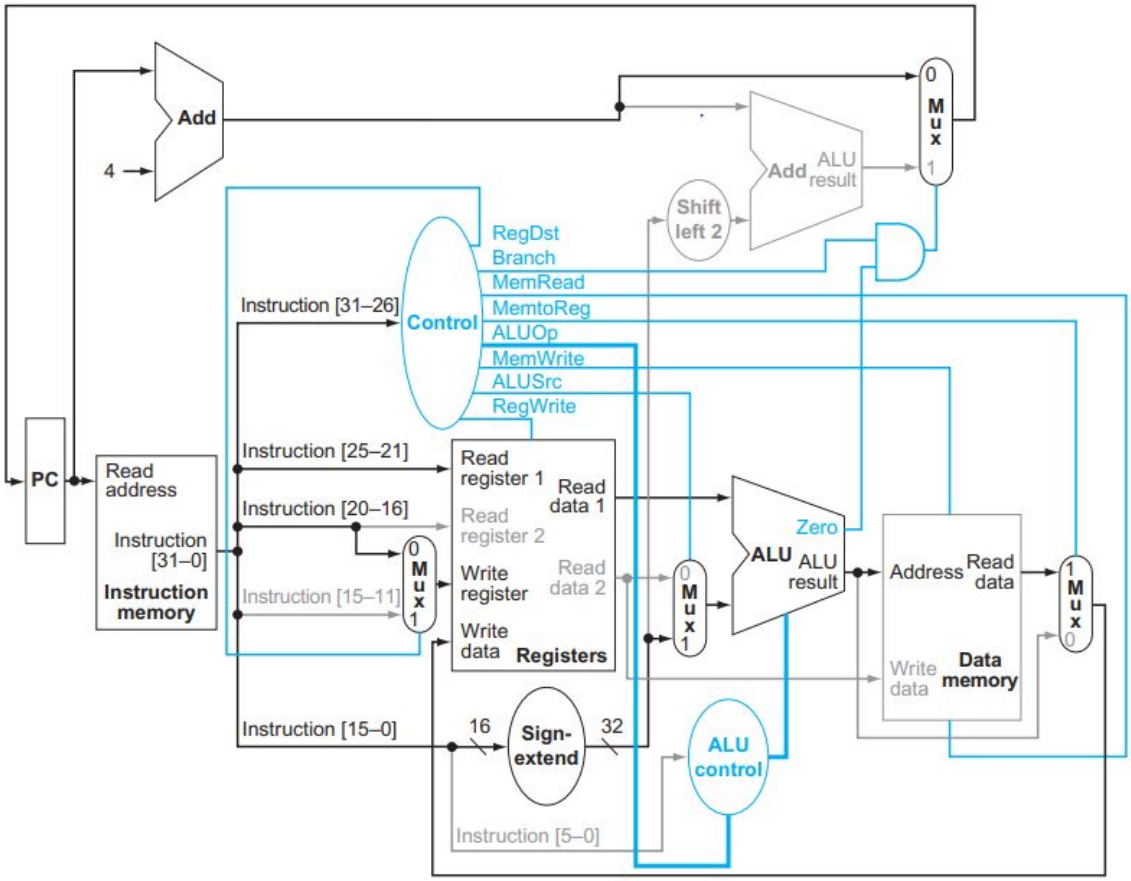
CprE 381 – Computer Organization and Assembly-Level Programming

Proj-B Report

Lab Partners Jacob Vaughtn
Nickolas Mitchell

Section / Lab Time 1/4:10-6:00

Make This



Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the Proj-B instructions for the context of the following questions.

a. [Part 0] How are instruction and data memory initialized in the simulation? How does MARS interface with ModelSim?

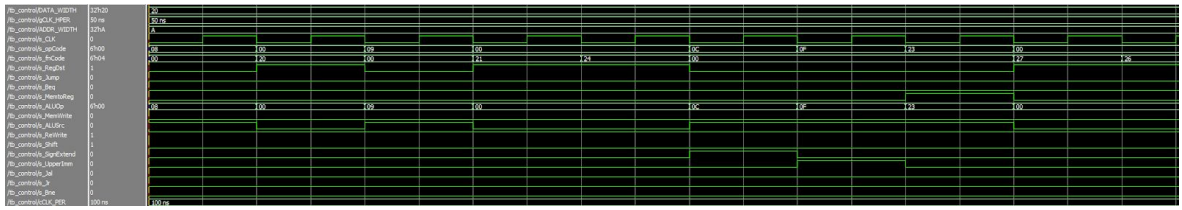
The data memory is being initialized inside of modelsim while the instructions are being initialized inside of MARS. A python script runs the code inside of MARS and modelsim and compares the results to see if the processor is working as expected.

b. [Part 0] In the MIPS skeleton VHDL, how is a halting / termination condition detected? What MIPS assembly instruction does this correspond to?

In modelsim, to halt a program check if s_halt == '1' and V0 = 10 (decimal). In MIPS this corresponds to syscall.

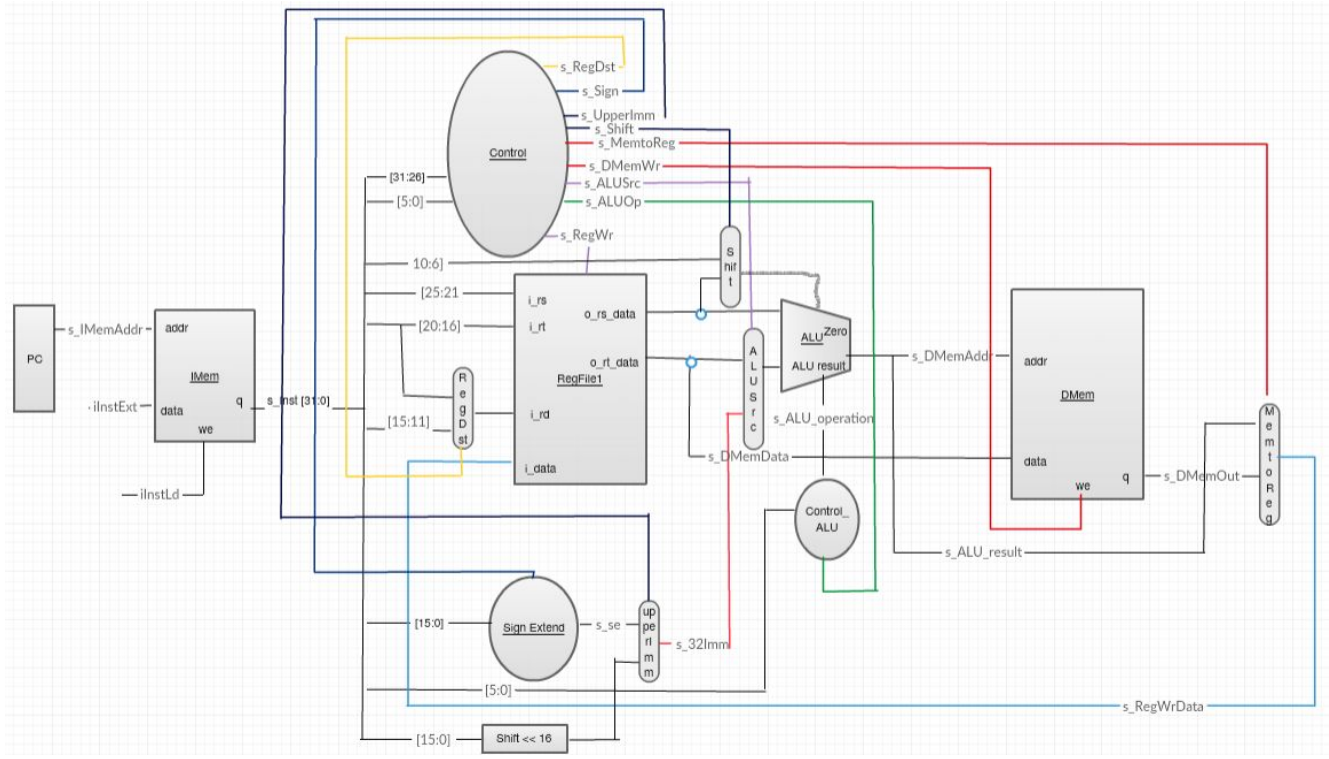
c. [Part 1 (a)] Modify the provided spreadsheet to include the list of *M* instructions to be supported in this phase alongside their binary Instruction OPcodes and Funct fields, if applicable. Create a separate column for each binary bit. Inside this spreadsheet, create a new column for the *N* control signals needed in your single-cycle processor implementation. The end result should be an *M*N* table where each row corresponds to the output of the control logic module for a given instruction. *[You can attach the spreadsheet as a separate file, with a single spreadsheet for all Phase I and Phase II instructions.]*

d. [Part 1 (b)] Implement the control logic module using whatever method and coding style you prefer. Create a testbench to test this module individually, and show that your output matches the expected control signals from part 1a). *[Please include waveforms and explanations.]*



Included waveform (Proj-B_1(b).wlf)

e. [Part 2] In your writeup, provide your schematic for this part, describe what challenges (if any) you faced in implementing this module.

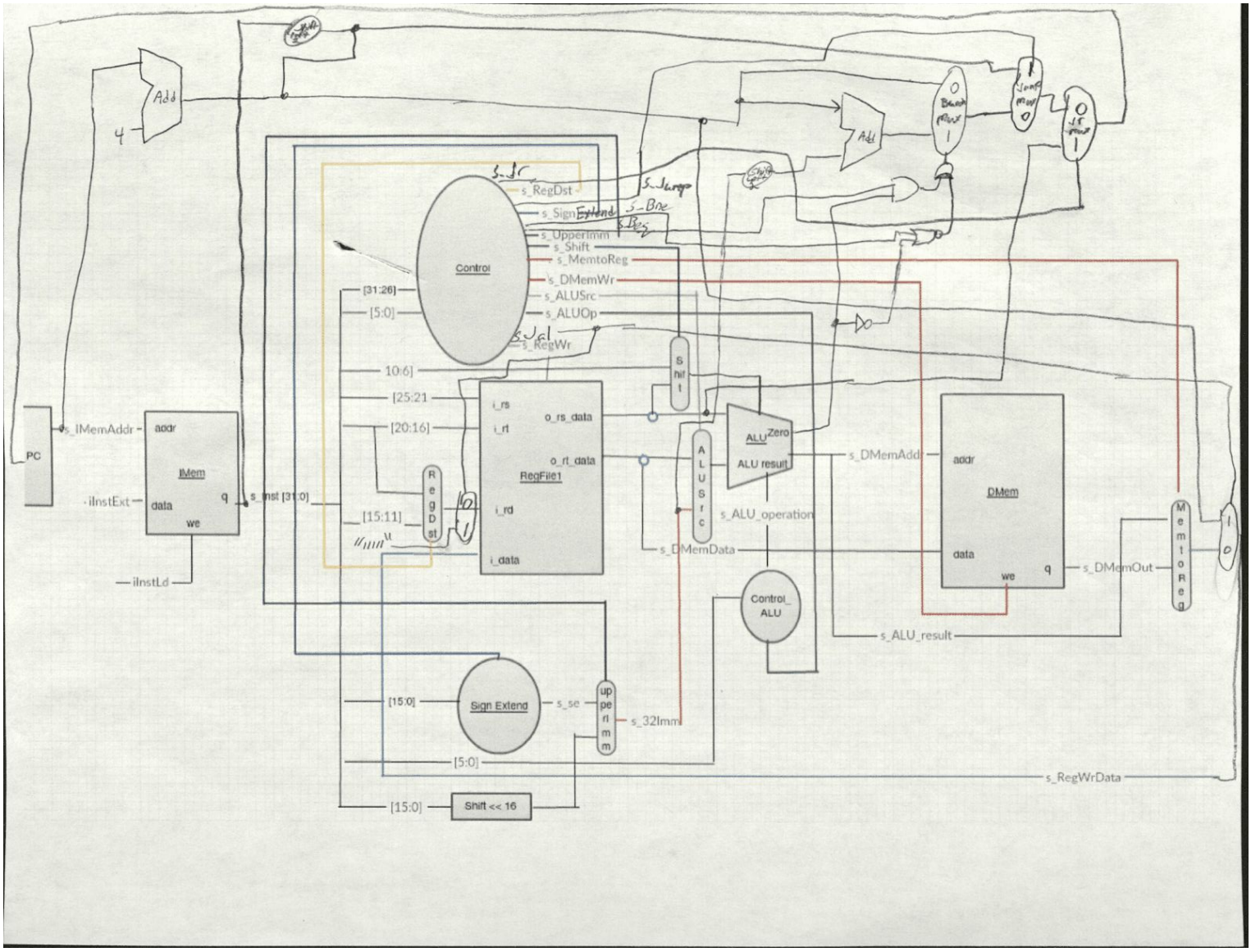


We ran into a lot of challenges finding a drawing tool that would save my work. I rebuilt this about 5 times from different points and don't have access to this schematic anymore. That being said there should be a PC + 4 leading into PC but it would take too much time to do that. I also renamed s_Sign to s_SignExtend

f. [Part 4 (a)] Describe these possibilities as a function of the different control flow-related instructions.

It must support beq, bne, Jump, jal, and jr.
beq and bne need to set the PC to the current pc +- the imm value
Jump and jal need to set the PC to what is in the Imm field for j type instruction.
jr needs to set the PC to the register files rs output.

g. [Part 4 (c)] Update your processor schematic for the instruction fetch logic and any other datapath modifications needed for control flow instructions. In your writeup, describe what additional control signals are needed.

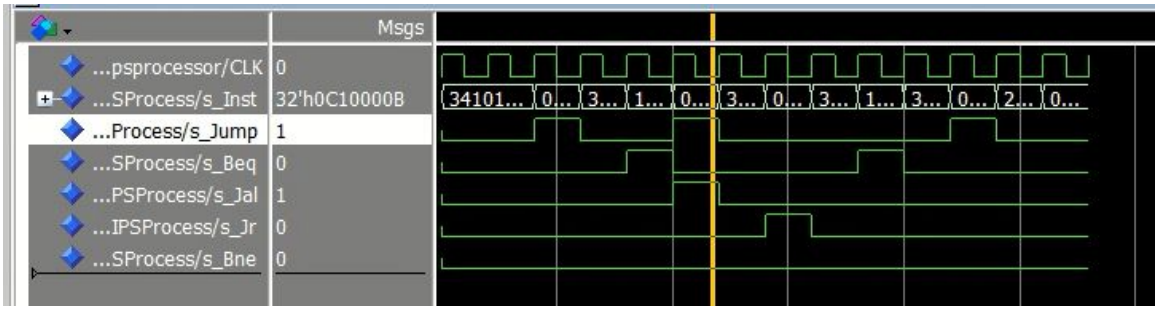


(Due to loss of my original schematic and a desperate time crunch this was done by hand on paper sorry for the mess it's the best I can do with what I have.)

Additional control signals we will need are going to be signals for beq, bne, jal, jump, and jr.

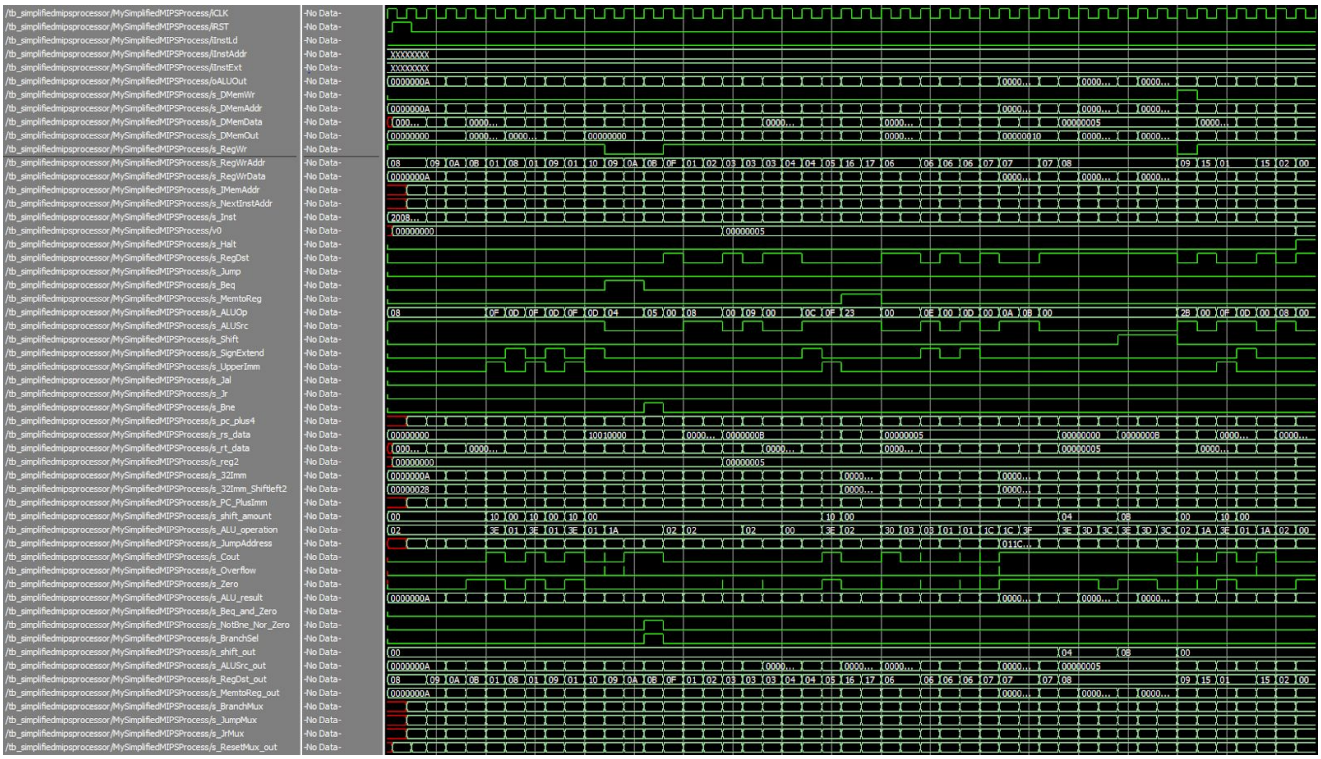
beq - is one when we want to beq and is anded with alu output s_Zero
bne - is one when we want to bne and is ored with alu output Not(s_Zero)
Jump - is one when we are setting PC to jump to the Imm field
jal - is one when we need to Jump and set reg 31 to PC + 4
jr - is one when we are doing a jr instruction and need to set PC = reg 31

h. [Part 4 (d)] Include ModelSim waveforms, and describe how the execution of the control flow possibilities corresponds to the waveforms in your writeup.



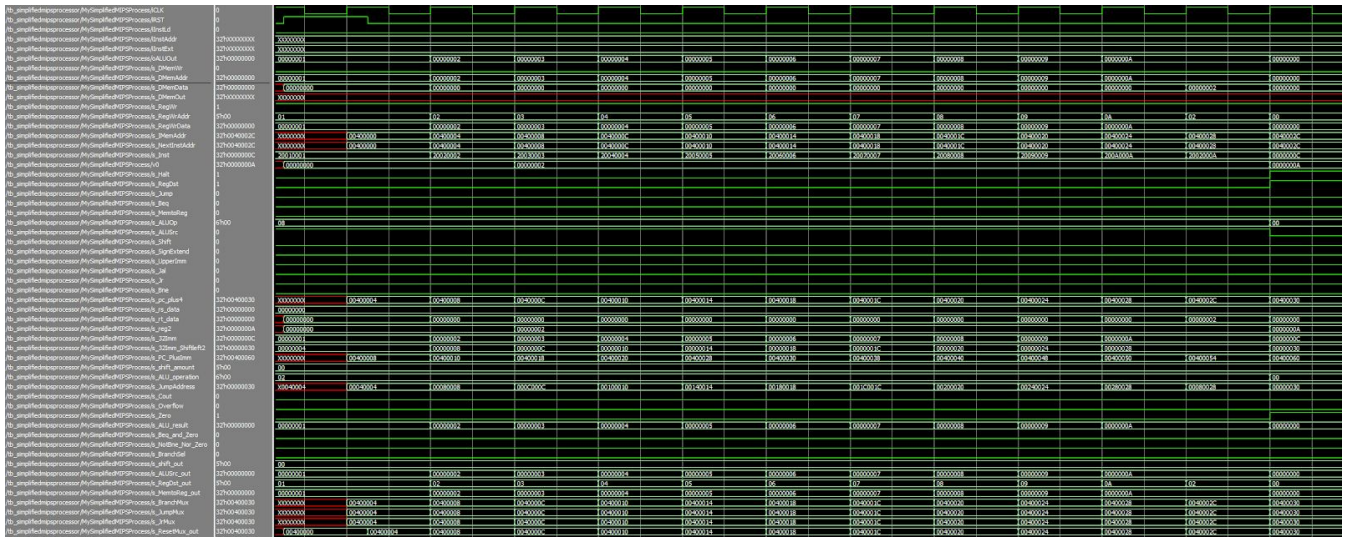
This waveform shows the output from the control.
When the op code is = 0x02 then s_Jump is set to 1. Zero if not j or jal
when the op code is = 0x04 then s_Beq is set to 1. Zero if otherwise.
when the op code is = 0x05 then s_Bne is set to 1. Zero if otherwise.
when the op code is = 0x03 then s_Jal and s_Jump are set to 1. Zero if otherwise.
when the op code is = 0x00 and the function is 0x03 then s_Jr is set to 1. Zero if otherwise.

i. [Part 5 (a)] In your writeup, show the ModelSim output for the individual instruction tests, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.



First of all, this is no great way to represent this waveform within our report so we have included the waveform file (Proj-B_5(a).wlf) to allow for a more detailed analysis. This waveform includes all of the tests from phase one and phase two. If you are curious to see what our tests consists of, the above waveform was generated using the Proj-B-test2.s file. As far as which tests we decided to write for our processor, we deemed necessary to test every single instruction for phase one at least once and to test our branch and jump instructions from part two a little more thoroughly.

j. [Part 5 (b)] In your writeup, show the ModelSim output for the Bubblesort test, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.

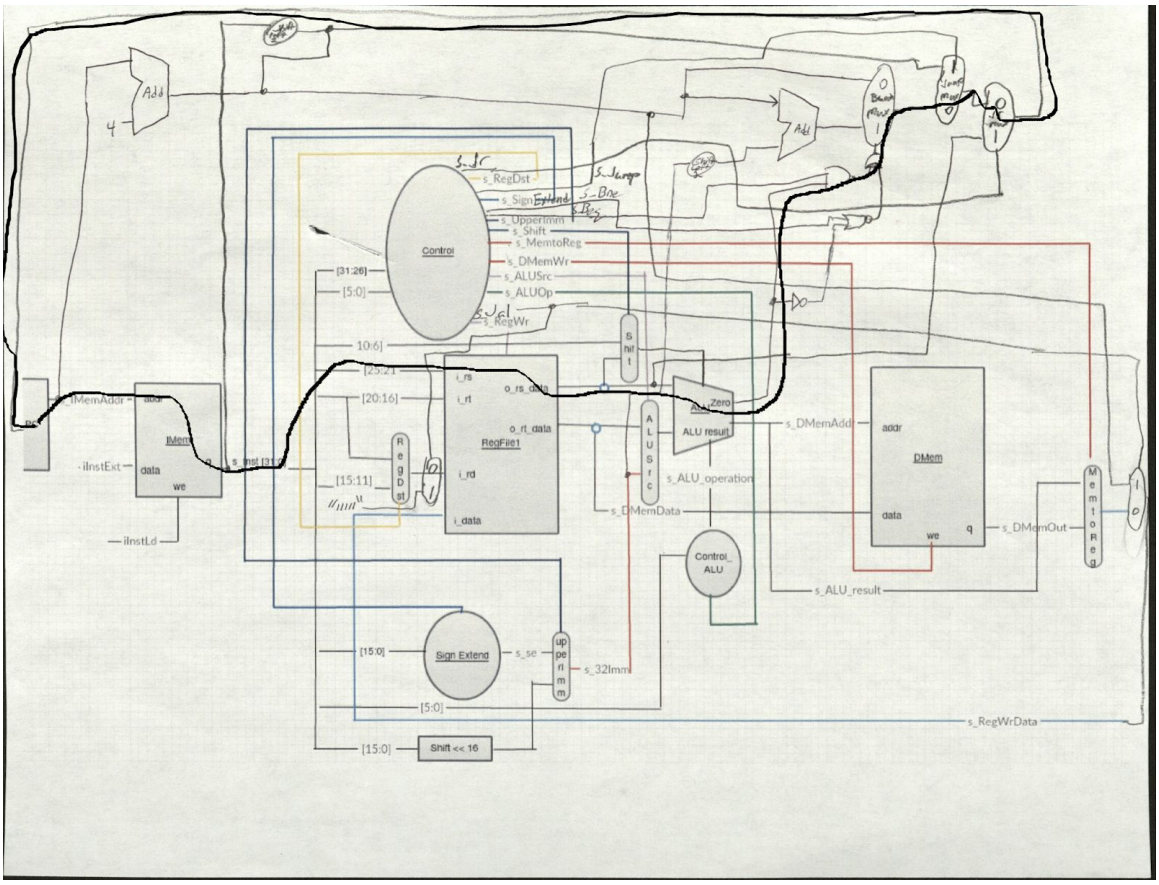


Again, the above waveform is very difficult to represent in the document so we have included the waveform file (Proj-B_5(b).wlf). Our bubble sort algorithm simply sorts a predetermined array therefore we can just print the results of the sort in MARs and check them against $v0, and $a0 within our register file.

k. [Part 5 (c) BONUS] In your writeup, show the ModelSim output for the Mergesort test, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.

l. [Part 7] Report the maximum frequency your processor can run at and determine what your critical path is. Draw this critical path on top of your top-level schematic. In your writeup, briefly discuss your critical path results. What components would you focus on to improve the frequency?

FMax: 27.28mhz



My max datapath goes from PC to IMem to Regfile to the ALU then to my JrMux which is the top farthest right mux.
It doesn't say that it is going through my jump mux or branch mux but I believe it is waiting for the input for the JrMux select = 0 input.
I would focus on the ALU and IMem to improve overall frequency. My ALU takes ~15ns to go through and the IMem takes ~10ns
Other things to consider are the Regfile and the input to the muxes. The regfile take ~5ns and the Jr mux is waiting for input for ~5ns
Overall the max path takes 39.630ns.

m. [Feedback] You must complete this section for your lab to be graded. Please complete each column **separately** for each team member; I expect it to take roughly 10 minutes (do not take more than 20 minutes).

i. How many hours did you spend on this lab?

Task	**During lab time**			**Outside of lab time**		
Team Initials	JV	NM		JV	NM	
Reading lab	.1	0.5		.2	0.5	
Pencil/paper design	5	0		4	0.5	
VHDL design	2.6	1		25.8	7	
Assembly coding	.1	1		2.5	5	

Simulation	.1	4		4	7	
Debugging	.1	1.5		4	3	
Report writing	0	0		1	0.5	
Other:	0	0		0	0	
Total	8	8		41.5	23.5	

ii. If you could change one thing about the lab experience, what would it be? Why?

Give more warning for sltu and sltui.
also more description on what happens for unsigned instructions.

iii. What was the most interesting part of the lab?

Making a processor that uses assembly input to execute instructions. Where all of the components are designed from scratch.