William Dalby

Friday 10/4/19

- Met for 1 hour about what type of ISA we wanted to use. We decided on a stack mixed with load store. We worked on the instruction types that we wanted and the instructions that we needed. Talked over the procedure calling convention and addressing modes.

Sunday 10/6/19

- Met for 2.5 hours on the code for relPrime. Much discussion about the procedure calling conventions and how deep the stack in use is. The layout and usage of the instructions were designed by the team. There was talk about addressing modes and how to handle essentially everything in the code.

Monday 10/7/19

- Met for 3 hours redesigning our language. As of this moment, I'm still not very sure I understand what's going on with our architecture. I try to ask questions but they end up not being very helpful or are obvious to my peers. It is slightly discouraging. I will have to spend time understanding what our architecture is trying to do on my own time or I feel I will be a bit behind. The deliverables are done at this point though. We didn't designate exactly what to do for the next milestone.

Tuesday 10/8/19

- Met for an hour cleaning up the document and our journals. Prepared for the meeting on Wednesday.

Wednesday 10/9/19

- Met for 2 hours correcting the issues with our document. We added an executive summary and title page. We started to designate where our memory was going to be so that we wouldn't have any variable addresses. We created a list of things that we needed to fix before starting the second milestone.

Sunday 10/13/19

- We met for 3 hours fixing all of the issues from milestone 1. This included making the memory stack and designating where things are allocated. We changed how we saved things to memory by utilizing a stack pointer. This changed our code substantially (knocking off about a page and a half of code) and it all had to be rewritten. All of the addresses for the machine code had to be changed and a number of opcodes were changed. In addition, we added a funct to our C-type ISA so that we could push and pop from both registers and memory separately. We started to add RTL for each of our operations. This was scratchwork and will need to be changed into the summary table later. We began to discuss what hardware we'd need but we definitely to look into that more.

William Dalby

Monday 10/14/19

- Met for 2 hours. The main things that were worked on were solidifying our RTL for all of our operations. We put all of our scrathwork into a summary chart that, frankly, looks terrible. This is not because the information is bad but because the formatting just doesn't look good. My bad. We made diagrams and input output descriptions for all of the hardware we think that we'll need. These look pretty decent (my good) and I feel that we're starting to actually have a project coming together. We solidified a couple of our ideas and how the programmer will need to use them. At this point I feel quite a bit better about the project. While there are certain aspects where I feel a little behind my partners at this point I am understanding our operations better and better. I feel that the level of abstraction is decreasing which is helping me quite a bit. We are still not designating what needs to be done for the next milestone ahead of time. I think that the reason for this is that we do all of our work together as a team. This may not work out in the long run but at the moment we're rolling with it.

As an aside, how formal should these journals be? Also, is dialogue between myself and yourself encouraged here?

Monday 10/21/19

- Met for 2 hours. Mainly worked on the three labs that are due on friday. I realized that I have messed up the ALU lab quite a bit by trying to jump straight to the 4bit alu rather than doing it incrementally. Will need to fix that. I also implemented the shifter but then lost it after a GIT accident.

Tuesday 10/22/19

- Tori and I met for 3 hours and worked on fixing the issues from the last milestone and testing our components. I worked mainly on adding the machine code for our common operations but I also remade the shifter. I also made the ALU but I am worried that it's implementation may not be the way that is desired by the project. I simply used a 17 bit register to store the output and then its most significant bit was the overflow detector. Also, there is no implementation of an adder. I just used the built in verilog commands. I feel as though I'm going to have to change this at some point to more closely resemble the ALU from the lab. I have not yet finished the lab yet so I'm going to roll with the ALU as it stands.

Carry out ≠ overflow

Wednesday 10/23/19

- Met for 2 hours during lab. I updated more of our machine code for our common operations. I worked on drawing the datapath . This was more out of lack of anything else to do as I cannot claim to be the best artist so I personally apologize for the appearance of the spaghetti that is our data path. I am a bit confused as to how it's all going to work, however. How will operations that do multiple pushes and pops work? The main one that I'm concerned by is the flip as it pushes a and then pushes b. SO that

do as one op (maybe move 2 instead of 1, e.g.)

would mean that we need to push and then push again and I’m not sure how that’s going to work on our datapath. The same goes for dup as it pushes multiple values but that command in itself is intimidating so there’s a number of things to be concerned about there as well. I worry that the items I implemented might be lacking in terms of test cases. I’m not sure how extensively to test them but I’m sure that I need to implement more test cases.

- One other thing that came up during our meeting was that our memory addresses can only be 10 bits. This is quite confusing to me. I will probably ask about it during our meeting but I was wondering a number of things; why is this, how will our memory stack have to change to accommodate this, what does this mean for our memory pushes and pops, etc.

You must figure out how to map 16-bit addresses to 10-bit "word addresses". Hint: you will have some addresses that do not correspond to available memory.