# CprE 3810, Computer Organization and

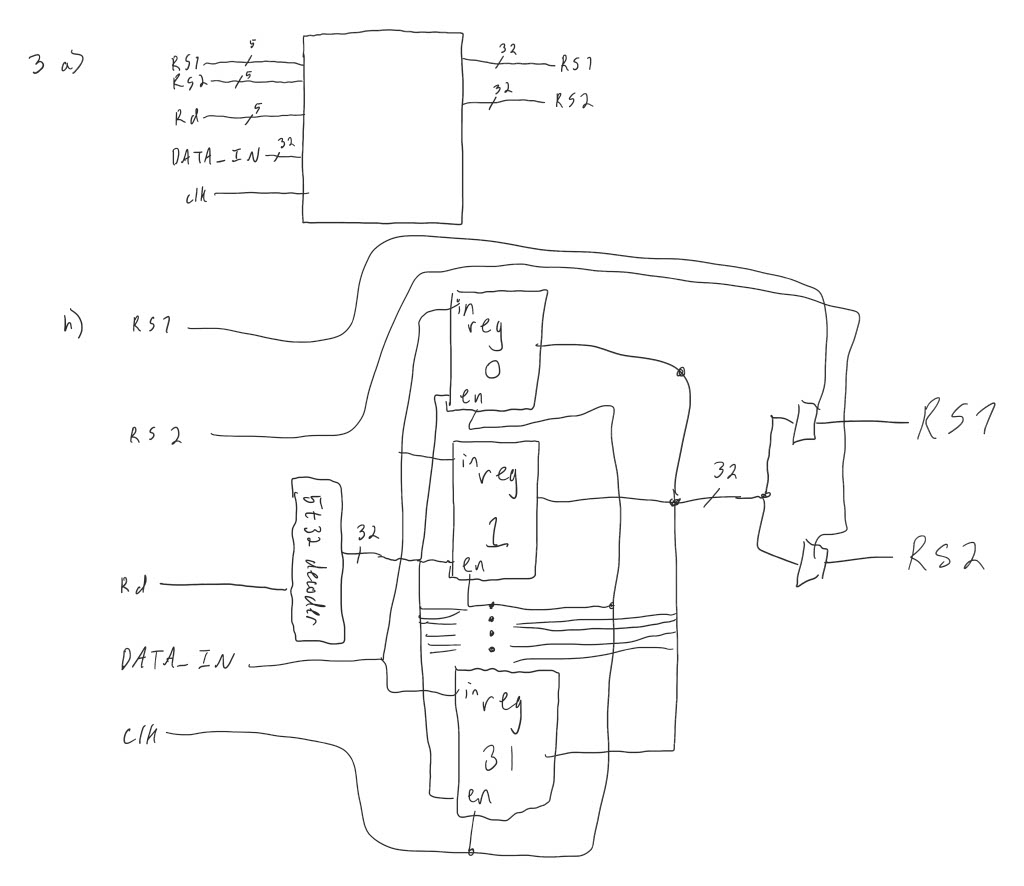
# Assembly-Level Programming

# Lab 2 Report

Student Name \_\_\_\_Zephaniah Gustafson\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

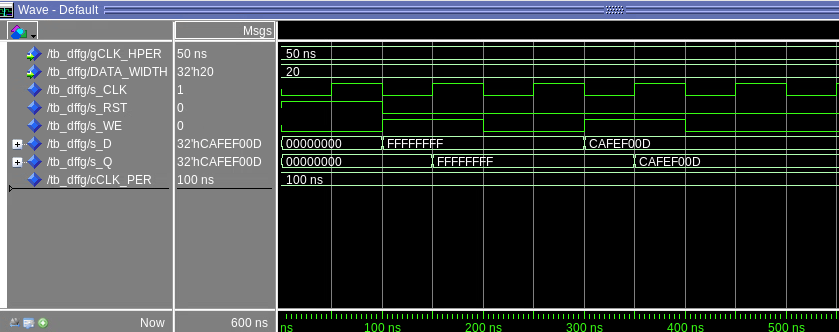
***Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the lab document for the context of the following questions****.*

[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.

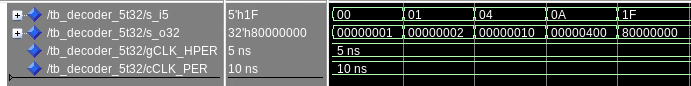
[Part 3 (c)] Waveform.



[Part 3 (d)] What type of decoder would be required by the RISC-V register file and why?

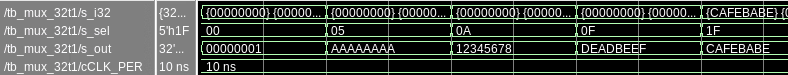
5 to 32 decoder because you need 32 inputs to enable writing to the 32 32-bit registers.

[Part 3 (e)] Waveform.

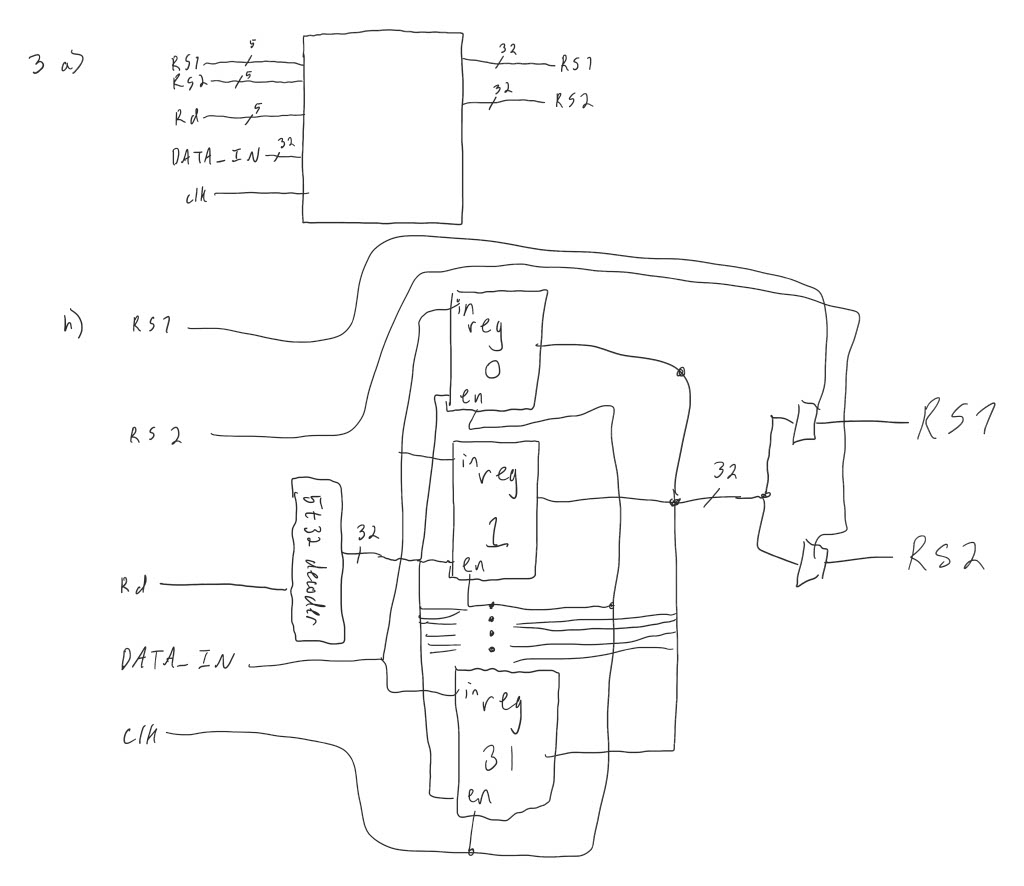


[Part 3 (f)] In your write-up, describe and defend the design you intend on implementing for the next part.

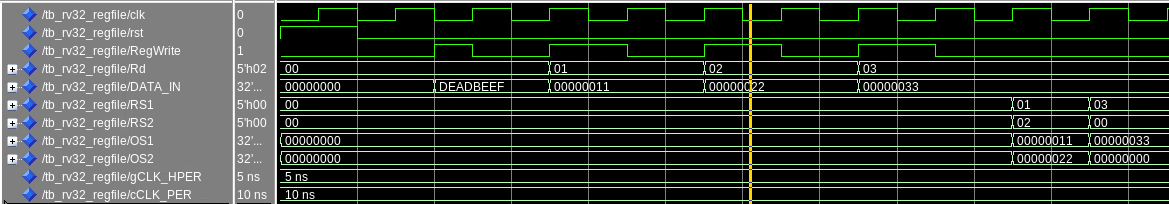
[Part 3 (g)] Waveform.



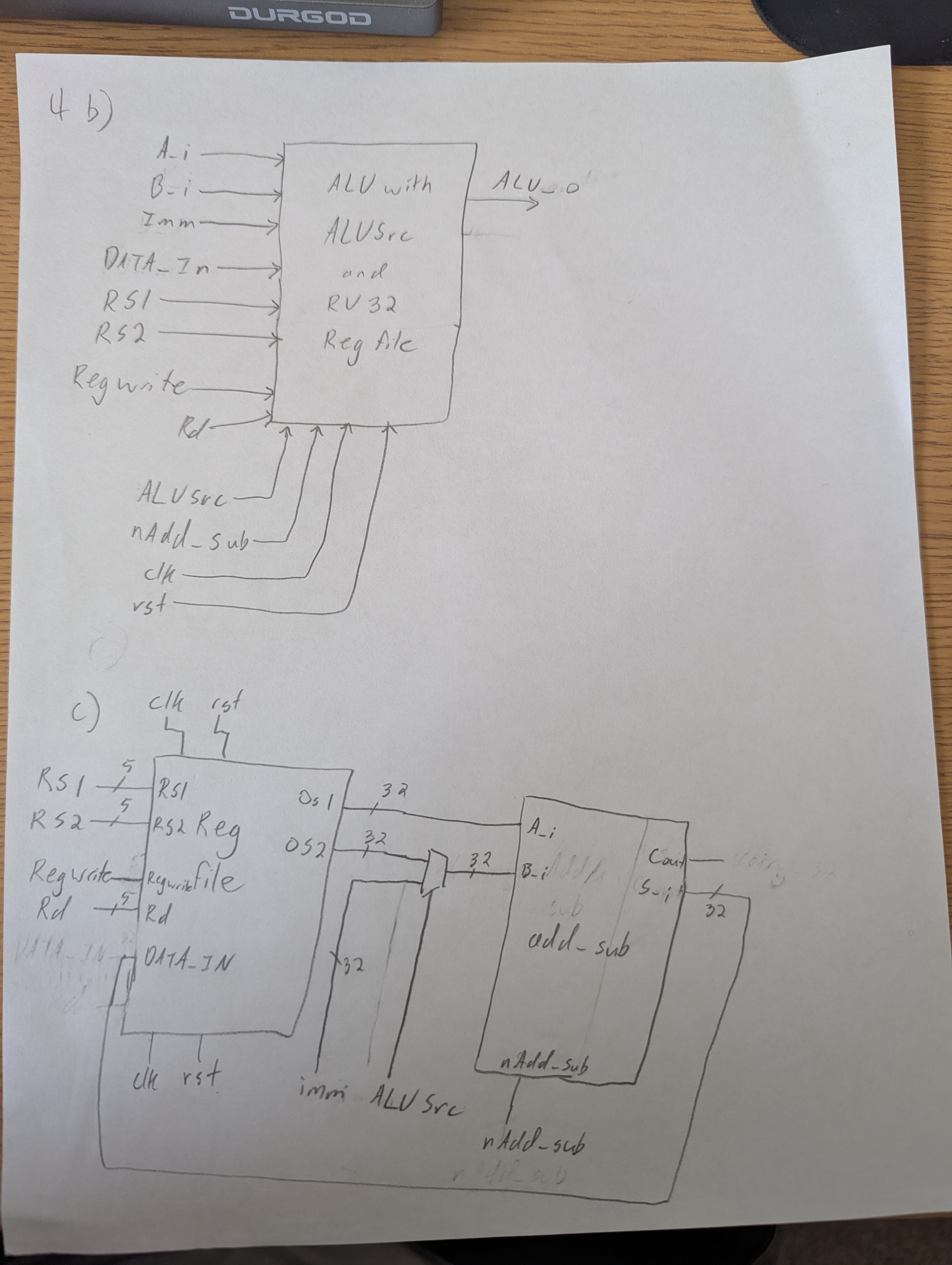
[Part 3 (h)] Draw a (simplified) schematic (i.e., components within the high-level blackbox) for the RISC-V register file, using the same top-level interface ports as in your solution describe above and using only the register, decoder, and mux VHDL components you have created.



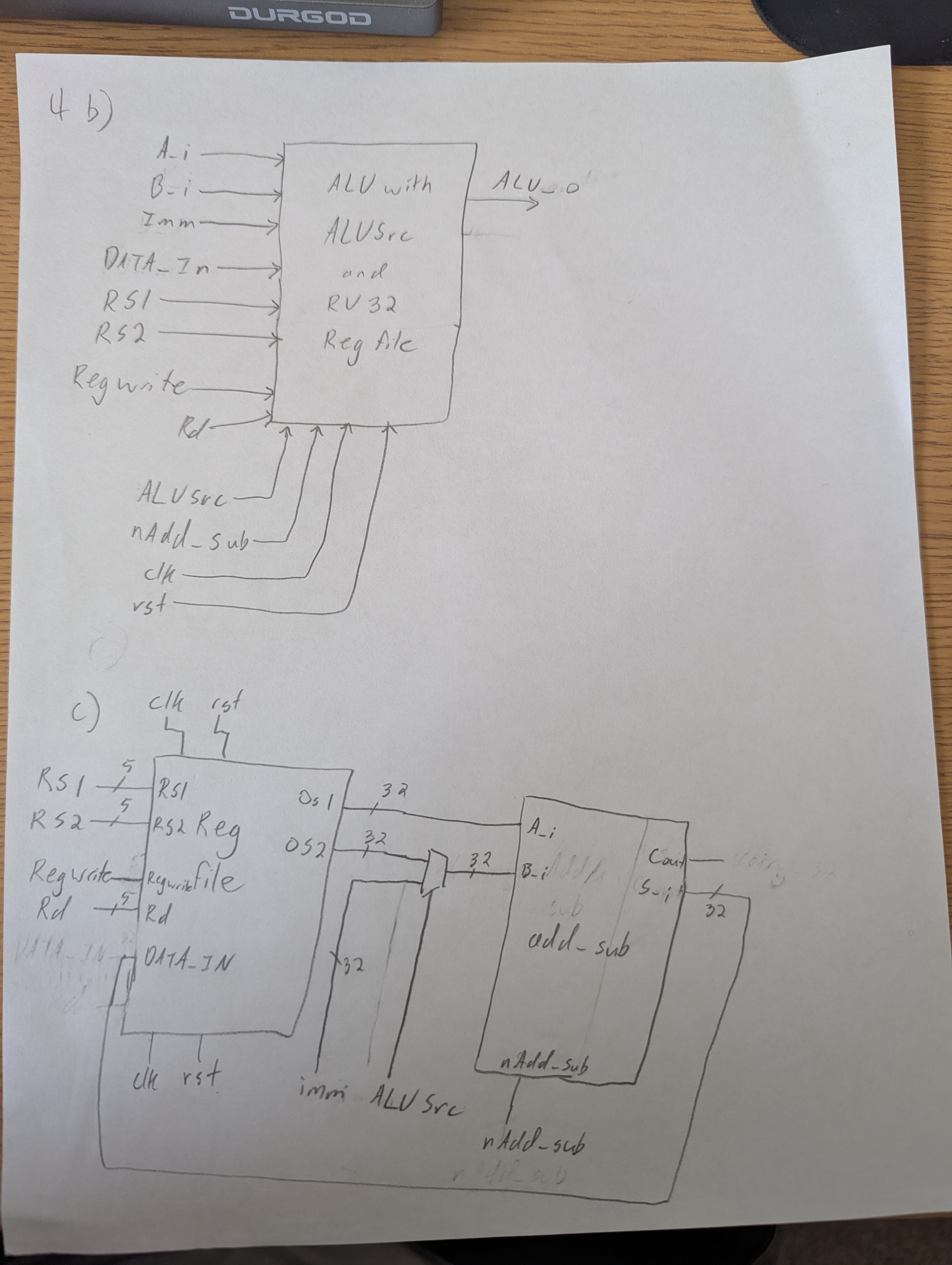
[Part 3 (i)] Waveform.



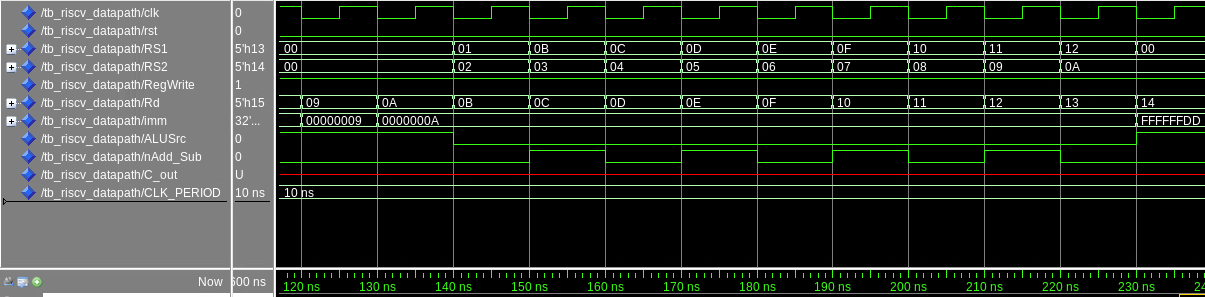
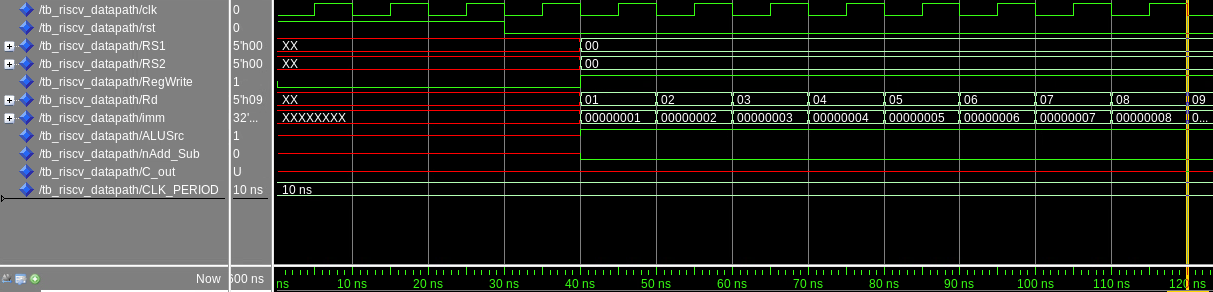
[Part 4 (b)] Draw a symbol for this RISC-V-like datapath.

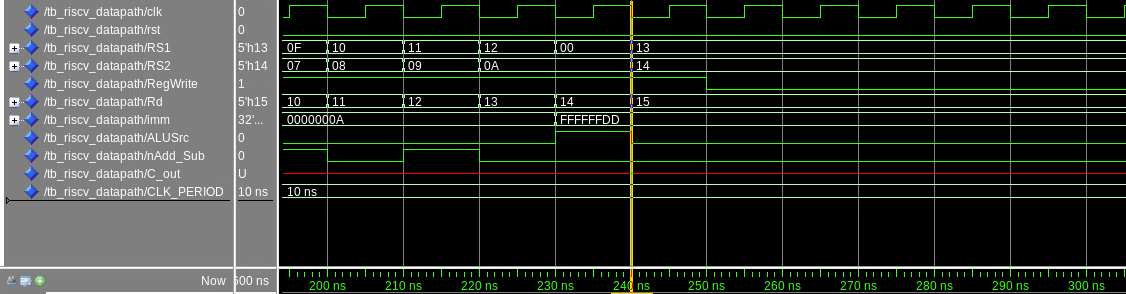


[Part 4 (c)] Draw a schematic of the simplified RISC-V processor datapath consisting only of the component described in part (a) and the register file from problem (1).



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





[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 is the size of the data that can be store in the registers. It is set to 32, the same size as a word, by default.

ADDR\_WIDTH is the size of the memory addresses for the register. Since this is set to 10 by default, that means that there are 2^10 registers or 1024 words.

Clk is the clock, it is required for updating the registers.

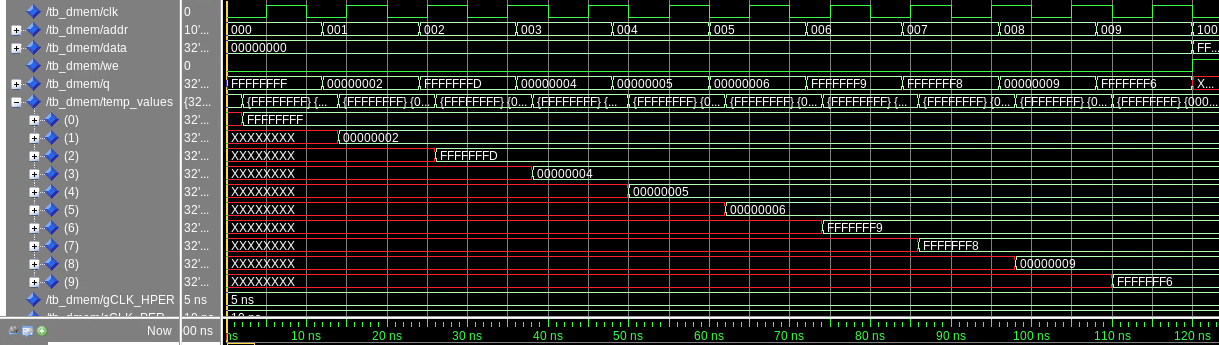
Addr is the address selector. It selects with word in memory we want to write to.

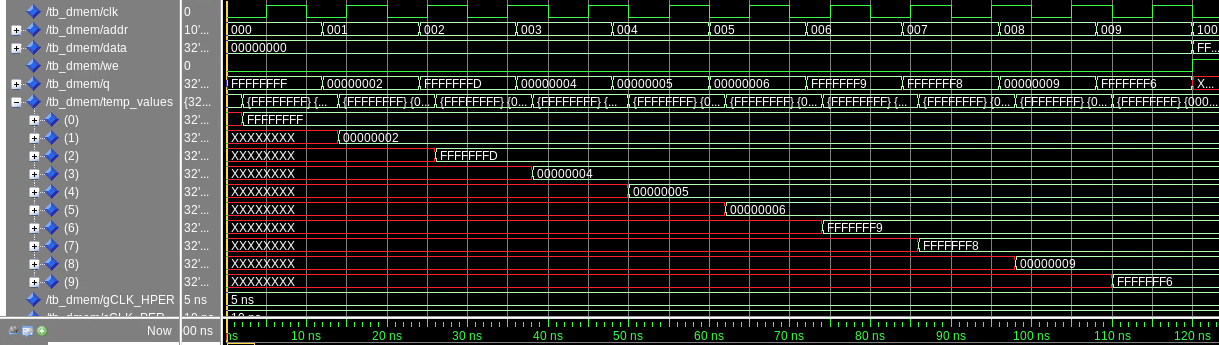
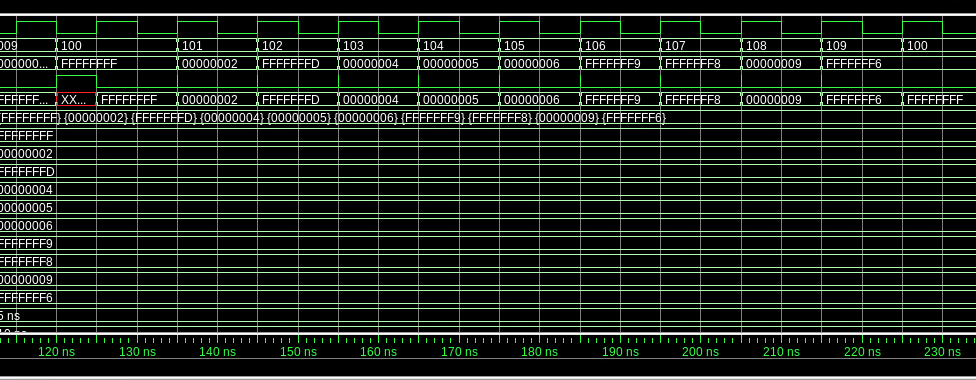
Data is the input bus of what we want to write to the word in memory.

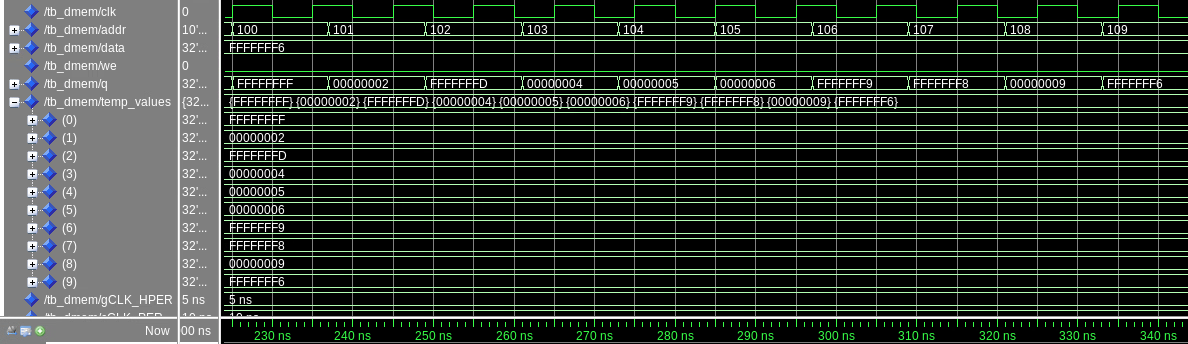
We is the write enable signal. It allows the memory to be changed.

Q is the output databus, it selected what is to be read from memory.

[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?

The RISC-V instructions that need to be sign extended are arithmetic immediates, loads, stores, branches, jump and links, and upper immediates. The instructions that require some values to be zero extended are unsinged ones like shift imeediate instructions and unsinged loads.

[Part 6 (b)] what are the different 12-bit to 32-bit “extender” components that would be required by a RISC-V processor implementation?

The different 16-bit to 32-bit extender components depend on the how the 12-bit immediate field is interpreted. The first component would be the sign extended that in the 12-bit immediate, would take the most significant digit and repeat it until you get 32 bits. This would be for I-type immediates and S-type offsets. Then the next component would be a zero extender component. This component would take the zero extended instructions mentioned in part a and add zeroes to the front.

[Part 6 (d)] Waveform.

JKJKJKJ

[Part 7 (a)] what control signals will need to be added to the simple processor from part 3? How do these control signals correspond to the ports on the mem.vhd component analyzed in part 4?

You would need to be a control signal for both the load and store controls, which would likely be hooked to a mux to control if the writing goes to the memory or registers. These correspond to the we and q prorts in part 4. Specifically, we controls store and q controls load. Although, q always loads to the given address.

[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.