CS433: Computer Architecture – Fall 2021
Homework 2
Total Points: 33 points
All students should solve all problems
Due Date: September 28, 2021 at 10:00 pm CT
(See course information slides for more details)

Directions:

- **All students must write and sign the following statement at the end of their homework submission. "I have read the honor code for this class in the course information handout and have done this homework in conformance with that code. I understand fully the penalty for violating the honor code policies for this class." No credit will be given for a submission that does not contain this signed statement.**
- **On top of the first page of your homework solution, please write your name and NETID, your partner's name and NETID, and whether you are an undergrad or grad student.**
- **Name your homework solution file as *firstname_lastname_hw2.pdf***
- **Please show all work that you used to arrive at your answer. Answers without justification will not receive credit. Errors in numerical calculations will not be penalized. Cascading errors will usually be penalized only once.**
- **See course information slides for more details.**

Problem 1 [5 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 15 ns. The second is a pipelined machine with 5 pipeline stages and a cycle time of 3ns.

Part (A) [1 point]

What is the speedup of the pipelined machine versus the single cycle machine assuming there are no stalls?

Part (B) [2 points]

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

Part (C) [2 points]

Now consider a 4 stage pipeline machine with a cycle time of 3.1 ns. Again, assuming no stalls, is this implementation faster or slower than the original 5 stage pipeline? Explain your answer.

Problem 2 [4 points]

Consider two different 5-stage pipeline machines (IF ID EX MEM WB). The first machine resolves branches in the ID stage, uses one branch delay slot, and can fill 80% of the delay slots with useful instructions. The second machine resolves branches in the EX stage and uses a predict-not-taken scheme. Assume that the cycle times of the machines are identical. Given that 35% of the instructions are branches, 25% of branches are taken, and that stalls are due to branches alone, which machine is faster? To get any credit, you must justify your answer.

Problem 3 [10 points]

Consider the following loop.

$loop$:

1. $ADDI\ R2,\ R2,\ \#1$
2. $LD\ R4,\ 0(R3)$
3. $LD\ R5,\ 4(R3)$
4. $ADD\ R6,\ R4,\ R5$
5. $MUL\ R4,\ R6,\ R7$
6. $SUBI\ R3,\ R3,\ \#8$
7. $BNEZ\ R2,\ loop$
8. $ADD\ R11,\ R12,\ R13$

Part (A) [4 points]

Identify all data dependencies (potential data hazards) in the given code snippet. Assume the loop takes exactly one iteration to complete. Specify if the data dependence is RAW, WAW or WAR.

Part (B) [2 points]

Assume a 5-stage pipeline (IF ID EX MEM WB) without any forwarding or bypassing hardware, but with support for a register read and write in the same cycle. Also assume that branches are resolved in the ID stage and handled by stalling the pipeline. All stages take 1 cycle. Again, the loop takes one iteration to complete. Which dependencies from part (a) cause stalls? How many cycles does the loop take to execute?

Part (C) [2 points]

Assume that the pipeline now supports full forwarding and bypassing. Furthermore, branches are handled as predicted-not-taken. As before, the loop takes one iteration to complete. Which dependencies from part (a) still cause stalls and why? How many cycles does the loop take to execute now?

Part (D) [2 points]

If the pipeline from part (c) instead uses a branch delay slot, how would you schedule the instructions in the loop to minimize stalls? For this part, assume the *loop takes multiple iterations to complete*. Explain your answer.

Problem 4 [14 points]

For this problem, we will explore a pipeline for a register-memory architecture. The architecture has two instruction formats: a register-register format and a register-memory format. In the register-memory format, one of the operands for an ALU instruction could come from memory.

There is a single memory-addressing mode (offset + base register). The only non-branch register-memory instructions available have the format:

Op Rdest, Rsrc1, Rsrc2

or

Op Rdest, Rsrc1, MEM

where Op is one of the following: Add, Subtract, And, Or, Load (in which case Rsrc1 is ignored), or Store. Rsrc1, Rsrc2, and Rdest are registers. MEM is a (base register, offset) pair.

Branches compare two registers and, depending on the outcome of the comparison, move to a target address. The target address can be specified as a PC-relative offset or in a register (with no offset). Assume that the pipeline structure of the machine is as follows:

IF RF ALU1 MEM ALU2 WB

The first ALU stage is used for effective address calculation for memory references and branches. The second ALU stage is used for operations and branch comparison. RF is both decode and register-fetch stage. Assume that when a register read and a register write of the same register occur in the same cycle, the write data is forwarded.

Part (A) [4 points]

Find the number of adders, counting any adder or incrementor, needed to minimize the number of structural hazards. Justify why you need this number of adders.

Part (B) [4 points]

Find the number of register read and write ports and memory read and write ports needed to minimize the number of structural hazards. Justify why you need this number of ports for the register file and memory.

Part (C) [3 points]

Will data forwarding from the ALU2 stage to any of ALU1, MEM, or ALU2 stages reduce or avoid stalls? Explain your answer for each stage.

Part (D) [3 points]

Will data forwarding from the MEM stage to any of ALU1, MEM, or ALU2 stages reduce or avoid stalls? Explain your answer for each stage.