CSSE232-04 Design Journal
Yellow-2324a-04
Blaise Swartwood

MILESTONE 1 WORK:

Wednesday, September 27th, 2023

Group work (1 hour)

- Decided on load-store architecture
- Met with team and discussed communication platform, group chat, official team outside meeting time on weekends – 11:00am Sunday
- Divided up work – I was assigned to work on converting Rel Prime to assembly code

Friday, September 29th, 2023

Individual work (1 hour)

- Worked on converting Euclid Algorithm/RelPrime into Assembly Code for easier translation into our machine code
- Started potential instruction list for our project and brainstormed possible registers

Saturday, September 30th, 2023

Individual work (30 min)

- Brainstormed some ideas of how our instructions could potentially be formatted and work
- Thought of 4 bit opcode, use func

Basic design: opcode 4 bits, register is 4 bits (16 regs)

How large should our immediate be? Minimum 4 – 8 bits? Func code for similar instruction types in addition to opcode for more instructions? → 16 is too few

Sunday, October 1st, 2023

Group work (4 hours)

- Basically completed Milestone 1 document entirely
- Discussed and created the Green sheet together
- Worked on design philosophy and what we could do

Individual work (during group meeting when broke it up) – 1.5 hours

- Converted RelPrime and GCD into our new machine code
- Wrote calling convention section and example
- Wrote out machine code for RelPrime

Assigned work for next milestone:

- I Type RTL
- Fix shifting
- Fix errors in GCD negative branching

MILESTONE 2 WORK:

Wednesday, October 4th, 2023

Group work (1.5 hour)

- Decided to combine all shifting into one instruction and use imm extra bits to determine shift left vs right and shift logical vs arithmetic
- Decided to have a single branching instruction beq to do all others
- Read and divided up work for Milestone 2 in more detail and planned our next meeting Sunday with the work done

Thursday, October 5th, 2023

Individual work (1.5 hour)

- Changed the relPrime instruction to have correct branching
- Made shifting more clear
- Added the I Type RTL instructions for each one and made sure they worked well

Sunday, October 8th, 2023

Group work (2 hours)

- Checked RTL for each instruction type of the group
- Checked the components table – it was missing outputs and bit numbers, needed to add
- Group discussion on RTL – changed the RTL jal instruction to make sure it stores the return address as necessary
- Wanted to make the shift instruction RTL more clear, but am not sure how to show that
- Decided to use pipelining for the datapath to make the project more complexity

Individual work

- Wrote the brief overview of process for talking checking for RTL errors
- Made the note of changes made to the Assembly and Machine Language specifications

MILESTONE 3 WORK:

Wednesday, October 11th, 2023

Group work (1.5 hour)

We all read the milestone again and split up work to do. I was assigned to write the test details, test cases, and implementation of the immediate genie, which I started to work on. We also went through and changed our multicycle RTL to be pipelined and cleaned up our document.

Thursday, October 12th, 2023

Individual (1 hour)

I already started creating the python file and text files to create the assembler to just imagine and visualize how to implement it if we wanted to go the assembler route we were told from the meeting. I set up the file reading and writing functions and the basic layout for the add instruction. I also wrote out the code for the immediate genie – it was rather simple.

Friday, October 13th, 2023

Group work (30 min)

We decided to choose the assembler + multicycle to get our extra points rather than pipeline. Realistically, the amount of extra work from pipelining and time is not worth it as we are busy with other projects in classes as well. Also, we do not know enough about pipelining to do a sufficient amount of work for this milestone. We also divided up dealing with the feedback we received at the meeting.

Individual (1.5 hour)

I filled out the rest of the assembler basics for each instruction we had and also dealt with immediates. Figuring out how to convert decimal to twos complement binary representation for the appropriate number of bits took some time. I decided that all instructions would take decimal inputs except for shift, where it is easier to understand what shift one is doing and how if they write it out in binary. This will need to be a full 8 bit binary value. I also wrote out the draft test bench file for the immediate genie and added 3 basic tests and made sure they passed and failed.

Saturday, October 14th, 2023

Individual (2 hour)

I thought about how to calculate the immediate of a branch instruction using a tag. The whole code would need to be given to the assembler at once, with no extra line spaces in between. I decided to use an initial loop to go through each line and add each tag into a dictionary. Then, I debated on how I could deal with the tags on the instruction statements. I could either save all the lines in the file and try to edit out the tags and replace it with immediate, or I could resave the data into another file. I ended up choosing to just make a new file that got rid of the placement tags because the rest of the code I wrote before used that, and then when encountering a tag in an instruction, I then used the current line and subtracted it from the one stored in the dictionary. Not an easy process to think through, but it works.

Sunday, October 15th, 2023

Individual (1 hour)

I finished writing some more of the immediate genie tests. I also looked at the work my group members did and made sure we were staying on task getting our designs done.

Tuesday, October 17th, 2023

Group (1 hour)

We did not assign enough work over the milestone and reg file and ALU still were not designed at all. Our Datapath design was not complete yet as well, and it was hard to write the implementation plan without it. I took on the ALU writing a plan and testing and started coding it in Verilog.

Individual (1 hour)

I coded up the ALU in Verilog and began writing my test cases.

Wednesday, October 18th, 2023

Individual (1 hour)

Finished up writing and testing the ALU to confirm that most operations worked correctly.

MILESTONE 4 WORK:

Wednesday, October 18th, 2023

Group work (1 hour)

We all read the milestone again and split up work to do. I feel like we are kind of behind in implementation, and we also decided that the cool logic for the shift would be handled in immediate genie sent to an ALU Control to send into the ALU. I had to add this bit sending function into the immediate genie and put it in a few tests. I also started to work on and created the ALU Control in Verilog and wrote tests for it.

Individual work (1 hour)

I finished writing the ALU Control implementation and test.

Thursday, October 19th, 2023

Individual (3 hour)

I started designing the implementation for ALU, ALUControl, and Immediate Genie. I realized that my ALUControl was not yet clocked, so I modified the ALUControl and its tests and added the clock. I also realized that for ALUSrcB mux, it was a 4:2 mux, which we did not have. I used the 2:1 mux and implemented the 4:2 mux. I also fixed issues in our 2:1 mux not supporting selection of 16 bits and made the code cleaner and easier to understand. I finished the implementation of this integration test.

Friday, October 20th, 2023

Individual (7 hour)

Pretty sure I'm about to lose my mind. But I got the assembler to take in pseudos in a specified format. I hate parsing file information and using re. I think it works but I am too tired to check. Also, having to add accounting for tags and take into account any shifting from replacing other instructions is very frustrating. There is definitely a lot more efficient and cleaner way to do this than writing to like 5 different files for each step, but what is done is done.

Besides the assembler, I worked on the integration plan a bit more. I did basic tests. I can forsee issues with the clock already and controlling the multicycle cycles. Like for beq, I can only test the comparison A and B parts for the ALU and not the PC + imm part yet since we don't have PC. Furthermore, I see that our lbi and lui instructions were not really implmeneted in any way. I am going to have the computation for these in the immediate genie and have an ALUOp that just selects input B (the immediate) in this case to get this to work. But that is tomorrow's me problem.

Saturday, October 21th, 2023

Individual (2 hour)

I changed lui and lbi to work correctly – I was not entirely sure how to implement them beforehand. I decided to have lui and lbi put 0s in the respective locations. Then I decided to make a "get B value" ALU op code operation for lui. However, lbi would 'or' its value in the register. So to load a full 16 bit value, I would first lui the top 8 bits and then lbi the bottom 8 bits so this can work. I made these changes and

tested them in the immediate genie. I also wrote more tests for the ALU_ALUControl_IG integration plan. I also remembered that we need to modify our datapath to work for jal and jalr. Will talk about that with the group tomorrow.

Sunday, October 22th, 2023

Individual (1 hour)

I finished reviewing the assembler and performed stress testing. It should be ready to go.

<<Missed several days due to Conference >>

MILESTONE 5 WORK:

Monday, October 30th, 2023

Individual (1 hour)

Reviewed the integration plan again and reworked some tests. The one I wrote should be good for full implementation addition now. I worked ahead so now I believe I am still on pace with this milestone and the work I am assigned.

Tuesday, October 31th, 2023

Individual (1 hour)

Started thinking about how we will combine all the components. Also checked in with my group has done to make sure I really know where we are.

Group (1 hour)

Started working together on the full final processor compilation together as a group, minus the control. We need to figure out how clock and cycles work.

Wednesday, November 1st, 2023

Individual (1 hour)

Started thinking about how we will combine all the components. Also checked in with my group has done to make sure I really know where we are.

Group (1 hour)

Started working together on the full final processor compilation together as a group, minus the control. We need to figure out how clock and cycles work.

Thursday, November 2st, 2023

Group (2 hour)

Started working and created the full datapath. We decided on the basic design for the input and output instructions. We talked about how we want our project to run overall and how to keep track of cycle information. The processor is not working at this time.

Individual (30 min)

Wrote the new input code necessary and decided on how to use the instruction to keep the looping. The logic of how the input instruction worked needed to be relogged so that a new instruction could be added together.

Friday, November 3st, 2023

Individual (1 hour)

Wrote the new input code necessary and decided on how to use the instruction.

Group (2 hour)

Started working together on the full final processor compilation together as a group, minus the control. We need to figure out how clock and cycles work.

MILESTONE 6 WORK:

Saturday, November 4st, 2023

Individual: 2 hours

Started working more on the complete processor but encountered many bugs. Got stuck for the night because I couldn't run two instructions back to back because the PC incrementing was mistimed.

Sunday, November 5st, 2023

Individual: 5 hours (9 am – 2pm)

Grind time on. I got add, sub, addi, and si to work. I created and named every component in our waveform nicely and saved the settings.

A few bugs that I managed to fix – Control needed to set the ALUControl alu opcode to add for every PC, which was not being done before. I added in this control signal and made the ALU Control update on the negative edge so that the ALU would add correctly during this cycle. As far as I have seen, this does not break any other ALU operations during the usual execution stage. Before this, it seems that whenever a non adding instruction was off, the PC address changed to some random value.

I also added in the logic and mux to have rd input for si also be an input for rs1 to do the actual shifting for the storeword. This was essentially an exam question of modifying the datapath, state diagram, and actually implementing this. We did not have this noted and written down beforehand. Now I am already thinking about branching, and how we will need another control signal to the register file to denote that we want to get the value of BR and put it to B automatically.

Monday*, November 6st, 2023*

Individual: 4 hours (forgot)

Grind time! I got our wave form revised and started revising more of our datapath. I'm honestly forgetting what I did because I am filling this out a few days later with my head quite tired from relPrime stuff, but I know I worked on several instructions and hashing out bugs in the code and test benches.

Tuesday*, November 7st, 2023*

Individual: 3 hours (7 am – 10 am)

I got jal, jalr, and beq to work. It was annoying to have to change and revise the datapath because we were so unprepared previously, but it worked now. A lot of offset errors by +2, -2, and weird behavior with setting the ALU behavior. I thought there was an error and wanted to meet with Williamson to discuss it.

Group: 1 hour

Our team went to meet with Williamson about a bug with jal not going to the write place. However, apparently I had already dealt with the issue and was just confusing myself.

Wednesday*, November 8st, 2023*

Individual: 2 hours (12 noon – 2 pm)

My teammates focused on fixing lw which was broken yesterday. Now that it worked, I fixed again some bugs in beq and jal. I also edited the assembler to fit our needs again. Then, I worked on fixing some issues with rel prime code – our sw values were negative when they shouldn't be and our blt template was flipped (I knew something with the branching templates were off from the beginning with how they were written and luckily that lead to this fix not taking me years to find). Relprime code worked almost first time besides these bugs. Relprime now works and I tested two different values in a row and it worked. I'm so relieved because that was so stressful.