CompArch Lab 3: CPU

Jacob Riedel, Jenny Vaccaro, Jennifer Wei

November 20, 2015

1 Processor Design

Our team created a single-cycle CPU. We used the same high-level design as we were given in class (Fig. 1). In order to focus on the success of our higher-level machine, we re-used modules from previous CompArch homeworks and labs.

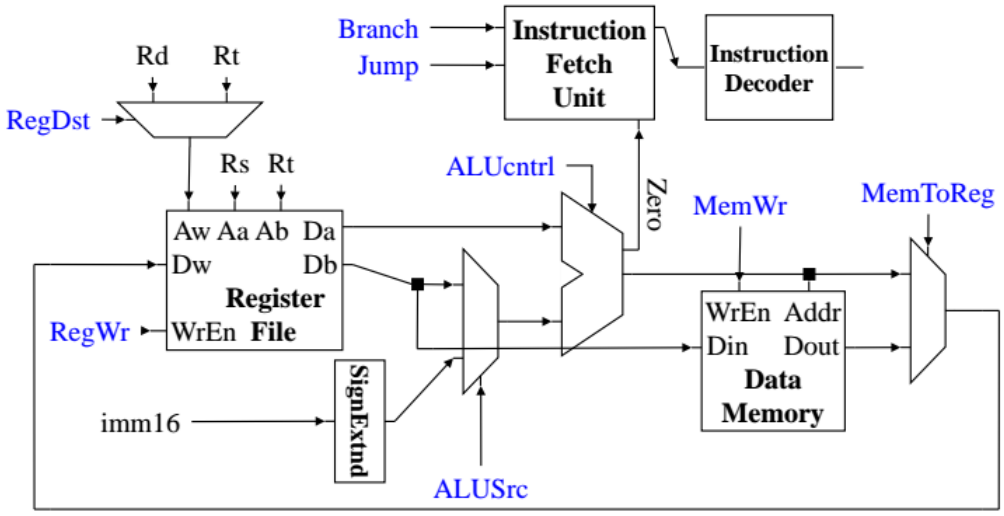

Figure 1: Block diagram of our single-cycle CPU.

The operational codes for the given instructions: `LW, SW, J, JR, JAL, BNE, XORI, ADD, SUB, SLT`: were placed in an LUT. If an instruction was unknown to our machine, the CPU would give an error message and continue on. This way, our CPU would never get "stuck" on an instruction too long. Our goal in this was to allow our CPU to fail gracefully.

2 Testing

2.1 Modules

In order to prevent errors propagating from the lowest levels of our CPU, we tested each module individually in the same spirit as previous assignments. Though we re-used several modules, we were able to improve upon some timing errors and poorly designed flags.

2.1.1 Instruction Fetch Unit

To test the IFU, we tested the three possible PC update cases:

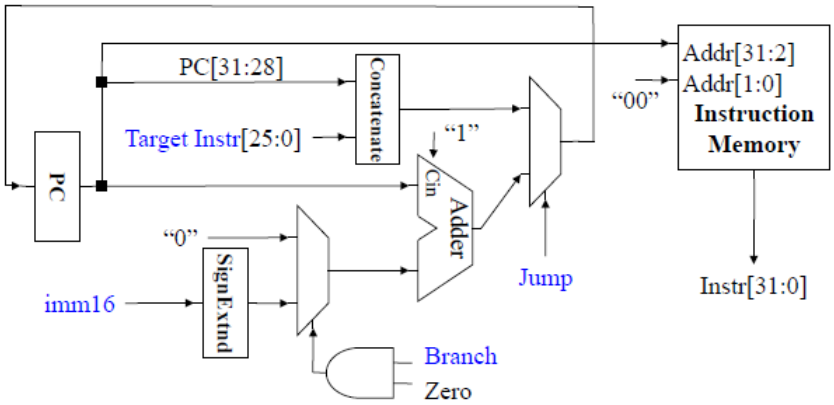

Figure 2: Block diagram of our instruction fetch unit (IFU).

1. PC + 4
2. Branch - where PC + 4 + an inputted amount
3. Jump - where PC jumps to the target address

After testing the three cases sequentially, we tested branch and PC+4 again just to make sure that going from jump to branch and from branch to PC+4 also worked (in addition to PC+4 to branch).

2.1.2 Instruction Decoder

2.1.3 ALU

To test the ALU, we used the tests written in Lab 1.

2.1.4 Register File

To test the regfile, we used the test file we wrote in HW4.

2.1.5 Memory

To test the memory, we looked out the waves of the CPU.

2.2 Assembly

For testing the CPU as a whole in a more realistic setting, we ran a suite of assembly programs written by our classmates. We created an assembly program which determined whether an input number was prime or not.

3 Performance

Since we were unable to get our CPU working, we unfortunately do not have any performance analysis.

4 Reflection

This lab ended up taking longer than expected throughout. We did not end up spending as much time on the lab during the first weekend, and this pushed us back significantly as we had the midterm the week after. Also, as we mentioned in the work plan, members of our team had several interviews over the two weeks of this lab, making it difficult to find time to meet.

In the end, we spent a lot more time than we had expected to on the single-cycle CPU, where we spent a good amount of time writing up the new modules, creating a table to understand how to stitch it all together, and integrating our Verilog with the Assembly programs. We then spent a significant amount of time debugging code (both the test files and the actual files), trying to figure out what was going wrong. We ultimately spent about 30 hours and did not end up getting our CPU to work. Much of that work is done in the 'cpu-debugging-0' branch.