RISC-V Assembler & Simulator

CS204 Project

Submitted by:

Ch.V.Rithish Reddy (2023csb1113)

D. Kumar Naidu (2023csb1115)

P.Tharun Sai (2023csb1144)

Date: March 24, 2025

Abstract

This document provides a detailed overview of the two-phase project involving a RISC-V assembler and simulator. **Phase 1** involved converting RISC-V assembly code into machine code, while **Phase 2** focused on executing this machine code using a functional simulator. The document covers the project structure, implementation details, and execution instructions.

Contents

1 Introduction

This project was divided into two phases:

- **Phase 1: Assembler** – Converts RISC-V assembly code into machine code.
- **Phase 2: Simulator** – Executes machine code instructions using a five-stage pipeline simulator.

2 Phase 1: Assembler

Objective: Develop a 32-bit RISC-V assembler.

Key Features

- Reads RISC-V assembly code from a `.asm` file.
- Outputs machine code to a `.mc` file.
- Supports 31 RISC-V instructions.
- Handles both text and data segments.

Assembler Code Example

```
add x1, x2, x3    # Assembly instruction
0000000 01011 01001 000 00100 0110011   # Machine code
```

Listing 1: Assembler Conversion Example

3 Phase 2: Simulator

Objective: Develop a functional simulator that executes machine code.

Execution Pipeline

The simulator implements a five-stage instruction pipeline:

- **Fetch** – Retrieves the instruction.
- **Decode** – Decodes the instruction.
- **Execute** – Executes the instruction.
- **Memory Access** – Accesses memory.
- **Writeback** – Writes back to registers.

4 Directory Structure

The project directory is organized as follows:

Folder	Description
bin/	Compiled simulator binary
doc/	Design documentation
include/	Header files
src/	Source files and Makefile
test/	Test cases in machine code format

5 How to Build

To compile the project, run the following commands:

```
cd src
make          # Compile the project
make clean    # Clean the build files
```

Listing 2: Build and Clean Commands

6 Execution Instructions

To run the simulator with a test case:

```
./myRISCVSim test/simple_add.mc
```

Listing 3: Simulator Execution Example

7 Test Programs

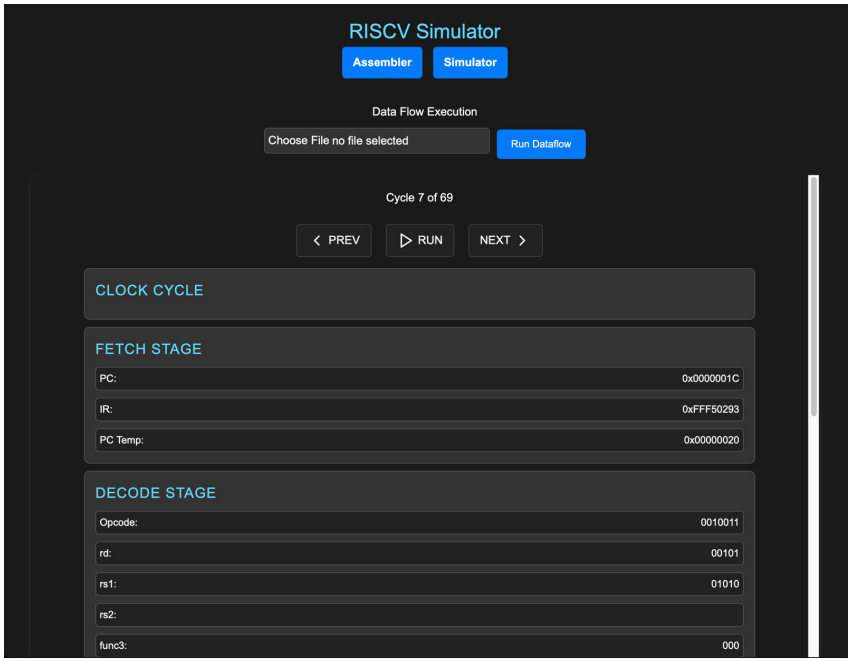
The simulator was tested with the following programs:

- Fibonacci sequence
- Factorial computation
- Bubble sort

8 Results

Key Outcomes

- Correct execution of RISC-V instructions.
- Accurate register and memory management.
- Pipeline stages simulated efficiently.



9 Conclusion

This project successfully implemented a **RISC-V assembler and simulator**. The two-phase approach enabled structured development, starting with machine code generation and culminating in functional simulation. For further details, refer to the design documentation in the `doc` directory.

10 References

- RISC-V Instruction Set Manual.
- CS204 Course Materials.
- Online resources and documentation.