Thursday, March 11, 2021 12:15 AM

1. Consider the following code.

```
addi  $sp, $sp, -8   I    // I1
sw    $ra, 4($sp)    I    // I2
sw    $a0, 0($sp)    I    // I3
slti  $t0, $a0, 1    I    // I4
addi  $t0, $t0, 1    I    // I5
addi  $sp, $sp, 8    I    // I6
```

2 dependencies

$I_0 \to I_1$ no $I_1 \to I_0$

a. Assume that we have a hazard detection unit in a 5-stage pipelined architecture as learned in the class. We do not have a forwarding unit. The hazard detection unit stalls instructions to resolve hazards. How many cycles would take to execute the code?

4 total stalls, since 2 WB instructions which are dependent sources.

N = 6 total instructions, k = 5 stage processor ∴ N + k − 1 + 2 × # of dependent sources = 6 + 5 − 1 + 2×2 = 14 cycles

b. Repeat a. but assume that we also have a forwarding unit that forwards data from EX/MEM to EXE and from MEM/WB to EXE.

N + k − 1 = 6 + 5 − 1 = 10 cycles since no stalls because for forwarding we have 0 stalls for any instruction other than lw.

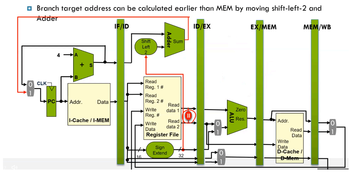
2. Consider the following code.

```
I1:   LOOP:   lw   $a0, 0($sp)
I2:           slti $t0, $a0, 1
I3:           beq  $t0, $zero, L1
I4:           addi $v0, $zero, 1
I5:           addi $sp, $sp, 8
I6:   L1:     lw   $ra, 4($sp)
I7:           addi $t4, $ra, 8
I8:           sw   $t4, 0($t1)
I9:           b    LOOP
I10:          add  $t4, $t4, $t3
```

Notation:
- IF = F
- ID = D
- EXE = E
- MEM = M
- WB = W
- Data hazard:
- F → Flush
- T → Take
- ↘ → Forward

Assume that we have data forwarding paths from EX/MEM to EXE stage and from MEM/WB to EXE stage. We do early branch determination (branch outcome is generated in ID stage and the PC value is updated in ID stage). Branches are predicted untaken but the actual outcomes are as shown in the table. Show the first 18-cycles of execution of the code in a timing diagram shown below. When an instruction is stalled in a pipeline stage, fill the pipeline stage's name for the instruction until the end of stall (as shown on the page 44 of the lecture slide, "CSE140_Lecture-4_Processor-4"). Draw additional rows if needed. Show instruction id to the first column.

ID	Branch instructions	Branch outcome (NT: not taken, T: taken)
I3	beq $t0, $zero, L1	T at the first iteration NT from the second iteration
I9	b LOOP	Always T



	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18
I1	F	D	E	M	W													
I2		F	D	D	E	M	W											
I3			F	F	D	D	E	M	W									
I4					F	F	F	F	F	F								
I6							F	D	E	M	W							
I7								F	D	D	E	M	W					
I8									F	F	D	E	M	W				
I9											F	D	E	M	W			
I10												F	F	F	F	F		
I1													F	D	E	M	W	
I2														F	D	D	E	M
I3															F	F	D	D
I4																	F	F

Notation:
- IF = F
- ID = D
- EXE = E
- MEM = M
- WB = W
- Data hazard
- F → Flush
- T → Take
- ↘ → Forward

3. Consider a branch instruction that has following outcomes:

NT, T, NT, T, NT, NT, NT, T, T, T, NT, T, NT, T

NT: 7
T: 7

a. What is the misprediction rate of always-taken and always-not-taken predictors for this sequence of branch outcomes?

Always taken: If we have an always taken approach, then we assume we always take the path meaning we would be wrong 7/14 times, since we have 7 NT. → 50%

Never Taken: Similarly, if we have an always never taken approach, we would be wrong 7/14 times, so 50% of the time

b. What is the misprediction rate of one-bit branch predictor for this sequence of branch outcomes? The initial predictor state is 0.

Val	0	0	1	0	1	0	0	0	1	1	1	0	1	0
Pred	NT	NT	T	NT	T	NT	NT	NT	T	T	T	NT	T	NT
Result	NT	T	NT	T	NT	NT	NT	T	T	T	NT	T	NT	T

+ predictions: 5
− predictions: 9
∴ 9/14 ≈ 64.28% misprediction rate

c. What is the misprediction rate of two-bit branch predictor for this sequence of branch outcomes? The initial predictor state is 00.

Val	00	00	01	00	01	00	00	00	01	10	11	10	11	10
Pred	NT	NT	NT	NT	NT	NT	NT	NT	NT	T	T	T	T	T
Result	NT	T	NT	T	NT	NT	NT	T	T	T	NT	T	NT	T

+ predictions: 8
− predictions: 6
∴ misprediction rate = 6/14 ≈ 42.85%