Lecture 17: Pipelining cont'd

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

Pipelining: an assembly line for instructions

- Idea: Divide instruction processing into distinct stages
  - Process a different instruction in each stage
  - Use all stages all the time to leave no resource idle!
  - → Instruction-level parallelism

Pipelining: an assembly line for instructions

- Idea: Divide instruction processing into distinct stages
  - Process a different instruction in each stage
  - Use all stages all the time to leave no resource idle!
  - → Instruction-level parallelism

- For every cycle
  - start an instruction
  - finish an instruction

- Does this help improve instruction processing latency or throughput?

Pipelining: an assembly line for instructions

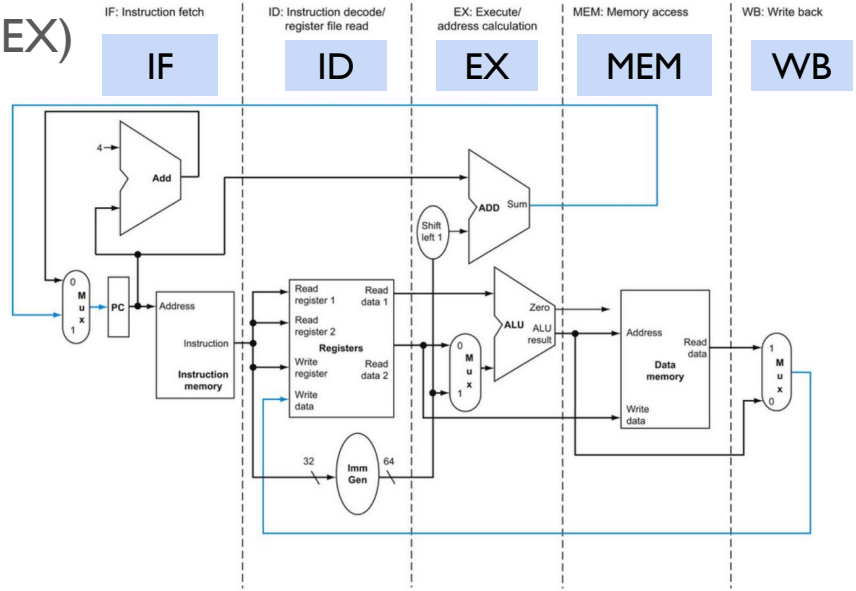
- Idea: Divide instruction processing into distinct stages
  - Process a different instruction in each stage
  - Use all stages all the time to leave no resource idle!
  - → Instruction-level parallelism

- For every cycle
  - start an instruction
  - finish an instruction

- Does this help improve instruction processing latency or throughput?
  - Increase instruction processing throughput

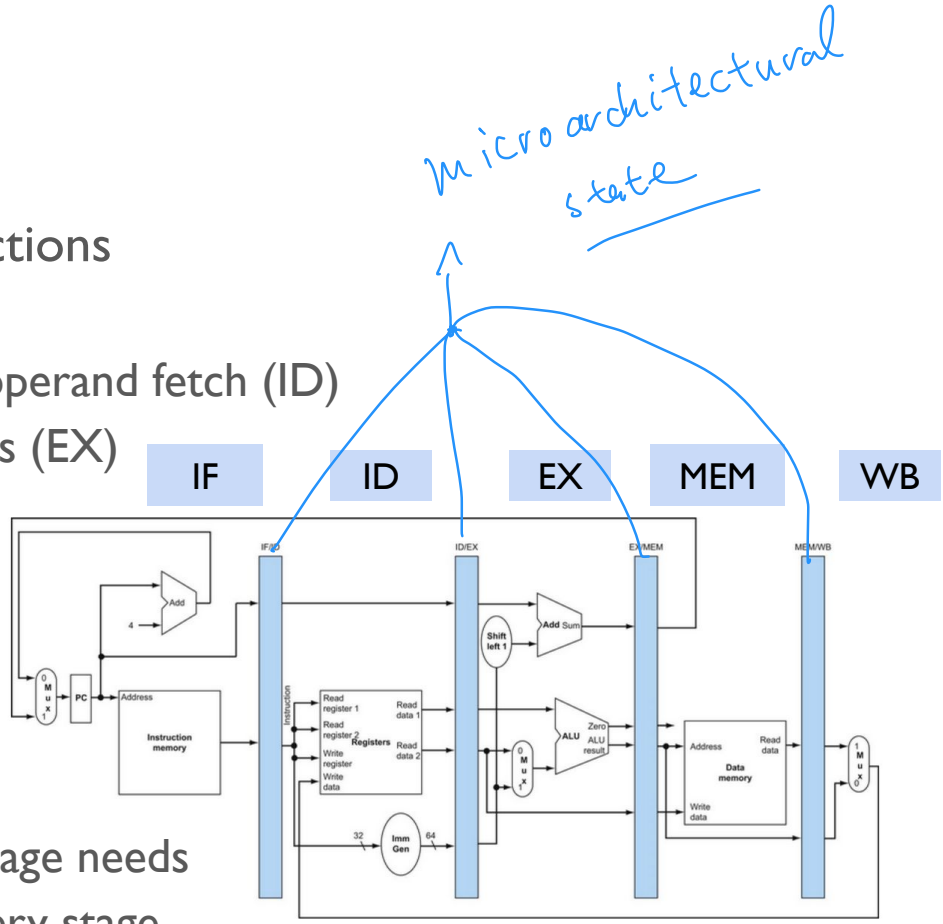
Pipeline stages

- 5 stages to process RISC-V instructions
  - Instruction fetch (IF)
  - Instruction decode and register operand fetch (ID)
  - Execute/evaluate memory address (EX)
  - Memory access (MEM)
  - Store/writeback result (WB)



Pipeline stages

- 5 stages to process RISC-V instructions
  - Instruction fetch (IF)
  - Instruction decode and register operand fetch (ID)
  - Execute/evaluate memory address (EX)
  - Memory access (MEM)
  - Store/writeback result (WB)



- Pipeline registers between stages
  - Store intermediate values each stage needs
  - Ex: instruction data needed in every stage

Pipelining in action

```
lw s2, 40(s0)

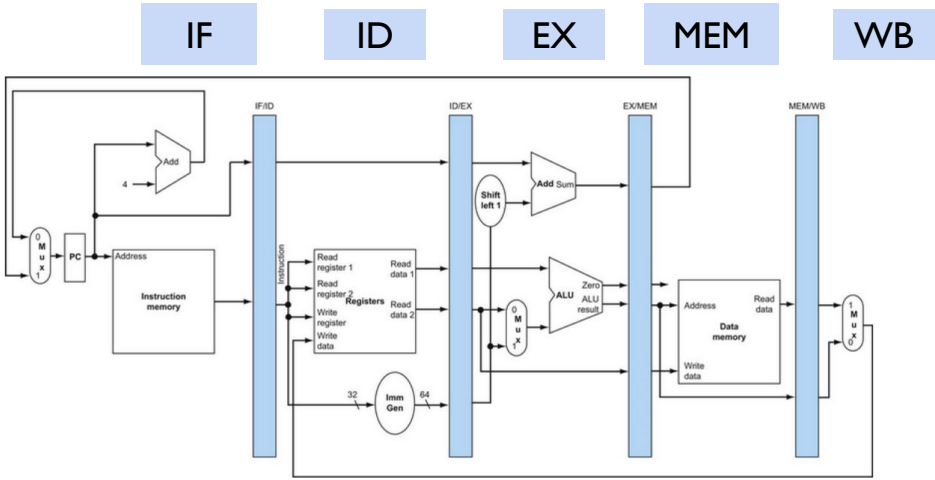
add s3, s9, s10

sub s4, t1, s8

and s5, s11, t0

sw s6, 20(t4)

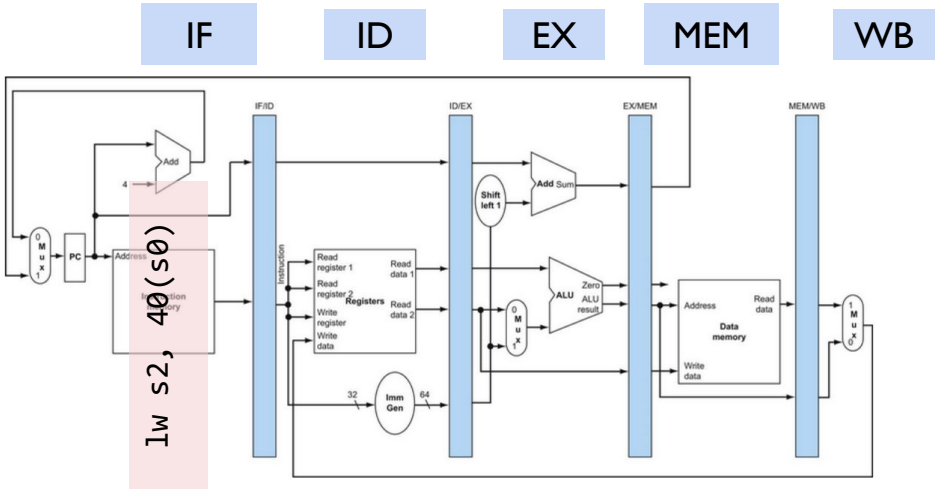
or s7, t2, t3
```



Pipelining in action

```
lw s2, 40(s0)
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw s6, 20(t4)
or s7, t2, t3
```

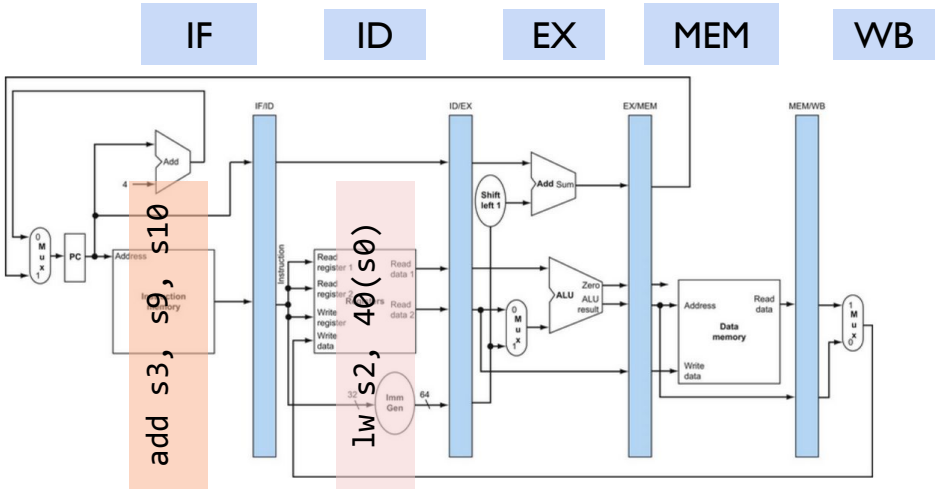
cycle 1



Pipelining in action

```
lw s2, 40(s0)
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw s6, 20(t4)
or s7, t2, t3
```

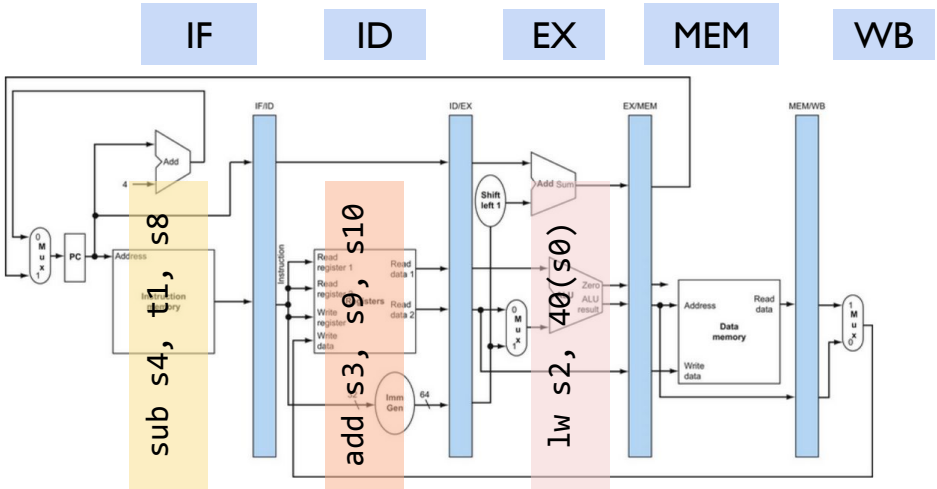
cycle 2



Pipelining in action

```
lw s2, 40(s0)
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw s6, 20(t4)
or s7, t2, t3
```

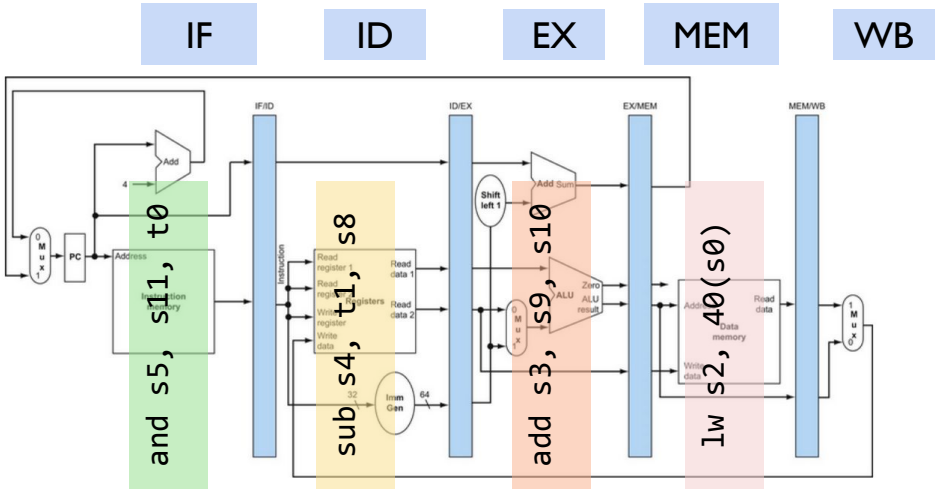
cycle 3



Pipelining in action

```
lw s2, 40(s0)
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw s6, 20(t4)
or s7, t2, t3
```

cycle 4



Pipelining in action

```
lw s2, 40(s0)

add s3, s9, s10

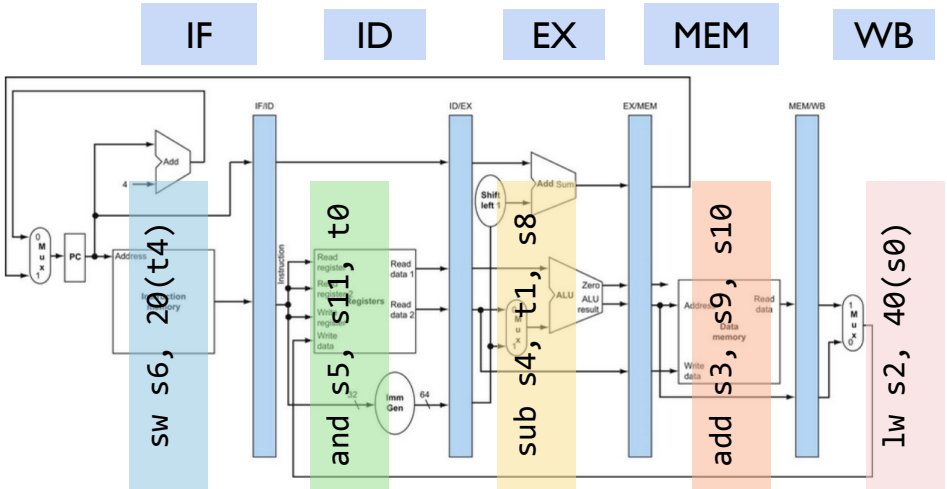
sub s4, t1, s8

and s5, s11, t0

sw s6, 20(t4)

or s7, t2, t3
```

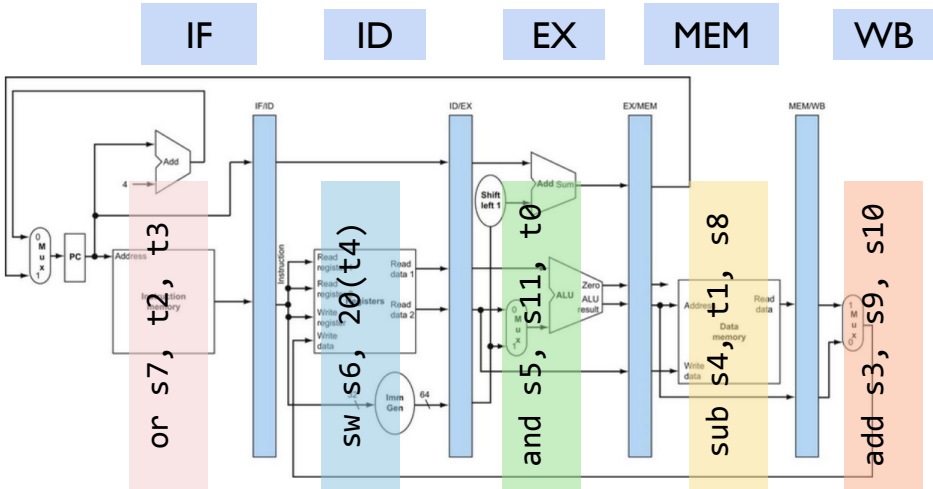
cycle 5



Pipelining in action

```
lw s2, 40(s0) - retired!
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw s6, 20(t4)
or s7, t2, t3
```

cycle 6



Pipelining in action

```
lw s2, 40(s0) - retired!

add s3, s9, s10 - retired!

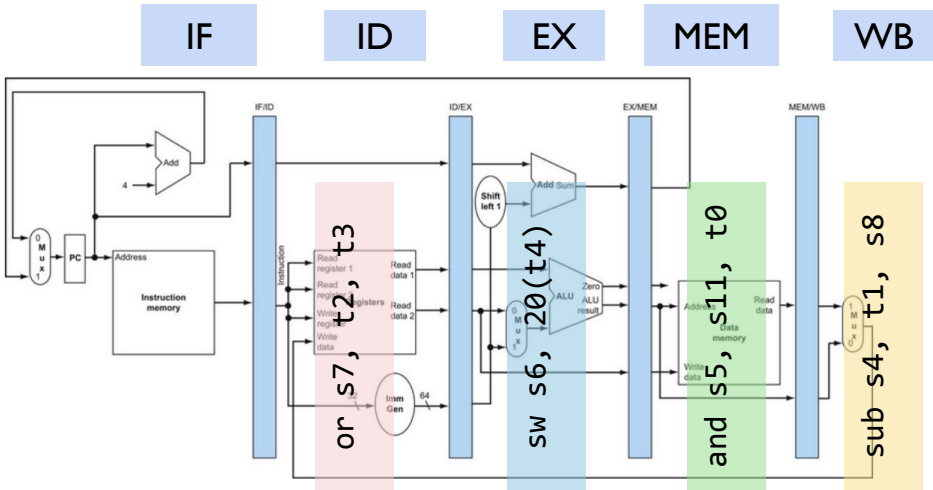
sub s4, t1, s8

and s5, s11, t0

sw s6, 20(t4)

or s7, t2, t3
```

cycle 7



Pipelining in action

```
lw s2, 40(s0) - retired!

add s3, s9, s10 - retired!

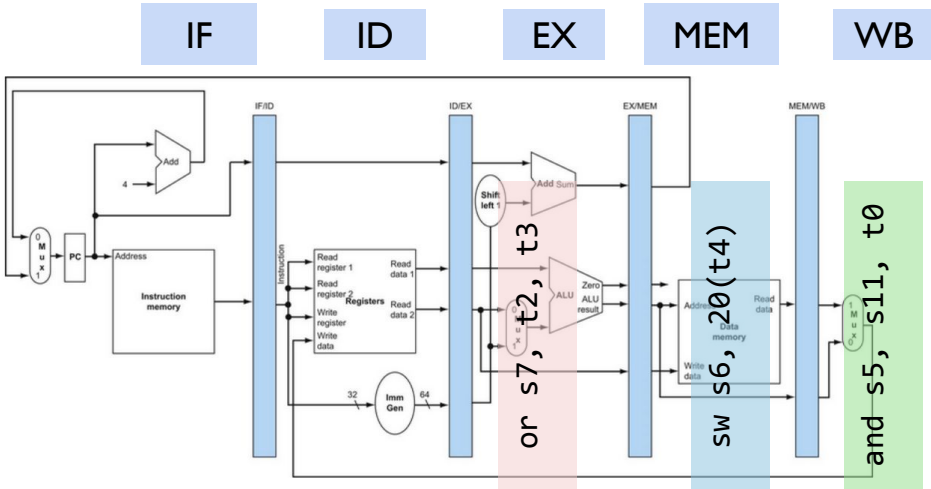
sub s4, t1, s8 - retired!

and s5, s11, t0

sw s6, 20(t4)

or s7, t2, t3
```

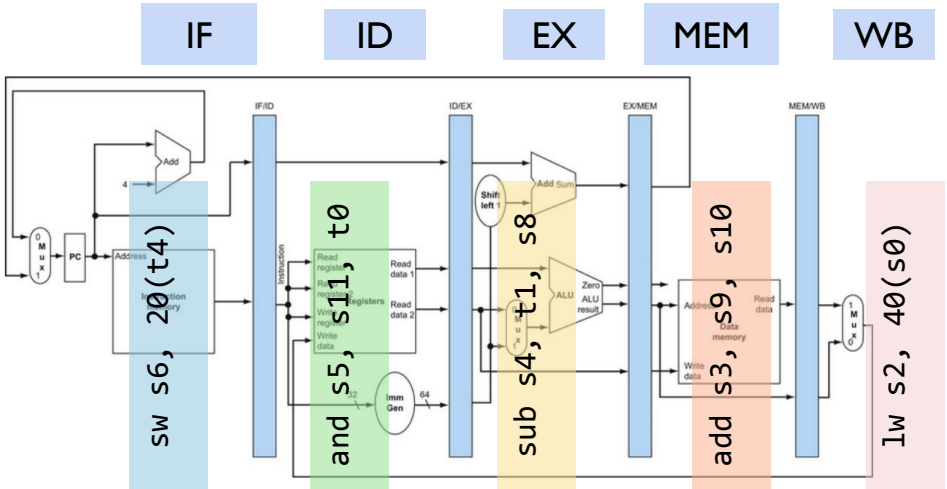
cycle 9



Pipelining in action

- In the steady state, every stage is active.
  - No idle hardware!

cycle 5

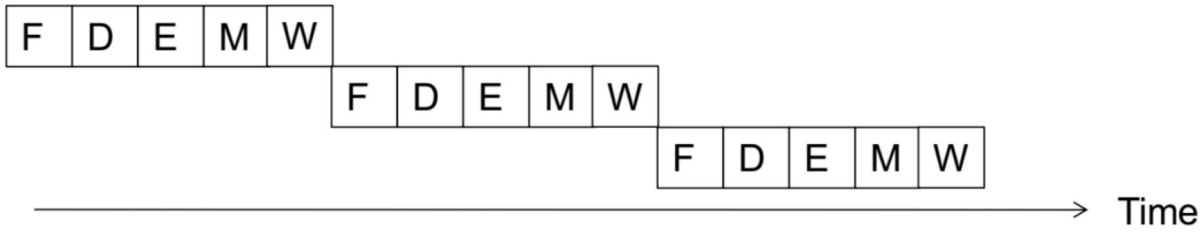


Pipelining improves instruction processing

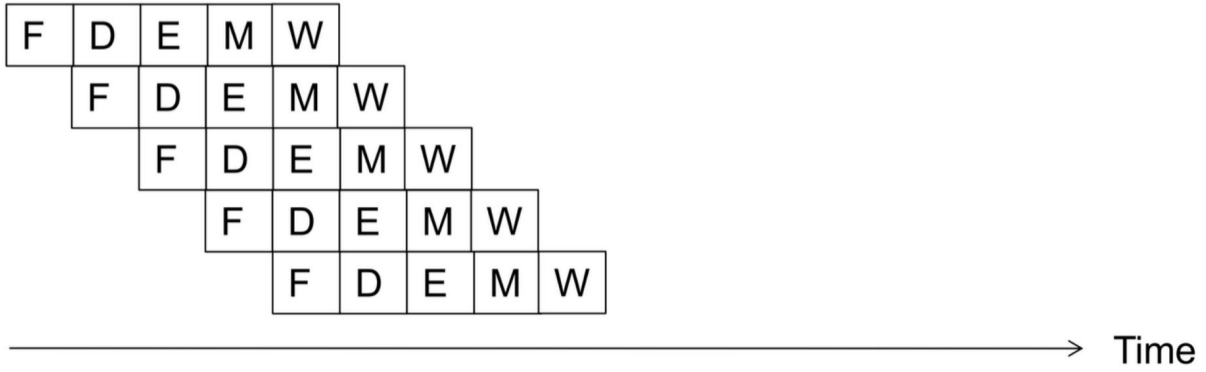
A. Throughput
B. Latency

Multi-cycle vs pipelined

- Multi-cycle: 5 cycles per instruction



- Pipelined: 5 cycles per 5 instructions (in steady state)



Pipelining and the performance equation

$ET = IC \times CPI \times CT$

- Instruction count
  - Determined by ISA and compiler

- CPI
  - Without pipelining, CPI = 5
  - With pipelining, CPI = 1, in the best case 1 instruction starts and finishes execution each cycle

- CT
  - Set by critical path in the slowest stage
  - Pipelining does not change much here.

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?

Is this instruction execution possible?

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

Is this instruction execution possible?

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

Is this instruction execution possible?

(1) When does s8 reg for add get updated?

	Cycle							
	1	2	3	4	5	6	7	8
add s8, s4, s5	F	D	E	M	W			
sub s2, s8, s3		F (A)	D (B)	E (C)	M (D)	W (E)		

(2) When does the s8 reg for sub get read?

F = instruction fetch

D = decode instr & register operand fetch

E = execute

Is this instruction execution possible?

	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		
`or  s9, t6, s8`								

Is this instruction execution possible?

	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		
`or s9, t6, s8`			F	D	E	M	W	

Known as a data dependency!

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: insert nop

- nop = "no operation" or "no op"
- Idea: insert enough nops for result to be ready

Data hazard solution: insert nop

- nop = "no operation" or "no op"
- Idea: insert enough nops for result to be ready

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

Data hazard solution: insert nop

- nop = "no operation" or "no op"
- Idea: insert enough nops for result to be ready
  - What is the CPI?

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

Data hazard solution: insert nop

- nop = "no operation" or "no op"
- Idea: insert enough nops for result to be ready
  - What is the CPI? 4 :(

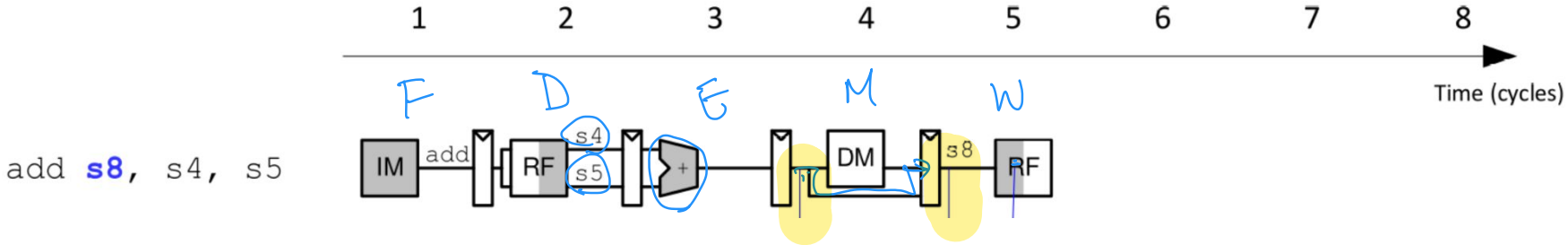
	Cycle							
	1	2	3	4	5	6	7	8
`add s8, s4, s5`	F	D	E	M	W			
`nop`		F	D	E	M	W		
`nop`			F	D	E	M	W	
`sub s2, s8, s3`				F	D	E	M	W

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

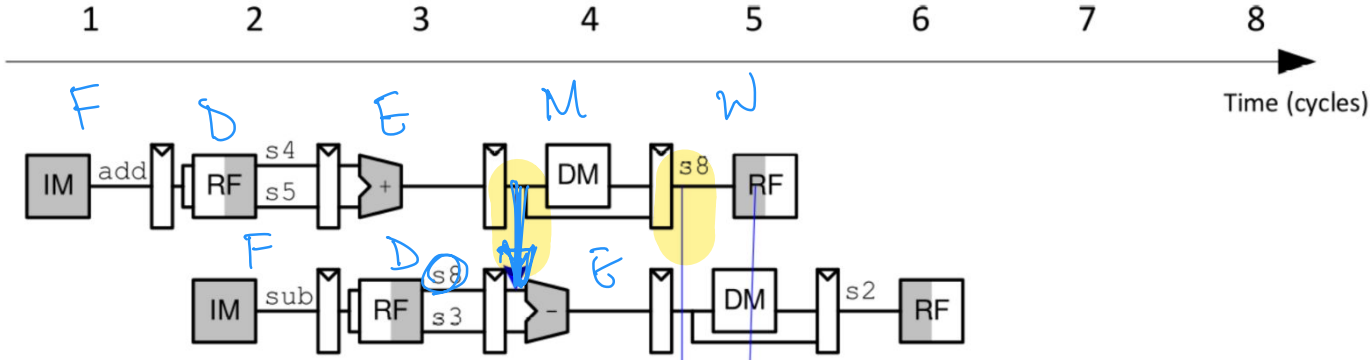
- Observation: Data is available on internal busses in the processor
  - Bus = wires
  - Just not in the register file yet

- 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

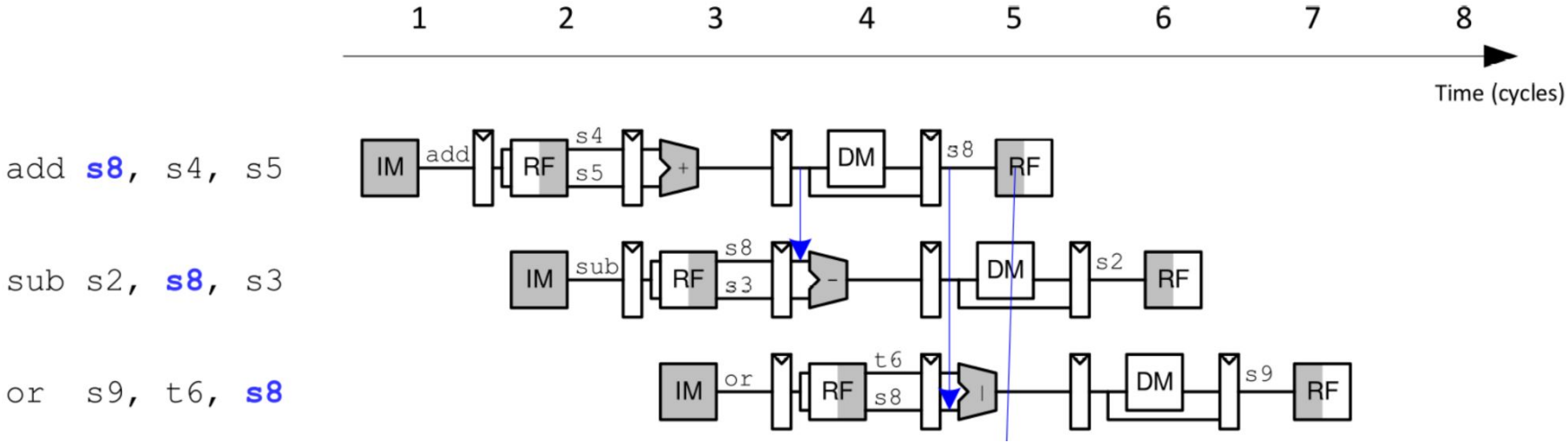
- Idea: Add new connections to bypass the register file
  - i.e., forward data from internal busses to Execute stage
  - Known as data forwarding

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



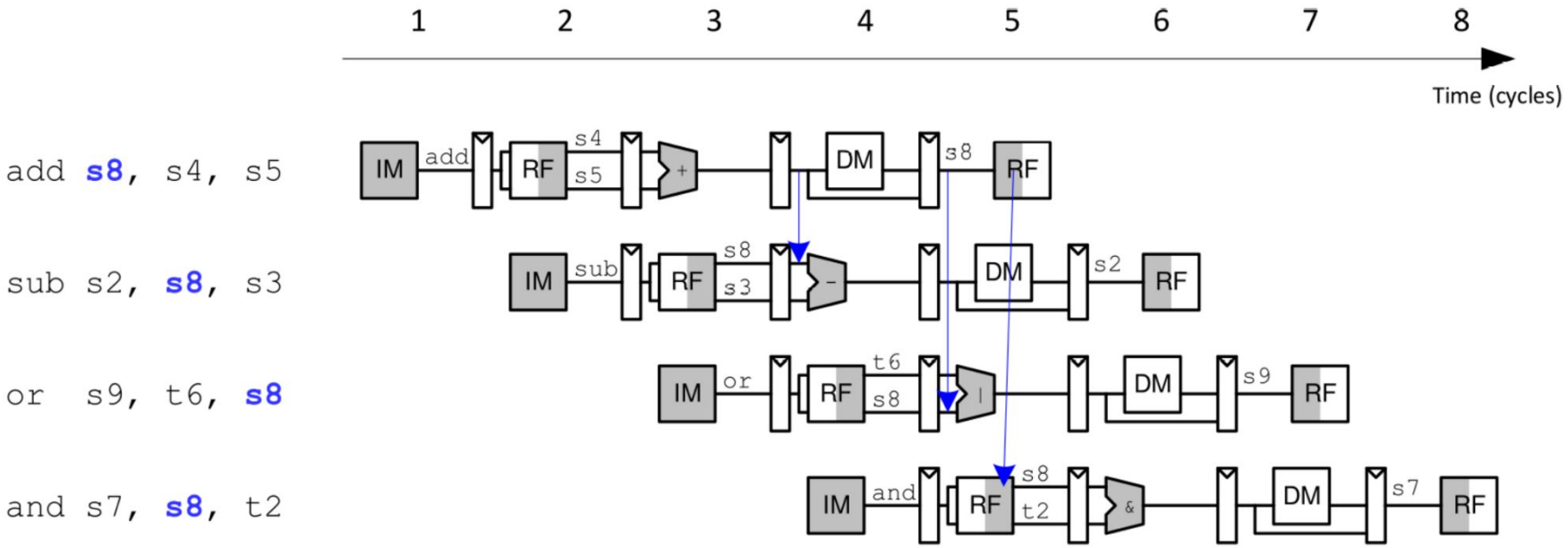
Data hazard solution: data forwarding

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



Data hazard solution: data forwarding

- What is the CPI now?



Data hazard solution: data forwarding

- What is the CPI now? 1 :)

