Nicolas Turano

ECEC355

Term Project report

**Part A:**

The instructions that were loaded into instruction memory are shown here:

* 0x8D100000
* 0x8d110004
* 0x12110002
* 0x02959820
* 0x08000006
* 0x02959822
* 0xAD130008

From what I can understand in the screenshots below, almost everything is working properly except for the ALU. Since the ALU is not calculating a proper value, the correct memory addresses were not calculated for the lw instruction, however, the data at memory address 0 was loaded into the registers $s0, and $s1, meaning that the right control signals were sent, and the registers, instruction memory, and data memory can all be read properly, and the registers can also be written to properly. If the processor was working correctly it would finish by writing the value 19 to address 8 in memory.

**Part B**:

This section had similar results to the first, but if it worked correctly it would’ve written 11 to memory address 8.

**Part C:**

For part C, the results are also unclear due to the Issue with the ALU, but the screenshots show that the correct instructions are running and data is being written to the registers even if it is the incorrect data.

These are the instructions both in assembly and machine language that were loaded into instruction memory for part C:

* Addi $s0, $zero, 10 - 0x2010000A
* Addi $s1, $zero, 20 – 0x20110014
* Addi, $s2, $zero, 0 – 0x20120000
* Addi $s3, $zero, 2 – 0x20130002
* Addi $s4, $zero, 0 - 0x20140000
* Addi $s5, $zero, 48 – 0x20150030
* Beq $s1, $s3, 10 – 0x1233000A
* Add $t0, s4, s2 – 0x02924020
* Lw $s6, 0(t0) - 0x8D160000
* Add $s7, $s6, $s0 – 0x02D0B820
* Add $s7, $s7, $s1 – 02F1B820
* Add $t1, s5, s2 – 0x02B24820
* Sw $s7, 0(t1) – 0xAD370000
* Addi $s0, $s0, -1 – 0x2210FFFF
* Addi $s1, $s1, -3 – 0x2231FFFD
* Addi $s2, $s2, 1 – 0x22520001
* J 6 – 0x08000006

The final values saved in the registers should be:

$s0 – 4

$s1 – 2

$s2 – 6

$s3 – 2

$s4 – 0

$s5 – 48

$s6 – 6

$s7 – 16

Array A in the data memory remains unchanged, Array B stores:

Address 48 – 31

Address 52 – 28

Address 56 – 25

Address 60 – 22

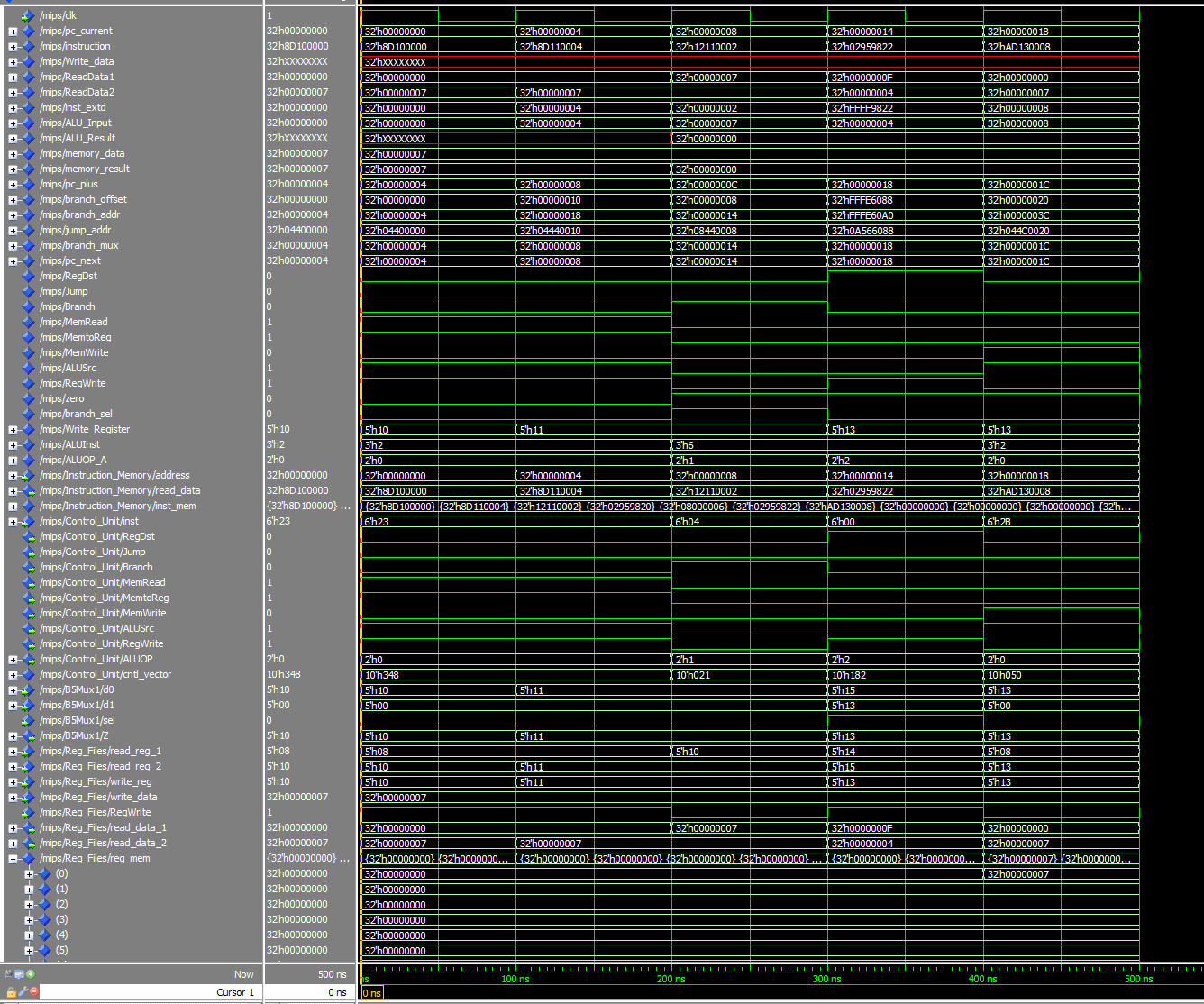
Address 64 – 19

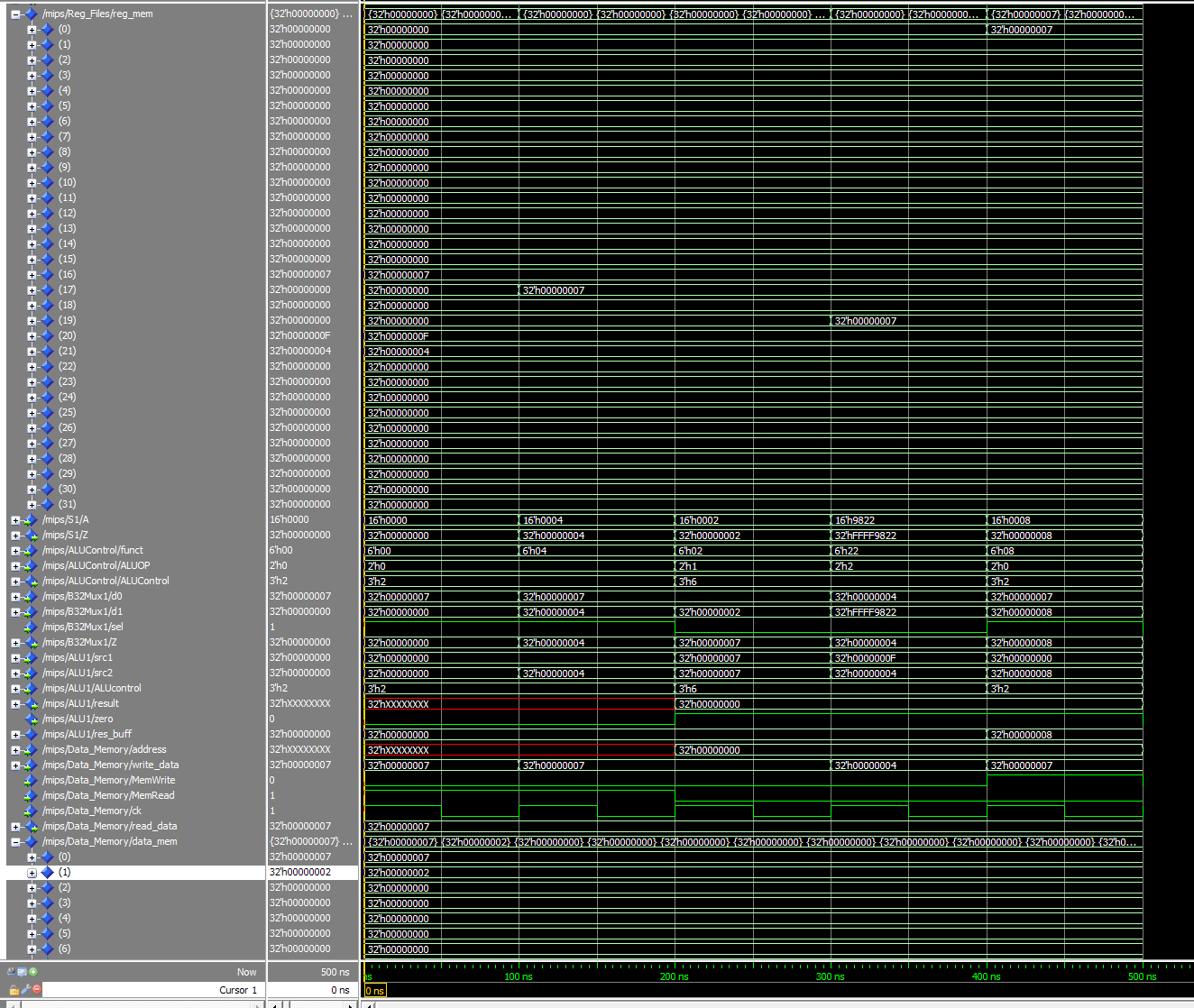
Address 68 – 16

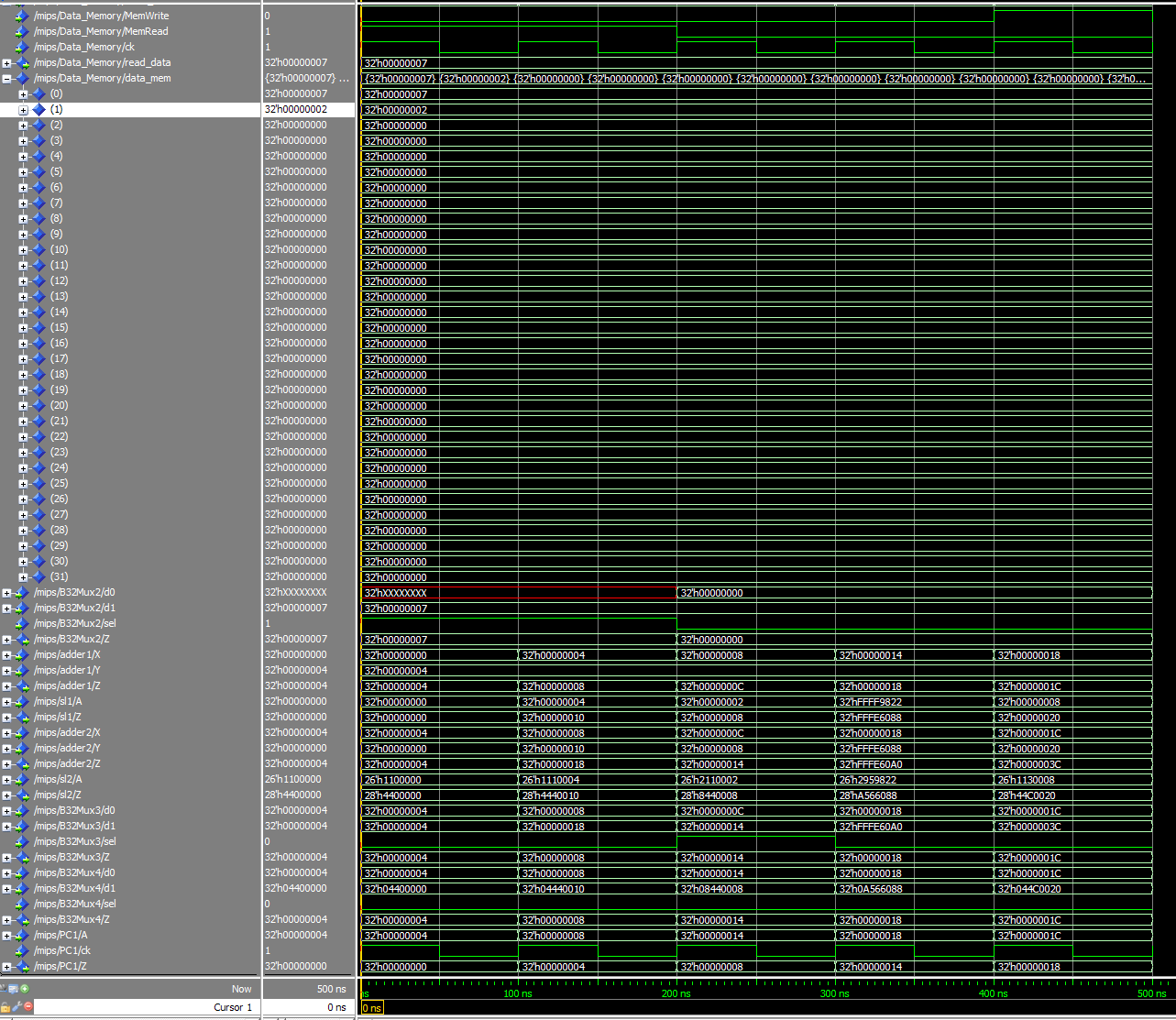
The code should run for 66 cycles before completing.

Note: I understand that the ALU is an integral part of the CPU and it is hard to see most results without it. But I believe that it is a small issue and once it is fixed the processor should be working as intended. I did my best to show which instructions did work without the ALU, and I completed the report questions as if it did work.

**Part A/B screenshots:**







**Part C Screenshots:**

