CprE 381, Computer Organization and Assembly-Level Programming

Lab 2 Report

Student Name William Clemmons

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 0] Describe the provided DFF in dffg.vhd in terms of edge sensitivity and reset type (active low/high and synchronous/asynchronous

dffg.vhd is an active high DFF with an asynchronous reset

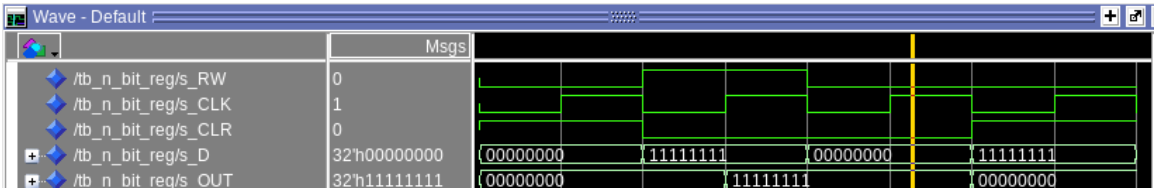
[Part 2 (a)] Draw the interface description for the MIPS register file. Which ports do you think are necessary, and how wide (in bits) do they need to be?

Needed Ports (bits needed):

- Inputs
  - value to be inputed (32)
  - read/write bit (1)
  - Address1 (32)
  - Clear (1)
  - Clock (1)
- Outputs
  - Current value of the register

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

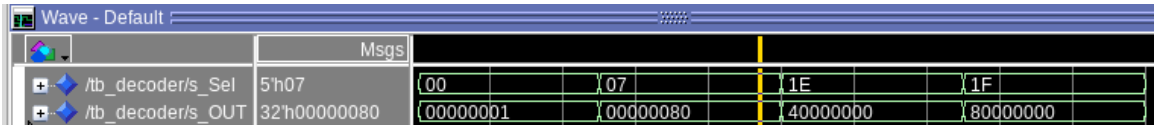
[Part 2 (c)] Waveform.



[Part 2 (d)] What type of decoder would be required by the MIPS register file and why?

We need a 5:32 register because we plan to have 32 register. That would require 4 bit to represent all numbers between 0 and 31

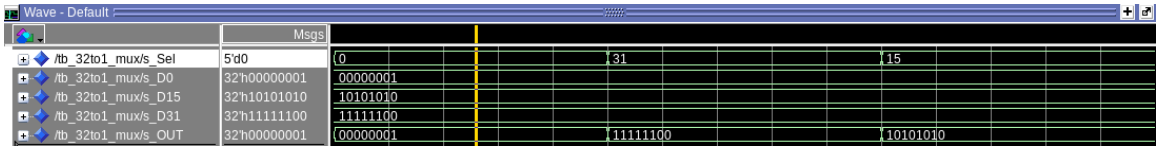
[Part 2 (e)] Waveform.



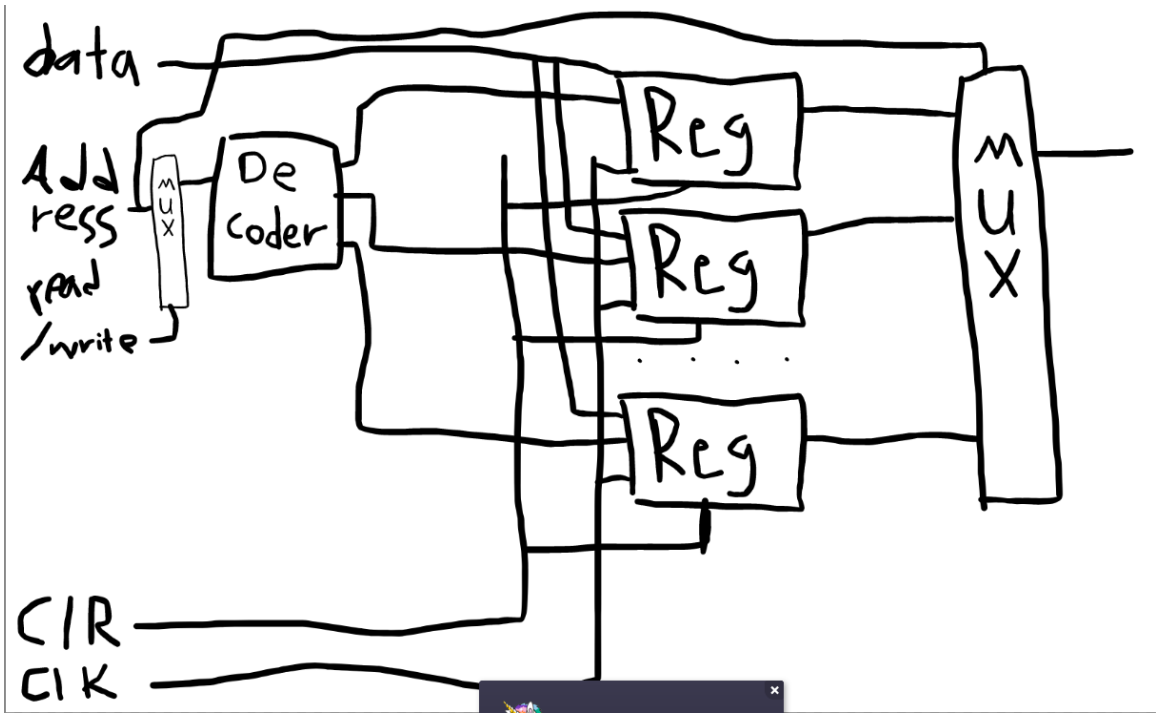
[Part 2 (f)] In your write-up, describe and defend the design you intend on implementing for the next part.

For my 32bit 32:1 Mux, I plan on creating one from scratch rather than trying to modify some of the code that I was give or the code that I have already wrote. My plan int to use data-flow vhdl to make coding the hardware easier since I wont have to manage many gates or components. The major benefits of doing it this way is that it will take less time to code, which allows me to start debugging quicker, and also that once I start debugging errors will be easier to find since each output would have its own line of code.

[Part 2 (g)] Waveform.

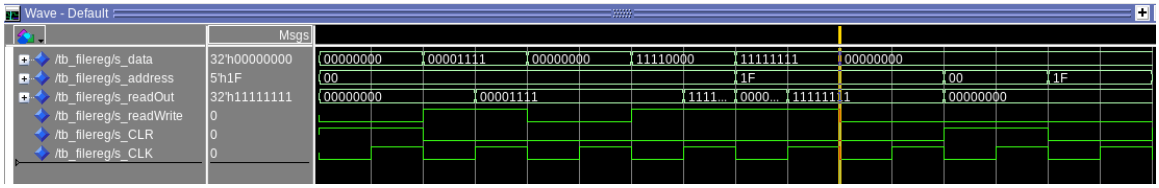


[Part 2 (h)] Draw a (simplified) schematic for the MIPS register file, using the same top-level interface ports as in your solution describe above and using only the register, decoder, and mux VHDL components you have created.

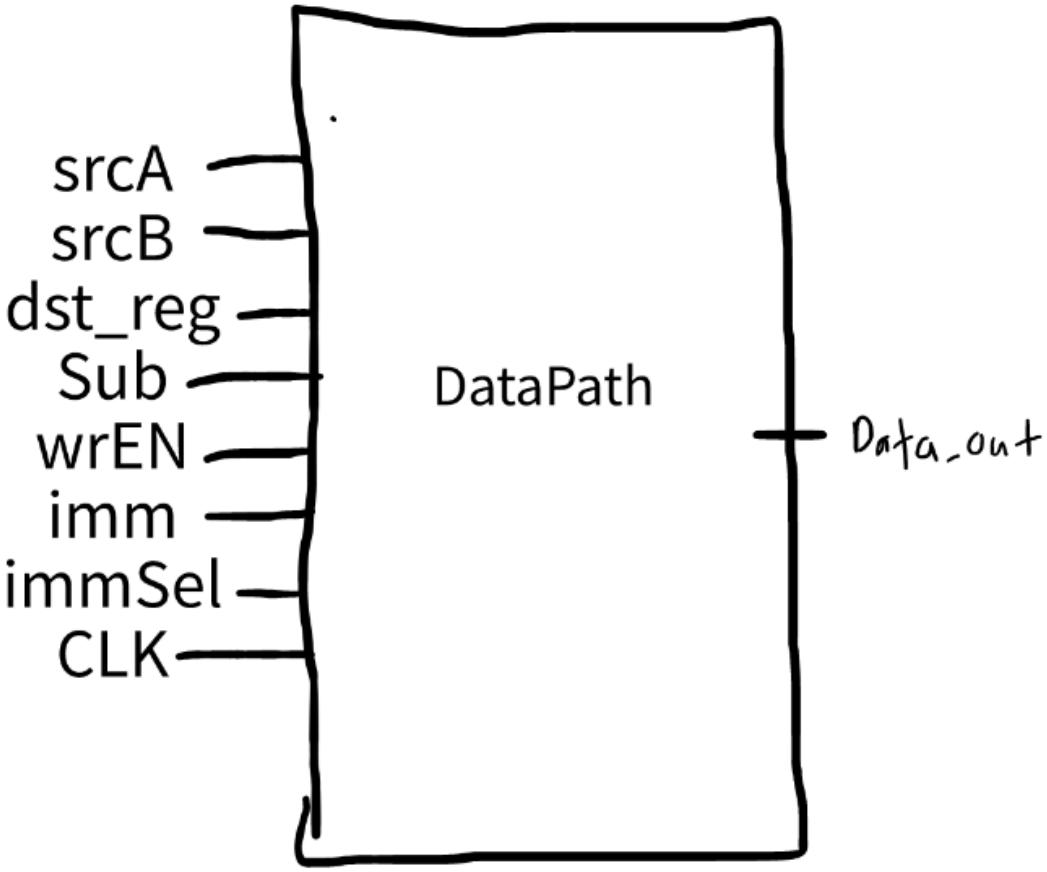


I realized that with the schematic above there would be no way for me to set all the register to read at the same time. To fix this, I modified the decoder to have a Write enable. If the write enable is 0 then all the decoders output will be 0. This will replace the mux that connects the address and read/write inputs to the decoder

[Part 2 (i)] Waveform.

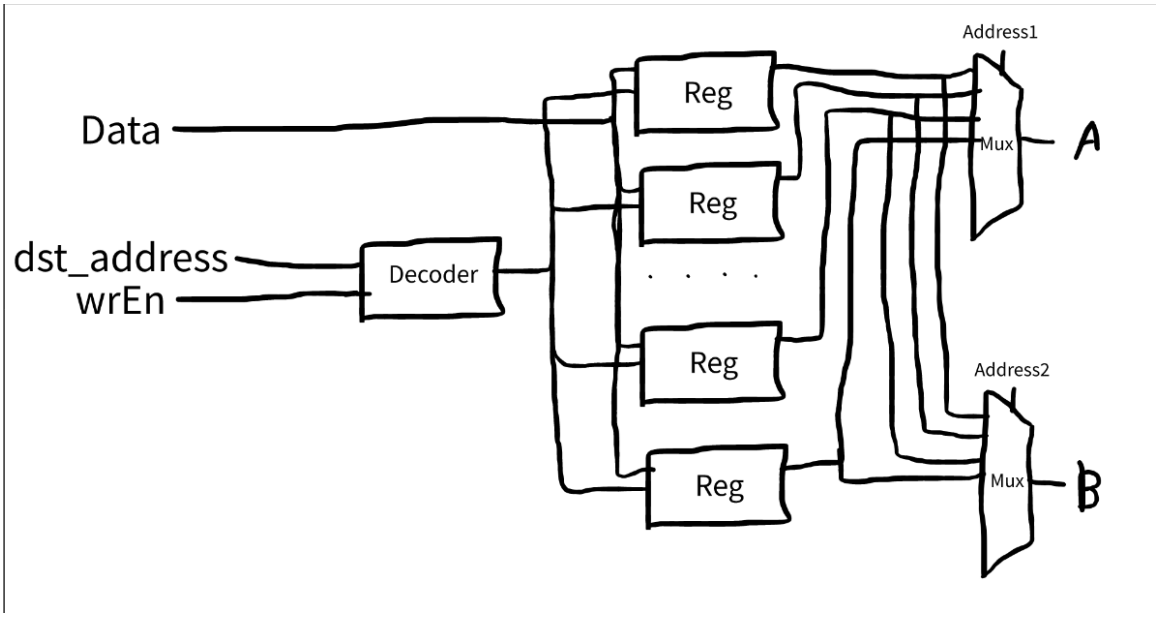


[Part 3 (b)] Draw a symbol for this MIPS-like datapath.

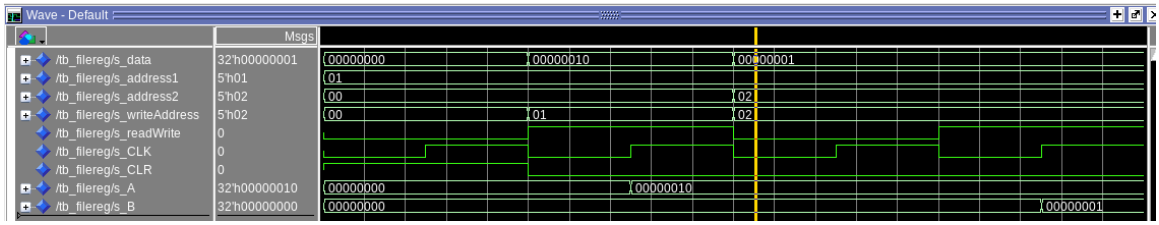


I have just realized that I created my register file incorrectly. It needs to have a two total outputs as well as two addition inputs. One to select the other register to read from and a write destination register.

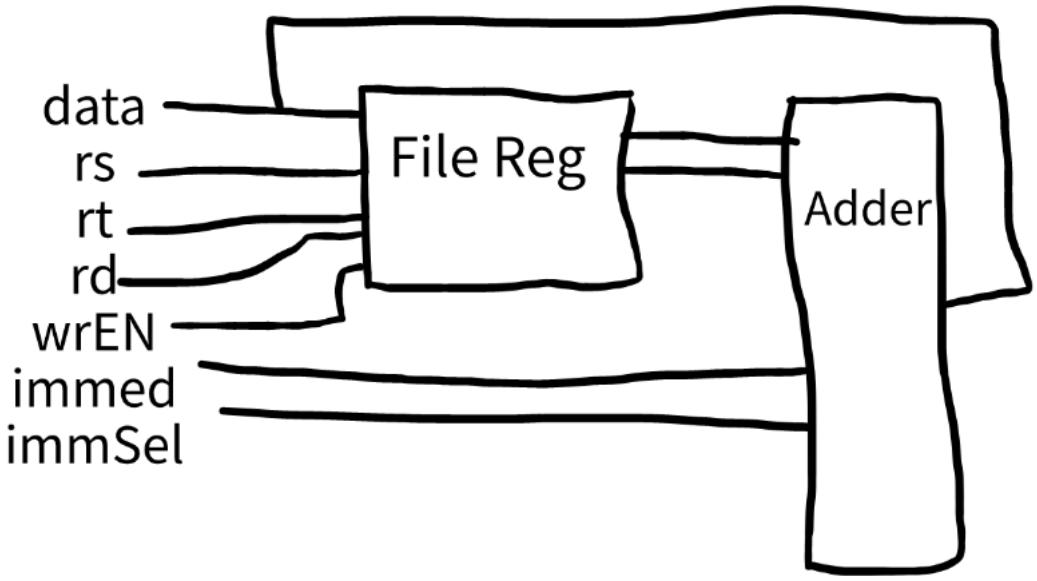
So now my register looks more like this



The new waveform look like the following



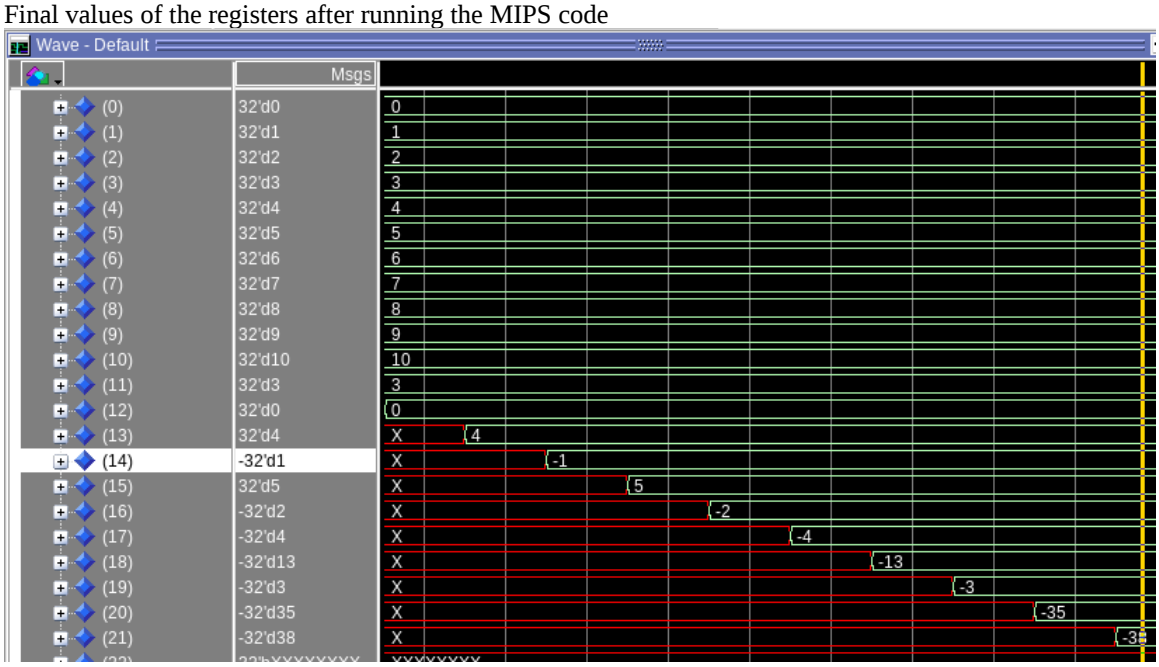
[Part 3 (c)] Draw a schematic of the simplified MIPS processor datapath consisting only of the component described in part (a) and the register file from problem (1).



Our datapath does not need to have a data input. The only way to directly change the value of a register should be by assigning a register to an immediate value. So, I have removed the data input from my design

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

Final values of the registers after running the MIPS code



[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 – the total amount of bits the data will consist of

ADDR_WIDTH – the amount of bits that will be used to represent memory address. In the given case, (2^10) – 1 is the highest memory address

Clk- system clock

address – The place in memory to be written to or read from

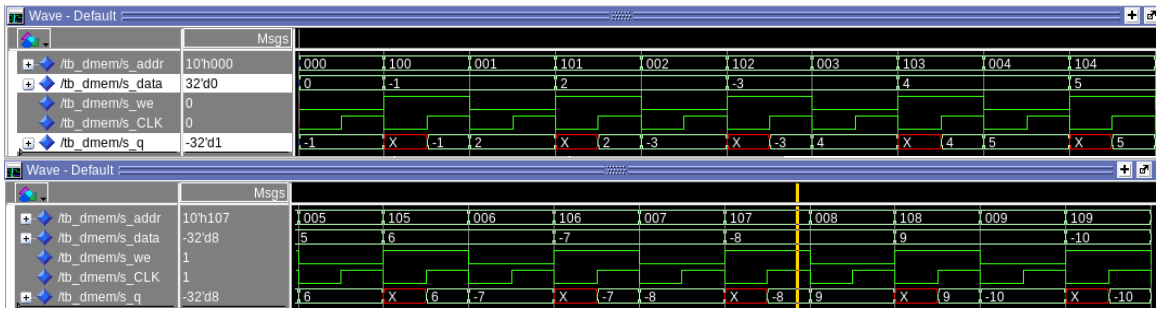
data – the value to be written to the memory address given

we – allows the memory to be written to

q – The data that was just assigned to a memory address is then read from the memory address and is given as an output

[Part 4 (c)] Waveforms.

For every two click cycles, I decided I wanted to read the data from the memory address then copy it over to the new address. I did it this way so that I would not have to do to passes of the memory, once to read and again to copy/write.



[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?

SignExtended – addi, lw, sw, sll

ZeroExtended – andi, ori

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

We are going to need a component to do sign extension and a component to do zero extension

[Part 5 (d)] Waveform.

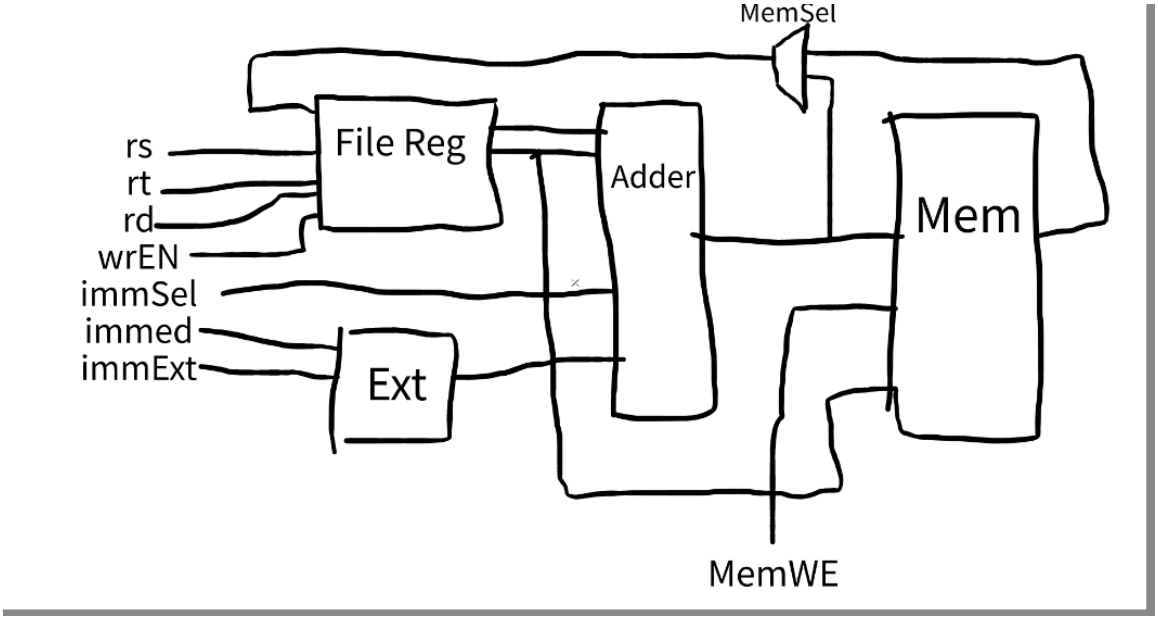
[Part 6 (a)] what control signals will need to be added to the simple processor from part 2? How do these control signals correspond to the ports on the mem.vhd component analyzed in part 3?

MemSel – Decides whether to read the value from the adder or read the value from memory for register data

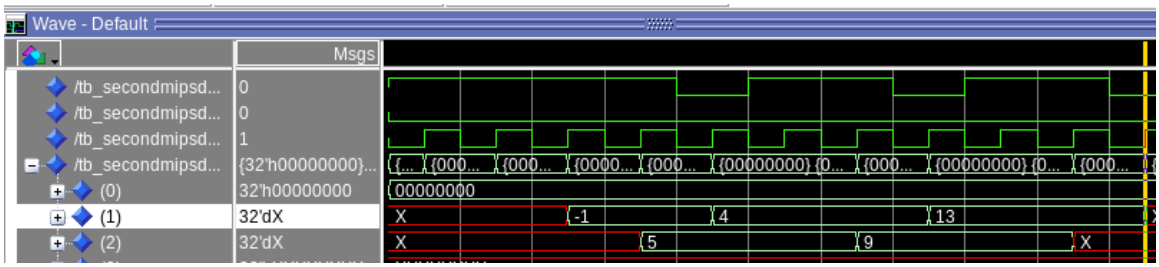
MemWE – allows the use of sw to write to the memory addres given by the adders instead of reading from it

ImmExt – Decides whether to do sign extension or zero extension

[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.



The code run for quite a while afterwards but there is no valid outputs because the addresses that are being accessed are not initialized