CSSE232-04 Design Journal

Yellow-2324a-04

Reilly Mooney

Wednesday, September 27, 2023

Individual Work: nothing for this meeting

Teamwork: decided on architecture type – Load/Store

Sunday, October 2, 2023

Individual Work: Participated in discussion about what instructions, data types, and registers we were going to use in our Load/Store architecture. Wrote out the machine code for relPrime according to our instructions and data types.

Teamwork: Completed Milestone 1. This includes a decision on the registers we are going to use, how our instructions will function, and creating new data types for the 16-bit instructions. We also wrote out the GCD and relPrime assembly code according to our instructions.

Wednesday, October 4, 2023

Individual Work: nothing for this meeting.

Teamwork: First milestone meeting with Dr. Williamson. We were advised to consider using one instruction, beq, for branching since we are using the br register for comparisons.

We want to be able to branch with an implementation of ONLY the beq instruction. The following table describes our thoughts on how we could do this...

Instruction
sub t2, t0, t1
sli t2, 15
add br, x0, t2
beq t2, BRANCH
add br, x0, t0
beq t1, BRANCH
add t0, t0, t1

First, subtract the values you want to compare. Shift the values 15 bits to the left. Because of 2’s complement, the first bit should determine if the value is negative (1) or positive (0). Put this value into the branch register (br). Use beq to branch accordingly.

Sunday, October 8, 2023

Individual Work: Wrote examples for branching and recursion, as well as the RTL for R-type instructions. The R-Type RTL is very similar to that of RISC-V, with the only real change being that PC will increment in 2 bits, rather than 4 since we are working with 16-bit instructions and not 32. The recursion example includes a function that will call itself until the branch register is updated and beq triggers a different function call.

Teamwork: Worked on completing Milestone 2. We reviewed the RTL diagrams we each worked on individually, as well as the list of components. Started to discuss the requirements for Milestone 3, which focuses on the datapath. Since we are still thinking of pipelining, we decided to wait on making any further decisions with this milestone until we have a lecture on pipelining in class.

Thursday, October 12, 2023

Individual Work: nothing for this meeting

Teamwork: We had our second milestone meeting with Dr. Williamson. In the meeting Dr. Williamson recommended we shift our project focus from pipelining to working with the assembler to implement more functionality in a multicycle datapath.

Pros to Pipelining	**Cons to Pipelining**
More applicable to modern architectures & industry	Still learning about this in lecture
Project bonus points ($$$)	Extremely difficult to implement
	Will make our lives more difficult as the project continues

We discussed this as a group and decided to ditch the pipeline idea and continue the project with focusing on gaining more bonus points with our implementation of the assembler instead. Following the meeting with Dr. Williamson, we divided the work we need to accomplish for the next milestone.

Tuesday, October 17, 2023

Individual Work: Before this meeting, completed the test plan for the memory component and added descriptions of CLK and RST signals for each component that needed them. Made the decision to hold off on writing the Verilog code for the memory component, as we will be working on that in lab 7. During the meeting, worked on writing descriptions for each control signal.

Teamwork: Reviewed the individual work done by each member of the group. Divided the remaining work for the milestone between team members.

<u>Wednesday, October 18, 2023</u>

Individual Work: none

Teamwork: Decided on using an ALUControl component. The ALUControl will take a 2-bit immediate from the ImmGen and a 4-bit opcode. The purpose of this control, and the reason we would separate it from the control unit we already have is so the ALUControl can focus on shifting. As of right now, there wasn't a good way we could tell the ALU we needed to do an arithmetic or logical shift. To fix this, we decided to implement the ALUControl, which will take care of this decision to do arithmetic shift, logical shift, or no shift at all.

<u>Sunday, October 22, 2023</u>

Individual Work: Worked with Spencer on the implementation of lab 7. This will give us an idea of how memory should be set up. Once lab 7 is completed, we can use what we did in that to implement memory in our project. Today's work was just a rough start to the lab, just completed the first Memory.v file and the test bench.

Teamwork: none (except for the work done with Spencer)

<u>Monday, October 23, 2023</u>

Individual Work: Worked with Spencer to complete the implementation of lab 7.

Things to be careful with in further implementation:

- The size of inputs and outputs
- Decimal vs hexadecimal vs binary

Teamwork: none (except for the work done with Spencer)

<u>Tuesday, October 24, 2023</u>

Individual Work: Met with Dr. Williamson and Spencer to discuss / demo lab 7. Note, lab 7 is currently not completed... What needs to be fixed is defined below:

- When we run the simulation, the control signals are not outputting anything.

- We have a warning that says the opcode does not match the port size.... we've checked all of our opcode definitions and can't seem to find the issue.

Teamwork: none (except for the work done with Spencer)

Thursday, October 26, 2023

Individual Work: Continued work on lab 7 with Spencer. Had a bug that took a while to figure out. Just a simple naming convention issue (i.e., something was named "clk" that should have been called "CLK".

Teamwork: none (except for the work done with Spencer)

Friday, October 27, 2023

Individual Work: Completed the lab 7 demo with Spencer and Dr. Williamson. Notes from the demo meeting...

- Be careful with clocked components. There will be a little lag between each clocked stage. (When reading the waveform, read clocked components in a diagonal manner due to this lag.)
- The lab works with a 10-bit address. We need to work with a 16-bit address in our processor. To handle this, we will need to make a new Verilog "wrapper" file to shift the 16-bit address right, so we work with a 10-bit address within memory and output a 16-bit address when exiting memory.
- There is a certain threshold (0x03FF) that we cannot exceed in our memory map. Any address that is greater than 0x03FF needs to be set to be 0x03FF.

Teamwork: none (except for the work done with Spencer)

Monday, October 30, 2023

Individual Work: Continued work on memory with Spencer. We started to convert lab 7 to fit the memory component in our project. Created a new component called "Memory_Wrapper.v". The memory wrapper will take a 16-bit address, shift it to become a 10-bit address, and output the new 16-bit address.

Teamwork: Had a brief meeting / general discussion about what to do with I/O. We decided that the easiest way to handle this is to just use the opcode to load the data in.

Tuesday, October 31, 2023

Individual Work: Continued work on memory. Trying to find an issue with the Memory Wrapper. When trying to set an address value that is greater than 0x03FF, with the goal to change that address to 0x03FF, our output does not give us what we need. The address from memory is not being received.

Teamwork: Began to integrate all our components for the final processor. Control was still in the process of being implemented on our own so we will add that into the final processor later.

Wednesday, November 1, 2023

Individual Work: Finished work on memory with Spencer. The issue we were having in the memory wrapper was a Verilog syntax error. You cannot assign a wire in an if statement in Verilog so we were having trouble assigning an address that was greater than 0x03FF to be 0x03FF. You can use a ternary comparator instead so we replaced out if statement with the line:

```
wire [9:0] newAddr = (memAddr > 'h3FF) ? 'h3FF : shiftedWire[9:0];
```

Teamwork: Continued work on integrating all our components. Input is now one of the arguments to the DataWriteMux. We have a new opcode called loadOutput now. This will give an output from the register file. Need to make a decision on how to handle the issue of clocking. All components are currently clocked on a positive clock edge. We think that some components may need to run on a positive edge while others are on the negative clock edge. We need to test our full integration before we can make a final decision on the clock.