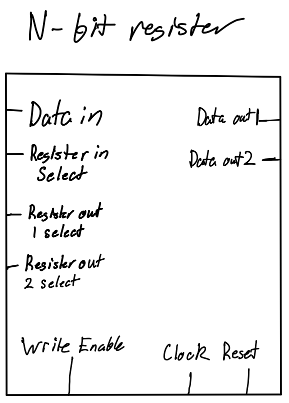
CprE 381, Computer Organization and Assembly-Level Programming

Lab 2 Report

Student Name Will Galles

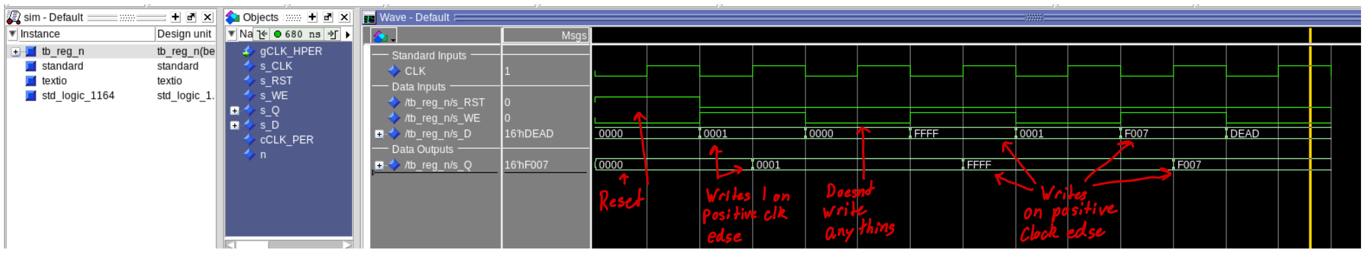
Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the lab document for the context of the following questions.

[Part 2 (a)] Draw the interface description (i.e., the "symbol" or high-level blackbox) 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.

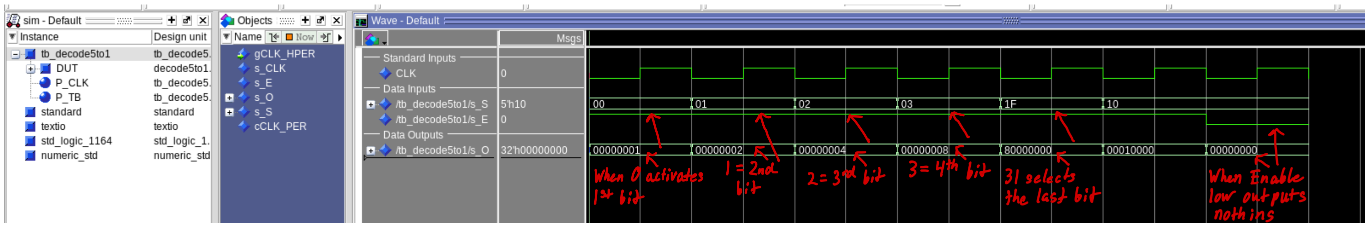
[Part 2 (c)] Waveform.



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

We need a 5 to 32 decoder with an enable function. We need five select bits to be able to produce 32 different outputs for all of our registers. We also wan the enable functionality to be able to control later if we write to the register file at all

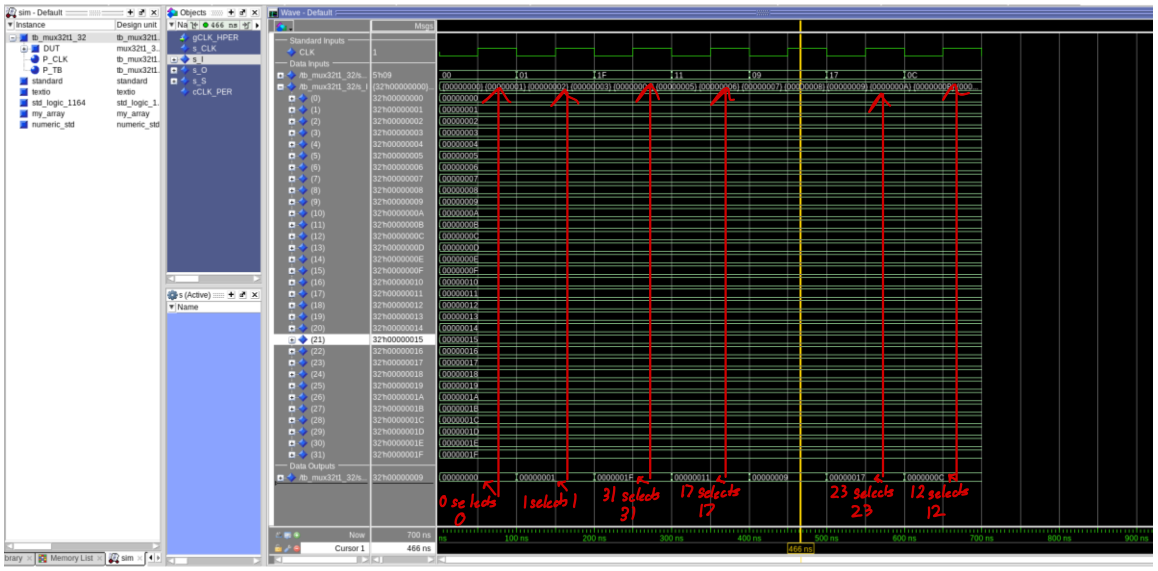
[Part 2 (e)] Waveform.



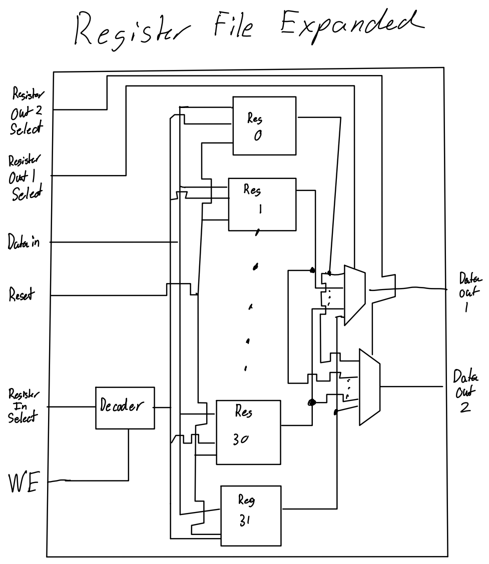
[Part 2 (f)] In your write-up, describe and defend the design you intend on implementing for the next part.

I decided to follow closer to a dataflow implementation for my 32 to 1 mux. This was because I found that it is easier to not mess up anything when you are not wiring all the internal signals between the layers of the muxes. The dataflow was also easier to read when I came back to review it at a later time.

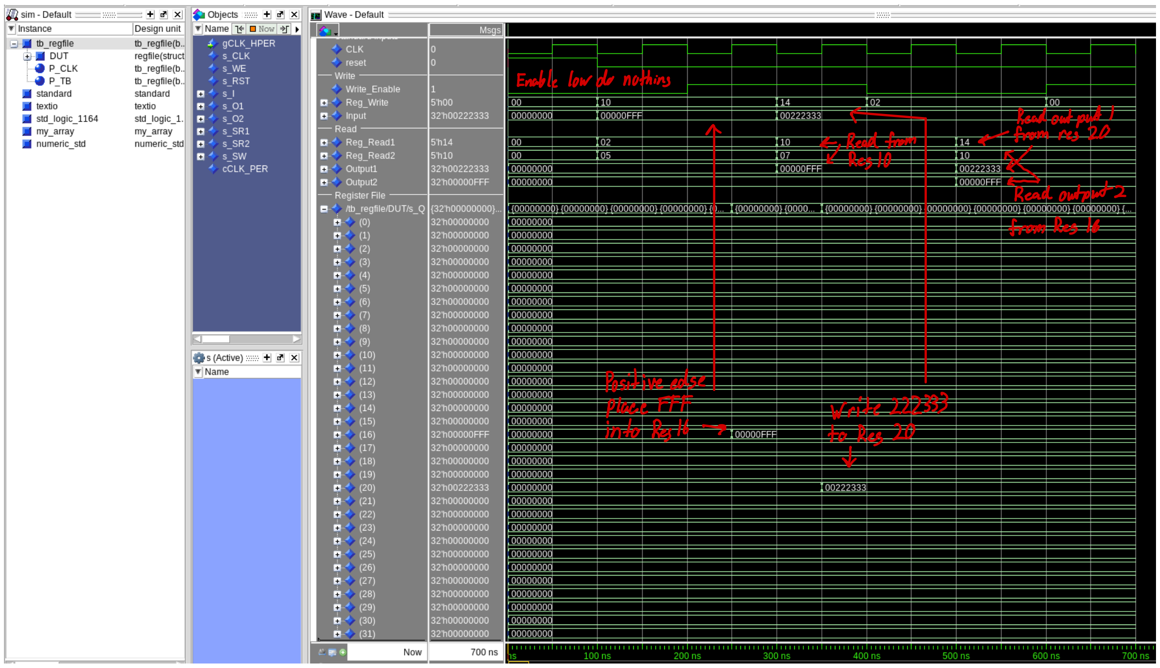
[Part 2 (g)] Waveform.



[Part 2 (h)] Draw a (simplified) schematic (i.e., components within the high-level blackbox) 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.



```
Instance        Design unit
tb_regfile      tb_regfile(b..
  DUT           regfile(struct
  P_CLK         tb_regfile(b..
  P_TB          tb_regfile(b..
standard        standard
textio          textio
std_logic_1164  std_logic_1.
my_array        my_array
numeric_std     numeric_std
```

```
Name
gCLK_HPER
s_CLK
s_WE
s_RST
s_I
s_O1
s_O2
s_SR1
s_SR2
s_SW
cCLK_PER
```

```
                        Msgs
CLK                     0
reset                   0
Write
Write_Enable            1
Reg_Write               5'h00            00        10        14        02        00
Input                   32'h00222333     00000000  00000FFF  00222333
Read
Reg_Read1               5'h14            00        02        10        14
Reg_Read2               5'h10            00        05        07        10
Output1                 32'h00222333     00000000  00000FFF  00222333
Output2                 32'h00000FFF     00000000  00000FFF
Register File
/tb_regfile/DUT/s_Q     {32'h00000000}...  {00000000} {00000000} {00000000} {00000000} {0...  {00000000} {0000...  {00000000} {00000000} {00000000} {00000000} {00000000} {00000000} {...
(0)                     32'h00000000     00000000
(1)                     32'h00000000     00000000
(2)                     32'h00000000     00000000
(3)                     32'h00000000     00000000
(4)                     32'h00000000     00000000
(5)                     32'h00000000     00000000
(6)                     32'h00000000     00000000
(7)                     32'h00000000     00000000
(8)                     32'h00000000     00000000
(9)                     32'h00000000     00000000
(10)                    32'h00000000     00000000
(11)                    32'h00000000     00000000
(12)                    32'h00000000     00000000
(13)                    32'h00000000     00000000
(14)                    32'h00000000     00000000
(15)                    32'h00000000     00000000
(16)                    32'h00000FFF     00000000  00000FFF
(17)                    32'h00000000     00000000
(18)                    32'h00000000     00000000
(19)                    32'h00000000     00000000
(20)                    32'h00222333     00000000  00222333
(21)                    32'h00000000     00000000
(22)                    32'h00000000     00000000
(23)                    32'h00000000     00000000
(24)                    32'h00000000     00000000
(25)                    32'h00000000     00000000
(26)                    32'h00000000     00000000
(27)                    32'h00000000     00000000
(28)                    32'h00000000     00000000
(29)                    32'h00000000     00000000
(30)                    32'h00000000     00000000
(31)                    32'h00000000     00000000
Now                     700 ns           ns   100 ns   200 ns   300 ns   400 ns   500 ns   600 ns   700 ns
```

Enable low do nothing

Read from Res 10

Read output 1 from res 20

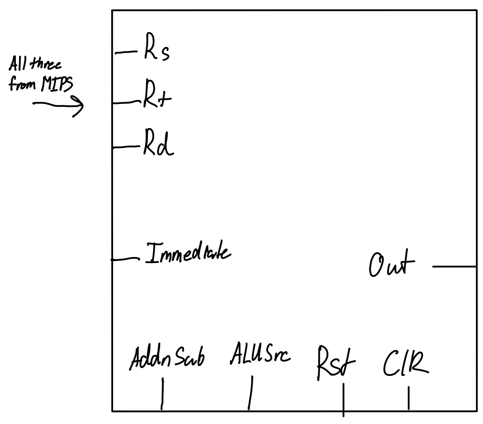
Read output 2 from Res 16

Positive edge place FFF into Res 16

Write 222333 to Res 20

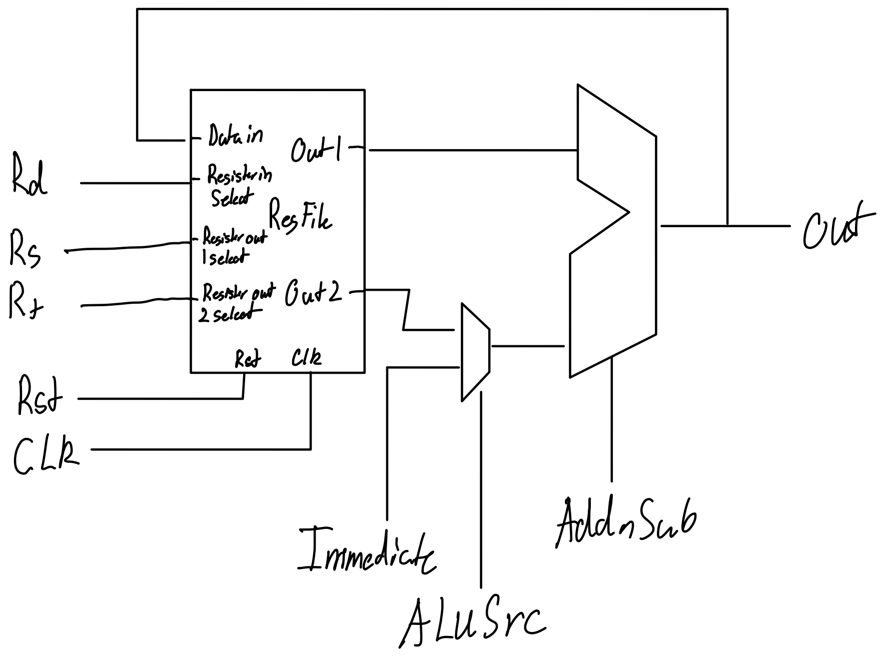
[Part 3 (b)] Draw a symbol for this MIPS-like datapath.

1st Data Path

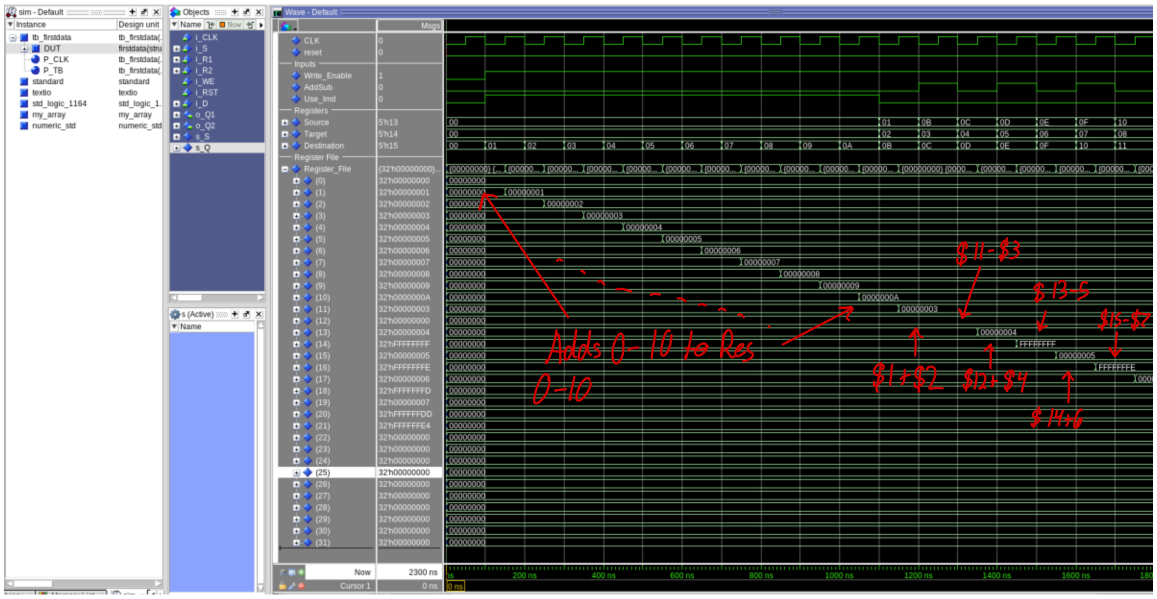


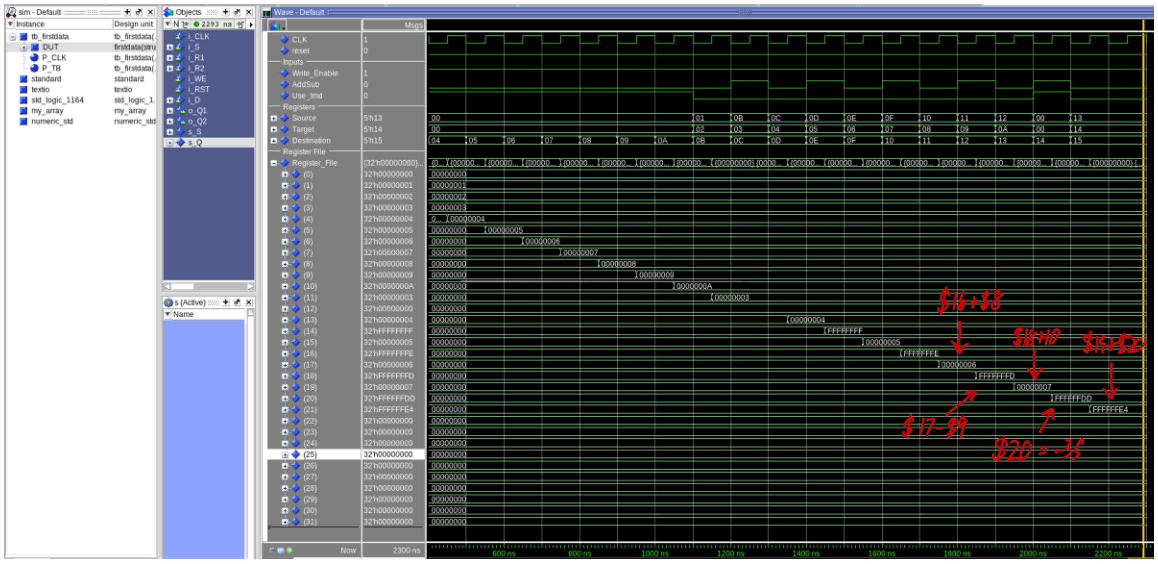
[Part 3 (c)] Draw a schematic of the simplified MIPS processor datapath consisting only of the component described in part (a) and the register file from problem (1).

ALu + Res File



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





Reg	Result	Reg	Result	Reg	Result	Reg	Result
0	0	8	8	16	-2	24	0
1	1	9	9	17	6	25	0
2	2	10	10	18	-3	26	0
3	3	11	3	19	7	27	0
4	4	12	0	20	-35	28	0
5	5	13	4	21	-32	29	0
6	6	14	-1	22	0	30	0
7	7	15	5	23	0	31	0

[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).

The clock port is the input for the clock that runs the internals of the memory system and provides and output and writes to itself on the positive clock edge.

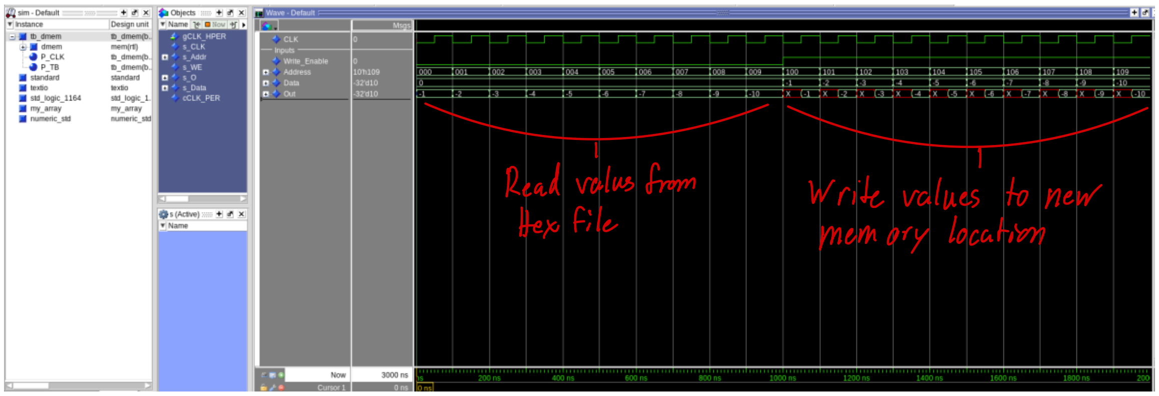
The addr port is the address port of the memory module. This is used to declare where in memory you are either retrieving a value or setting a new value.

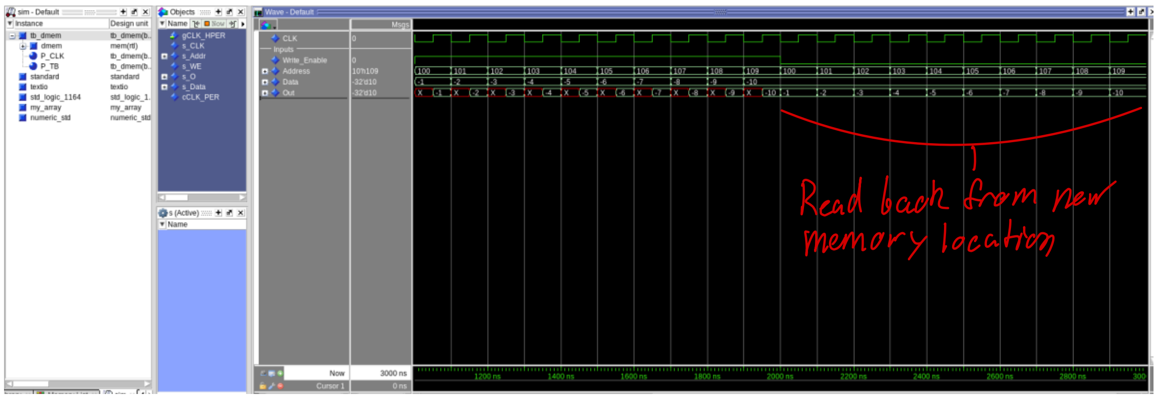
The data port is the input port to the memory module. This is where values are entered to be saved into memory based on the given address

The WE port is the write enable for the memory module. This is there so that you are able to read from the memory module without overwriting what you already have saved in it.

The Q port is the data output of the memory module. This port is responsible for displaying the value stored in the memory module at the given adderess

[Part 4 (c)] Waveforms.





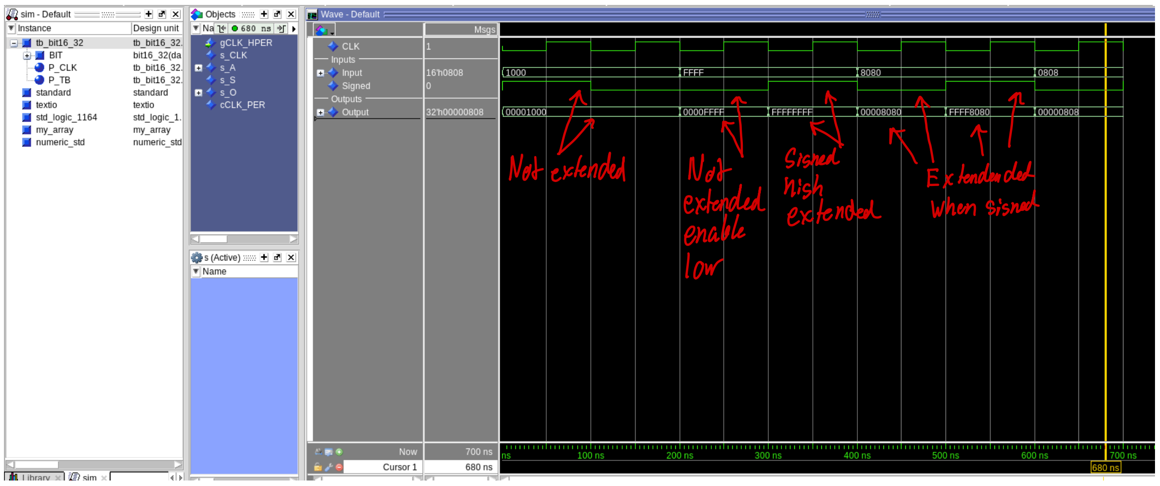
[Part 5 (a)] What are the MIPS instructions that require some value to be sign extended? What are the MIPS instructions that require some value to be zero extended?

Any MIPS function that uses the immediate in basic arithmetic operations requires it to be sign extended. Some commands that would you this would be addi and addiu along with the basic load and store commands
The MIPS commands that primarily utilize the zero extension would be the bitwise immediate operations such as andi and 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 would need one of these extenders in place to take our 16 bit immediates and translate the into 32 bits to be used in the ALU. We would also need one of these when we are using offsets when accessing memory to take our 16 bit immediate offset and change it to 32 bits for the addressing.

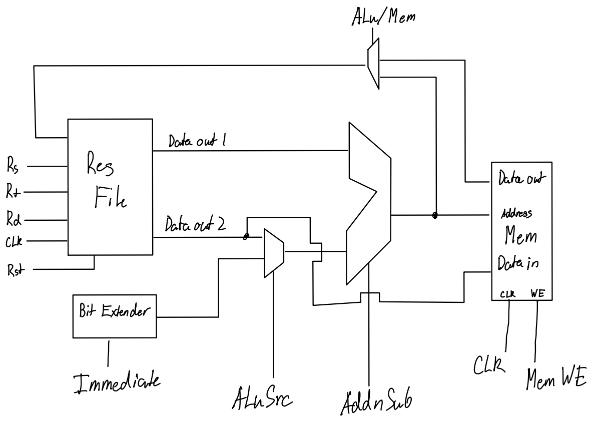
[Part 5 (d)] Waveform.



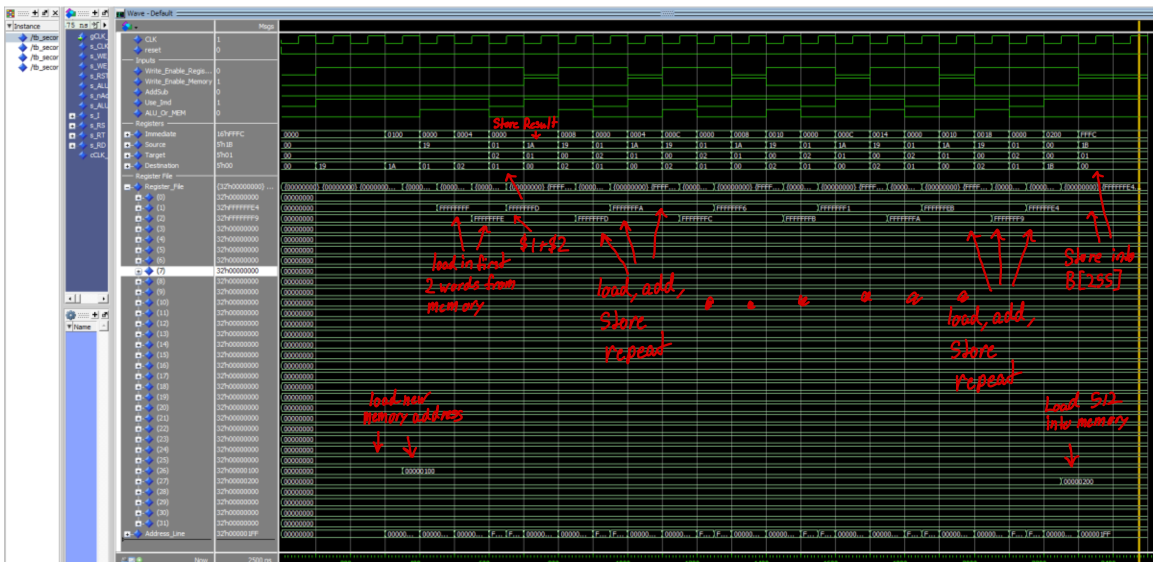
[Part 6 (a)] what control signals will need to be added to the simple processor from part 2? How do these control signals correspond to the ports on the mem.vhd component analyzed in part 3?

There are a couple of extra control signals that we need to add to this second data path. We need to add a write enable for the memory along with a select line to choose whether we are writing the output of the ALU or the output of the memory module back to the register file.

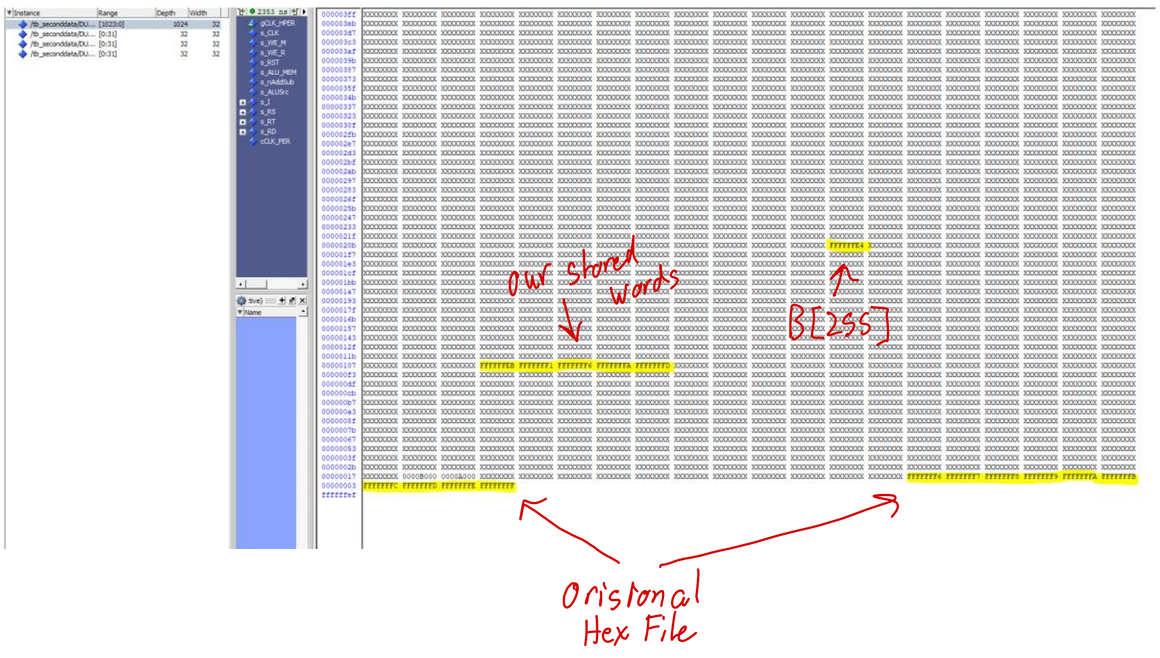
[Part 6 (b)] Draw a schematic of a simplified MIPS processor consisting only of the base components used in part 2, the extender component described in part 4, and the data memory from part 3.



[Part 6 (c)] Waveform.



Wave - Default
CLK
reset
Inputs
Write_Enable_Regis...
Write_Enable_Memory
AddSub
Use_Imd
ALU_Or_MEM
Registers
Immediate
Source
Target
Destination
Register File
Register_File
Address_Line
Store Result
$1+$2
load in first 2 words from memory
load, add, store repeat
load, add, store repeat
Store into B[255]
load new memory address
Load 512 into memory



Our stored words
B[255]
Orisional Hex File