Homework 5

Name: Yiqiao Jin
UID: 305107551

4.13.1

4.13 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)
```

4.13.1 [5] <§4.7> If there is no forwarding or hazard detection, insert nops to ensure correct execution.

If the current instruction uses the output of the previous instruction, then there must be at least 2 instructions in-between. The ID stage of current instruction must be in the same cycle or previous cycles of the previous instruction. This dependency relation exists between:

- the first **add** and the first **lw**
- the first **lw** and the **or**
- the **or** and the **sw**

The first and third cases both need two **nop**'s. The second one requires 1 **nop** since there is a second **lw** in-between which does not have any dependency on previous instructions.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
add r5 r2 r1	IF	ID	EX	M	WB										
nop			NOP												
nop			NOP												
lw r3 4(r5)				IF	ID	EX	M	WB							
lw r2 0(r2)					IF	ID	EX	M	WB						
nop							NOP								
or r3 r5 r3							IF	ID	EX	M	WB				
nop								NOP							
nop								NOP							
sw r3 0(r5)										IF	ID	EX	M	WB	

With forwarding logic

The only case we need to care about is when an instruction uses a source register that was written by a preceding **lw** instruction (MEM stage if one instruction to its 1st next instruction). Both of the **lw** instructions here do NOT have such dependency. So the instructions will take 9 cycles: (5-1) + 5 = 9

4.14.1

4.14.1 show the pipeline diagram for:

a) full forwarding, branches resolved in EX

b) full forwarding, branches resolved in ID

Pipeline execution diagram

Branch resolved in EX: 15 cycles

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lw $r2 0($r1)	IF	ID	EX	M	WB										
beq $r2 $r0 Label2 → NOT Taken		IF	(ID)	ID	EX	M	WB								
sw $r1 0($r2)			(IF)	IF	ID	EX	/								
NOP					IF	ID	/								
NOP						IF	/								
lw $r3 0($r2)							IF	ID	EX	M	WB				
beq $r3 $r0 Label1								IF	(ID)	ID	EX	M	WB		
beq $r2 $r0 Label2 → Taken									(IF)	IF	ID	EX	M	WB	
sw $r1 0($r2)											IF	ID	EX	M	WB

Branch resolved in ID: 13 cycles

Branch resolved in ID

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lw $r2 0($r1)	IF	ID	EX	M	WB										
beq $r2 $r0 Label2 → not taken		IF	(ID)	ID	EX	M	WB								
sw $r1 0($r2)				IF	/	/	/								
lw $r3 0($r2)					IF	ID	EX	M	WB						
beq $r3 $r0 Label1 → taken					IF	(ID)	ID	EX	M	WB					
beq $r2 $r0 Label2 → Taken							IF	ID	EX	M	WB				
sw $r1 0($r2)								IF	(ID)	ID	EX	M	WB		

4.16.1-4.16.3

4.16 This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT

4.16.1 [5] <§4.8> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

4.16.2 [5] <§4.8> 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 state from Figure 4.63 (predict not taken)?

4.16.3 [10] <§4.8> What is the accuracy of the two-bit predictor if this pattern is repeated forever?

4.16.1

Accuracy:
Always taken: 3/5 or 60%; Always not taken: 2/5 or 40%

4.16.2

Before each prediction is made, the state we are in are: 00, 01, 00, 01.

Note that 00 is "Strong not taken" and 01 is "Weak not taken"

Our prediction will be NT, NT, NT, NT

This yields an accuracy of 25%

4.16.3

Our states: 00, 01, 00, 01, 10 | 01, 10, 01, 10, 11 | 10, 11, 10, 11, 11 ...

Actual Sequence: T, NT, T, T, NT | T, NT, T, T, NT | T, NT, T, T, NT ...

Our predictions: NT, NT, NT, NT, T | NT, T, NT, T, T | T, T, T, T, T

Correctness: N, Y, N, N, N | N, N, N, Y, N | Y, N, Y, Y, N

At the end of the loop, we are always in "Strong taken" state or State 11. The first two iterations have accuracy of 20%, and the accuracy will stay at 60% for each followup iteration.

Thus, given i iterations the overall accuracy is $(0.2 \times 2 + 0.6 \times (i-2))/i$