Multi-Cycle CPU Project Milestone 4: Working System

Austin M. Welch

December 3, 2015

Abstract

This is the final report for the multi-cycle CPU project. It includes a project description, list of instructions completed, state transition diagram, schematic of datapath and control, waveforms, list of files included in the final design, and comments on design choices. I completed the demonstration for my final Verilog design on December 2nd with 7/8 of the extra credit instructions (everything except for JAL).

1 Project Description

This project was to design a multi-cycle CPU in Verilog HDL. The two main components are the control and the datapath, which are the main deliverables for Milestone 2 and Milestone 3, respectively. Because this is a multi-cycle CPU, the number of clock cycles is not constant, and depends on which instruction is fetched from memory. It is not pipelined, so it will process one instruction at a time. The shortest instruction is the NOOP with only two stages, and the longest instructions are the LWI and LW with five stages. Because the CPU takes multiple clock cycles for each instruction it requires registers to hold intermediate values between stages. The instructions are similar to MIPS instruction encoding, but slightly different. A list of Opcodes for each required instruction has been provided. The Opcode is 6 bits long. The two most significant bits designate the type of operation: 00 is jump, 01 is arithmetic/logical R-type, 10 is branch (I-type), and 11 is arithmetic/logical I-type. The other four bits of the opcode will be used by the main control and ALU control. The final goal of this project is to take a given set of instructions, put them in the instruction memory, and have the CPU execute each instruction.

2 Classification of Instructions by Type

R–Type Instructions

MOV:	0	1	0	0	0	0
NOT:	0	1	0	0	0	1
ADD:	0	1	0	0	1	0
SUB:	0	1	0	0	1	1
OR:	0	1	0	1	0	0

AND: 0 1 0 1 0 1
SLT: 0 1 0 1 1 1
XOR: 0 1 0 1 1 0

I−Type Instructions
BEQ: 1 0 0 0 0 0
BNE: 1 0 0 0 0 1
BLT: 1 0 0 0 1 0
BLE: 1 0 0 0 1 1
ADDI: 1 1 0 0 1 0
SUBI: 1 1 0 0 1 1
ORI: 1 1 0 1 0 0
ANDI: 1 1 0 1 0 1
SLTI: 1 1 0 1 1 1
LI: 1 1 1 0 0 1
LUI: 1 1 1 0 1 0
LWI: 1 1 1 0 1 1
SWI: 1 1 1 1 0 0
XORI: 1 1 0 1 1 0
LW: 1 1 0 0 0 0
SW: 1 1 0 0 0 1

J−Type Instructions
NOOP: 0 0 0 0 0 0
J: 0 0 0 0 0 1

The extra credit instructions I completed were:
XOR, XORI, BLT, BLE, LUI, LW, and SW.

3 State Transition Diagram

An overall state transition diagram is shown in figure 1. My design has a total of 18 states. Each state represents one clock cycle, therefore the shortest instruction is NOOP with two states/clock cycles. The longest instructions are LWI and LW with five states/clock cycles. All of the R-type and I-type (including branch) instructions have four states/clock cycles.

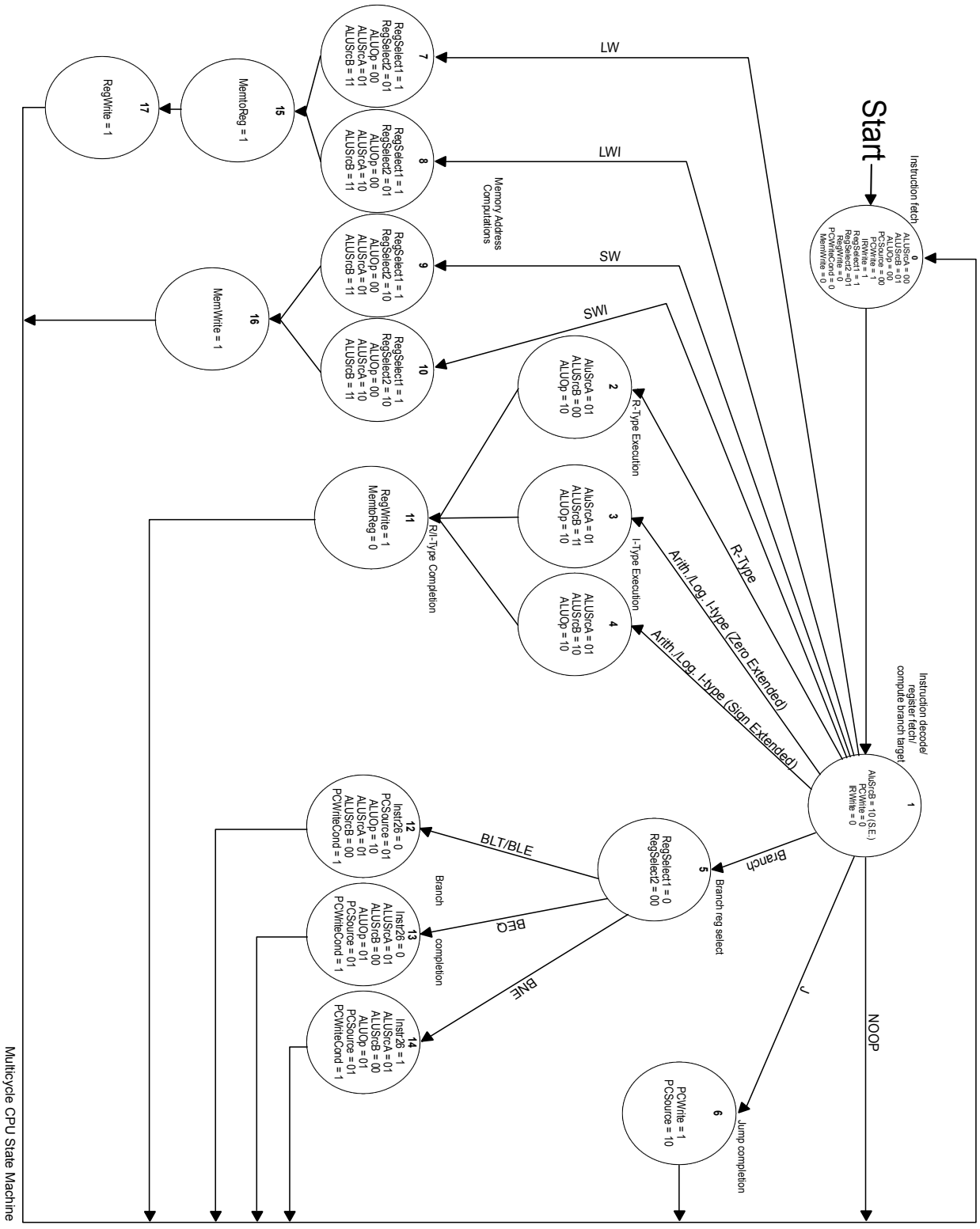

Figure 1: **State Transition Diagram.**

4 CPU Schematic

A schematic for the datapath and control is shown in figure 2.

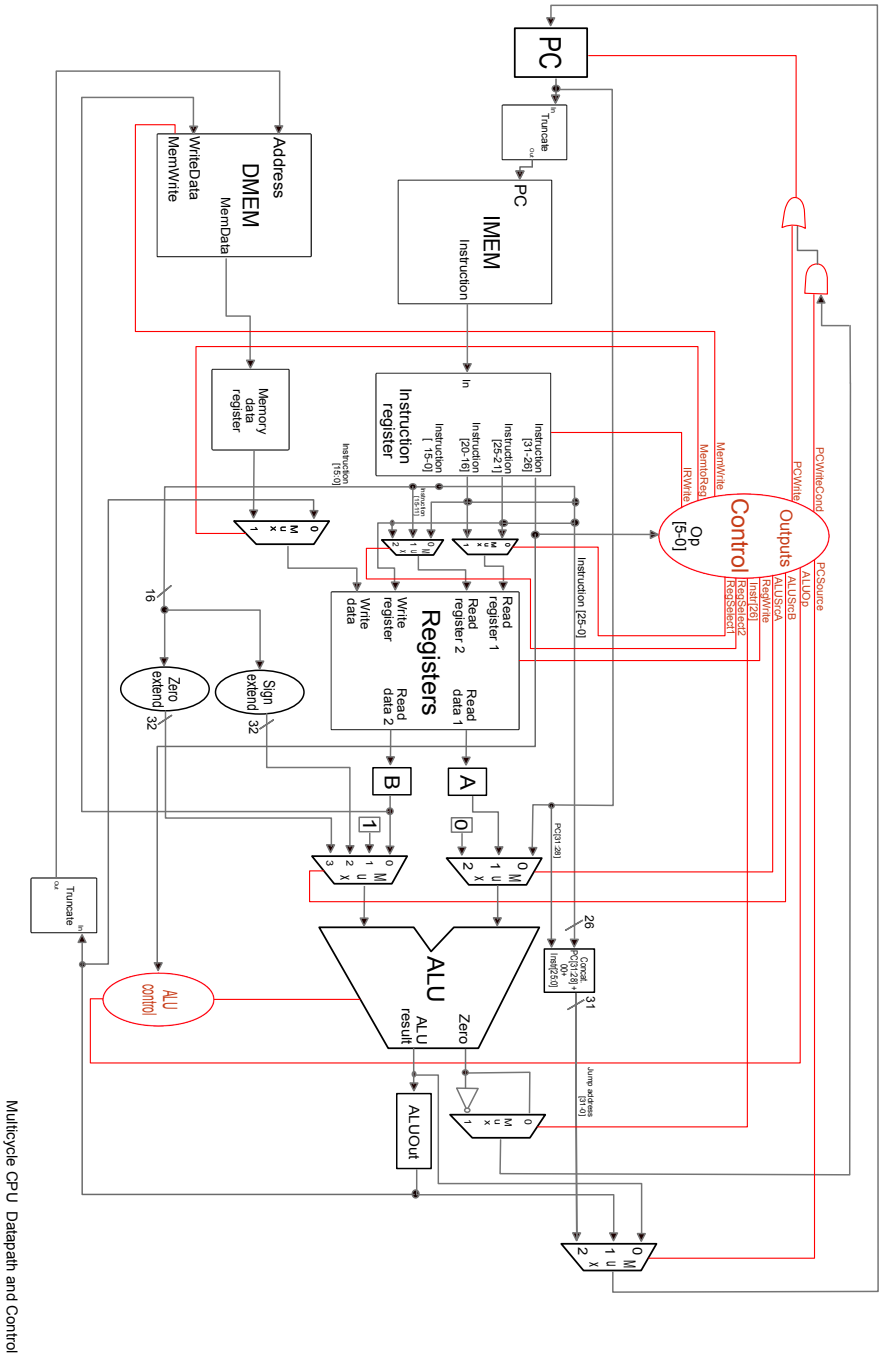

Figure 2: **CPU schematic.** Complete schematic of my multi-cycle CPU design including datapath and control.

5 Waveforms

The Instructions can be viewed from the file CPU_test.v. The waveforms for the required instructions can be seen in figure 3. The extra credit instruction waveforms are shown in figure 4. Each of the two instruction code blocks were commented out and ran separately from the IMEM.v module.

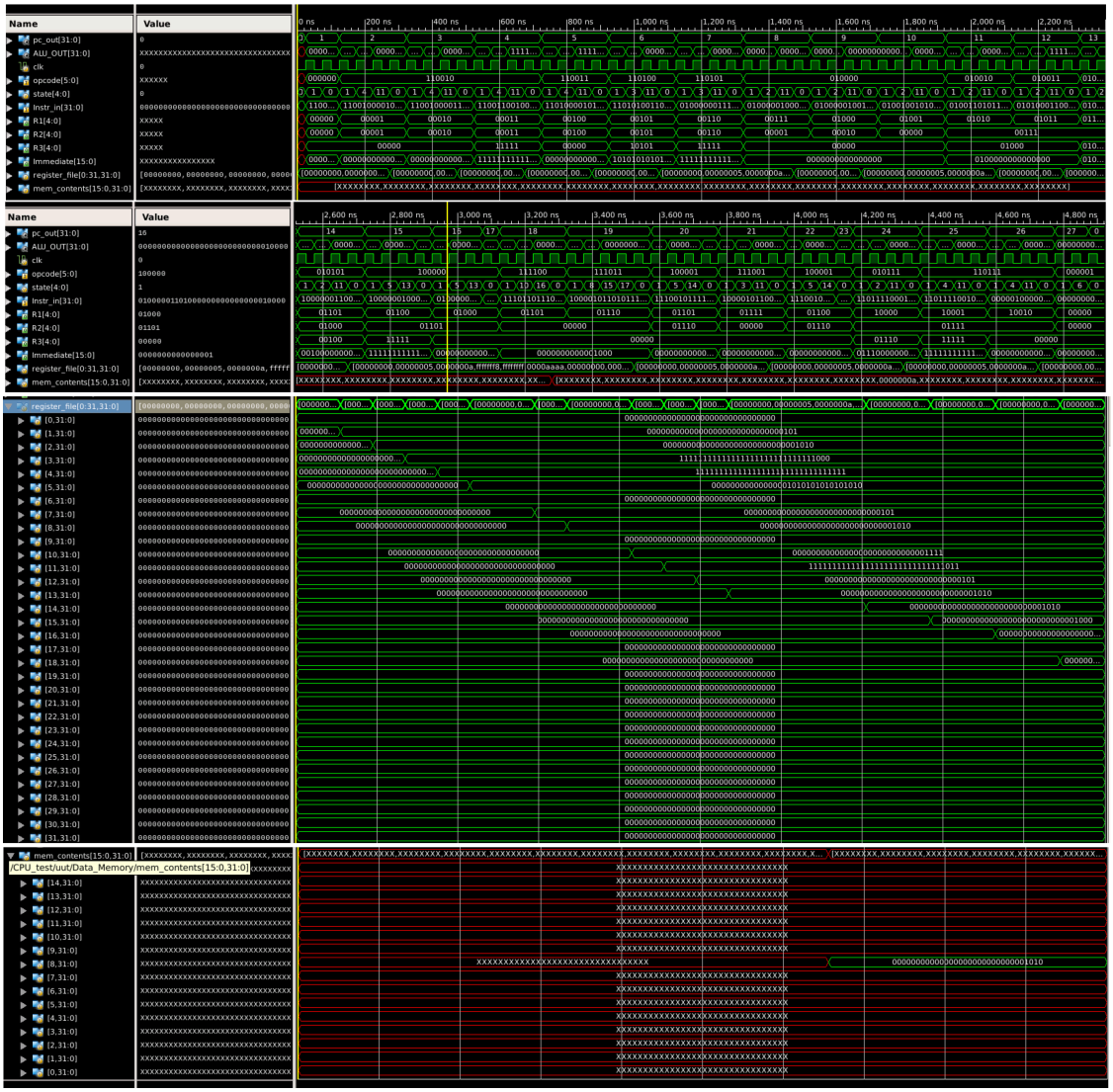

Figure 3: **Main instruction waveforms.** Waveforms for the required instructions showing the program counter, states, instructions, register and data memory contents..

Figure 4: **Extra credit instruction waveforms.** Waveforms for the extra credit instructions showing the program counter, states, instructions, register and data memory contents..

6 List of Included Files

22 project files in total.
Modules: CPU.v Control.v aluControl.v IMem.v DMem.v A_register.v B_register.v ALU.v ALUOut_register.v Instruction_register.v MemoryData_register.v JumpAddress.v nbit_register_file.v PC32bit.v signextender.v zeroextender.v trucate32to16.v mux2.v mux3.v mux4.v
Test benches: CPU_test.v Control_tb.v

7 Comments on Design Choices

When I began this project I expected the schematic to be very similar to the multi-cycle CPU we went over in class, but the decision to not use MIPS encoding format as well as being required to separate the instruction memory from the data memory caused me to have to make more design changes than I originally anticipated. I added an extra multiplexer in front of the main register to accommodate the fact that most of the instructions operated on R2 and R3, while the branch instructions operated on R1 and R2. I realized later that I could have used one less mux because when comparing R1 and R2 for the branch instructions

the order that they are subtracted in doesn't matter, but I think the way I did it made it so I could use less logic for the BLT and BLE instructions. I controlled these branch instructions by sending the entire opcode to the aluControl and modifying the Zero output accordingly if it happened to be one of these rather than doing it entirely through hardware. Overall the project took a considerable amount of time, but I learned a lot, especially about the state machines used for control, which I didn't have any experience with previously.