Team 3v: Design Process Journal

Member: Maura Coriale

Milestone 1:

Meeting 1:
September 28, 2018
I worked with Joy and Matthew to start creating the basis of our instruction-set architecture and began coming up with lists of instructions along with various conventions and necessary functionality. Decided to base it on accumulator, but quickly realized the need to add some aspects of stack.

Meeting 2:
September 29, 2018
Continued working with Joy and Matthew to try and work out problems with our existing design choices, including problems caused by backing up values to the stack. Solved by creating new commands along with small register file.

Meeting 3:
September 30, 2018
Worked with Ben, Joy, and Matthew to flesh out more details of the first milestone. Debated with Matthew and Ben how to best fix our existing problems with our architecture, which concluded with deciding on a two-accumulator architecture. Could not decide whether or not to make it an implicitly or explicitly used system of accumulators; implicit had the cool factor, while explicit was simpler but the syntax was not consistent with the rest of our code and seemed similar to Load-Store.

Meeting 4:
October 1, 2018
Worked with Ben, Joy, and Matthew to make more decisions and revisions based on our feedback from Sid. Decided on a compromise between Sid, Matthew, and Ben's idea with a swap accumulator, which works as a backup for the main accumulator value and the main is implicitly called. Combined our two

instruction types into one for simplicity when assembling, and Ben wrote a basic Python script to interpret the assembly into machine code. I came up with a flag bit to choose whether a command will be done to the accumulators or an accumulator and a stack, which fixed a previous issue with our current approach, which would not work for "and"ing or "or"ing things.

Milestone 2:

Meeting 5:
October 3, 2018
Worked with Ben, Joy, and Matthew to decide our game plan for the next milestone. We split up the commands we have created so far, and each took 6 to turn into RTL. I took the first 6, since we went alphabetically, and created the RTL for aput, sput, aadd, asub, spek, and spop. Some I divided into two, depending on the flagbit (eg. aput, aadd…). We also decided, after some discussion and some advice from upperclassmen, NOT to pipeline our CPU, but still do multicycle.

Meeting 6:
October 5, 2018
Met early in the morning before classes to go over our commands we had made and compare them, made some decisions about how to handle punctuality for team meetings, and decided on some more meeting times for the weekend to work on Milestone 2.

Meeting 7:
October 6, 2018
We each took a type of command and wrote the basic outline of the State Diagram, and each took turns writing it up with everyone else editing and commenting on it as we went along. I worked with stack, since I wrote the RTL for most of the stack commands. I proposed that we have our own "loop" for sp, just like pc in MIPS, since we will have lots of stack operations and will need to change the stack pointer frequently.

Meeting 8:
October 6, 2018
We met up again to go over the rest of the commands types' State Diagrams, and started coming up with the grocery list of sorts for what we need in our circuit diagram. I started organizing all of the state diagrams and editing the ones we had for clarity within our design document.

October 6-9, 2018:
Due to difficulties with schedules, homework, and stress we mostly worked individually on tasks from October 6 - October 9. I worked on the "mashed together" version of the RTL's for the giant RTL table. I also worked on the state tables for each instruction type. I also stressed a lot. I don't remember times, but it was >2 hrs.

Milestone 3:

Meeting 12:
October 15, 2018
We divided the work and decided who was doing what portion of the third milestone. I chose to do multiple smaller portions, since the logic was simple and intuitive, meaning I could focus more on getting the correct syntax in Verilog. I began with the Zero-extender, since it was the simplest of the four.

Meeting 13:
October 16, 2018
We continued working on our individual parts. I ran into lots of syntax errors with Verilog, and got help from Matthew and Ben in class, as well as in the evening. I spent at least 3 hours debugging and working on the modules outside of class.

Meeting 14:
October 17, 2018
After finally getting past some of the syntax issues I was running into while getting acclimated to Verilog again, I was able to finish all the modules and their tests.

Milestone 4:

October 18 - October 23, 2018

Unfortunately, I don't have a daily log for this milestone like the others. I had a lot of work to do for this project and my other classes, and journaling got pushed to the backburner.

This milestone, my responsibility was implementing and testing the control unit, which consisted of creating a Verilog module containing approximately 42 if statements that would set control signals based on the OPCODE and flagbit sent in. I pretty much reversed that process for the test bench, where I set the OPCODE and flagbit and checked the output values for each possible OPCODE and accompanying flagbit.

There's still more to do for implementing the clock and cycle functionality of the control unit, which I'm hoping Matthew will help me with.

So far, I've spent at least 8 hours creating and testing the Control Unit. I ran into several weird bugs along the way, which I was able to fix, learning lots more about Verilog syntax along the way. I'm getting a little nervous about integrating the whole thing together; we have our integration started, but hardware is not our strong suit so I'm a little worried that it may take us longer than we'd like.