
**ECE 486/586 Winter 2022 Final Project Report**
**Group 9**

# RISC-V ISA Simulator
# Using C++

**Supreet Gulavani**
**PSU ID: 949311031**
**sg7@pdx.edu**

**Yashodhan Wagle**
**PSU ID: 930027080**
**ywagle@pdx.edu**

**Ramaa Potnis**
**PSU ID: 937862463**
**rgp2@pdx.edu**

**Preksha Gandhi**
**PSU ID: 930514375**
**preksha@pdx.edu**

## RV32I Base Integer

The instruction set architecture level simulator simulates the RV32I Base Integer subset of the RISC-V ISA. The implemented instructions are as follows:

**ADD, ADDI, AND, ANDI, AUIPC, BEQ, BGE, BGEU, BLT, BLTU, BNE, JAL, JALR, LB, LBU, LH, LHU, LUI, LW, OR, ORI, SB, SH, SLL, SLLI, SLT, SLTI, SLTIU, SLTU, SRA, SRAI, SRL, SRLI, SUB, SW, XOR, XORI.**

Runs in Verbose and Debug (silent) mode. Arguments are accepted at run time.

- Debug aka Silent mode just displays the register and program counter at the end of execution
- In verbose mode, register, program counter and the type of instruction executed is displayed.

## Extra Credit

1. **RV32M Multiply Extension**

   Implements multiply, divide operations on signed and unsigned numbers. REM instruction returns the remainder value from the division operation.

   **MUL, MULH, MULHSU, MULHU, DIV, DIVU, REM, REMU**

2. **Step and Breakpoint**

   This provides the ability to step through the instructions as they are executed in order to view the registers, program counter and memory.
   Multiple breakpoints can be created during execution by providing the program counter as the input.
   **The step and breakpoints have a different make target**

```
make  debug          :    with debug info
make  release        :    without debug info
make  step           :    stepping execution
make  breakpoint     :    multiple breakpoint execution
```

   Default make target is release i.e no debug info

3. **ECALL Instruction**

   ECALL, also known as Environment Call, is an instruction to make system calls. Three system calls have been addressed: read, write and exit.

# Test Report

In order to test the functionality of the simulator thoroughly, the following test cases were implemented:

- Unit level tests for all instructions
- Testing for misaligned memory references
- Testing the zeroth register
- Invalid instruction.
- Testing the instruction with the correct sign(wherever necessary)

The below illustrates our testing ideology showing the different scenarios simulated:

| Instruction Type | Instruction Name | Address Decoding | Instruction Verification |
|---|---|---|---|
| R | ADD, XOR, OR, AND, SLL, SRL, SKT, SLTU, SUB, SRA | Pass | Pass |
| I | ADDI, XOR, ORI, ANDI, SLLI, SRLI, SRAI, SLTI, SLTIU | Pass | Pass |
| S | SW, SB, SH | Pass | Pass |
| B | BNE, BEQ, BLT, BGE, BLTU | Pass | Pass |
| M | MUL, MULH, MULHSU, MULHU, DIV, DIVU, REM, REMU | Pass | Pass |