CMPE 411
Computer Architecture

Lecture 13

Introduction to Pipelining

October 12, 2017

www.csee.umbc.edu/~younis/CMPE411/CMPE411.htm

Lecture's Overview

❑ *Previous Lecture*

➔ Micro-programmed control
- PLA versus ROM based control unit design
- Horizontal versus vertical micro-coding
- Designing a micro-instruction set

➔ Processor exceptions
- Exceptions are the hardest part of control
- MIPS interrupts and exceptions support
- Detecting exceptions by the control unit

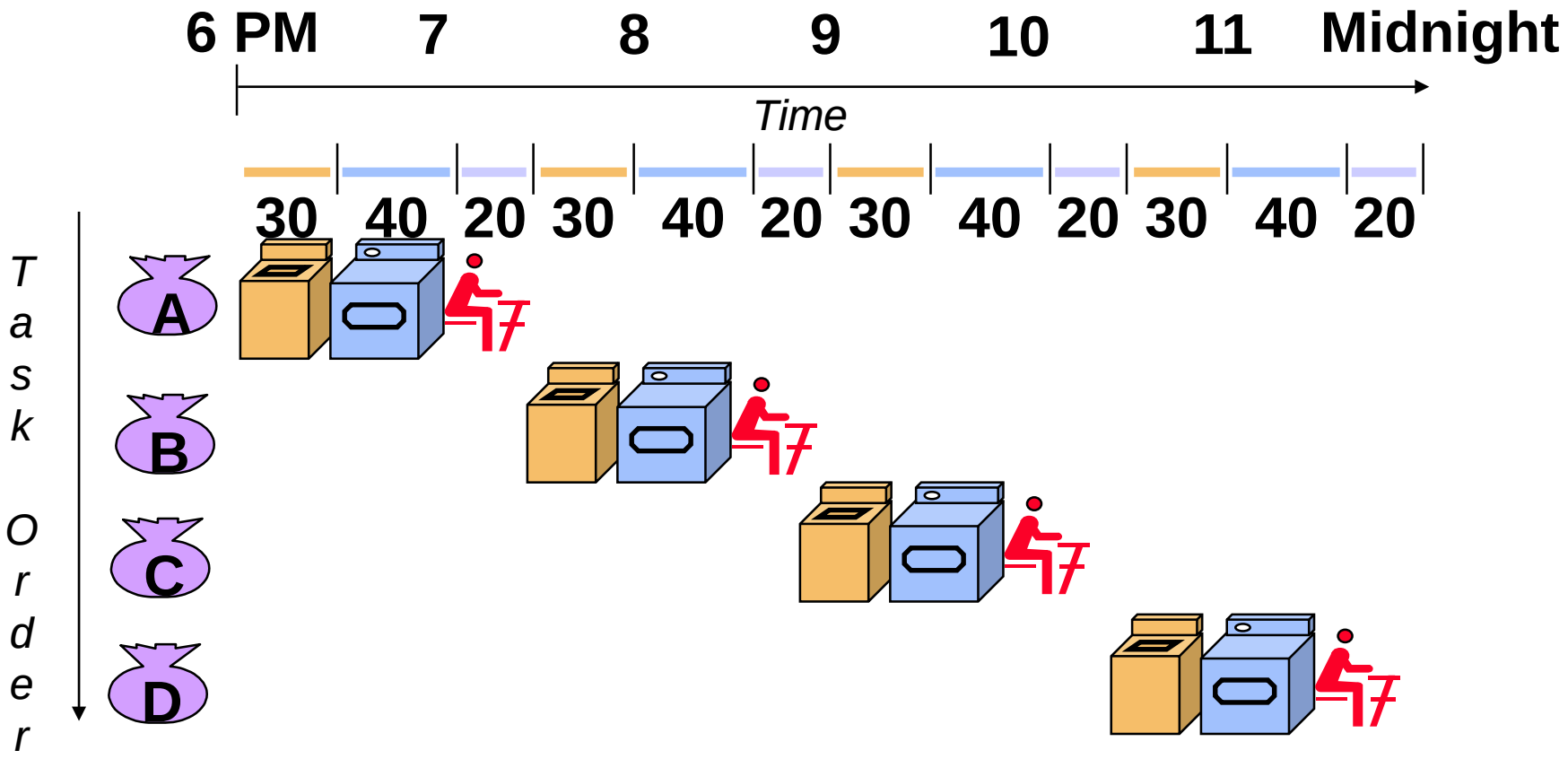
❑ *This Lecture*

➔ An overview of Pipelining

➔ A pipelined datapath

➔ Pipelined control

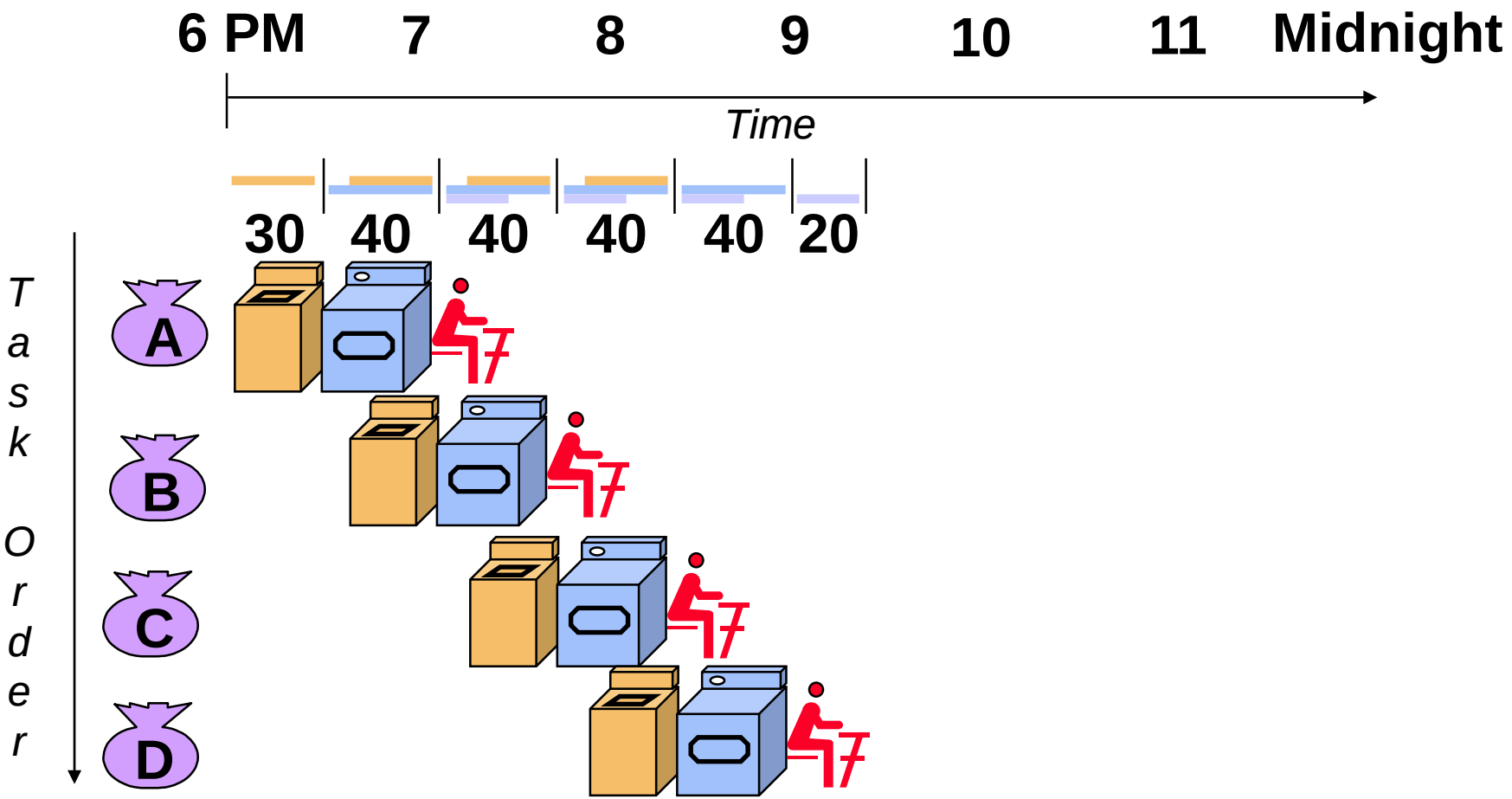
Sequential Laundry



- Washer takes 30 min, Dryer takes 40 min, folding takes 20 min
- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

* Slide is courtesy of Dave Patterson

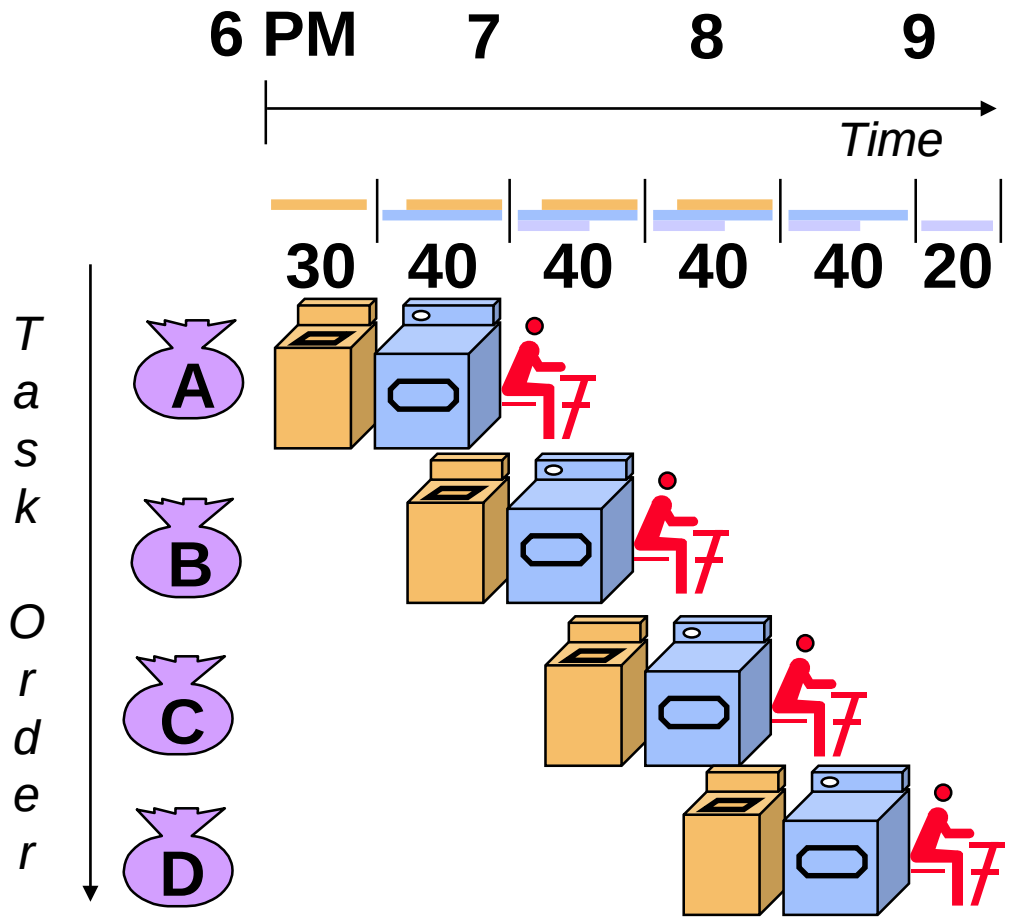
Pipelined Laundry



- Pipelining means start work as soon as possible
- Pipelined laundry takes 3.5 hours for 4 loads

* Slide is courtesy of Dave Patterson

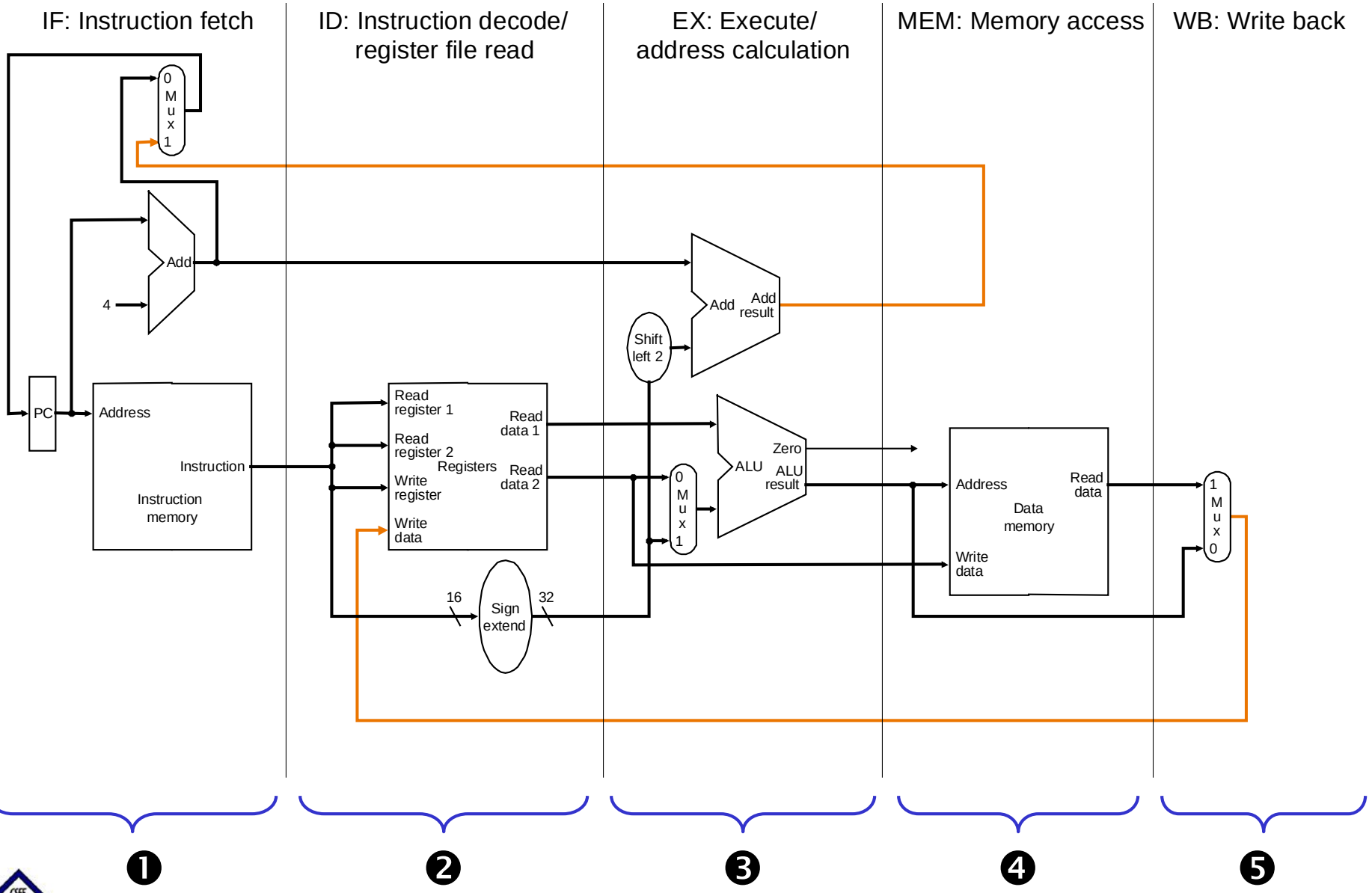
Pipelining Lessons



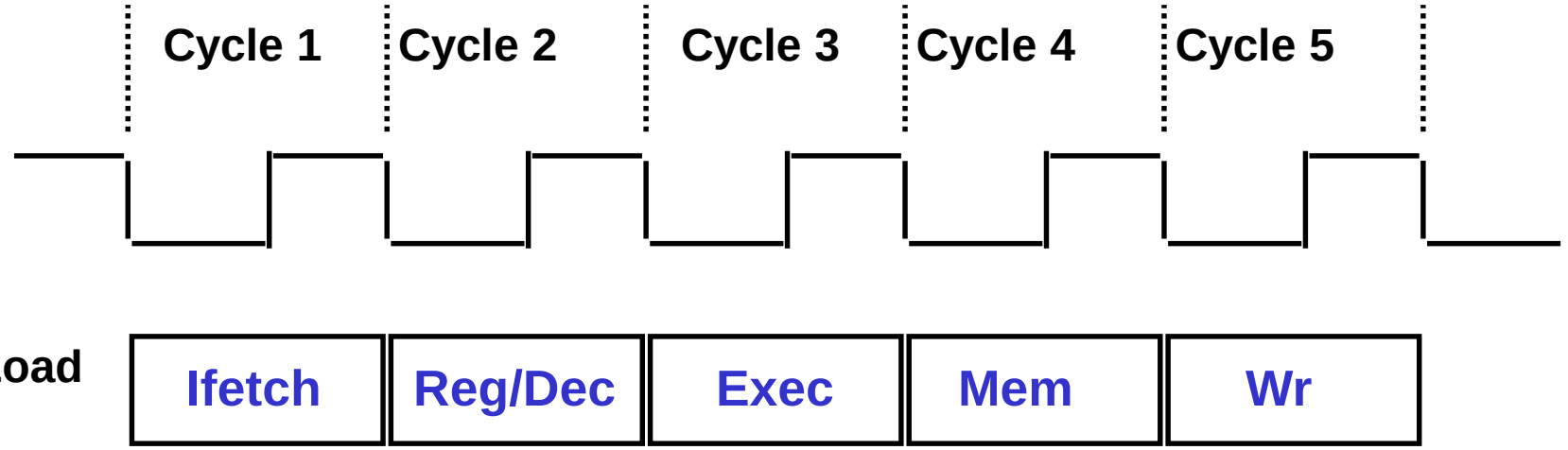
- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduce speedup
- Stall for Dependencies

* Slide is courtesy of Dave Patterson

Multi-cycle Instruction Execution



Stages of Instruction Execution

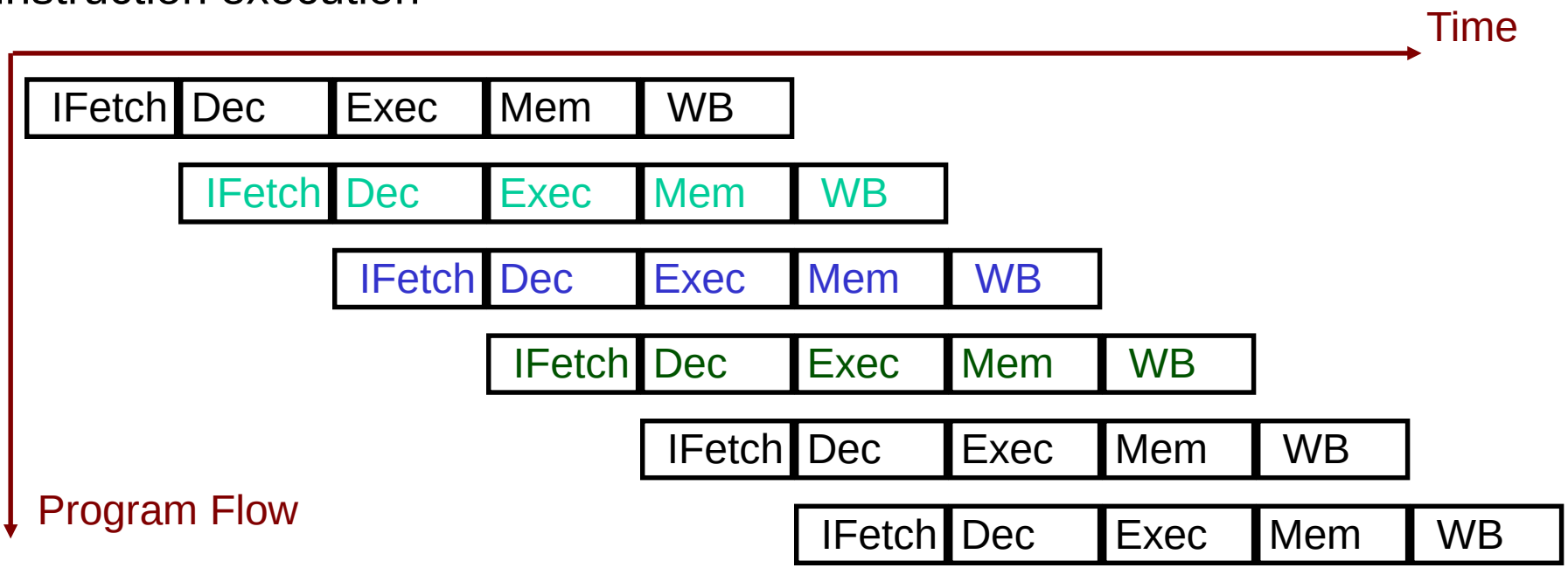


- The load instruction is the longest
- All instructions follows at most the following five steps:
  - ➔ Ifetch: Instruction Fetch
    - Fetch the instruction from the Instruction Memory
  - ➔ Reg/Dec: Registers Fetch and Instruction Decode
  - ➔ Exec: Calculate the memory address
  - ➔ Mem: Read the data from the Data Memory
  - ➔ Wr: Write the data back to the register file

* Slide is courtesy of Dave Patterson

Instruction Pipelining

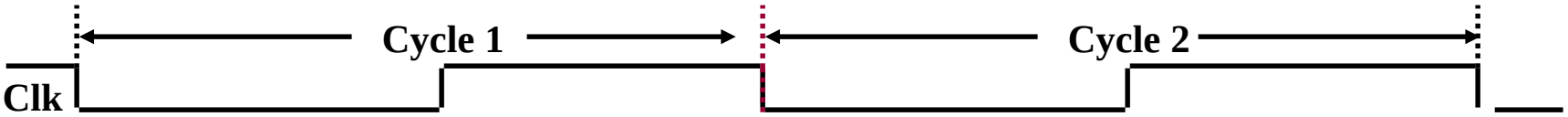
- Start handling of next instruction while the current instruction is in progress
- Pipelining is feasible when different devices are used at different stages of instruction execution



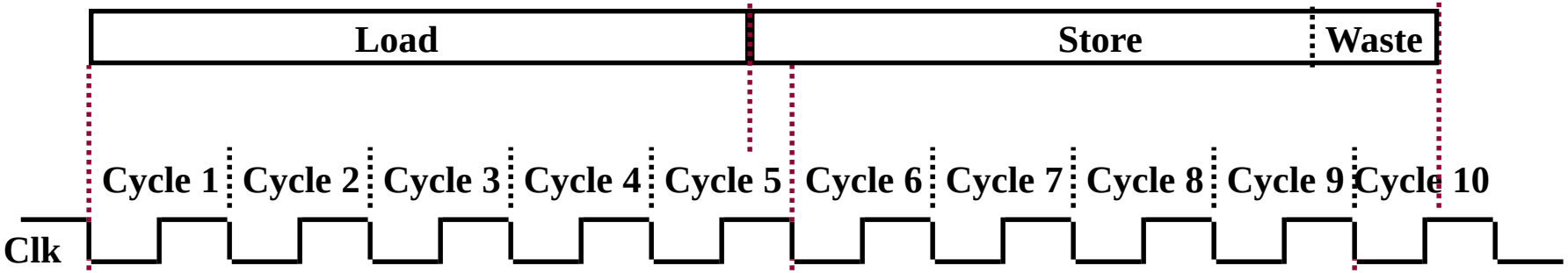
$$\text{Time between instructions}_{pipelined} = \frac{\text{Time between instructions}_{nonpipelined}}{\text{Number of pipe stages}}$$

Pipelining improves performance by increasing instruction throughput

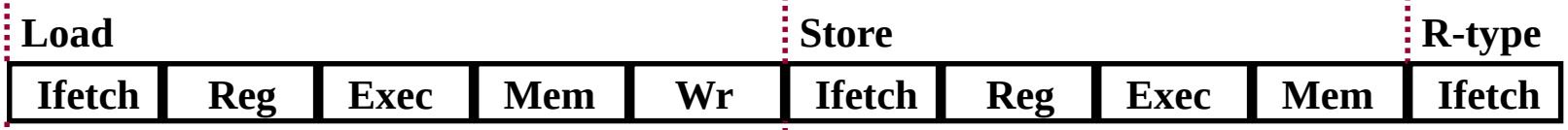
Single Cycle, Multiple Cycle, vs. Pipeline



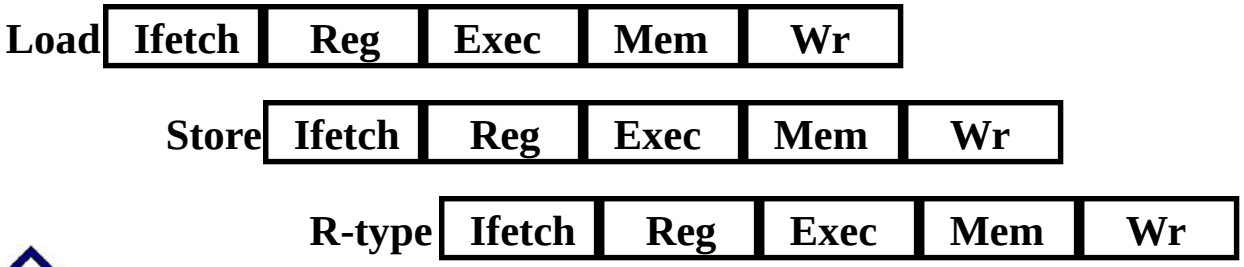
Single Cycle Implementation:



Multiple Cycle Implementation:

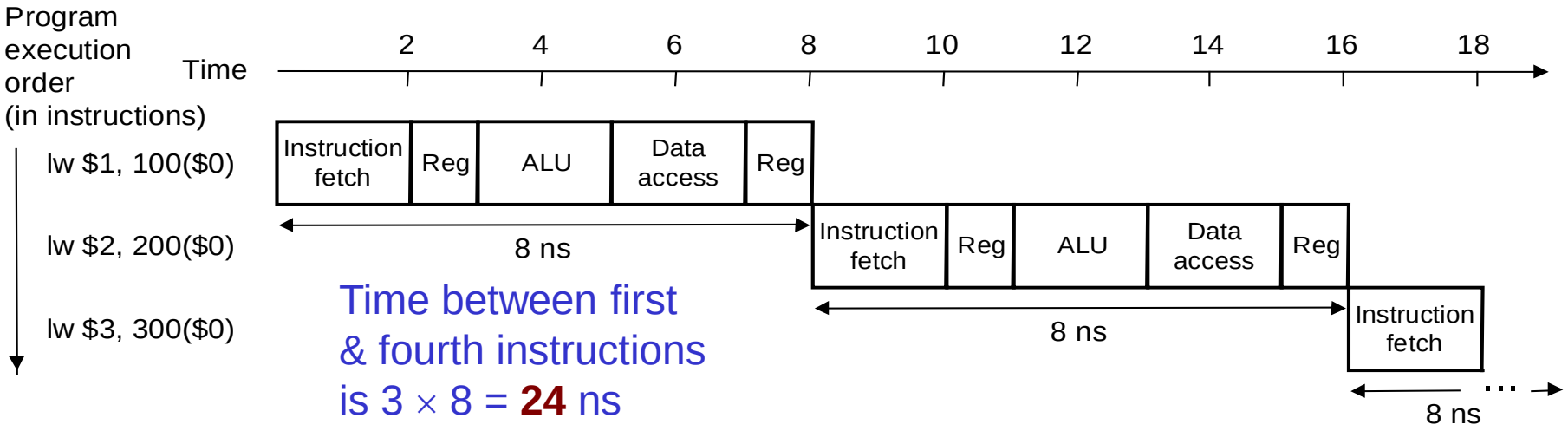


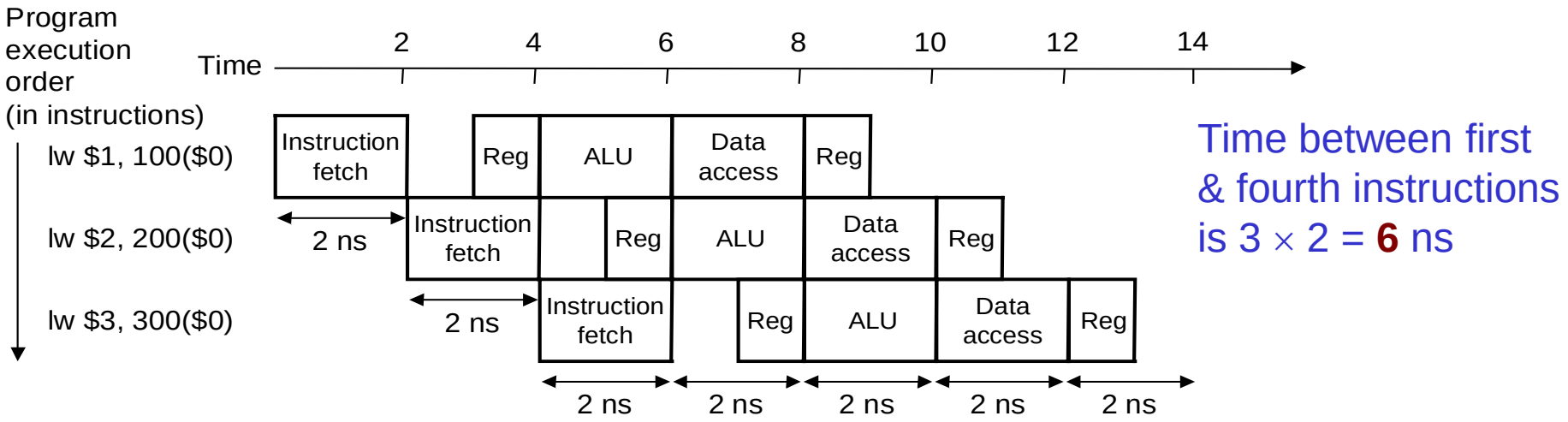
Pipeline Implementation:



* Slide is courtesy of Dave Patterson

Example of Instruction Pipelining





Ideal and upper bound for speedup is number of stages in the pipeline

Pipeline Performance

- Suppose we execute 100 instructions:
  - ➔ Single Cycle Machine
    - 45 ns/cycle x 1 CPI x 100 inst = 4500 ns
  - ➔ Multi-cycle Machine
    - 10 ns/cycle x 4.2 CPI (due to inst mix) x 100 inst = 4200 ns
  - ➔ Ideal 5 stages pipelined machine
    - 10 ns/cycle x (1 CPI x 100 inst + 4 cycle drain) = 1040 ns

- Due to fill and drain effects of a pipeline ideal performance can be achieved only for very large instructions

Example:

a sequence of 1000 load instructions would take 5000 cycles on a multi-cycle machine while taking 1004 on a pipeline machine

$\Rightarrow$ speedup = 5000/1004 $\cong$ 5

Pipeline Hazards

- Pipeline hazards are cases that affect instruction execution semantics and thus need to be detected and corrected
- Hazards types

Structural hazard: attempt to use a resource two different ways at same time

- ➔ E.g., combined washer/dryer would be a structural hazard or folder busy doing something else (watching TV)
- ➔ Single memory for instruction and data

Data hazard: attempt to use item before it is ready

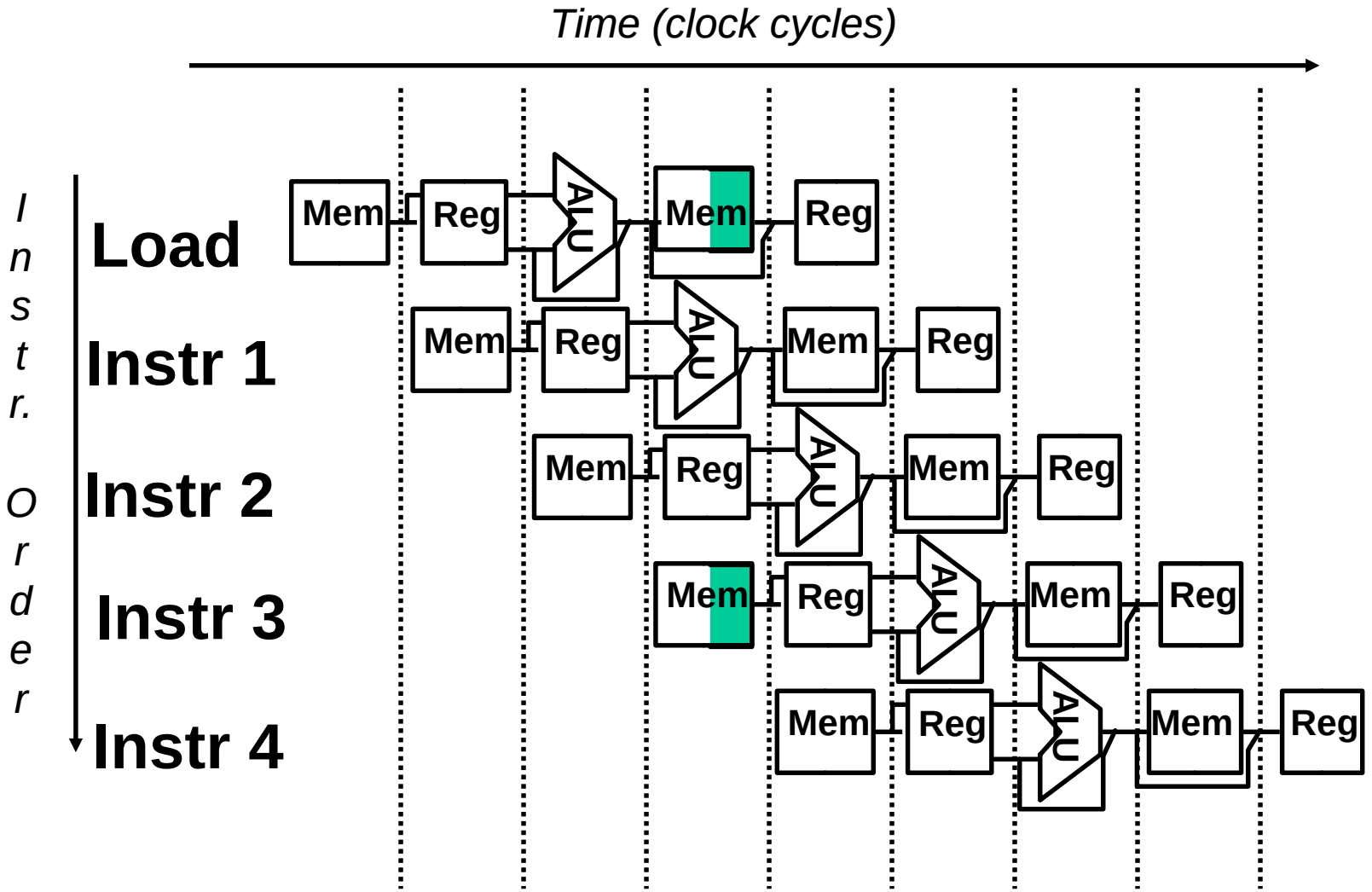
- ➔ E.g., one sock of pair in dryer and one in washer; can't fold until get sock from washer through dryer
- ➔ instruction depends on result of prior instruction still in the pipeline

Control hazard: attempt to make a decision before condition is evaluated

- ➔ E.g., washing football uniforms and need to get proper detergent level; need to see after dryer before next load in
- ➔ branch instructions

- Hazards can always be resolved by waiting

Single Memory is a Structural Hazard

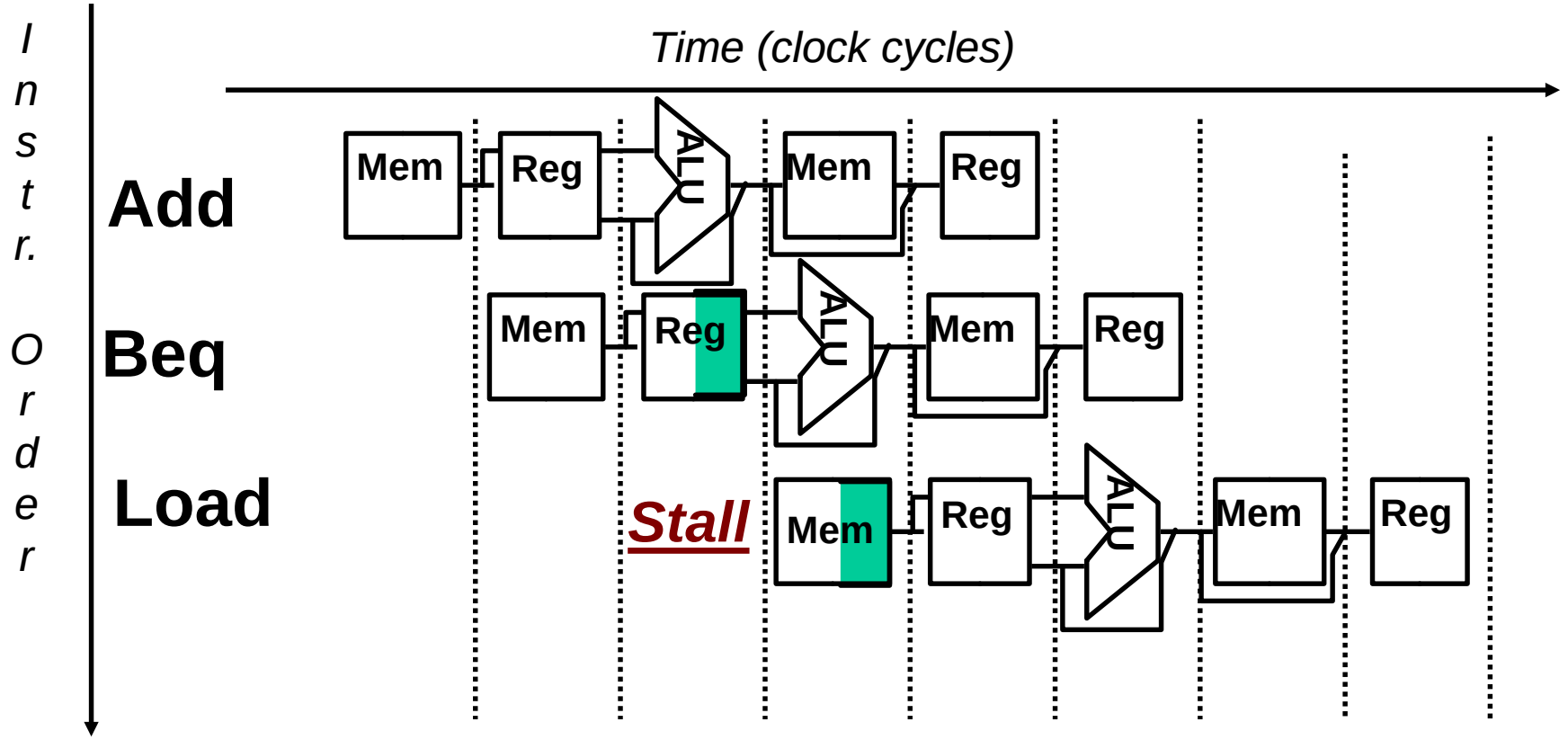


- Can be easily detected
- Resolved by inserting idle cycles

* Slide is courtesy of Dave Patterson

Control Hazard

- Stall: wait until decision is clear
  - ➔ It is possible to move up decision to 2nd stage by adding hardware to check registers as being read

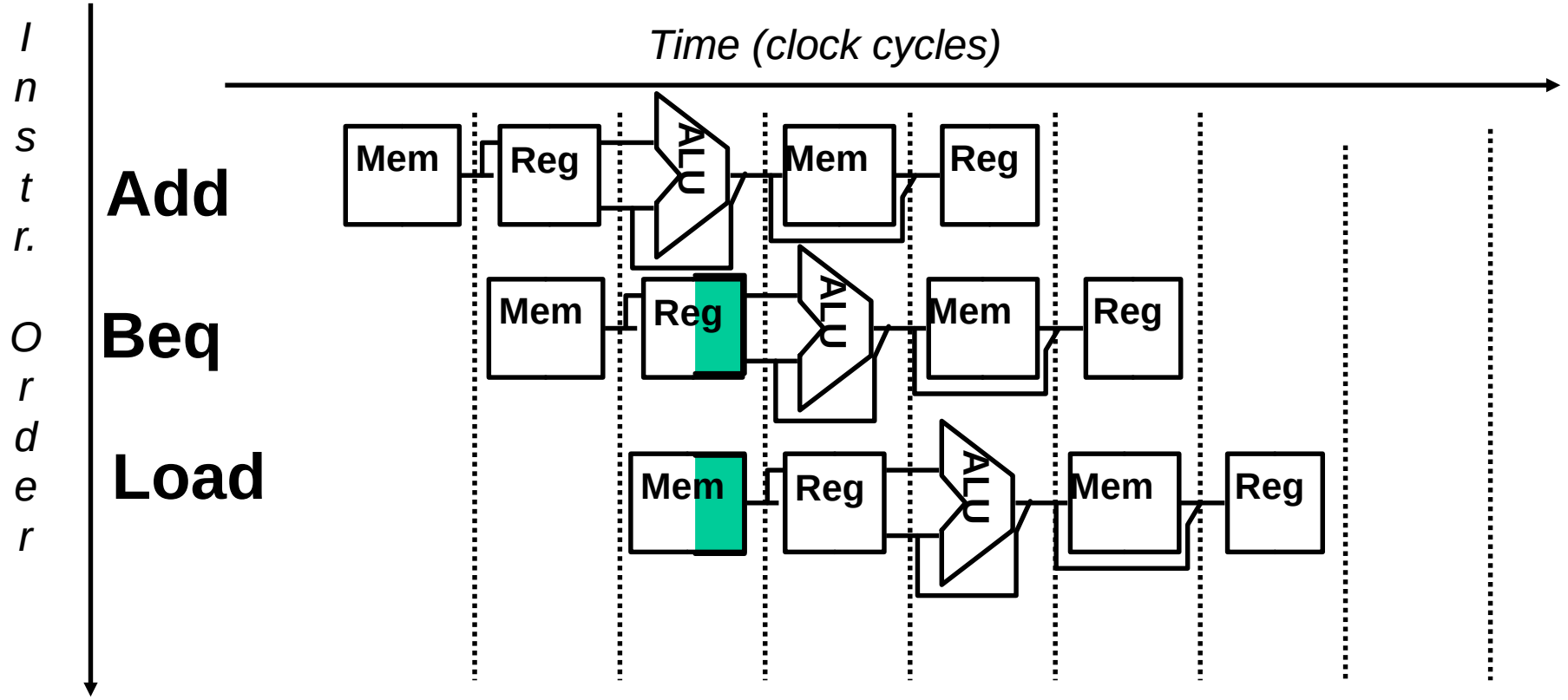


- Impact: 2 clock cycles per branch instruction ⇒ slow

* Slide is courtesy of Dave Patterson

Control Hazard Solution

- Predict: guess one direction then back up if wrong
  - ➔ Predict not taken

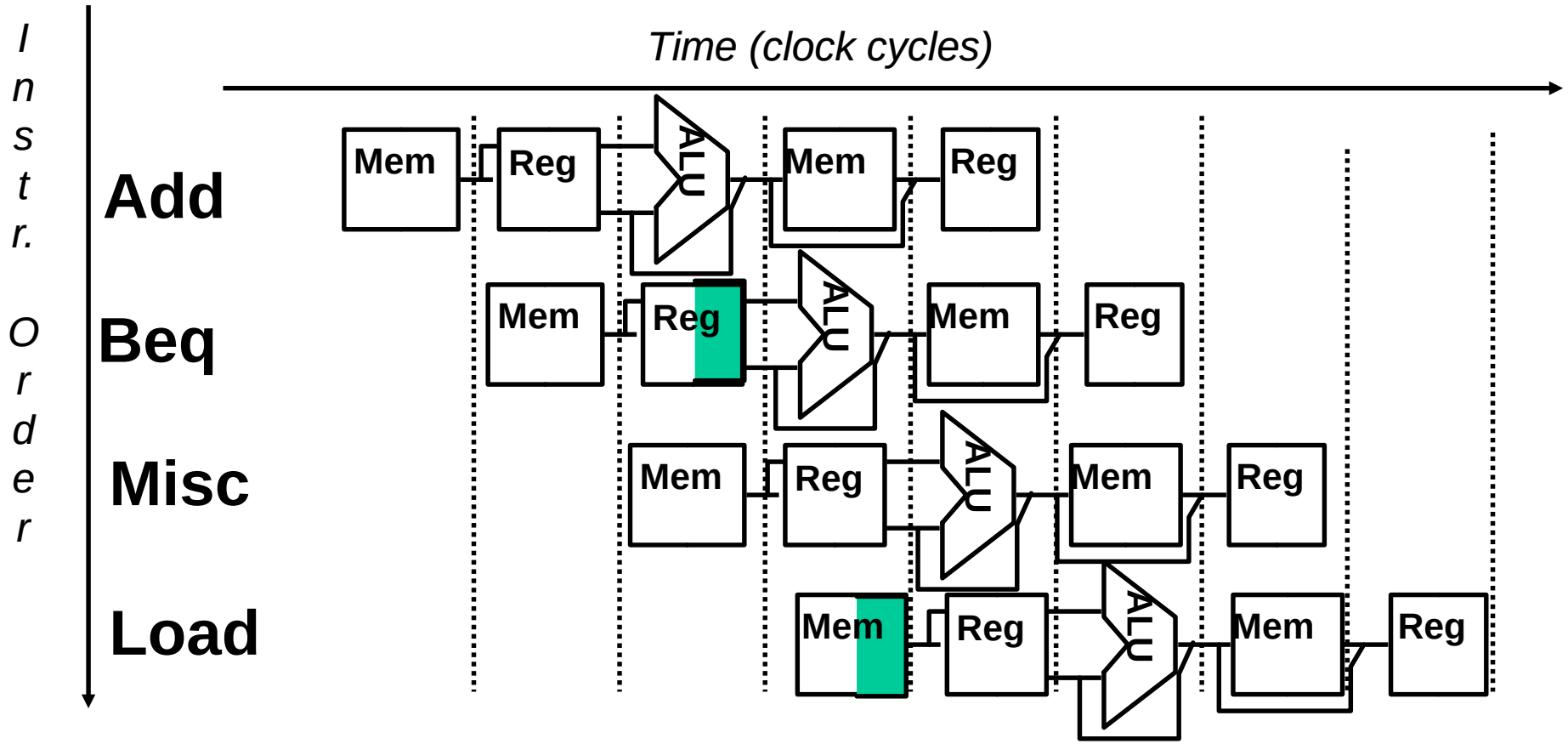


- Impact: 1 clock cycles per branch instruction if right, 2 if wrong (right 50% of time)
- More dynamic scheme: history of 1 branch (90%)

* Slide is courtesy of Dave Patterson

Control Hazard Solution

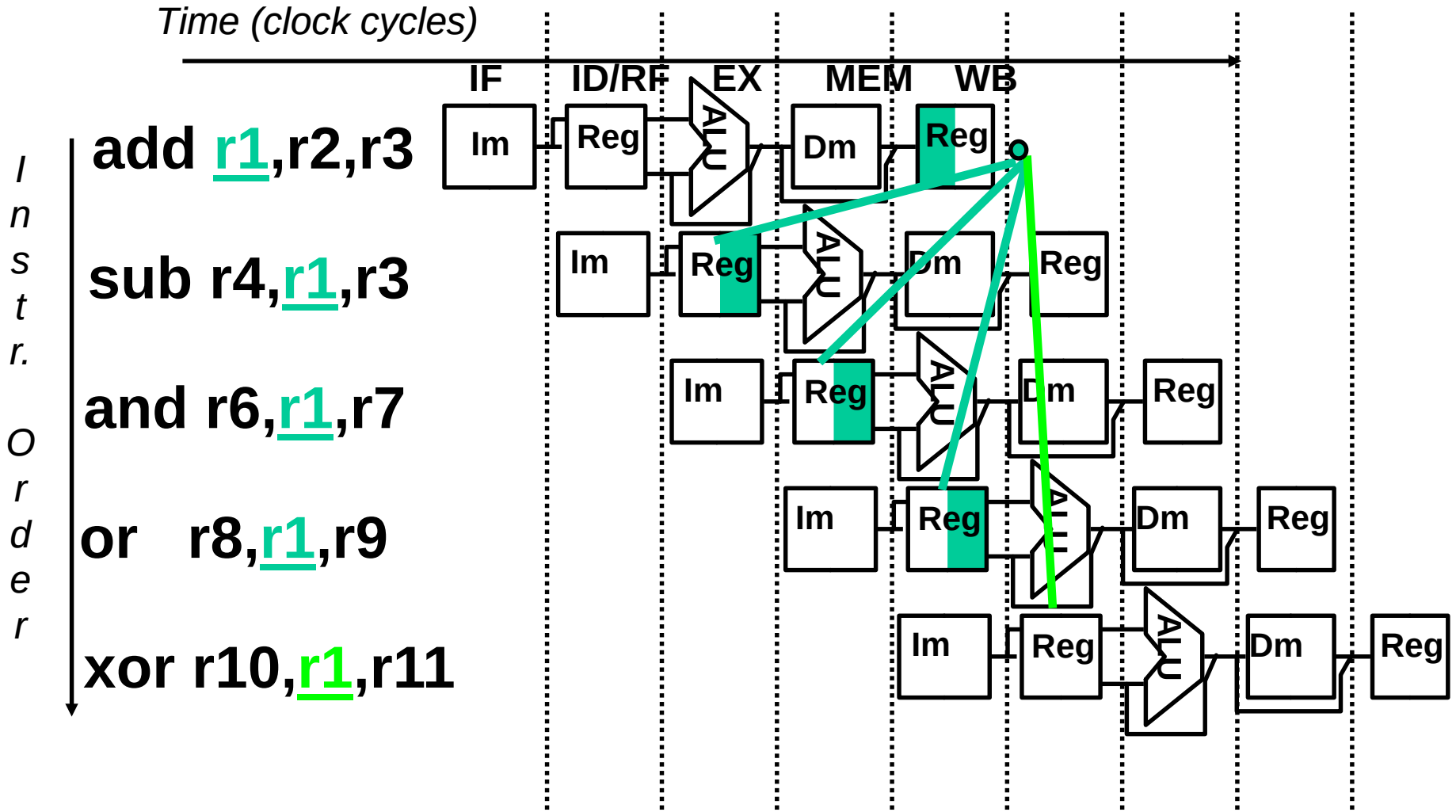
- Redefine branch behavior (takes place after next instruction)
  "delayed branch"



- Impact: 0 clock cycles per branch instruction if can find instruction to put in "slot" (50% of time)

* Slide is courtesy of Dave Patterson

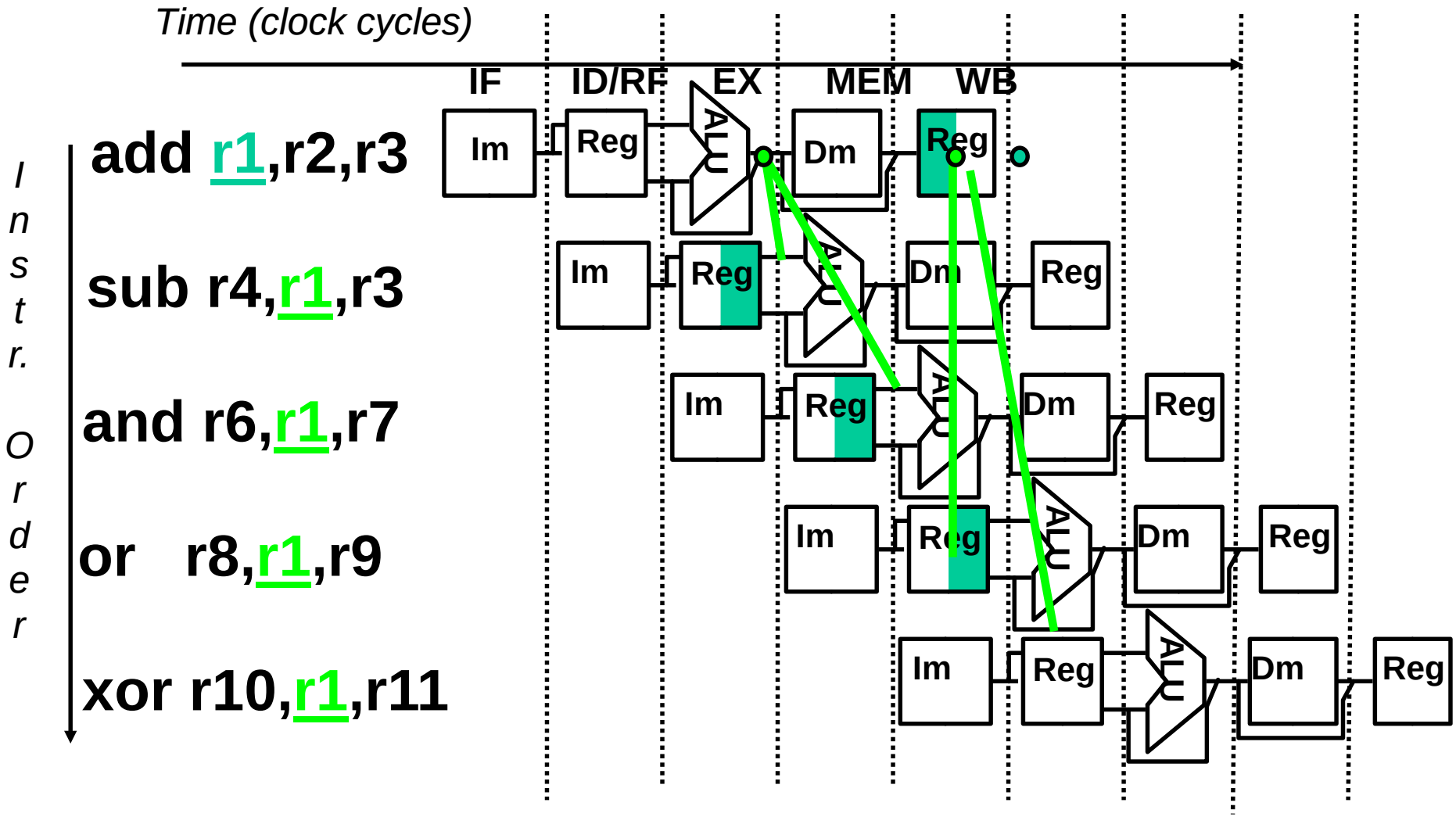
Data Hazard



Dependencies backwards in time are hazards

* Slide is courtesy of Dave Patterson

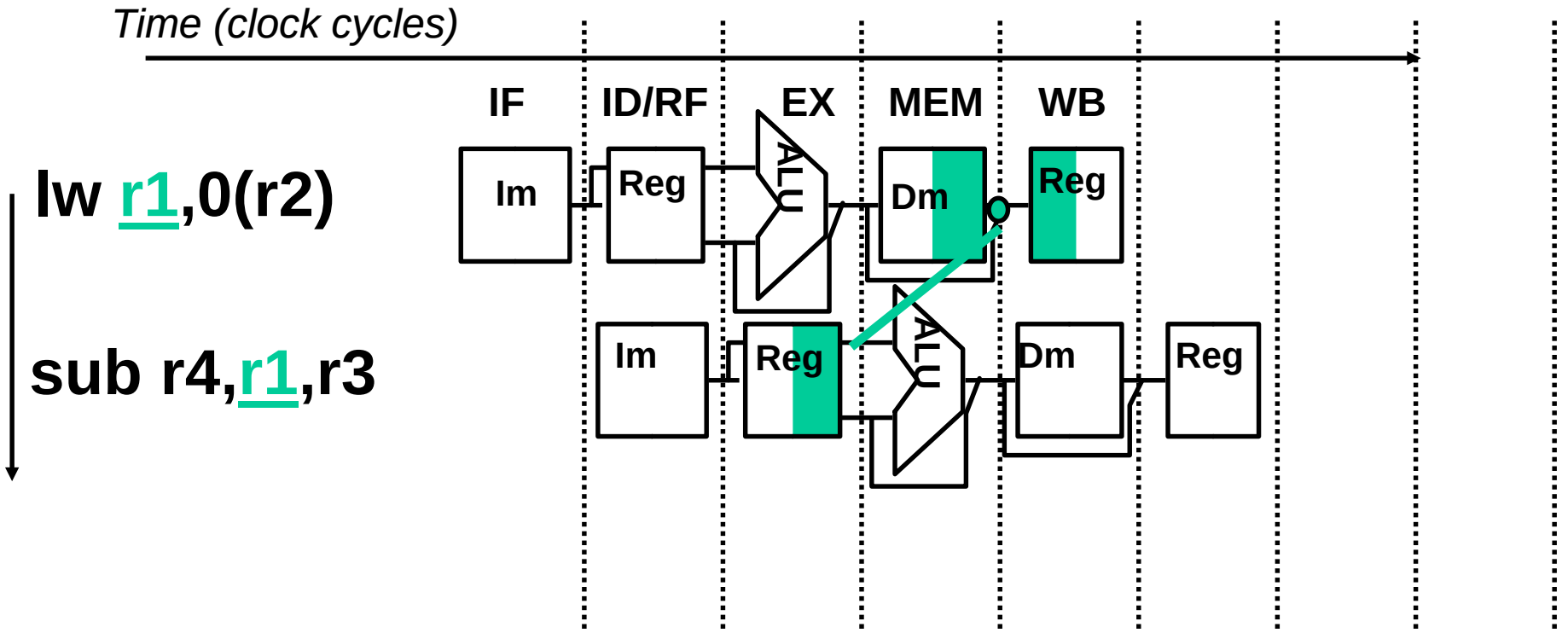
Data Hazard Solution



"Forward" result from one stage to another

* Slide is courtesy of Dave Patterson

Resolving Data Hazards for Loads



- Dependencies backwards in time are hazards
- Cannot solve with forwarding
- Must delay/stall instruction dependent on loads

* Slide is courtesy of Dave Patterson

Conclusion

❑ *Summary*

➔ An overview of Pipelining
- Pipelining concept is natural
- Start handling of next instruction while current one is in progress

➔ Pipeline performance
- Performance improvement by increasing instruction throughput
- Ideal and upper bound for speedup is number of stages in pipeline

➔ Pipelined hazards
- Structural, data and control hazards
- Hazard resolution techniques

❑ *Next Lecture*

➔ Designing a pipelined datapath

➔ Pipelined control

Read section 4.5 in the 5th Ed., or 4.5 in the 4th Ed. of the textbook