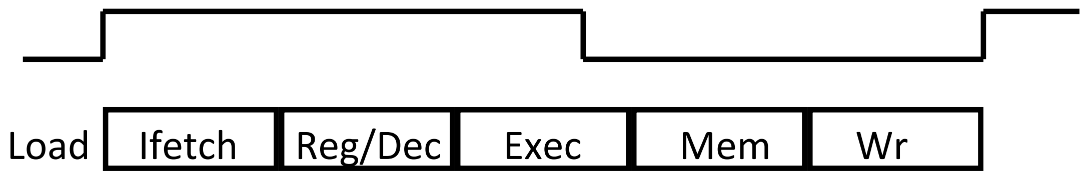
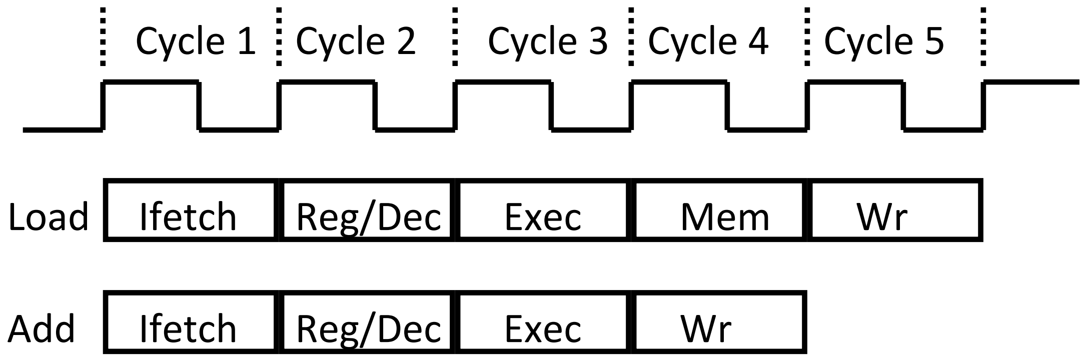
Review -- Instruction Latencies

•Single-Cycle CPU

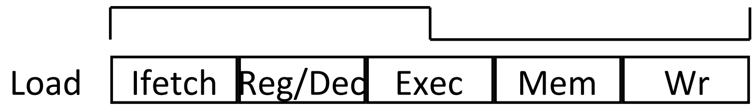


•Multiple Cycle CPU

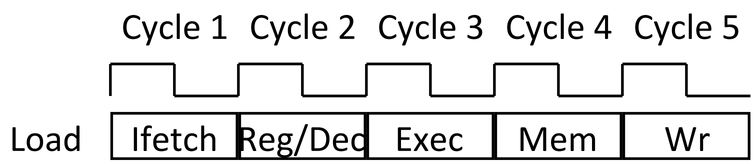


Instruction Latencies and Throughput

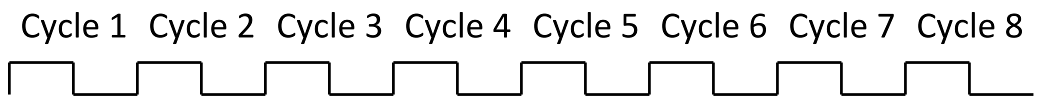
- **Single-Cycle CPU**



- **Multiple Cycle CPU**



- **Pipelined CPU**



Instruction Latencies and Throughput

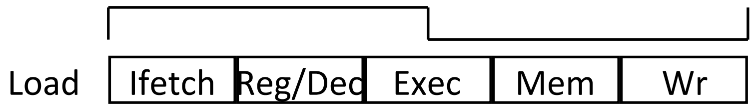
Which of the statements below is true about a pipelined processor?

Selection	Statement
A	Instruction latency remains essentially unchanged from single-cycle (minus some overheads); Instruction throughput increases
B	Instruction latency remains essentially unchanged from multi-cycle (minus some overheads); Instruction throughput increases
C	Instruction latency improves by a factor of 5 over single-cycle (minus some overheads); Instruction throughput increases
D	Instruction latency improves by a factor of 5 over multi-cycle (minus some overheads); Instruction throughput increases
E	None of the above

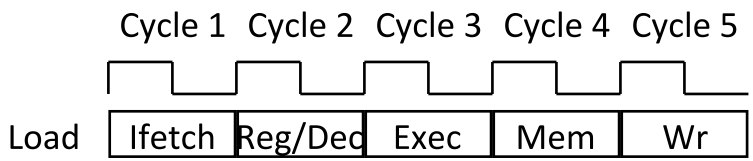
- Answer C

Instruction Latencies and Throughput

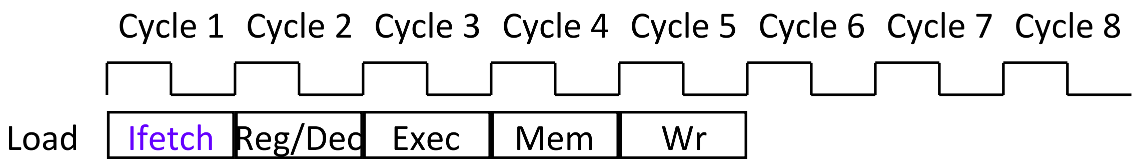
- **Single-Cycle CPU**



- **Multiple Cycle CPU**

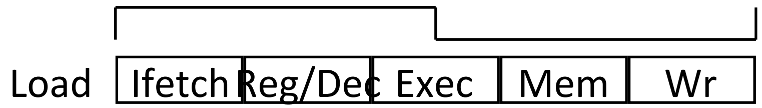


- **Pipelined CPU**

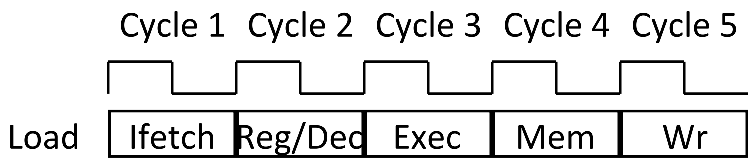


Instruction Latencies and Throughput

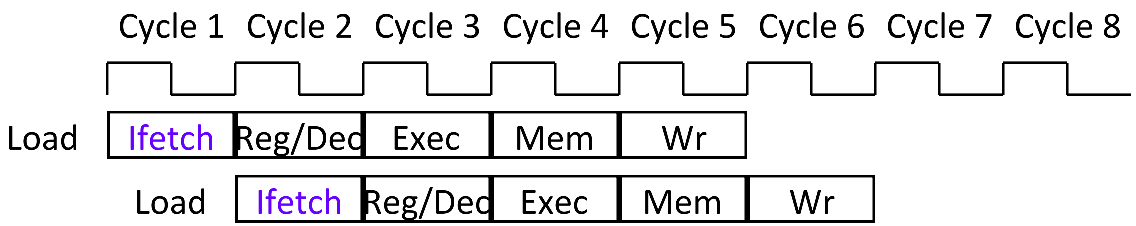
- **Single-Cycle CPU**



- **Multiple Cycle CPU**

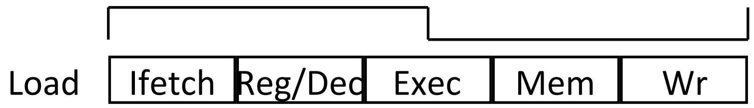


- **Pipelined CPU**

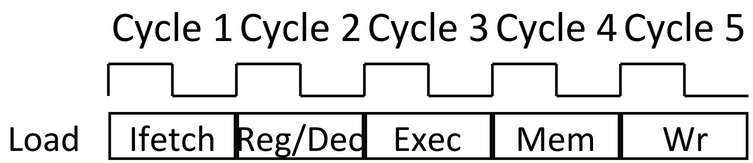


Instruction Latencies and Throughput

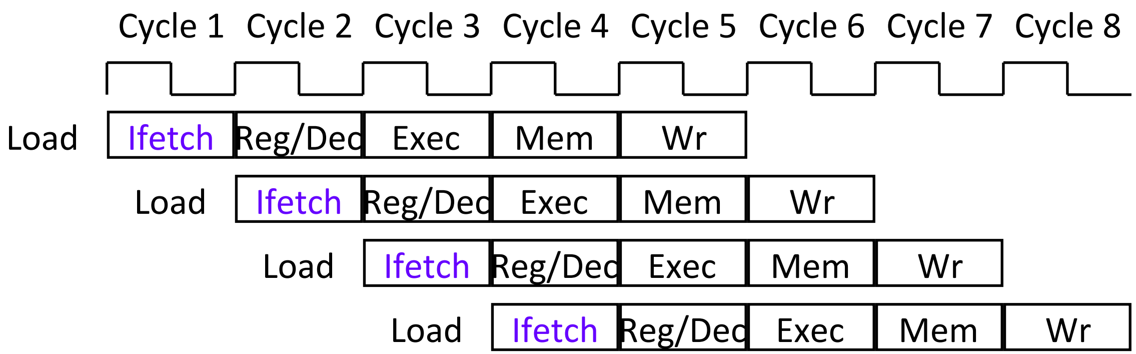
- **Single-Cycle CPU**



- **Multiple Cycle CPU**



- **Pipelined CPU**



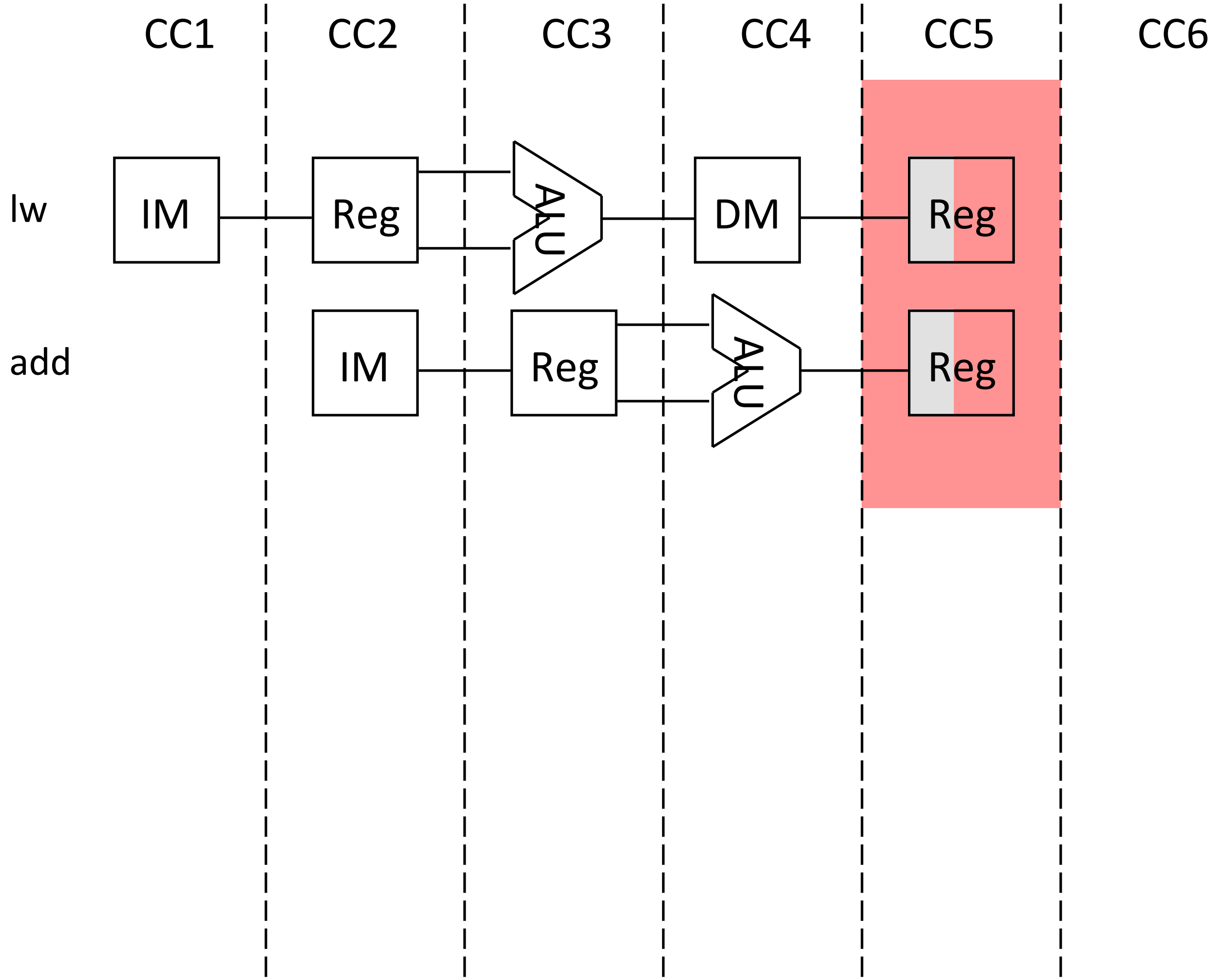
Pipeline Stages

Should we force every instruction to go through all 5 stages? Can we break it up like we did for multi-cycle, with R-type taking 4 cycles instead of 5?

Selection	Yes/No	Reason (Choose BEST answer)
A	Yes	Decreasing R-type to 4 cycles improves instruction throughput
B	Yes	Decreasing R-type to 4 cycles improves instruction latency
C	No	Decreasing R-type to 4 cycles causes hazards
D	No	Decreasing R-type to 4 cycles causes hazards and doesn't impact throughput
E	No	Decreasing R-type to 4 cycles causes hazards and doesn't impact latency

- Answer C

Mixed Instructions in the Pipeline



Pipeline Principles

- All instructions that share a pipeline must have the same *stages* in the same *order*.
  - therefore, *add* does nothing during Mem stage
  - *sw* does nothing during WB stage

Suppose EX is the longest (in time) pipeline stage. To reduce CT, we split it in half. Given the following pipeline:

IF ID EX1 EX2 M WB

Assume the input data must be available at the start of EX1 and the output is available after EX2. How many hardware stalls would be required in the following code (assuming hardware forwarding wherever possible)?

add r1, r2, r3
add r4, r1, r3

Selection	Number of stalls
A	0
B	1
C	2
D	3
E	4

- Answer B

Suppose EX is the longest (in time) pipeline stage. To reduce CT, we split it in half. Given the following pipeline:

IF ID EX1 EX2 M WB

Assume the input data must be available at the start of EX1 and the output is available after EX2.How many hardware stalls would be required in the following code (assuming hardware forwarding wherever possible)?

```
lw r1, 0(r3)
add r2, r1, r3
```

Selection	Number of stalls
A	0
B	1
C	2
D	3
E	4

- Answer C