## **Computer Sciences Department**

## **University of Wisconsin-Madison**

## **CS/ECE 552 – Introduction to Computer Architecture**

## **In-Class Exercise (02/18)**

**Answers to all questions should be uploaded on Canvas.**

1. [3 points] Consider the following MIPS assembly code:

|  |  |  |
| --- | --- | --- |
| **L0.** | **sw** | **$s1, 100($s3)** |
| **L1.** | **add** | **$s3, $s1, $s0** |
| **L2.** | **lw** | **$s2, 100($s3)** |
| **L3.** | **lw** | **$s3, 0($s2)** |
| **L4.** | **beq** | **$s3, $s2, 20** |

In the table below, list all the data dependences (RAW, WAW, WAR) and indicate whether a data hazard may occur when executing on the five-stage pipeline as described in the textbook. If more rows are needed, just add them manually. Include instruction line numbers for both the instruction that produces the value and instruction that need to use that value (e.g., L0, L1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Register** | **Instruction Line #** | **Type of Dependency** | **Data Hazard?** |
| $s1 | 0-1 | RAW | Y |
| $s3 | 0-1 | WAR | Y |
|  | 0-2 | WAW | Y |
|  | 0-3 | WAR | y |
|  | 1-2 | RAW | y |
|  | 1-3 | WAW | y |
|  | 1-4 | WAW | y |
|  | 2-3, 2-4 | WAR | y |
|  | 1-4 | WAW | y |
| $s2 | 2-3, 2-4 | RAW | y |
|  |  |  |  |

2. [3 points] Consider two different machines. The first has a single cycle datapath (i.e., a single stage, non-pipelined machine) with a cycle time of 5 ns. The second is a pipelined machine with 5 pipeline stages and a cycle time of 1ns.

(a) [1 point] What is the speedup of the pipelined machine versus the single cycle machine assuming there are no stalls?

5:1

1. [1 point] What is the speedup of the pipelined machine versus the single cycle machine if the pipeline stalls 1 cycle for 25% of the instructions?

1:4

1. [1 point] Now consider a 4-stage pipeline machine with a cycle time of 1.1 ns. Again, assuming no stalls, is this implementation faster or slower than the original 5 stage pipeline? Explain your answer.

4.4<5

3. [2 points] Consider the following MIPS assembly code, running on the 5-stage pipeline shown below:

|  |  |  |
| --- | --- | --- |
| **L0:** | **addi** | **$s1, $zero, 6** |
| **L1:** | **lw** | **$s3, 0($s4)** |
| **L2:** | **ori** | **$s2, $zero, 7** |
| **L3:** | **sw** | **$s0, 4($s4)** |
| **L4:** | **add** | **$s2, $s1, $s1** |

f0630

The following pipeline diagram shows which stage each instruction is performing at each cycle: F=Fetch (IF), D=Decode (ID), X=Execute (EX), M=Memory (MEM) or W=Writeback (WB).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Clock Cycle** | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| L0 | F | D | X | M | W |  |  |  |  |
| L1 |  | F | D | X | M | W |  |  |  |
| L2 |  |  | F | D | X | M | W |  |  |
| L3 |  |  |  | F | D | X | M | W |  |
| L4 |  |  |  |  | F | D | X | M | W |

1. [1 point] Specify all cycles when **ID/EX.ALUSrc** (i.e., the ALUSrc control bit stored in the ID/EX pipeline register) is set to 1.

**3,4,5,6**

1. [1 point] Assume that in the beginning of cycle 4, a fault occurs that forces **ID/EX.RegDst** (i.e., the RegDst control bit stored in the ID/EX pipeline register) to get stuck at 0 for the rest of the program. What are the contents of registers $s1 and $s2 at the end of the program due to this fault?

S1-12, s2 - 7

4. [2 points] Assume the five-stage pipeline with a write-before-read RF (i.e., RF bypassing) but without support for EX-stage forwarding (i.e., EX-to-EX forwarding) nor MEM-stage forwarding (i.e., MEM-to-EX forwarding). Given the following code sequence, how many cycles will it take to complete and which instructions will need to stall?

|  |  |
| --- | --- |
| **1. or** | **$s5, $s2, $s3** |
| **2. add** | **$s2, $s2, $s3** |
| **3. slt** | **$s1, $s5, $zero** |
| **4. lw** | **$s2, 0($s1)** |
| **5. lw** | **$s3, 4($s2)** |
| **6. add** | **$s4, $s3, $s3** |
| **7. sw** | **$s2, 0($s5)** |
| **8. and** | **$s5, $s3, $s4** |
| **9. sw** | **$s5, 0($s1)** |
| **10. beq** | **$s2, $zero, 24** |

23