CprE 381: Computer Organization and Assembly Level Programming

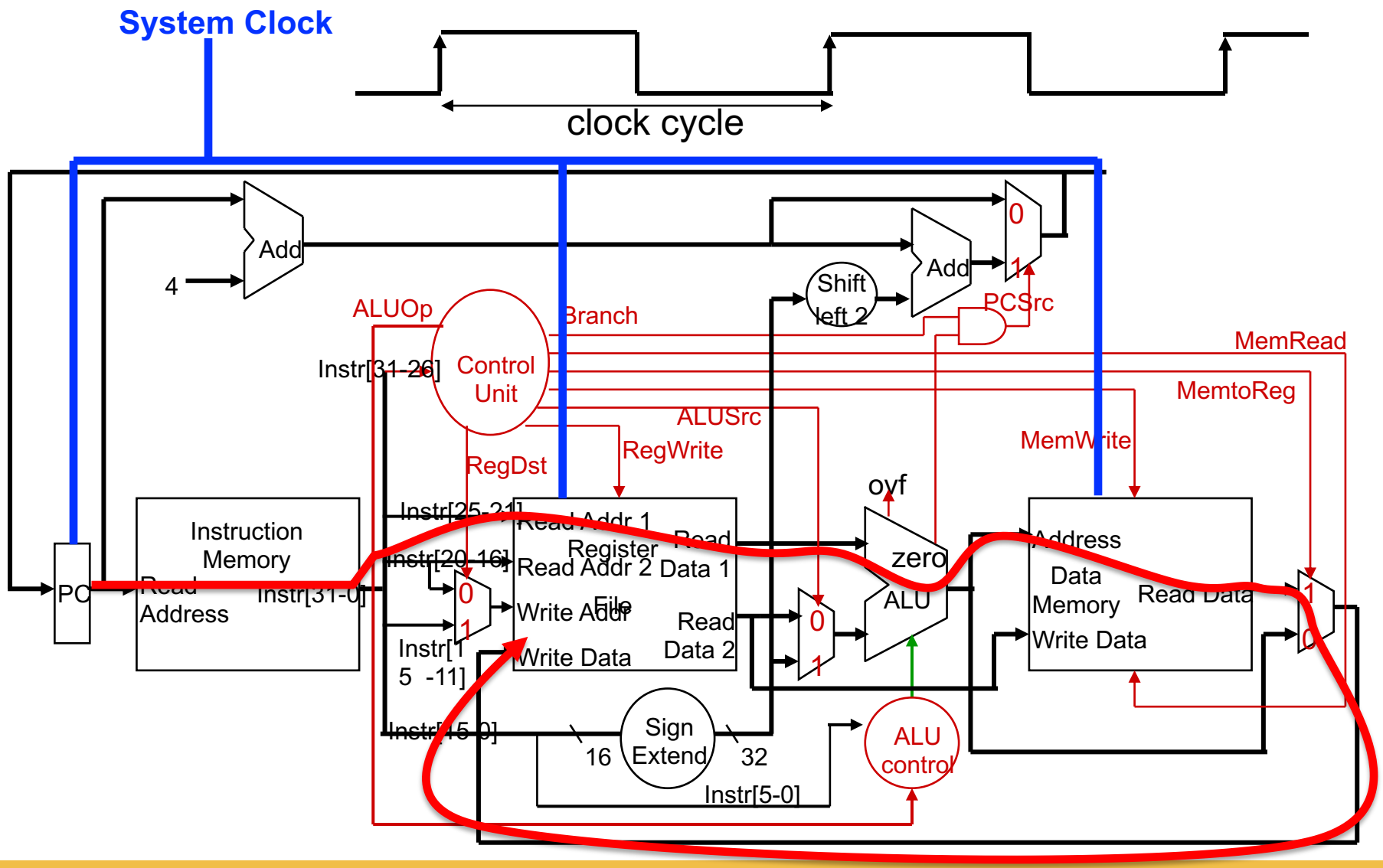
Data Hazards

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Administrative

- Term Proj 2 due week after Spring Break
  - For keeps
- HW7
  - Due Mar 27 11:59pm
  - Last HW for exam 2 coverage
- Exam 2
  - When: April 1
  - Where: **TBD**
  - What: MIPS Arithmetic through pipelining (including hazards)
  - Why:

Review: Worst Case Timing (Load Instruction)



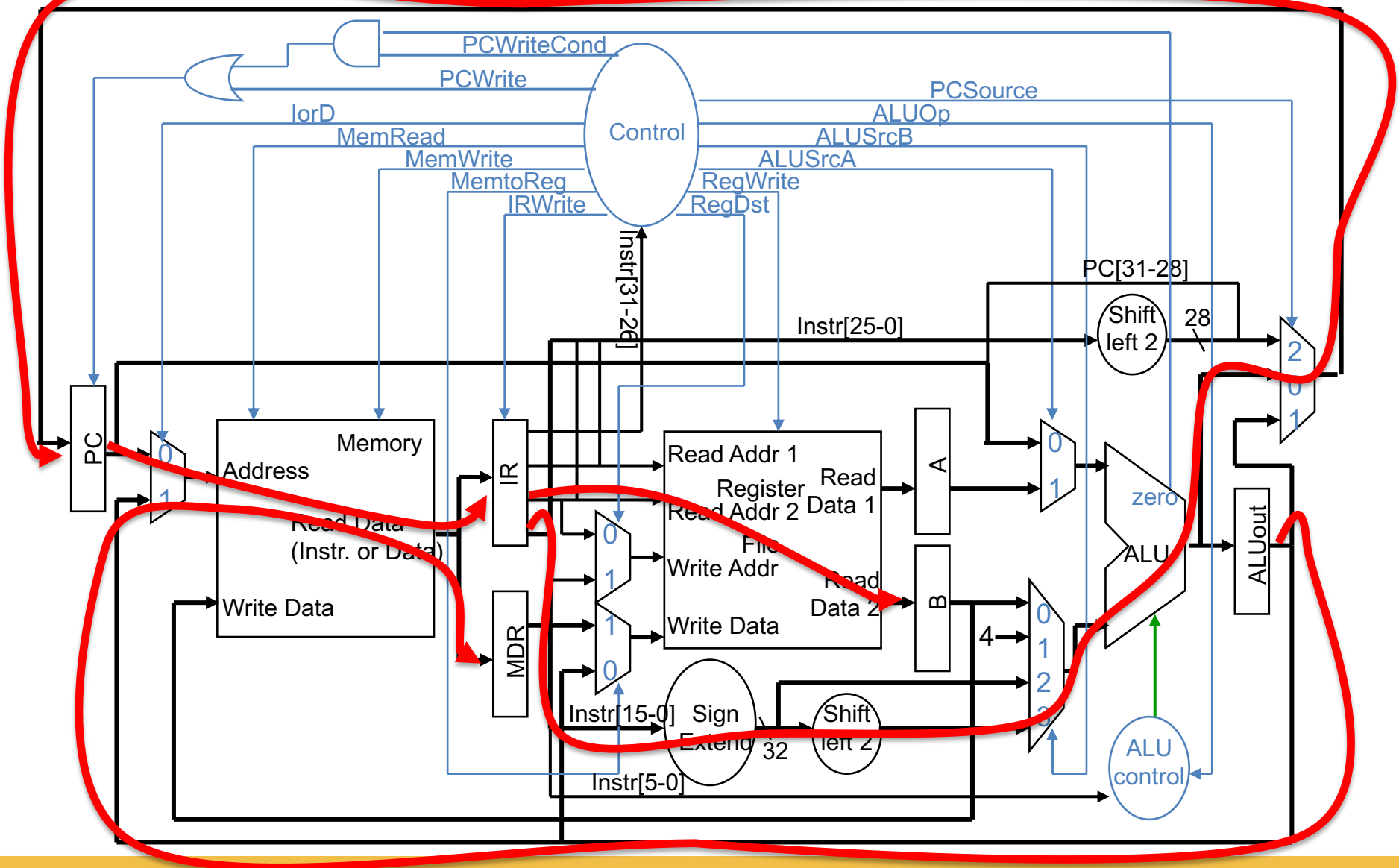
Review: Execution Time

- Drawing on the previous equation:

$$\textit{Execution Time} = \#\ \textit{Instructions} \times \frac{\textit{Cycles}}{\textit{Instruction}} \times \frac{\textit{Seconds}}{\textit{Cycle}}$$

- To improve performance (i.e., reduce execution time)
  - Increase clock rate (decrease clock cycle time) OR
  - Decrease CPI OR
  - Reduce the number of instructions
- Designers balance cycle time against the number of cycles required
  - Improving one factor may make the other one worse...

Review: Multicycle Processor



Review: Execution Time

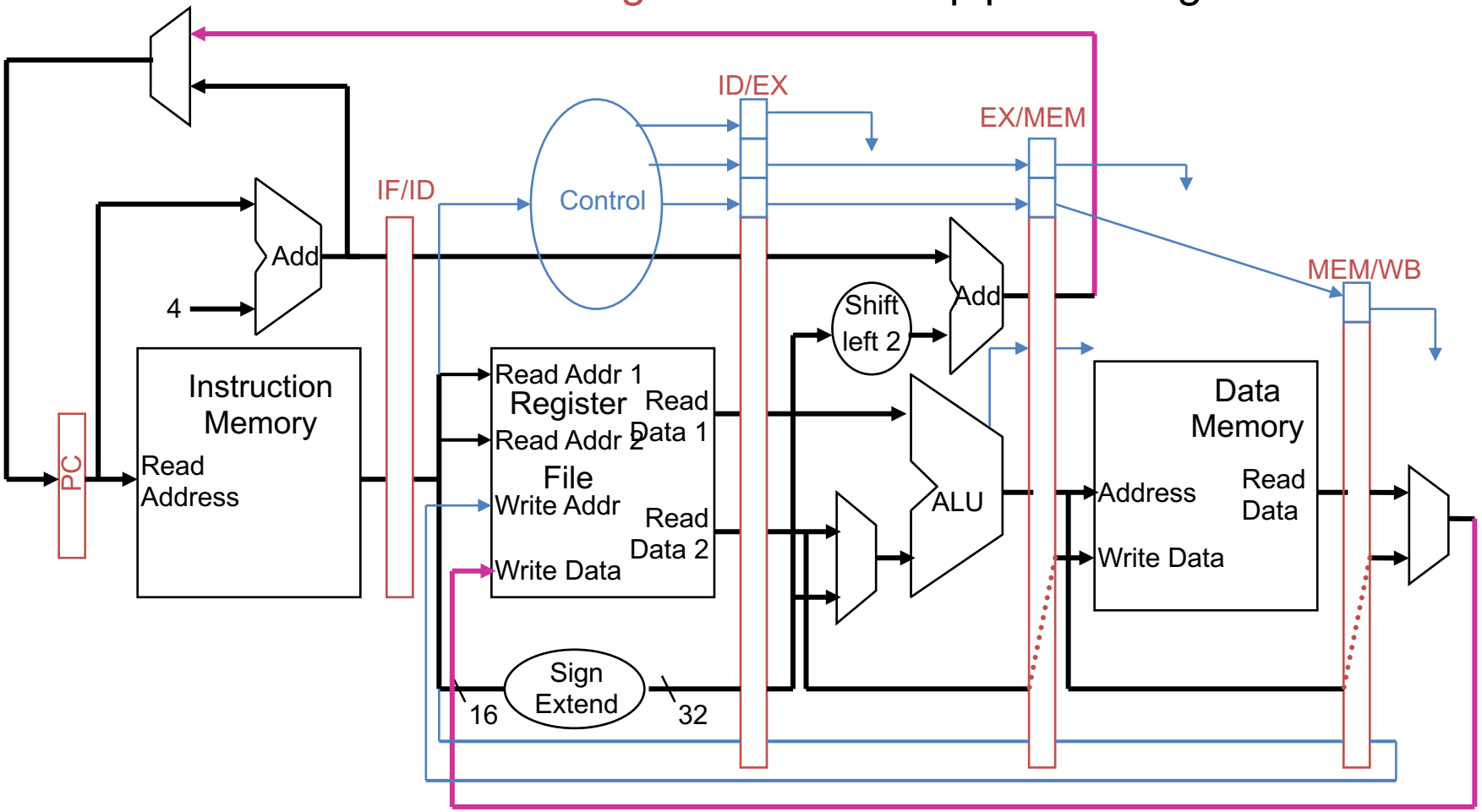
- Drawing on the previous equation:

$$\textit{Execution Time} = \#\ \textit{Instructions} \times \frac{\textit{Cycles}}{\textit{Instruction}} \times \frac{\textit{Seconds}}{\textit{Cycle}}$$

- To improve performance (i.e., reduce execution time)
  - Increase clock rate (decrease clock cycle time) OR
  - Decrease CPI OR
  - Reduce the number of instructions
- Designers balance cycle time against the number of cycles required
  - Improving one factor may make the other one worse...

Review: A Simple MIPS Pipeline

- All control signals can be determined during Decode
  - And held in the state registers between pipeline stages



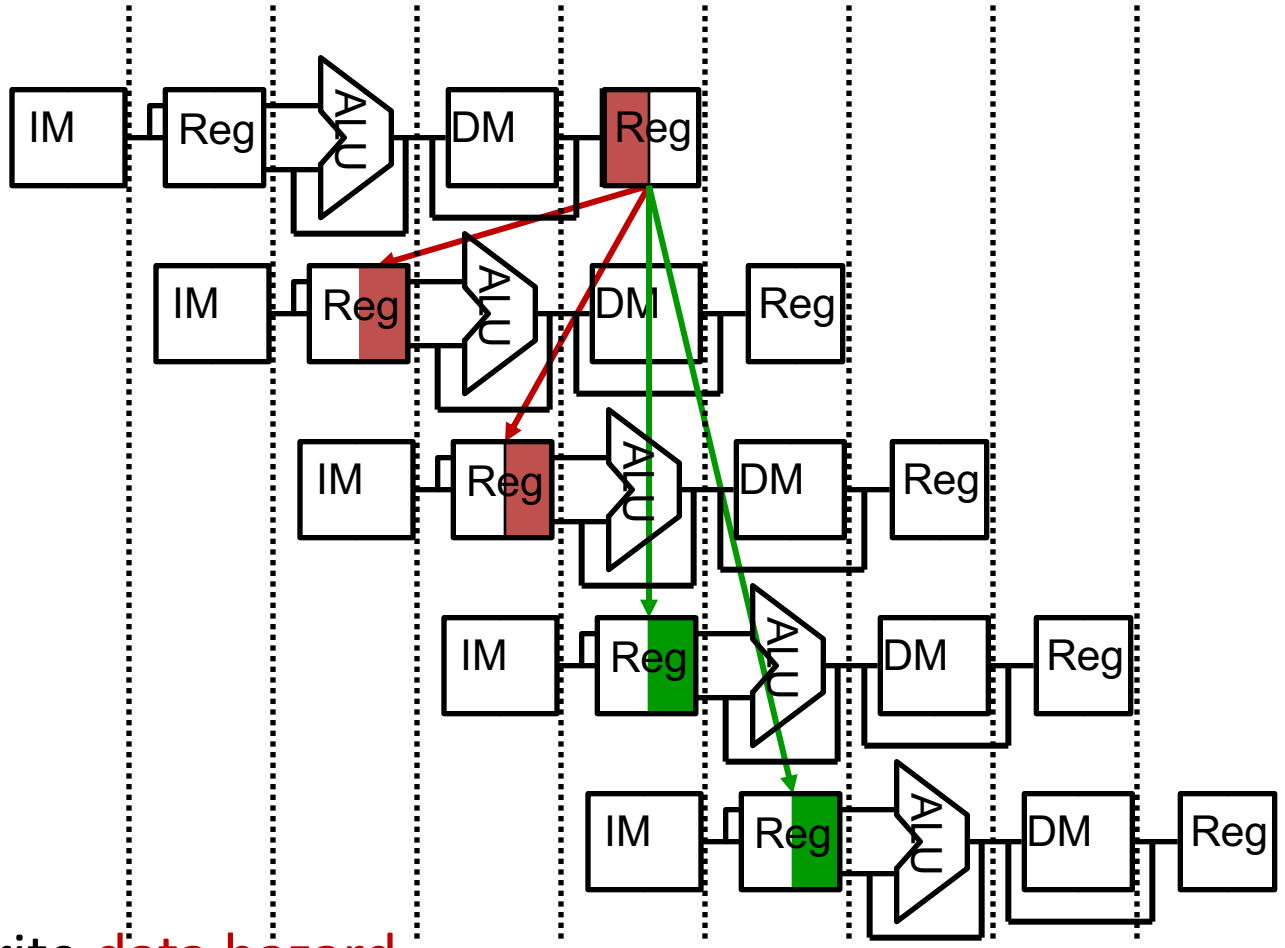
Review: Oh, the *Hazards* I've Seen

- Pipeline Hazards
  - Structural hazards: attempt to use the same resource by two different instructions at the same time
  - Data hazards: attempt to use data before it is ready
    - An instruction's source operand(s) are produced by a prior instruction still in the pipeline
  - Control hazards: attempt to make a decision about program control flow before the condition has been evaluated and the new PC target address calculated
    - Branch and jump instructions, exceptions
- Can always resolve hazards by **waiting**
  - Pipeline control must detect the hazard
  - And take action to resolve hazards

Review: Usage Can Cause Data Hazards

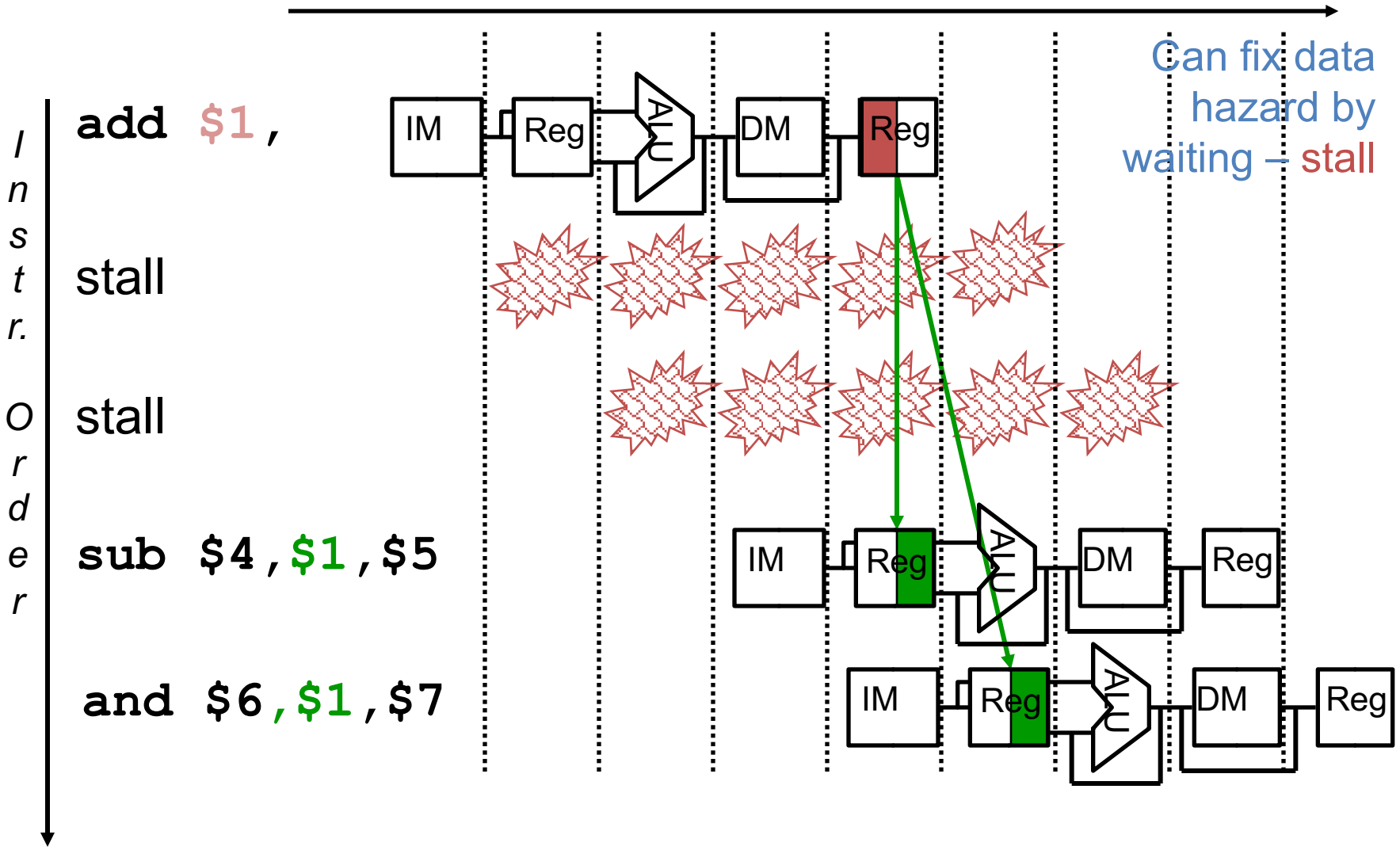
- Dependencies backward in time cause hazards

```
add $1,
sub $4,$1,$5
and $6,$1,$7
or  $8,$1,$9
xor $4,$1,$5
```



- Read before write data hazard

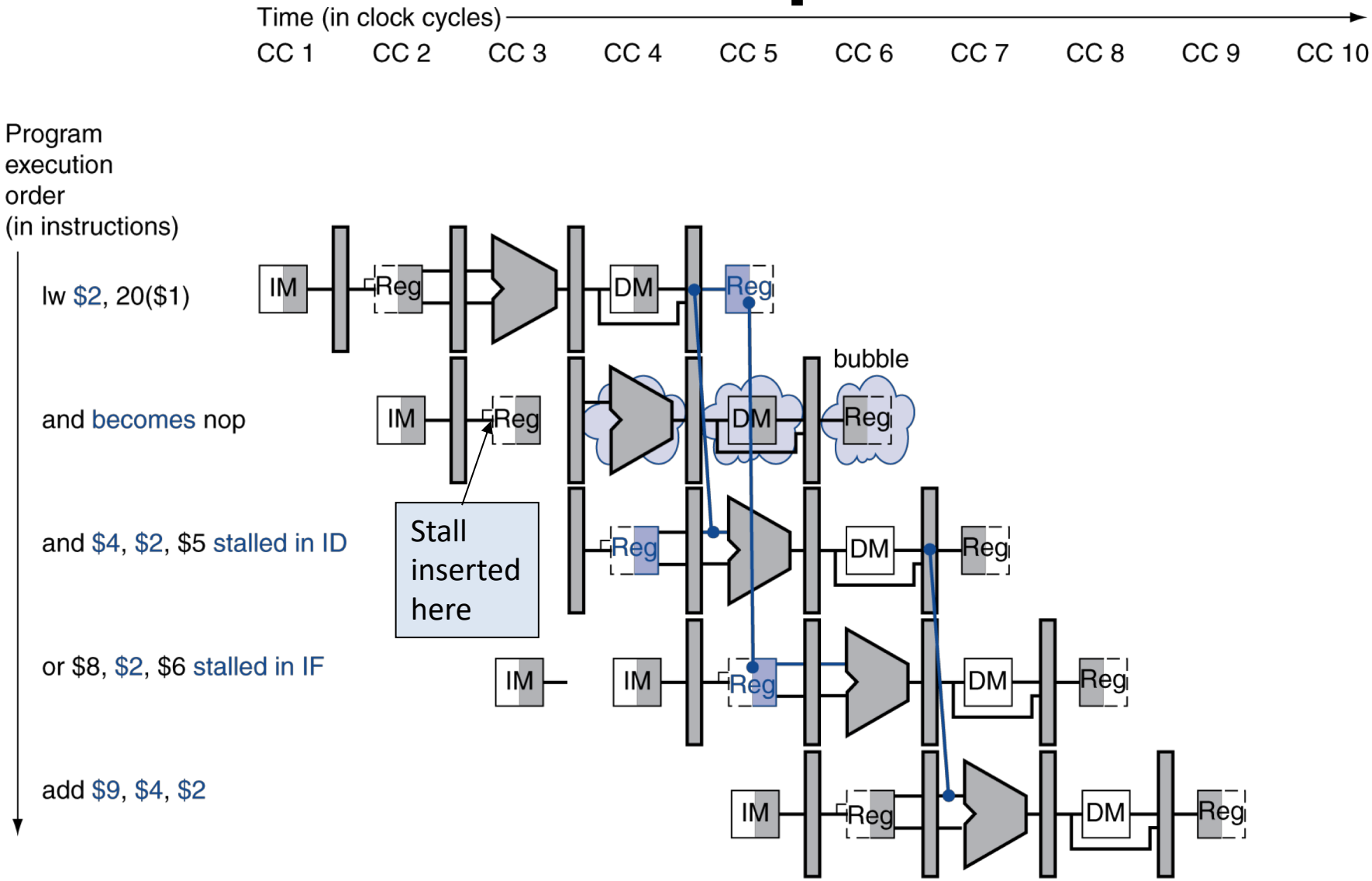
One Way to "Fix" a Data Hazard



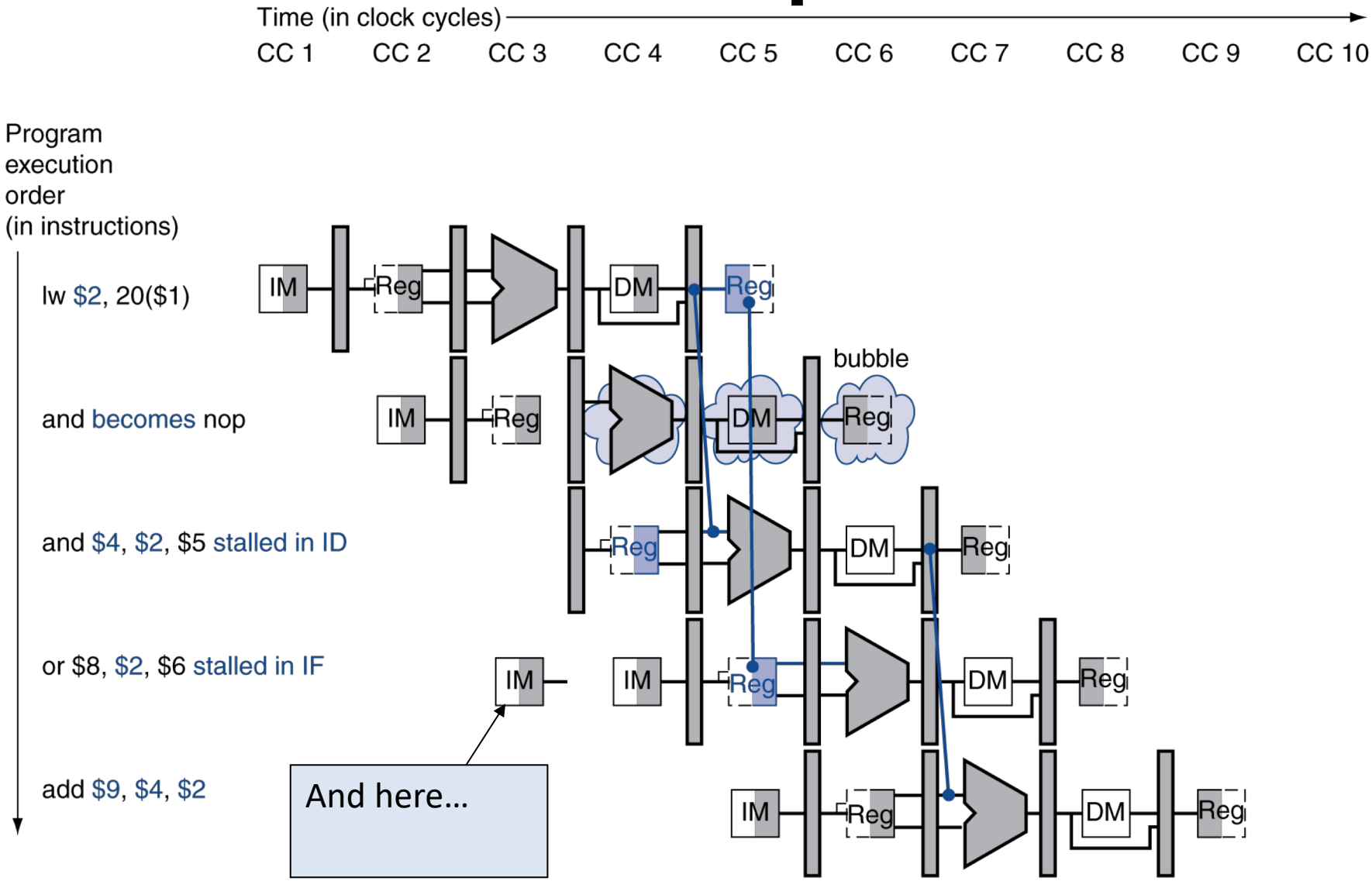
How to Stall the Pipeline (in HW)

- Force control values in ID/EX register to 0
  - EX, MEM and WB do nop (no-operation)
- Prevent update of PC and IF/ID register
  - Dependent instruction is decoded again
  - Following instruction is fetched again
  - 1-cycle stall allows MEM to read data for lw
    - Can subsequently forward to EX stage

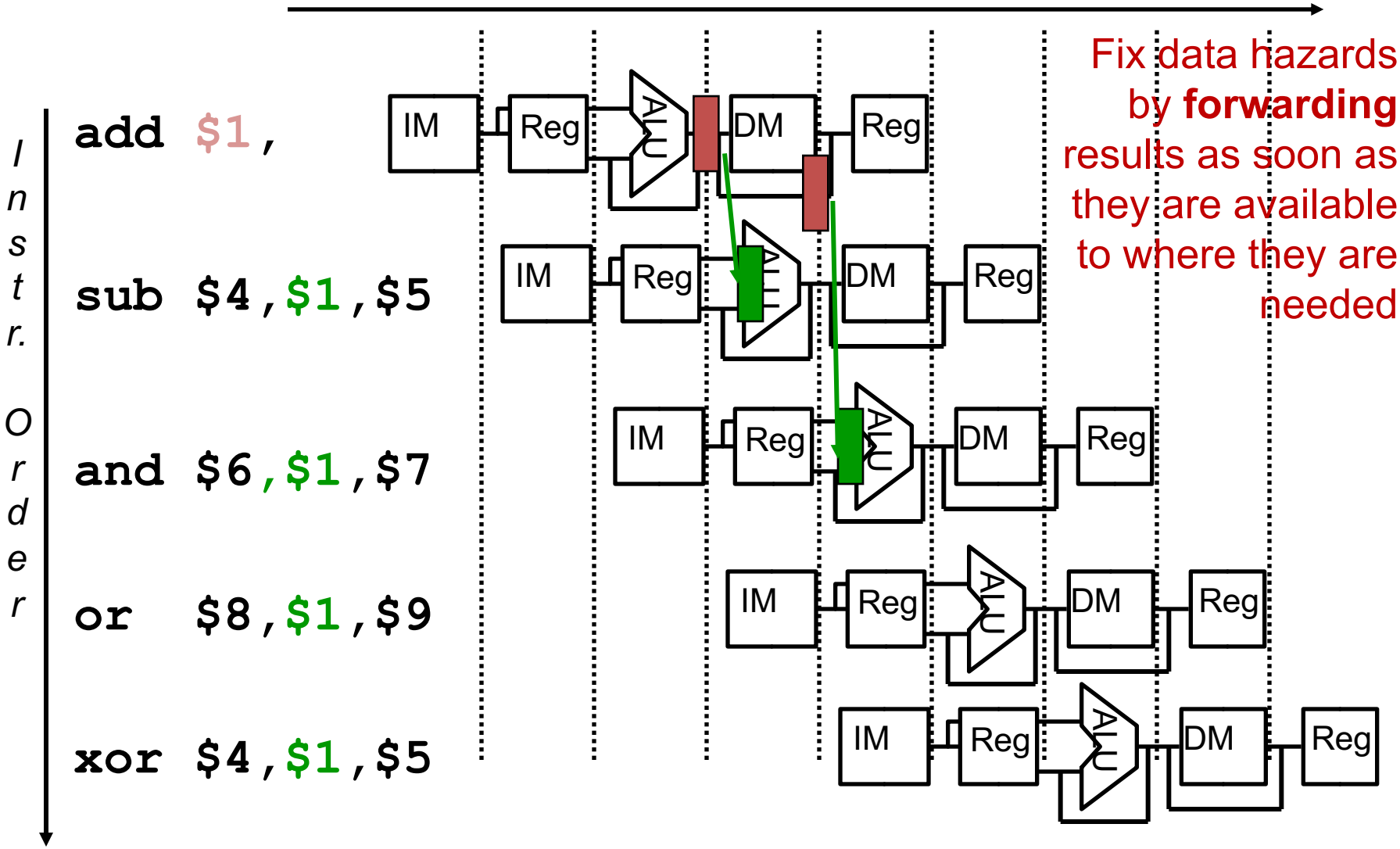
Stall/Bubble in the Pipeline



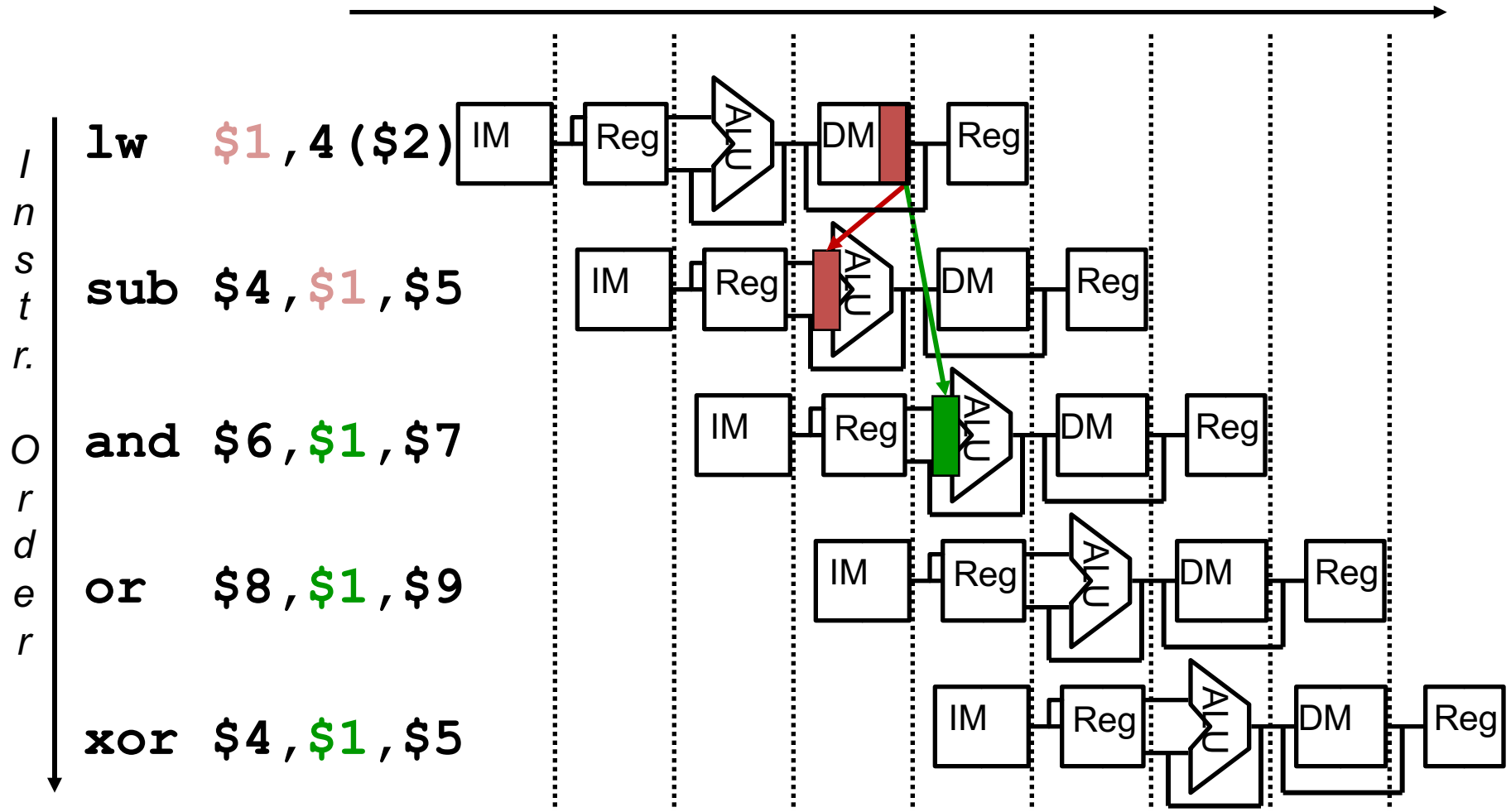
Stall/Bubble in the Pipeline



Another Way to "Fix" a Data Hazard

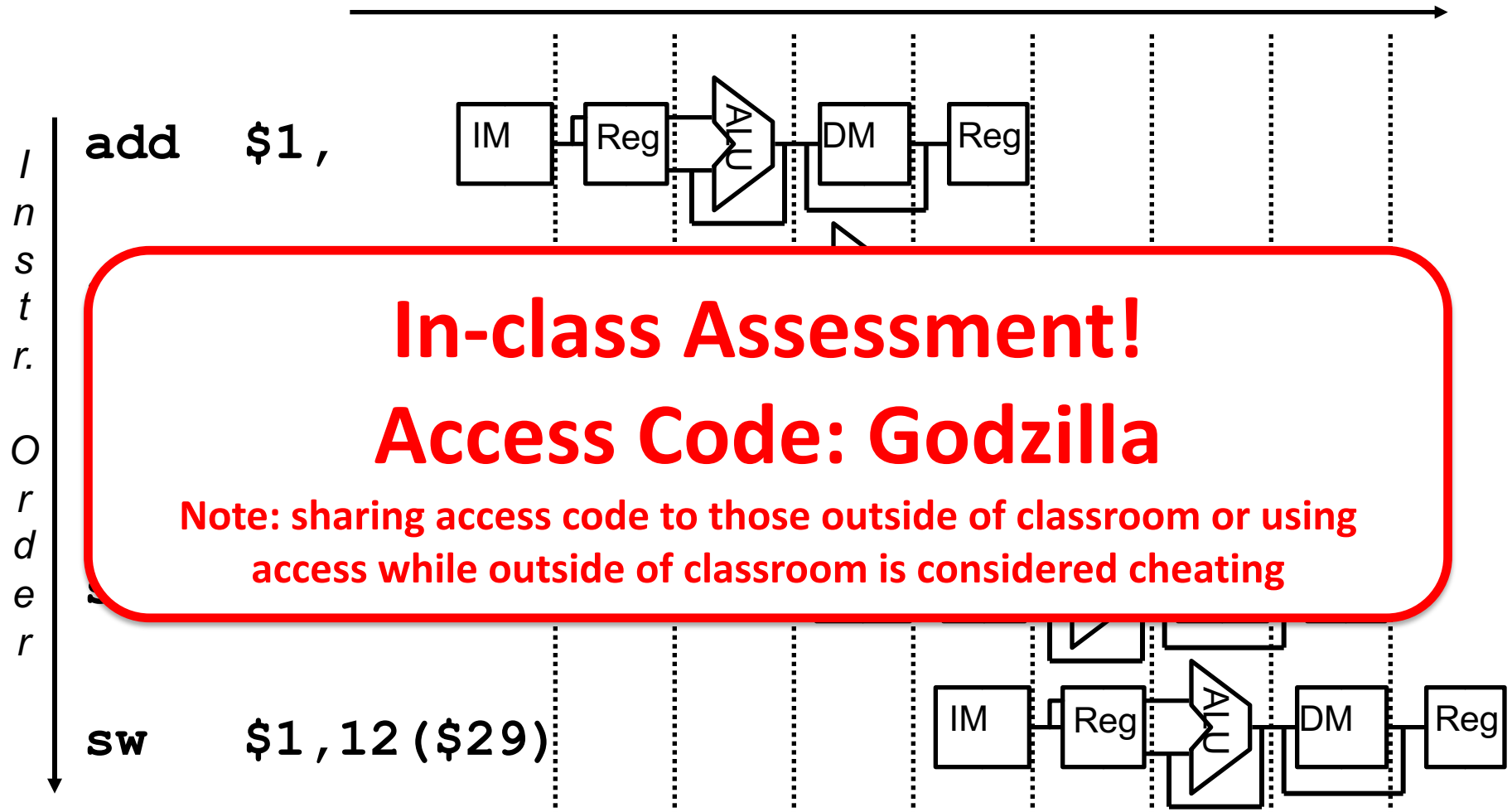


Forwarding with Load-use Data Hazards

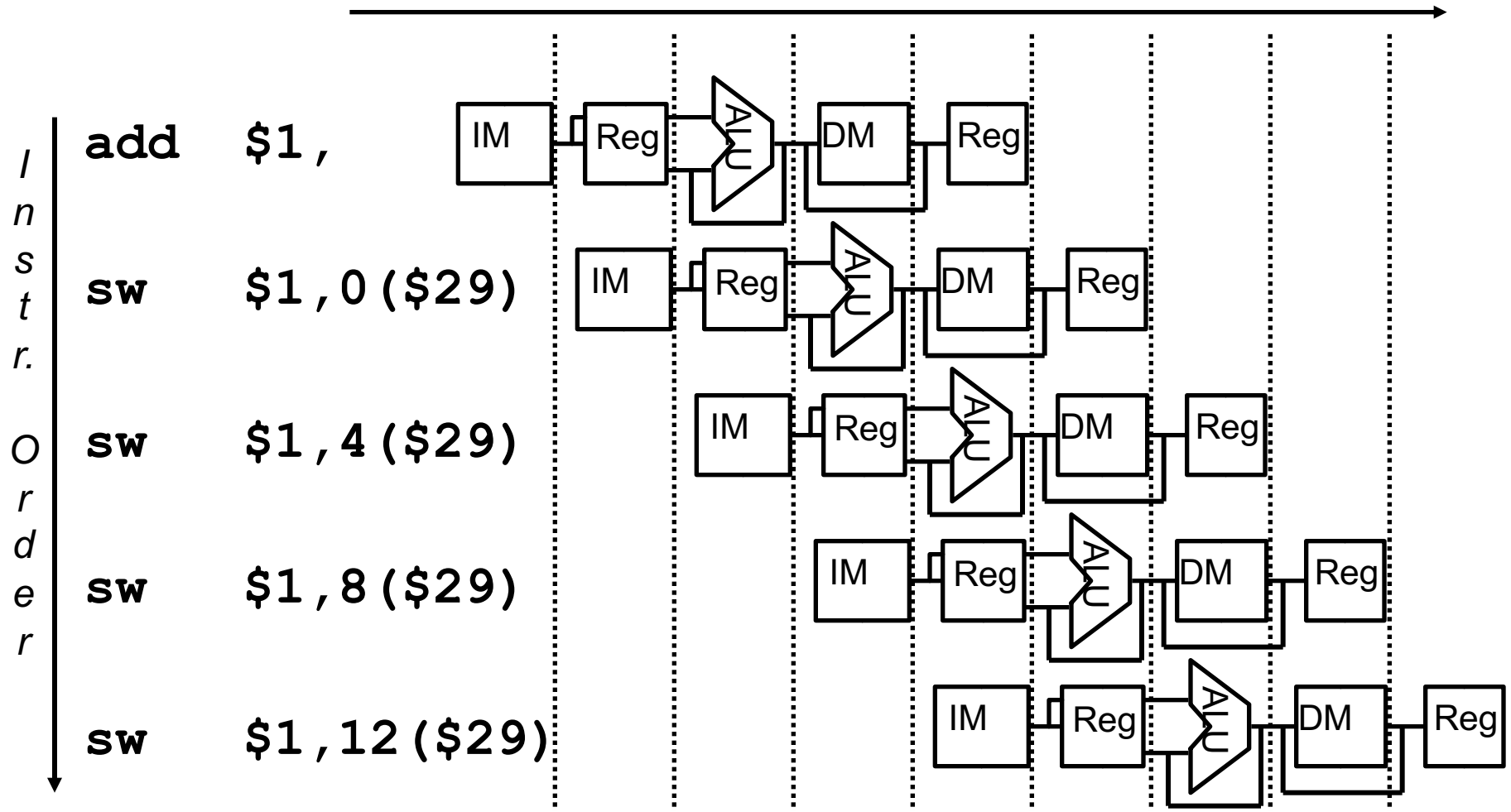


- Will still need one stall cycle even with forwarding

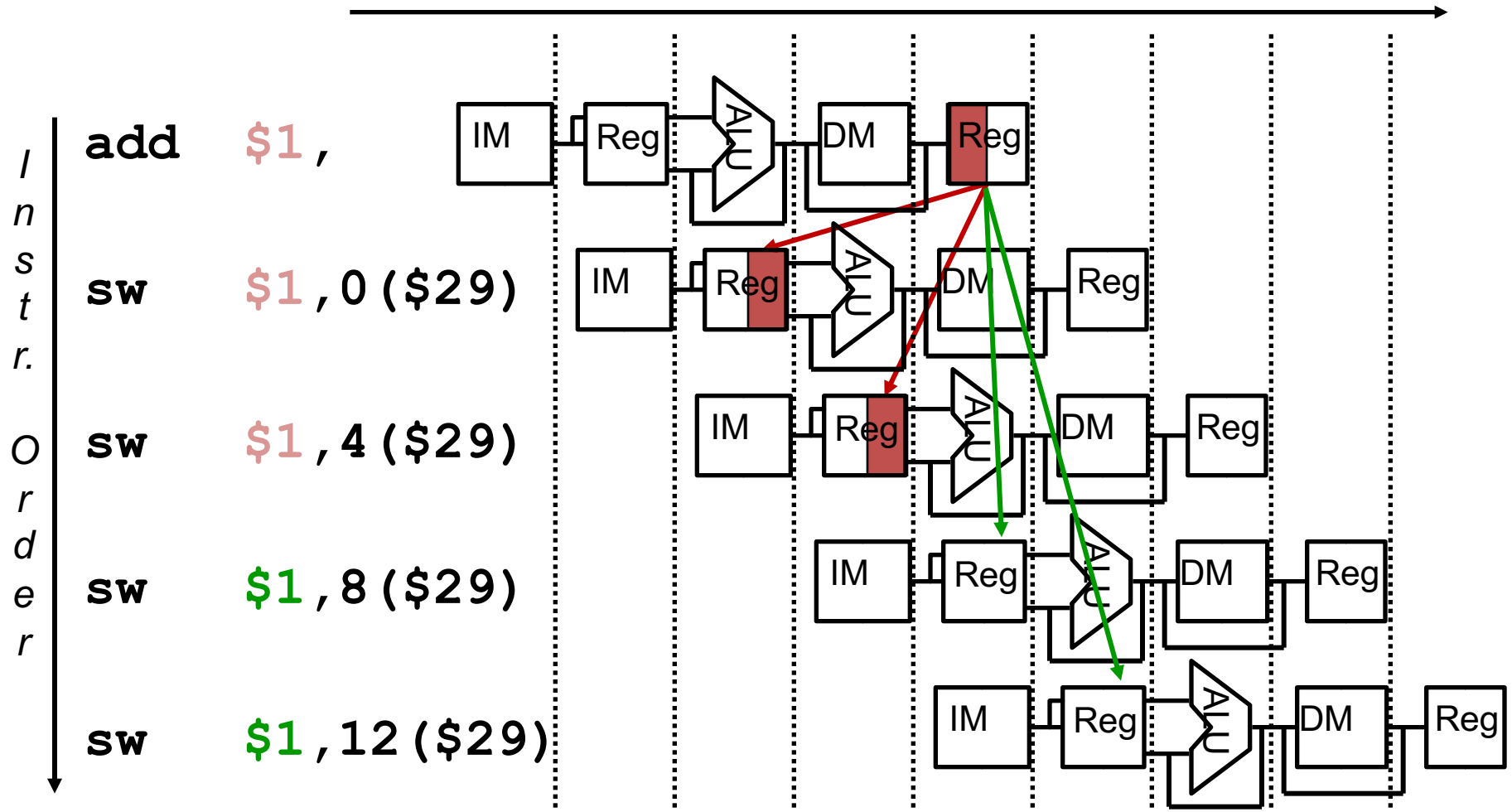
Forwarding with Store Data Hazards?



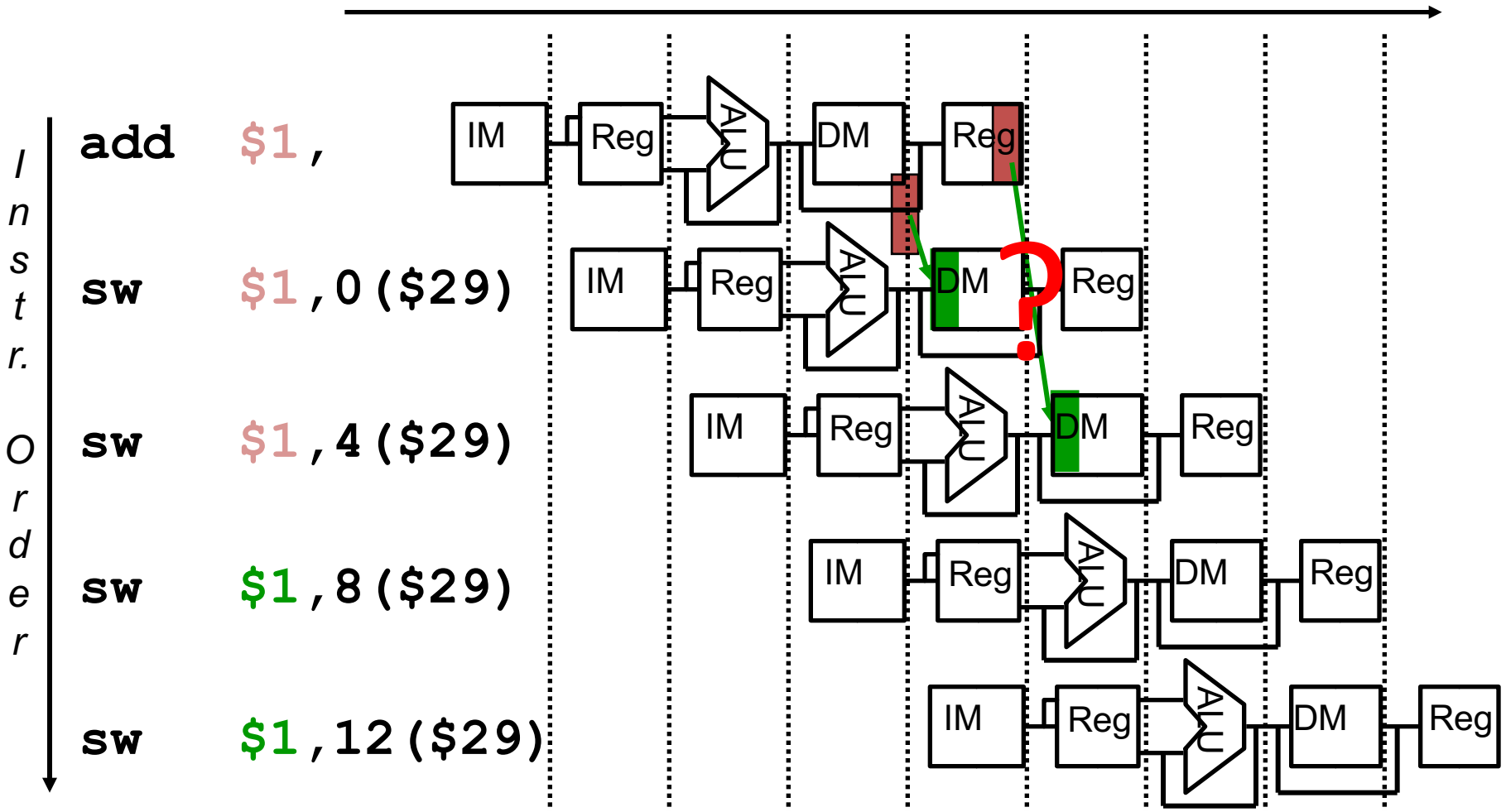
Forwarding with Store Data Hazards?



Store Dependencies & Hazards



Store Forwarding



Forwarding Hardware

- What does forwarding cost?
  - Need to add stuff to datapath and stuff to control
- Datapath
  - Multiplexers at the input of functional units
    - If the mux is already there, make wider (e.g., from 2-1 to 4-1)
  - Wires from forwarding sources to forwarding destinations
- Control
  - Additional control signals for multiplexers
- Notes
  - Adding these muxes increases the critical path of design
  - Needs to be designed carefully

Preview: Implementing Forwarding Paths

- **WARNING**: DETAILED, METHODICAL WORK NEEDED

- Simple procedure → but need to cover all cases
  - Identify all pipeline stages that produce new values
    - In our case, EX and MEM
  - All pipeline registers after the earliest producer can be the source of a forwarded operand
    - In our case, EX/MEM, MEM/WB (although this is multiple registers)
  - Identify all pipeline stages that really consume values
    - In our case, EX (both A and B ports) and MEM (only data port)
  - These stages are the destinations of a forwarded operand
  - Add multiplexor input for each pair of source/destination stages

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)