UNIVERSITY OF CALIFORNIA IRVINE COURSE: ORGANIZATION OF DIGITAL COMPUTERS
LABORATORY (112L)
WINTER 2016

Lab 3 Report - Jump and Branch Instruction

Prepared by: Team The Powerful Processors
Jon Raphael Apostol - 32302252
Binh Nguyen - 34707912
Yixiang Yan - 16392389
James Yi - 17492099

February 14, 2016

1 INTRODUCTION

In this lab, we implemented the jump and branch MIPS instructions into our single-cycle MIPS processor in VHDL. The design section will be a description of the blocks we put in the design. The following instructions are the ones that we implemented:

Table 1: Instructions that need to be supported in Lab-3

Branches	**Memory operation**	**Arithmetic**	**Shift**
BEQ	LB	ADDU	SLL
BNE	LH	ADDIU	SRL
BLTZ	SB	SUBU	SRA
BGEZ	SH	ANDI	SLLV
BLEZ	LBU	ORI	SRLV
BGTZ	LHU	XORI	SRAV
JUMP		LUI	
JR		SLT	
JAL		SLTU	
JALR		SLTI	
		SLTIU	

2 DESIGN

Image of the current design:

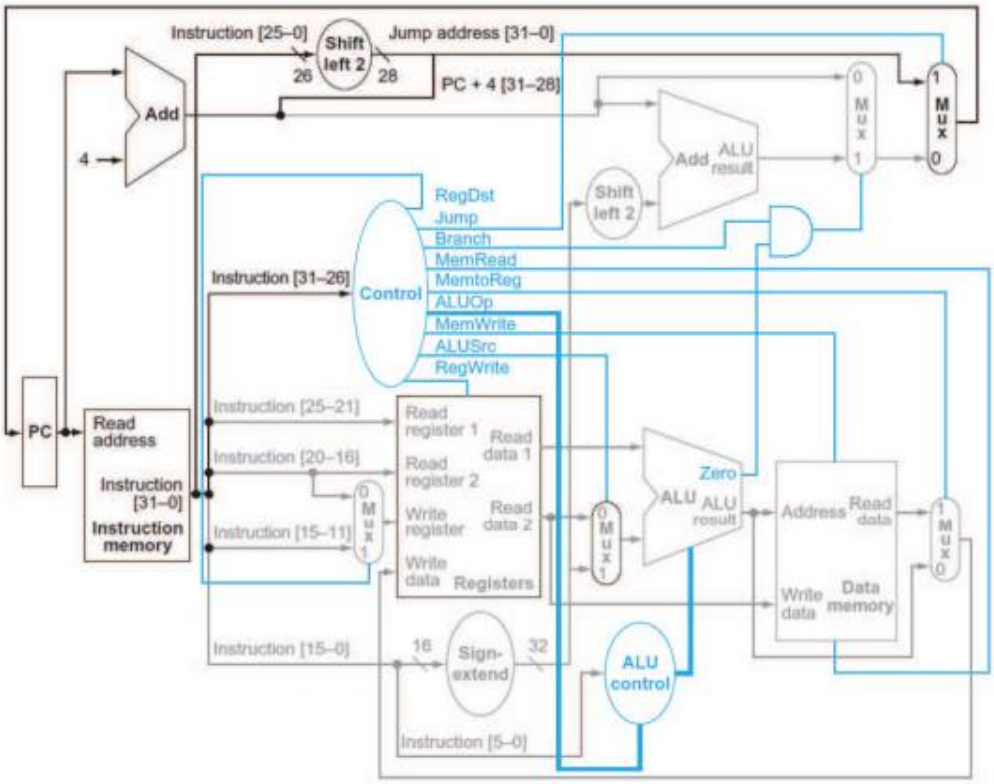


Figure 1: Single-cycle MIPS processor architecture

For our processor, we designed it in the following different blocks:

- The Program Counter (PC)
  1. The program counter contains the address of the instruction that is being executed.
  2. For our program counter we add 1 to the address everytime the clock hits a rising edge.
- PC Adder
  1. This is where we add 1 to the program counter, while the program counter sends the address to the ROM.
- The Instruction Memory (ROM)

1. The instruction memory is implemented as a ROM. This is where the processor will receive the instructions from.
2. The imem.h file is where we store instructions in hexadecimal form.

- The Register File
  1. The register file contains all the registers that each may store 32-bits of data.
- The Data Memory (RAM)
  1. The data memory will be a 512 long array of words with 32-bits each word.
- The ALU
  1. The ALU is where the mathematics of the processor is carried out and then stored in either the register file or the data memory.
  2. The ALU outputs a 0 or a 1 depending if there is a branch instruction.
- The Sign Extender
  1. The sign extender lengthens the 15 downto 0 bits of the instruction to 32-bits as a way to carry out I-type instructions.
- The Multiplexer
  1. The multiplexer selects one of two choices. This is used throughout the processor to select the type of instruction.
- The Controller
  1. The controller chooses the ALU control and is also a factor in controlling the type of instruction.
  2. This helps in controlling the jump and branch instructions as well sending a signal to control each of the muxs.
- The ALU Control
  1. The ALU Control selects the operation the ALU will be doing.
  2. This was implemented inside our processor.

3 TESTS

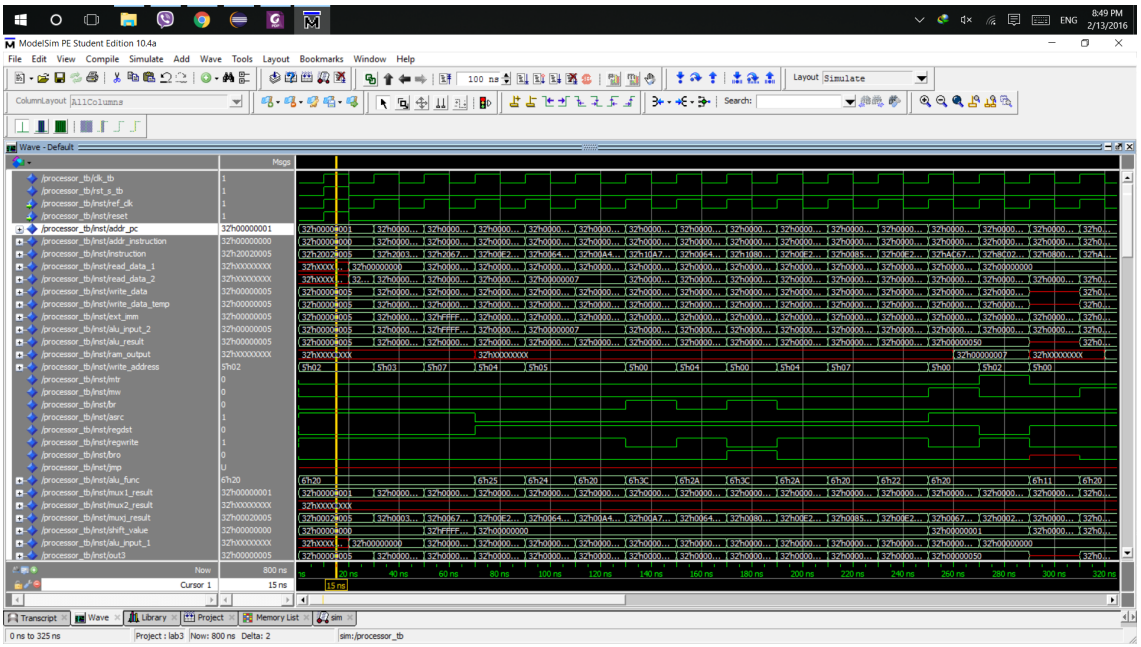
For our tests we first did a reset instruction, then continued with the following commands in our imem.h file:

Code 1: Sample Test program

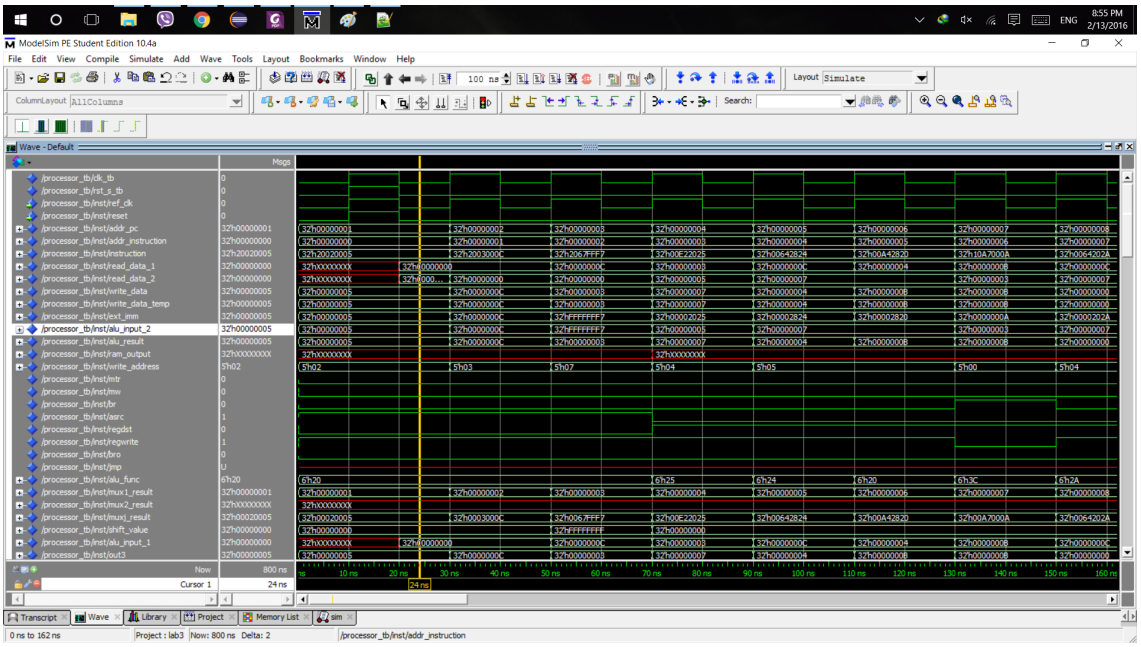
```
20020005
2003000c
2067fff7
00e22025
00642824
00a42820
10a7000a
0064202a
10800001
20050000
00e2202a
00853820
00e23822
ac670044
8c020050
08000011
20020001
ac020054
```

Commands (Translated) and Images:

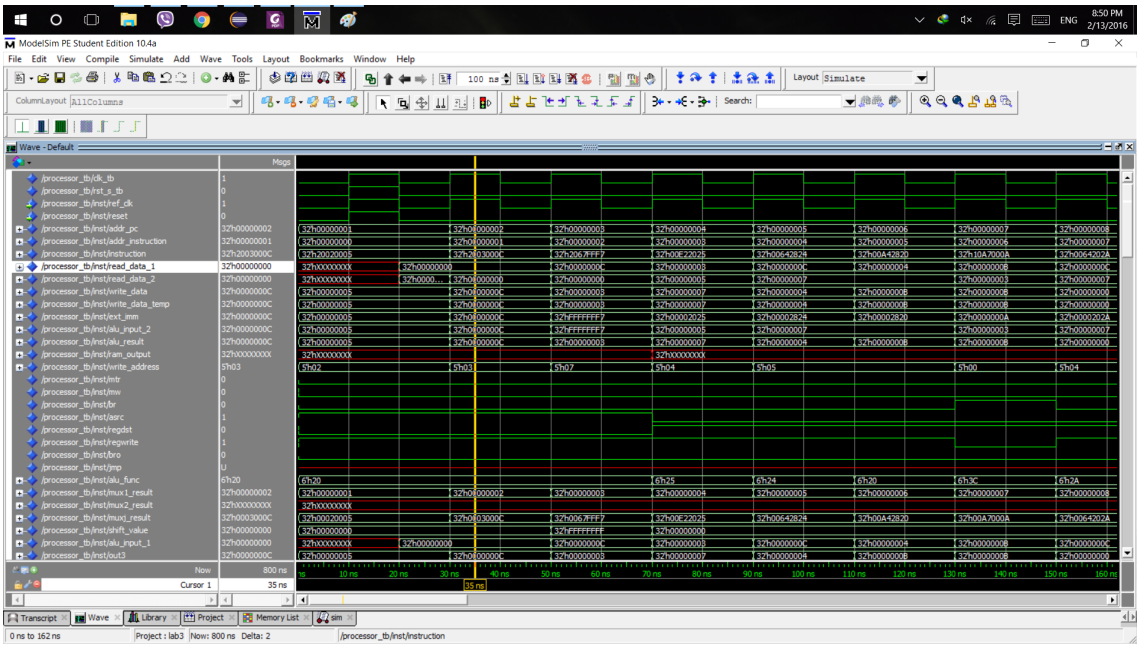
RESET



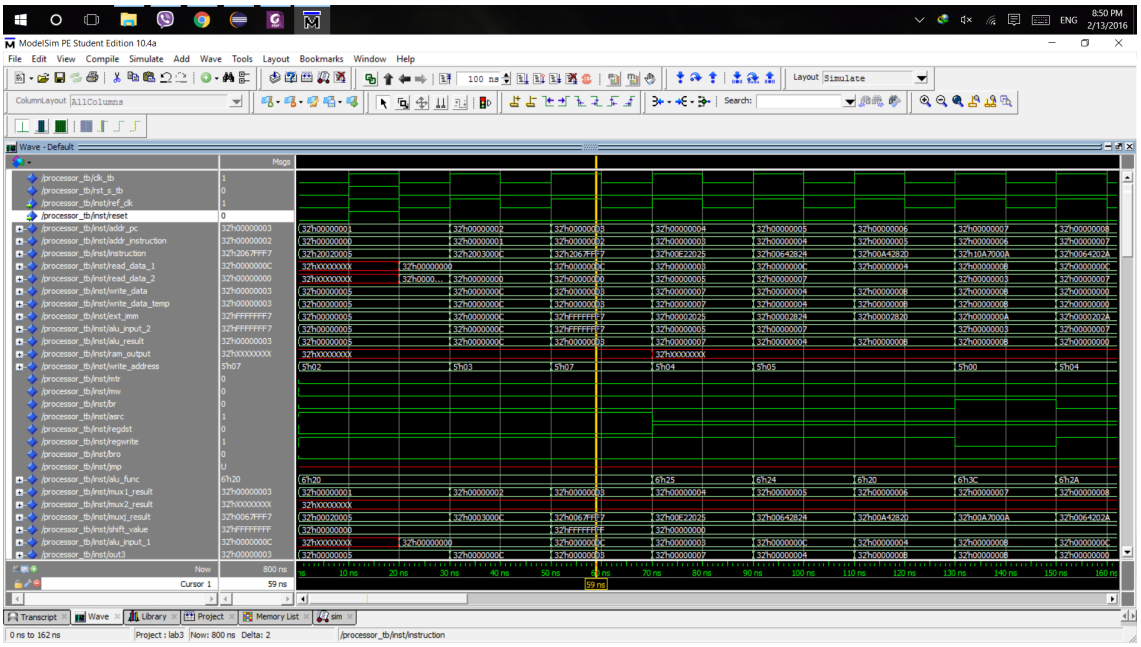
ADDI 2zero 5



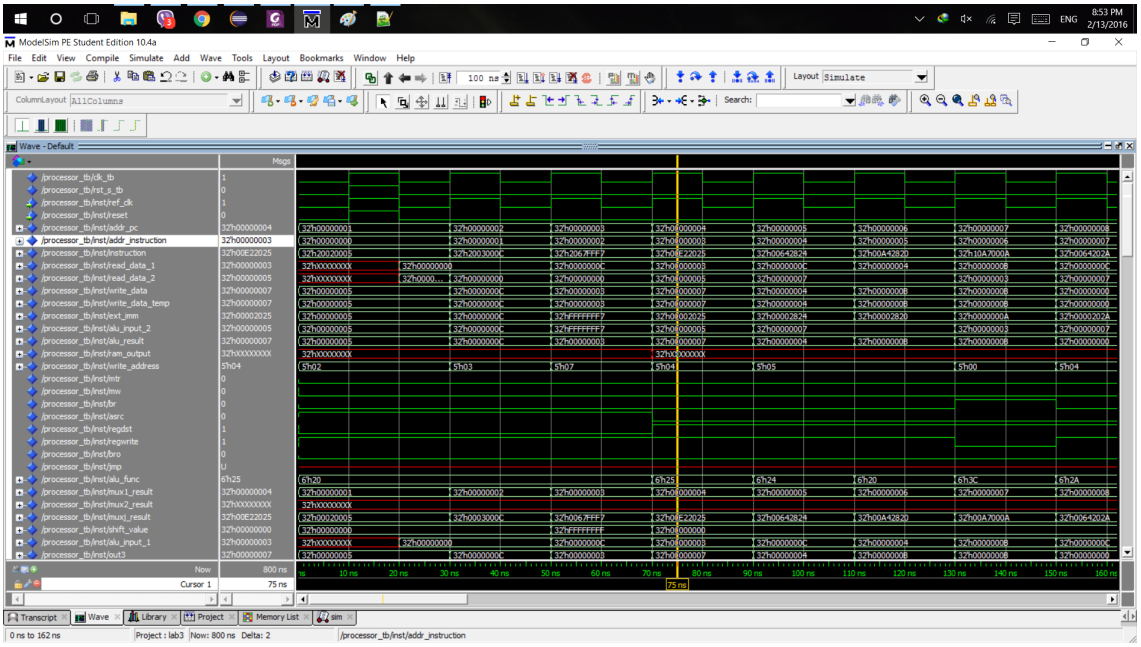
ADDI 3zero 12



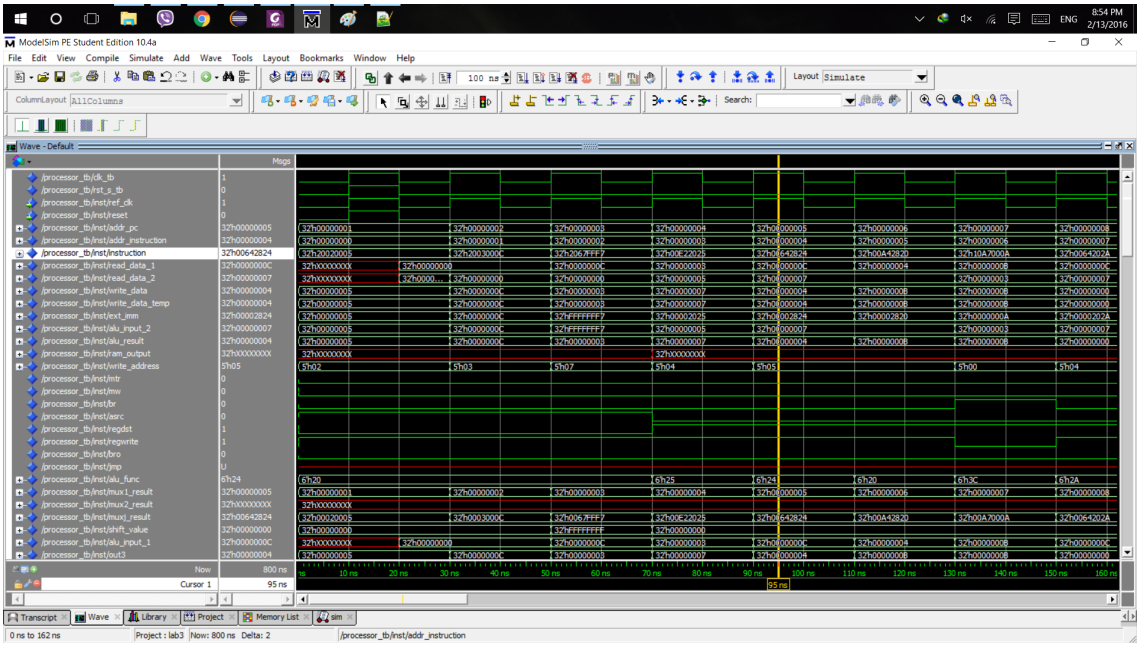
ADDI 73 -9



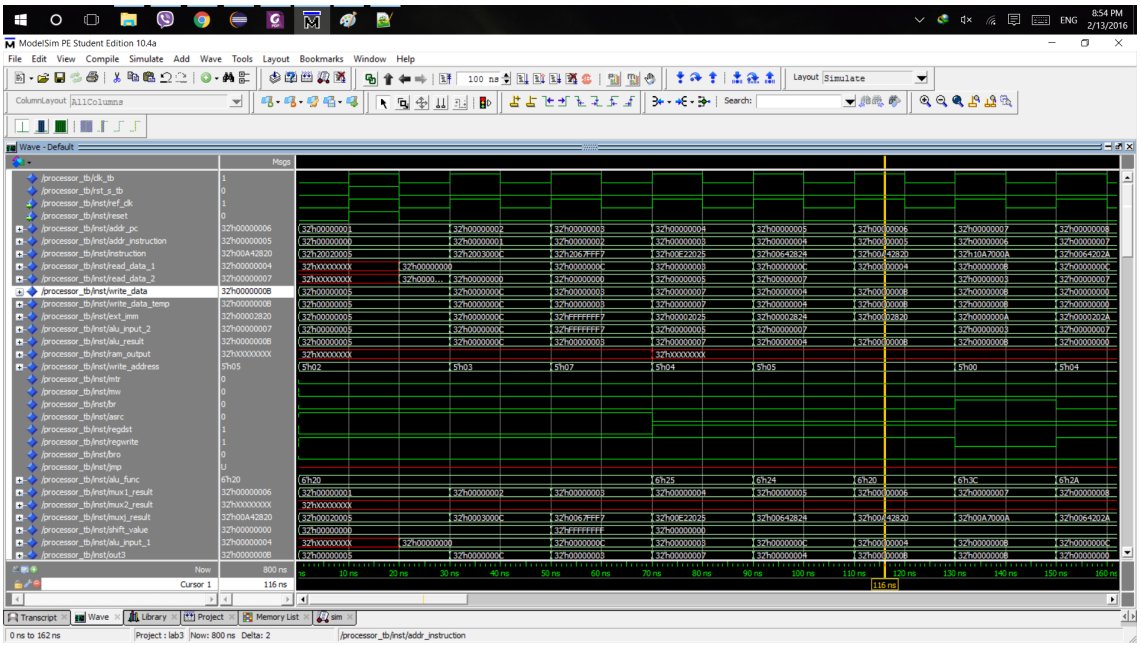
OR $4 $7 $2



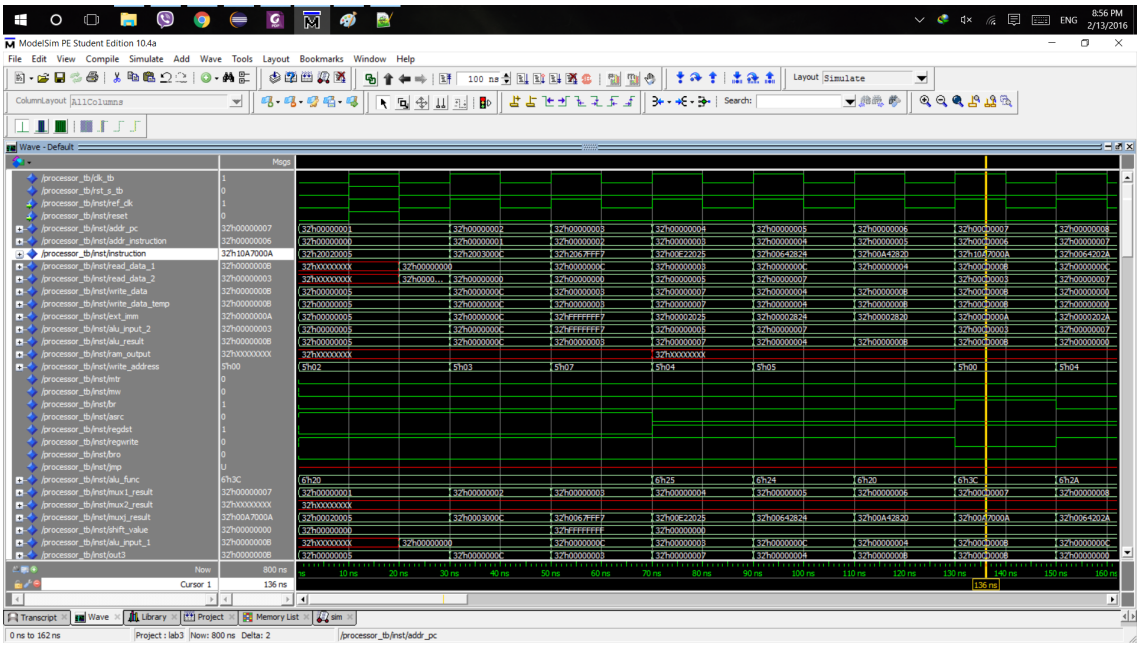
AND $5 $3 $4



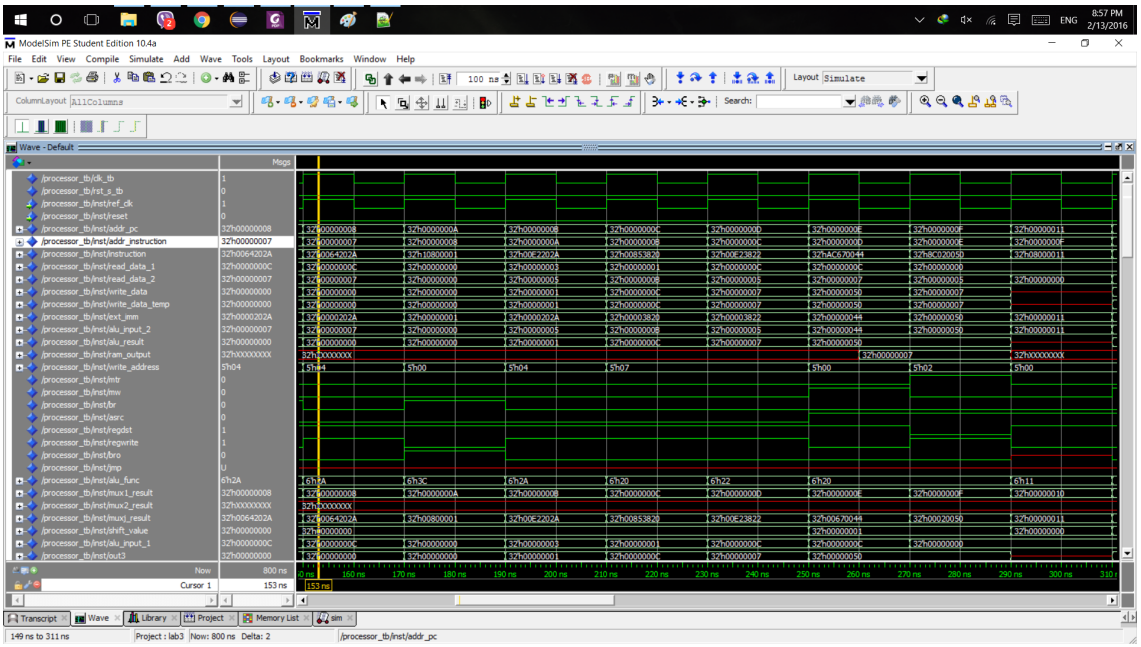
ADD $5 $5 $4



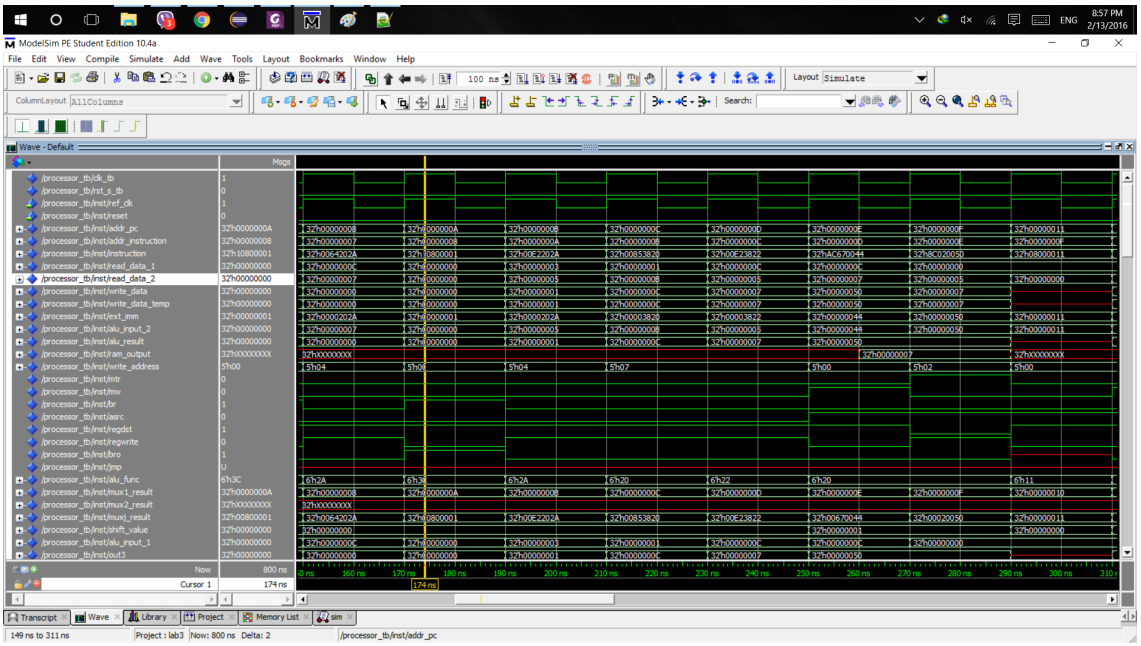
BEQ $5 $7 10



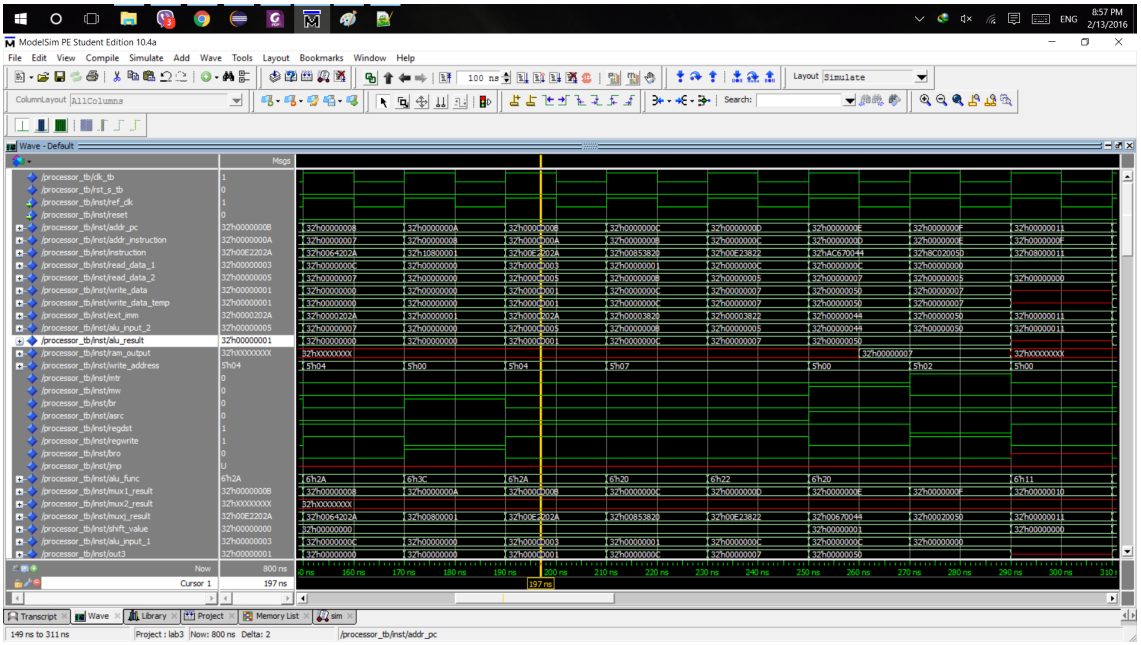
SLT $4 $3 $4



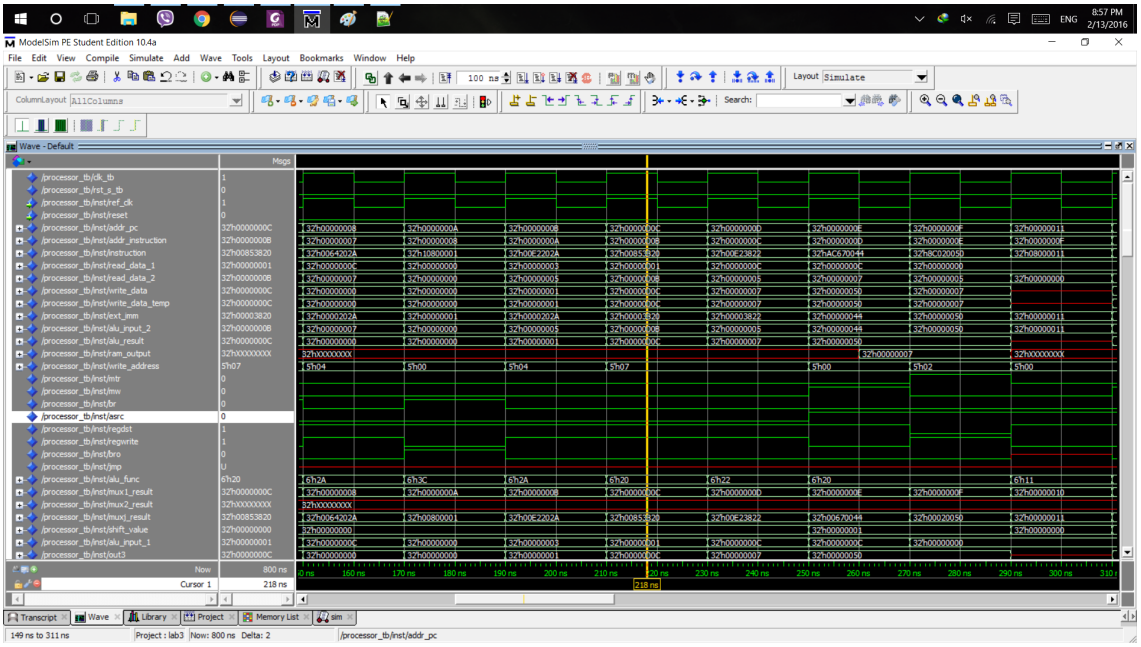
BEQ $4 $zero 1



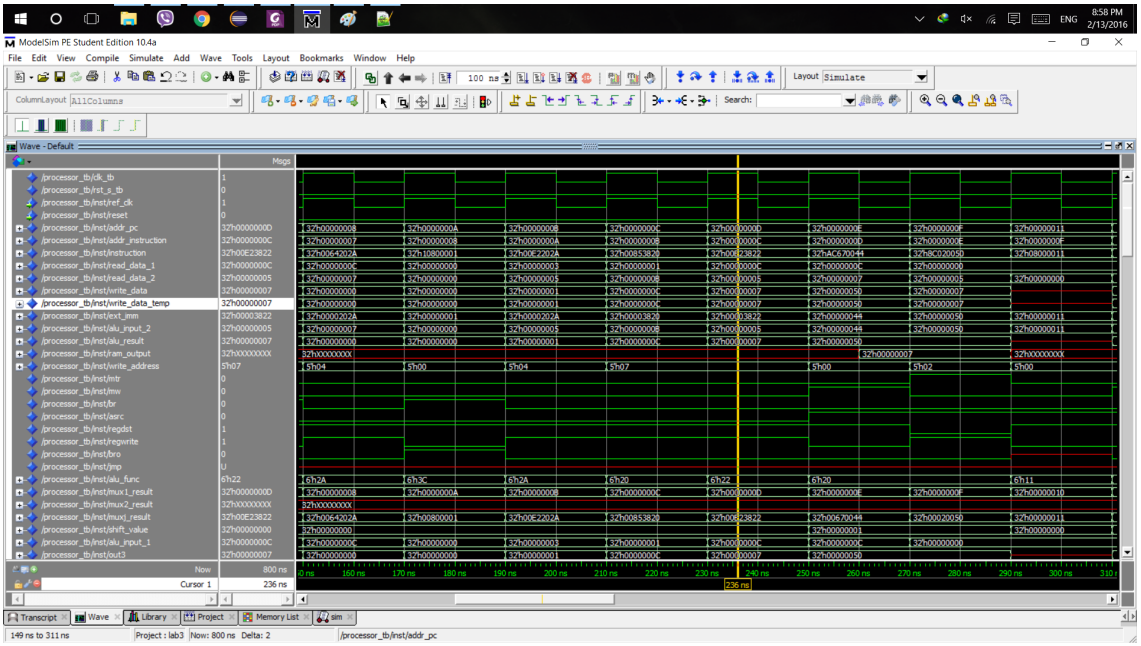
ADDI $5 $zero 0
This instruction does not execute due to the branch.
SLT $4 $7 $2


ModelSim PE Student Edition 10.4a
Wave - Default
Cursor 1
197 ns

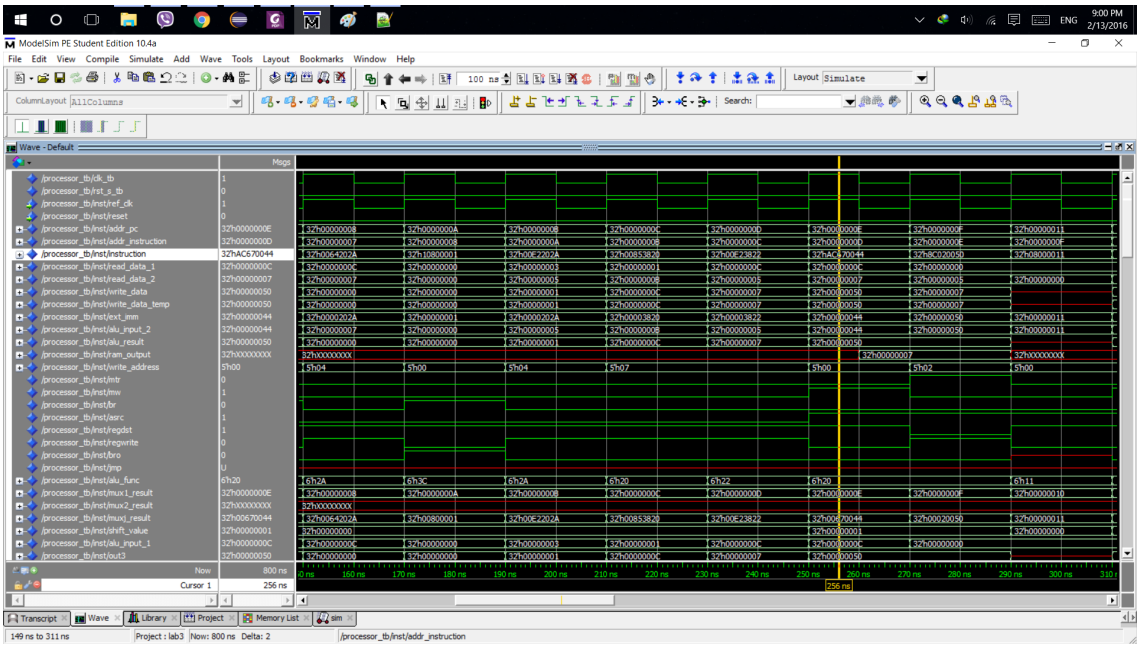
ADD $7 $4 $5


ModelSim PE Student Edition 10.4a
Wave - Default
Cursor 1
218 ns

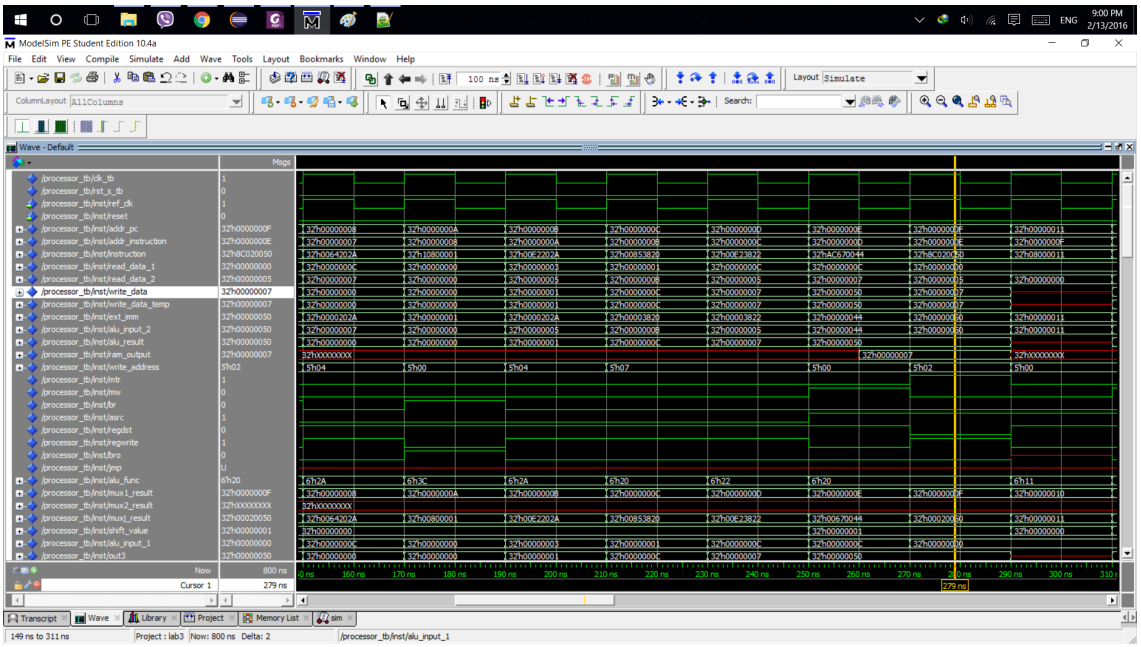
SUB $7 $7 $2



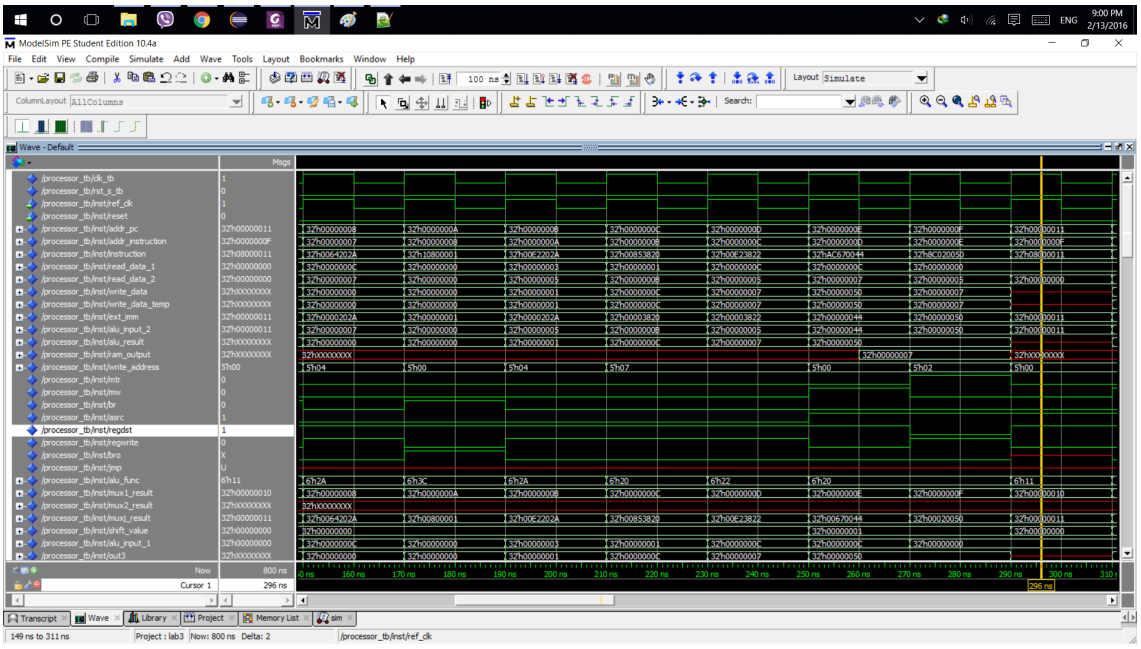
SW $7 68($3)



LW $2 80($zero)

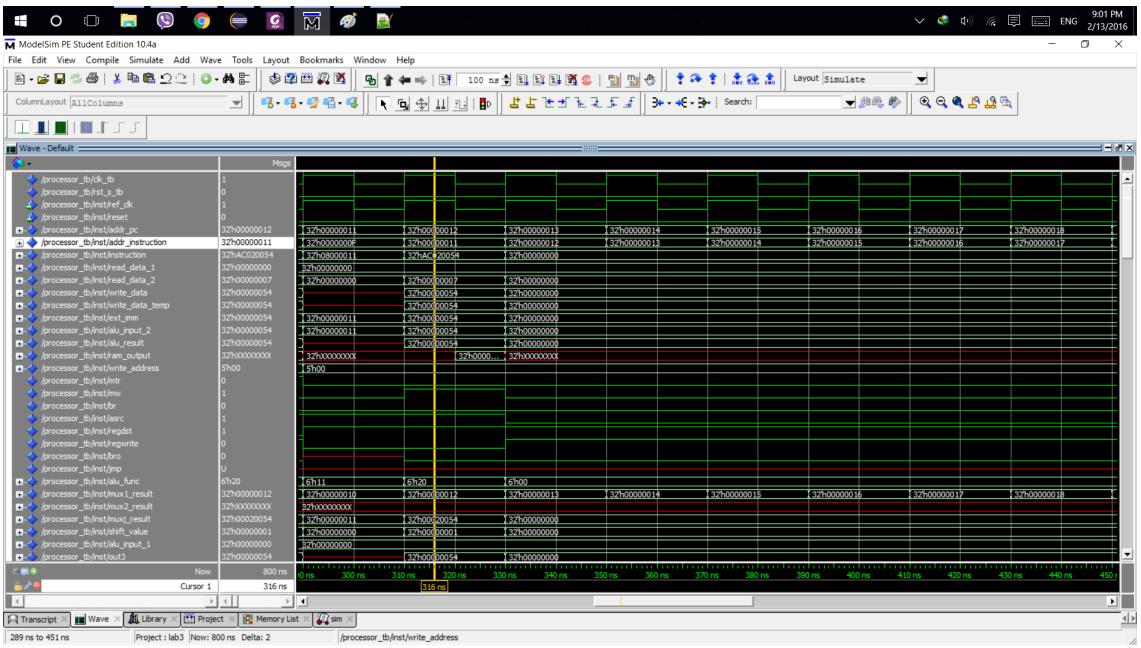


J 17



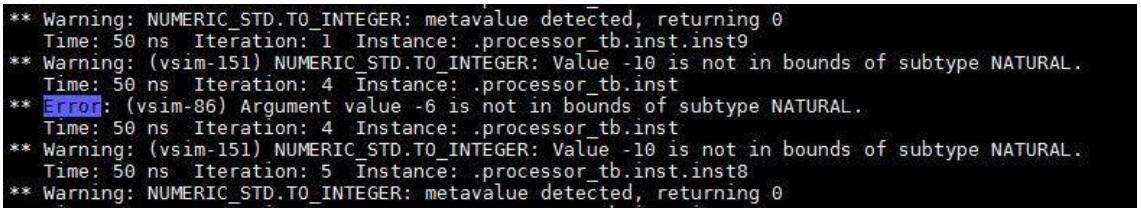
ADDI $2 $zero 1
This instruction does not execute due to jump.

SW $2 84($zero)



4 SYNTHESIS

For the synthesis we received the following results:
Total area = 1685.002450
Power (flat) = 129.350uW
For the more detailed reports of the synthesis of our design they can be found inside the synth/MIPS/reports folder.



5 CONCLUSION

In the end, our program compiled fully with some warnings here and there (the warnings are shown below). These warnings appear at the third instruction we load, which is the "ADDI $7 $3 -9" instruction; however, these warnings do not affect the results of the instructions loaded.