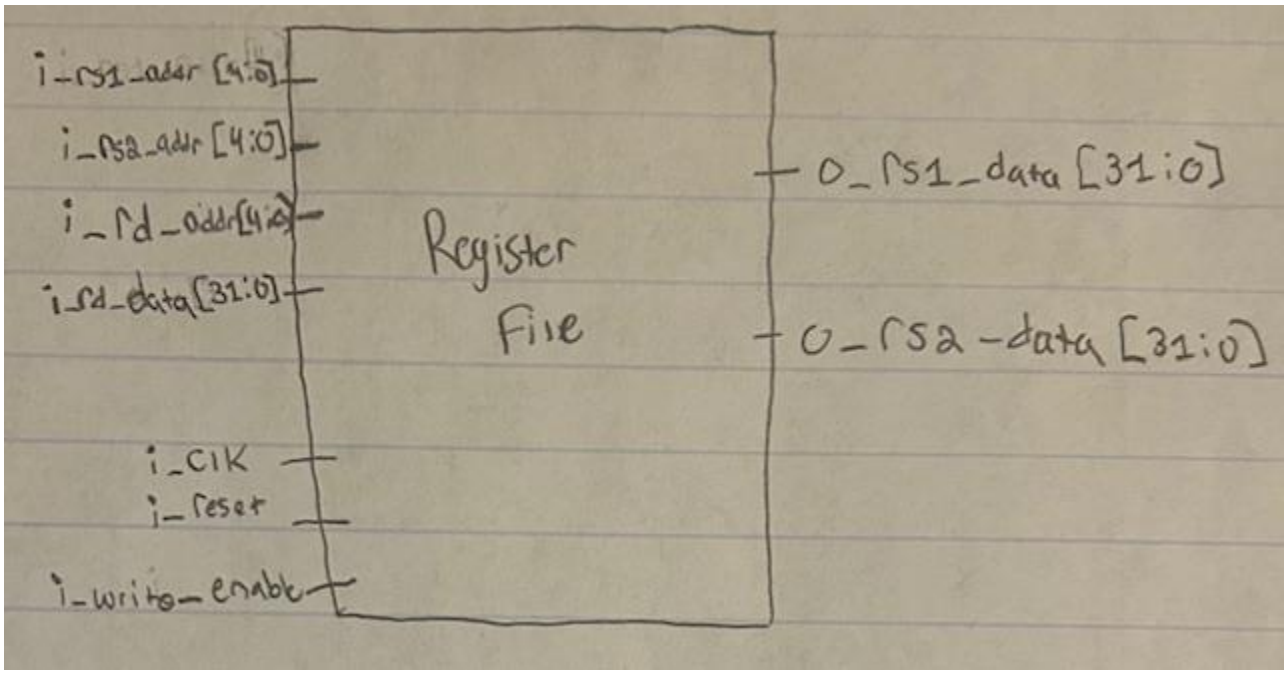
CprE 3810, Computer Organization and Assembly-Level Programming

Lab 2 Report

Student Name **Connor Moroney**

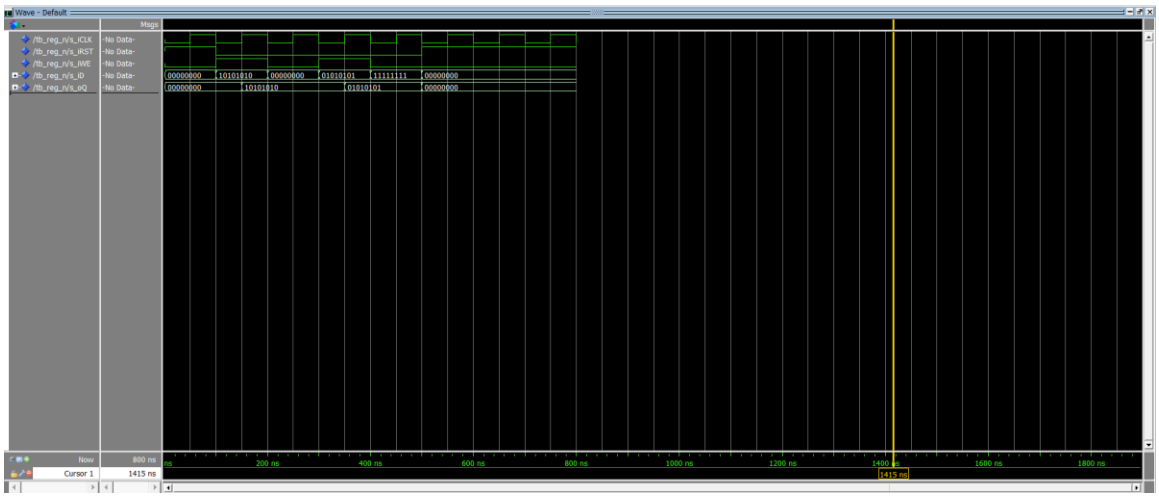
Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the lab document for the context of the following questions.

[Part 3 (a)] Draw the interface description (i.e., the "symbol" or high-level blackbox) for the RISC-V register file. Which ports do you think are necessary, and how wide (in bits) do they need to be?



[Part 3 (b)] Create an N-bit register using this flip-flop as your basis.
Done (See Waveform Below)

[Part 3 (c)] Waveform.

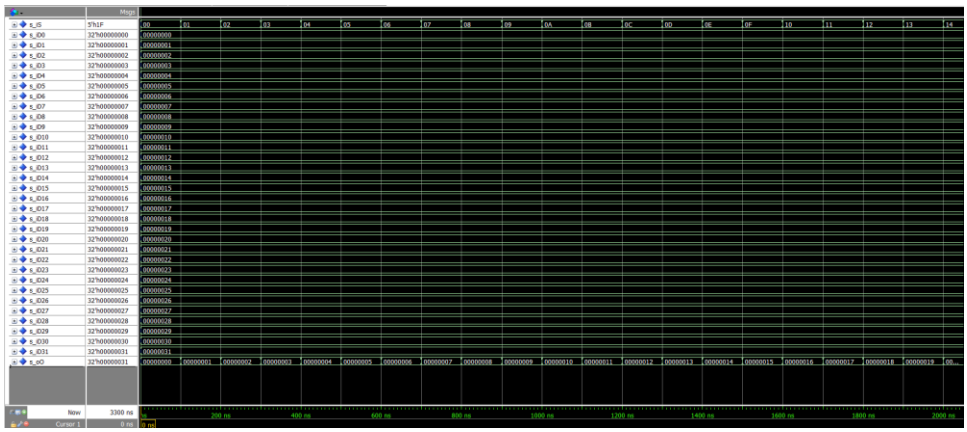


[Part 3 (d)] What type of decoder would be required by the RISC-V register file and why?
We need a 5:32 decoder because the register file contains 32 registers. The 5 selection bits are needed because Log2(32) = 5. The selection bits allow us to select any one of the registers for writing.

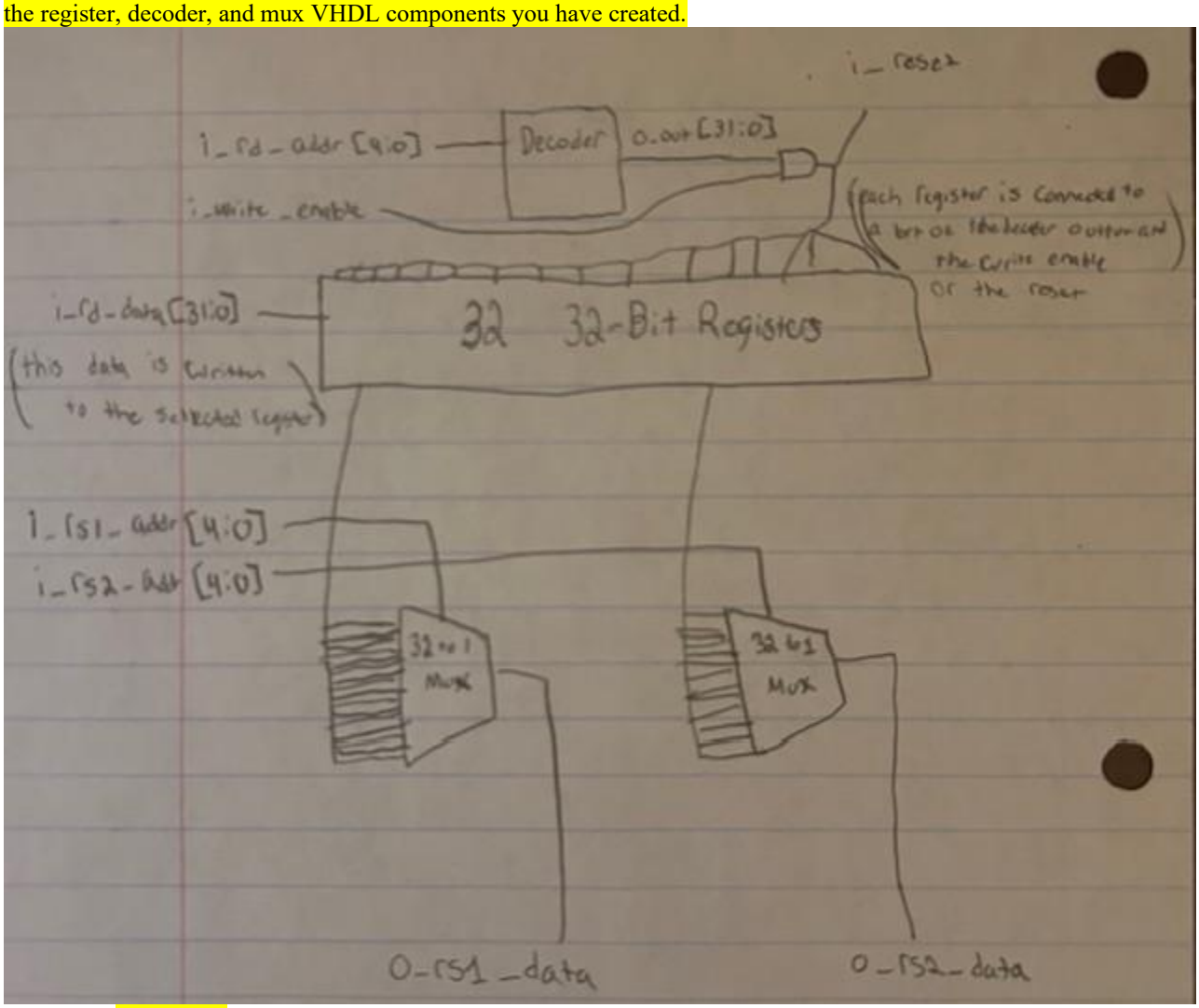
[Part 3 (e)] Waveform.

[Part 3 (f)] In your write-up, describe and defend the design you intend on implementing for the next part.
I am planning on using dataflow VHDL because it is very straight forward to implement. Using structural VHDL would require a lot of gates and could make it easy to mess up. With dataflow, I can directly set the output based on the input, making it easier to spot bugs.

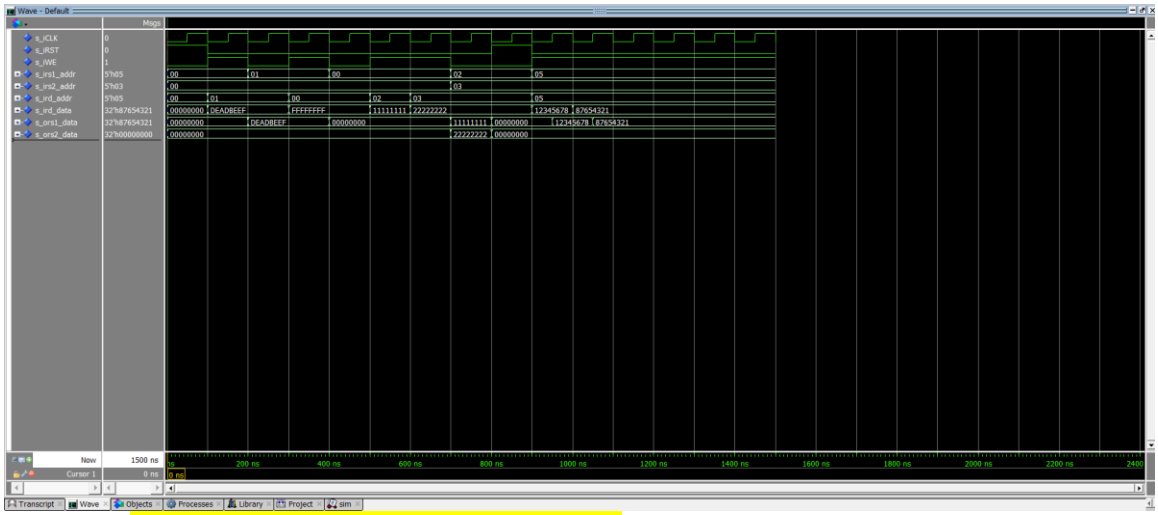
[Part 3 (g)] Waveform.



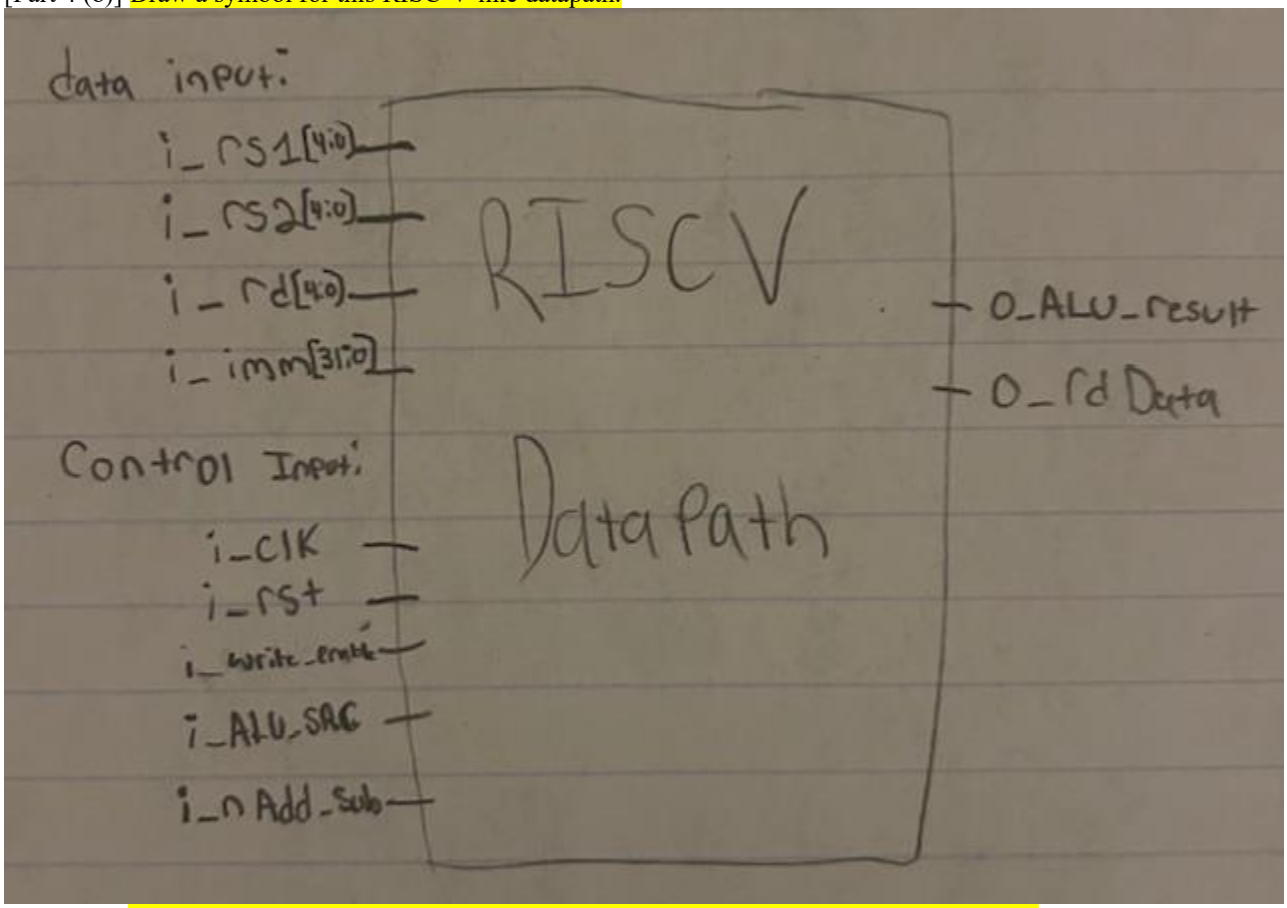
[Part 3 (h)] Draw a (simplified) schematic (i.e., components within the high-level blackbox) for the RISC-V register file, using the same top-level interface ports as in your solution describe above and using only the register, decoder, and mux VHDL components you have created.



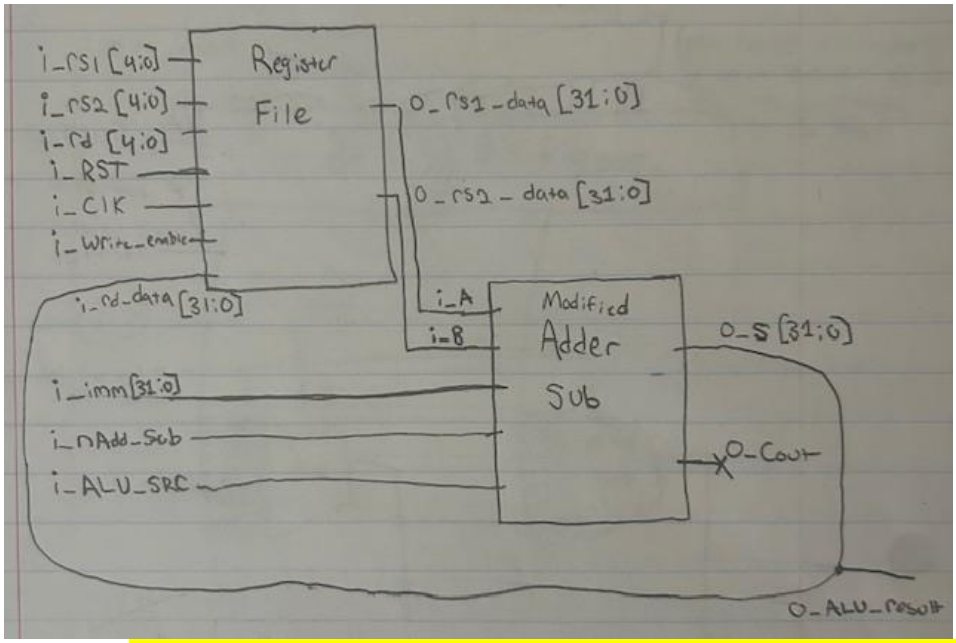
[Part 3 (i)] Waveform.



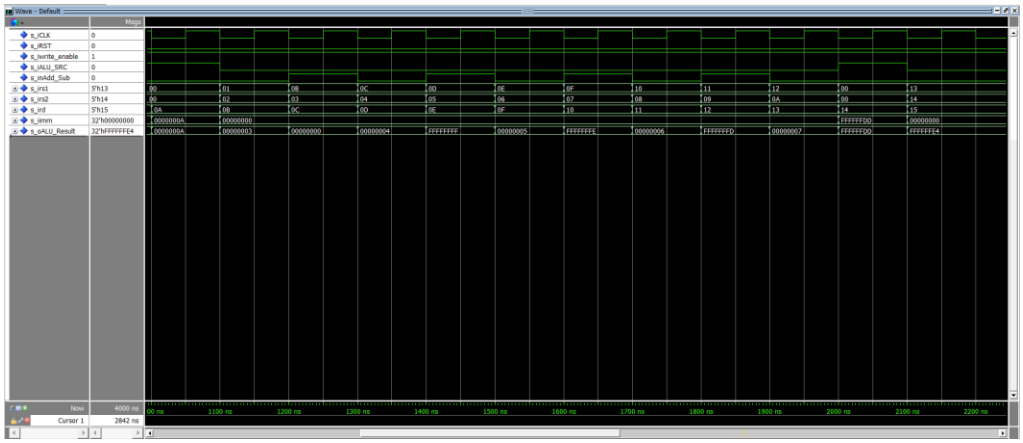
[Part 4 (b)] Draw a symbol for this RISC-V-like datapath.



[Part 4 (c)] Draw a schematic of the simplified RISC-V processor datapath consisting only of the component described in part (a) and the register file from problem (1).



[Part 4 (d)] Include in your report waveform screenshots that demonstrate your properly functioning design. Annotate what the final register file state should be.



Register	Value	Calculation
x1	1	constant
x2	2	constant
x3	3	constant
x4	4	constant
x5	5	constant
x6	6	constant
x7	7	constant
x8	8	constant
x9	9	constant
x10	10	constant
x11	3	1 + 2
x12	0	3 - 3
x13	4	0 + 4
x14	-1	4 - 5
x15	5	-1 + 6
x16	-2	5 - 7
x17	6	-2 + 8
x18	-3	6 - 9
x19	7	-3 + 10
x20	-35	constant
x21	-28	7 + (-35)

[Part 5 (a)] Read through the mem.vhd file, and based on your understanding of the VHDL implementation, provide a 2-3 sentence description of each of the individual ports (both generic and regular).

DATA_WIDTH **- Declares the bit width of each word in memory. The default value is 32, meaning that each word takes up 32 bits of space in memory by default**

ADDR_WIDTH **– Declares the bit width of each memory address. The default value is 10, meaning that each address is 10 bits long.**

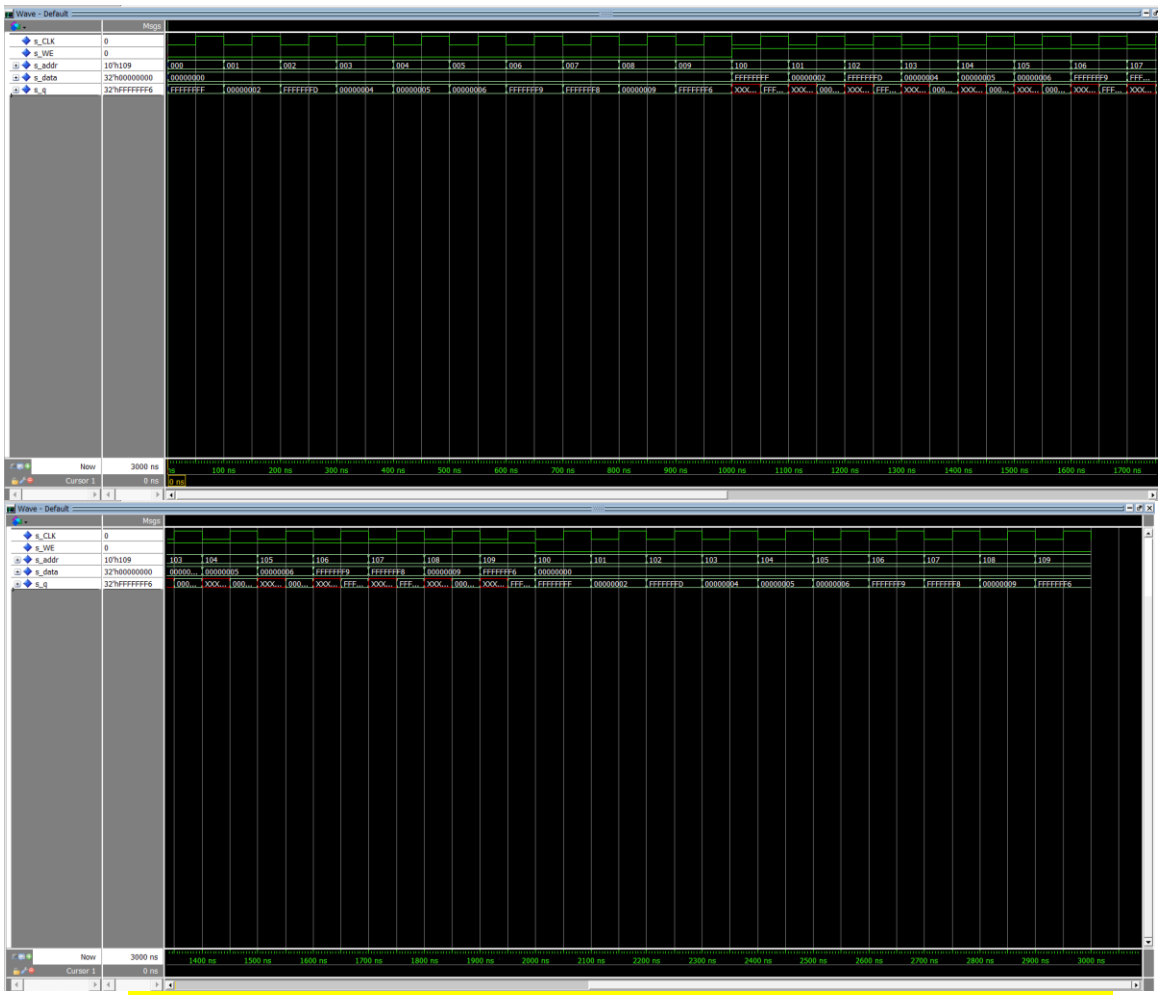
clk **– This is a standard clock input. It controls when writing to memory occurs. In this case, writing occurs on the rising-edge of the clock.**

addr **– This is the address of the data a user wants to access or modify from memory. The address is used in both read and write operations.**

we **– This is a write enable bit. When 1, the selected address can be written to on the positive-edge of the clock. Without setting this bit (we = 0) no write operations can take place. The default value is 1.**

q **– This outputs the data that is currently stored in the memory address addr. It is asynch ronous meaning it will give the data stored in real time.**

[Part 5 (c)] Waveforms.



[Part 6 (a)] What are the RISC-V instructions that require some value to be sign extended? What are the RISC-V instructions that require some value to be zero extended?

Some RISC-V instructions that require sign extension are ADDI, SLTI, ANDI, LW, LB, SW, SB. When a value is signed, if it needs to be extended, we use sign extension.

Some RISC-V instructions that require zero extension are LBU, SLLI, and SRLI. When a value is unsigned, if it needs to be extended, we use zero extension.

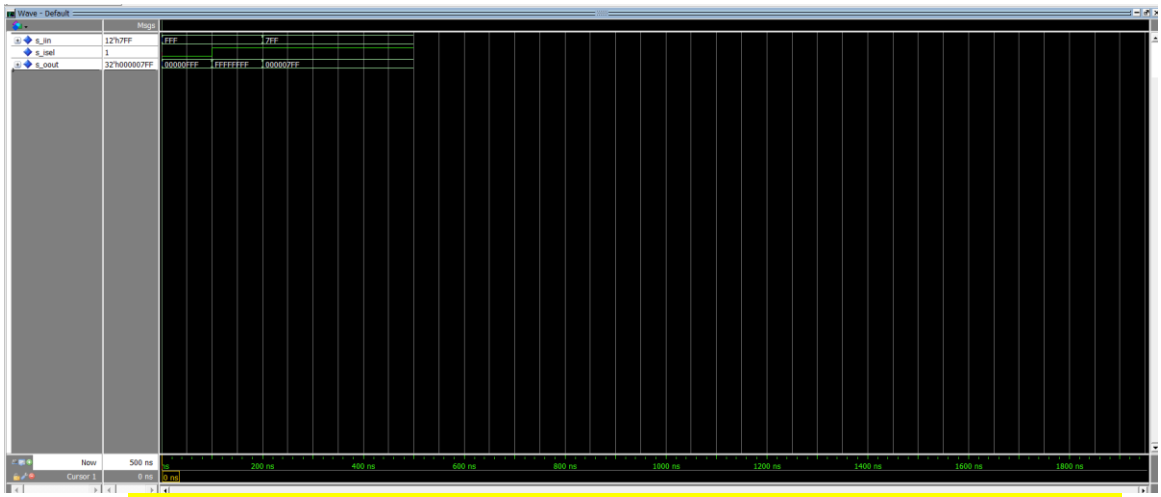
[Part 6 (b)] what are the different 16-bit to 32-bit "extender" components that would be required by a RISC-V processor implementation?

We need a Sign Extender and a Zero Extender component in order to implement all instruction types.

A zero extender is needed to implement unsigned instruction such as: LBU, SLLI, and SRLI.

A sign extender is needed to implement signed instructions such as: ADDI, SLTI, ANDI, LW, LB, SW, SB.

[Part 6 (d)] Waveform.



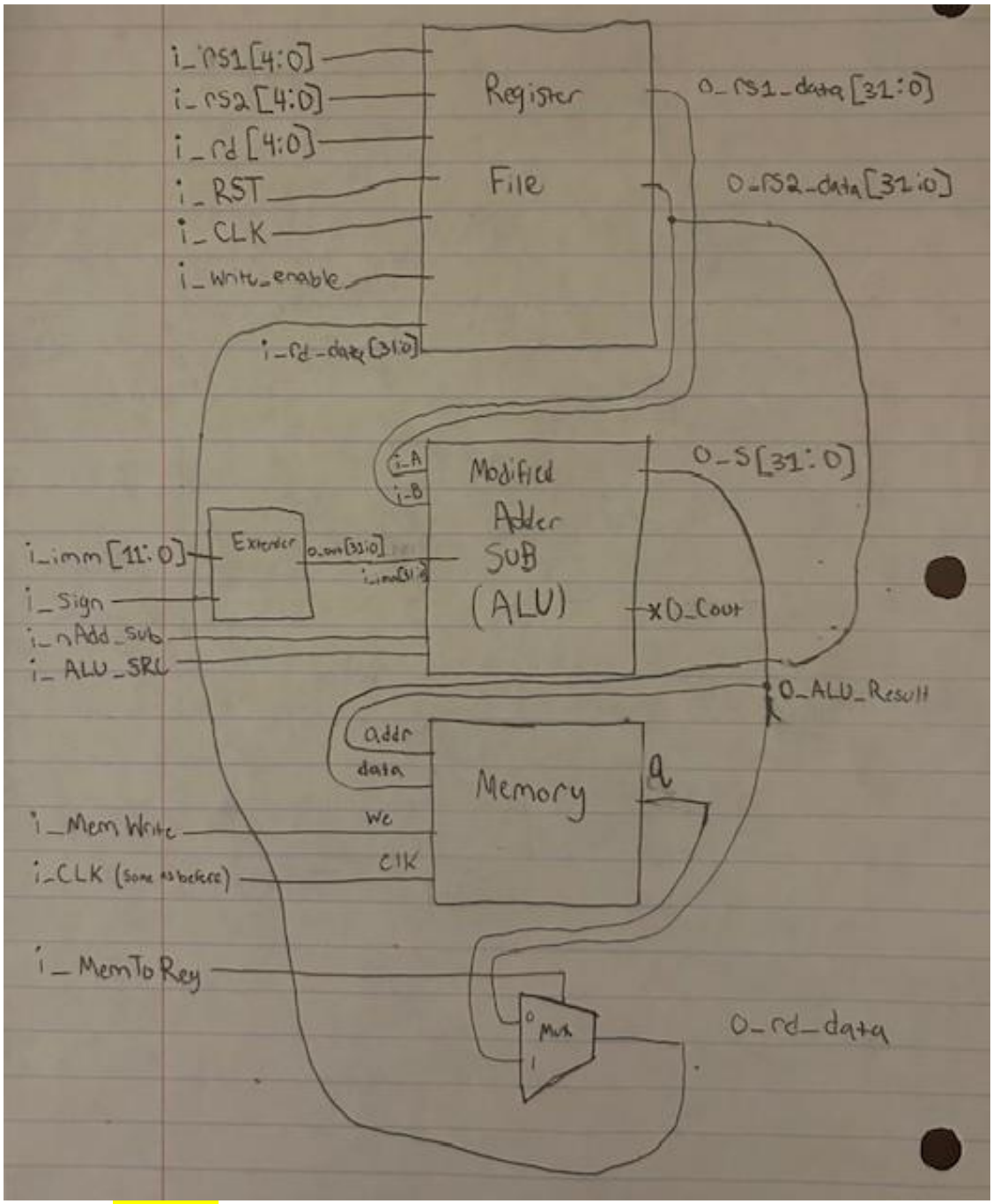
[Part 7 (a)] what control signals will need to be added to the simple processor from part 2? How do these control signals correspond to the ports on the mem.vhd component analyzed in part 3?

To support load and store instructions, we need to add an i_MemWrite signal to enable writing to memory (For SW) and an i_MemToReg signal to indicate if we want to save the ALU result or the memory result (FOR LW) to a register.

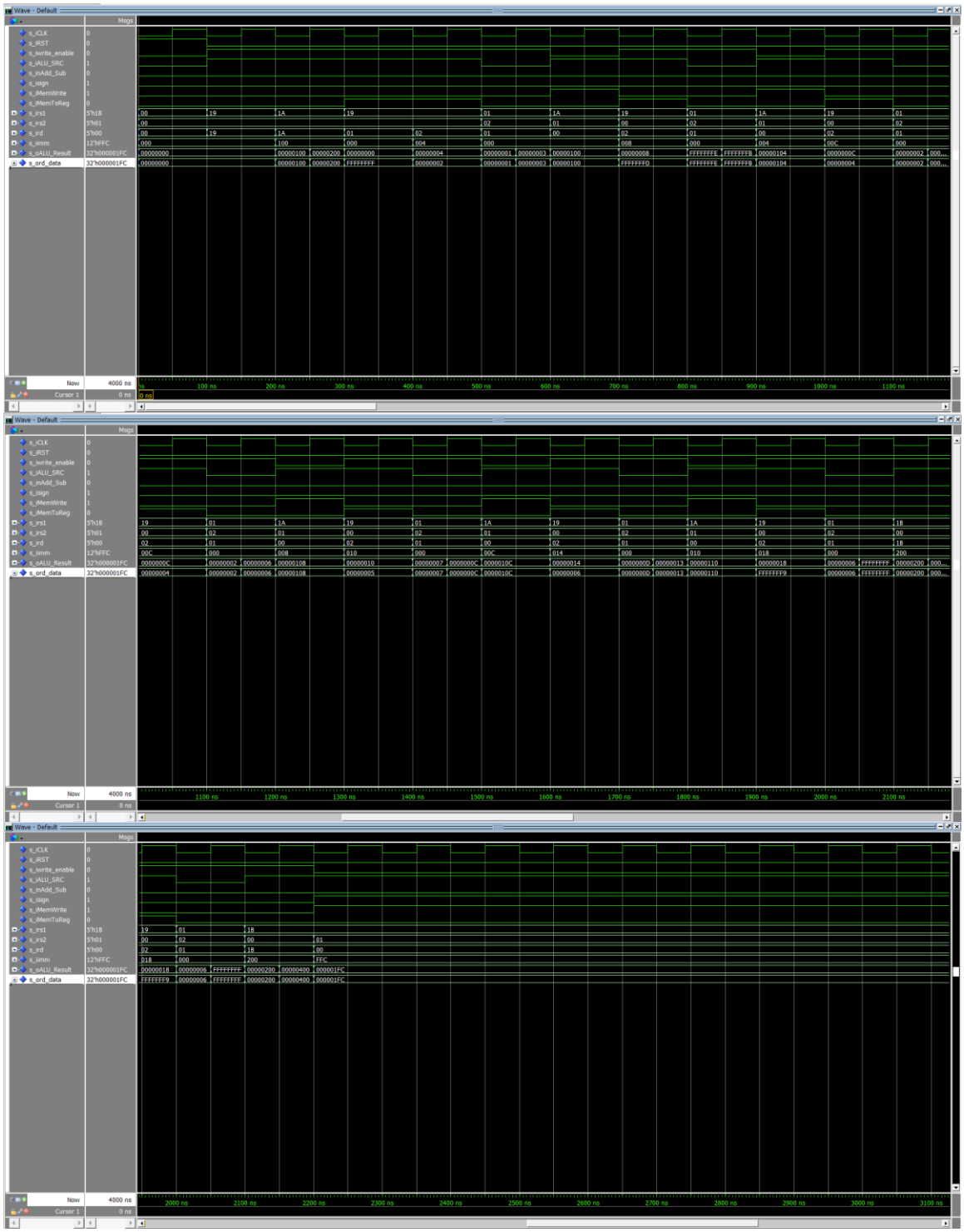
i_MemWrite – connects to the write enable (we) of the mem.vhd

i_MemToReg – is connected with the output (q) of the mem.vhd and the result of the ALU (o_ALU_Result) to select what will be written to the rd register.

[Part 7 (b)] Draw a schematic of a simplified RISC-V processor consisting only of the base components used in part 2, the extender component described in part 4, and the data memory from part 3.



[Part 7 (c)] Waveform.


Wave - Default
Msgs
s_iCLK
s_iRST
s_iwrite_enable
s_iALU_SRC
s_inAdd_Sub
s_isign
s_iMemWrite
s_iMemToReg
s_irs1
s_irs2
s_ird
s_iimm
s_sALU_Result
s_ord_data
Now
Cursor 1
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
1000 ns
1100 ns
1200 ns
1300 ns
1400 ns
1500 ns
1600 ns
1700 ns
1800 ns
1900 ns
2000 ns
2100 ns
2200 ns
2300 ns
2400 ns
2500 ns
2600 ns
2700 ns
2800 ns
2900 ns
3000 ns
3100 ns