Poll Q: What affects throughput? Peak throughput depends on…

	Single Cycle	Multi-Cycle	Pipeline
A	Longest Instruction	Cycle Time	Average Instruction
B	Longest Instruction	Cycle Time	Longest Instruction
C	Longest Instruction	Average Instruction	Cycle Time
D	Average Instruction	Longest Instruction	Cycle Time
E	*None of the above*		

Poll Q: What affects throughput? Peak throughput depends on...

	Single Cycle	Multi-Cycle	Pipeline
C	Longest Instruction	Average Instruction	Cycle Time

Throughput is useful work over time – one measure: insts / sec

$$\boxed{\overset{s}{\text{ET}} = \overset{inst}{\text{Inst}} * \underline{\text{CPI * CT}}} \rightarrow \frac{s}{inst}$$

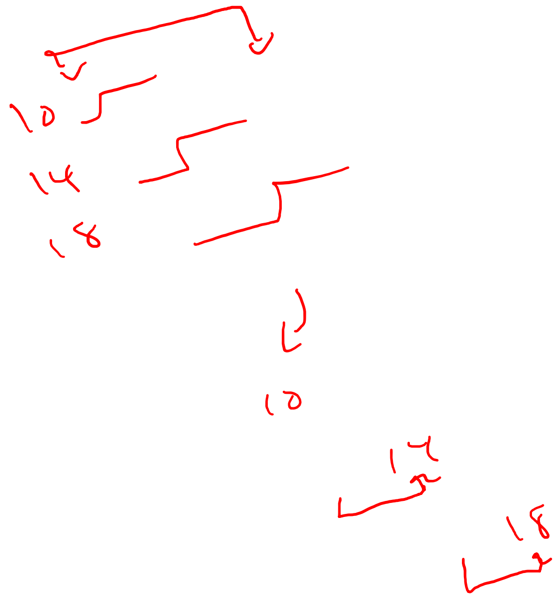
Single Cycle: ET = Inst * $\underline{1}$ * $\boxed{\text{BIG}}$

Multi Cycle: ET = Inst * [3 .. 5] * CT ↑↓

Pipeline: ET = Inst * ~~1~~ 2 5 * CT ↑↓

Pipelining in Modern CPUs

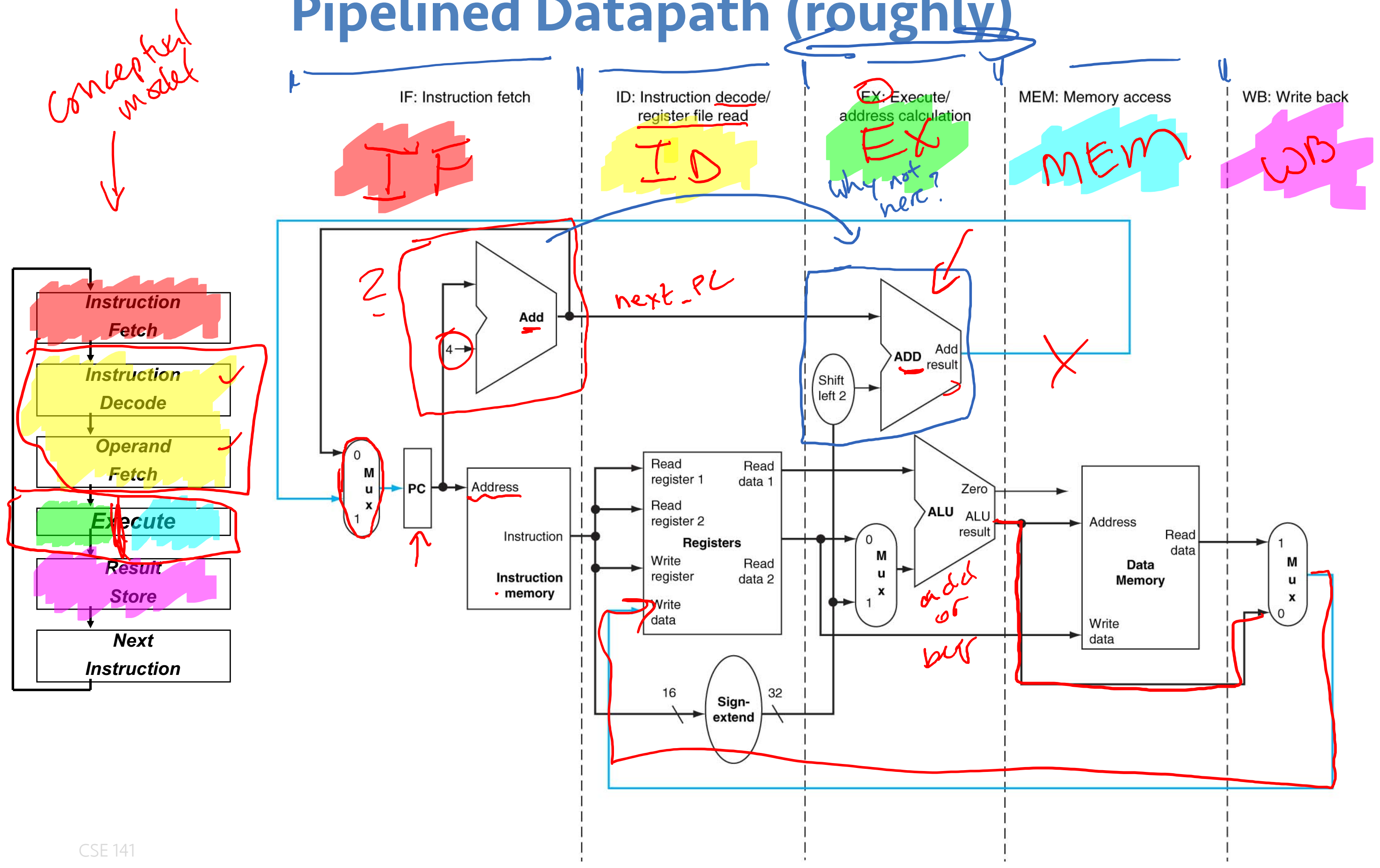
- CPU Datapath
- Arithmetic Units – pipelined multiplier
- System Buses
- Software (at multiple levels)
- etc…



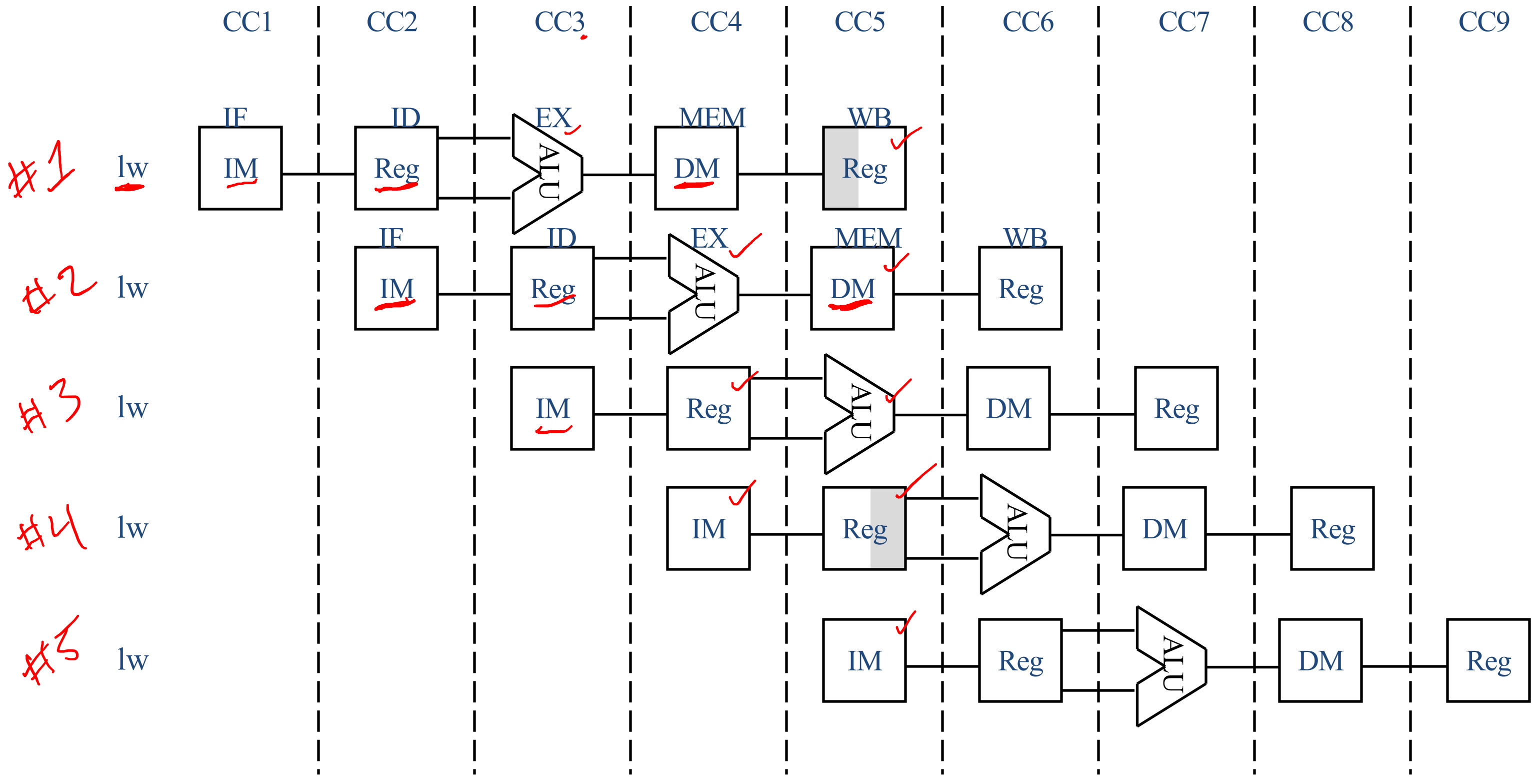
A Pipelined Datapath

IF	Instruction fetch
ID	Instruction decode and register fetch
EX	Execution and effective address calculation
MEM	Memory access
WB	Write back

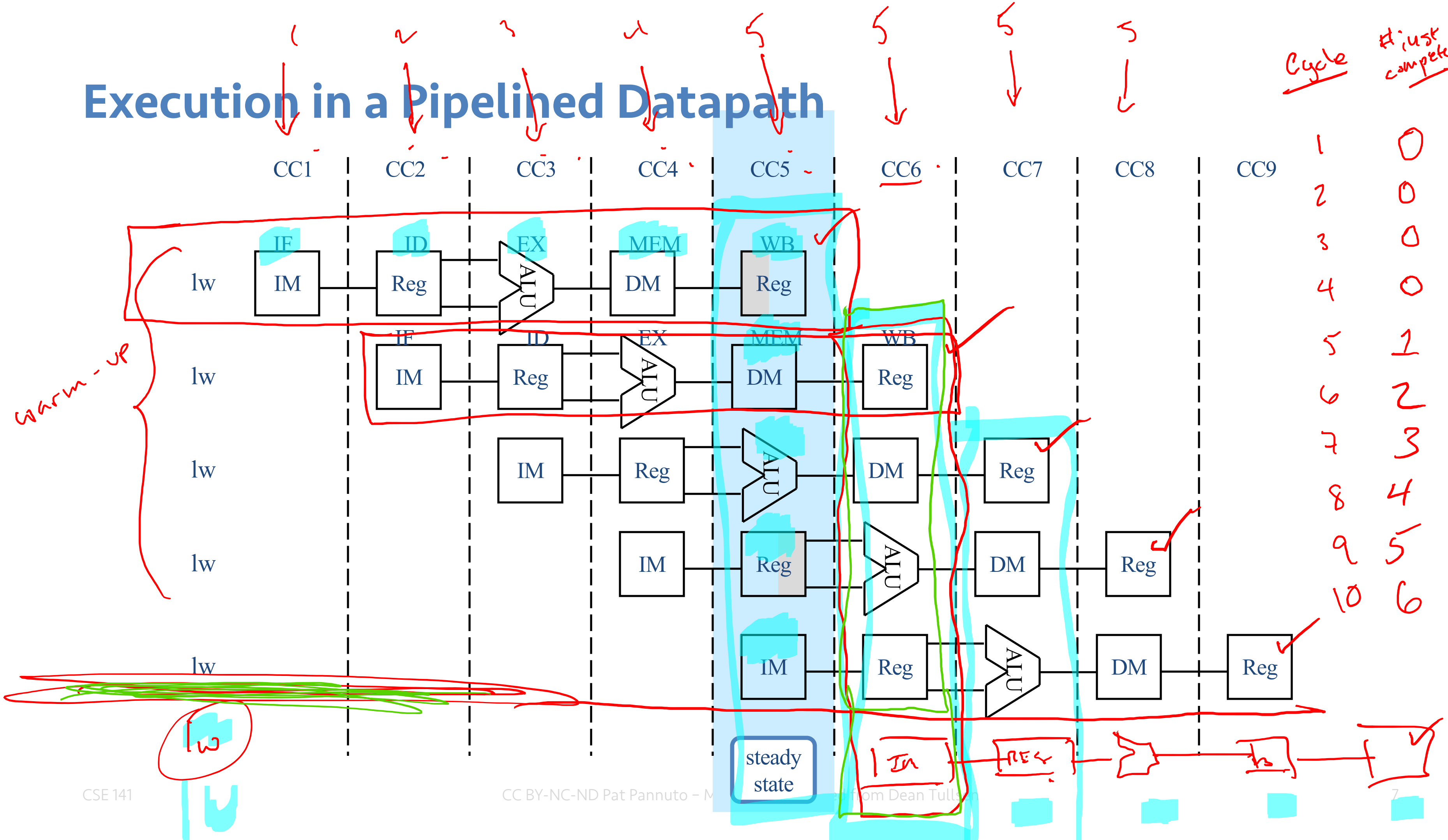
Pipelined Datapath (roughly)



Execution in a Pipelined Datapath

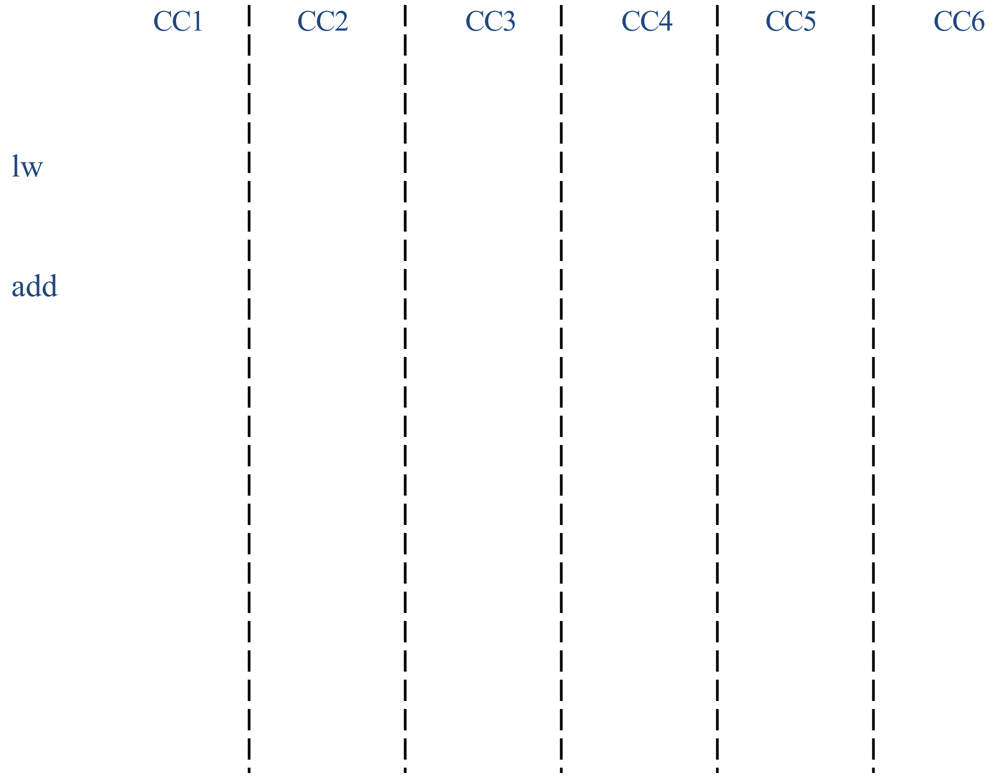


Execution in a Pipelined Datapath

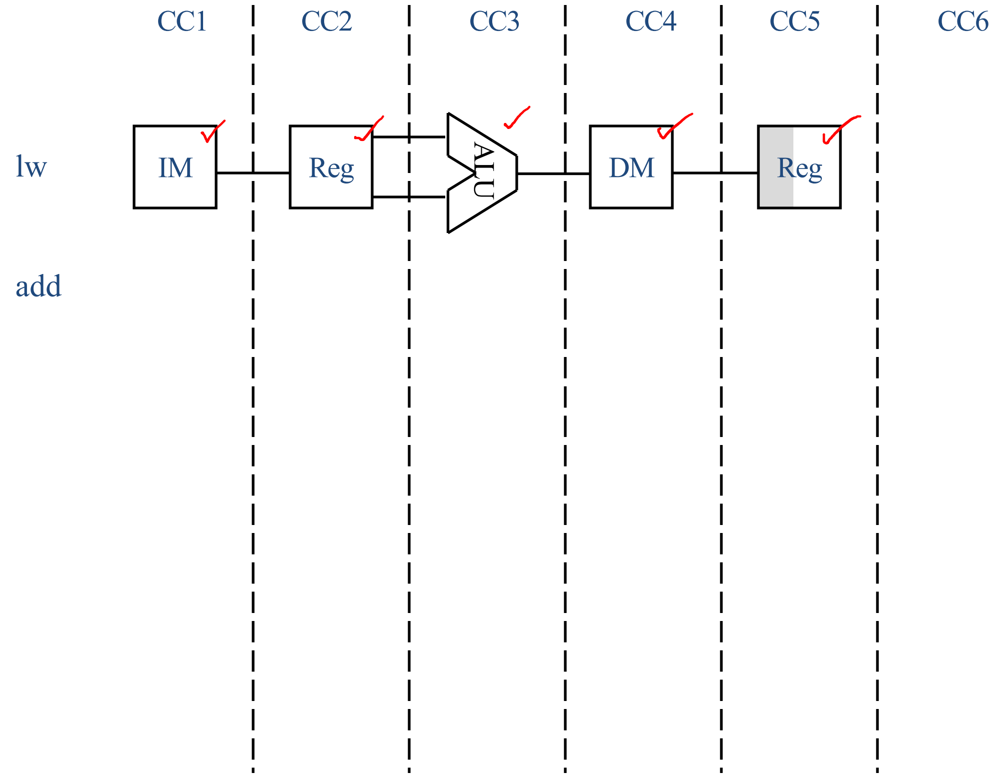


Cycle	# inst completed
1	0
2	0
3	0
4	0
5	1
6	2
7	3
8	4
9	5
10	6

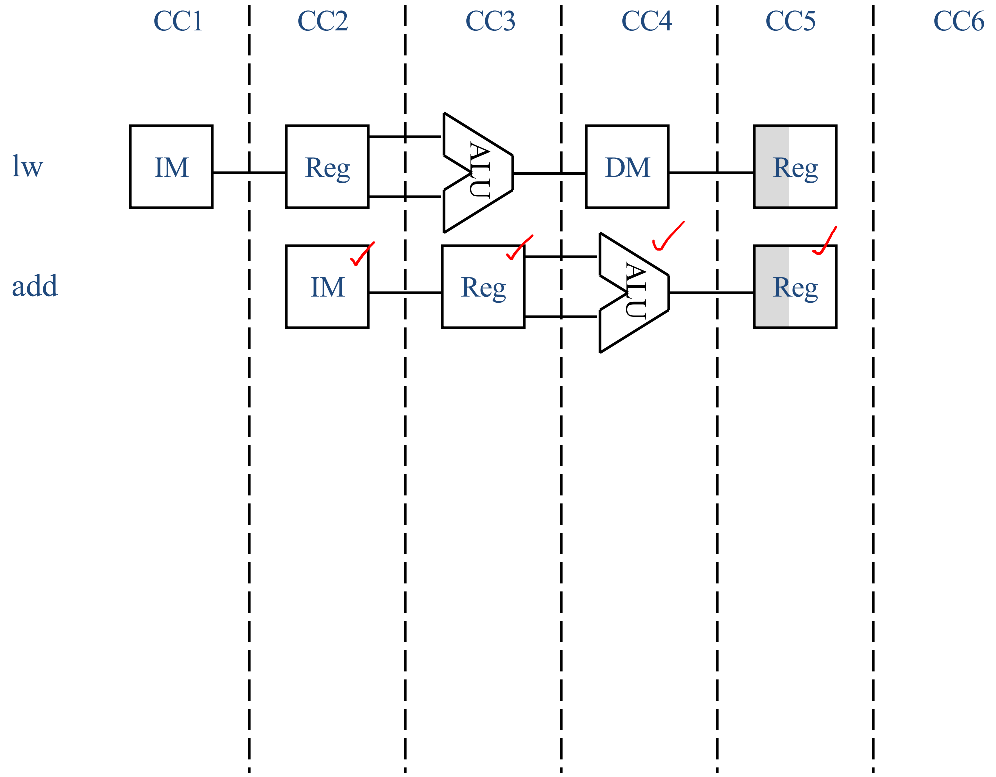
Mixed Instructions in the Pipeline



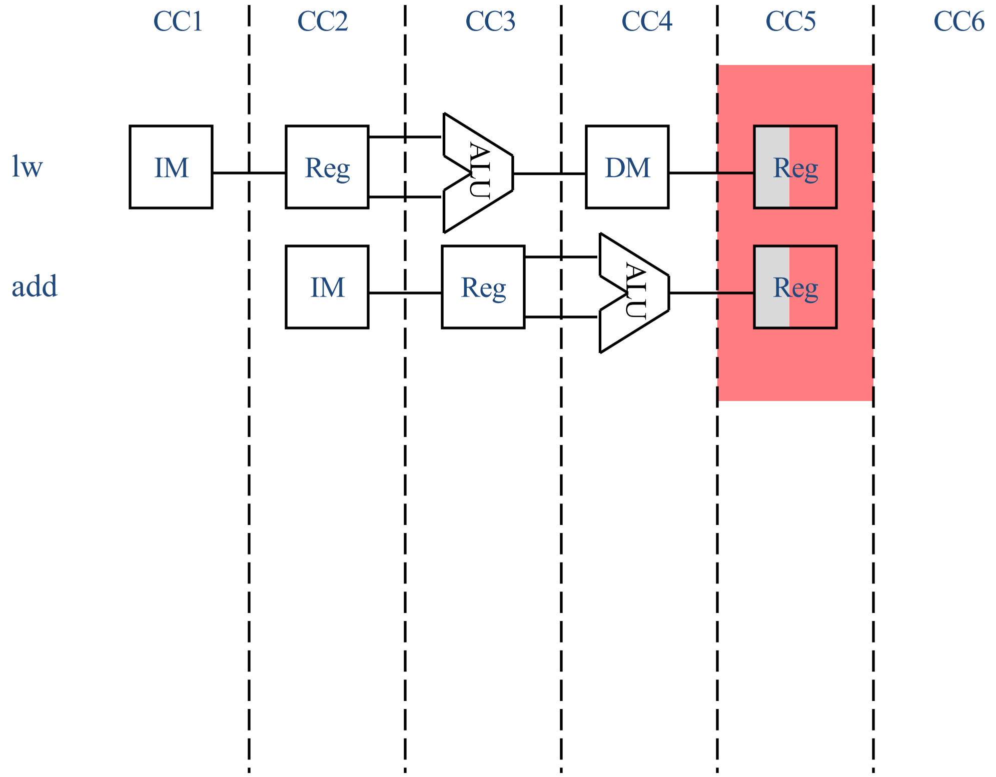
Mixed Instructions in the Pipeline



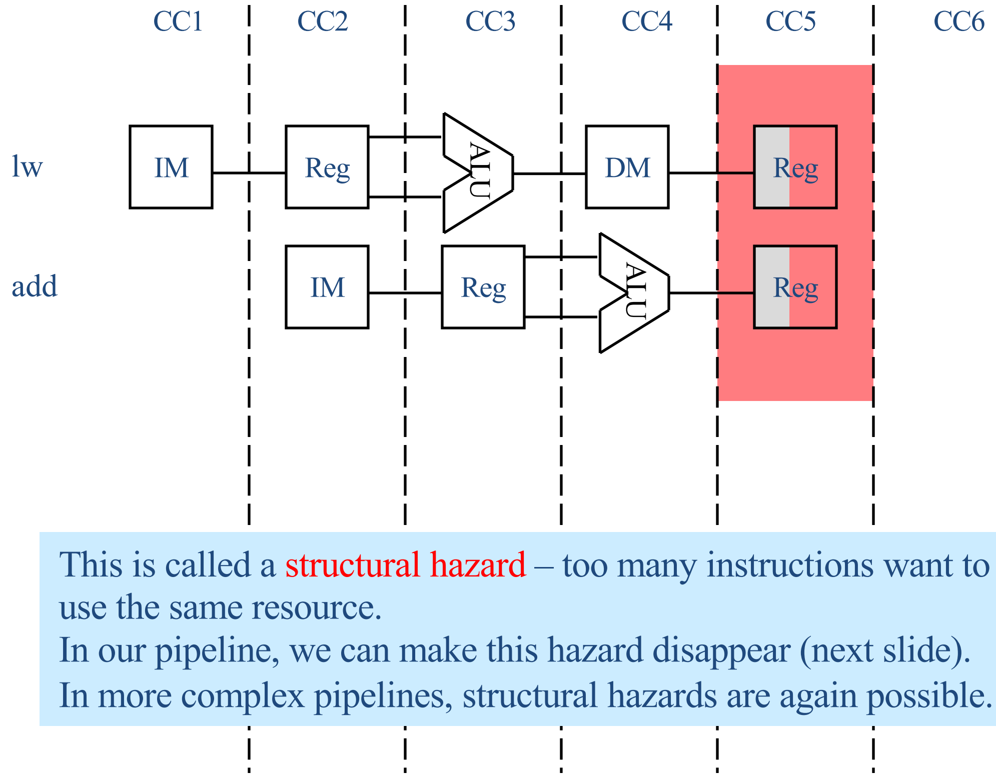
Mixed Instructions in the Pipeline



Mixed Instructions in the Pipeline



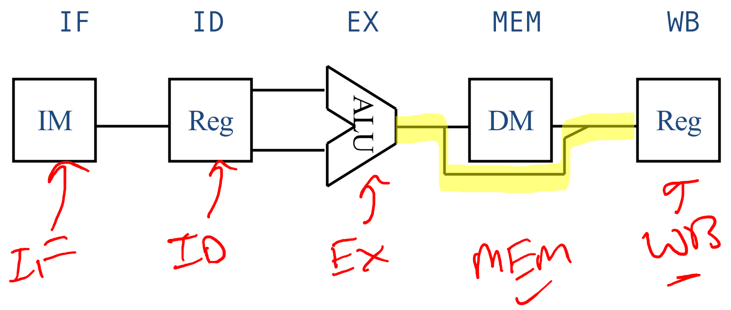
Mixed Instructions in the Pipeline



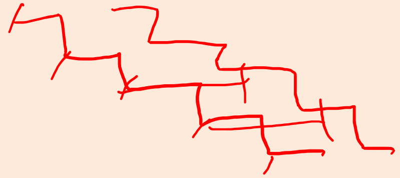
This is called a structural hazard – too many instructions want to use the same resource.
In our pipeline, we can make this hazard disappear (next slide).
In more complex pipelines, structural hazards are again possible.

Pipeline Principles

- All instructions that share a pipeline should have the same *stages* in the same *order*.
  - therefore, *add* does nothing during Mem stage
  - *sw* does nothing during WB stage
- All intermediate values must be latched each cycle.



Pipeline stages



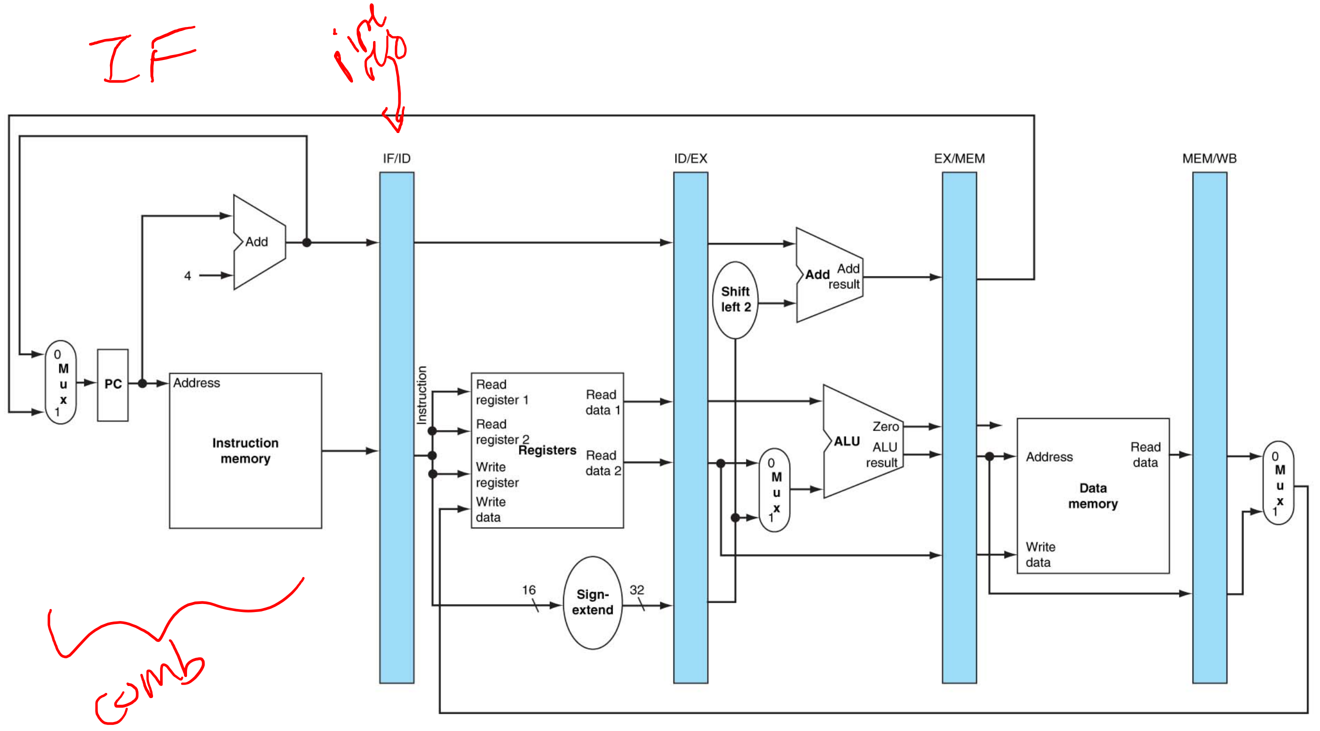
Cycle	#inst
1	0
2	0
3	0
4	0
5	1
6	2
7	3

- What is the performance implication of making every instruction go through all 5 stages? (e.g., instead of 4 for add, 3 for beq, etc.)

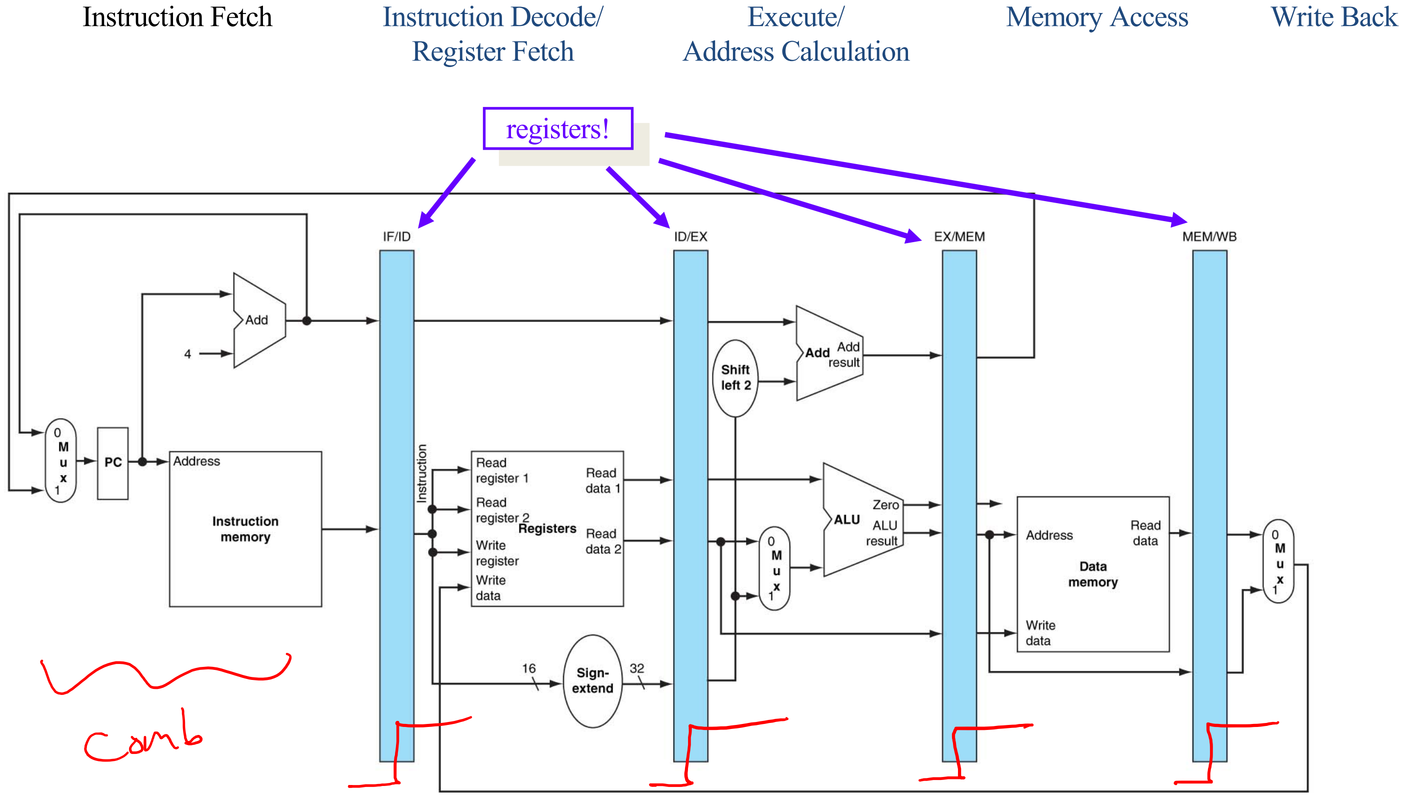
	(Choose BEST answer)	
25%	A	~~Decreases peak throughput by 20%~~
20%	B	~~Increases program latency by 20%~~
5%	[✓ C]	No significant impact on peak throughput or program latency
50%	D	~~Depends on how many R-type instructions, beq, etc.~~
	E	None of the above

Pipelined Datapath

Instruction Fetch | Instruction Decode/ Register Fetch | Execute/ Address Calculation | Memory Access | Write Back



Pipelined Datapath



Instruction Fetch
Instruction Decode/ Register Fetch
Execute/ Address Calculation
Memory Access
Write Back
registers!
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
comb