Homework Assignment 9

CS/ECE 3810: Computer Organization
Nov 09, 2020

Pipelining and Hazards

Due Date: Nov 16, 2020
(100 points)

Important Notes:

- Solutions turned in must be your own. Please, mention references (if any) at the end of each question. *Please refrain from cheating.*
- All solutions must be accompanied by the equations used/logic/intermediate steps. Writing only the final answer will receive **zero** credits.
- Partial score of every question is dedicated to each correct final answer provided by you. Please ensure both your equation/logic and final answer are correct. Moreover, you are expected to provide explanation for your solutions.
- All units must be mentioned wherever required.
- Late submissions **(after 11:59PM on 11/16/2020)** will not be accepted.
- We encourage all solutions to be typed in for which you could use software programs like LaTeX, Microsoft Word etc. If you submit handwritten solutions, they must be readable by the TAs to receive credits.
- Submit a .pdf file containing your answers for questions: 1-4.

Single-Cycle Processor. A single-cycle processor is a processor that completes processing each instruction in a single clock cycle.
Refer to slides #9-16 of the 'Single-Cycle and Pipelined CPU' lecture.

Question 1. Consider a single-cycle MIPS processor that takes 40 nanoseconds(ns) (CPU Time) to process 10 instructions. Assume that these instructions do not have any data dependencies, hazards, or stalls.

Question 1A. Compute the cycles-per-instruction (CPI) of the processor. **(5 points)**

CPU Time $= 10\ instructions * 4\ ns * x$ where x is the CPI. Thus $x = CPI = 1\ cycle$.

Question 1B. Compute the cycle time (CT) of the processor in ns.**(5 points)**

Cycle time = $40\ ns/10\ instructions = 4\ ns$

Pipelined Processor. Pipelining increases the number of simultaneously processed instructions and the rate at which instructions are started and completed. Pipelining does not reduce the time it takes to complete an individual instruction, also called the instruction processing latency.
Refer to slide #16 of the 'Single-Cycle and Pipelined CPU' lecture and IVC week 10.

Question 2. In this question, consider a 5-stage pipelined architecture similar to the one discussed in the class. The 5 stages are fetch (IF), decode (ID), execute (EX), memory (MEM), and write back (WB). The processor executes a program with 12 instructions. The below table mentions the latency of each stage in nanoseconds (ns).

Stage	Fetch(IF)	Decode(ID)	Execute(EX)	Memory(MEM)	Write-Back(WB)
Latency(ns)	3	2	3.5	10	4

Question 2.A. What is the cycle time of this pipelined processor? State the reason for your answer. **(10 points)**

CT = 10 ns. This is because cycle time for a pipelined processor is calculated using the maximum delay due to any stage. Delay due to register may also be added to the CT but none is specified in the question.

Question 2.B. Consider an ideal pipelining for this 5-stage processor with the cycle time calculated in part 2.A, compute the CPU time. **(15 points)**

CPU Time = 10 ns * 12 instructions = 120 ns. Since we're considering an ideal pipelining for this processor, no time needs to be added to this total.

Question 2.C. Compute the IPC (Instructions per cycle) for this processor considering that the 5^{th} instruction requires 3 stall cycles and the 7^{th} and 9^{th} instructions, each requires 1 stall cycle. Round up your answers to two decimal places. **(15 points)**

There are 12 instructions to be executed. Using the Cycle Time from part (a), that takes 120 ns. The 3 stall cycles for the 5th instruction and the 1 stall cycle for the 7th and 9th instructions bring the total number of cycles to 17. So we have 12 instructions taking a total of 17 cycles. Thus, IPC = 0.71.

Data Hazards. As mentioned in the class, there are three types of data hazards as follows:

RAW (Read after write)
add $1, $2, $3
add $4, $1, $0

WAW (Write after write)
add $1, $2, $3
add $1, $4, $6

WAR (Write after read)

lw $t1, 0($t2)
add $t2, $t3, $t4

Refer to slides #12-24 of the 'Pipeline Hazards I' lecture and IVC week 10.
Consider the following snippet:
1: add $1,$2,$3
2: add $5,$1,$0
3: add $3,$1,$0
4: sub $4,$1,$3
We have identified a data hazard for the above snippet in the following format:
(<instruction number where hazard occurs>, <instruction number where hazard occurs>, <type of hazard>, <register name>).
(1, 2, RAW, $1)

Question 3. Identify all data hazards in the given snippets in the format above:

A. **(10 points)**
1: lw $t2, 0($t2)
2: add $t2, $t2, $t3
3: lw $t3, 0($t2)
4: sw $t2, 0($t4)

(1, 2, WAR, $t2)

(1, 2, RAW, $t2)

(1, 3, WAR, $t2)

(1, 4, WAR, $t2)

(2, 2, WAR, $t2)

(2, 3, RAW, $t2)

(2, 3, RAW, $t3)

(2, 4, WAW, $t2)

(3, 4, WAR, $t2)

B. **(20 points)**
1: lw $f6, 20($gp)
2: lw $f2, 28($gp)
3: mult $f0, $f2, $f4
4: sw $f3, 32($gp)
5: sub $f8, $f6, $f3
6: div $f10, $f0, $f6
7: sub $f6, $f8, $f2

8: sw $f8, 50($gp)

(1, 5, RAW, $f6)

(1, 6, RAW, $f6)

(1, 7, WAW, $f6)

(2, 3, RAW, $f2)

(2, 7, RAW, $f2)

(3, 6, RAW, $f0)

(4, 5, RAW, $f4)

(5, 7, RAW, $f8)

(5, 8, WAW, $f8)

(6, 6, WAR, $f10)

(7, 8, WAR, $f8)

Stalls. In the design of pipelined processor cores, a pipeline stall is a delay in processing of an instruction in order to resolve a hazard. Consider the below snippet where a stall is introduced to resolve a control hazard:

1: for: beq $0, $1, next
stall
2: add $2, $2, $1

In the above example, there is one stall cycle between instructions 1 and 2, represented as (1, 2, 1).
General format is: (instruction A, instruction B, n). There are n stall cycles between instruction #A and instruction #B

Question 4. Consider a 5-stage pipeline with separate memory for data and instruction and a register file with 2 read ports and 1 write port performing write in the first half of the cycle and reads in the second half. As a result, the pipeline does not require any stalls for structural hazards. The pipeline does not have any register bypass and ALU forwarding/bypassing mechanisms. Therefore, stall cycles are necessary to mitigate data hazards. The outcome of each branch is sent to the fetch stage from the execute stage. (Hint: One stall cycle is necessary for control hazards). Identify all data and control hazards in the following snippet. Indicate where you need to introduce stalls in the code. **(20 points)**
1: addi $t0, $zero, 130
2: bne $t0, $zero, next
3: addi $s0, $s0, 13
4: next: add $s1, $s1, $t0

5: addi $t0, $t0, -1

Data hazards:

(1, 2, WAW, $t0)

(1, 4, RAW, $t0)

(1, 5, WAW, $t0)

(1, 5, RAW, $t0)

(2, 4, RAW, $t0)

(2, 5, RAW, $t0)

(2, 5, WAW, $t0)

(3, 3, WAR, $s0)

(4, 4, WAR, $s1)

(5, 5, WAR, $t0)

Control hazards:

(1, 4, 1)

(2, 4, 1)