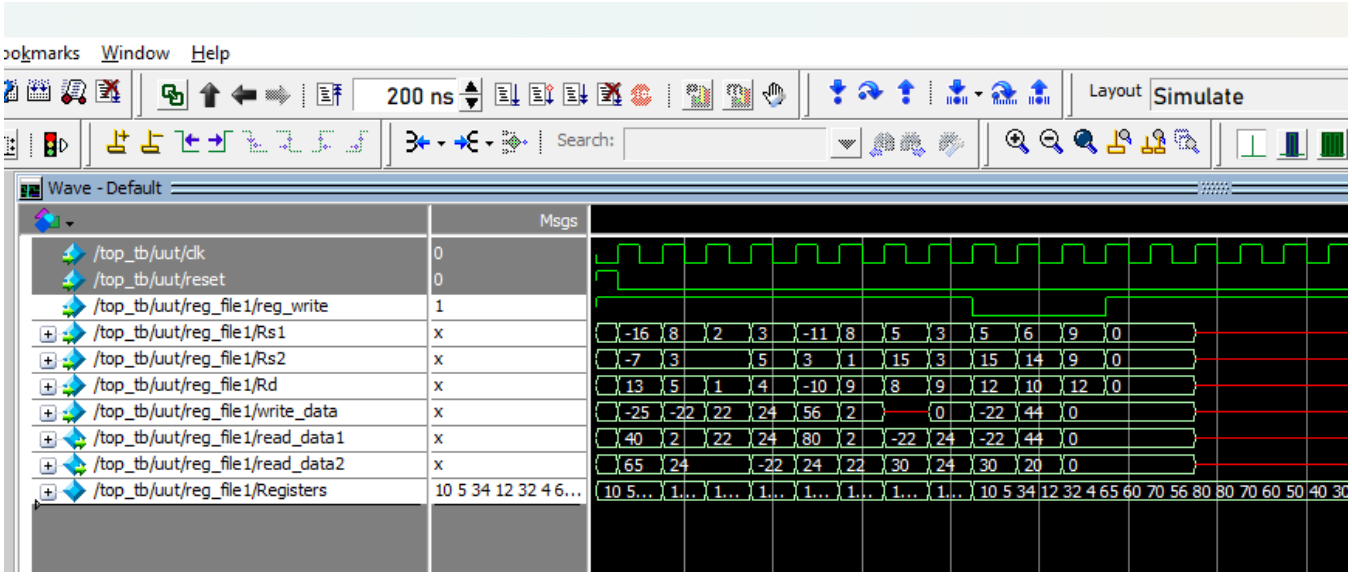

- **Instruction Memory:** This module stores the machine instructions of the program. The PC fetches the current instruction from this memory. The fetched instruction is then decoded to control the operation of the processor.



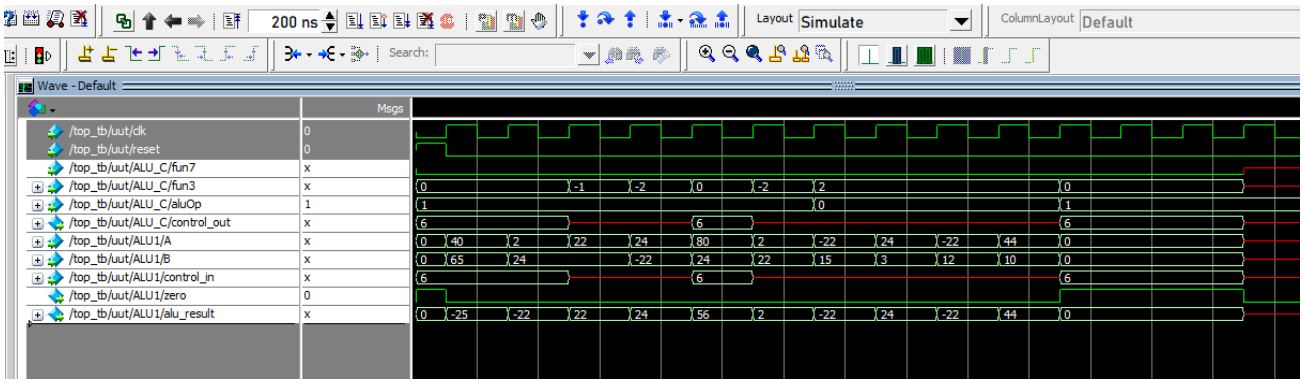

- **Control Unit:** The control unit generates signals to coordinate the flow of data within the processor. It interprets the opcode of the fetched instruction and sets the control signals to drive the ALU, memory access, and data routing in multiplexers.



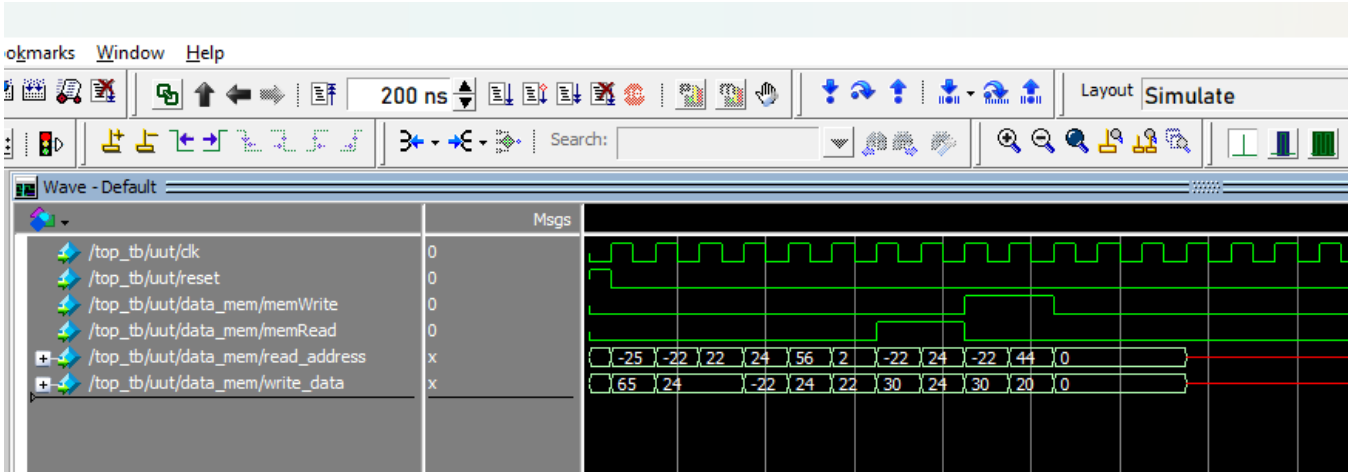

- **Register File:** The register file contains 32 general-purpose registers (x0 to x31). It allows reading two registers and writing one register simultaneously. The register at x0 is hardwired to zero, as per RISC-V convention.



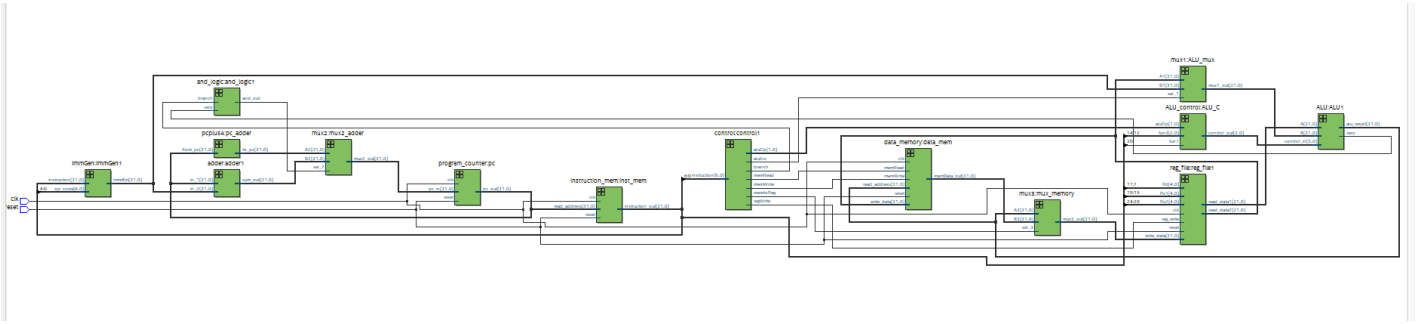

- **ALU (Arithmetic Logic Unit):** The ALU performs arithmetic and logical operations, such as addition, subtraction, AND, OR, and comparisons. The ALU's operations are determined by control signals based on the instruction type (e.g., R-type, I-type).

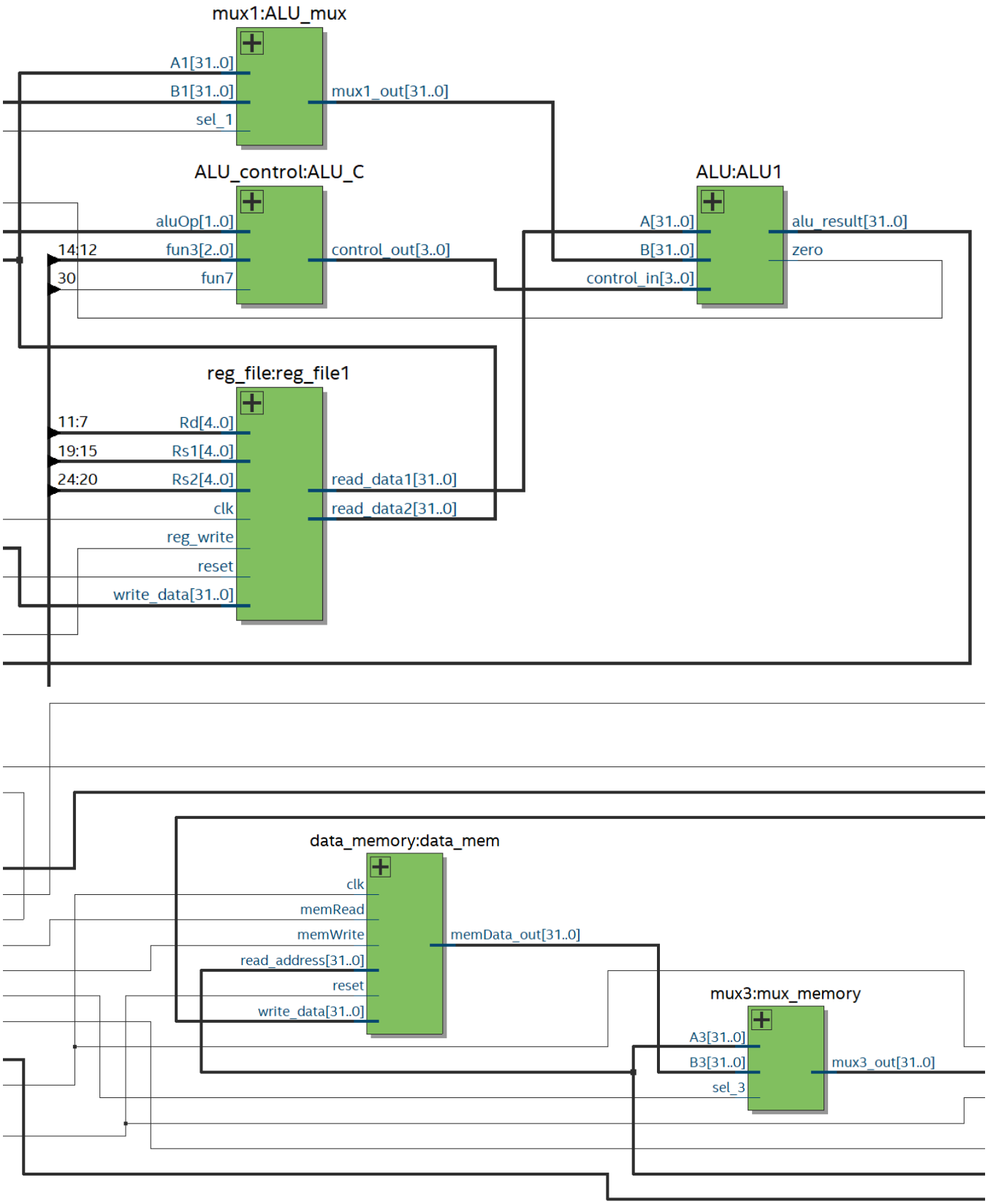


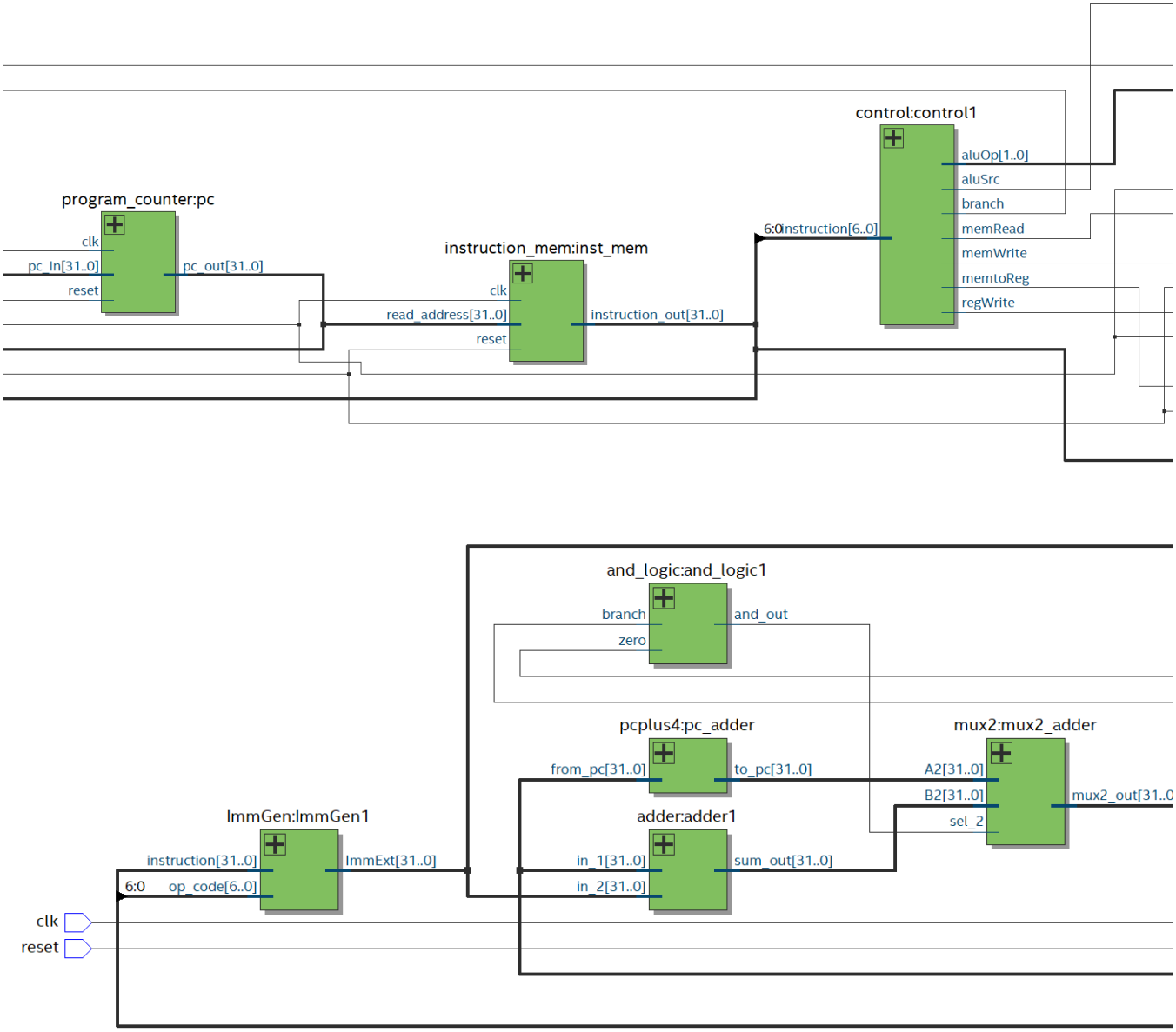

- **Data Memory:** Data memory is used to perform load and store operations. It reads data from memory on load instructions and writes data to memory on store instructions, based on the address computed by the ALU.




## 6. RTL Synthesis Block Diagram




mux1:ALU_mux
A1[31..0]
B1[31..0]
sel_1
mux1_out[31..0]
ALU_control:ALU_C
aluOp[1..0]
14:12
fun3[2..0]
30
fun7
control_out[3..0]
ALU:ALU1
A[31..0]
B[31..0]
control_in[3..0]
alu_result[31..0]
zero
reg_file:reg_file1
11:7
Rd[4..0]
19:15
Rs1[4..0]
24:20
Rs2[4..0]
clk
reg_write
reset
write_data[31..0]
read_data1[31..0]
read_data2[31..0]
data_memory:data_mem
clk
memRead
memWrite
read_address[31..0]
reset
write_data[31..0]
memData_out[31..0]
mux3:mux_memory
A3[31..0]
B3[31..0]
sel_3
mux3_out[31..0]




## 7. Future Work

To improve the processor's performance, future work could focus on implementing a pipelined architecture, which can overlap the execution of multiple instructions. Additionally, adding more complex instructions and support for floating-point operations could extend the processor's capabilities.

## 8. Conclusion

This document provides a brief overview of a single-cycle RISC-V processor. The processor is a good starting point for understanding the basic principles of processor design and can be used as a foundation for more complex designs.