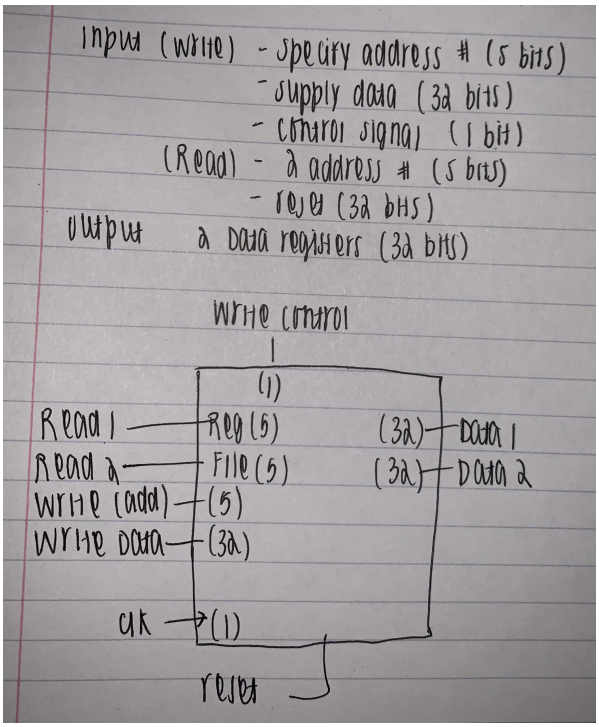
CprE 381, Computer Organization and Assembly-Level Programming

Lab 2 Report

Student Name _____Nicaela Rose_____

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 2 (a)] Draw the interface description for the MIPS register file. Which ports do you think are necessary, and how wide (in bits) do they need to be?

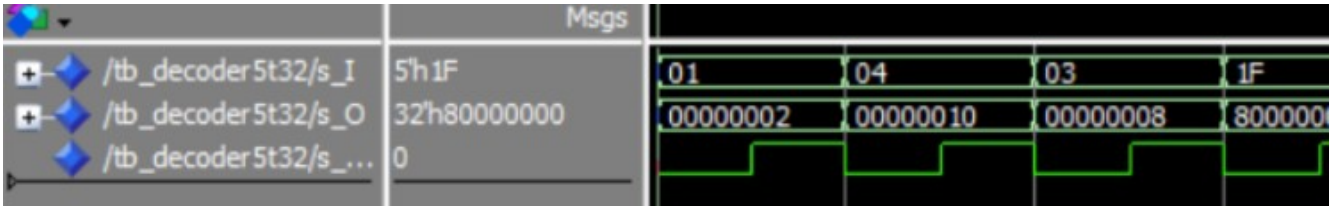


[Part 2 (b)] Create an N-bit register using this flip-flop as your basis.

[Part 2 (c)] Waveform.

[Part 2 (d)] What type of decoder would be required by the MIPS register file and why?
Since there are 32 registers in the MIPS register file, we need a 5:32 decoder to determine which register to write to

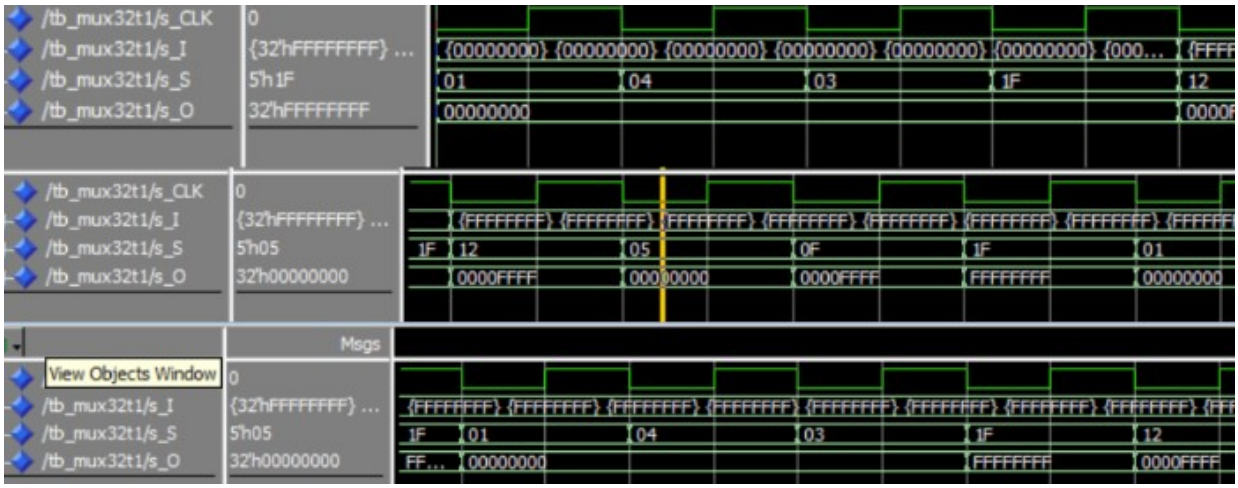
[Part 2 (e)] Waveform.



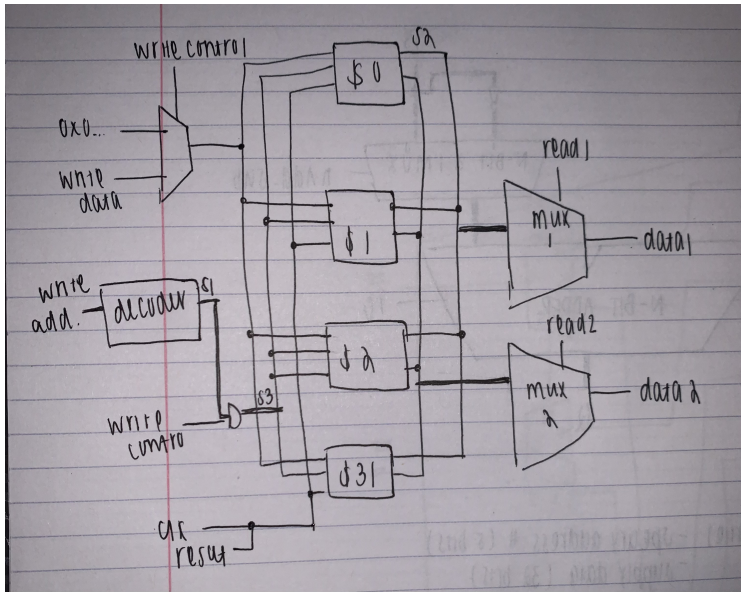
[Part 2 (f)] In your write-up, describe and defend the design you intend on implementing for the next part.

I am going to implement a 2D array that is 32x32. The 5 selected bits will choose which of the 32 inputs to load and transfer the 1D array of 32 bits through to the output

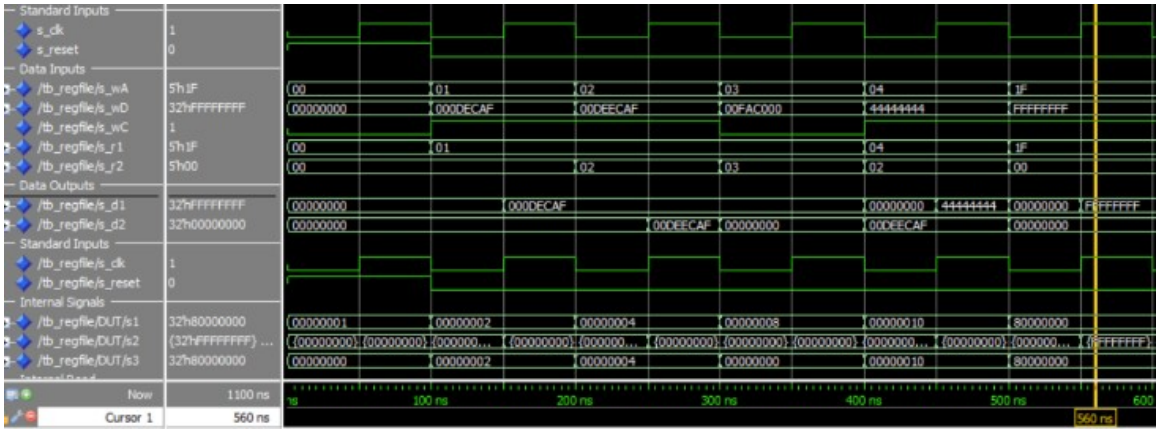
[Part 2 (g)] Waveform.



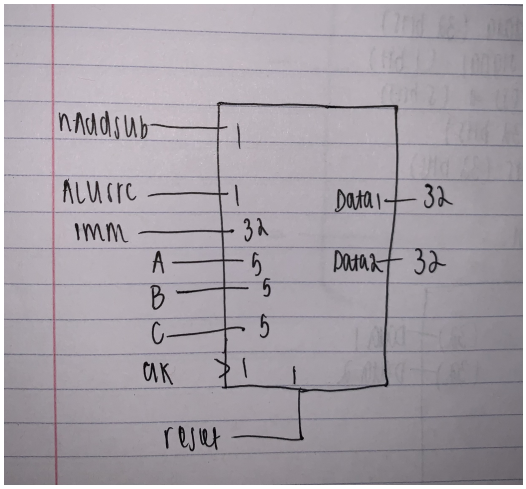
[Part 2 (h)] Draw a (simplified) schematic for the MIPS register file, using the same top-level interface ports as in your solution describe above and using only the register, decoder, and mux VHDL components you have created.



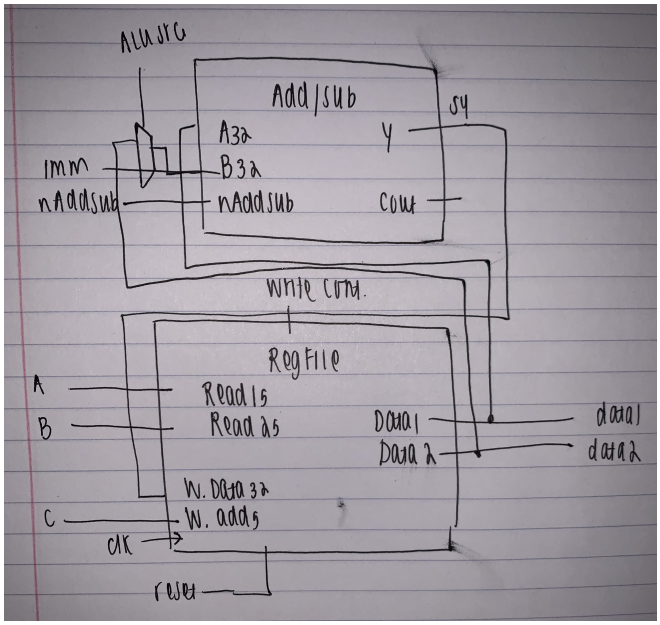
[Part 2 (i)] Waveform.



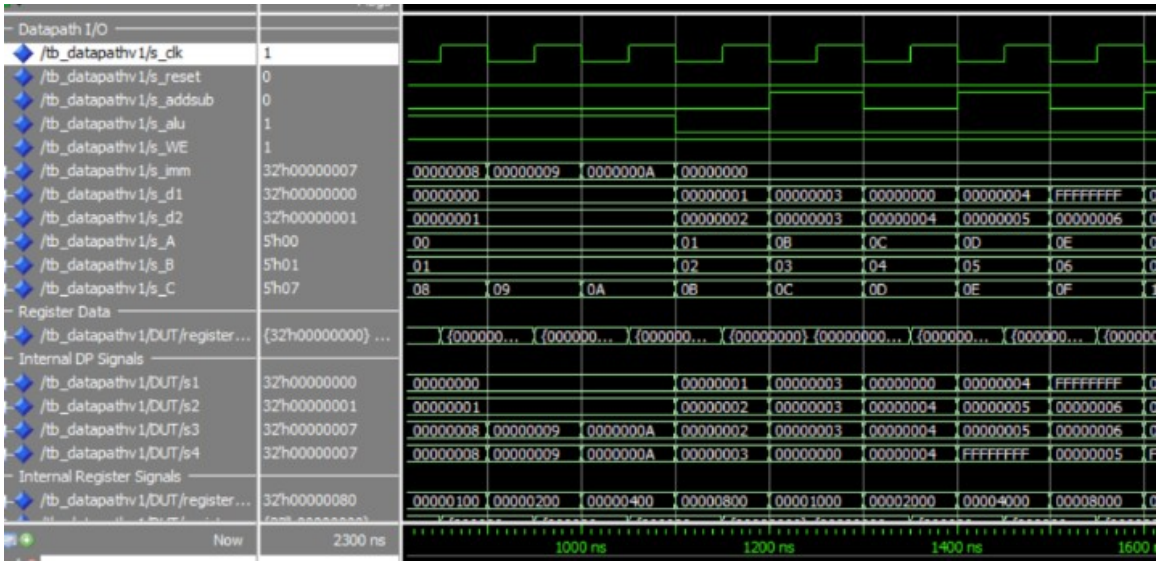
[Part 3 (b)] Draw a symbol for this MIPS-like datapath.

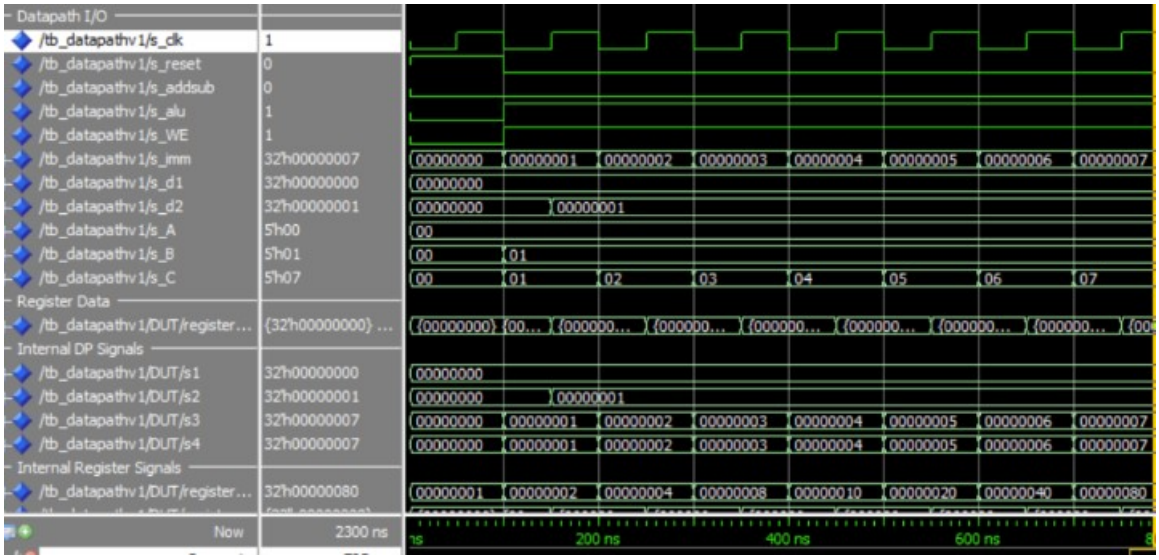


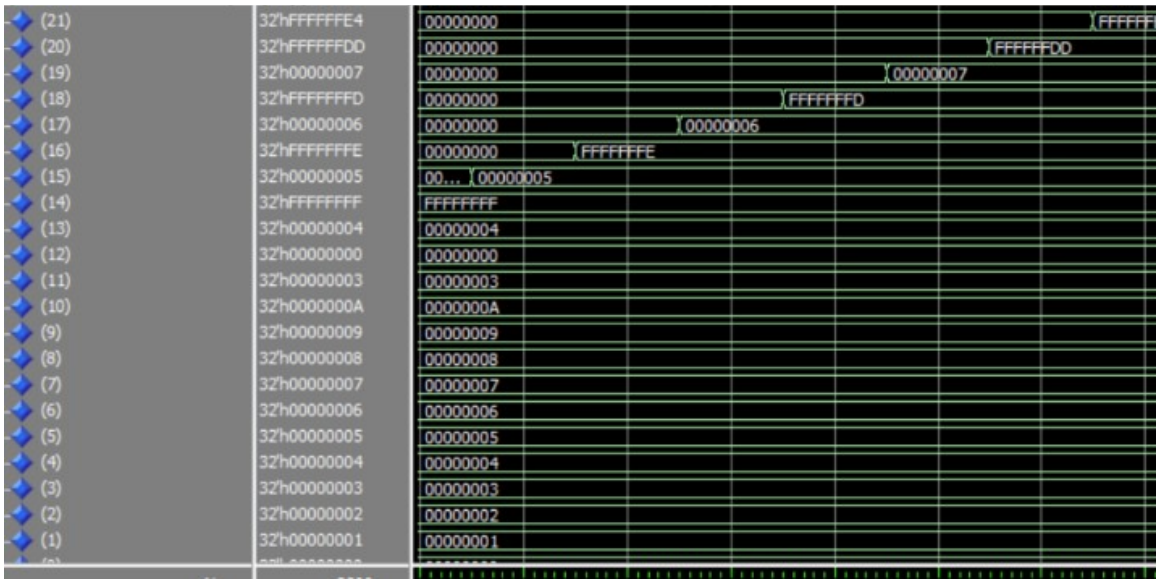
[Part 3 (c)] Draw a schematic of the simplified MIPS processor datapath consisting only of the component described in part (a) and the register file from problem (1).



[Part 3 (d)] Include in your report waveform screenshots that demonstrate your properly functioning design. Annotate what the final register file state should be.







[Part 4 (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 a generic that describes the size of the memory module. Like the registers, it is sized at 32 bits

Address Width is a generic that describes how many bits are needed to specify a module and it shows how many modules there are. Because the number of modules is often based on the address width, its a good assumption to say there is 1024 = 2^10 modules

Clk is a clock that allows the module to function on edges

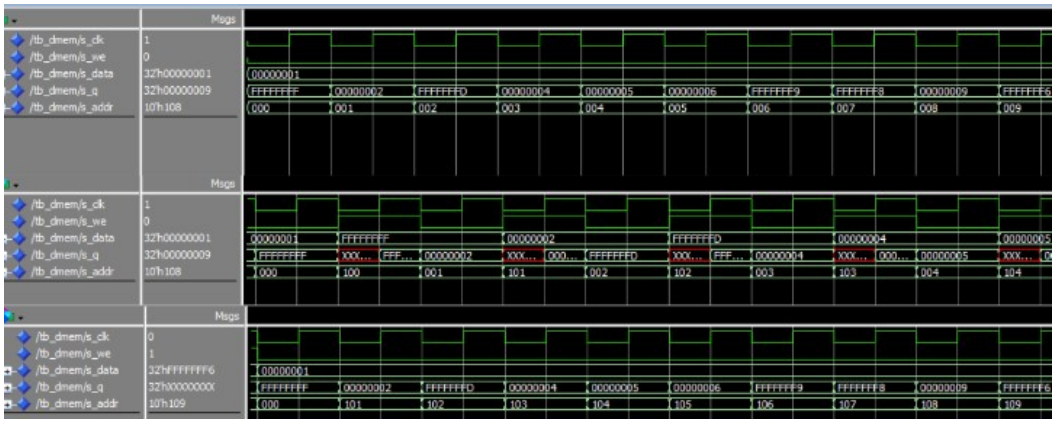
Addr is a memory address that is being written to and outputting dataflow

Data describes the input data to the memory module

WE describes write enable. There always a data input to the register, but only overwrites the current contents when WE is highlighted

Q describes the output of memory and returns the data width. Internally, its plugged back into the input in case the WE is low so that the module can remember

[Part 4 (c)] Waveforms.



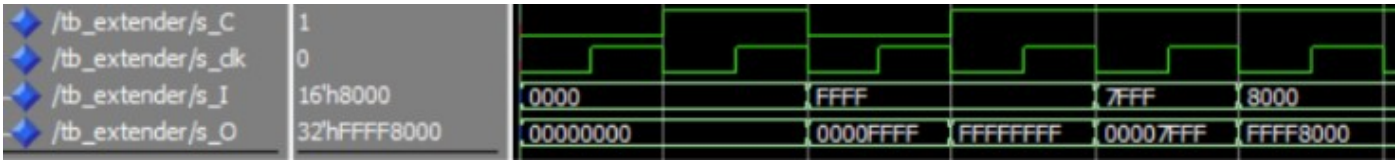
[Part 5 (a)] What are the MIPS instructions that require some value to be sign extended? What are the MIPS instructions that require some value to be zero extended?

Instructions such as addiu deal in unsigned numbers. Therefore we want to zero extend the immediate. Addi, store, and load give signed immediates because sometimes their values are negative so they should be zero extended. Logic instructions will also be zero extended.

[Part 5 (b)] what are the different 16-bit to 32-bit “extender” components that would be required by a MIPS processor implementation?

The MIPS processor requires a sign extender and a zero extender. The first 16 bits will be copied, and the last 16 bits are set to the 15 sign bit and the control signal. Therefore if the control signal is zero, then the bits will also be zero. If its 1, then the bits will be set to the value of the sign bit

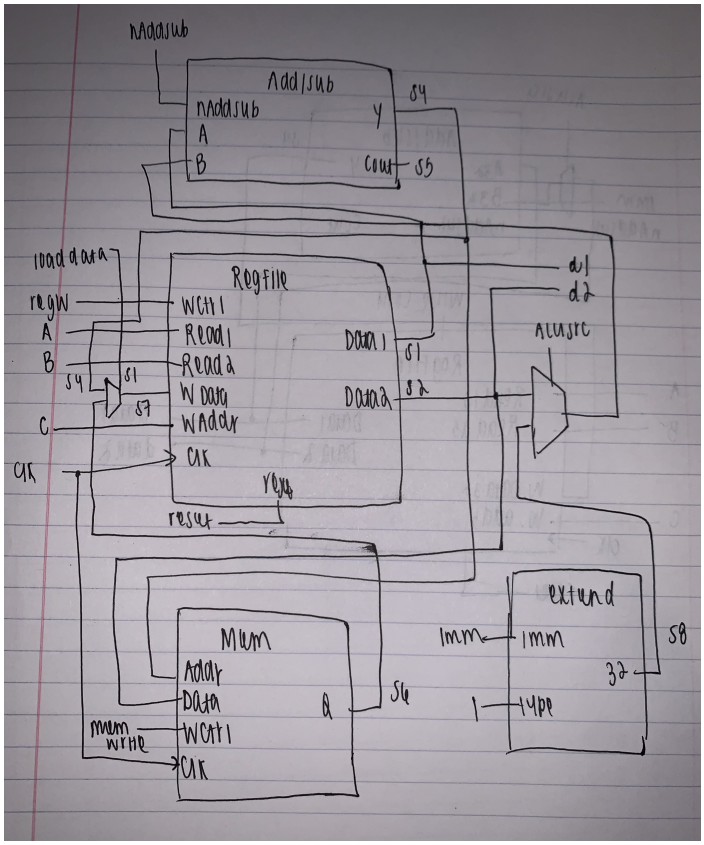
[Part 5 (d)] Waveform.



[Part 6 (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?

MemoryWrite is to determine when the memory should change to reflect the input and LoadData which is 0 if the data should be accessed from the ALU but is 1 if data should be loaded from the memory. I did not add a control signal for the extend type since all the instructions require sign extension, so I wrote 1 to the input.

[Part 6 (b)] Draw a schematic of a simplified MIPS 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 6 (c)] Waveform.