Design and Implementation of an 8-bit Harvard-Style CPU in SystemVerilog

Robert Freeman

October 28, 2025

Abstract

This project presents the design and SystemVerilog implementation of a custom 8-bit CPU with Harvard architecture. The processor uses a compact 8-bit instruction set designed to capture the core functionality of a general-purpose processor while operating within a small instruction size. A program designed to calculate the Fibonacci sequence is included to demonstrate the processor's capabilities.

1 Introduction

Microprocessors are a fundamental building block of many devices and machines. The goal of this project was to create a processor with a compact 8-bit instruction set to simulate design constraints while still being able to perform a full range of functions for coprocessing or educational purposes.

Key features:

- 8-bit datapath
- Harvard architecture
- Custom instruction set architecture
- Fibonacci sequence demo via controller input

2 Architecture Overview

The CPU consists of a program counter, datapath, control unit, program memory, and data memory. The program and data memory can be safely accessed externally by setting the service mode signal high.

2.1 Datapath

Figure 1 shows the top-level datapath, including the ALU, registers, instruction memory, and data memory.

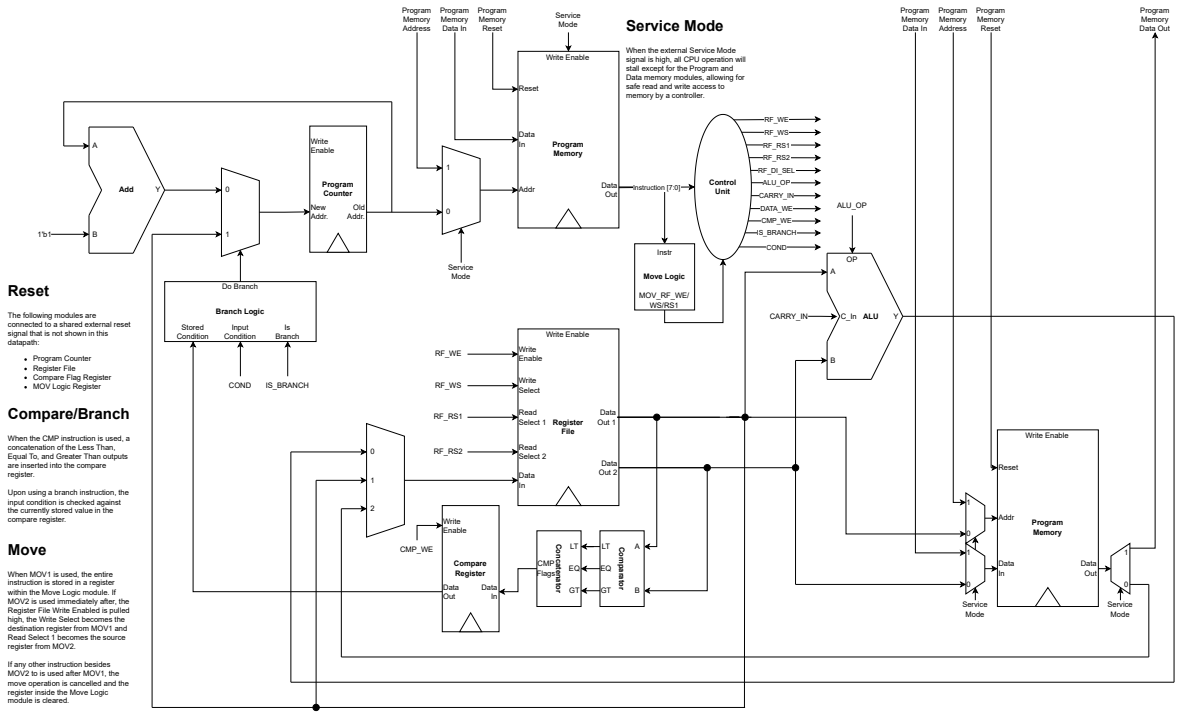

Figure 1: CPU datapath

2.2 Instruction Set

Table 1 summarizes the instruction set and encoding.

Mnemonic	Opcode	Description	Mnemonic	Opcode	Description
ADD Rd, Ra, Rb	0000	Addition	LSL Rd, Ra, Rb	1000	Logical shift left
ADDI Rd, Ra, Imm	0001	Addition with immediate	LSR Rd, Ra, Rb	1001	Logical shift right
SUB Rd, Ra, Rb	0010	Subtraction	LOAD Rd, Raddr	1010	Load from memory
SUBI Rd, Ra, Imm	0011	Subtraction with immediate	STOR Rs, Raddr	1011	Store to memory
AND Rd, Ra, Rb	0100	Bitwise AND	CMP Ra, Rb	1100	Compare two numbers
OR Rd, Ra, Rb	0101	Bitwise OR	B cond, Raddr	1101	Branch if condition met
XOR Rd, Ra, Rb	0110	Bitwise XOR	MOV1 Rdest	1110	Define destination of move
NOT Rd, Ra	0111	Bitwise NOT	MOV2 Rsource	1111	Define source of move

Table 1: Instruction Set Architecture (ISA)

3 Implementation and Verification

The design was written in synthesizable SystemVerilog and verified through Verilator cycle-accurate simulation.

Figure 2 shows a simulation waveform demonstrating the retrieval of the Fibonacci sequence after it is computed and stored in data memory.

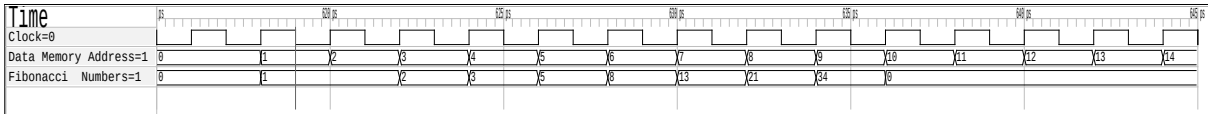

Figure 2: Calculated Fibonacci Sequence read from data memory after execution.

4 Future Extensions

Potential extensions include:

- Reconfiguring the MOV instructions into a single opcode.
- Calculating gate delay with SPICE tools to configure proper clock speed.
- Implementing two-stage pipelining to increase instruction throughput.
- Adding a stack pointer and link register.

Full source code and documentation are available at: `https://github.com/RobsEvilLairOfCode/8-Bit-CPU-Core`