Design Journal

Jacob Scheibe

Milestone 1

Wednesday, September 27th
Time Spent: 1 hour

- Met in class, decided on which architecture to implement. We settled on a load store architecture.
- Divided up tasks to complete by our next meeting, I was assigned to write something for designing performance.
- We also did some general brainstorming as to how we wanted to go about completing milestone 1, and the major observation we made was it would be easier to write relprime in risc-v so we had a good idea of the level of complexity of programs our processor should be able to run.

What I wrote for performance:

We will measure our performance based on versatility and ease of us (programmability) of our processor. The instructions are straight forward and simple to use while maintaining acceptable execution time.

Sunday, October 1st
Time Spent: 4 hours

- Met in the union and worked for just over 4 hours completely finishing milestone 1
- The first 2 hours were spent making the instructions as a team, as well as discussing other important design decisions such as a 5-bit vs 4-bit opcode. We landed on a 4-bit opcode, as we needed more than 8 registers.
- We also had to figure out how to do 16-bit immediate computation, and sign extension, and jump ranges.
- After we finished making those logistical discussions, we began implementing each step in the milestone

The tasks I completed are listed below:

1. Instruction Type Section
2. Part of Measuring Performance
3. Memory Allocation
4. Addition using 16-bit immediate example
5. Instruction Description
6. Register Table

Milestone 2

Wednesday, October 4th
Time Spent: 1 hour

Before Meeting the Professor
10 Minutes
Before our meeting with the professor, we met for about ten minutes to discuss potential questions to ask. During the meeting we asked if we should do multicycle RTL, or if we should wait to complete the RTL so we can do pipelining.

- We decided to complete the RTL in multicycle, and then change it later on, as we hadn't quite learned how to do pipelining yet.

After Meeting the Professor
50 Minutes
After our meeting, there were several topics to discuss in order to fix mistakes we made with our previous milestone, as well as what to do for the next one.

- During our meeting we were given the idea to only have one branch instruction beq, and to complete every branch operation using that. We decided this would be a great idea and would add uniqueness to our instruction set. We spent about 15 minutes writing the algorithm for blt, then after the meeting I made a template with how to complete each branch instruction.

Then we decided who would do what small tasks to fix our processor, and what new tasks for the next milestone. Listed below are those tasks.

Old Tasks

1. Fix document names
2. Fix instructions to increment 2-bits rather than 4-bits
3. Add a recursion example
4. Change instruction type names

New Tasks

1. Template for branch instructions
2. Complete the RTL
3. List and describe the components

Sunday, October 8th
Time Spent: 1 hour

Tasks I Completed Beforehand

1. Branching description and template

2. Jal and Si RTL
3. Changed our memory map to only have stack and dynamic data
4. Components table
5. Removed and fixed branch operations throughout the document to fit the template

During this meeting we discussed what we had done before the meeting, and what still needed to be done, during the meeting certain tasks were finished up. I finished the components table during the meeting.

- We realized we weren't sure how to do our immGen, specifically how many bits it would take in, and whether or not to have a control signal. We weren't sure so I asked the next day in class.
- We also decided that both our jal and jalr would need to store a return address, so we fixed our RTL to do so

The rest of the time was just spent working, and finishing all of our tasks until we thought we were in a good place.

Milestone 3

Wednesday, October 13th
Time Spent: 1 hour

This was a meeting I was not able to attend as I had left the night before to attend a wedding. During the meeting, work to complete by the next meeting was split up. They also decided to continue with a multi-cycle design rather than a pipeline design for a couple of reasons.

- Firstly, pipelining seemed like it would be significantly harder and take up more time as appose to multi-cycle. Second, we learned that pipelining is taught is computer architecture 2, so we would eventually get experience with it anyways. Regardless of these reasons, we still thought pipelining would ensure us max extra feature points, however our professor gave us the idea of doing a more complicated assembler, which we agreed was easier, but still as interesting.

Monday, October 16th
Time Spent: 1 hour

This meeting was relatively brief, outside of class, we went over what progress we had made over the weekend, and decided what we still needed to do. I had implemented part of the datapath and still needed to finish it up. After the meeting, I finished the datapath and I implemented the integration plan. The integration plan was something we had discussed briefly.

- During class, not necessarily a meeting, we had discussion on whether we should have an ALU control or not. Since our ALU operations depends on the Opcode and the immediate, we thought it natural to actually implement the component.

Milestone 4

Time Spent: 5 hours

We didn't end up meeting this week as we all knew what we needed to do. I was assigned the Control and Controls test bench as well as the state diagram. There were some design decisions that I had to make along the way that I hadn't quite considered.

- When making the state diagram, I noticed that we didn't have hardware to support the jalr instruction. I looked at the datapath, and decided to attach a signal from the A register to the mux that decides whether the pc should be incremented by 4, or a new value like in a branch or jump operation. I realized for the jalr that the pc gets the value of whats in register A. Now this could be done by adding 0 to the ALUSrcB register and treating it like a normal branch operation. However, I found this less intuitive than just connecting the value of register A to mux and adding a bit to the control signal. I had also realized that this created an entire other state in the state machine, which was unavoidable.
- While writing the Control test, I had to make an important distinction between testing the control signals or the states themselves. I could test both, but I thought this would be repetitive. Also, I realized that at the lowest level, what actually matters are the control signal values. If I tested the current state, there's a possibility that it could be correct, but the signals wrong.

Milestone 5

Time Spent: 6 hours

We meet once this week, generally to discuss what we needed to do. For this milestone I ensured that the control, the datapath, and the state diagram were all complete.

- When we started making the final top-level file, we had to add input and output, which led to discussion about the best way to do it. We decided we would do something similar to RISC-V, and put the input into a register through an external wire, and do the inverse for the output. To do this however I ended up needing to add two more states
- I also removed a control signal in the datapath called regDst, which was a signal to choose between which register to write data to. I realized that we actually did not need this signal.
- Most of the time this milestone for me involved debugging the control