SANTA CLARA UNIVERSITY	ECEN 122 Winter 2024	Hoeseok Yang

Homework #7: RISC-V Forwarding/Hazard Detection Unit and Memory

Hierarchy Answer the following problems and upload your answers to Camino in pdf format. Posted: February 27, 2024, Due by: 9pm on March 6, 2024

1. [1.0 pts] Below is a part of a Verilog implementation of the forwarding unit.

```
1  module forwarding_unit(
2      input [4:0] EX_MEM_Rd,   /* Rd of the previous instr */
3      input [4:0] MEM_WB_Rd,   /* Rd of the 2nd previous instr */
4      input EX_MEM_RegWrite,   /* RegWrite for the previous instr */
5      input MEM_B_RegWrite,    /* RegWrite for the 2nd previous instr */
6      input 4:0] ID_EX_Rs1,    /* Rs1 of the current instr */
7      input [4:0] ID_EX_Rs2,   /* Rs2 of the current instr */
8      output [1:0] ForwardA,   /* MUX sel. for the Ist ALU operand */
9      output [1:0] ForwardB    /* MUX sel. for the 2nd ALU operand */
10 );
11
12   always@(*)
13   begin
14
15     if(EX_MEM_RegWrite == 1'b1) /* data hazard from the previous instr. */
16       if (EX MEM Rd != 5'b00000)
17         if (EX_MEM_Rd == ID EX_Rs1) ForwardA <= 2 b'10;
18         else ForwardA <= 2'b00;
19       else
20         ForwardA <= 2'b00;
21     else
22       ForwardA <= 2'b00;
23
24     if(MEM_W_RegWrite == 1'b1) /* data hazard from the 2nd previous instr. */
25       if(MEM WB Rd != 5'b00000)
26         if (MEM_WB_Rd == ID_EX_Rs1) ForwardA <= 2'b01;
27         else Forward <= 2'b00;
28       else
29         ForwardA <= 2'b00;
30     else
31       ForwardA <= 2'b00;
32
```

A. (0.5 pts) Briefly state the roles of line #15 and line #16, respectively.

Lines 15 and 16 test if we wrote to a register (that's not r0) as a part of the previous instruction that we want to use now.

B. (0.5 pts) This code may work well in most cases, but has a potential problem. What is the problem of this implementation?

If the prev destination reg was r0, the code won't forward. There also may be a missing use case, like the load use case.

2. [1 pts] Consider the following RISC-V assembly code:

```
…
add x1, x2, x3
addi x6, x0, 32
add x1, x1, x4
lw x5, 0(x6)
beq x1, x5, L1
…
```

A. (0.5 pts) If there is a forwarding and the branch condition is evaluated in the second stage (ID), how many stalls does this sequence cause?

2 stall (after lw and after add x1, x1, x4)

B. (0.5 pts) Based upon the above answer, reschedule the given sequence to minimize the number of stalls. What is the number of stalls of the rearranged code?

```
addi x6, x0, 32
add x1, x2, x3
lw x5, 0(x6)
add x1, x1, x4
beq x1, x5, L1
```

One stall at the end

3. [1 pts] Consider the following C code:

```
for(i=0;i<1000;i++)
    for(j=0;j<1000;j++)
        accum = accum + arr[j*1000+i];
```

A. (0.5 pts) Which of these variables has temporal locality? (check all that apply)

□ *i* □ *j* □ *accum* □ each element of *arr*

It's i, j, and accum

B. (0.5 pts) Rewrite the code to maximize the spatial locality while keeping the original behavior of the given code.

Invert i and j:

```
for(j=0;j<1000;j++)
        for(i=0;i<1000;i++)
            accum = accum + arr[j*1000+i];
```