Jonas Vinter-Jensen
912941515
Spring-18

Project 1

Problem 1

<u>Case 1: Cycle time = 400 ps, 5-stage pipeline</u>

Execution cycles = Cycles Per Instruction ・ Instruction Count + Pipeline Stall Cycles

= CPI ・ IC + Pipeline Stall Cycles.

For this project, the implementation is assumed to be single-cycle for the sake of simplicity, which gives a CPI = 1 for every instruction. Realistically, CPI > 1 for single-issue processors, but this would lead to two unknowns in the resulting expressions. With this assumption, the following execution cycles can be calculated:

load execution cycles = load cycles + load-use stalls = 0.35 ・ IC + 0.55 ・ 0.35 ・ IC

= 1.55 ・ 0.35 ・ IC

Store execution cycles = 0.10 ・ IC

ALU execution cycles = 0.35 ・ IC

Branch execution cycles = 0.20 ・ IC + 0.65 ・ 0.20 ・ IC = 1.65 ・ 0.20 ・ IC

Total execution time = execution cycles ・ cycle time = execution cycles ・ 400 ps

Load execution time = (1.55・0.35・IC)・400 ps = 217・IC ps

Store execution time = (0.10・IC)・400 ps = 40・IC ps

ALU execution time = 140・IC ps

Branch execution time = (1.65・0.20・IC)・400 ps = 132・IC ps

Total execution time = load execution time + store execution time + ...

= 217・IC ps + 40・IC ps + 140・IC ps + 132・IC ps = <u>529・IC ps</u>

<u>Case 2: Cycle time = 250 ps, 8-stage pipeline</u>

Since the pipeline now consists of 8 stages, the number of stalls will increase as shown below:

New load-use stall

lw	IF1	IF2	ID	EX	ME1	ME2	ME3	WB				
add		IF1	IF2	ID	st	st	st	EX	ME1	ME2	ME3	WB

Since lw previously had one stall, every lw now has three times as many stalls in total, requiring the number of stalls for loads to be multiplied by 3.

New branch stall (branch taken, instruction fetched again)

beq	IF1	IF2	ID	EX	ME1	ME2	ME3	WB		
		IF1	IF2	IF1	IF2	ID	EX	ME1	ME2	WB

Since the sequence in the 5-stage pipeline was IF IF, branches now have 3 stalls instead of 1, which means that the previous number of stalls for the 5-stage pipeline must be multiplied by 3.

load execution cycles = $0.35 \cdot IC + 3 \cdot 0.55 \cdot 0.35 \cdot IC = 0.928 \cdot IC$

Store execution cycles = $0.10 \cdot IC$

ALU execution cycles = $0.35 \cdot IC$

Branch execution cycles = $0.20 \cdot IC + 3 \cdot 0.65 \cdot 0.20 \cdot IC = 0.59 \cdot IC$

Total execution time = execution cycles $\cdot$ cycle time = execution cycles $\cdot$ 250 ps

The new execution times are found by inserting the new expressions in the execution time formula:

Load execution time = $(0.928 \cdot IC) \cdot 250$ ps = $232 \cdot IC$ ps

Store execution time = $(0.10 \cdot IC) \cdot 250$ ps = $25 \cdot IC$ ps

ALU execution time = $(0.35 \cdot IC) \cdot 250$ ps = $87.5 \cdot IC$ ps

Branch execution time = $(0.59 \cdot IC) \cdot 250$ ps = $147.5 \cdot IC$ ps

Total execution time = $\underline{492 \cdot IC \text{ ps}}$

Jonas Vinter-Jensen
912941515
Spring-18

Problem 2

a)		1	2	3	4	5	6	7	8	9	10
loop:	lw $17, 0($16)	IF	ID	EX	MEM	WB					
	lw $12, 0($17)		IF	ID	st	EX	MEM	WB			
	sub $12, $12, $14			IF	ID	st	st	EX	MEM	WB	
	add $20, $20, $12				IF	ID	st	st	EX	MEM	WB

Cycle 5: ForwardA = 01

Cycle 7: ForwardA = 01

Cycle 8: ForwardB = 10

b)

```
loop:   lw   $t0, 0($16)        1)
        lw   $t1, 8($16)        2)
        lw   $t2, 16($16)       3)
        lw   $t3, 24($16)       4)
        lw   $t4, 4($16)        5)
        lw   $t5, 12($16)       6)
        lw   $t6, 20($16)       7)
        lw   $t7, 28($16)       8)
        add $t0, $t4, $t0       9)
        add $t1, $t5, $t1       10)
        add $t2, $t6, $t2       11)
        add $t3, $t7, $t3       12)
        sw $t0, 0($23)          13)
        sw $t1, 4($23)          14)
        sw $t2, 8($23)          15)
        addi $16, $16, 32       16)
        addi $23, $23, 16       17)
        sw $t3, -4($23)         18)
        bne $16, $2, loop       19)
```

Timing for 2b:

1)	IF	ID	EX	MEM	WB						
2)		IF	ID	EX	MEM	WB					
3)			IF	ID	EX	MEM	WB				
4)				IF	ID	EX	MEM	WB			
5)					IF	ID	EX	MEM	WB		
6)						IF	ID	EX	MEM	WB	
7)			(...)								
8)			(...)								
9)									IF	ID	(...)
10)			(...)								
11)			(...)								
12)			(...)								
13)			(...)								
14)			(...)								
15)			(...)								
16)	IF	ID	EX	MEM	WB						
17)		IF	ID	EX ↘	MEM	WB					
18)			IF	ID	EX	MEM	WB				
19)				IF	ID	EX	MEM	WB			

Since the lw at 5) depends on the lw at 1), 1) will be in WB pipeline stage long before 5) gets to the EX stage, which resolves the data hazard present before the loop unroll. This distance of between 5) and 1) is maintained for the first eight lw instructions and indicates no stalls.

The add at line 9) relies on the $t4 data - which is its latest dependency - fetched during lw at 5) and since the EX of 9) is after the WB of 5), no stalls are present and no forwarding is required. The sw at 13) relies on 9), which causes no stalls, and the same applies for the sw instructions at 14) and 15). The addi at 16) is far from having any stalls, and the addi at 17) forwards the result in $23 to sw with ForwardA = 10.

The bne at 19) will cause branch delay stall every time the branch is taken, but this is hard to avoid, since moving the instruction to a different line would break the program.

c)

	ALU/branch	Load/store	Clock Cycle
Loop:		lw $t0, 0($16)	1
		lw $t4, 4($16)	2
		lw $t1, 8($16)	3
	add $t0, $t4, $t0	lw $t5, 12($16)	4
		lw $t2, 16($16)	5
	add $t1, $t5, $t1	lw $t6, 20($16)	6
		lw $t3, 24($16)	7
	add $t2, $t6, $t2	lw $t7, 28($16)	8
		sw $t0, 0($23)	9
	add $t3, $t7, $t3	sw $t1, 4($23)	10
	addi $16, $16, 32	sw $t2, 8($23)	11
	addi $23, $23, 16	sw $t3, 12($23)	12
	bne $16, $2, loop		13

At clock cycle 12, since sw does not lag behind addi, addi can not forward to sw in the cycle where sw reaches ID/EX. For this reason, the offset for the sw instruction needs to be 12 instead of -4.