Homework 6 – Pipelining and Hazards, Exceptions

Out: 3.26.21
Due: 4.5.21

1. [Pipelining and Hazards]
   In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
sw    r16,12(r6)
lw    r16,8(r6)
beq   r5,r4,Label    # Assume r5!=r4
add   r5,r1,r4
slt   r5,r15,r4
```

   Assume that individual pipeline stages have the following latencies:
   IF – 200ps, ID – 120ps, EX – 150ps, MEM – 190ps, WB – 100ps

   1) For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have one memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage pipelining that only has one memory? We have seen that data hazards can be eliminated by adding nops to the code. Can you do the same with this structural hazard? Why?

   2) For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we change load/store instructions to use a register (without an offset) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only 4 stages. Change this code to accommodate this changed ISA. Assuming this change does not affect clock cycle time, what speedup is achieved in this instruction sequence?

   3) Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the exaction where branch outcomes are determined in the EX stage?

   4) Given these pipeline stage latencies, repeat the speedup calculation from part (2), but take into account the (possible) change in clock cycle time. When EX and MEM are done in a single stage, most of their work can be done in parallel. As a result, the resulting EX/MEM stage has a latency that is the larger of the original

two, plus 20 ps needed for the work that could not be done in parallel.

5) Given these pipeline stage latencies, repeat the speedup calculation from part (3), taking into account the (possible) change in clock cycle time. Assume that the latency of the ID stage increased by 50% and the latency of the EX stage decreased by 10ps when branch outcomes resolution is moved from EX to ID.

2. [Pipelining and Hazards]
Assume that of all of the instructions executed in the MIPS pipelined datapath, the following fraction of these instructions have a particular type of RAW data dependence. The type of RAW data dependence is identified by the stage that produces the result (EX or MEM0 and the instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). Also assume that internal register file bypass is utilized, so no "EX to 3rd" or "MEM to 3rd" dependences are encountered.

EX to 1st only – 5%, MEM to 1st only – 20%, EX to 2nd only – 5%, MEM to 2nd only – 10%, EX to 1st and MEM to 2nd – 10%, Other RAW dependences – 10%

Assume the following latencies for individual pipeline stages. For the EX stage, latencies are given separately for a processor without forwarding and for a processor with different kinds of forwarding. You may assume that base CPI of the processor is 1 if there are no hazards.

IF – 150ps, ID – 100ps, EX (no FW) – 120ps, EX (full FW) – 150ps,
EX (FW from EX/MEM only) – 140ps, EX (FW from MEM/WB only) – 130ps,
WB – 100ps

1) If we use no forwarding, what fraction of cycles are we stalling due to data hazards?

2) If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we stalling due to data hazards?

3. [Pipelining and Hazards]
This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

```
add   r5,r2,r1
lw    r3,4(r5)
lw    r2,0(r2)
or    r3,r5,r3
sw    r3,0(r5)
```

1) If there is no forwarding or hazard detection, insert nops to ensure correct execution.

2) Repeat pat (1) but now use nops only when a hazard cannot be avoided by changing or rearranging these instructions. You can assume register R7 can be used to hold temporary values in your modified code.

3) If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when this code executes?

4) If there is forwarding, for the first five cycles during the execution of this code, specify which signals are asserted in each cycle by the hazard detection and forwarding units.

4. [Pipelining and Hazards]
The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how uch time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-Type – 40%, BEQ – 25%, JMP – 5%, LW – 25%, SW – 5%

Also, assume the following branch predictor accuracies:

Always-Taken – 45%, Always-Not-Taken – 55%, 2-Bit – 85%

1) Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mipredicted branches with the always-taken predictor? Assume that branch outcomes are determined I the EX stage, that there are no data hazards, and that no delay slots are used.

2) Repeat part (1) for the always-not-taken predictor.

3) Repeat part (1) for the 2-bit predictor.

5. [Pipelining and Hazards]
This exercise examines the accuracy of various branch prediction for the following repeating pattern (e.g., ion a loop) of branch outcomes: T, NT, T, T, NT

1) What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

2) What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left stage (predict not

taken)?

3) What is the accuracy of the two-bit predictor if this pattern is repeated forever?

6. [Exceptions]
This exercise explores how exception handling affects pipeline design. Refer to the following two instructions:

Instruction 1: BNE R1,R2, Label
Instruction 2: LW R1,0(R1)

1) Which exceptions can each of these instructions trigger? For each of these exceptions, specify the pipeline stage in which it is detected.

2) If there is a separate handler address for each exception, show how the pipeline organization must be changed to be able to handle this exception. You can assume that the address of these handlers are known when the processor is designed.

3) If the second instruction is fetched right after the first instruction, describe what happens in the pipeline when the first instruction causes the first exception you listed in part (1). Show the pipeline execution diagram from the time the first instruction is fetched until the time the first instruction of the exception handler is completed.