Lecture 18: More pipeline hazards

CMPE 120: Computer Organization & Architecture

Olivia Weng

Logistics

- Reminder: check-in with professor is 1% of your final grade
  - Sign up link

- HW4 out - start early!!
  - 3 weeks to do for a reason

Mid-semester feedback survey

https://forms.gle/KD69yT5jKM9r5deq8

Pipelining considerations

- What is required for the best case (CPI = 1)?
  - 1 instruction starts every cycle
  - 1 instruction finishes every cycle

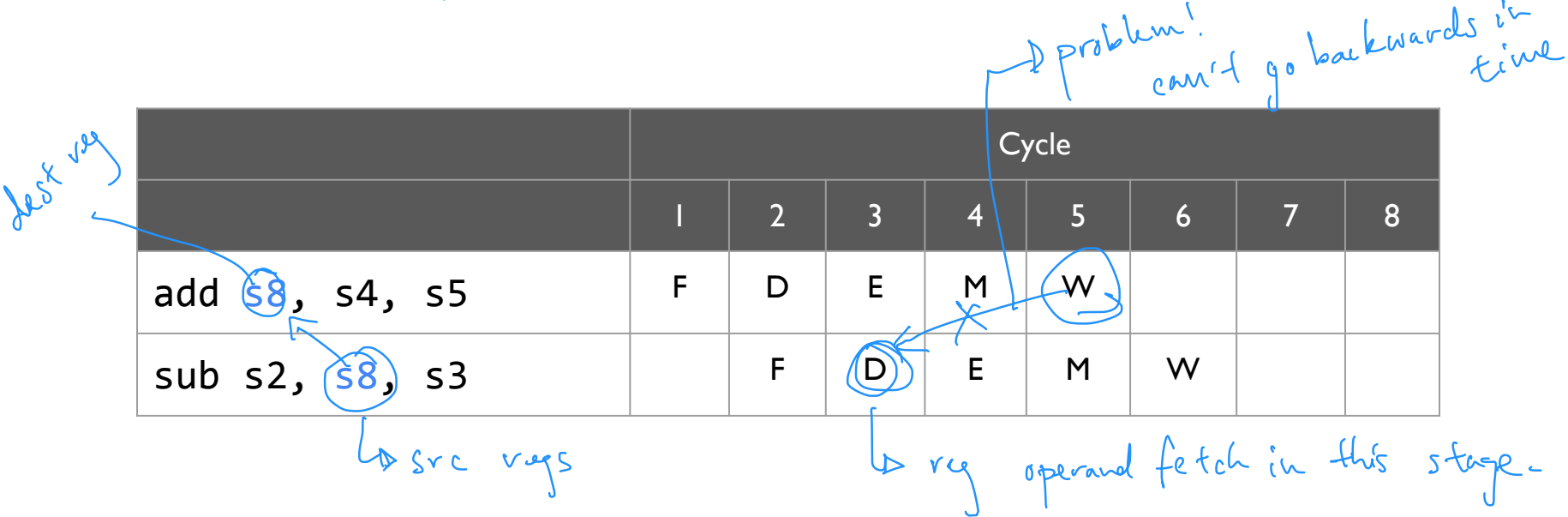
- Is this always possible?

Data hazards

- Data hazard: when an instruction needs a register value that is not available
  - Data dependencies can cause data hazards

Data hazards

- Data hazard: when an instruction needs a register value that is not available
  - Data dependencies can cause data hazards
  - Ex: register s8 is updated in cycle 5, but sub needs to read it in cycle 3



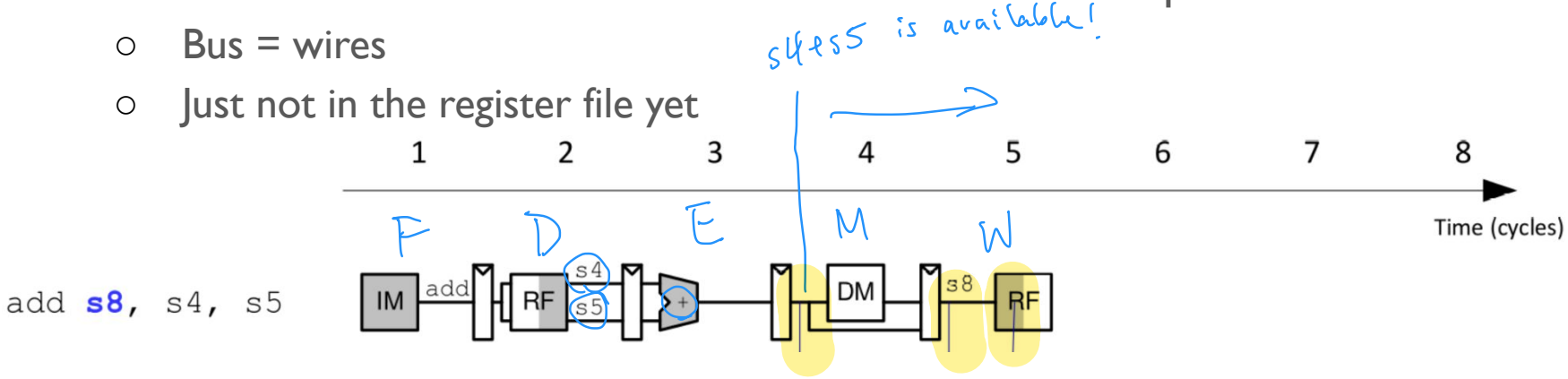
	Cycle							
	1	2	3	4	5	6	7	8
`add s8, s4, s5`	F	D	E	M	W			
`sub s2, s8, s3`		F	D	E	M	W		

Data hazard solution: data forwarding

- Observation: Data is available on internal busses in the processor
  - Bus = wires
  - Just not in the register file yet

Data hazard solution: data forwarding

- Observation: Data is available on internal busses in the processor
  - Bus = wires
  - Just not in the register file yet



Data hazard solution: data forwarding

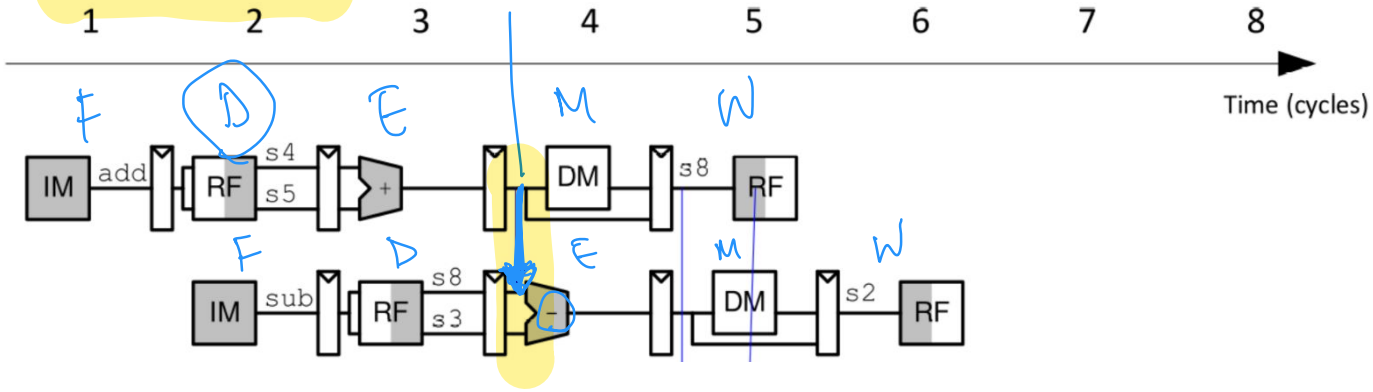
- Observation: Data is available on internal busses in the processor
  - Bus = wires
  - Just not in the register file yet

- Idea: Add new connections to bypass the register file
  - i.e., forward data from internal busses to Execute stage
  - Known as data forwarding

Data hazard solution: data forwarding

- Idea: Add new connections to bypass the register file
  - i.e., forward data from internal busses to Execute stage
  - Known as data forwarding

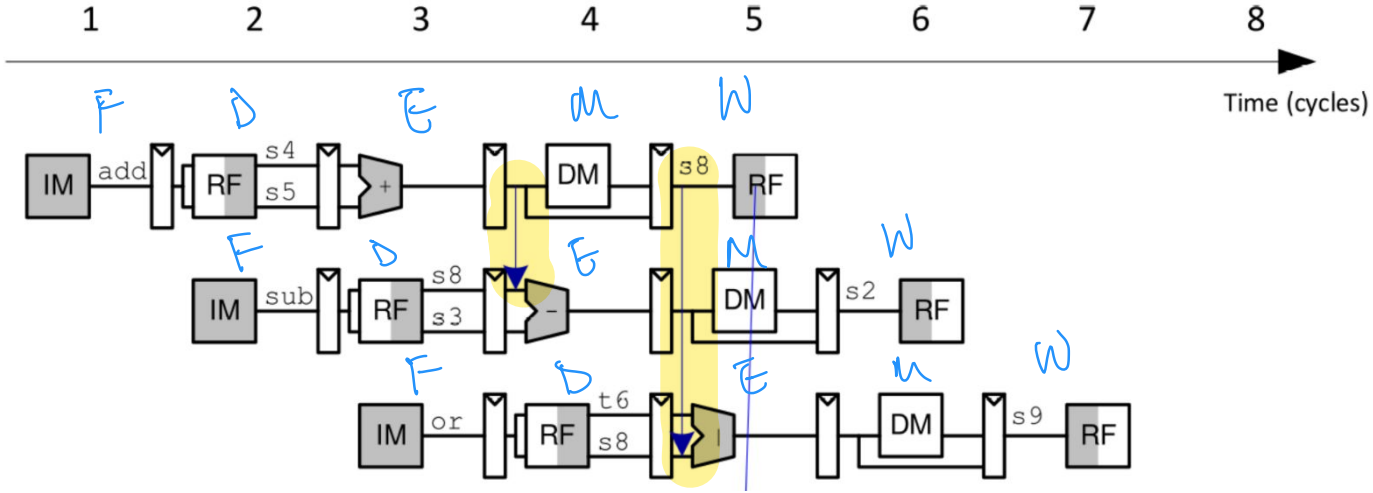
```
add s8, s4, s5
sub s2, s8, s3
```



Data hazard solution: data forwarding

- Idea: Add new connections to bypass the register file
  - i.e., forward data from internal busses to Execute stage
  - Known as data forwarding

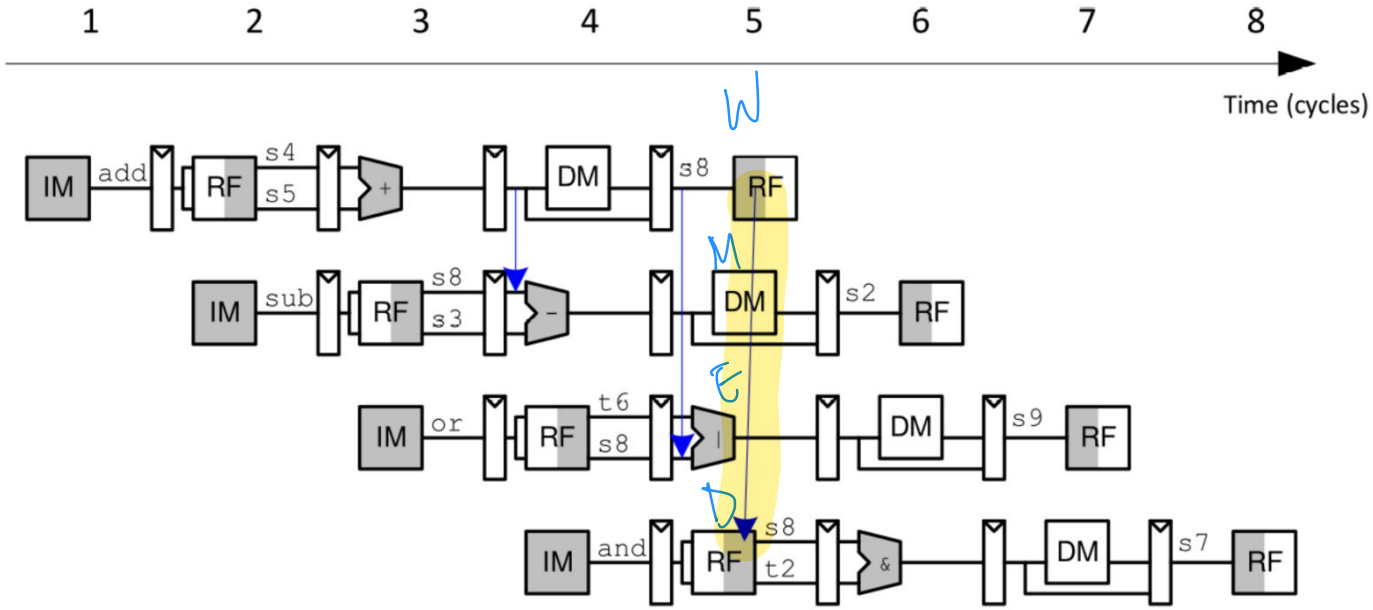
```
add s8, s4, s5
sub s2, s8, s3
or  s9, t6, s8
```



Data hazard solution: data forwarding

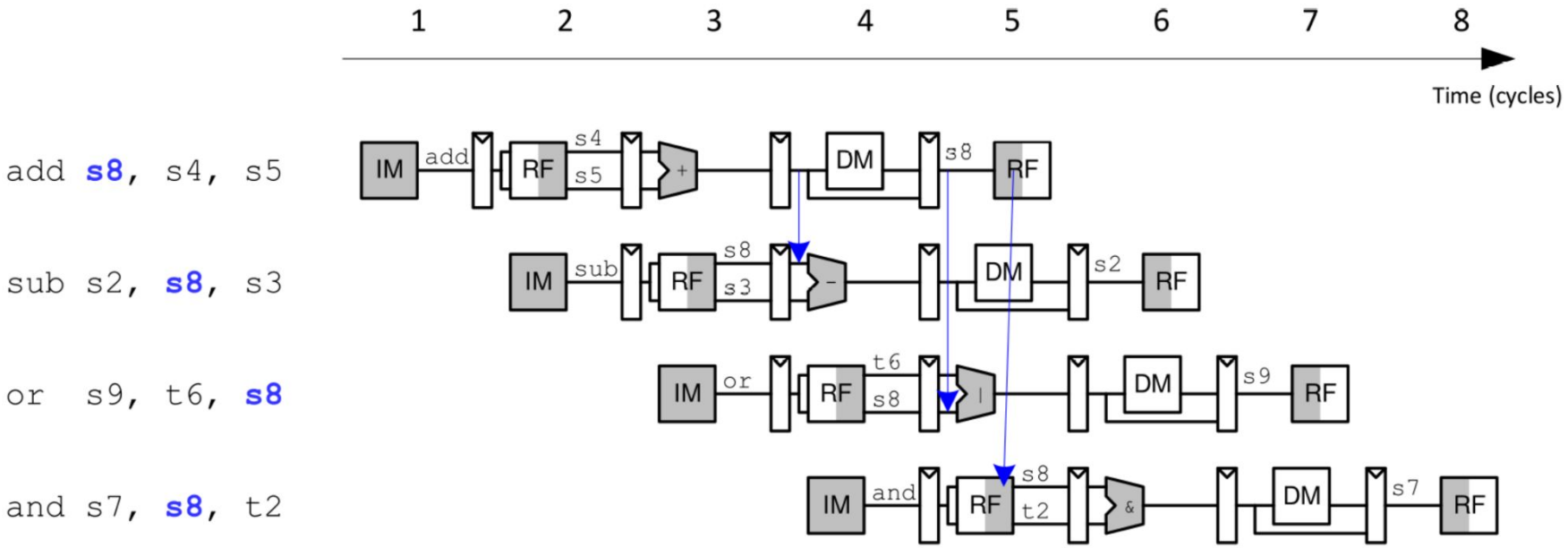
- What is the CPI now?

```
add s8, s4, s5
sub s2, s8, s3
or  s9, t6, s8
and s7, s8, t2
```



Data hazard solution: data forwarding

- What is the CPI now? 1 :)



Data hazard solution: data forwarding

- if Execute stage's `rs1` or `rs2` matches Memory stage's `rd`?
  - Forward `rd`'s data from Memory stage to Execute stage

- else if Execute stage's `rs1` or `rs2` matches Writeback stage's `rd`?
  - Forward `rd`'s data from Writeback stage to Execute stage

- else use value read from register file as usual

Is this instruction execution possible?

A: Yes B: Maybe/Depends C: No

mem addr = s5 + 40

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E (s5+40)	M (read)	W (write to s7)			
`and s8, s7, t3`		F	D (read s7)	E	M	W		

can't go backwards in time

Is this instruction execution possible?

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E	M	W			
`and s8, s7, t3`		F	D	E	M	W		

Data hazard from load-use dependency!

Load-use dependency solution: stall

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E	M	W			
`and s8, s7, t3`		F	D	D	E	M	W	

- Stall: stage waits until data is available
  - Forwarding is not possible because memory data not available till end of cycle

Load-use dependency solution: stall

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E	M	W			
`and s8, s7, t3`		F	D	D	E	M	W	
`or t2, s6, s7`			F	F	D	E	M	W

- Stall: stage waits until data is available
  - Forwarding is not possible because memory data not available till end of cycle

Load-use dependency solution: stall

- Stall: stage waits until data is available

- Is either source register in Decode stage same as destination register in Execute?
  - AND is the instruction in Execute is a load?

- Then stall the Fetch and Decode stages AND
  - Insert Bubble into Execute stage so it does not accidentally compute incorrect data

Load-use dependency solution: stall

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E	M	W			
`and s8, s7, t3`		F	D	D	E	M	W	
`or t2, s6, s7`			F	F	D	E	M	W

- Stall: stage waits until data is available

Load-use dependency solution: stall

	Cycle							
	1	2	3	4	5	6	7	8
`lw s7, 40(s5)`	F	D	E	M	W			
`bubble/nop`				E	M	W		
`and s8, s7, t3`		F	D	D	E	M	W	
`or t2, s6, s7`			F	F	D	E	M	W

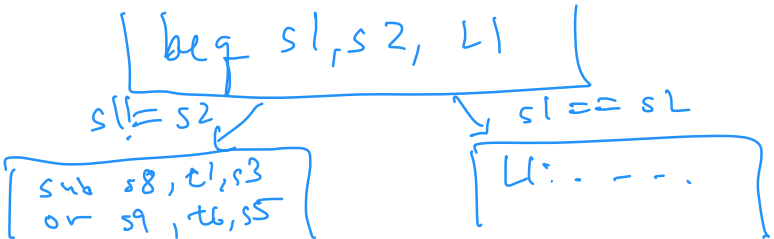
- Stall: stage waits until data is available
  - Insert Bubble into Execute stage so it does not accidentally compute incorrect data

Is this instruction processing correct?

~~Is there any issue here?~~

A: Yes B: Maybe/Depends C: No.

	Cycle							
	1	2	3	4	5	6	7	8
`beq s1, s2, L1`	F	D	E	M	W			
`sub s8, t1, s3`		F	D	E	M	W		
`or s9, t6, s5`			F	D	E	M	W	



Is there any issue here?

if (s1 == s2)

else:
    s8 = t1 - s3
    s9 = t6 | s5

	Cycle							
	1	2	3	4	5	6	7	8
`beq s1, s2, L1`	F	D	E	M	W			
`sub s8, t1, s3`		F	D	E	M	W		
`or s9, t6, s5`			F	D	E	M	W	

- Depends! If branch is not taken, then no. If branch taken, then yes!

Control hazard

- Branch condition not resolved until the Execute stage
  - Instructions will continue to be fetched !
  - These 2 instructions must be flushed if branch is taken

Control hazard

- Branch condition not resolved until the Execute stage
  - Instructions will continue to be fetched !
  - These 2 instructions must be flushed if branch is taken

- Control hazard: next instruction not decided yet (caused by branch)

Control hazard

- Branch condition not resolved until the Execute stage
  - Instructions will continue to be fetched !
  - These 2 instructions must be flushed if branch is taken

	Cycle							
	1	2	3	4	5	6	7	8
`beq s1, s2, L1`	F	D	E	M	W			
`sub s8, t1, s3`		F	D	E	M	W		
`or s9, t6, s5`			F	D	E	M	W	

Control hazard

	Cycle							
	1	2	3	4	5	6	7	8
`beq s1, s2, L1`	F	D	E	M	W			
`sub s8, t1, s3`		F	D	E	M	W		
`or s9, t6, s5`			F	D	E	M	W	
…								
`L1: add s7, s3, s4`				F	D	E	M	W