ASSIGNMENT 4

LONG NGUYEN

EEL4713

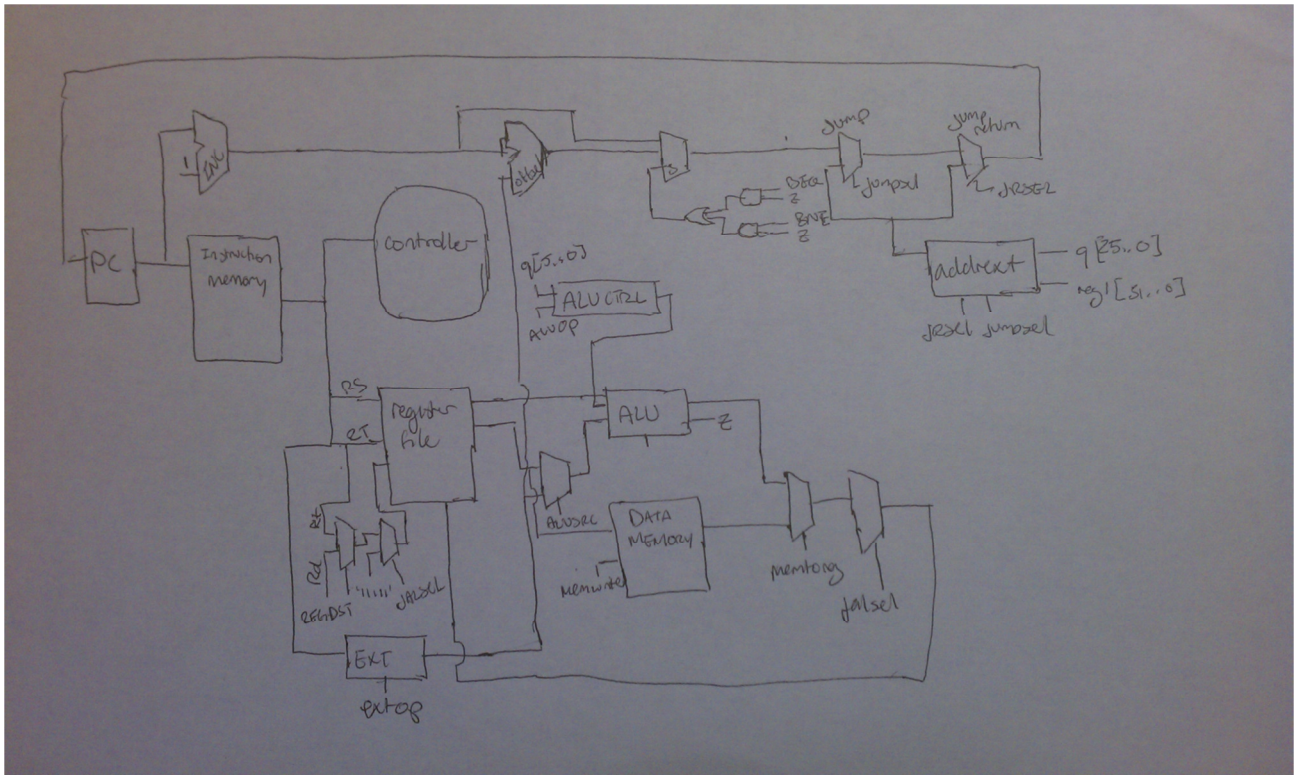
3/17/2012

Introduction

In this lab I worked on designing a single cycle MIPS processor which executed the 29 Core Instructions. Parts from the previous lab 2 were combined with additional hardware in order to create the mips_core file in this lab. A major goal of this lab is to make sure the all the instructions work for the next labs.

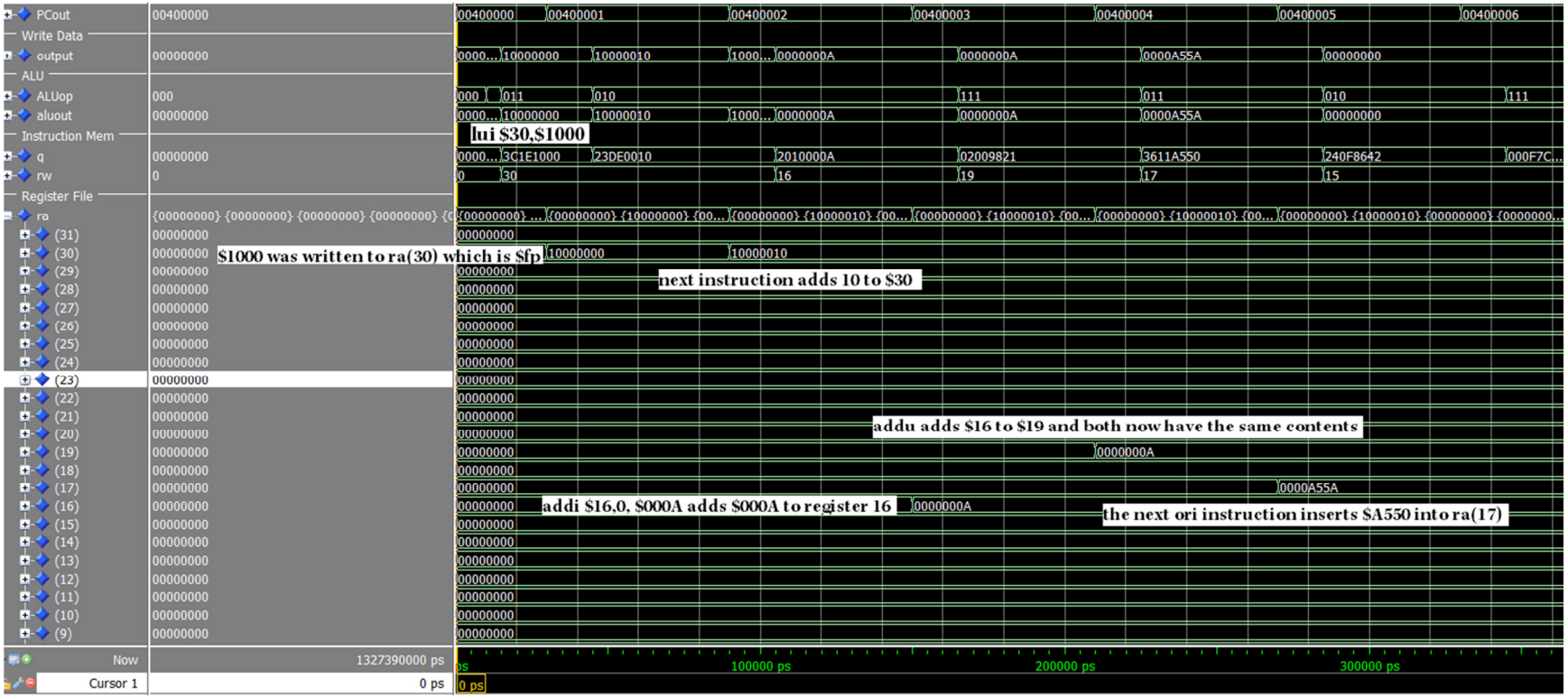
Design and Testing

The following is the single cycle datapath. A more detailed datapath can be found in the bdf files,



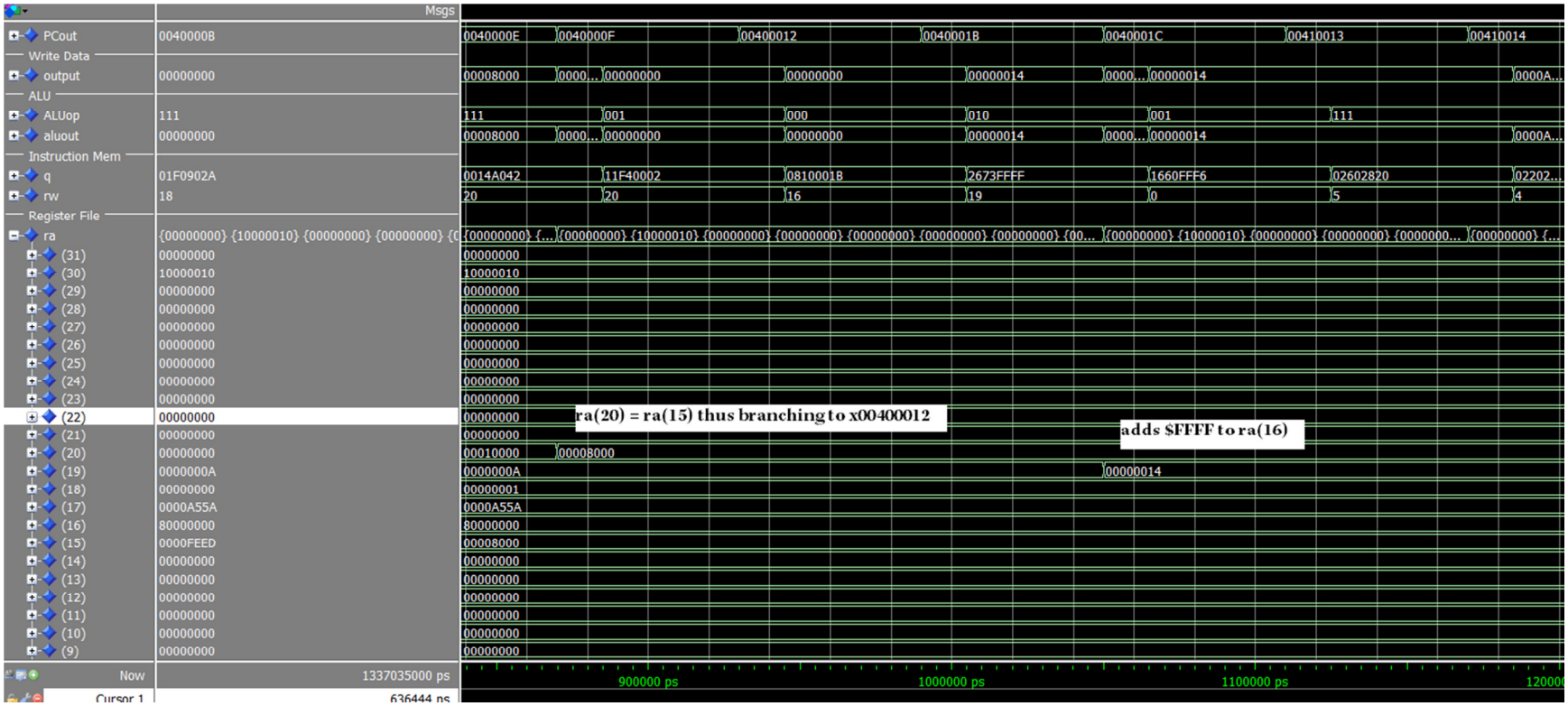
LAB4_TEST

The first instruction test for the processor came from the lab4_test.mif. This test gave me a number of difficulties in when trying to simulate in model sim. As it would turn out, the register file was not functioning properly and became the source of a very long headache since beginning in the week.



Once the datapath was working correctly again, this simulation was successfully completed.


PCin
0040000C
PCout
0040000B
Write Data
output
00000000
ALU
ALUop
111
aluout
00000000
Instruction Mem
q
01F0902A
rw
18
Register File
ra
(31)
(30)
(29)
(28)
(27)
(26)
(25)
(24)
(23)
(22)
(21)
(20)
(19)
(18)
(17)
(16)
(15)
(14)
(13)
(12)
(11)
00400008
00400009
0040000A
00400008
0040000C
0040000D
00400007
00400008
00400009
0040000A
0040000B
0040000C
0000FEED
00008000
80000000
00000001
00008000
011
000
111
35EFFEED
31F08000
00108400
01F0902A
01F0902B
00127BC0
15
16
18
10000010
0000000A
0000A55A
00008000
80000000
0000FEED
ors ra(15) with itself to
insert xFEED
the next instruction ands xFEED with x8000
and shifted right by 4
bits are then shifted left by 4
the first two shifts are done signed while the last is unsigned.
Now
1337035000 ps
Cursor 1
636444 ps
400000 ps
500000 ps
600000 ps
700000 ps

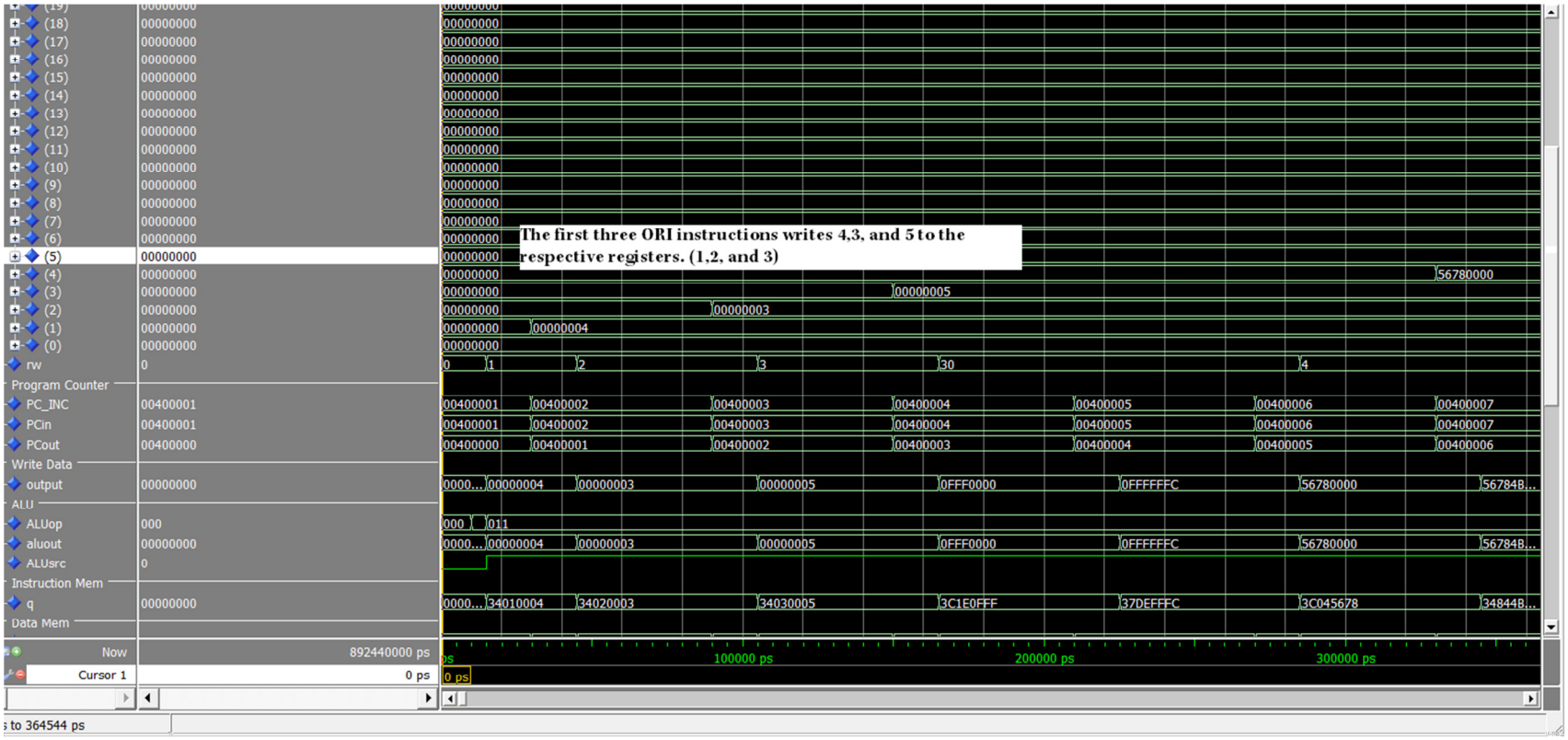


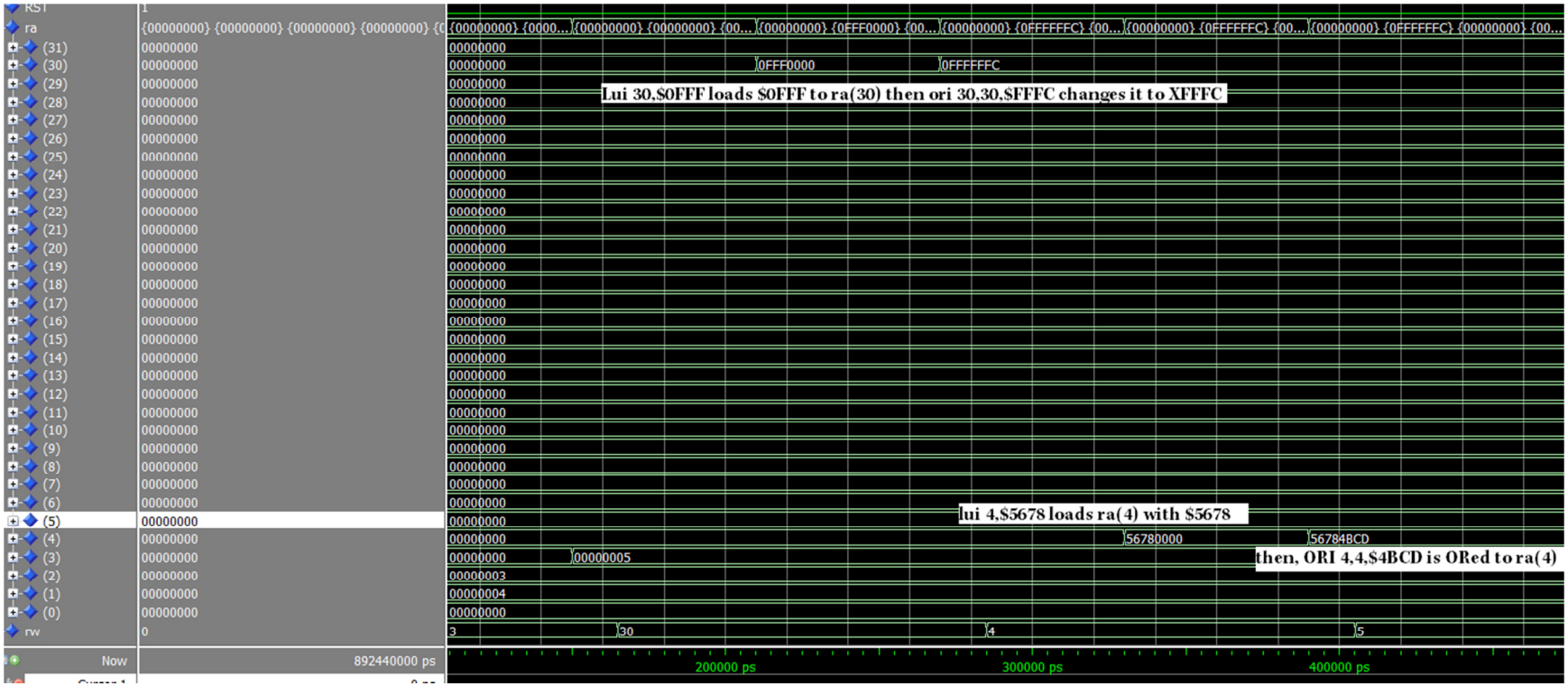
Unfortunately it seems the branch functions did not properly branch as expected. For instruction x11F0002, the signals should've branched back to x00400010. In this figure the branch jumped too far ahead. Due to time constraints, a throughout troubleshooting was not performed. However, there could be multiple reasons for this failure. Firstly, I would check the controller to make sure the BEQ and BNE opcodes were correctly initialized. If not that, I would check connections in the bdf and begin checking the hardware.

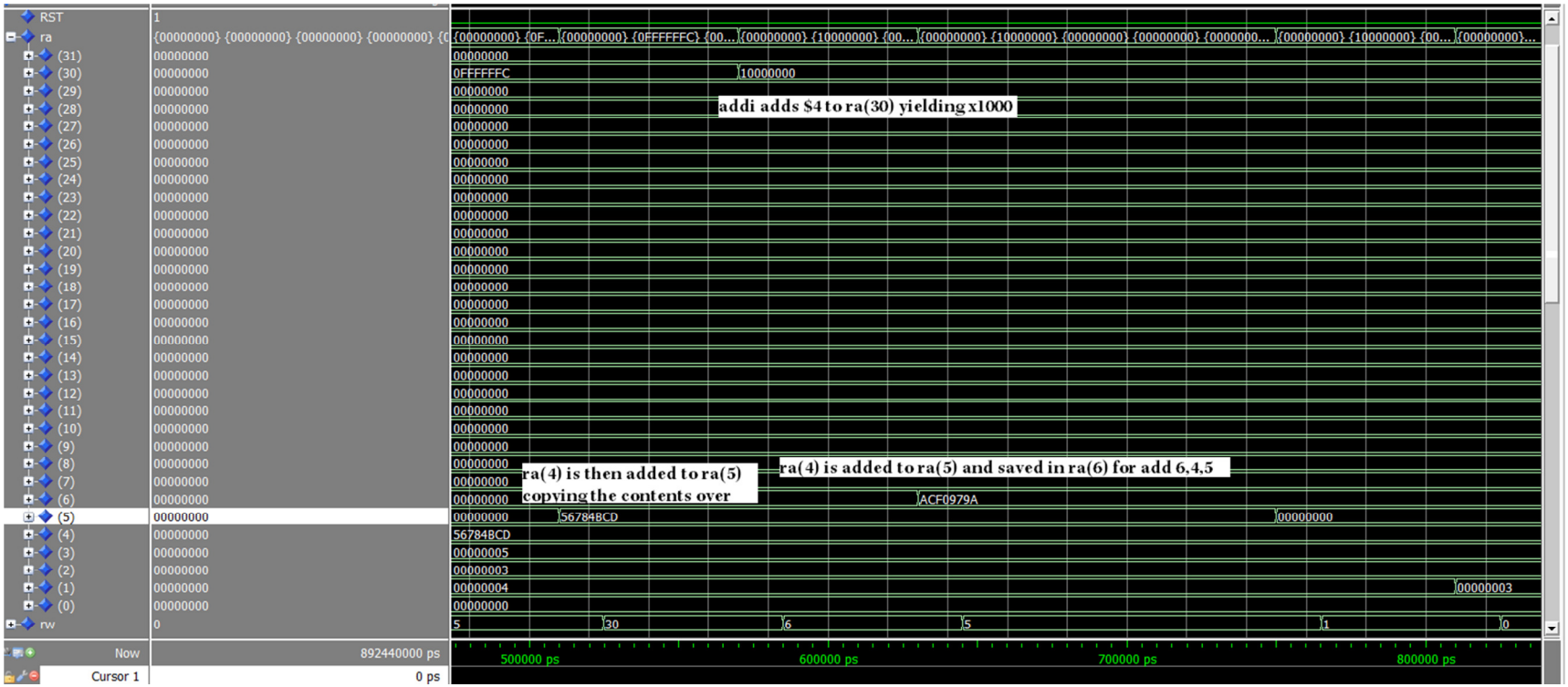
What this instruction should've continued to do would be to add further numbers into ra(16) and ra(17). Then the next instruction would be a jump and if correctly initialized the jump would take the PC to 0040001B. There is a BNE after the jump to branch back to 00400013. The instructions would continue to test the jump register function and move on to load word. The load word function should be able to read from the data memory into the register. Continuing, the program ends with a BNE which depending on the value would loop through the instructions more.

LAB4_DEMO

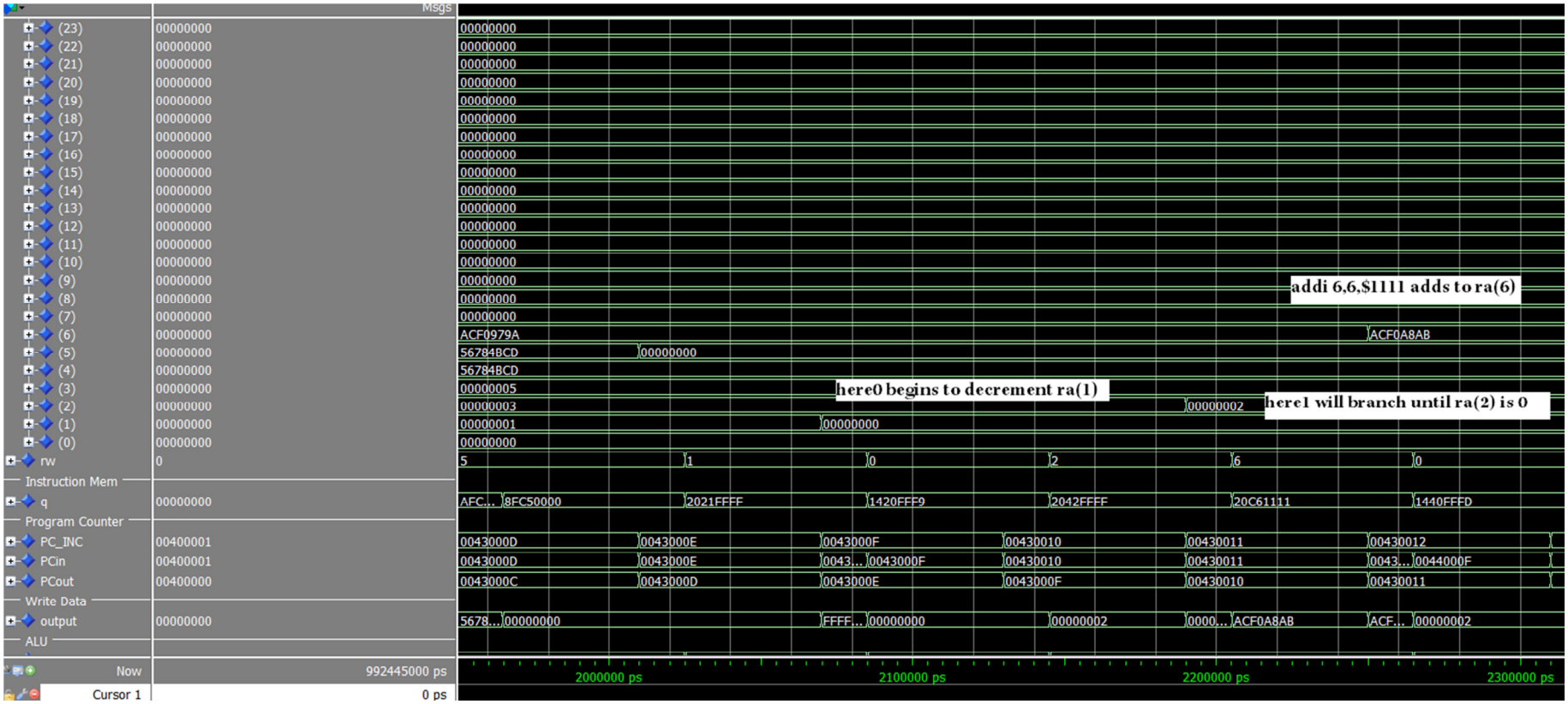
The lab4_demo file tests many of the instructions further. As before, the branch functions were not able to be fixed just yet. The simulation up to that section is as follows:

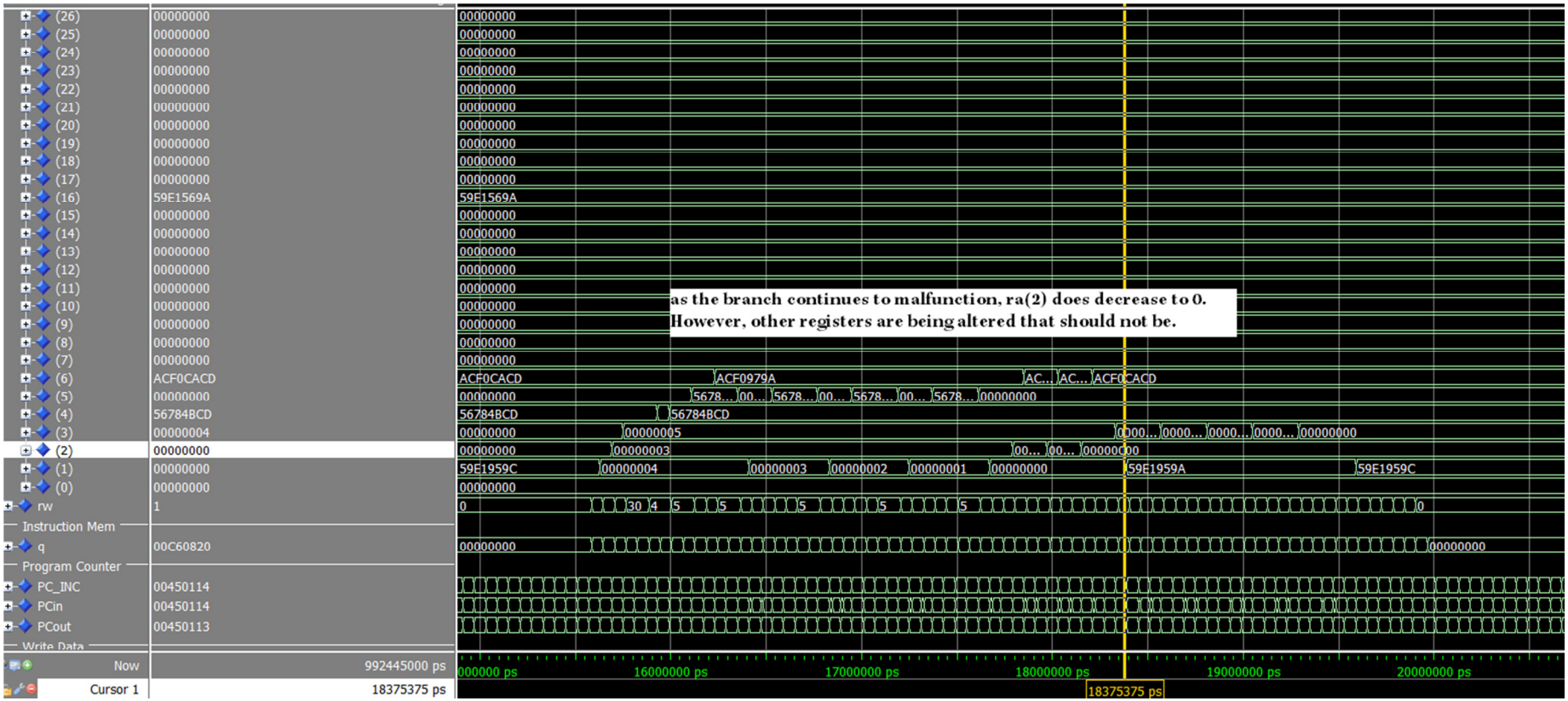



RST
ra
(31)
(30)
(29)
(28)
(27)
(26)
(25)
(24)
(23)
(22)
(21)
(20)
(19)
(18)
(17)
(16)
(15)
(14)
(13)
(12)
(11)
(10)
(9)
(8)
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
rw
Now
892440000 ps
0FFF0000
0FFFFFFC
Lui 30,$0FFF loads $0FFF to ra(30) then ori 30,30,$FFFC changes it to XFFFC
lui 4,$5678 loads ra(4) with $5678
56780000
56784BCD
then, ORI 4,4,$4BCD is ORed to ra(4)
00000005
00000003
00000004
200000 ps
300000 ps
400000 ps



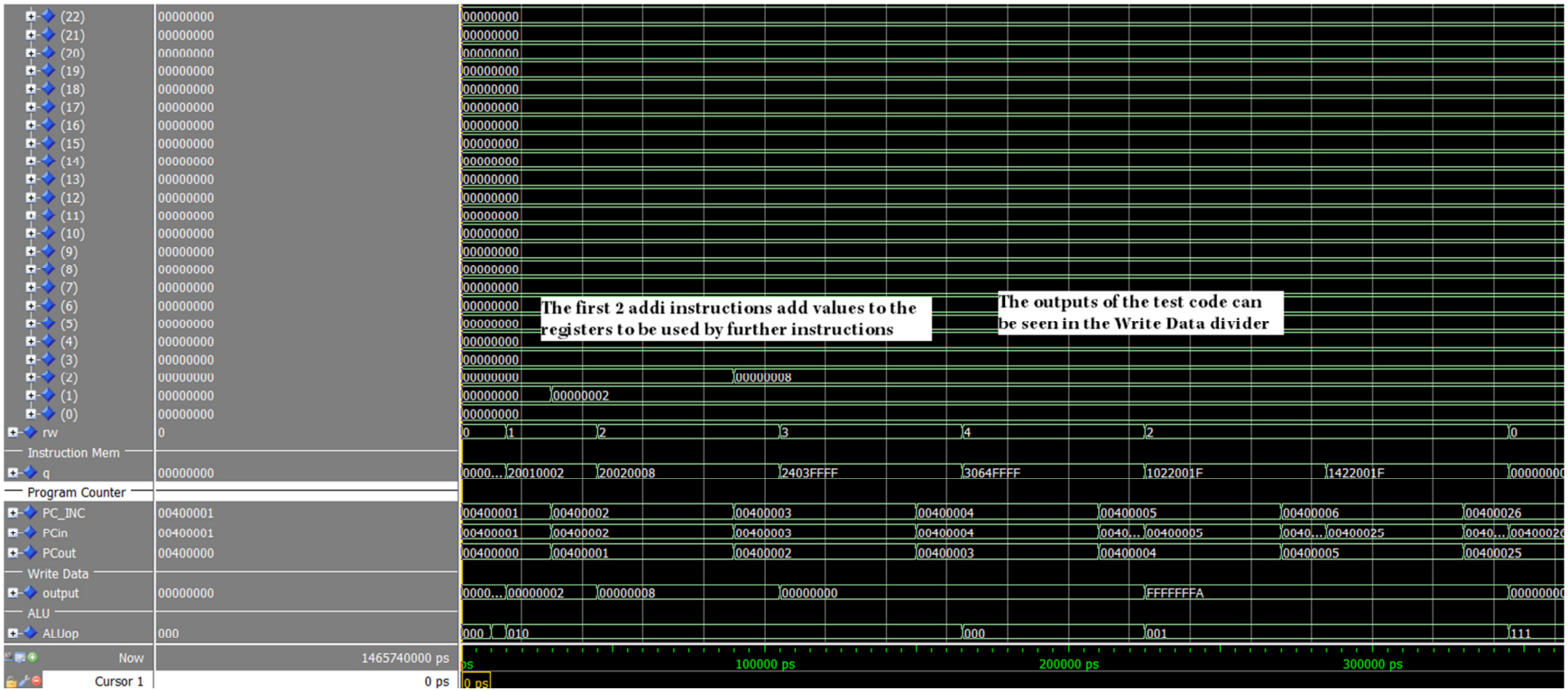
Branch here0 begins after this instruction . The loop should decrement ra(1) until it is 0. Here1 does the same for ra(2). Here2 should also decrement ra(3), however, the malfunction begins to throw the instructions off.


Msgs
(23) 00000000
(22) 00000000
(21) 00000000
(20) 00000000
(19) 00000000
(18) 00000000
(17) 00000000
(16) 00000000
(15) 00000000
(14) 00000000
(13) 00000000
(12) 00000000
(11) 00000000
(10) 00000000
(9) 00000000
(8) 00000000
(7) 00000000
(6) 00000000
(5) 00000000
(4) 00000000
(3) 00000000
(2) 00000000
(1) 00000000
(0) 00000000
rw 0
Instruction Mem
q 00000000
Program Counter
PC_INC 00400001
PCin 00400001
PCout 00400000
Write Data
output 00000000
ALU
Now 992445000 ps
Cursor 1 0 ps
addi 6,6,$1111 adds to ra(6)
here0 begins to decrement ra(1)
here1 will branch until ra(2) is 0
2000000 ps
2100000 ps
2200000 ps
2300000 ps


as the branch continues to malfunction, ra(2) does decrease to 0.
However, other registers are being altered that should not be.
(26) 00000000
(25) 00000000
(24) 00000000
(23) 00000000
(22) 00000000
(21) 00000000
(20) 00000000
(19) 00000000
(18) 00000000
(17) 00000000
(16) 59E1569A
(15) 00000000
(14) 00000000
(13) 00000000
(12) 00000000
(11) 00000000
(10) 00000000
(9) 00000000
(8) 00000000
(7) 00000000
(6) ACF0CACD
(5) 00000000
(4) 56784BCD
(3) 00000004
(2) 00000000
(1) 00000000
(0) 00000000
rw 1
Instruction Mem
q 00C60820
Program Counter
PC_INC 00450114
PCin 00450114
PCout 00450113
Write Data
Now 992445000 ps
Cursor 1 18375375 ps
000000 ps
16000000 ps
17000000 ps
18000000 ps
19000000 ps
20000000 ps
18375375 ps

LAB4_INSTTEST

This test is supposed to demonstrate the functionality of all 29 instructions. However, due to time constraints, I was only able to do some, and hope to troubleshoot the rest after this lab is turned in.



The current functionality of the datapath is limited due to the failure to implement several instructions. Work will continue on adding the necessary hardware to perform the last instructions needed before the next assignment.

Timing Simulation Speed

Timing analysis of the processor was performed in Quartus. For some reason, the aluctrl was clocked as well.

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	31.07 MHz	31.07 MHz	clk	
2	31.66 MHz	31.66 MHz	mclk	
3	64.61 MHz	64.61 MHz	alu32control:aluctrl|control[0]	
4	334.45 MHz	225.43 MHz	InstructionMem:instmem|altsyncram:altsy...t|altsyncram_b971:auto_generated|q_a[0]	limit due to hold check
5	417.71 MHz	120.34 MHz	InstructionMem:instmem|altsyncram:altsy...|altsyncram_b971:auto_generated|q_a[26]	limit due to hold check

Conclusion

Without a properly functioning datapath, I am unable to satisfactorily conclude this assignment. The datapath is however very close to being finished and troubleshooting should not take too long, though it is longer than I have to turn this paper in. All VHDL and BDF files are included in the submission.