L16_1 Pipeline-Performance_Data-Hazards

EECS 370 – Introduction to Computer Organization – Fall 2020

Learning Objectives

- To identify and apply performance metrics related to data hazards for the LC2K pipeline datapath.

Building with Pipelines

Pipeline Performance

- CPI for pipelining:
  - 1 (ideal case - no stalls)
  - > 1 (reality, depends on program)
- What if we want to improve performance more?
  - Want CPI as low as possible – lower than 1
- Use Parallelism
  - Instruction Level Parallelism (ILP) – Within task
  - Thread Level Parallelism (TLP) – Having many tasks
  - Data Level Parallelism (DLP) – Many tasks with same instructions

LC2K Pipeline Summary

Pipeline Performance

Fetch	Decode	Execute	Memory	WB
PC read		Need register values	Branches resolved	Register values produced

- Data hazards
  - Hazard exists if producer-consumer of a register within a 2-instruction window
    - Note for project, the window is 3 instructions
  - Detect and stall – insert enough noops to separate producer and consumer by > 2 instructions (> 3 instructions for project)
  - Detect and forward
    - Handles all cases except LW-USE, need 1 noop here
- Control hazards
  - Detect and stall – needs 3 noops inserted after each branch
  - Predict and squash
    - Zero noops if predict correctly
    - 3 if predict incorrectly

Review: Basic Performance Equation

Pipeline Performance

- Execution time (Time/Program) =
  - # of instr (I/P) x CPI (C/I) x cycle time (T/C)

- Multi-cycle decreases cycle time but increases CPI

- Pipelining decreases CPI
  - Approaches 1.0 if no stalls (hazards that are fixed by stalling)

Calculating Performance with No Stalls

Pipeline Performance

How many cycles does this code take to execute?

```
add   1   2   3
nor   1   4   5
add   4   6   7
```


What value is written to the ALU result field of the Mem/WB pipeline register at the end of cycle 5.

Pipeline Performance

Calculating Performance with No Stalls

How many cycles does this code take to execute?

No stalls – Final WB @ cycle 7

add 1 2 3
nor 1 4 5
add 4 6 7

Time:	1	2	3	4	5	6	7	8
add 1 2 3	IF	ID	EX	ME	WB			
nor 3 4 5		IF	ID	EX	ME	WB		
add 3 5 6			IF	ID	EX	ME	WB	

What value is written to the ALU result field of the Mem/WB pipeline register at the end of cycle 5.

nor result

Performance: Data Hazards -Detect and Stall

Pipeline Performance

```
add  1  2  3
nor  3  4  5
add  3  5  6
```

How many data hazards are there in this code?

How many stall cycles if we use detect and stall to handle the hazards?

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
nor 3 4 5											
add 3 5 6											

Performance: Data Hazards -Detect and Stall

Pipeline Performance

add 1 2 3
nor 3 4 5
add 3 5 6

How many data hazards are there in this code?

3 data hazards

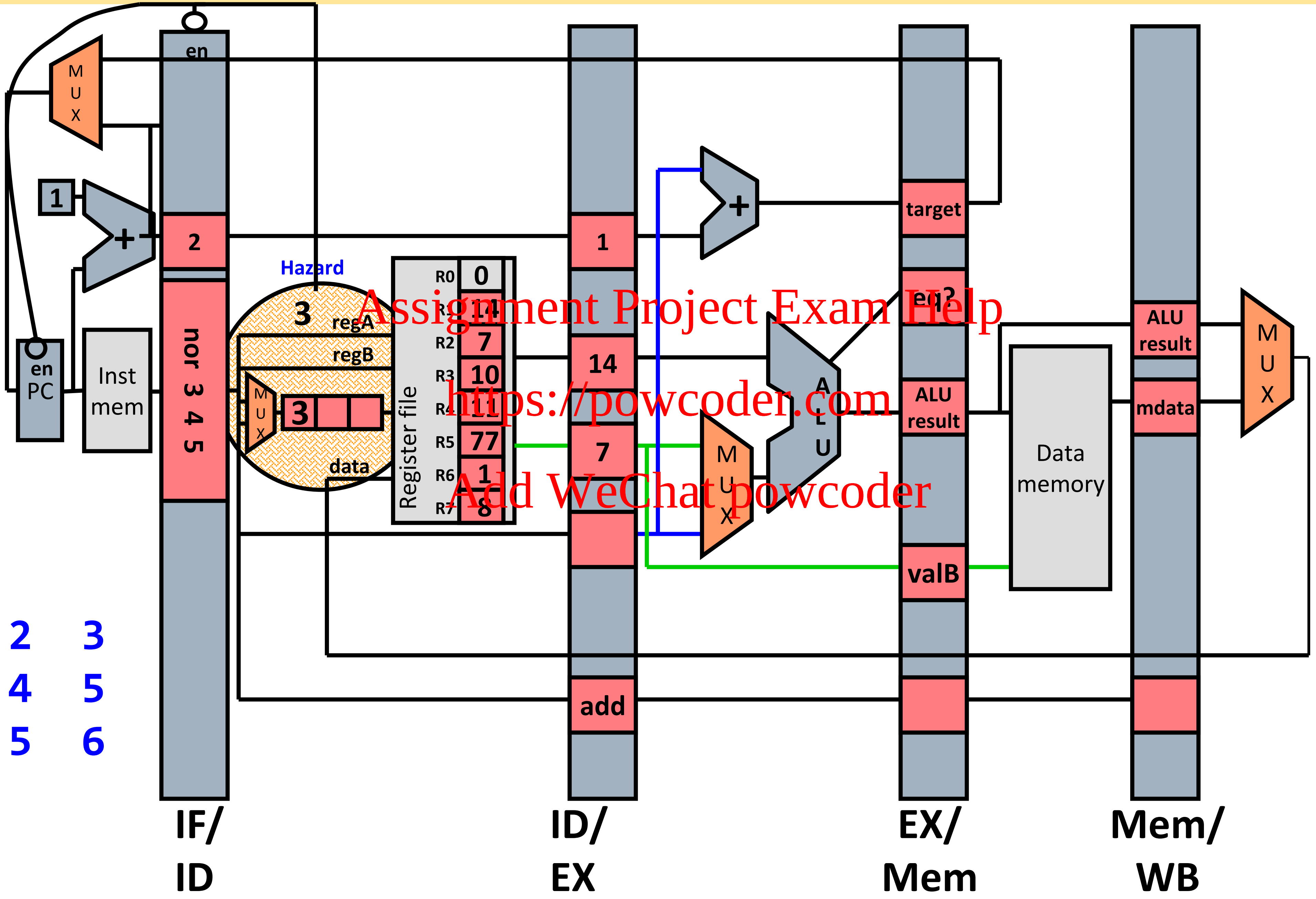
How many stall cycles if we use detect and stall to handle the hazards?

Stall : 4 cycles
Total : 11 cycles

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
nor 3 4 5		IF	ID*	ID*	ID	EX	ME	WB			
add 3 5 6			IF*	IF*	IF	ID*	ID*	ID	EX	ME	WB

First half of cycle 3

Data Hazards



add 1 2 3
nor 3 4 5
add 3 5 6

Performance: Data Hazards -Detect and Forward

Pipeline Performance

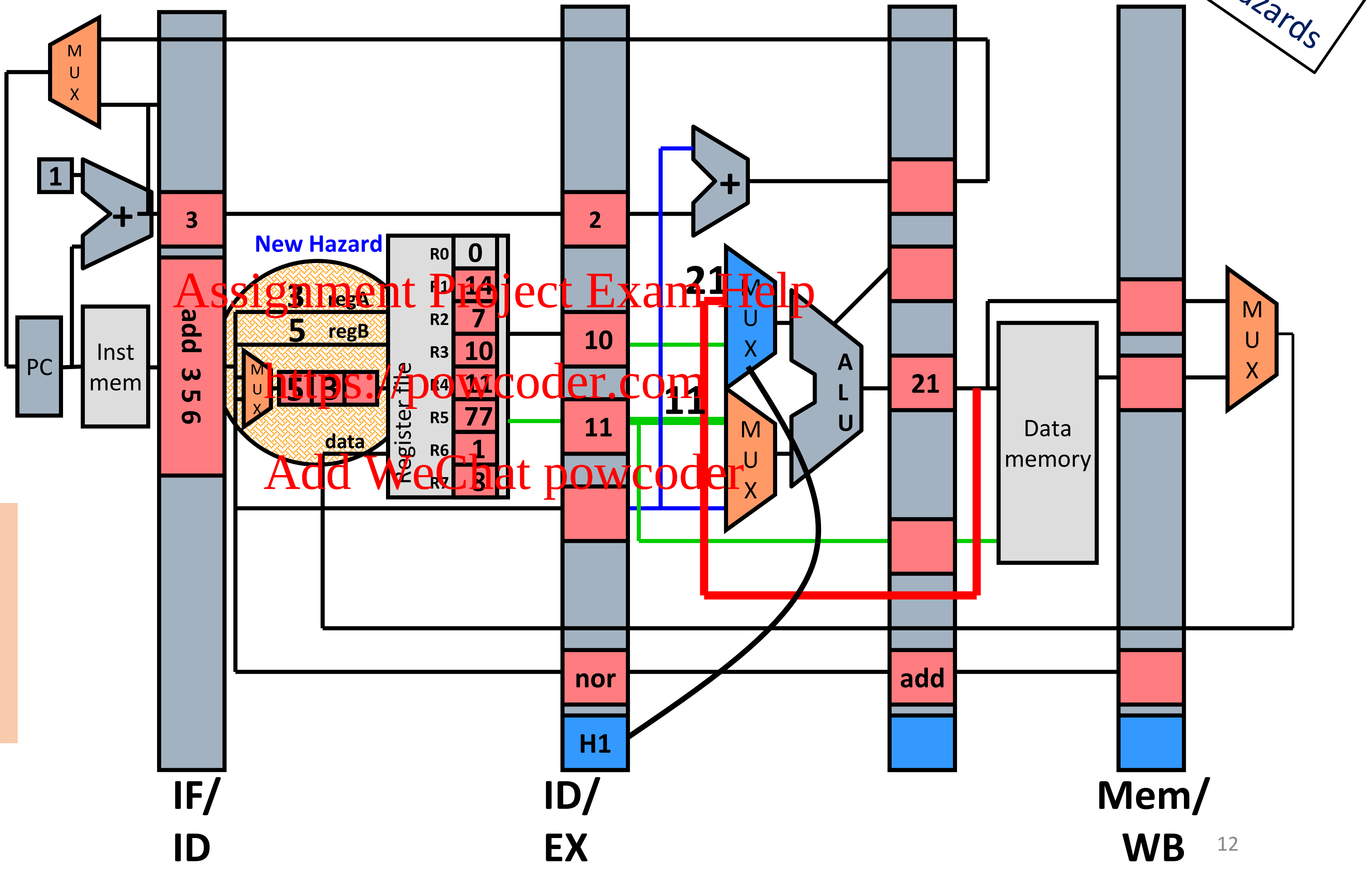
Where do the values for the second add instruction come from?

```
add 1   2   3
nor 3   4   5
add 3   5   6
lw  3   6   7
add 6   6   1
```

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
nor 3 4 5											
add 3 5 6											
lw 3 6 7											
add 6 6 1											

How many stall cycles on the LC2K pipelined datapath with data forwarding from lecture?

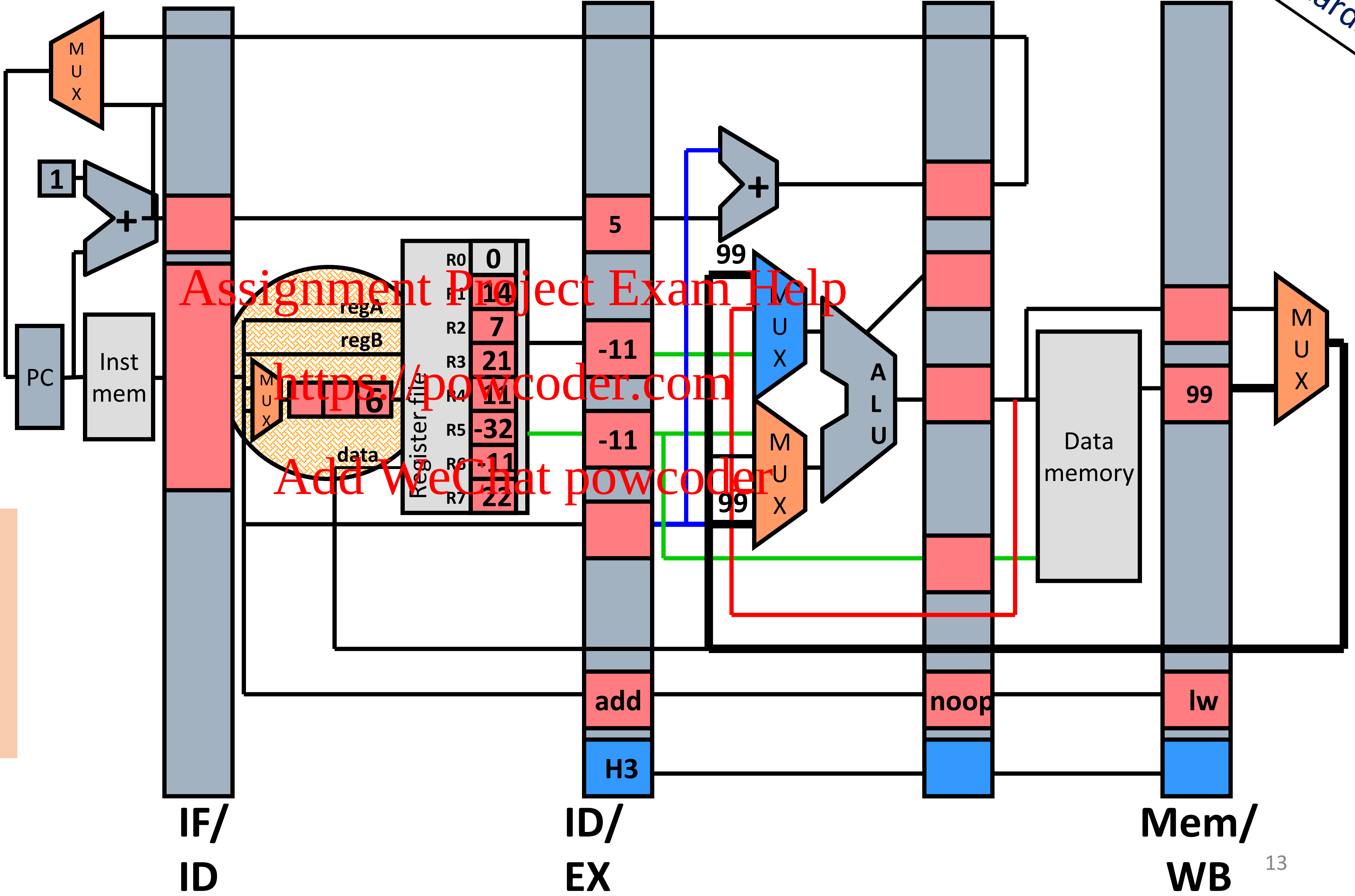
First half of cycle 4



1.	add	1	2	3
2.	nor	3	4	5
3.	add	3	5	6
4.	lw	3	6	7
5.	add	6	6	1

First half of cycle 8

Data Hazards



1. add 1 2 3
2. nor 3 4 5
3. add 3 5 6
4. lw 3 6 7
5. add 6 6 1

Performance: Data Hazards -Detect and Forward

Where do the values for the second add instruction come from?

From Mem/WB and EX/Mem

```
add 1 2 3
nor 3 4 5
add 3 5 6
lw  3 6 7
add 6 6 1
```

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
nor 3 4 5		IF	ID	EX	ME	WB					
add 3 5 6			IF	ID	EX	ME	WB				
lw 3 6 7				IF	ID	EX	ME	WB			
add 6 6 1					IF	ID*	ID	EX	ME	WB	

Data forward

How many stall cycles on the LC2K pipelined datapath with data forwarding from lecture?

1 stall for lw ➜ add

Logistics

- There are 3 videos for lecture 16
  - L16_1 – Pipeline-Performance_Data-Hazards
  - L16_2 – Pipeline-Performance_Control-Hazards
  - L16_3 – Pipeline-Performance

- There is one worksheet for lecture 16
  1. L16 worksheet

L16_2 Pipeline-Performance_Control-Hazards

EECS 370 – Introduction to Computer Organization – Fall 2020

Learning Objectives

- To identify and apply performance metrics related to control hazards for the LC2K pipeline datapath.

Performance: Control Hazards – Speculate and Squash

Pipeline Performance

- How many cycles are saved if you perform speculate and squash for the following code (assume that branches are predicted to be not taken)?

```
add  1  2  3
beq  1  5  1
nor  6  4  1
add  3  4  5
```


- Assume the branch is taken: How many cycles to execute this code?

- Assume the branch is not taken: How many cycles execute this code?

Performance: Control Hazards – Speculate and Squash

Pipeline Performance

Branch prediction not taken

add 1 2 3
beq 1 5 1
nor 6 4 1
add 3 4 5

Branch taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1											
nor 6 4 1											
add 3 4 5											

Branch not taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1											
nor 6 4 1											
add 3 4 5											

Performance: Control Hazards – Speculate and Squash

Pipeline Performance

Branch prediction not taken

add 1 2 3
beq 1 5 1
nor 6 4 1
add 3 4 5

Branch taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1		IF	ID	EX	ME	WB					
nor 6 4 1			~~IF~~	~~ID~~	~~EX~~						
add 3 4 5				~~IF~~	~~ID~~	IF	ID	EX	ME	WB	

Branch not taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1		IF	ID	EX	ME	WB					
nor 6 4 1			IF	ID	EX	ME	WB				
add 3 4 5				IF	ID	EX	ME	WB			

Pipeline Performance

Performance: Control Hazards – Detect and Stall

Same code,
detect and stall

add 1 2 3
beq 1 5 1
nor 6 4 1
add 3 4 5

Branch taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1		IF	ID	EX	ME	WB					
nor 6 4 1			IF*	IF*	IF*						
add 3 4 5						IF	ID	EX	ME	WB	

Branch not taken

Time:	1	2	3	4	5	6	7	8	9	10	11
add 1 2 3	IF	ID	EX	ME	WB						
beq 1 5 1		IF	ID	EX	ME	WB					
nor 6 4 1			IF*	IF*	IF*	IF	ID	EX	ME	WB	
add 3 4 5							IF	ID	EX	ME	WB

Performance: Control Hazards – Speculate and Squash

Pipeline Performance

- How many cycles are saved if you perform speculate and squash for the following code (assume that branches are predicted to be not taken)?

```
add  1  2  3
beq  1  5  1
nor  6  4  1
add  3  4  5
```


- Assume the branch is taken: How many cycles to execute this code?

  3 instructions + 3 stalls + 4 to empty pipe = 10 cycles

- Assume the branch is not taken: How many cycles execute this code?

  4 instructions + 4 to empty pipe = 8 cycles

Performance: Control Hazards I

Pipeline Performance

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

1. How many cycles does the code take assuming *detect and stall* for control hazards?

```
add 1   2   3
beq 1   5   1
lw  6   4   1
add 3   4   5
beq 5   7   -5
halt
```

Control Hazards - Stall

Control Hazards Review

```
beq   1   1   10
add   3   4   5
```


time →

beq	fetch	decode	execute	memory	writeback
add		fetch*	fetch*	fetch*	fetch
					OR
			Branch target address		fetch

Time Graph – Detect and Forward

Data Hazards Review

Time:	1	2	3	4	5	6	7	8	9	10	11	12	13
add 1 2 3	IF	ID	EX	ME	WB								
nor 3 4 5		IF	ID	EX	ME	WB							
add 6 3 7			IF	ID	EX	ME	WB						
lw 3 6 10				IF	ID	EX	ME	WB					
sw 6 2 12					IF	**ID***	ID	EX	ME	WB			

Data forward

Performance: Control Hazards I

Pipeline Performance

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

1. How many cycles does the code take assuming *detect and stall* for control hazards?

```
add  1  2  3
beq  1  5  1
lw   6  4  1
add  3  4  5
beq  5  7  -5
halt
```


\# Instructions = 100*(0.5*5 + 0.5*4) + 1 = 451

Time = 451 + load stalls + branch stalls + empty pipe

Time = 451 + 100*0.5*1 +

Time = 451 + 100*0.5*1 + (100*3 + 100*3) + 4

Time = 1105

Performance: Control Hazards II

Pipeline Performance

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

2. How many cycles does the code take assuming *speculate and squash* where all branches are predicted not taken?

```
add 1  2  3
beq 1  5  1
lw  6  4  1
add 3  4  5
beq 5  7  -5
halt
```

Performance: Control Hazards II

Pipeline Performance

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

2. How many cycles does the code take assuming *speculate and squash* where all branches are predicted not taken?

```
add 1  2  3
beq 1  5  1
lw  6  4  1
add 3  4  5
beq 5  7  -5
halt
```


Instructions = 100*(0.5*5 + 0.5*4) + 1 = 451

Time = 451 + load stalls + branch stalls + empty pipe

Time = 451 + 100*0.5*1 + (100*0.5*3 + 99*3) + 4

Time = 952

Performance: Control Hazards III

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

3. How many cycles does the code take assuming *speculate and squash* where backward branches are predicted taken and forward branches not taken (*BTFN*)? Assume that the predictor has a BTB with entries for both branches to start.

```
add 1   2   3
beq 1   5   1
lw  6   4   1
add 3   4   5
beq 5   7   -5
halt
```

Performance: Control Hazards III

Pipeline Performance

Assume halt is resolved in WB stage

Assume the first branch is taken 50% of the time and the loop iterates 100 times and forwarding for all data hazards.

3. How many cycles does the code take assuming *speculate and squash* where backward branches are predicted taken and forward branches not taken (*BTFN*)? Assume that the predictor has a BTB with entries for both branches to start.

```
add  1  2  3
beq  1  5  1
lw   6  4  1
add  3  4  5
beq  5  7  -5
halt
```


\# Instructions = 100*(0.5*5 + 0.5*4) + 1 = 451

Time = 451 + load stalls + branch stalls + empty pipe

Time = 451 + 100*0.5*1 + (100*0.5*3 + 1*3) + 4

Time = 658

Performance: Control Hazards IV

Pipeline Performance

Assume halt is resolved in WB stage

```
add 1   2   3
beq 1   5   1
lw  6   4   1
add 3   4   5
beq 5   7   -5
halt
```

Assume the first branch has the pattern **TTTN** that repeats, and the loop is iterated 100 times and forwarding for all data hazards.

4. How many cycles does the code take if a 2-bit counter BTB is used to predict each branch, how many cycles does the code take? Assume initial state of branch predictor counter is "10" (WT)

Performance: Control Hazards IV

Pipeline Performance

Assume halt is resolved in WB stage

```
add 1  2  3
beq 1  5  1
lw  6  4  1
add 3  4  5
beq 5  7  -5
halt
```

Assume the first branch has the pattern **TTTN** that repeats, and the loop is iterated 100 times and forwarding for all data hazards.

4. How many cycles does the code take if a 2-bit counter BTB is used to predict each branch, how many cycles does the code take? Assume initial state of branch predictor counter is "10" (WT)

`beq 1 5 1`

√ √ √ X √ √ √ X √ √ √ X
T T T N T T T N T T T N•••

Instructions = 100*(0.25*5 + 0.75*4) + 1 = 426

Time = 426 + load stalls + branch stalls + empty pipe

Time = 426 + 100*0.25*1 + 100*0.25*3 + 1*3 + 4

Time = 533

`beq 5 7 -5` is correct 99 times, then incorrect last iteration

Logistics

- There are 3 videos for lecture 16
  - L16_1 – Pipeline-Performance_Data-Hazards
  - L16_2 – Pipeline-Performance_Control-Hazards
  - L16_3 – Pipeline-Performance

- There is one worksheet for lecture 16
  1. L16 worksheet

- There are optional, supplementary videos with detailed walk-through for the examples
  - These are optional, if you want to see the (repetitious) walk-through for examples in the lecture.

L16_3 Pipeline-Performance

EECS 370 – Introduction to Computer Organization – Fall 2020

Learning Objectives

- To identify and apply performance metrics related to all hazards for the LC2K pipeline datapath.

Classic Performance Problem

Pipeline Performance

- Program with following instruction breakdown:

`lw`	10%
`sw`	15%
`beq`	25%
`R-type`	50%

- Speculate "always not-taken" and squash. 80% of branches not-taken

- Full forwarding to execute stage. 20% of loads stall for 1 cycle
- What is the CPI of the program?
- What is the total execution time if cycle frequency is 100MHz?

Classic Performance Problem

Pipeline Performance

- Program with following instruction breakdown:

lw	10%
sw	15%
beq	25%
R-type	50%

- Speculate "always not-taken" and squash. 80% of branches not-taken
- Full forwarding to execute stage. 20% of loads stall for 1 cycle
- What is the CPI of the program?
- What is the total execution time if cycle frequency is 100MHz?

CPI = 1 + 0.10 (loads) * 0.20 (load use stall)*1

+ 0.25 (branch) * 0.20 (miss rate)*3

CPI = 1 + 0.02 + 0.15 = 1.17

Time = 1.17 * 10ns = 11.7ns

Classic Performance Problem 2.0

Pipeline Performance

- Assume branches are resolved at Execute?
  - What is the CPI?
  - What happens to cycle time?
  - What is the total execution time?

Classic Performance Problem 2.0

Pipeline Performance

- Assume branches are resolved at Execute?
  - What is the CPI?
  - What happens to cycle time?
  - What is the total execution time?

CPI = 1 + 0.10 (loads) *0.20 (load use stall)*1

+ 0.25 (branch) * 0.20 (miss rate)*2

CPI = 1 + 0.02 + 0.1 = 1.12

Pipeline Performance

Performance: Deeper Pipelines

- Assume the setup of the previous problem.
- What if we have a 10-stage pipeline?
  - Instructions are fetched at stage 1.
  - Register file is read at stage 3.
  - Execution begins at stage 5.
  - Branches are resolved at stage 7.
  - Memory access is complete in stage 9.
- What's the CPI of the program?
- If the clock rate was doubled by doubling the pipeline depth, is performance also doubled?

Pipeline Performance

Performance: Deeper Pipelines

- Assume the setup of the previous problem.
- What if we have a 10-stage pipeline?
  - Instructions are fetched at stage 1.
  - Register file is read at stage 3.
  - Execution begins at stage 5.
  - Branches are resolved at stage 7.
  - Memory access is complete in stage 9.
- What's the CPI of the program?
- If the clock rate was doubled by doubling the pipeline depth, is performance also doubled?

CPI = 1
+ 0.10 (loads) * 0.20 (load use stall) * ???
+ 0.25 (branch) * 0.20 (N stalls) * ???

Performance: Deeper Pipelines

Pipeline Performance

- Assume the setup of the previous problem.
- What if we have a 10-stage pipeline?
  - Instructions are fetched at stage 1.
  - Register file is read at stage 3.
  - Execution begins at stage 5.
  - Branches are resolved at stage 7.
  - Memory access is complete in stage 9.
- What's the CPI of the program?
- If the clock frequency was doubled by doubling the pipeline depth, is performance also doubled?

CPI = 1
+ 0.10 (loads) * 0.20 (load use stall) * 4
+ 0.25 (branch) * 0.20 (N stalls) * 6

CPI = 1 + 0.08 + 0.30 = 1.38

Time = 1.38 * 5ns = 6.9 ns

Up Next… Caches

- This is the last lecture on pipeline datapath.
- Next lecture: caches
  - Usually memory hierarchy between the processor and main memory

- Starting Thursday Prof. Satish Narayanasamy will be recording lectures and holding office hours

- It was great to teach you!

Logistics

- There are 3 videos for lecture 16
  - L16_1 – Pipeline-Performance_Data-Hazards
  - L16_2 – Pipeline-Performance_Control-Hazards
  - L16_3 – Pipeline-Performance

- There is one worksheet for lecture 16
  1. L16 worksheet