CSCE 5610 – Computer System Architecture

Assignment 2

Due: 10/16/2024 11:59PM on Canvas

(100 points)

Question 1 (20 points). In this problem, we will explore how deepening the pipeline affects performance in two ways: faster clock cycles and increased stalls due to data and control hazards. Assume that the original machine is a 5-stage pipeline with a 1ns clock cycle. The second machine is a 12-stage pipeline with a 0.6ns clock cycle. The 5-stage pipeline experiences a stall due to a data hazard every five instructions, whereas the 12-stage pipeline experiences three stalls every eight instructions. In addition, branches constitute 20% of the instructions, and the misprediction rate for both machines is 5%.

a). What is the speedup of the 12-stage pipeline over the 5-stage pipeline, taking into only data hazards?

b). If the branch mispredict penalty for the first machine is 2 cycles but the second machine is 5 cycles, what are the CPIs of each, taking into account the stalls due to branch misprediction?

Answer:

a). (10 points)

$$Execution\ Time\ =\ I\ *\ CPI\ *\ Cycle\ Time, Speedup\ =\ \frac{I * \frac{6}{5} * 1}{I * \frac{11}{8} * 0.6} =\ 1.45$$ (10 points)

b).

$CPI_{5-stg} = \frac{6}{5} + 0.2 * 0.05 * 2 = 1.22$ (5 points), $CPI_{12-stg} = \frac{11}{8} + 0.2 * 0.05 * 5 = 1.425$ (5 points)

Question 2 (10 points). **Given the program below,**

```
      li $t1, 0
      li $t2, 10
L:    beq $t1, $t2, Exit
      addi $t1, $t1, 1
      j     L
Exit:
```

1). How many static instructions in the program? How many dynamic instructions in the program? ["Exit" is a label not an instruction]

Answer: static instruction counts: 5 (2 points)

dynamic instruction counts: 33 (2 points)

2). Compare the performance of the following two processors:

P1: 1GHz, with a CPI of 1.2 for the given program

P2: 2GHz, with a CPI of 1 for the same program

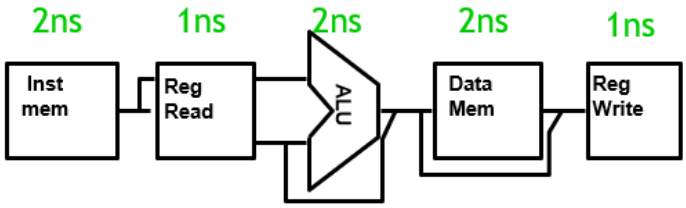
Calculate the CPU execution time of these two processors on the given program. Which processor is faster in executing the given program? How much faster?

Answer: $CPU\ time_{P1} = clock\ cycles * cycle\ time = Instruction\ count * CPI * cycle\ time = 33 * 1.2 * \frac{1}{1*10^9} = 39.6ns$ (2 points)

$CPU\ time_{P2} = clock\ cycles * cycle\ time = Instruction\ count * CPI * cycle\ time = 33 * 1 * \frac{1}{2*10^9} = 16.5ns$ (2 points)

P2 is faster than P1, by $\frac{39.6}{16.5} = 2.4$ times. (2 points)

Question 3 (20 points). Assuming the following functional unit latencies:



1). What is the cycle time of a single-cycle implementation? What is the cycle time of a pipelined implementation?

Answer: single-cycle implementation: 8ns (2.5 points). Pipelined implementation: 2ns (2.5 points).

Given the program below:

```
lw $a2, 0($a2)
subi $t0, $t1, 1
addi $v0, $0, 0
add $t0, $0, $0
add $t4, $a0, $a1
```

2). The assembly code is executed using a 5-stage MIPS processor. Plot the pipeline diagram to show the execution of these five instructions. [hint: The instruction sequence is shown vertically, from top to bottom. Clock cycles are shown horizontally, from left to right. Each instruction is divided into its component stages.].

Answer (1 points/each, be careful about the NOP stages):

	1	2	3	4	5	6	7	8	9
lw $a2, 0($a2)	IF	ID	EXE	MEM	WB				
subi $t0, $t1,1		IF	ID	EXE	NOP	NOP			
addi $v0, $0, 0			IF	ID	EXE	NOP	WB		
add $t0, $0, $0				IF	ID	EXE	NOP	WB	
add $t4, $a0, $a1					IF	ID	EXE	NOP	WB

[If they use MEM stage instead of NOP, also correct]

3). If the given program is run on the single-cycle implementation, how many cycle does it take to execute these five instructions? How long does it take to complete execution (ns)?

Answer: 5 cycles. 5*8ns=40ns. (5 points)

4). If the given grogram is run on the pipelined implementation, how many cycle does it take to execute these five instructions? How long does it take to complete execution (ns)?

Answer: 4+5=9 cycles. 9*2=18ns. (5 points)

Question 4 (30 points). **A MIPS assembly code is shown below. The assembly code is executed using a 5-stage MIPS processor and we can actually decide the branch a little earlier, in ID instead of EX. Answer the following questions:**

```
Loop: sub $19, $19, $20
      addi $9, $20, 5
      sw $9, 0($21)
      addi $10, $19, 2
      sll $10, $10, $3
      add $10, $10, $21
      addi $11, $20, 1
      sll $11, $11, 3
      add $11, $11, $21
      lw $11, 0($11)
      sub $9, $9, $11
      sw $9, 0($10)
      beq $9, $0, Loop
      addi $9, $12, 2
```

a). **Identify all the stalls** (which two instructions or which register are the cause of the stall) and the **total number of stalls** in the above assembly code. Assume there is no forwarding, no branch prediction, and no duplicate copies of instruction memory or data memory.

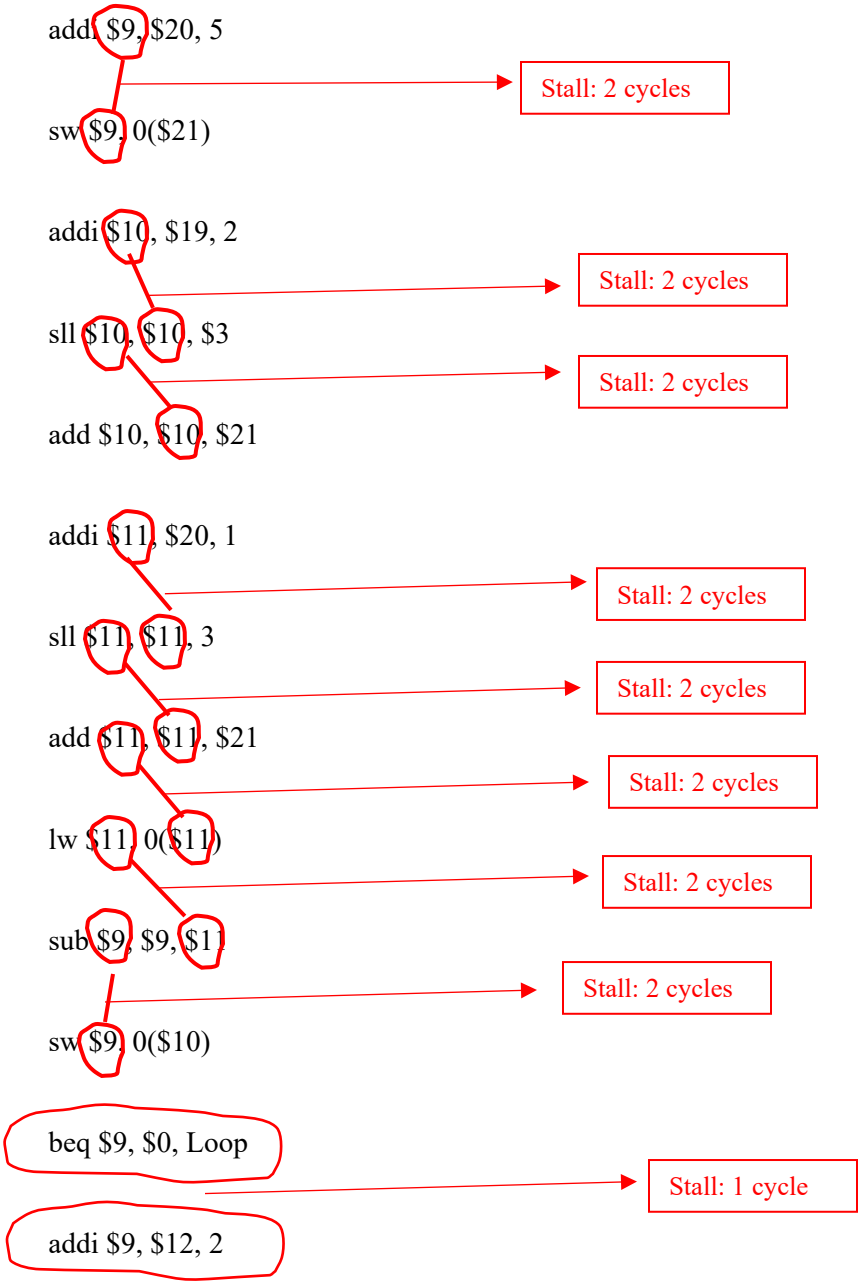
b). Reorder the above code to reduce the stalls (get the smallest number of stalls) without changing the functionality. Assume there is no forwarding, no branch prediction, and no duplicate copies of instruction memory or data memory. **Identify all the stalls** (which two instructions or which register are the cause of the stall) and the **total number of stalls** in the reordered assembly code.

c). Reorder the above code to reduce the stalls without changing the functionality (get the smallest number of stalls). Assume there is forwarding, no branch prediction, and no duplicate copies of instruction memory or data memory. **Identify all the stalls** (which two instructions or which register are the cause of the stall) and the **total number of stalls** in the reordered assembly code.

d). Based on c), **plot the pipeline diagram to show the execution of these instructions and clearly show the data *forwarding/bypassing* in the pipeline diagram.** [hint: The instruction sequence is shown vertically, from top to bottom. Clock cycles are shown horizontally, from left to right. Each instruction is divided into its component stages.].

Answer:

a). Loop: sub $19, $19, $20



(1 point/each stall, total 9 points)

Total: 17 stall cycles. (1 point)

b). Loop: addi $11, $20, 1

sub $19, $19, $20

addi $9, $20, 5

sll $11, $11, 3

addi $10, $19, 2

sw $9, 0($21)

add $11, $11, $21

sll $10, $10, $3

lw $11, 0($11)

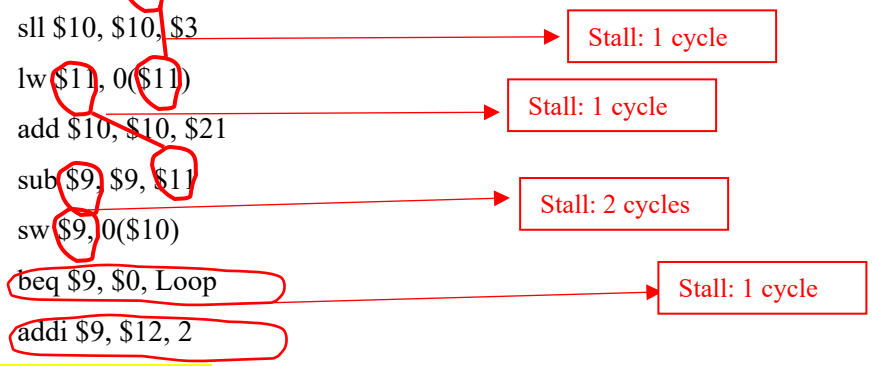
add $10, $10, $21

sub $9, $9, $11

sw $9, 0($10)

beq $9, $0, Loop

addi $9, $12, 2



(1 points/each stall, total 4 points)

Total: 5 stalls (1 points)

c). Loop: sub $19, $19, $20

addi $9, $20, 5

sw $9, 0($21)

addi $10, $19, 2

sll $10, $10, $3

addi $11, $20, 1

sll $11, $11, 3

add $11, $11, $21

lw $11, 0($11)

add $10, $10, $21

sub $9, $9, $11

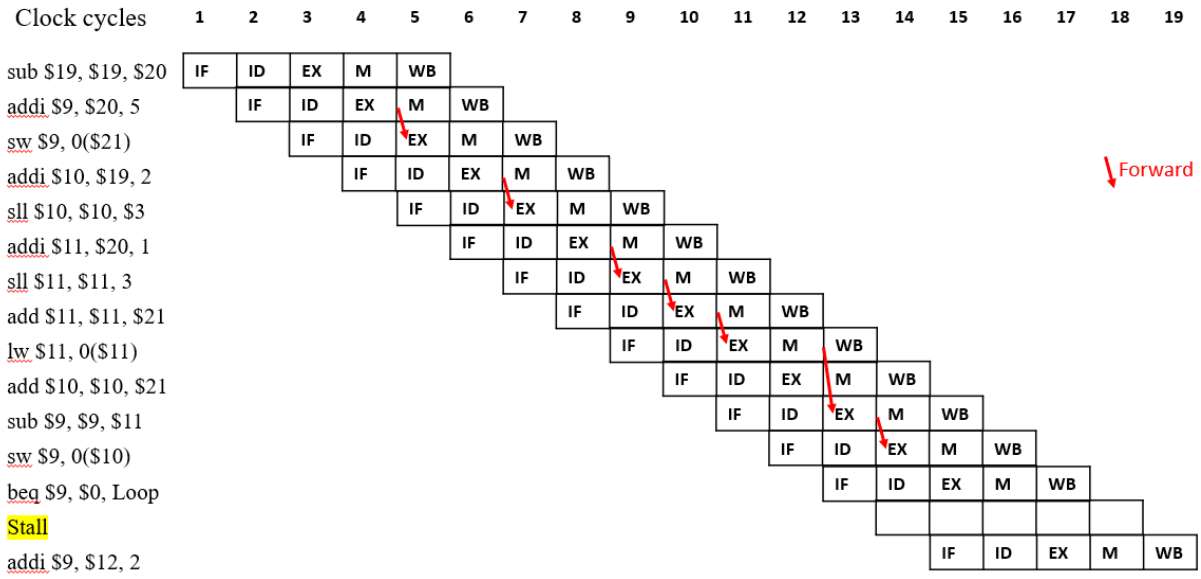
sw $9, 0($10)

beq $9, $0, Loop

addi $9, $12, 2

Total: 1 stall (5 points)

d).



1 points/per forward, total: 7 points

One stall before the last instruction in the diagram (3 points)

Question 5 (20 points). **Compute the effective CPI given the following information, assume we have made the following measurements of average CPI for each of the instruction types (Assume that 50% of the conditional branches are taken):**

Instruction	Clock cycles
All ALU operations	1.0
Loads	3.5
Stores	2.8
Branches	
Taken	4.0
Not taken	2.0
Jumps	2.4

Hint: Average the instruction frequency of **gobmk** and **mcf** to obtain the instruction mix. You may assume that all other instructions (for instructions not accounted for by the type in the above table) require 3.0 clock cycles each.

Program	Loads	Stores	Branches	Jumps	ALU operations
gobmk	21%	12%	14%	2%	50%
mcf	35%	11%	24%	1%	29%

Answer: (give partial points based on the solution)

$$Effective\ CPI = \sum Instruction\ frequency * clock\ cycles =$$
$$0.395 * 1.0 + 0.28 * 3.5 + 0.115 * 2.8 + 0.19 * 3 + 0.015 * 2.4 +$$
$$0.005 * 3.0 = 2.318$$ (20 points)