

# Design and Implementation of a Single-Cycle RISC-V Processor

Ahmed Mohamed Abdelmaboud, Yahia Hany Gaber, Omar Hesham Desouky, Mahmoud Khaled, Mazen Ahmed

## Abstract

This paper presents the design and implementation of a 64-bit single-cycle RISC-V processor. The processor executes each instruction in a single clock cycle and supports arithmetic and logic operations, memory access instructions, and conditional branching. The design follows the standard single-cycle RISC-V datapath, with a merged Main Control Unit and ALU Control Unit. All modules were implemented using Verilog HDL and verified through simulation.

## Index Terms

RISC-V, Single-Cycle Processor, Computer Architecture, Verilog, CPU Design

## I. INTRODUCTION

The purpose of this project is to design and implement a single-cycle RISC-V processor as part of a Computer Architecture course. In a single-cycle architecture, every instruction is fetched, decoded, executed, and completed within one clock cycle. While this approach is not optimized for performance, it provides a clear and educational view of processor datapath and control design.

The processor supports R-type arithmetic and logic instructions, I-type arithmetic instructions, memory access operations, and conditional branching.

## II. SUPPORTED INSTRUCTION SET

The processor supports the following RISC-V instruction categories:

Instruction Type	Instructions
R-Type Instructions	add, sub and, or, xor sll, srl
I-Type Arithmetic Instructions	addi, andi, ori, xori
Shift Instructions	Immediate shifts: slli, srli Register-based shifts: sll, srl
Memory Access Instructions	Load doubleword (ld) Store doubleword (sd)
Branch Instruction	beq

## III. DESIGN METHODOLOGY

A modular design approach was followed. Each processor component was implemented as a separate Verilog module and verified using an individual testbench. After verification, all modules were integrated into a top-level CPU module.

The ALU Control logic was merged with the Main Control Unit to simplify control signal generation.

## IV. CONTROL UNIT

The Control Unit decodes the instruction opcode, funct3, and funct7 fields and generates all necessary control signals, including RegWrite, MemRead, MemWrite, ALUSrc, MemToReg, Branch, and ALUControl.

TABLE I  
FULL CONTROL UNIT TRUTH TABLE

Instruction	Opcode	funct3	funct7	RegWrite	MemRead	MemWrite	MemToReg	Branch	ALUSrc	ALUctl	pc_src	imm_type
ADD	0110011	000	0000000	1	0	0	0	0	0	0010 (ADD)	00	-
SUB	0110011	000	0100000	1	0	0	0	0	0	0110 (SUB)	00	-
AND	0110011	111	0000000	1	0	0	0	0	0	0000 (AND)	00	-
OR	0110011	110	0000000	1	0	0	0	0	0	0001 (OR)	00	-
XOR	0110011	100	0000000	1	0	0	0	0	0	0011 (XOR)	00	-
SLL	0110011	001	0000000	1	0	0	0	0	0	0100 (SLL)	00	-
SRL	0110011	101	0000000	1	0	0	0	0	0	0101 (SRL)	00	-
SRA	0110011	101	0100000	1	0	0	0	0	0	1001 (SRA)	00	-
SLT	0110011	010	0000000	1	0	0	0	0	0	0111 (SLT)	00	-
SLTU	0110011	011	0000000	1	0	0	0	0	0	1000 (SLTU)	00	-
ADDI	0010011	000	-	1	0	0	0	0	1	0010 (ADD)	00	000
ANDI	0010011	111	-	1	0	0	0	0	1	0000 (AND)	00	000
ORI	0010011	110	-	1	0	0	0	0	1	0001 (OR)	00	000
XORI	0010011	100	-	1	0	0	0	0	1	0011 (XOR)	00	000
SLLI	0010011	001	0000000	1	0	0	0	0	1	0100 (SLL)	00	000
SRLI	0010011	101	0000000	1	0	0	0	0	1	0101 (SRL)	00	000
SRAI	0010011	101	0100000	1	0	0	0	0	1	1001 (SRA)	00	000
SLTI	0010011	010	-	1	0	0	0	0	1	0111 (SLT)	00	000
SLTIU	0010011	011	-	1	0	0	0	0	1	1000 (SLTU)	00	000
LW	0000011	-	-	1	1	0	1	0	1	0010 (ADD)	00	000
SW	0100011	-	-	0	0	1	0	0	1	0010 (ADD)	00	001
BEQ	1100011	000	-	0	0	0	0	1	0	0110 (SUB)	01	010
JAL	1101111	-	-	1	0	0	0	0	0	-	10	100
JALR	1100111	-	-	1	0	0	0	0	1	0010 (ADD)	10	000
LUI	0110111	-	-	1	0	0	0	0	1	0010 (ADD)	00	011
AUIPC	0010111	-	-	1	0	0	0	0	1	0010 (ADD)	00	011

## V. TEST PROGRAM

To verify correct functionality of the single-cycle RISC-V processor, a custom assembly test program was written to exercise arithmetic, logical, shift, memory, and branch instructions.

```

addi x0, x0, 7
addi x11, x0, 196
addi x3, x0, 4
add x2, x11, x3
and x24, x2, x11
srl x27, x24, x3
sub x22, x27, x27
sd x27, 0(x0)
ld x19, 0(x0)
beq x19, x27, L1
addi x13, x27, 7
beq x0, x0, END
L1: addi x13, x0, 17
END: addi x0, x0, 0

```

### A. Expected Results

- Register x0 = 0 (hardwired to zero)
- Register x27 = 12
- Register x22 = 0
- Register x19 = 12
- Register x13 = 17
- Memory location mem[0] = 12

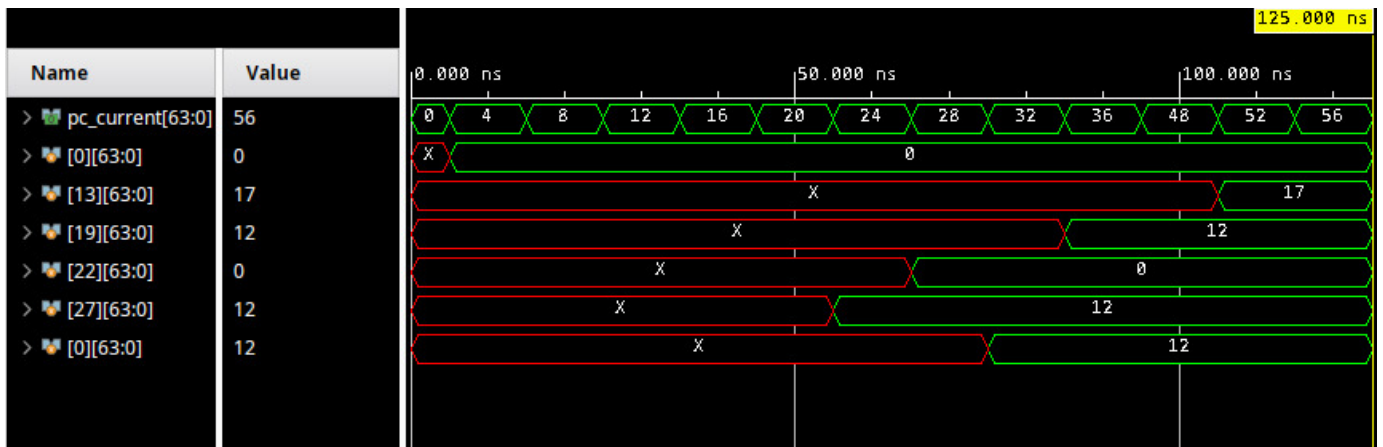


Fig. 1. Vivado simulation waveform the expected final states (registers and memory) and the value of the pc for the test program.

## VI. MODULE IMPLEMENTATION

### A. Arithmetic Logic Unit

The ALU supports arithmetic, logical, and shift operations required by R-type and I-type instructions.

### B. Register File

The register file contains 32 general-purpose 64-bit registers. Register x0 is permanently hardwired to zero. The module supports two asynchronous read ports and one synchronous write port.

### C. Immediate Generator

The Immediate Generator extracts and sign-extends immediates for I-type, S-type, B-type, and shift-immediate instructions.

### D. Program Counter and Branch Logic

The PC updates every clock cycle. Branch target addresses are calculated using the sign-extended immediate shifted left by one bit. Branches are taken when the Branch signal is asserted and the ALU zero flag is set.

### E. Memory Units

Instruction memory is preloaded with RISC-V machine code. Data memory supports 64-bit load and store operations.

## VII. SIMULATION RESULTS

Simulation was performed using Verilog testbenches. The results verified correct PC updates, instruction execution, register writes, ALU outputs, memory operations, and branch behavior. Waveforms demonstrate correct processor functionality.

## VIII. DATAPATH DESIGN

The processor datapath is based on the standard single-cycle RISC-V architecture. It includes the Program Counter (PC), Instruction Memory, Register File, Immediate Generator, Arithmetic Logic Unit (ALU), Data Memory, Control Unit, and branch logic.

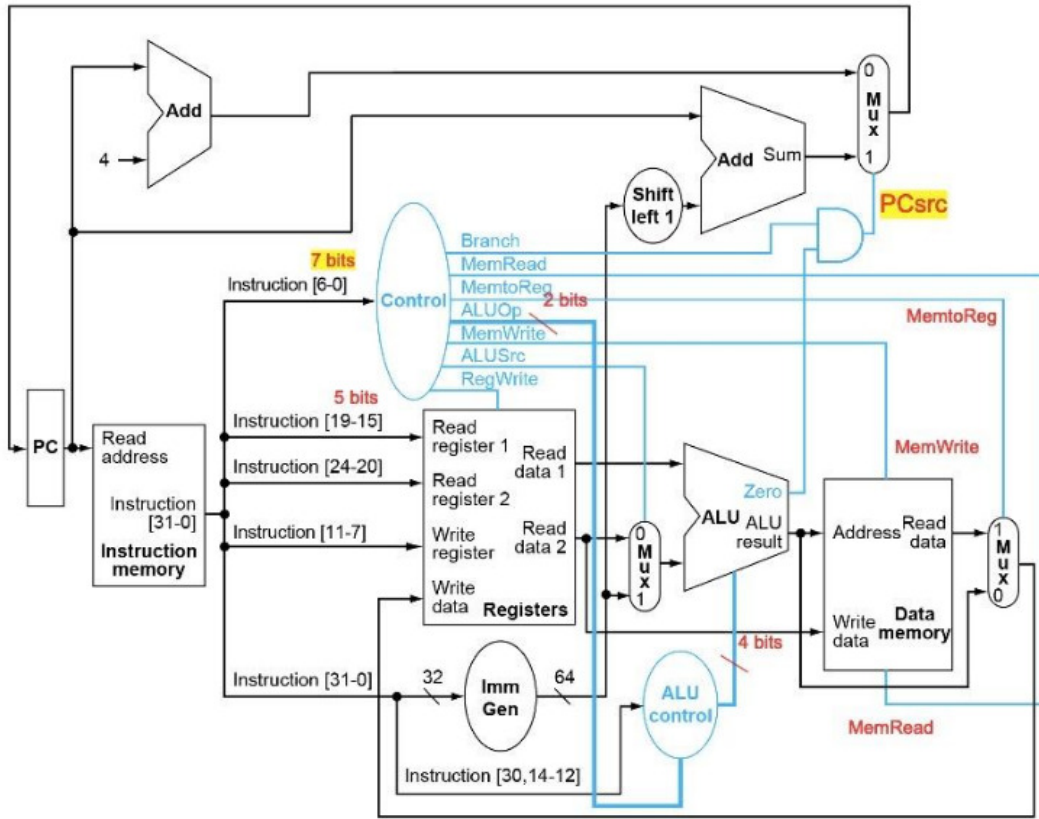


Fig. 2. Single-Cycle RISC-V Datapath

## IX. CONCLUSION

This project successfully demonstrates the design and implementation of a 64-bit single-cycle RISC-V processor. All required instruction categories were supported and verified through simulation. The design provides a strong foundation for understanding more advanced processor architectures.

## REFERENCES

- [1] RISC-V Foundation, *The RISC-V Instruction Set Manual*, Volume I.
- [2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: RISC-V Edition*.