EE 361L Implementation Notes for Subproject 1

You are to implement the PMIPSL0 so that it can execute the instructions:

- addi
- R-type (add, sub, and, or, slt)
- beq

Assume the controller of the PMIPSL0 works according to state diagram shown in Figure 1.

State 0 is the "instruction fetch" state. During this state, the PMIPSL0 will read in the instruction pointed to by the program counter (PC). At the same time, it will increment the PC by 2. The way the datapath is organized, instructions are always being loaded into the IF/ID pipeline register. Thus, the controller only needs to let the PC to be incremented by 2. In other words, it must *not* stall the PC. In addition, a bubble is sent into ID/EX pipeline register (and the datapath) so that the datapath doesn't do anything during this instruction fetch period.

State 1 is the "instruction decode" state. Note that this state is after the instruction fetch stage. Thus, we have the instruction and its opcode in the IF/ID pipeline register. The opcode is an input into the Controller. Given the opcode value, the Controller will determine what is the current instruction and then set its outputs so that the datapath will execute that instruction. These outputs are RegWrite, RegDst, ALUOp, ALUSrc, MemRead, MemWrite, and MemtoReg. These output signals are sent to the ID/EX pipeline register. Since we are in the process of executing an instruction, we want to stall the PC, i.e., hold its value.

States 2 and 3 correspond to the "execute" and "memory access" stages, respectively. During these states, the Controller basically waits while the datapath processes the instruction. The Controller will stall the PC and insert bubbles into the ID/EX pipeline register. Note that we do not need an additional state 4 for the write-back stage because the register file is triggered on a negative clock edge. Thus, the register file is written-back between state 3 ("memory access") and state 0 (the next "instruction fetch").

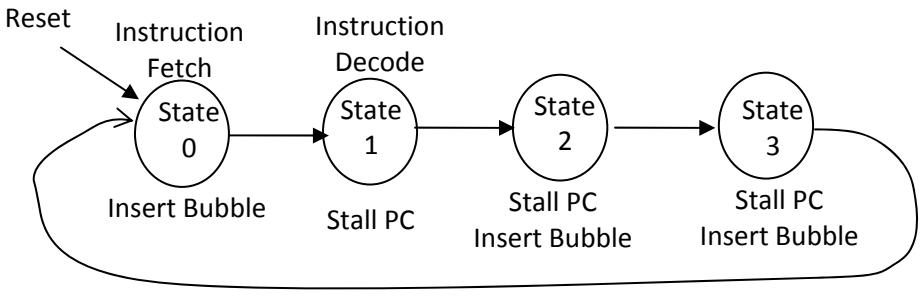

Figure 1. Controller State

Also remember that in the register file should be triggered on the negative clock edge, so make that change in the register file verilog module.

In addition, remember that the Controller should not have PCSrc as an output. Recall that I had made the mistake of having PCSrc as an output of the Controller in the Control.V file that I posted earlier. Note that PCSrc is the output of an AND circuit, which has inputs Branch and ALUZero from the EX/MEM pipeline register.

You are to complete PMIPSL0 processer so that it can run the program in IM1.V. The following program that multiplies 3 by 5.

```
Address         Instruction
0               L0: addi    $2,$0,3  # Initialize $2 with 3.  $2 is used as a counter
2                   add     $4,$0,$0 # Clear $4.  $4 will be used to compute the product
4               L1: beq     $2,$0,L0  # This loop will keep adding 5 to register $4 until counter $2 is 0
6                   addi    $4,$4,5
8                   addi    $2,$2,-1 # Decrement the counter
10                  beq     $0,$0,L1  # This is basically an unconditional branch
```

Your PMIPSL0 should run with test bench file testbenchSub1.V. Note that your project should have the testbenchSub1.V, MIPS-Parts.V , Control.V , PMPLS0.V, and IM1.V files. You can find the testbenchSub1.V and IM1.V files in the same folder as the current document.

The following is output of veriwell when running the testbench.

```
C1> .
IMem(PC,Instr),ALU(Output), Dmem(Addr) [Clock,Reset]

PC(   x,0000000000000000) ALU(   x) Dmem(   x) [0,1]          Reset
PC(   0,0110000100000011) ALU(   x) Dmem(   x) [1,1]          L0:  addi $2,$0,3
PC(   0,0110000100000011) ALU(   x) Dmem(   x) [0,0]
PC(   2,0000000001000011) ALU(   0) Dmem(   x) [1,0]
PC(   2,0000000001000011) ALU(   0) Dmem(   x) [0,0]
PC(   2,0000000001000011) ALU(   3) Dmem(   0) [1,0]
PC(   2,0000000001000011) ALU(   3) Dmem(   0) [0,0]
PC(   2,0000000001000011) ALU(   0) Dmem(   3) [1,0]
PC(   2,0000000001000011) ALU(   0) Dmem(   3) [0,0]
PC(   2,0000000001000011) ALU(   0) Dmem(   0) [1,0]            add $4,$0,$0
PC(   2,0000000001000011) ALU(   0) Dmem(   0) [0,0]
PC(   4,0100100001111101) ALU(   0) Dmem(   0) [1,0]
PC(   4,0100100001111101) ALU(   0) Dmem(   0) [0,0]
PC(   4,0100100001111101) ALU(   0) Dmem(   0) [1,0]
PC(   4,0100100001111101) ALU(   0) Dmem(   0) [0,0]
PC(   4,0100100001111101) ALU(   3) Dmem(   0) [1,0]
PC(   4,0100100001111101) ALU(   3) Dmem(   0) [0,0]
PC(   4,0100100001111101) ALU(   3) Dmem(   3) [1,0]          L1:  beq $2,$0,L0
PC(   4,0100100001111101) ALU(   3) Dmem(   3) [0,0]
PC(   6,0111001000000101) ALU(   3) Dmem(   3) [1,0]
PC(   6,0111001000000101) ALU(   3) Dmem(   3) [0,0]
PC(   6,0111001000000101) ALU(   3) Dmem(   3) [1,0]
PC(   6,0111001000000101) ALU(   3) Dmem(   3) [0,0]
```

```
PC(    6,0111001000000101) ALU(    0) Dmem(    3) [1,0]
PC(    6,0111001000000101) ALU(    0) Dmem(    3) [0,0]
PC(    6,0111001000000101) ALU(    0) Dmem(    0) [1,0]          addi    $4,$4,5
PC(    6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
PC(    8,0110100101111111) ALU(    0) Dmem(    0) [1,0]
PC(    8,0110100101111111) ALU(    0) Dmem(    0) [0,0]
PC(    8,0110100101111111) ALU(    5) Dmem(    0) [1,0]          product being updated
PC(    8,0110100101111111) ALU(    5) Dmem(    0) [0,0]
PC(    8,0110100101111111) ALU(    6) Dmem(    5) [1,0]
PC(    8,0110100101111111) ALU(    6) Dmem(    5) [0,0]
PC(    8,0110100101111111) ALU(    6) Dmem(    6) [1,0]          addi    $2,$2,-1
PC(    8,0110100101111111) ALU(    6) Dmem(    6) [0,0]
PC(   10,0100000001111100) ALU(    6) Dmem(    6) [1,0]
PC(   10,0100000001111100) ALU(    6) Dmem(    6) [0,0]
PC(   10,0100000001111100) ALU(    2) Dmem(    6) [1,0]
PC(   10,0100000001111100) ALU(    2) Dmem(    6) [0,0]
PC(   10,0100000001111100) ALU(    0) Dmem(    2) [1,0]
PC(   10,0100000001111100) ALU(    0) Dmem(    2) [0,0]
PC(   10,0100000001111100) ALU(    0) Dmem(    0) [1,0]          beq     $0,$0,L1
PC(   10,0100000001111100) ALU(    0) Dmem(    0) [0,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(   12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(    4,0100100001111101) ALU(    0) Dmem(    0) [1,0]          L1:  beq $2,$0,L0
PC(    4,0100100001111101) ALU(    0) Dmem(    0) [0,0]
PC(    6,0111001000000101) ALU(    0) Dmem(    0) [1,0]
PC(    6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
PC(    6,0111001000000101) ALU(    2) Dmem(    0) [1,0]
PC(    6,0111001000000101) ALU(    2) Dmem(    0) [0,0]
PC(    6,0111001000000101) ALU(   10) Dmem(    2) [1,0]
PC(    6,0111001000000101) ALU(   10) Dmem(    2) [0,0]
PC(    6,0111001000000101) ALU(   10) Dmem(   10) [1,0]          addi    $4,$4,5
PC(    6,0111001000000101) ALU(   10) Dmem(   10) [0,0]
PC(    8,0110100101111111) ALU(   10) Dmem(   10) [1,0]
PC(    8,0110100101111111) ALU(   10) Dmem(   10) [0,0]
PC(    8,0110100101111111) ALU(   10) Dmem(   10) [1,0]                product being updated
PC(    8,0110100101111111) ALU(   10) Dmem(   10) [0,0]
PC(    8,0110100101111111) ALU(    4) Dmem(   10) [1,0]
PC(    8,0110100101111111) ALU(    4) Dmem(   10) [0,0]
PC(    8,0110100101111111) ALU(    4) Dmem(    4) [1,0]          addi    $2,$2,-1
PC(    8,0110100101111111) ALU(    4) Dmem(    4) [0,0]
PC(   10,0100000001111100) ALU(    4) Dmem(    4) [1,0]
PC(   10,0100000001111100) ALU(    4) Dmem(    4) [0,0]
PC(   10,0100000001111100) ALU(    1) Dmem(    4) [1,0]
PC(   10,0100000001111100) ALU(    1) Dmem(    4) [0,0]
```

```
PC(  10,0100000001111100) ALU(    0) Dmem(    1) [1,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    1) [0,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [1,0]          beq     $0,$0,L1
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [1,0]          L1:  beq $2,$0,L0
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [0,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [1,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
PC(   6,0111001000000101) ALU(    1) Dmem(    0) [1,0]
PC(   6,0111001000000101) ALU(    1) Dmem(    0) [0,0]
PC(   6,0111001000000101) ALU(   20) Dmem(    1) [1,0]
PC(   6,0111001000000101) ALU(   20) Dmem(    1) [0,0]
PC(   6,0111001000000101) ALU(   20) Dmem(   20) [1,0]          addi    $4,$4,5
PC(   6,0111001000000101) ALU(   20) Dmem(   20) [0,0]
PC(   8,0110100101111111) ALU(   20) Dmem(   20) [1,0]
PC(   8,0110100101111111) ALU(   20) Dmem(   20) [0,0]
PC(   8,0110100101111111) ALU(   15) Dmem(   20) [1,0]                product being updated
PC(   8,0110100101111111) ALU(   15) Dmem(   20) [0,0]
PC(   8,0110100101111111) ALU(    2) Dmem(   15) [1,0]
PC(   8,0110100101111111) ALU(    2) Dmem(   15) [0,0]
PC(   8,0110100101111111) ALU(    2) Dmem(    2) [1,0]          addi    $2,$2,-1
PC(   8,0110100101111111) ALU(    2) Dmem(    2) [0,0]
PC(  10,0100000001111100) ALU(    2) Dmem(    2) [1,0]
PC(  10,0100000001111100) ALU(    2) Dmem(    2) [0,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    2) [1,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    2) [0,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [1,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [0,0]
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [1,0]          beq     $0,$0,L1
PC(  10,0100000001111100) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [1,0]
PC(  12,0000000000000000) ALU(    0) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [1,0]          L1:  beq $2,$0,L0
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [0,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [1,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [1,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
```

```
PC(   6,0111001000000101) ALU(   30) Dmem(    0) [1,0]
PC(   6,0111001000000101) ALU(   30) Dmem(    0) [0,0]
PC(   0,0110000100000011) ALU(   30) Dmem(   30) [1,0]          L0:  addi $2,$0,3
PC(   0,0110000100000011) ALU(   30) Dmem(   30) [0,0]
PC(   2,0000000001000011) ALU(   30) Dmem(   30) [1,0]
PC(   2,0000000001000011) ALU(   30) Dmem(   30) [0,0]
PC(   2,0000000001000011) ALU(    3) Dmem(   30) [1,0]
PC(   2,0000000001000011) ALU(    3) Dmem(   30) [0,0]
PC(   2,0000000001000011) ALU(    0) Dmem(    3) [1,0]
PC(   2,0000000001000011) ALU(    0) Dmem(    3) [0,0]          add $4,$0,$0
PC(   2,0000000001000011) ALU(    0) Dmem(    0) [1,0]
PC(   2,0000000001000011) ALU(    0) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [1,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [1,0]
PC(   4,0100100001111101) ALU(    0) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    3) Dmem(    0) [1,0]
PC(   4,0100100001111101) ALU(    3) Dmem(    0) [0,0]
PC(   4,0100100001111101) ALU(    3) Dmem(    3) [1,0]          L1:  beq $2,$0,L0
PC(   4,0100100001111101) ALU(    3) Dmem(    3) [0,0]
PC(   6,0111001000000101) ALU(    3) Dmem(    3) [1,0]
PC(   6,0111001000000101) ALU(    3) Dmem(    3) [0,0]
PC(   6,0111001000000101) ALU(    3) Dmem(    3) [1,0]
PC(   6,0111001000000101) ALU(    3) Dmem(    3) [0,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    3) [1,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    3) [0,0]
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [1,0]          addi    $4,$4,5
PC(   6,0111001000000101) ALU(    0) Dmem(    0) [0,0]
PC(   8,0110100101111111) ALU(    0) Dmem(    0) [1,0]
PC(   8,0110100101111111) ALU(    0) Dmem(    0) [0,0]
PC(   8,0110100101111111) ALU(    5) Dmem(    0) [1,0]          product being updated
PC(   8,0110100101111111) ALU(    5) Dmem(    0) [0,0]
PC(   8,0110100101111111) ALU(    6) Dmem(    5) [1,0]
Stop at simulation time 152
C1>
```