Ethan White

Lab 9 — CPU Control Unit (Connected with Datapath)

Objectives

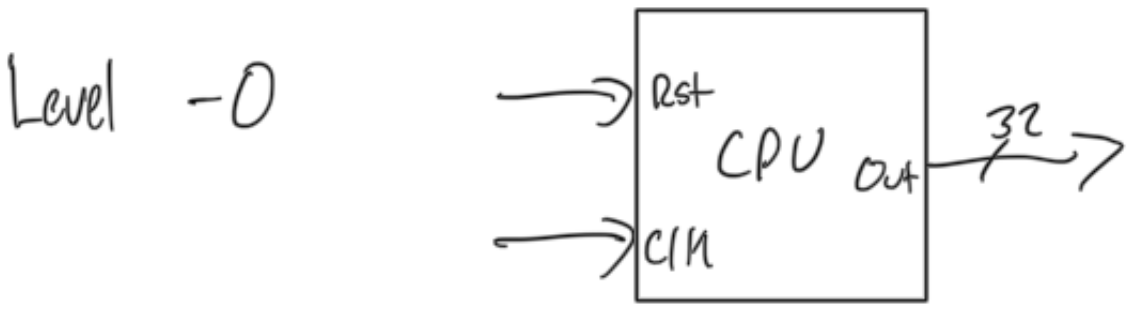
For this lab, I am creating a CPU control unit. To test this, I will be connecting it to the datapath I made in Lab 8 to create a full RISC-V ISA processor which is capable of executing R, I, S, and B type instructions.

Introduction

A CPU is made of two main parts: a datapath which executes instructions and a control unit which properly configures the datapath based on the current instruction. The control unit looks at certain bits of the instruction which determine the type of instruction. It then outputs the correct inputs for the datapath to, for example, choose a source for writing back to the register file.

Methodology

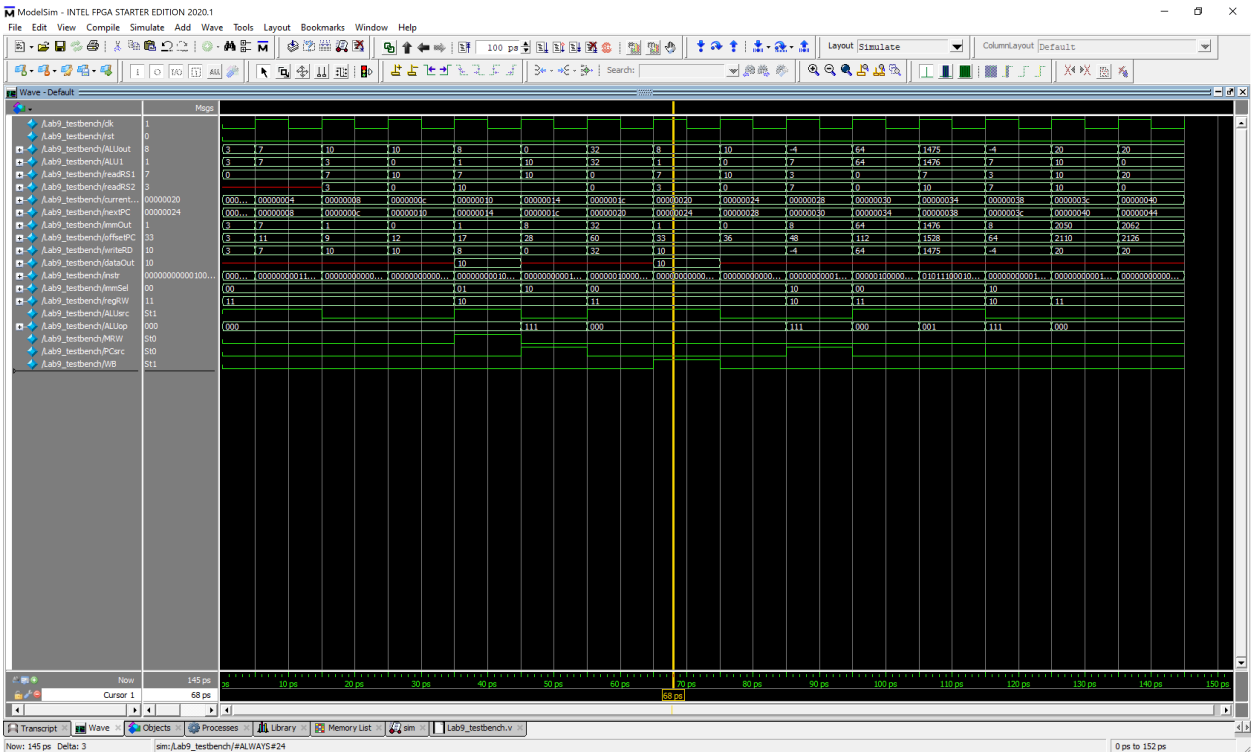
The following is a level -0 diagram of my CPU. Its inputs are clock and reset. Its output is the final output from the datapath.



The control unit works by using an if statement in an always block. This top level if statement evaluates the opcode from the decoded instruction and determines which type of instruction format it is (R, I, S, B). Within these if statements, each datapath input is set. For certain datapath inputs, such as ALUop, there is a second nested if statement which looks at func7 and func3 to determine which operation needs to be performed. Additionally, for I type instructions, the WB select input is determined as well.

Result and analysis

When tested in ModelSim, the CPU is found to be fully functional. As seen below, the output of the CPU is equal to the expected instruction output on each program count. Branch instructions are also tested to both properly branch when applicable, and not branch when applicable. Memory is tested for loading and storing.



Program Counter	Assembly Code	Instruction Type	Machine Code (b)	Equivalent Statement	Output	Executed
0x00	addi x1, x0, 3	I	000000000011_00000_000_00001_0010011	x1 = 0 + 3	3	y
0x04	addi x2, x0, 7	I	000000000111_00000_000_00010_0010011	x2 = 0 + 7	7	y
0x08	add x3, x2, x1	R	0000000_00001_00010_000_00011_0110011	x3 = 7 + 3	10	y
0x0C	add x4, x3, x0	R	0000000_00000_00011_000_00100_0110011	x4 = 10 + 0	10	y
0x10	sw x4, 1(x2)	S	0000000_00100_00010_010_00001_0100011	RAM[1+7] = 10	8	y
0x14	beq x4, x3, 8	B	0000000_00011_00100_000_01000_1100011	if(10 == 10) PC += 8	0	y
0x18	addi x7, x0, 16	A	000000010000_00000_000_00111_0010011	x7 = 0 + 16	16	n
0x1C	addi x8, x0, 32	A	000000100000_00000_000_01000_0010011	x8 = 0 + 32	32	y
0x20	lw x9, 1(x2)	I (lw)	000000000001_00010_010_01001_0000011	x9 = RAM[1+7]	10	y
0x24	add x10, x9, x0	R	0000000_00000_01001_000_01010_0110011	x10 = 10 + 0	10	y
0x28	blt x1, x2, 8	B	0000000_00010_00001_100_01000_1100011	if(3 < 7) PC += 8	-4	y
0x2C	addi x7, x0, 16	I	000000010000_00000_000_00111_0010011	x7 = 0 + 16	16	n
0x30	addi x11, x0, 64	I	000001000000_00000_000_01011_0010011	x11 = 0 + 64	64	y
0x34	xori x8, x2, 1476	R (logical)	010111000100_00010_100_01000_0010011	x8 = 7 ^(xor) 1476	1475	y
0x38	bge x1, x2, 8	B	0000000_00010_00001_101_01000_1100011	if(3 >= 7) PC += 8	-4	y
0x3C	add x3, x3, x3	R	0000000_00011_00011_000_00011_0110011	x3 = 10 + 10	20	y
0x40	add x16, x3, x0	R	0000000_00000_00011_000_01111_0110011	x16 = 20 + 0	20	Y

All code, diagrams, and simulation results can be found as files on GitHub: https://github.com/Eth7an/Lab-9.git

A video explanation of the top level design and ModelSim simulation can be found here:
https://youtu.be/-sRj1edUkgU

Discussion and Conclusion

To conclude, I have built a fully working and tested central processing unit. This RISC-V ISA CPU has been tested with each type of instruction format (R, I, S, B), as well as with RAM writing and reading. Both regular and logical R type instructions have also been tested.