CprE 381: Computer Organization and Assembly Level Programming, Spring 2018

Report – Project Part 2

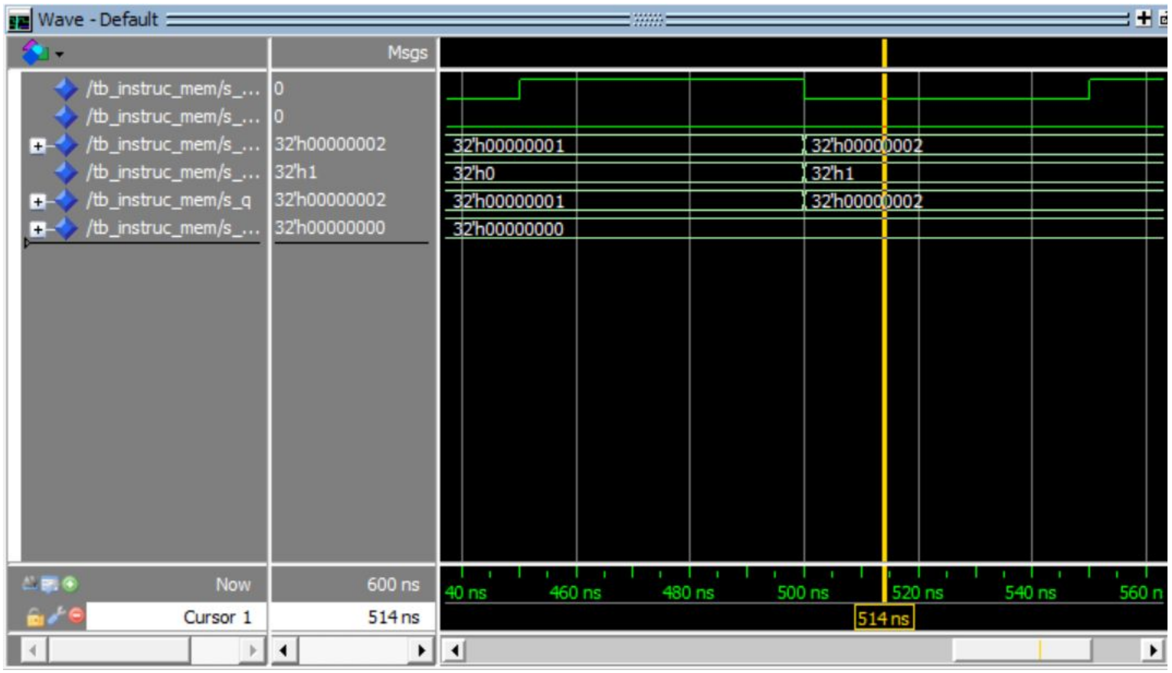
Lab Partners	Daniel Limanowski and Vishal Joel
Section/Lab Time	B/Thurs @ 10AM
Canvas Group ID	B09

***Refer to the highlighted language in the project 2 instruction for the context of the following questions*.**

a. [Part (a.1.a)] Create a spreadsheet detailing the list of M instructions to be supported in this part alongside their binary 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 by your datapath implementation. *[You can attach the spreadsheet as a separate file.]*

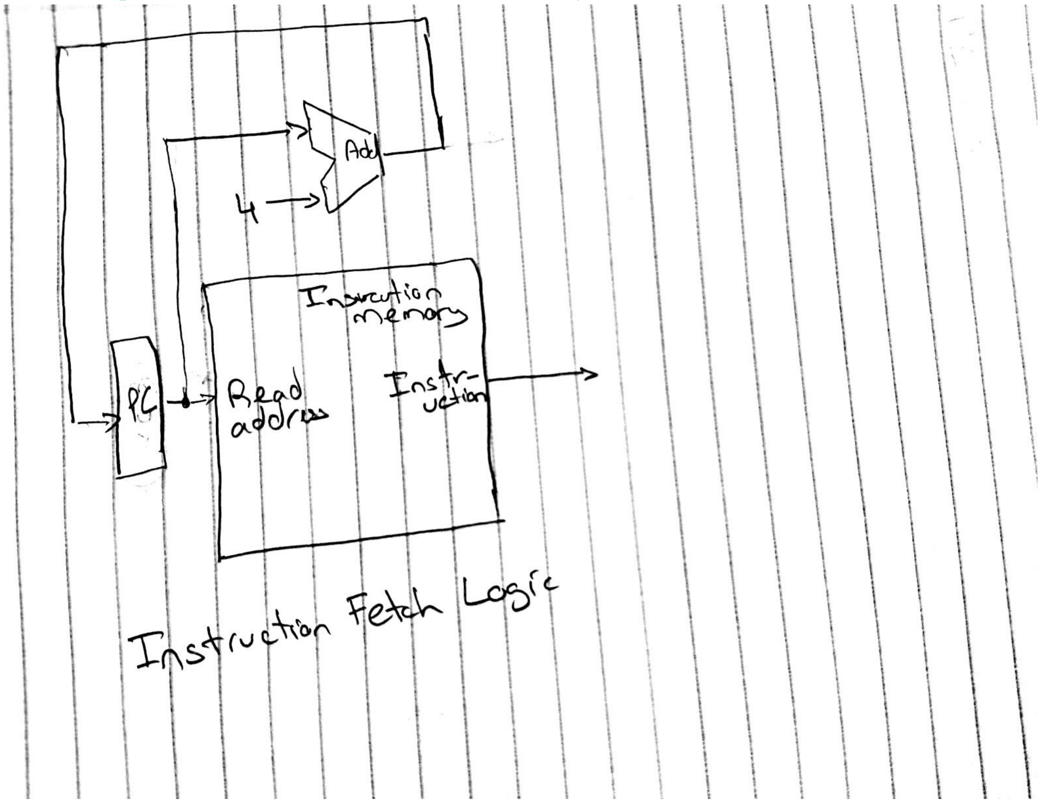
The spreadsheet has been attached as a separate file.

b. [Part (a.1.b)] Create a testbench to test this module individually, and show that your output matches the expected control signals from problem 1(a). *[Please attach waveforms and explanations.]*



c. [Part (a.2.b)] Provide a simple testbench that confirms that you can instantiate and test the mem.vhd component as would be needed for a MIPS instruction memory. Briefly describe what ports are **not** needed for instruction memory, and how you wired the mem.vhd module accordingly.

The ports that are not needed are WE (write enable) and data (the data to write to the memory. We are only reading from the instruction memory on clock high, so we can simply force WE and DATA to 0 by wiring them as such. The program counter value goes into the mem module as addr and q is what we feed into the rest of the MIPS processor. Our clock is fed into the memory's clk.

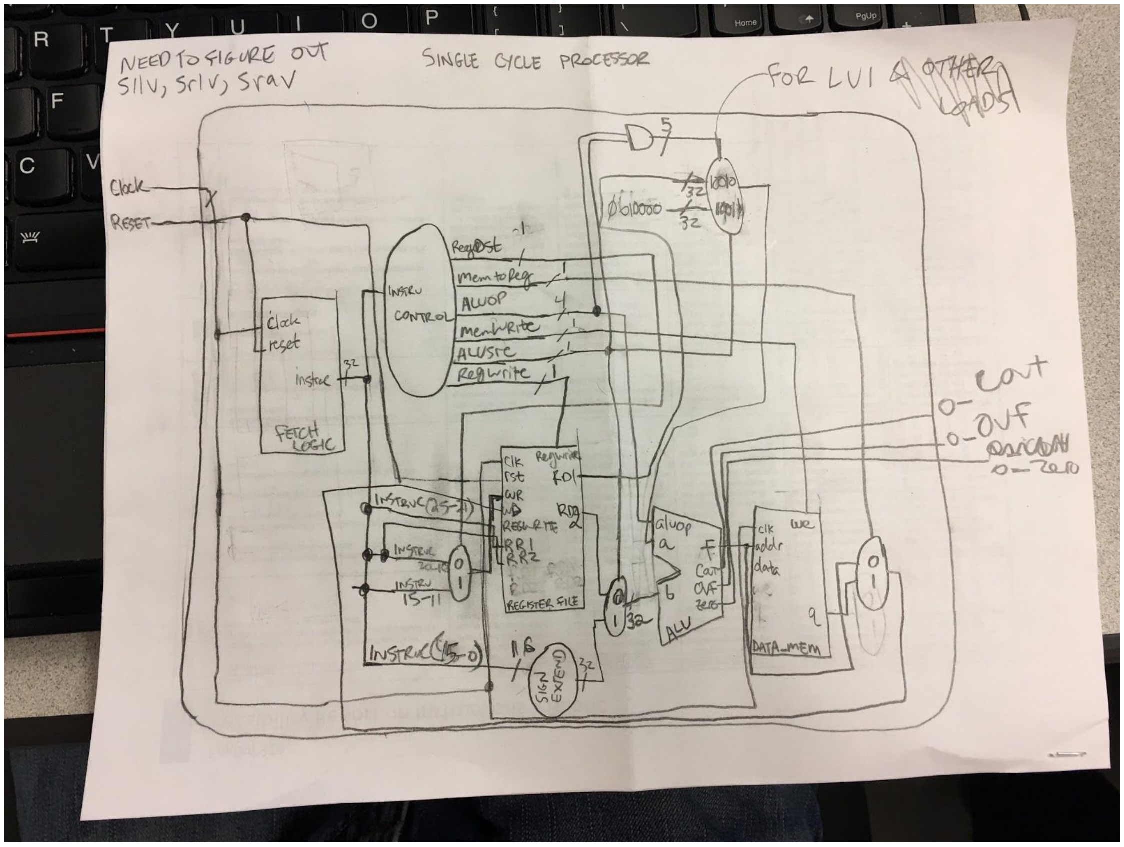


Instruction Fetch Logic Schematic

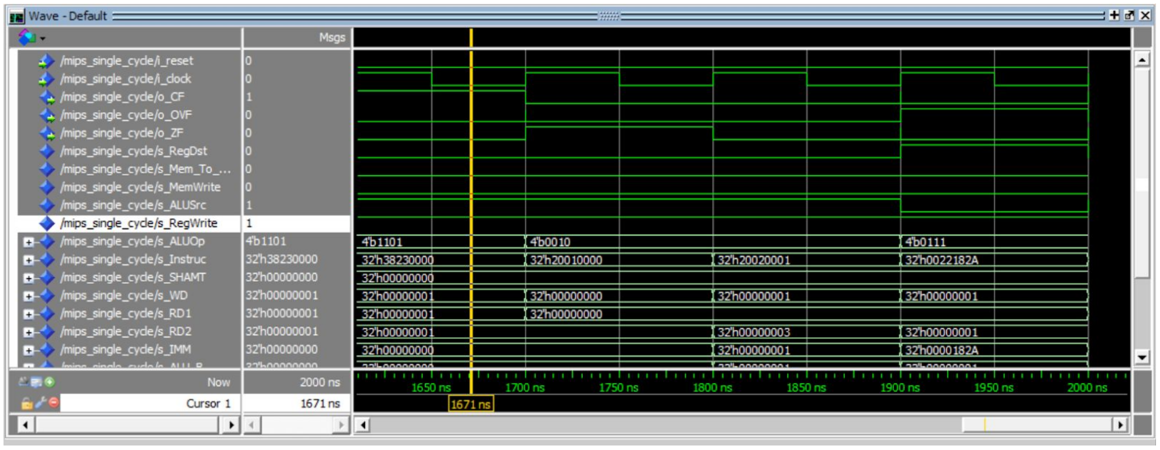
d. [Part (a.3)] In your writeup, describe what challenges (if any) you faced in implementing this module.

The processor required support for all shift operations. This caused issues with sending data to ALU "A" argument to support all the shifting instructions. To make this work, I had to make a new module, using structural VHDL, to select the proper data to go to the ALU's "A" port. The data was selected based on the current values of ALUOP and ALUSrc

Diagram:

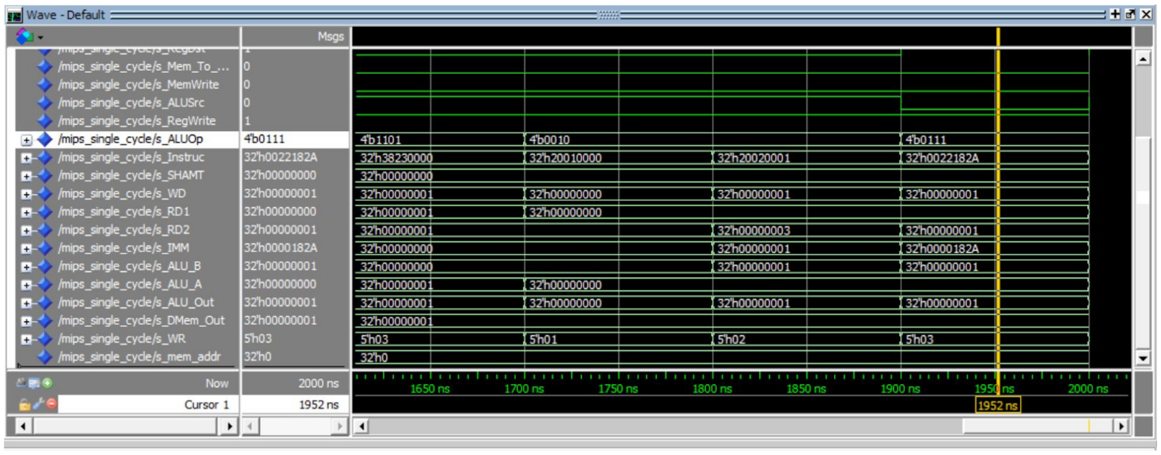


e. [Part (a.4)] Create a test application that makes use of every arithmetic/logical/shift instruction, and attach waveforms.


```
Wave - Default
                                  Msgs
/mips_single_cycle/i_reset        0
/mips_single_cycle/i_clock        0
/mips_single_cycle/o_CF           1
/mips_single_cycle/o_OVF          0
/mips_single_cycle/o_ZF           0
/mips_single_cycle/s_RegDst       0
/mips_single_cycle/s_Mem_To_...   0
/mips_single_cycle/s_MemWrite     0
/mips_single_cycle/s_ALUSrc       1
/mips_single_cycle/s_RegWrite     1
/mips_single_cycle/s_ALUOp        4'b1101       4'b1101      4'b0010                    4'b0111
/mips_single_cycle/s_Instruc      32'h38230000  32'h38230000 32'h20010000 32'h20020001 32'h0022182A
/mips_single_cycle/s_SHAMT        32'h00000000  32'h00000000
/mips_single_cycle/s_WD           32'h00000001  32'h00000001 32'h00000000 32'h00000001 32'h00000001
/mips_single_cycle/s_RD1          32'h00000001  32'h00000001 32'h00000000
/mips_single_cycle/s_RD2          32'h00000001  32'h00000001              32'h00000003 32'h00000001
/mips_single_cycle/s_IMM          32'h00000000  32'h00000000              32'h00000001 32'h0000182A

Now       2000 ns    1650 ns  1700 ns  1750 ns  1800 ns  1850 ns  1900 ns  1950 ns  2000 ns
Cursor 1  1671 ns    1671 ns
```

MIPS Single Cycle - XOR


Wave - Default

	Msgs
/mips_single_cycle/s_RegDst	1
/mips_single_cycle/s_Mem_To_...	0
/mips_single_cycle/s_MemWrite	0
/mips_single_cycle/s_ALUSrc	0
/mips_single_cycle/s_RegWrite	1
/mips_single_cycle/s_ALUOp	4'b0111
/mips_single_cycle/s_Instruc	32'h0022182A
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000001
/mips_single_cycle/s_RD1	32'h00000000
/mips_single_cycle/s_RD2	32'h00000001
/mips_single_cycle/s_IMM	32'h0000182A
/mips_single_cycle/s_ALU_B	32'h00000001
/mips_single_cycle/s_ALU_A	32'h00000000
/mips_single_cycle/s_ALU_Out	32'h00000001
/mips_single_cycle/s_DMem_Out	32'h00000001
/mips_single_cycle/s_WR	5'h03
/mips_single_cycle/s_mem_addr	32'h0
Now	2000 ns
Cursor 1	1952 ns

MIPS Single Cycle - SLT

f. [Part (b.6.a)] Describe these possibilities as a function of the different control flow-related instructions from Part (a.1.a).

There are four possible ways that the control flow instructions can affect instruction fetching. They are:

1) "Regular" instructions (add, sub, sll) that increment PC by PC + 4
2) Conditional jump instructions (bne, beq) that alter PC by IMM on a specific condition
3) Unconditional jump instructions (J, JAL) that alter PC by IMM always
4) Jump register instructions that alter PC by a register's contents

Our instruction fetch logic must support these four ways of altering Program Counter (PC) with the different instructions.

g. [Part (b.6.b)] Draw a schematic for the instruction fetch logic and any other datapath modifications needed for control flow instructions. What additional control signals are needed?

Additional control signals needed:

- o_JAL
- o_JR
- o_J
- o_BNE
- o_BEQ

Each new control signal identifies when that instruction is in place. These in turn act as multiplexor selectors and inputs to some and/or gates to ensure the right PC is calculated going forward.

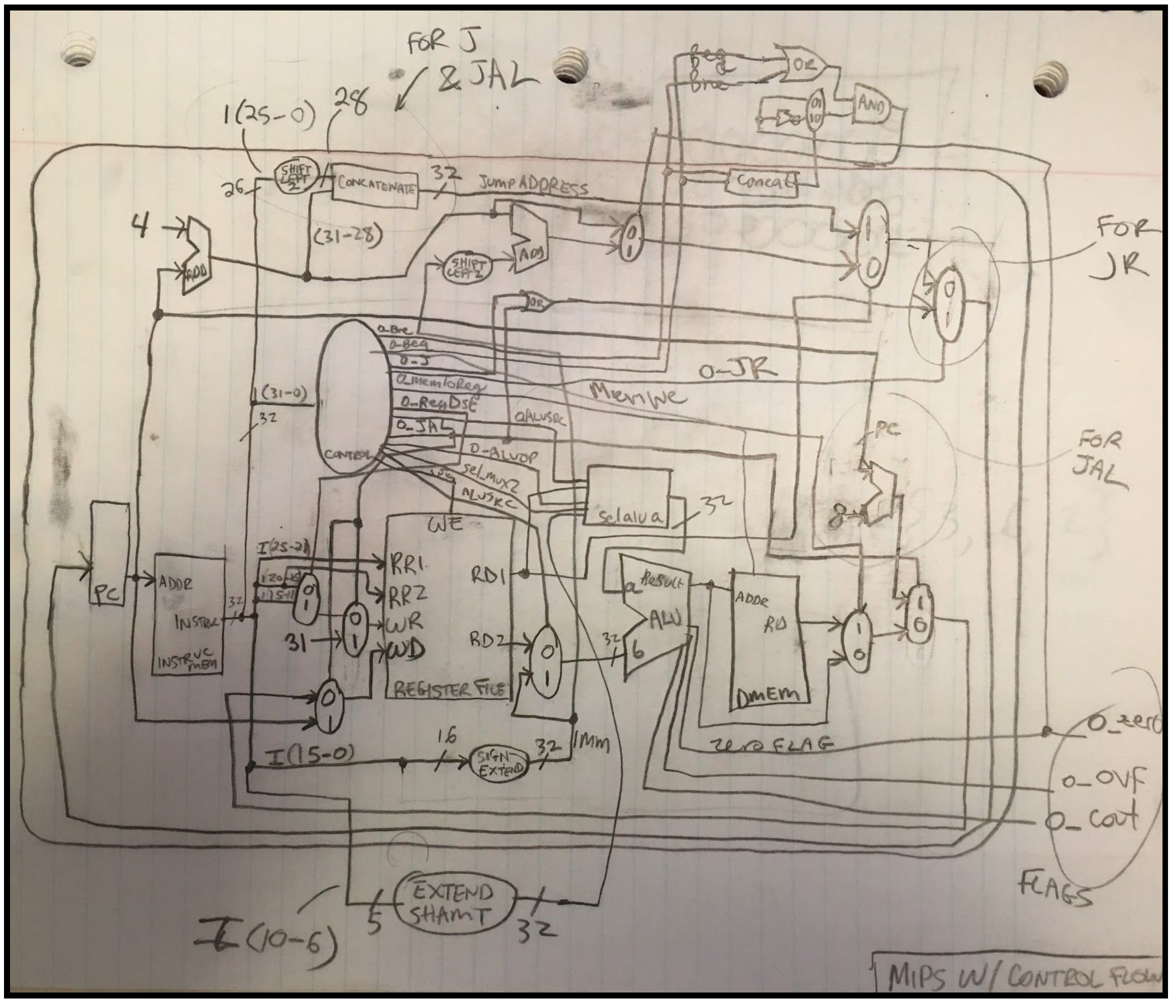


Photo: Wiring of the datapath with the new control flow implemented. Clock and reset signals have been left out for brevity and easier reading.

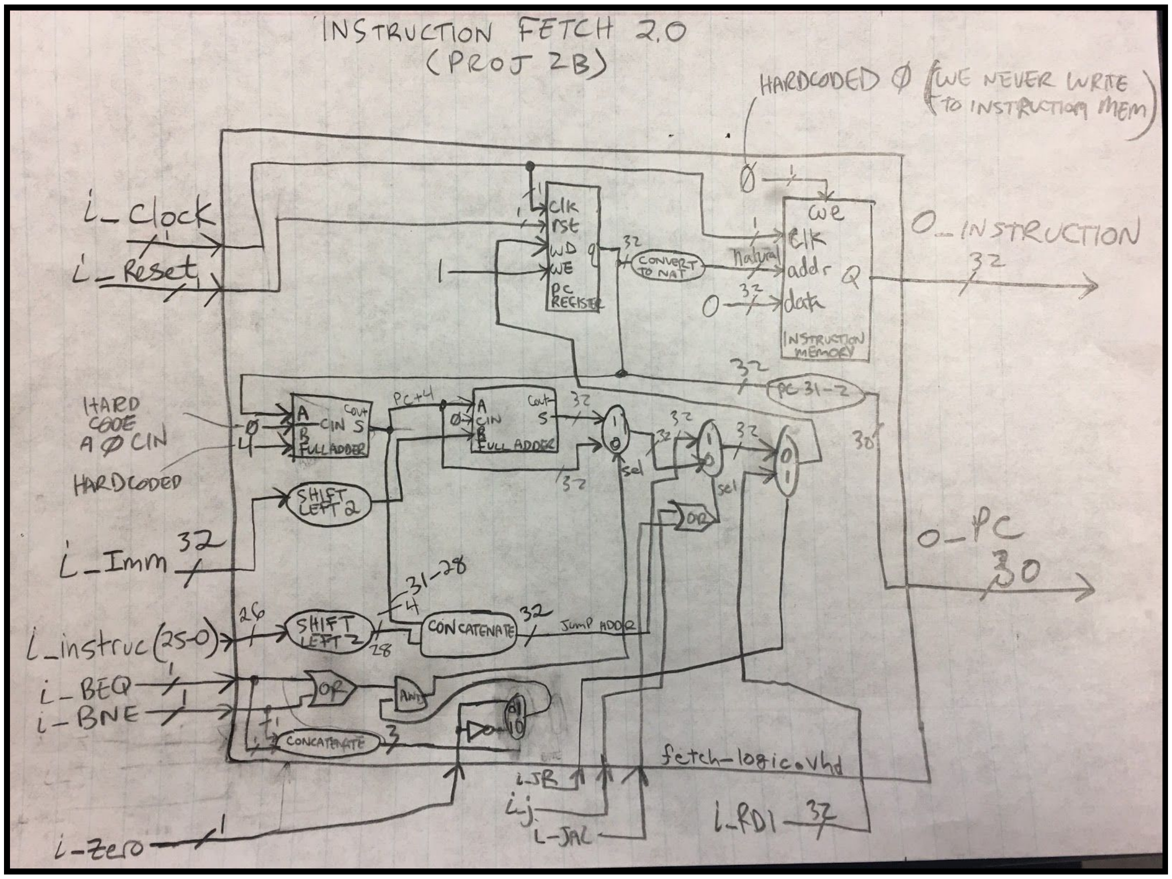
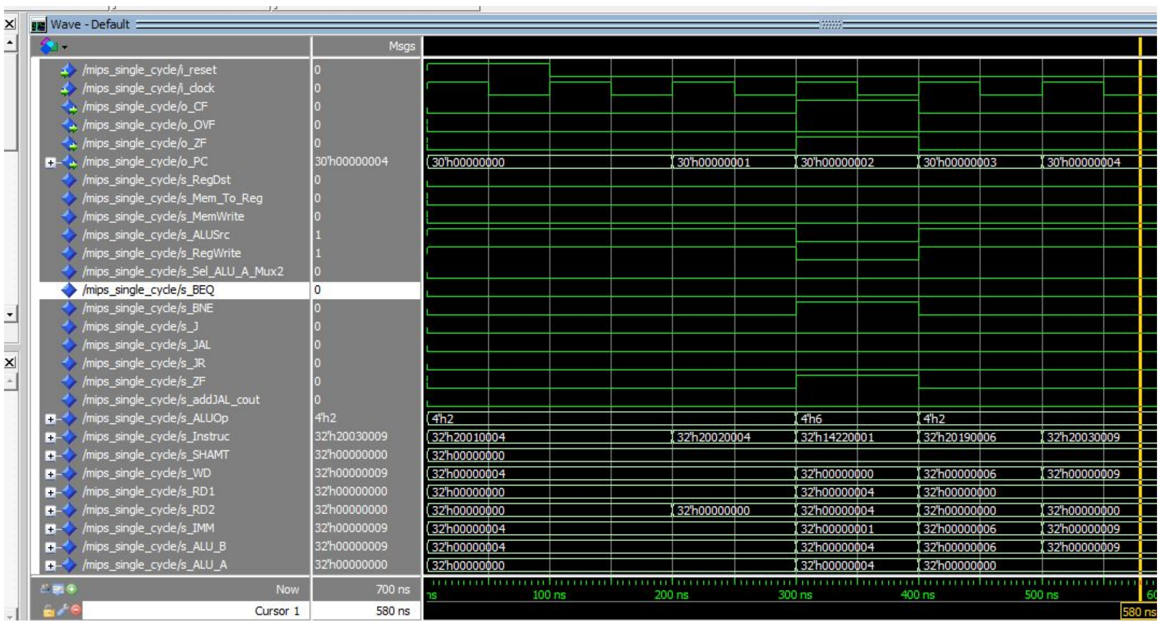


Photo: Instruction Fetch Module (updated for control flow instructions)

h. [Part (b.6.c)] Attach waveforms, and describe how the execution of the control flow possibilities corresponds to the Modelsim waveforms in your writeup.

BNE

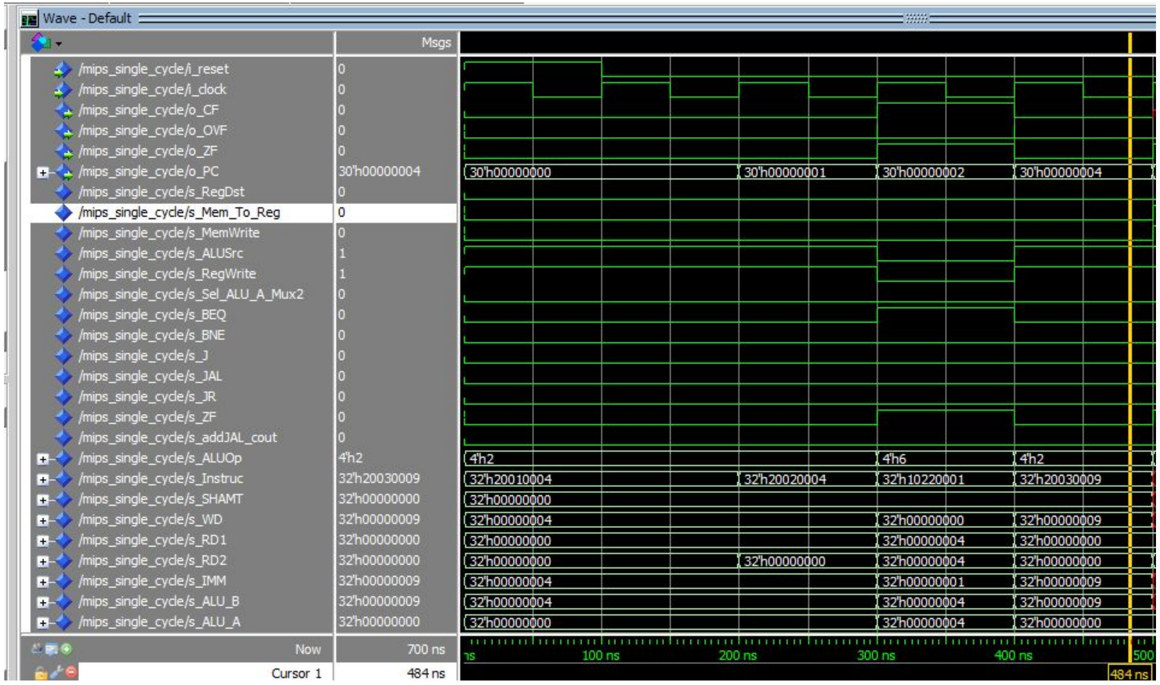
In the photo below, we branch to the add 5 instruction when $1 and $2 do NOT equal each other.



```
Wave - Default
                                   Msgs
/mips_single_cycle/i_reset         0
/mips_single_cycle/i_clock         0
/mips_single_cycle/o_CF            0
/mips_single_cycle/o_OVF           0
/mips_single_cycle/o_ZF            0
/mips_single_cycle/o_PC            30'h00000004   30'h00000000   30'h00000001   30'h00000002   30'h00000003   30'h00000004
/mips_single_cycle/s_RegDst        0
/mips_single_cycle/s_Mem_To_Reg    0
/mips_single_cycle/s_MemWrite      0
/mips_single_cycle/s_ALUSrc        1
/mips_single_cycle/s_RegWrite      1
/mips_single_cycle/s_Sel_ALU_A_Mux2 0
/mips_single_cycle/s_BEQ           0
/mips_single_cycle/s_BNE           0
/mips_single_cycle/s_J             0
/mips_single_cycle/s_JAL           0
/mips_single_cycle/s_JR            0
/mips_single_cycle/s_ZF            0
/mips_single_cycle/s_addJAL_cout   0
/mips_single_cycle/s_ALUOp         4'h2           4'h2           4'h6           4'h2
/mips_single_cycle/s_Instruc       32'h20030009   32'h20010004   32'h20020004   32'h14220001   32'h20190006   32'h20030009
/mips_single_cycle/s_SHAMT         32'h00000000   32'h00000000
/mips_single_cycle/s_WD            32'h00000009   32'h00000004   32'h00000000   32'h00000006   32'h00000009
/mips_single_cycle/s_RD1           32'h00000000   32'h00000000   32'h00000004   32'h00000000
/mips_single_cycle/s_RD2           32'h00000000   32'h00000000   32'h00000000   32'h00000004   32'h00000000   32'h00000000
/mips_single_cycle/s_IMM           32'h00000009   32'h00000004   32'h00000001   32'h00000006   32'h00000009
/mips_single_cycle/s_ALU_B         32'h00000009   32'h00000004   32'h00000004   32'h00000006   32'h00000009
/mips_single_cycle/s_ALU_A         32'h00000000   32'h00000000   32'h00000004   32'h00000000
Now        700 ns     ns   100 ns   200 ns   300 ns   400 ns   500 ns
Cursor 1   580 ns     580 ns
```

BEQ

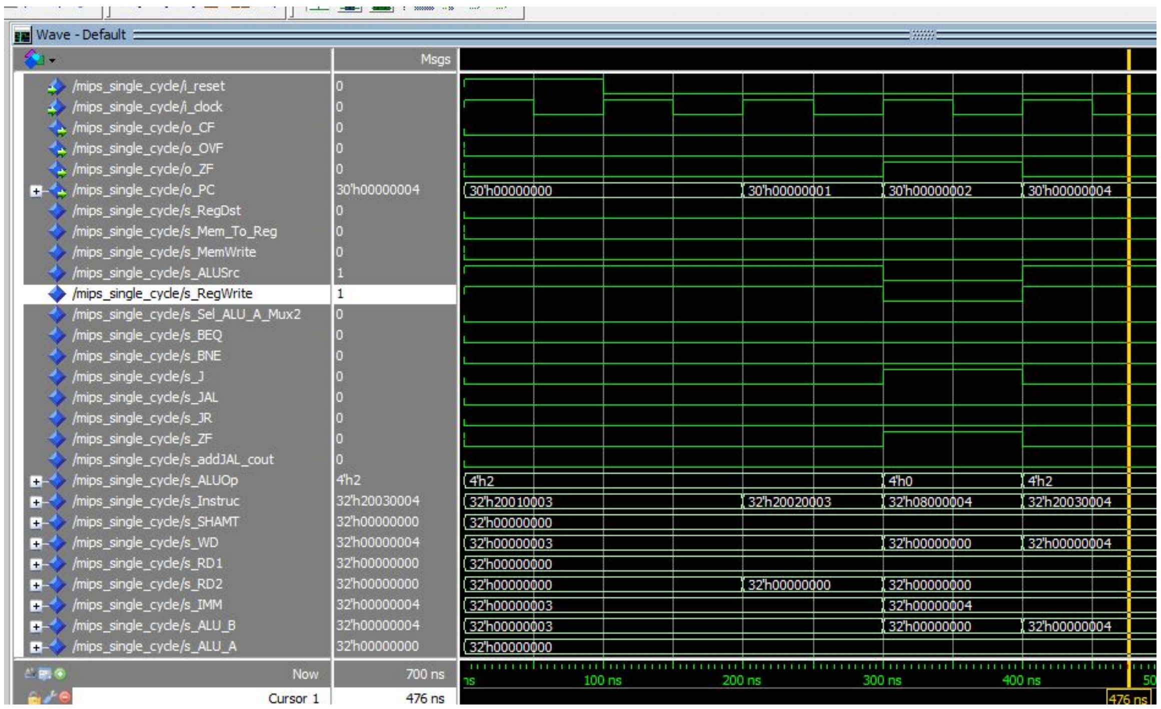
In the photo below, we branch to the add 5 instruction when $1 and $2 do equal each other.



```
Wave - Default
                                   Msgs
/mips_single_cycle/i_reset         0
/mips_single_cycle/i_clock         0
/mips_single_cycle/o_CF            0
/mips_single_cycle/o_OVF           0
/mips_single_cycle/o_ZF            0
/mips_single_cycle/o_PC            30'h00000004   30'h00000000   30'h00000001   30'h00000002   30'h00000004
/mips_single_cycle/s_RegDst        0
/mips_single_cycle/s_Mem_To_Reg    0
/mips_single_cycle/s_MemWrite      0
/mips_single_cycle/s_ALUSrc        1
/mips_single_cycle/s_RegWrite      1
/mips_single_cycle/s_Sel_ALU_A_Mux2 0
/mips_single_cycle/s_BEQ           0
/mips_single_cycle/s_BNE           0
/mips_single_cycle/s_J             0
/mips_single_cycle/s_JAL           0
/mips_single_cycle/s_JR            0
/mips_single_cycle/s_ZF            0
/mips_single_cycle/s_addJAL_cout   0
/mips_single_cycle/s_ALUOp         4'h2           4'h2           4'h6           4'h2
/mips_single_cycle/s_Instruc       32'h20030009   32'h20010004   32'h20020004   32'h10220001   32'h20030009
/mips_single_cycle/s_SHAMT         32'h00000000   32'h00000000
/mips_single_cycle/s_WD            32'h00000009   32'h00000004   32'h00000000   32'h00000009
/mips_single_cycle/s_RD1           32'h00000000   32'h00000000   32'h00000004   32'h00000000
/mips_single_cycle/s_RD2           32'h00000000   32'h00000000   32'h00000000   32'h00000004   32'h00000000
/mips_single_cycle/s_IMM           32'h00000009   32'h00000004   32'h00000001   32'h00000009
/mips_single_cycle/s_ALU_B         32'h00000009   32'h00000004   32'h00000004   32'h00000009
/mips_single_cycle/s_ALU_A         32'h00000000   32'h00000000   32'h00000004   32'h00000000
Now        700 ns     ns   100 ns   200 ns   300 ns   400 ns   500
Cursor 1   484 ns     484 ns
```

J

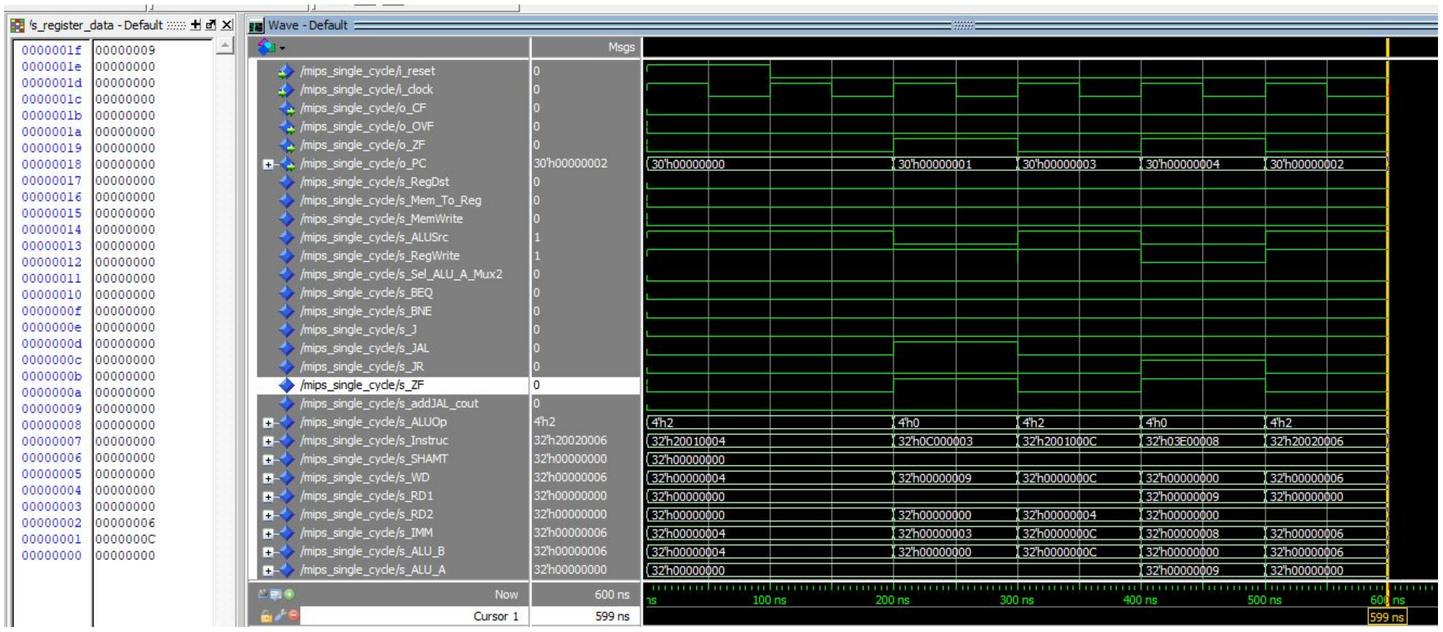
In the photo below, we jump to a label.



	Msgs
/mips_single_cycle/i_reset	0
/mips_single_cycle/i_clock	0
/mips_single_cycle/o_CF	0
/mips_single_cycle/o_OVF	0
/mips_single_cycle/o_ZF	0
/mips_single_cycle/o_PC	30'h00000004
/mips_single_cycle/s_RegDst	0
/mips_single_cycle/s_Mem_To_Reg	0
/mips_single_cycle/s_MemWrite	0
/mips_single_cycle/s_ALUSrc	1
/mips_single_cycle/s_RegWrite	1
/mips_single_cycle/s_Sel_ALU_A_Mux2	0
/mips_single_cycle/s_BEQ	0
/mips_single_cycle/s_BNE	0
/mips_single_cycle/s_J	0
/mips_single_cycle/s_JAL	0
/mips_single_cycle/s_JR	0
/mips_single_cycle/s_ZF	0
/mips_single_cycle/s_addJAL_cout	0
/mips_single_cycle/s_ALUOp	4'h2
/mips_single_cycle/s_Instruc	32'h20030004
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000004
/mips_single_cycle/s_RD1	32'h00000000
/mips_single_cycle/s_RD2	32'h00000000
/mips_single_cycle/s_IMM	32'h00000004
/mips_single_cycle/s_ALU_B	32'h00000004
/mips_single_cycle/s_ALU_A	32'h00000000
Now	700 ns
Cursor 1	476 ns

JAL and JR

In the photo below, we jump to a label to do an add C then we return to complete the add 6. Note the values of PC changing as we jump around.

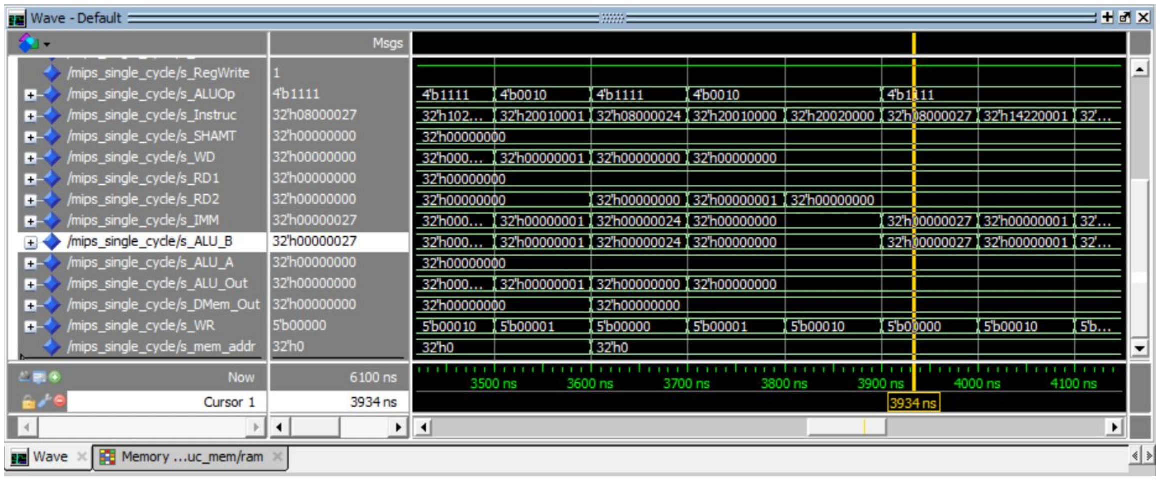


0000001f	00000009
0000001e	00000000
0000001d	00000000
0000001c	00000000
0000001b	00000000
0000001a	00000000
00000019	00000000
00000018	00000000
00000017	00000000
00000016	00000000
00000015	00000000
00000014	00000000
00000013	00000000
00000012	00000000
00000011	00000000
00000010	00000000
0000000f	00000000
0000000e	00000000
0000000d	00000000
0000000c	00000000
0000000b	00000000
0000000a	00000000
00000009	00000000
00000008	00000000
00000007	00000000
00000006	00000000
00000005	00000000
00000004	00000000
00000003	00000000
00000002	00000006
00000001	0000000C
00000000	00000000

	Msgs
/mips_single_cycle/i_reset	0
/mips_single_cycle/i_clock	0
/mips_single_cycle/o_CF	0
/mips_single_cycle/o_OVF	0
/mips_single_cycle/o_ZF	0
/mips_single_cycle/o_PC	30'h00000002
/mips_single_cycle/s_RegDst	0
/mips_single_cycle/s_Mem_To_Reg	0
/mips_single_cycle/s_MemWrite	0
/mips_single_cycle/s_ALUSrc	1
/mips_single_cycle/s_RegWrite	1
/mips_single_cycle/s_Sel_ALU_A_Mux2	0
/mips_single_cycle/s_BEQ	0
/mips_single_cycle/s_BNE	0
/mips_single_cycle/s_J	0
/mips_single_cycle/s_JAL	0
/mips_single_cycle/s_JR	0
/mips_single_cycle/s_ZF	0
/mips_single_cycle/s_addJAL_cout	0
/mips_single_cycle/s_ALUOp	4'h2
/mips_single_cycle/s_Instruc	32'h20020006
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000006
/mips_single_cycle/s_RD1	32'h00000000
/mips_single_cycle/s_RD2	32'h00000000
/mips_single_cycle/s_IMM	32'h00000006
/mips_single_cycle/s_ALU_B	32'h00000006
/mips_single_cycle/s_ALU_A	32'h00000000
Now	600 ns
Cursor 1	599 ns

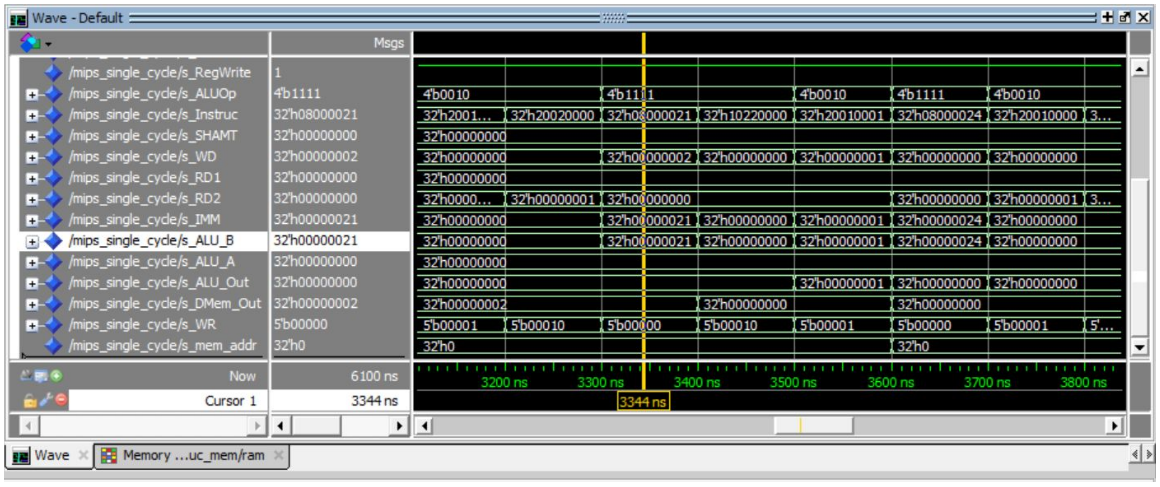
i. [Part (b.7)] In your writeup, show the Modelsim output for each of the following tests, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.

a. [Part (b.7.a)] Modify your application from Part (a.4) to include tests for each control flow instruction in addition to the data-handling instructions. Confirm your processor is completely functional.



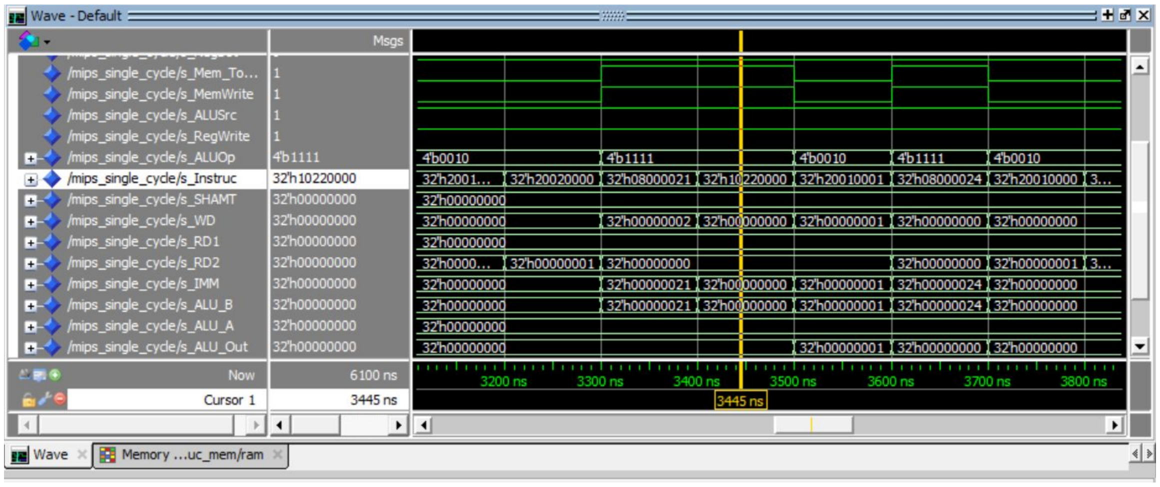
```
Wave - Default
                                    Msgs
/mips_single_cycle/s_RegWrite       1
/mips_single_cycle/s_ALUOp          4'b1111       4'b1111 | 4'b0010 | 4'b1111 | 4'b0010 | 4'b1111
/mips_single_cycle/s_Instruc        32'h08000027  32'h102... | 32'h20010001 | 32'h08000024 | 32'h20010000 | 32'h20020000 | 32'h08000027 | 32'h14220001 | 32'...
/mips_single_cycle/s_SHAMT          32'h00000000  32'h00000000
/mips_single_cycle/s_WD             32'h00000000  32'h000... | 32'h00000001 | 32'h00000000 | 32'h00000000
/mips_single_cycle/s_RD1            32'h00000000  32'h00000000
/mips_single_cycle/s_RD2            32'h00000000  32'h00000000 | 32'h00000000 | 32'h00000001 | 32'h00000000
/mips_single_cycle/s_IMM            32'h00000027  32'h000... | 32'h00000001 | 32'h00000024 | 32'h00000000 | 32'h00000027 | 32'h00000001 | 32'...
/mips_single_cycle/s_ALU_B          32'h00000027  32'h000... | 32'h00000001 | 32'h00000024 | 32'h00000000 | 32'h00000027 | 32'h00000001 | 32'...
/mips_single_cycle/s_ALU_A          32'h00000000  32'h00000000
/mips_single_cycle/s_ALU_Out        32'h00000000  32'h000... | 32'h00000001 | 32'h00000000 | 32'h00000000
/mips_single_cycle/s_DMem_Out       32'h00000000  32'h00000000 | 32'h00000000
/mips_single_cycle/s_WR             5'b00000      5'b00010 | 5'b00001 | 5'b00000 | 5'b00001 | 5'b00010 | 5'b00000 | 5'b00010 | 5'b...
/mips_single_cycle/s_mem_addr       32'h0         32'h0 | 32'h0
Now                                 6100 ns       3500 ns  3600 ns  3700 ns  3800 ns  3900 ns  4000 ns  4100 ns
Cursor 1                            3934 ns       3934 ns
Wave | Memory ...uc_mem/ram
```

bneTest - Two addi followed by a jump to bneCond (ALUOp 1111 is jump/branch)



```
Wave - Default
                                    Msgs
/mips_single_cycle/s_RegWrite       1
/mips_single_cycle/s_ALUOp          4'b1111       4'b0010 | 4'b1111 | 4'b0010 | 4'b1111 | 4'b0010
/mips_single_cycle/s_Instruc        32'h08000021  32'h2001... | 32'h20020000 | 32'h08000021 | 32'h10220000 | 32'h20010001 | 32'h08000024 | 32'h20010000 | 3...
/mips_single_cycle/s_SHAMT          32'h00000000  32'h00000000
/mips_single_cycle/s_WD             32'h00000002  32'h00000000 | 32'h00000002 | 32'h00000000 | 32'h00000001 | 32'h00000000 | 32'h00000000
/mips_single_cycle/s_RD1            32'h00000000  32'h00000000
/mips_single_cycle/s_RD2            32'h00000000  32'h0000... | 32'h00000001 | 32'h00000000 | 32'h00000000 | 32'h00000001 | 3...
/mips_single_cycle/s_IMM            32'h00000021  32'h00000000 | 32'h00000021 | 32'h00000000 | 32'h00000001 | 32'h00000024 | 32'h00000000
/mips_single_cycle/s_ALU_B          32'h00000021  32'h00000000 | 32'h00000021 | 32'h00000000 | 32'h00000001 | 32'h00000024 | 32'h00000000
/mips_single_cycle/s_ALU_A          32'h00000000  32'h00000000
/mips_single_cycle/s_ALU_Out        32'h00000000  32'h00000000 | 32'h00000001 | 32'h00000000 | 32'h00000000
/mips_single_cycle/s_DMem_Out       32'h00000002  32'h00000002 | 32'h00000000 | 32'h00000000
/mips_single_cycle/s_WR             5'b00000      5'b00001 | 5'b00010 | 5'b00000 | 5'b00010 | 5'b00001 | 5'b00000 | 5'b00001 | 5'...
/mips_single_cycle/s_mem_addr       32'h0         32'h0 | 32'h0
Now                                 6100 ns       3200 ns  3300 ns  3400 ns  3500 ns  3600 ns  3700 ns  3800 ns
Cursor 1                            3344 ns       3344 ns
Wave | Memory ...uc_mem/ram
```

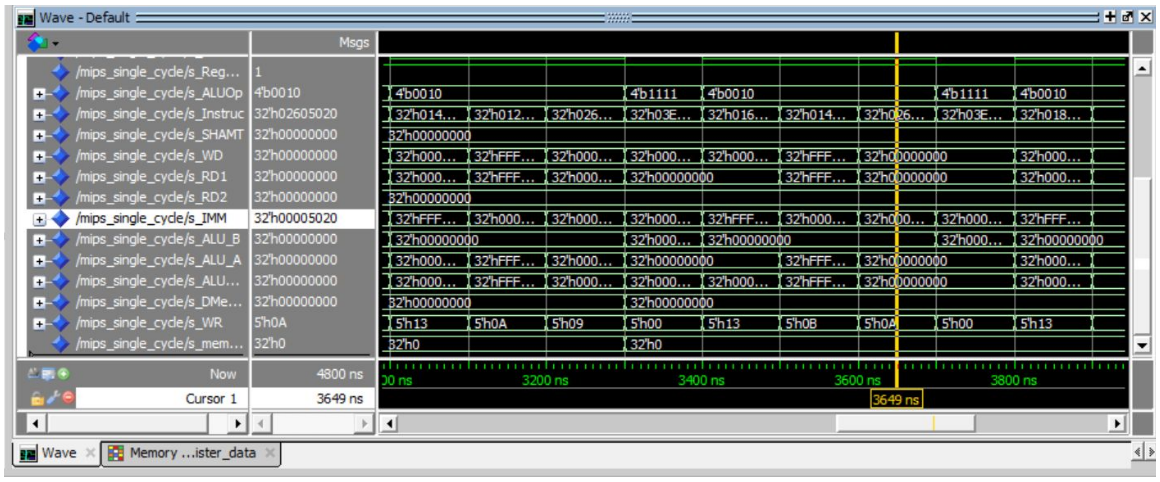
Jump to beq condition (ALUOp 1111 is jump/branch)



Wave - Default	Msgs
/mips_single_cycle/s_Mem_To...	1
/mips_single_cycle/s_MemWrite	1
/mips_single_cycle/s_ALUSrc	1
/mips_single_cycle/s_RegWrite	1
/mips_single_cycle/s_ALUOp	4'b1111
/mips_single_cycle/s_Instruc	32'h10220000
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000000
/mips_single_cycle/s_RD1	32'h00000000
/mips_single_cycle/s_RD2	32'h00000000
/mips_single_cycle/s_IMM	32'h00000000
/mips_single_cycle/s_ALU_B	32'h00000000
/mips_single_cycle/s_ALU_A	32'h00000000
/mips_single_cycle/s_ALU_Out	32'h00000000
Now	6100 ns
Cursor 1	3445 ns

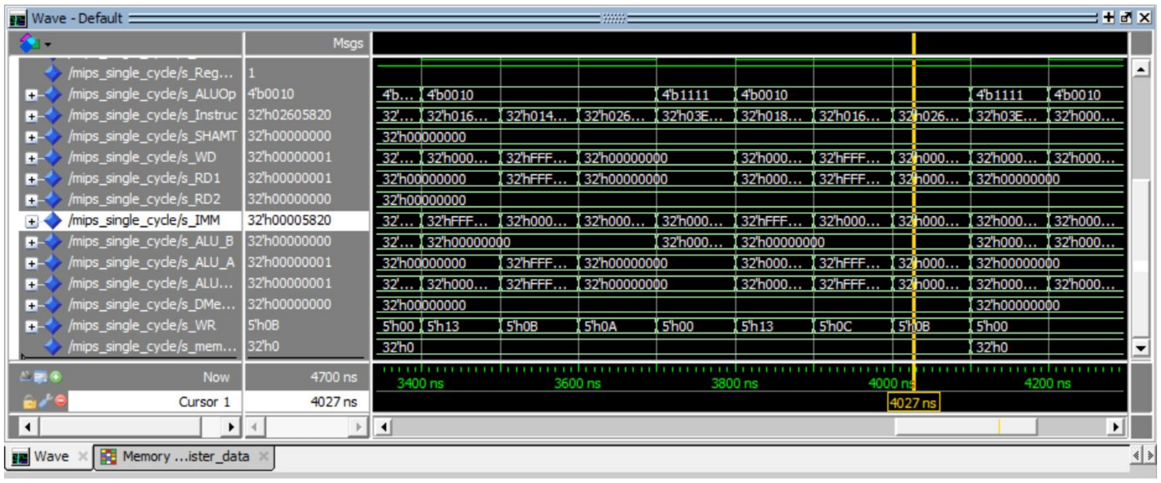
If $1 (0) == $2(0) branch to beqTrue;

b. [Part (b.7.b)] Create and test an application that sorts an array with N elements using the BubbleSort algorithm



Wave - Default	Msgs
/mips_single_cycle/s_Reg...	1
/mips_single_cycle/s_ALUOp	4'b0010
/mips_single_cycle/s_Instruc	32'h02605020
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000000
/mips_single_cycle/s_RD1	32'h00000000
/mips_single_cycle/s_RD2	32'h00000000
/mips_single_cycle/s_IMM	32'h00005020
/mips_single_cycle/s_ALU_B	32'h00000000
/mips_single_cycle/s_ALU_A	32'h00000000
/mips_single_cycle/s_ALU...	32'h00000000
/mips_single_cycle/s_DMe...	32'h00000000
/mips_single_cycle/s_WR	5'h0A
/mips_single_cycle/s_mem...	32'h0
Now	4800 ns
Cursor 1	3649 ns

swap element 2 and 3 (0 and -6)



Wave - Default	Msgs
/mips_single_cycle/s_Reg...	1
/mips_single_cycle/s_ALUOp	4'b0010
/mips_single_cycle/s_Instruc	32'h02605820
/mips_single_cycle/s_SHAMT	32'h00000000
/mips_single_cycle/s_WD	32'h00000001
/mips_single_cycle/s_RD1	32'h00000001
/mips_single_cycle/s_RD2	32'h00000000
/mips_single_cycle/s_IMM	32'h00005820
/mips_single_cycle/s_ALU_B	32'h00000000
/mips_single_cycle/s_ALU_A	32'h00000001
/mips_single_cycle/s_ALU...	32'h00000001
/mips_single_cycle/s_DMe...	32'h00000000
/mips_single_cycle/s_WR	5'h0B
/mips_single_cycle/s_mem...	32'h0
Now	4700 ns
Cursor 1	4027 ns

Swap element 3 and 4 (1 and -6)

c. [Part (b.7.c)] Create and test an application that sorts an array with N elements using the MergeSort algorithm

j. [Part (b.9)] Report the maximum frequency your processor can run at and determine what your critical path is.