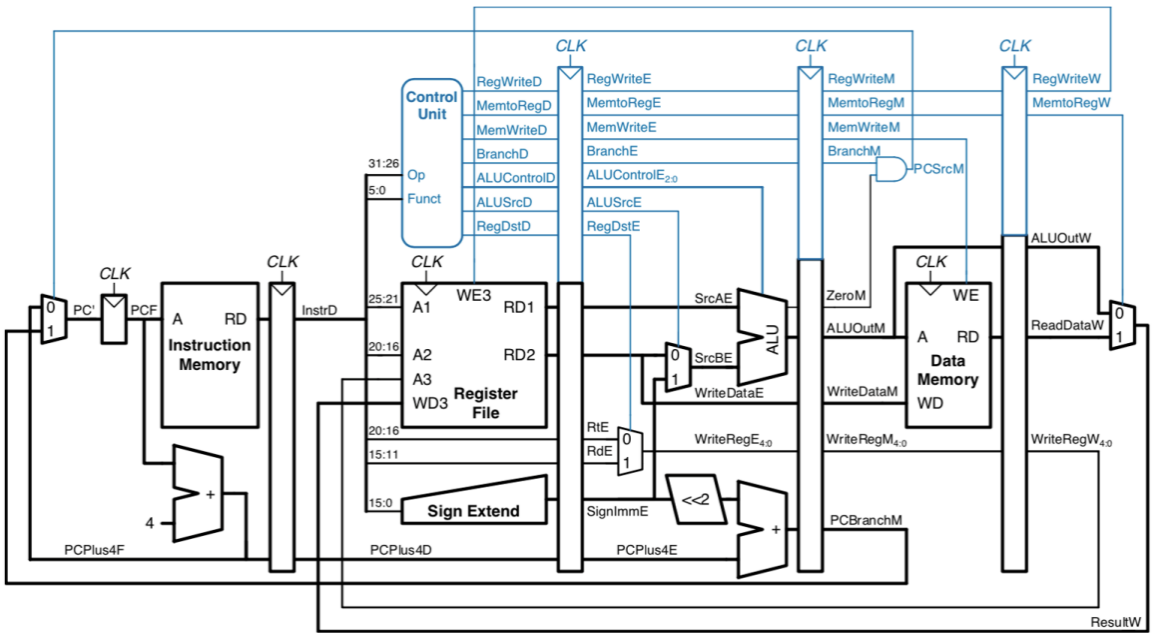
Assignment #4 Documentation

118010488
Hyeonwook Kim

Overview

The following programming project is an implementation of a 5 cycle MIPS CPU. Icarus Verilog was used to simulate the hardware components. The code depends on the Instruction Memory and Data memory files given, as well as a slightly modified version of the ALU simulation from Assignment #3.



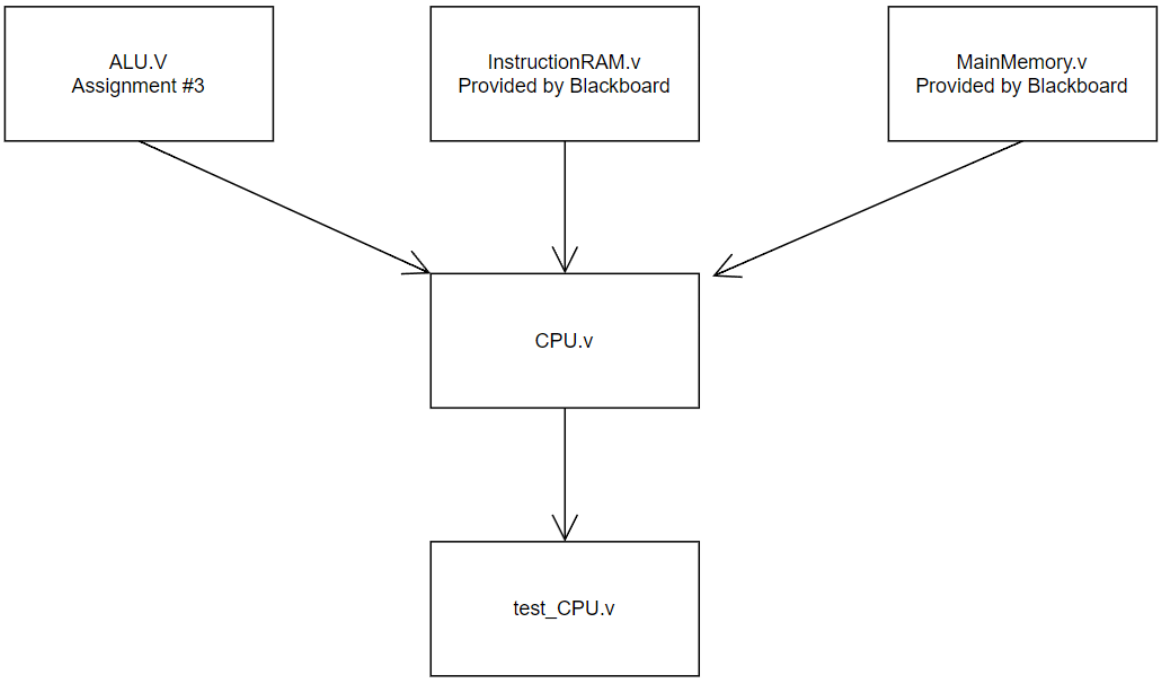
When the PC passes through the Instruction memory, the corresponding instruction given from the address is sent to the Control Unit, Register File and Sign Extender. The PC is also added by 4 at this stage. The Opcode and Function Code are sent to the Control Unit to first determine what pins should be activated. The Control pins are activated accordingly. Then the Register file takes the register address inputs and gives out the appropriate outputs from the Register Addresses. After passing through a multiplexer, the signal RD2 is again met up with RD1 in the ALU, where the appropriate calculations are made. If the data is to be read or written in relation to the main memory, the ALU outputs the address into the Data memory, and RD2 is responsible for the contents of the input data. MemWrite from the main control unit determines if the data should be written to memory. After the ALU output and the Read Data are sorted through another multiplexer, RegWrite determines if the data should be written to the register or not. Finally, if there were any jump or branch instructions, the Control Unit determines which PC is to be input back into the instruction memory, and this ends the cycle. A for loop continues this cycle until all instructions have been parsed or the instruction reads an instruction of full 1's. The Instruction memory and Data memory code have largely been unmodified, and takes instruction.bin with 512 lines of binary code as the default. The assumption is that the test file will also be 512 lines. Meanwhile, the ALU code from Assignment 3 has been tweaked to work with 32 registers instead of 2. Overall, the

assignment mirrors the second programming project in its nature, but instead of using C++ / Python, we are using Verilog HDL.

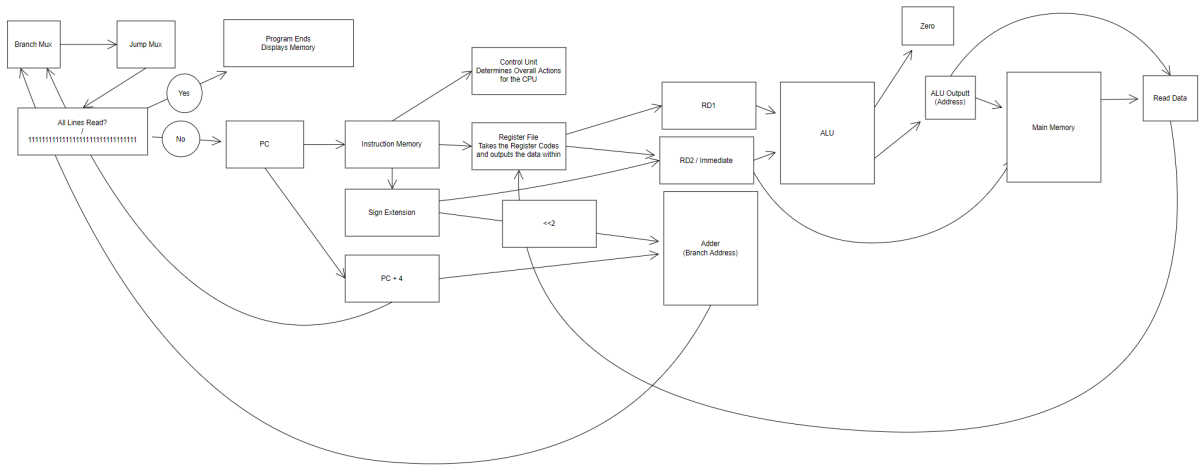
Testing & Results

The testing of the 5 cycle was performed by putting in the binary code inside a file named instructions.bin. The provided instruction memory only seems to be capable of handling instructions that are exactly 512 lines, and so the empty lines were filled with 1's to test the reliability of the simulation. All 8 exercises were completed without errors, and the output memory results were the same as provided by blackboard.

Code Dependencies



Workflow / Flowchart



Although the steps are complex, the mechanism is essentially the same as the wiring shown in the first page.