The Processor: Improving Performance Data Hazards

Computer Organization

Sunday, 09 October 2022

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

TÉCNICO LISBOA

Summary

- Previous Class
  - Pipeline
- Today:
  - Reducing pipeline data hazards
    - Forwarding
    - Stalls

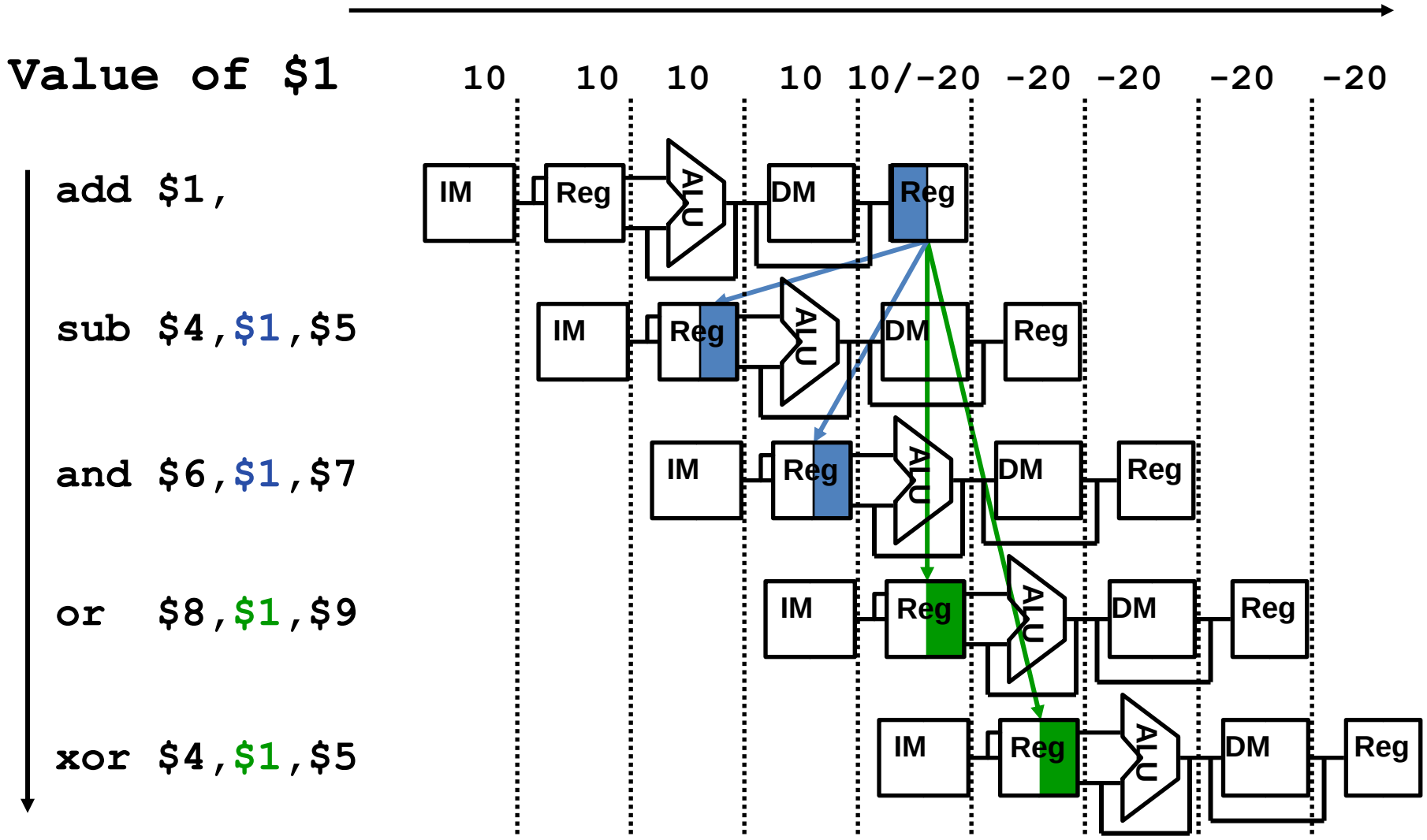
Can Pipelining Get Us Into Trouble?

- Pipeline Hazards - Situations that prevent starting the next instruction in the next cycle
  - structural hazards: attempt to use the same resource by two different instructions at the same time
  - data hazards: Deciding on control action depends on previous instruction
    - 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 by a previous instruction
    - branch and jump instructions, exceptions

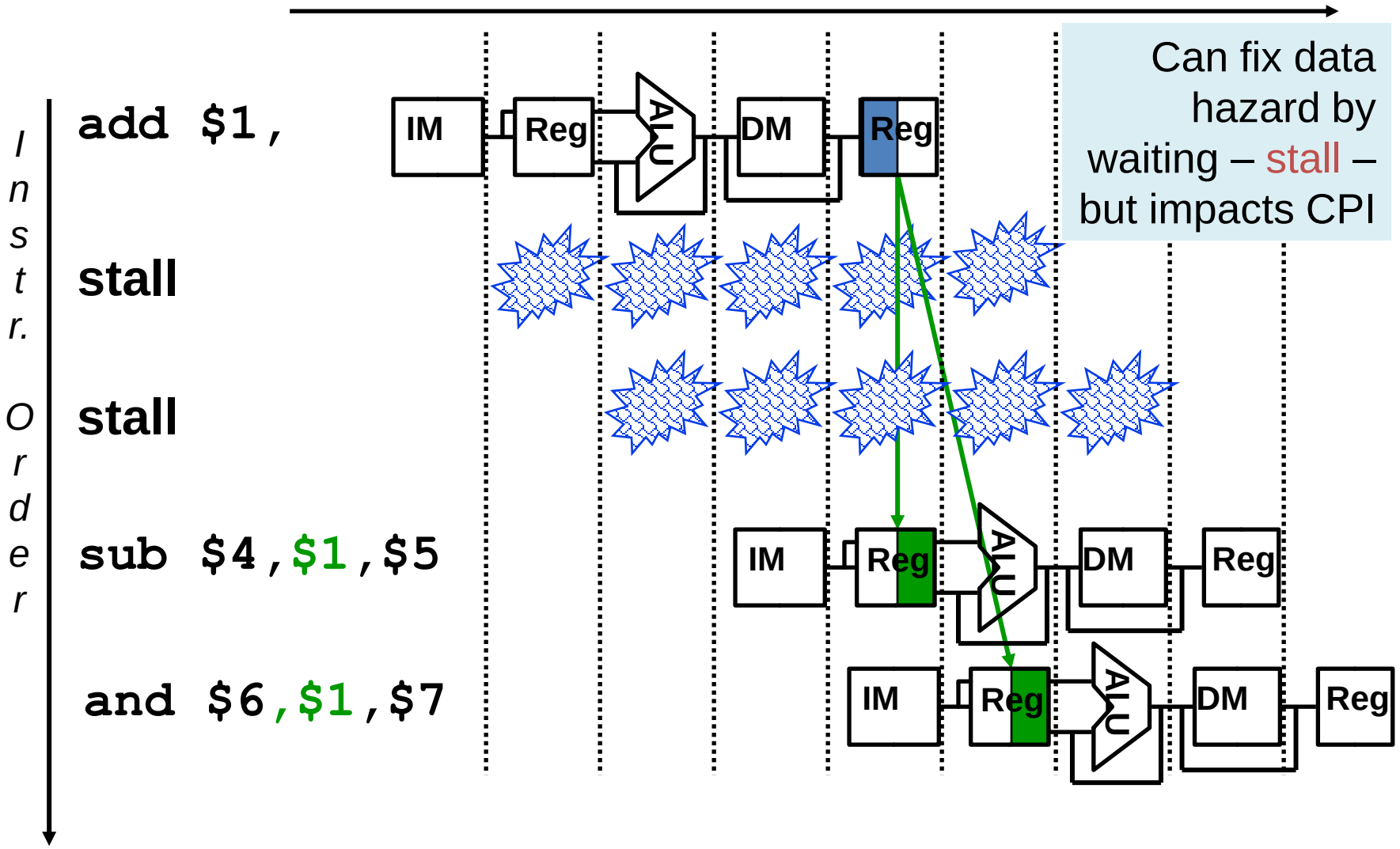
- Can usually resolve hazards by waiting (stall)
  - pipeline control must detect the hazard
  - and take action to resolve hazards

Review: Register Usage Can Cause Data Hazards

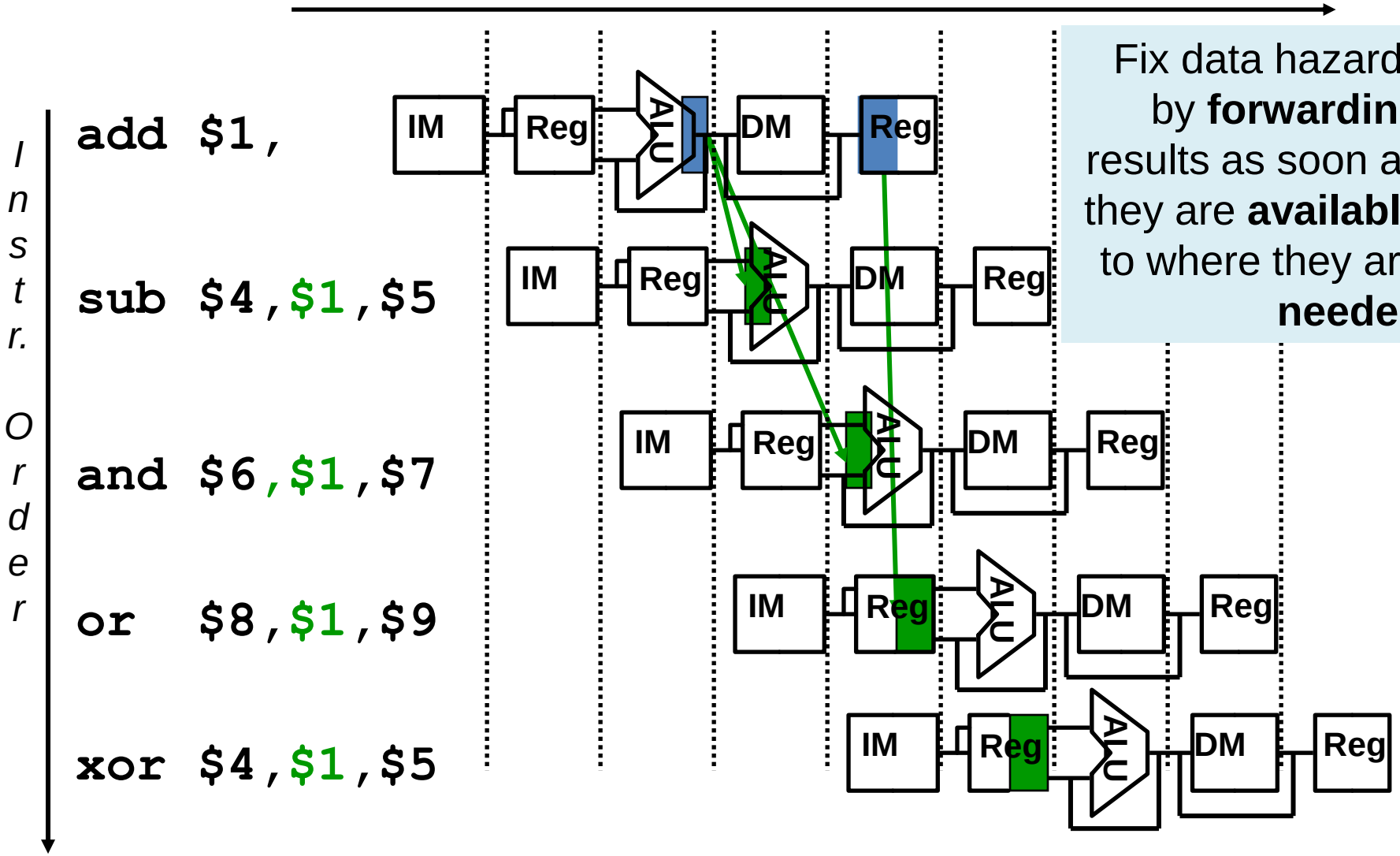
- Read before write data hazard



One Way to “Fix” a Data Hazard - Stall



Another Way to “Fix” a Data Hazard - Forwarding

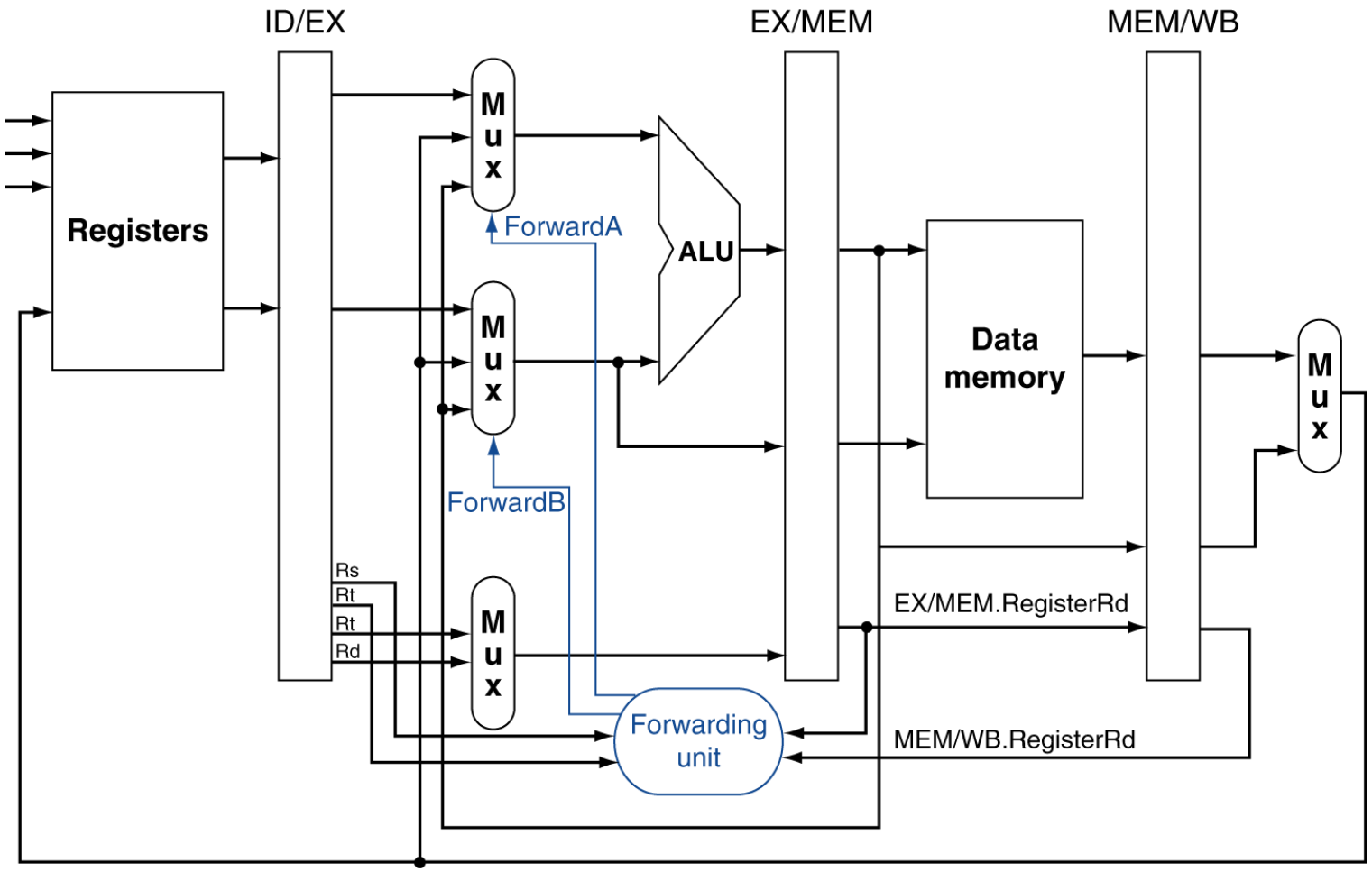


Fix data hazards by **forwarding** results as soon as they are **available** to where they are **needed**

Data Forwarding (aka Bypassing)

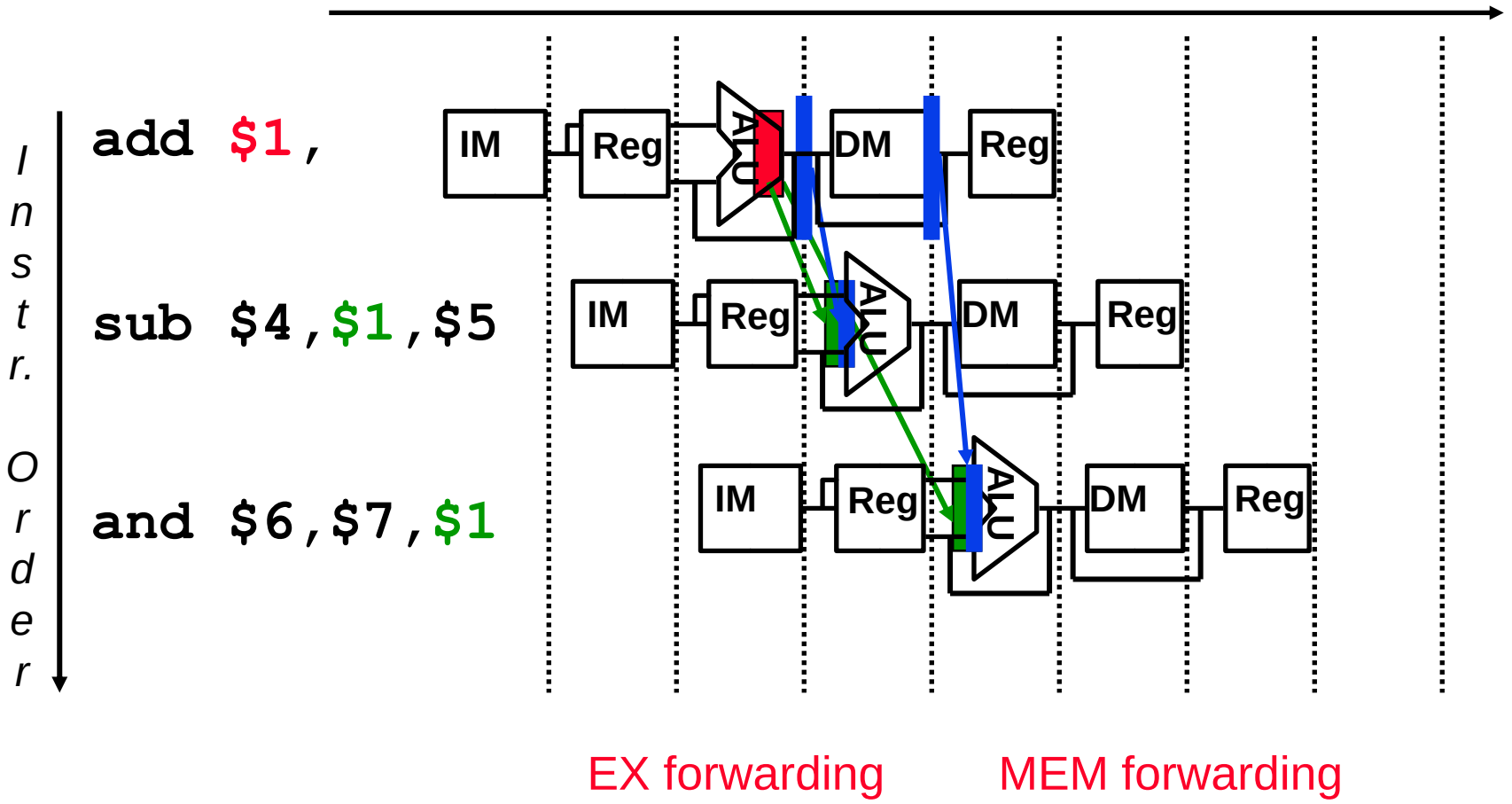
- Take the result from the earliest point that it exists in any of the pipeline state registers and forward it to the functional units (e.g., the ALU) that need it that cycle
- For ALU functional unit: the inputs can come from any pipeline register rather than just from ID/EX by
  - adding multiplexors to the inputs of the ALU
  - connecting the Rd write data in EX/MEM or MEM/WB to either (or both) of the EX's stage Rs and Rt ALU mux inputs
  - adding the proper control hardware to control the new muxes
- Other functional units may need similar forwarding logic (e.g., the Data Memory)
- With forwarding can achieve a CPI of 1 even in the presence of data dependencies

Forwarding Paths



b. With forwarding

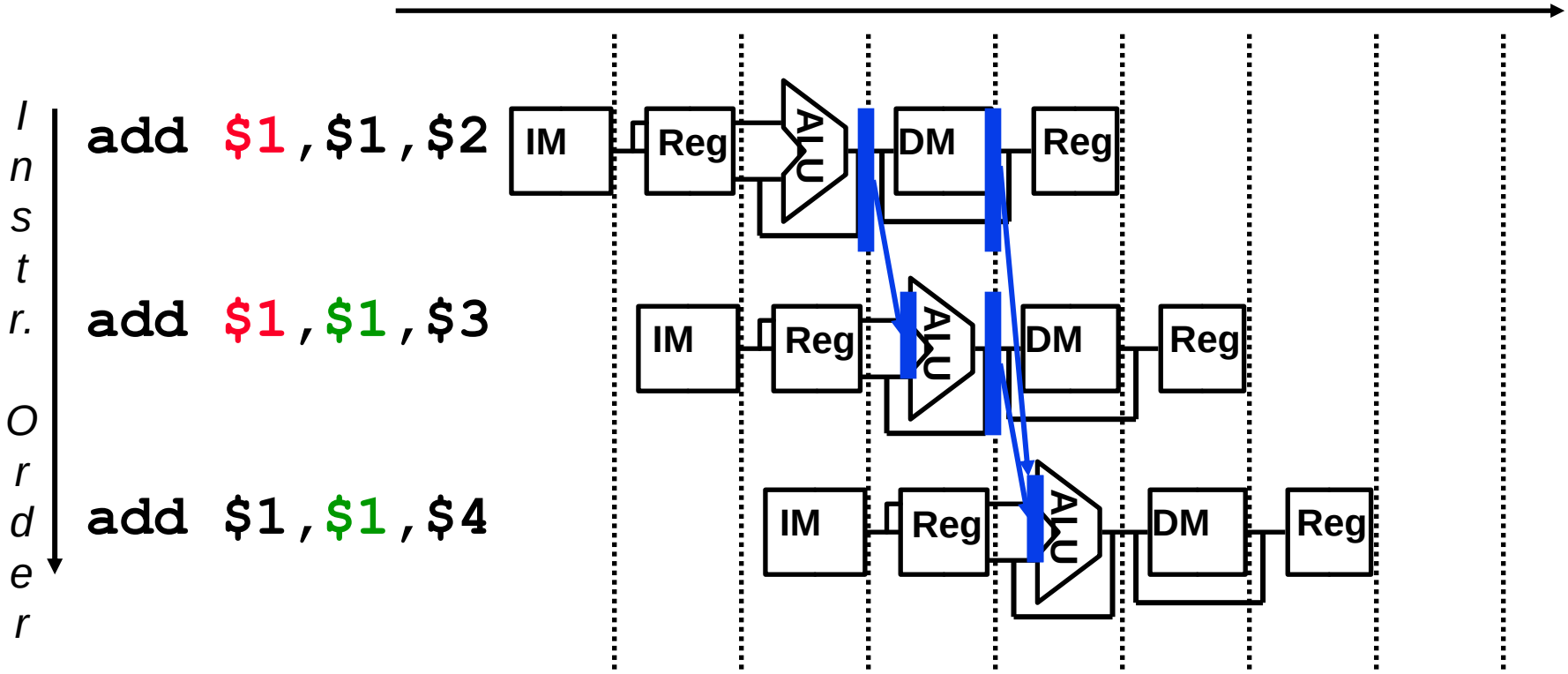
Forwarding Illustration



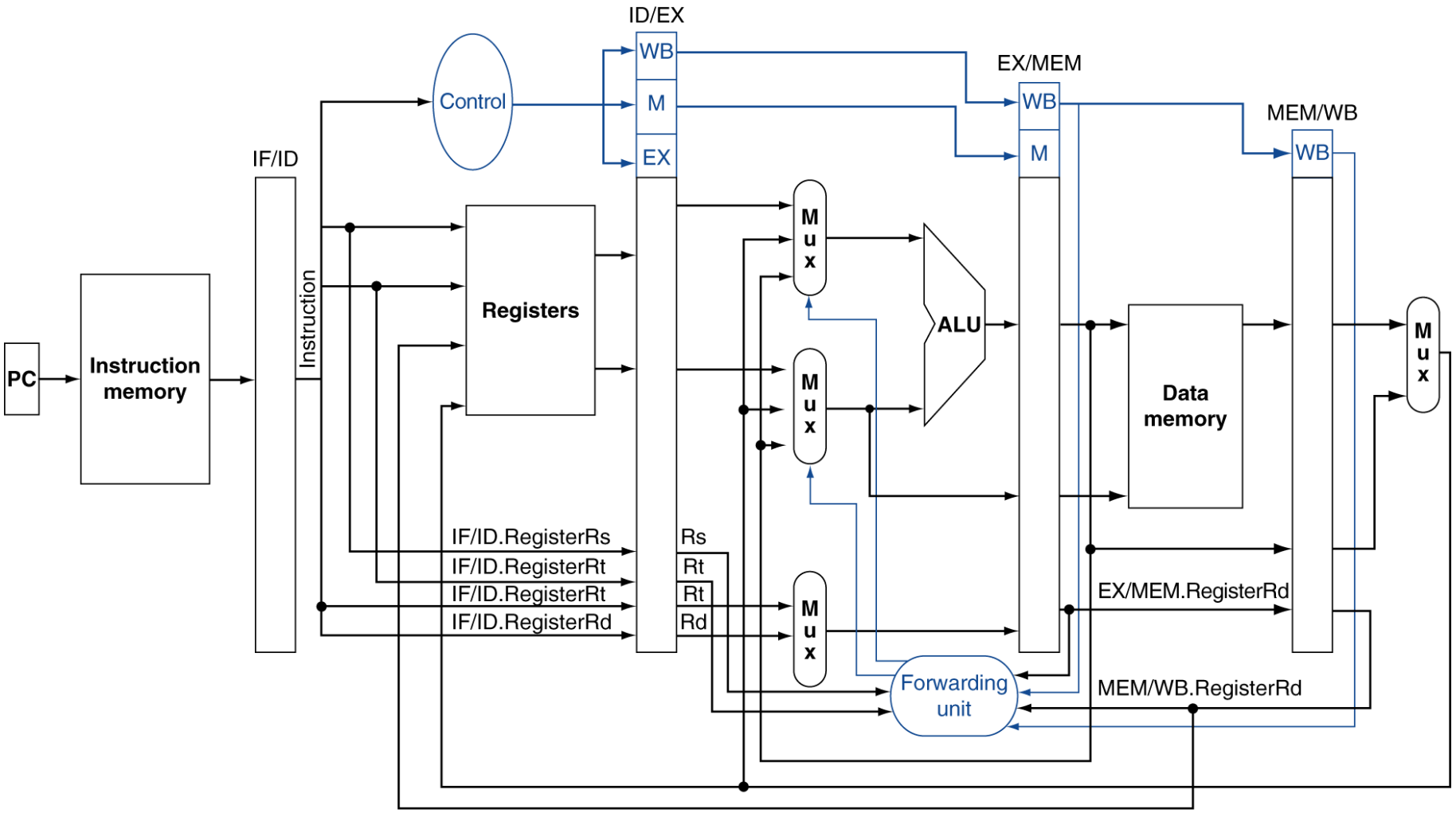
Forwarding - Another Complication!

Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction

- which should be forwarded?



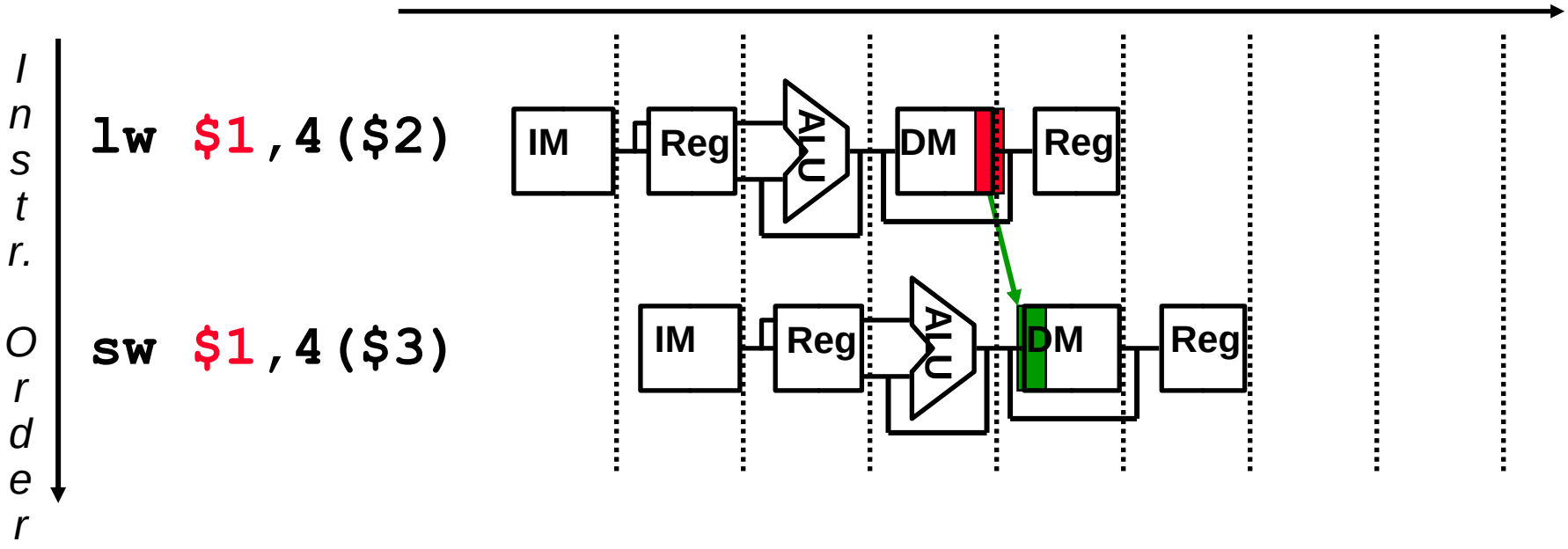
Datapath with Forwarding and Control



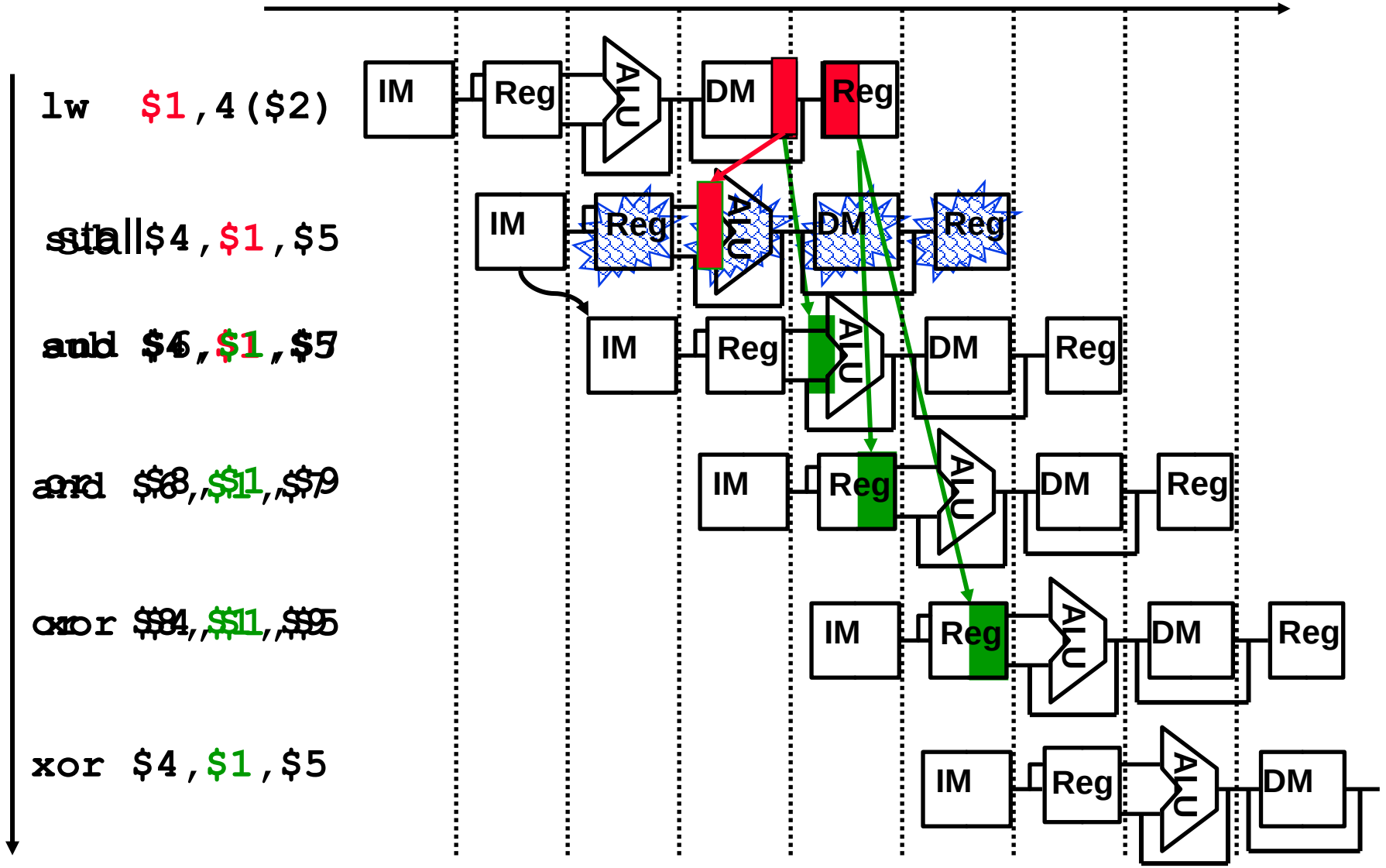
Memory-to-Memory Copies

For loads immediately followed by stores (memory-to-memory copies) can avoid a stall by adding forwarding hardware from the MEM/WB register to the data memory input.

- Would need to add a Forward Unit and a mux to the MEM stage

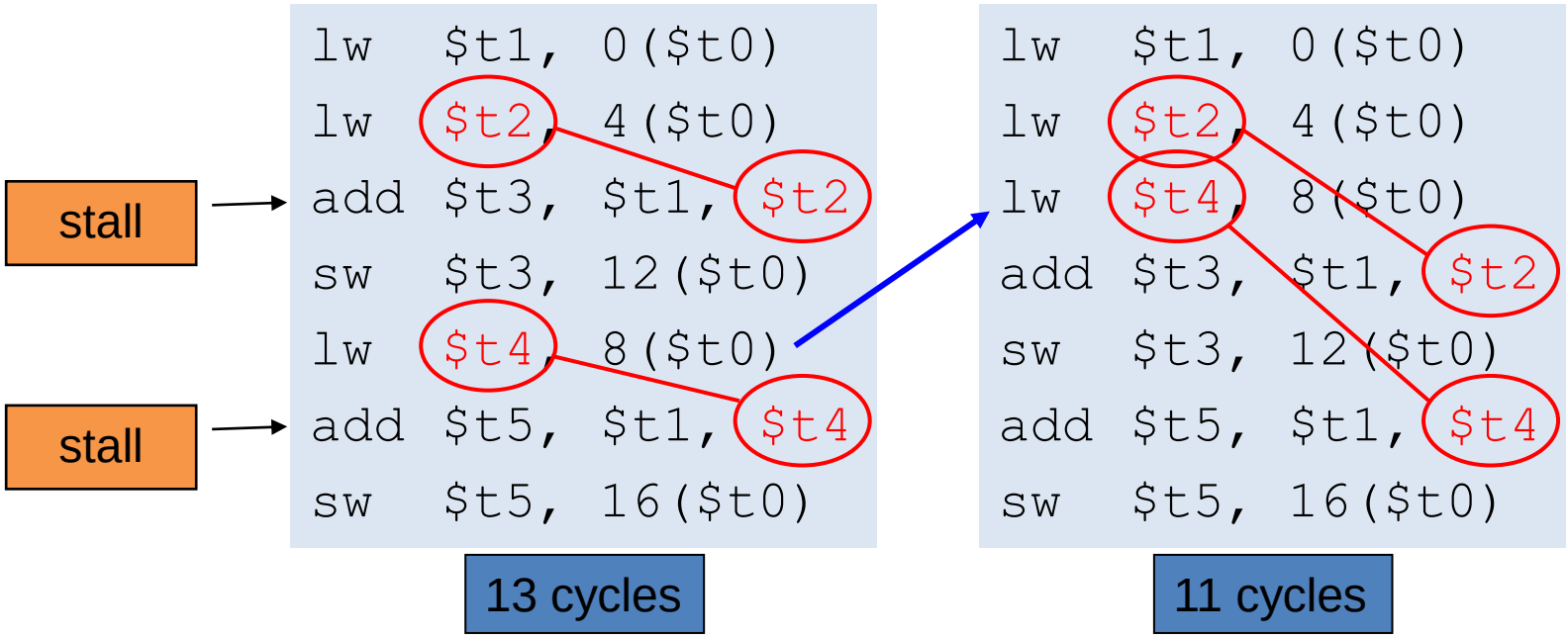


Forwarding with Load-use Data Hazards



Code Scheduling to Avoid Stalls

- Reorder code to avoid use of load result in the next instruction
- C code for `A = B + E; C = B + F;`

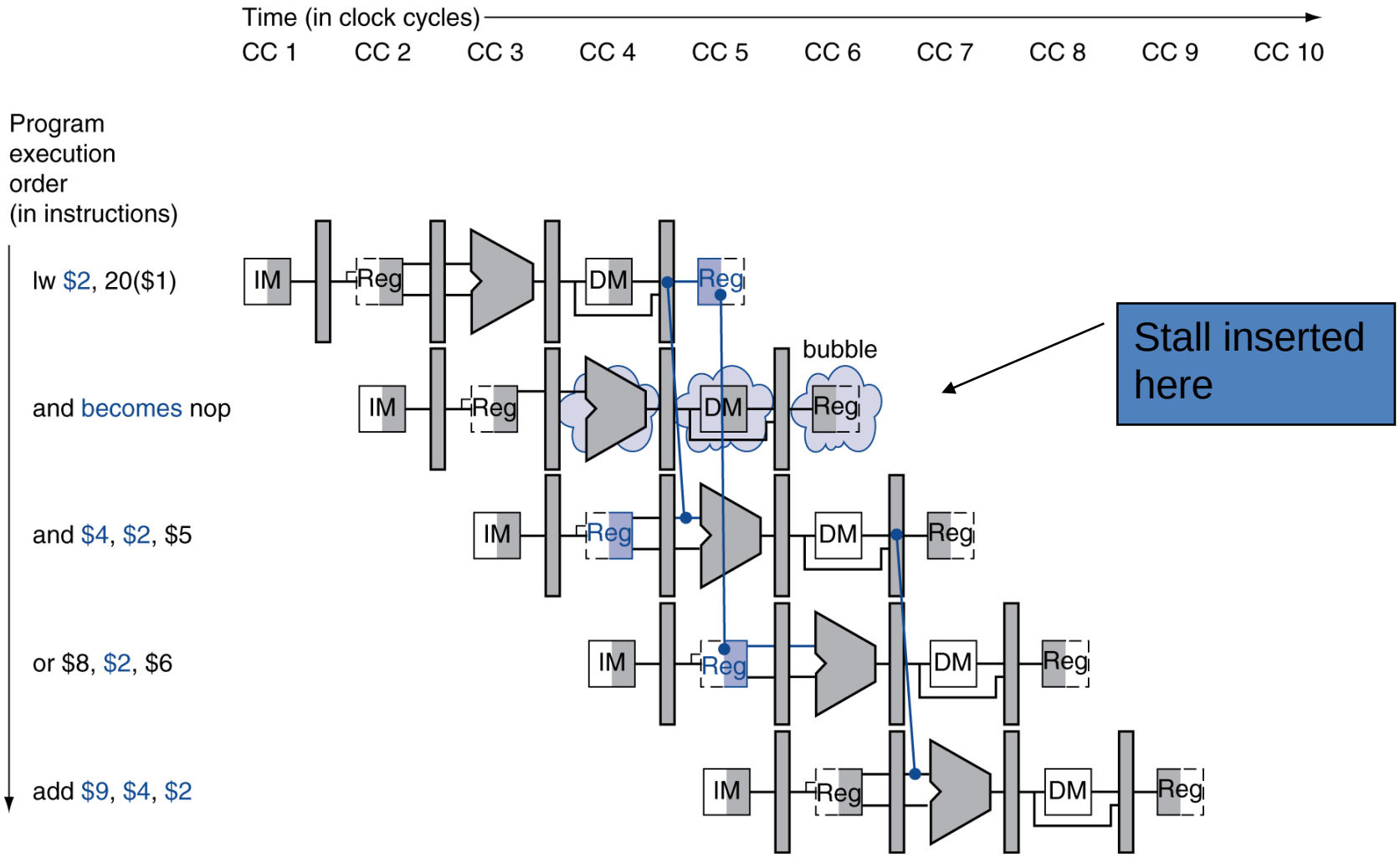


How to Stall the Pipeline

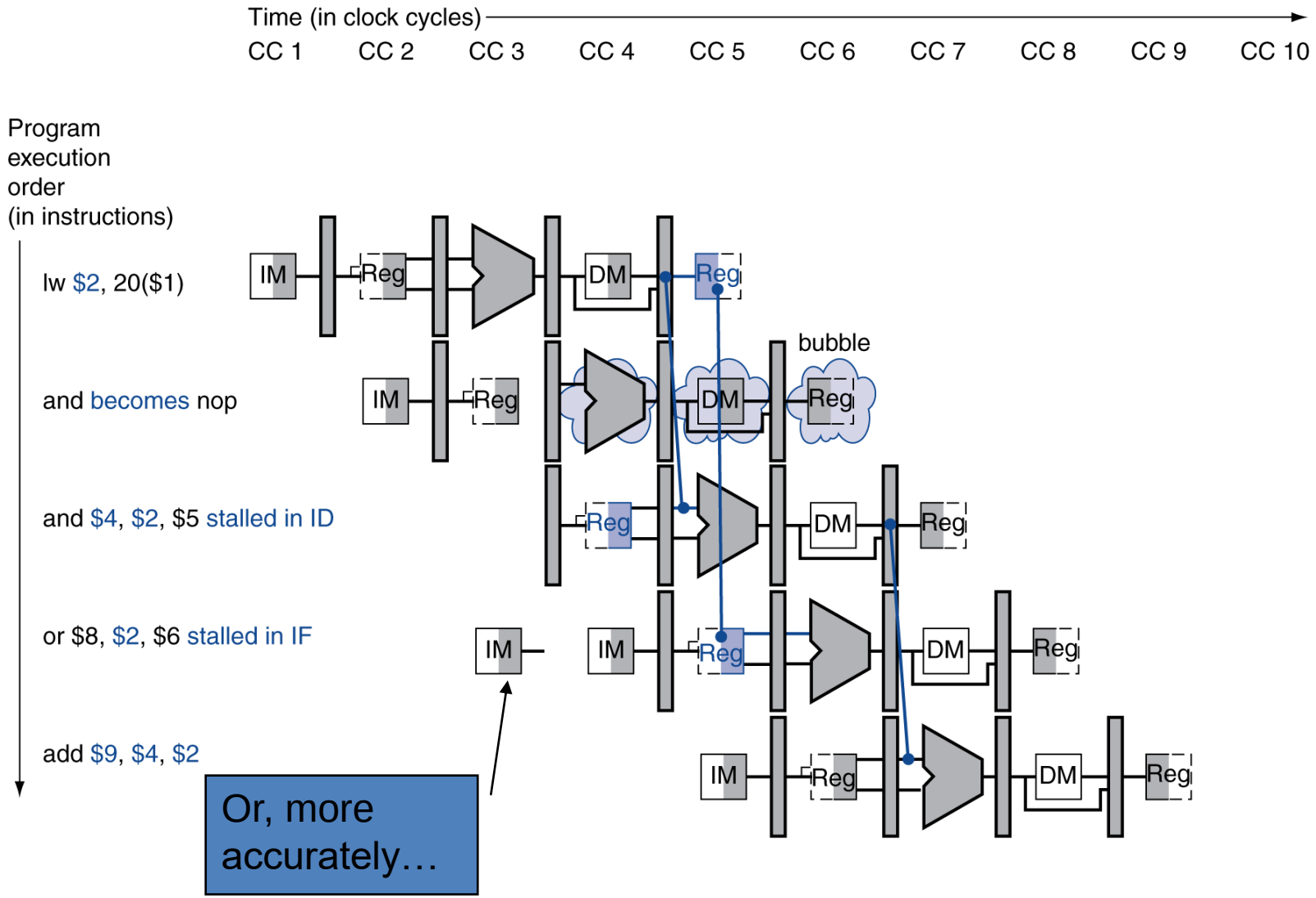
- Force control values in ID/EX register to 0
  - EX, MEM and WB do `nop` (no-operation)

- Prevent update of PC and IF/ID register
  - Using 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

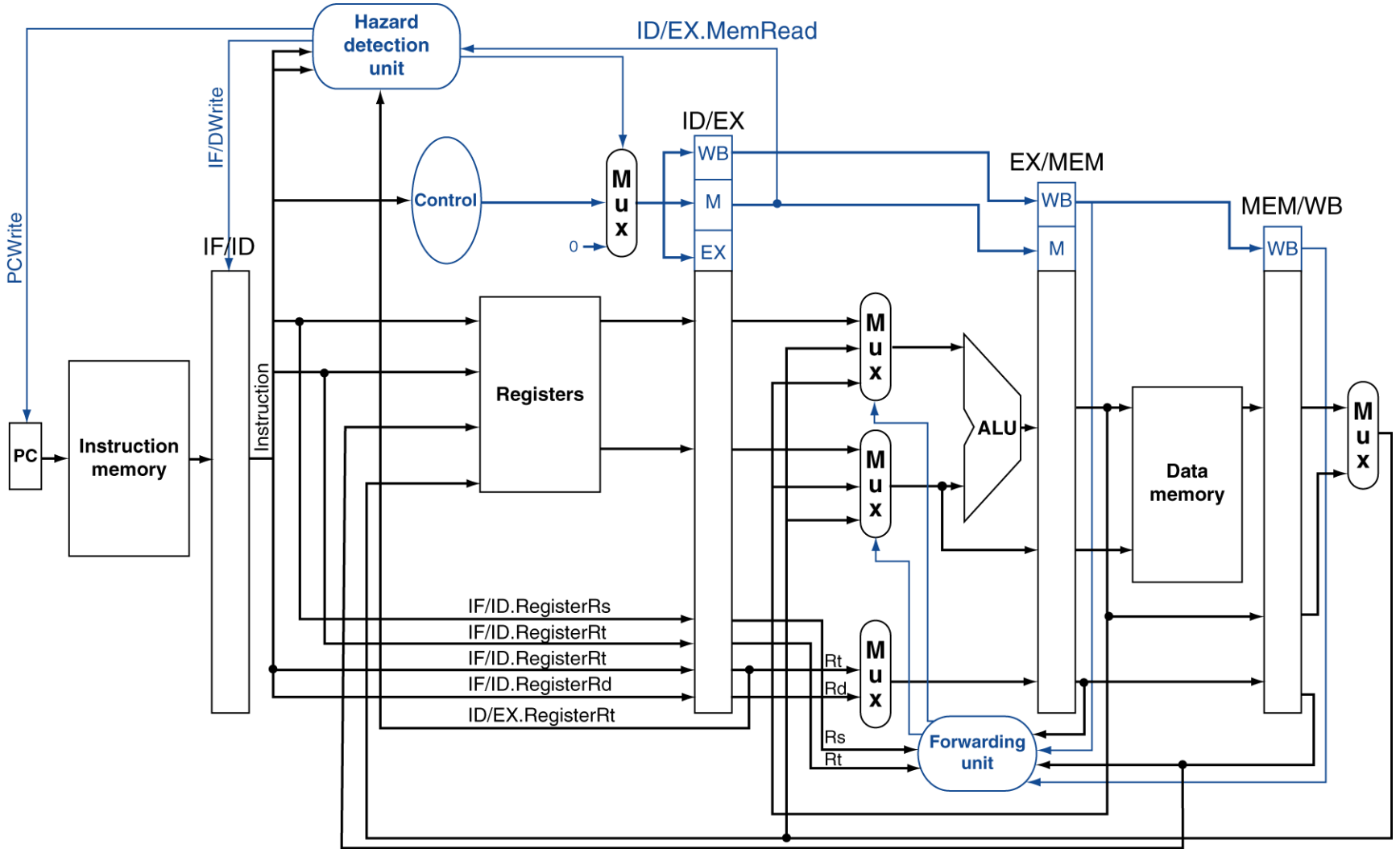
Stall/Bubble in the Pipeline



Stall/Bubble in the Pipeline



Datapath with Hazard Detection



Stalls and Performance

The BIG Picture

- Stalls reduce performance
  - But are required to get correct results
- Compiler can arrange code to avoid hazards and stalls
  - Requires knowledge of the pipeline structure

Conclusion

- All modern day processors use pipelining for performance
  - a CPI of 1 and faster CC
- Pipeline clock rate limited by slowest pipeline stage
  - designing a balanced pipeline is important
- Must detect and resolve hazards
  - Structural hazards
    - resolved by designing the pipeline correctly
  - Data hazards
    - Stall (impacts CPI)
    - Forward (requires hardware support)

Next Class

- Reducing pipeline control hazards
- Exceptions and Interrupts

The Processor: Improving Performance Data Hazards

Computer Organization

Sunday, 09 October 2022

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

TÉCNICO LISBOA