CS 3432Computer Architecture I Lab Assignment 4

Teaching Assistant: Steven Ibarra
Instructional Assistant: Alan Perez & Esteban Munios
Assigned: November 02 , 2021 Milestone 1 Due: November 7, 2021
Milestone 2 Due: November 14, 2021

Introduction:

For this lab, you will finish constructing a RISC-V RV32 processor in Logisim. For Lab 3, you built and connected the parts of the processor needed to execute arithmetic-logical instructions. For Lab 4, you will add the components needed for memory access and branch instructions. The tasks you are to carry out are given below. You should thoroughly test your circuits at each step to make sure each one works correctly before going on to the next step.

Milestone 1 due Sunday, November 7

1. Construct an immediate generator (ImmGen) ciruit that will take a 32-bit RISC-V I, S, or SB format instruction as input and output the correct 32-bit sign-extended immediate value. The procedure for constructing the ImmGen circuit was explained in lecture on Oct. 28.
2. Construct the main control circuit (Control). This circuit should take the seven opcode bits from a RISC-V instruction as inputs and output the five one-bit control signals MemtoReg, RegWrite, MemRead, MemWrite, and Branch.
   a. Complete the truth table shown in Table 1 below by filling in the opcode bits and the control outputs for the immediate arithmetic-logical instructions.
   b. Use the completed truth table to generate the Control circuit. You may either construct the circuit by hand (e.g., by using a sum of products approach) or by using a tool such as Logisim's Analyze Circuit tool.
3. Construct a 16 MB instruction memory with 32-bit read-write data width and byte addressability. You may use the RAM component in the Logisim Memory library. We will only be addressing 32-bit words in the instruction memory, so for that memory you may simply throw away the two lowest order bits of the address input. Since the instruction memory will only be read and not written during program execution, you need not connect your processor clock to the memory's clock input. Since we want to execution each instruction in a single cycle, you should set the Asynchronous read property to Yes. However, we do not want the output of the instruction memory to change until the current instruction has finished executing. To prevent having it change too early, you can use a clock input, negate the clock input value, and connect the negated value to the output enable of the instruction memory. Doing this will cause the instruction memory to be read only during the second half of the clock cycle.
4. Construct a 64 MB data memory with 32-bit read-write data width and byte addressability. You may use the RAM component in the Logisim Memory library. Unless you are doing extra credit task 9 below, you can also simply throw away the two lowest order bits of the address input for data memory, since for the required part of the lab, you will only be

accessing memory with lw and sw instructions which load or store an entire 32-bit word and are assumed to be 32-bit word aligned. Configure the memory component to trigger on the rising edge of the clock and set asynchronous read to yes.

Table 1. Main Control Unit Truth Table

	Signal Name	R-format	I-format (not lw)	lw	sw	beq
Inputs	I6	0		0	0	1
	I5	1		0	1	1
	I4	1		0	0	0
	I3	0		0	0	0
	I2	0		0	0	0
	I1	1		1	1	1
	I0	1		1	1	1
Outputs	MemtoReg	0		1	X	X
	RegWrite	1		1	0	0
	MemRead	0		1	0	0
	MemWrite	0		0	1	0
	Branch	0		0	0	1

You need only turn in your circuits for Milestone 1. You do not need to schedule a demo until you have completed the entire lab.

Milestone 2 due Sunday, November 14

4. Construct the instruction fetch datapath, memory access datapath, and branch datapath, and add any additional circuits and logic to connect these datapaths with your already constructed components. You may use the comparator component available in the Logisim Arithmetic library for your branch datapath.
5. Connect all your components from Labs 3 and 4 so that you have a working simulation of a RISC-V processor that implements our chosen subset of RISC-V RV32I instructions (and, or, xor, add, sub, andi, ori, xori, addi, lw, sw, beq). When finished, your lab4 circuit should look similar to the circuit shown in Figure 1.
6. You should now be able to load and execute a RISC-V program that uses any of the instructions in our chosen subset. To load a RISC-V machine language program into your instruction memory, select your instruction memory while in Simulate mode and select Load Image from the popup menu. Develop a set of test programs and use them to demonstrate that your RISC-V simulator works correctly.
7. **Extra Credit** (optional, 5 extra lab points): In addition to the beq instruction, implement the bne, bge, and blt instructions.

8. **Extra Credit** (optional, 5 extra lab points): In addition to the conditional branch instructions, implement the jal and jalr instructions.
9. **Extra Credit** (optional, 10 extra lab points)**:** In addition to lw and sw, implement the lh, sh, lb, and sb instructions.

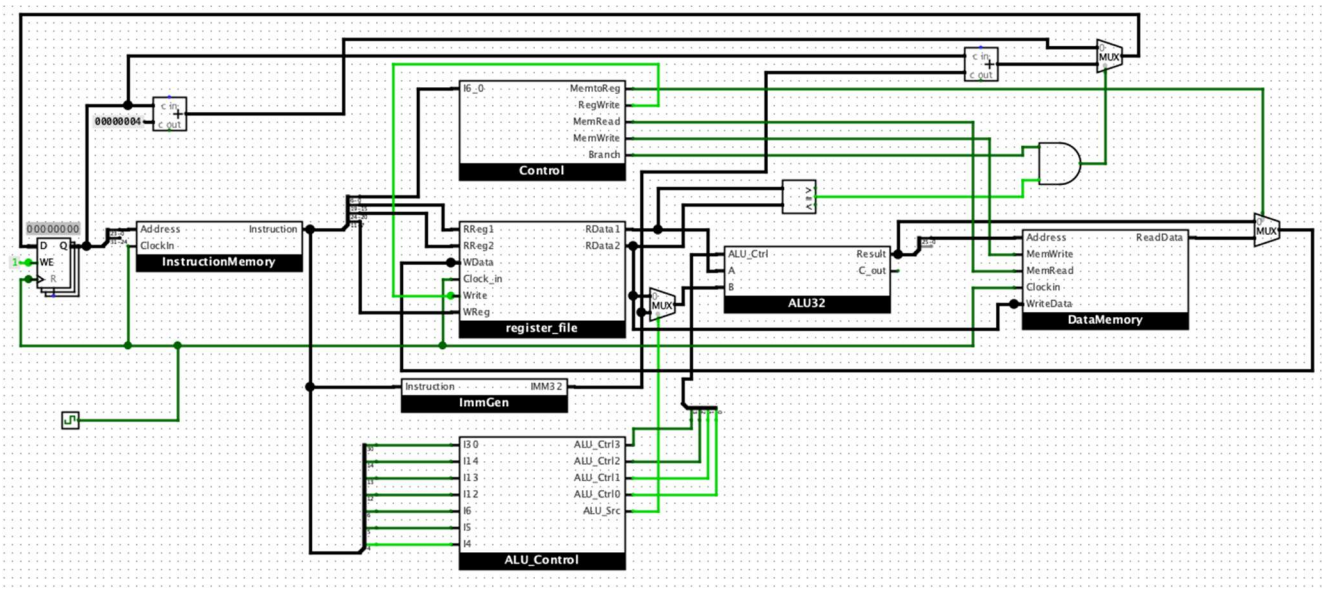

Figure 1. Completed processor for the required subset of RISC-V RV32I instructions

Grading (50 total points)|20 points Extra Credit

Milestone 1 completed on time: **10 points**
Properly functioning immediate generation and main control circuits: **10 points**
Properly configured and functioning instruction and data memories: **10 points**
Correct instruction fetch, memory access, and branch data paths: **10 points**
Demo with adequate set of test cases: **10 points**

Hand-In Procedures

The first Milestone will be turned into teams on November 7, 2021 @ 11:59. For milestone 2 everything needs to pushed to your GitHub by November 14,2021 @ 11:59, where will demo from using the checkout instruction.

Demos

You will not need to demo for milestone 1 but will demo once you turn in milestone 2. Demos are to be performed 1 week after the due date for milestone 2.