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

March 12, 2016

1 INTRODUCTION

In this lab, we implemented the pipeline optimization technique to speed up the processor in the previous lab. The design section will be a description of the blocks we put in the design.

2 Design

Image of the current design:

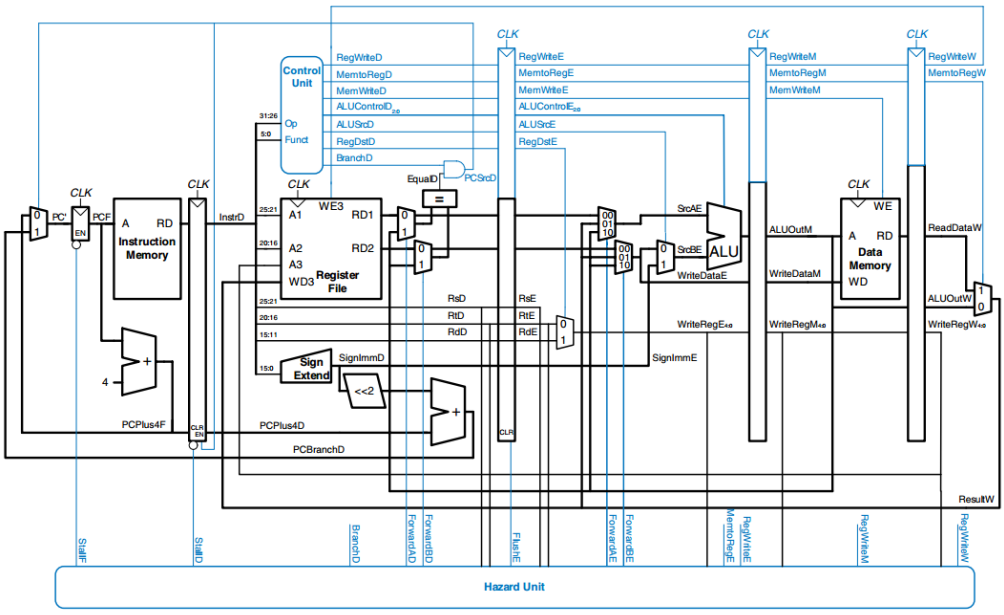


Figure 7.58 **Pipelined processor with full hazard handling**

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 muxes.
- The Hazard Control
    1. The Hazard Control determines how to handle Data and Control Hazard.
    2. This helped forwarding data and stall when needed.
- The D Flip Flop
    1. The D Flip Flop is used to control data transfer between different stages of the pipelining.

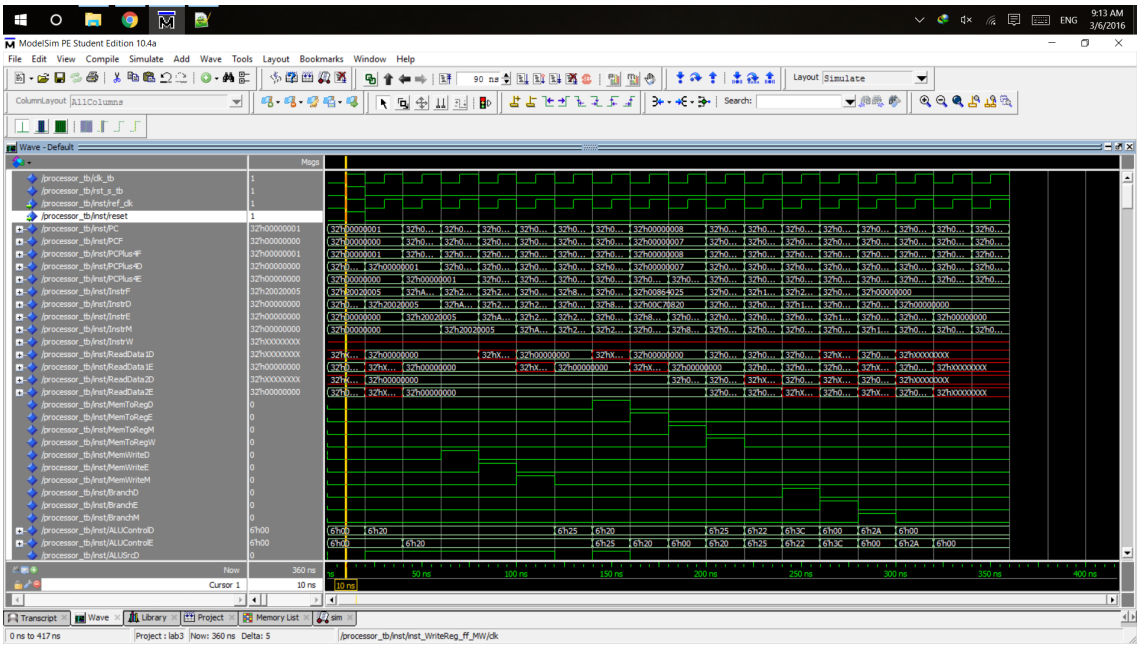
3 TESTS

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

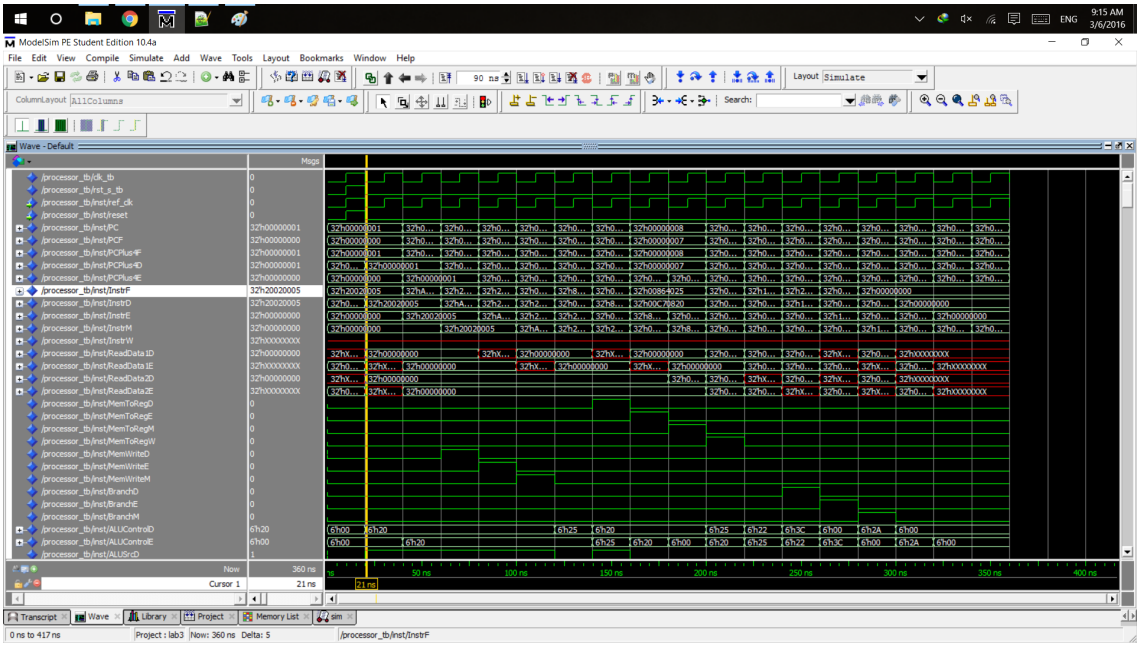
20020005
ac020002
2003000c
2067fff7
00e32025
8c060002
00c70820
00864025
00c04822
10880002
24ce000c
20020001
0064202a

Commands (Translated) and Images:

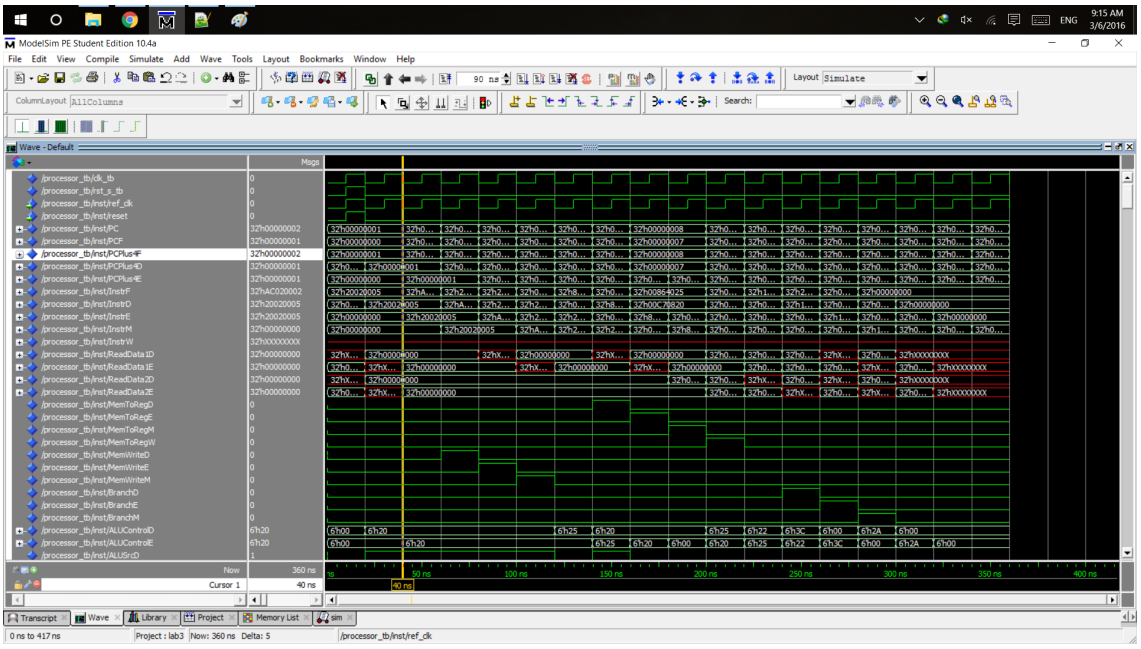
RESET



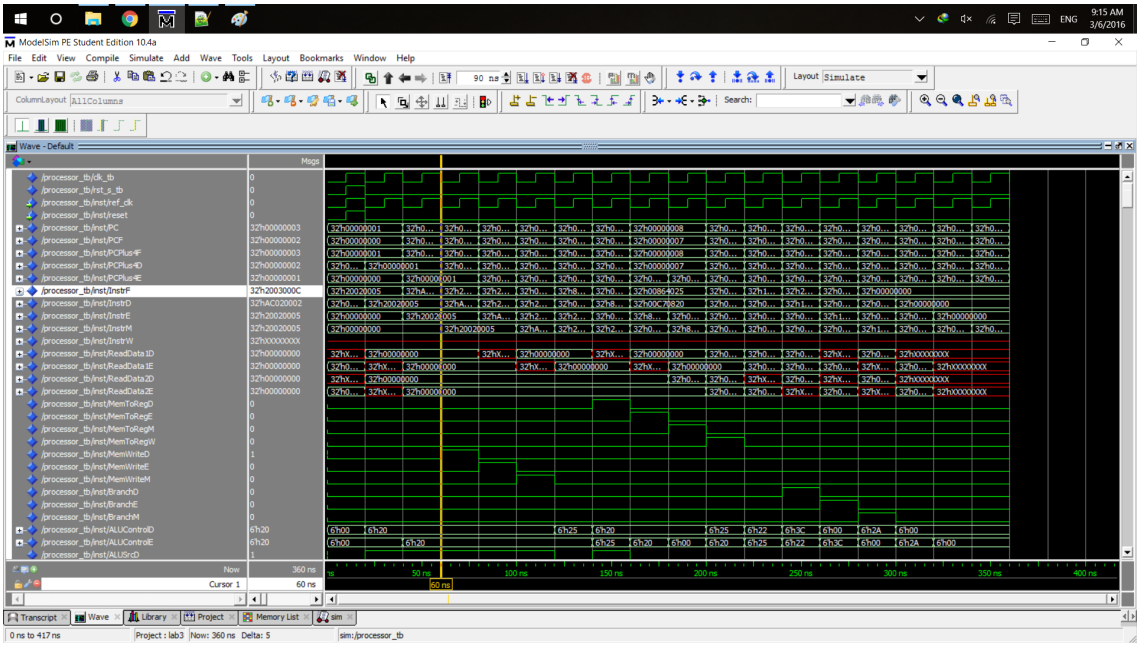
ADDI $2 $zero 5



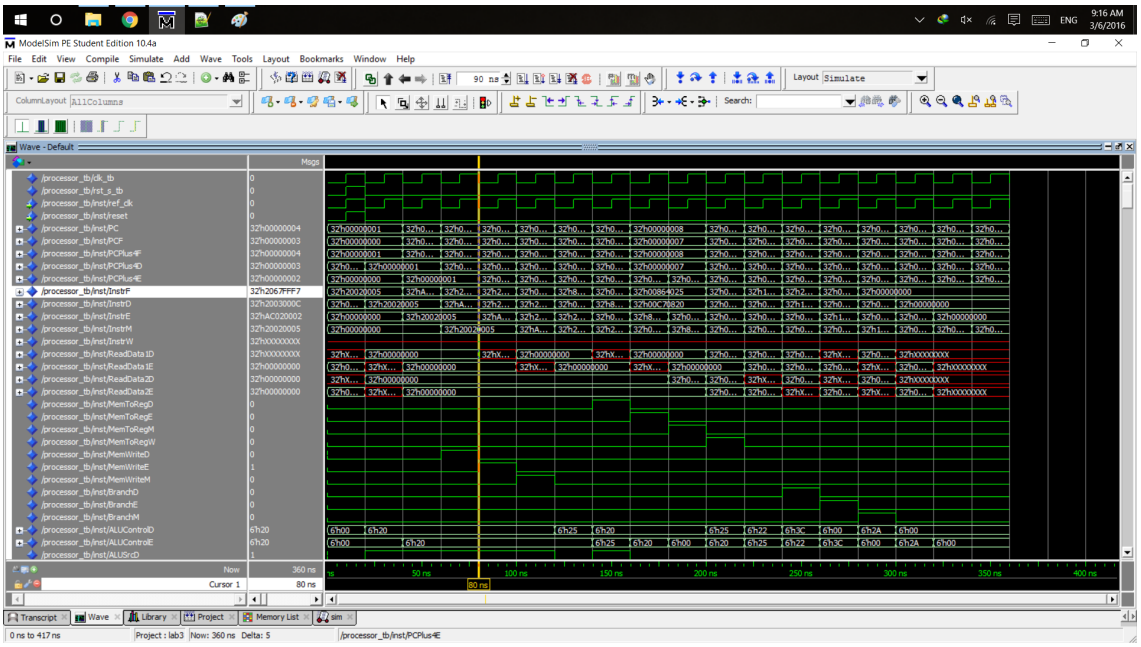
SW $2 2($zero)



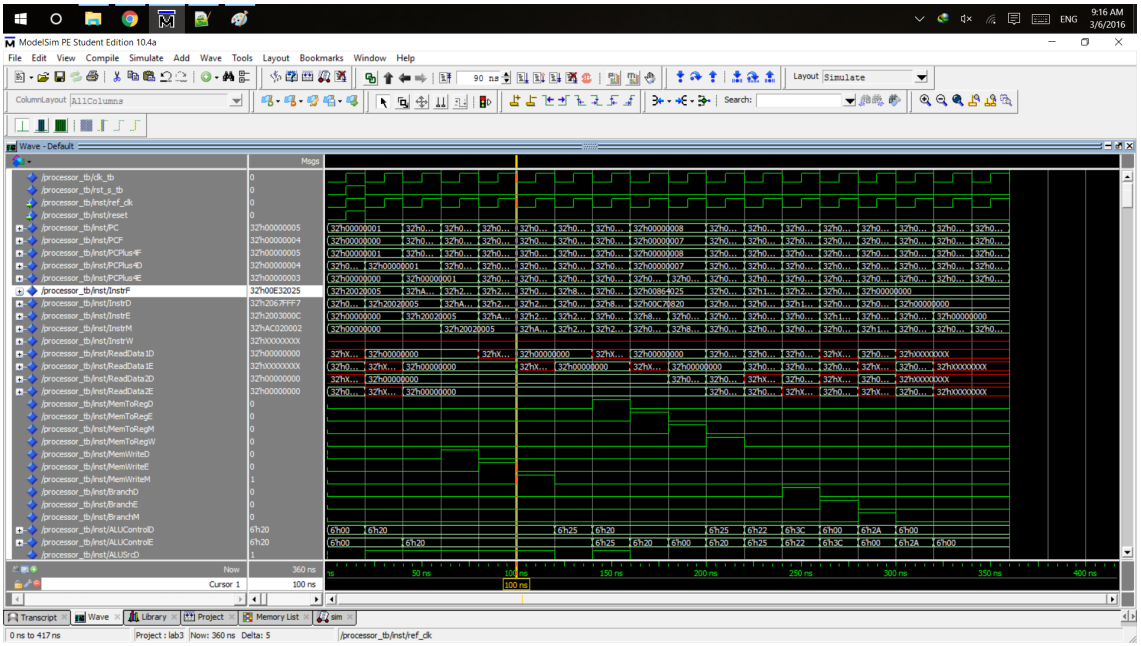
ADDI $3 $zero 12



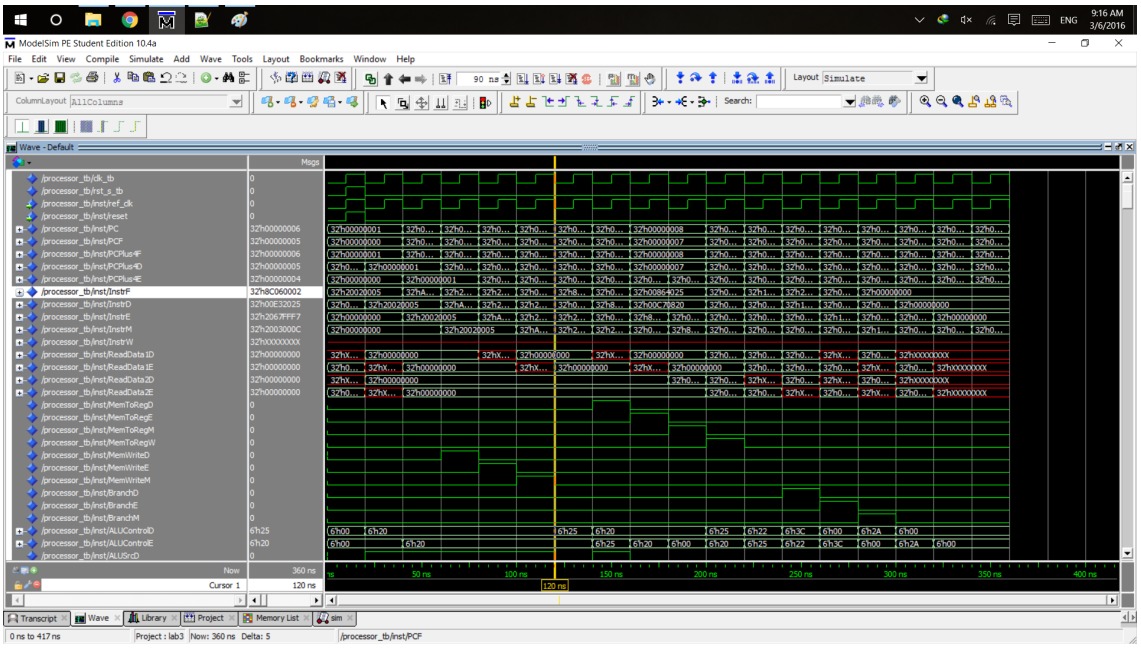
ADDI $7 $3 -9



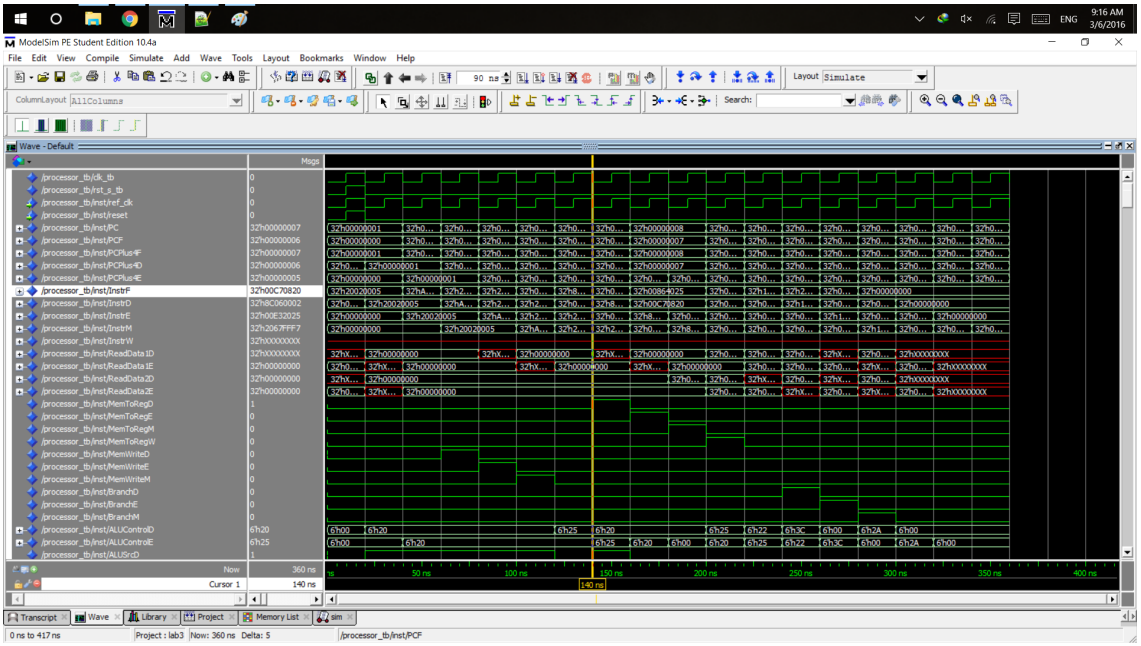
OR $4 $7 $3



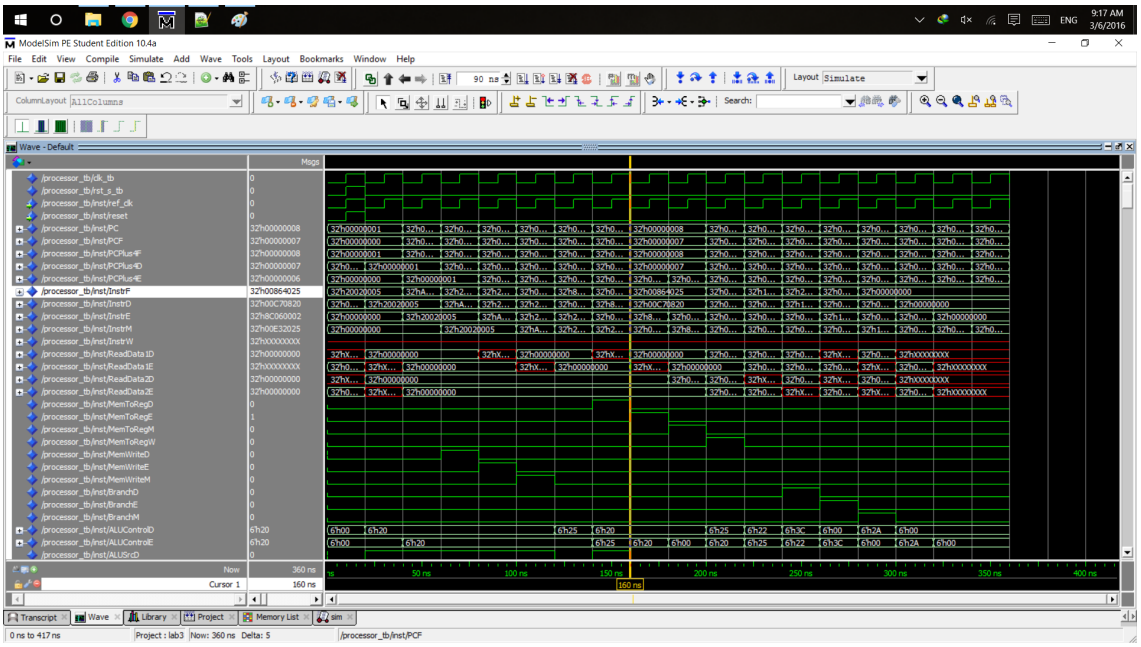
LW $6 2($zero)



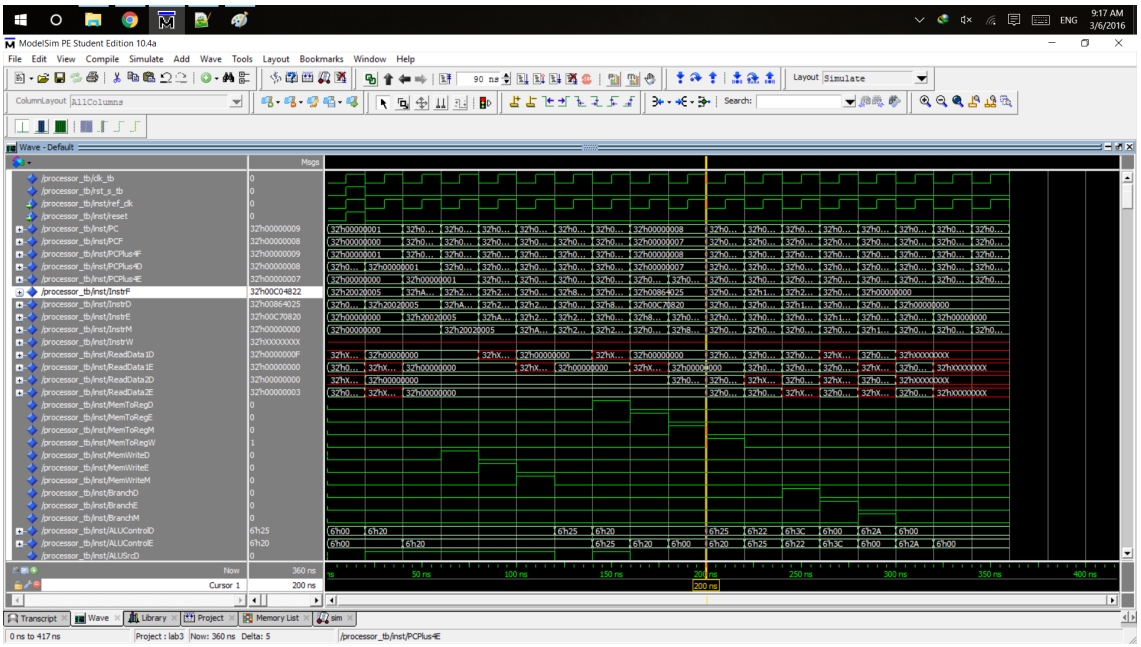
ADD $1 $6 $7



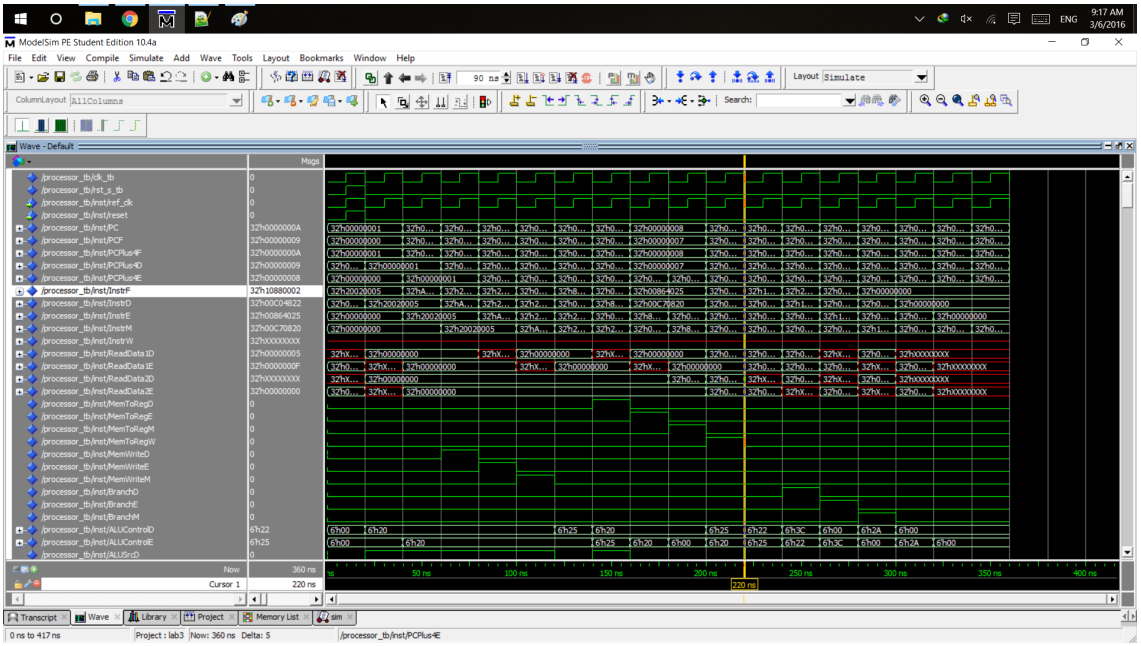
OR $8 $4 $6



SUB $9 $6 $zero

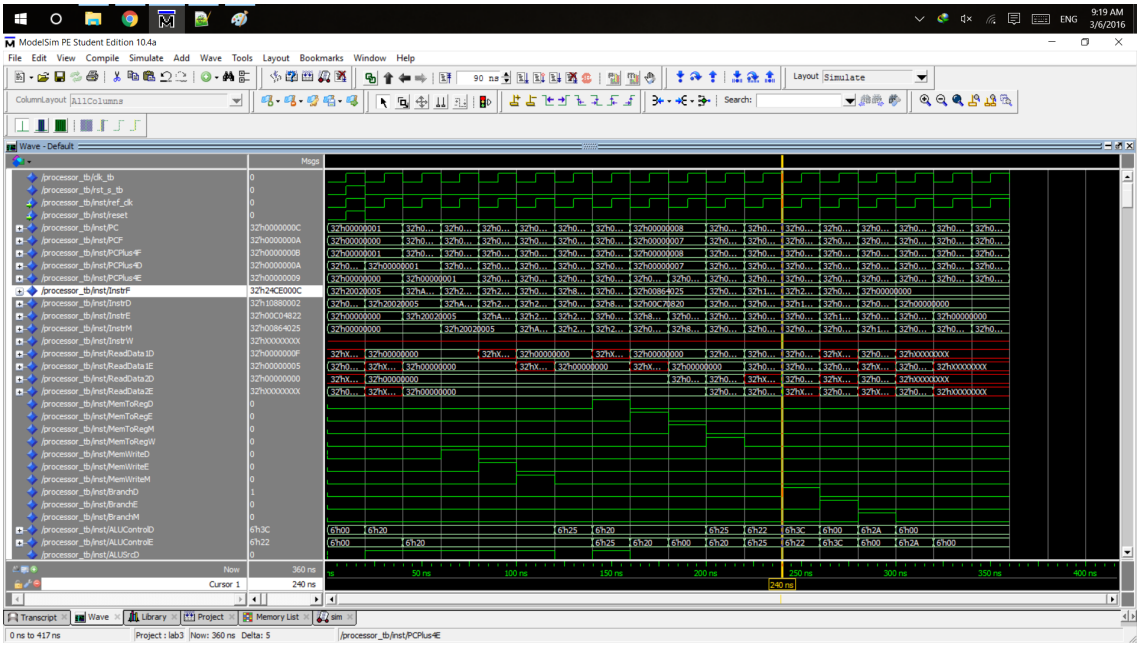


BEQ $4 $8 2

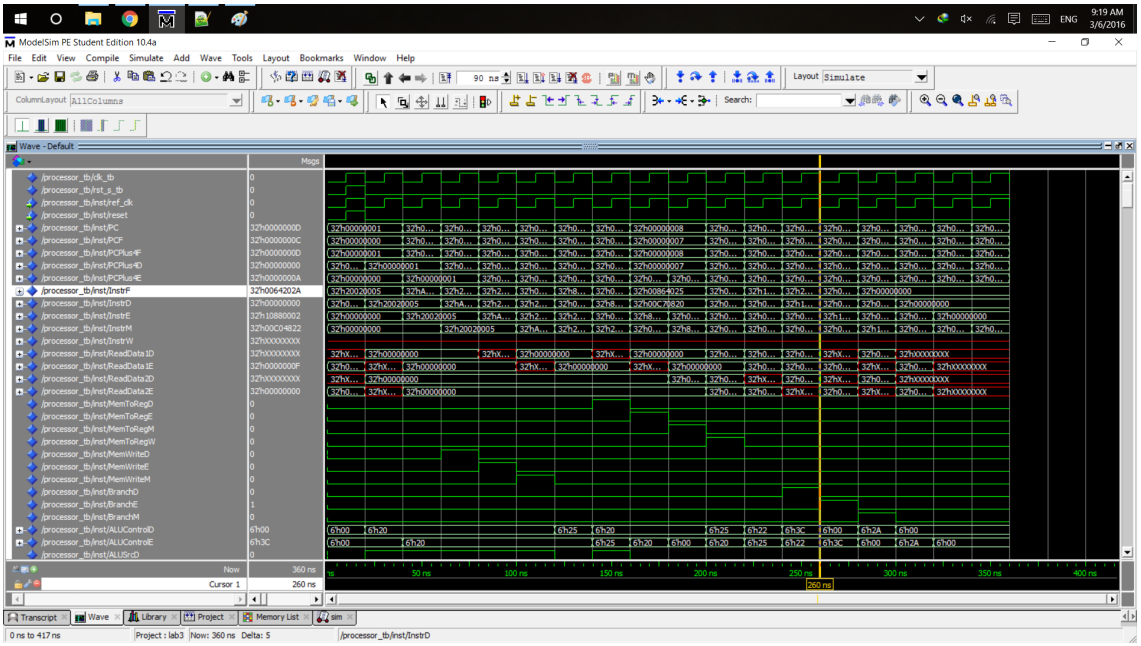


ADDI $14 $6 12

This instruction is fetched, but then flushed because the branching condition is met.



SLT $4 $3 $4


ModelSim PE Student Edition 10.4a
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
/processor_tb/clk_tb
/processor_tb/rst_s_tb
/processor_tb/inst/ref_clk
/processor_tb/inst/reset
/processor_tb/inst/PC
/processor_tb/inst/PCF
/processor_tb/inst/PCPlus4F
/processor_tb/inst/PCPlus4D
/processor_tb/inst/PCPlus4E
/processor_tb/inst/InstrF
/processor_tb/inst/InstrD
/processor_tb/inst/InstrE
/processor_tb/inst/InstrM
/processor_tb/inst/InstrW
/processor_tb/inst/ReadData1D
/processor_tb/inst/ReadData1E
/processor_tb/inst/ReadData2D
/processor_tb/inst/ReadData2E
/processor_tb/inst/MemToRegD
/processor_tb/inst/MemToRegE
/processor_tb/inst/MemToRegM
/processor_tb/inst/MemToRegW
/processor_tb/inst/MemWriteD
/processor_tb/inst/MemWriteE
/processor_tb/inst/MemWriteM
/processor_tb/inst/BranchD
/processor_tb/inst/BranchE
/processor_tb/inst/BranchM
/processor_tb/inst/ALUControlD
/processor_tb/inst/ALUControlE
/processor_tb/inst/ALUSrcD
Now
Cursor 1
Transcript
Wave
Library
Project
Memory List
sim
0 ns to 417 ns
Project : lab3
Now: 360 ns
Delta: 5
/processor_tb/inst/InstrD

4 SYNTHESIS

For the synthesis we received the following results:
Cell Area = 422.895
Design Area = 539.083
Power = 558.645
Critical Path = 0.07 Max frequency = 1/0.07 = 14.28
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. These warnings do not affect the results of the instructions loaded. All instructions were tested, data hazard and control hazard are handled properly.

In addition, we weren't be able to perform the synthesis part of the last assignment, so there are no data to be compared. However, theoretically, we believe our CPU has been speed up significantly as a result of pipelining implementation.