Eli Granade’s Journal

MILESTONE4:

10/21/23

Started implementation of ALU. Ran into immediate problem of how to start the files. I dove back into our lab 4 files to figure out how to make it react to the clock. Realized that we may have forgotten a = zero output, take a look and see if we need it. Upon beginning to write the test benches, I realized that I had no clue once more. I pulled our tb_counter tests from the same lab. In my creation of the ALU code, I designated any inputs on the opcode input that aren’t defined output a zero on the mux. This may allow for a noop command. I would put this in the zero slot, so an all zero command could be inputted for this noop command. Another possible upgrade/fix would be to take the shift left and shift right immediate and combine them into a shift right, and let negative numbers shift it left. I spent about an hour working on the test bench and the ALU. transferred to another section fo the work, the intermediate registers between the stages. I was again unsure how to proceed, but this time I looked into what my teammates did for the single use register. This process was a lot of copy paste replace combos. This took about another hour. I double checked the program, and realized that several of the outputs were just wires, not registers. Make a point of double checking the code as we go forward.

10/24/2023

One of my teammates told me that the register blocks and ALU were having trouble compiling. I realised that the issue was the end of the input/output block cannot have a , character. In addition, I had ended many of my lines with a , instead of an ; which causes problems. Fixed in the register blocks, but not the ALU.

MILESTONE3:

10/13/23

Discussed what our language should denote as comments during the in-class work time. Decided that the # symbol would be good, since it is a simple, one character symbol that does not show up in any of our commands. In addition, I worked on the compiler/translator, laying the foundation for a possible runtime environment, or just a way to translate our register programming to base binary. I worked on this since I was unable to assist in the making of the data path, since editing it can only be done by one person at a time.

10/17/23

Worked for about 30 minutes further developing the translator. Only major decision is that future R types need to follow the pattern of one character commands, since that is easier to hardcode in than track variable sizes. Programming for the I types will be a challenge because of this. Will look into making the program more flexible on what it can accept in terms of actual commands.

10/18/23

Worked for approx an hour on finishing touches for milestone 3. I made two major decisions, the naming conventions for components, and the Naming convention for inputs/outputs. I chose camel case since most of us have coded in java, so this will make verilog easier. I chose I and O at the beginning so that we can easily tell whether something is an input or an output in the verilog code.

For Milestone 3 I am going to be working on the ALU implementation and register file, and continuing work on the translator.

MILESTONE2:

10/4/23
I helped plan the next milestone, and finish up one. We decided to change our command conventions, opting for the final register to be the destination in all cases. In addition, to make typing faster, we decided to change the names to be mathematical symbols or sets of relevant symbols, such as Y for the branch command since it delineates a fork.

10/10/2023
Spent about twenty minutes contributing to the RTL, and considering what else needed to be done for milestone two, and future plans. Discussed with the team why the destination register was named 3rd arg. While we left it for the speed of jump commands, it may become an issue if we develop the processor to be a pipeline. We left it as is for now. I wonder if pipelining could be even faster, should we decide to disregard the actual final cost of the machine. One possible way to do this is by swapping all non-essential registers (i.e. the input/ return argument registers) to be saved or all to temporary registers. This may make the procedure calls better overall. Will bring up at tomorrow's meeting.

10/11
Proposed again the mark command. Was again refused due to the simplicity of the current commands. Decided to swap our work from multi-cycle to a pipeline system. It will be harder, but fastest possible system. Decided to double check the RTL by
Went over the work we did for milestone 2, and confirmed that it met the requirements.

For the next milestone, I will be assisting Drew on the datapath construction, probably taking less than an hour, and possibly taking some time to contribute to the tests and basic quartus work, depending on how long I have.

MILESTONE 1:

9/27/2023
We only spent the remaining class time that we had at the time.
We decided to do a load store implementation becuase it is the fastest and most simplest for our purposes.
We decided to have the following registers:
A zero/ground register, a return address register, a stack pointer, two argument/return registers, and three temporary registers. We chose 8 registers so our main add command can be 12 bits

for registers and a four bit opcode. We chose a four bit opcode since we're restricted to 16 bit commands.

10/3/23
I helped complete the design document that is due for milestone one, spending an hour after class.
Gave us the ability to add more R types by extending the opcode of the R type instruction, while simultaneously removing dead bits in the instruction type.