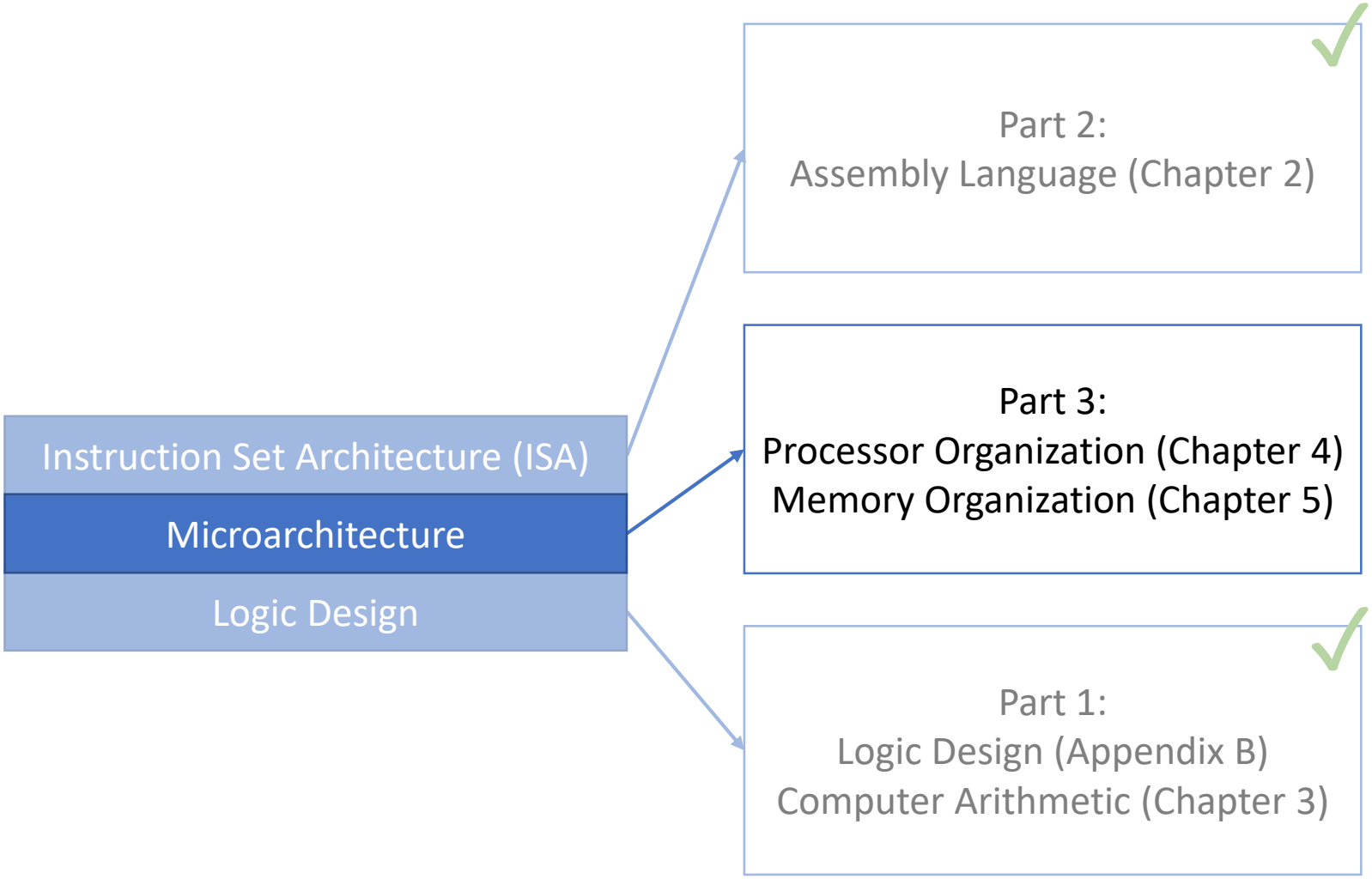
Lecture 26: Stalling, Branch Data Hazards

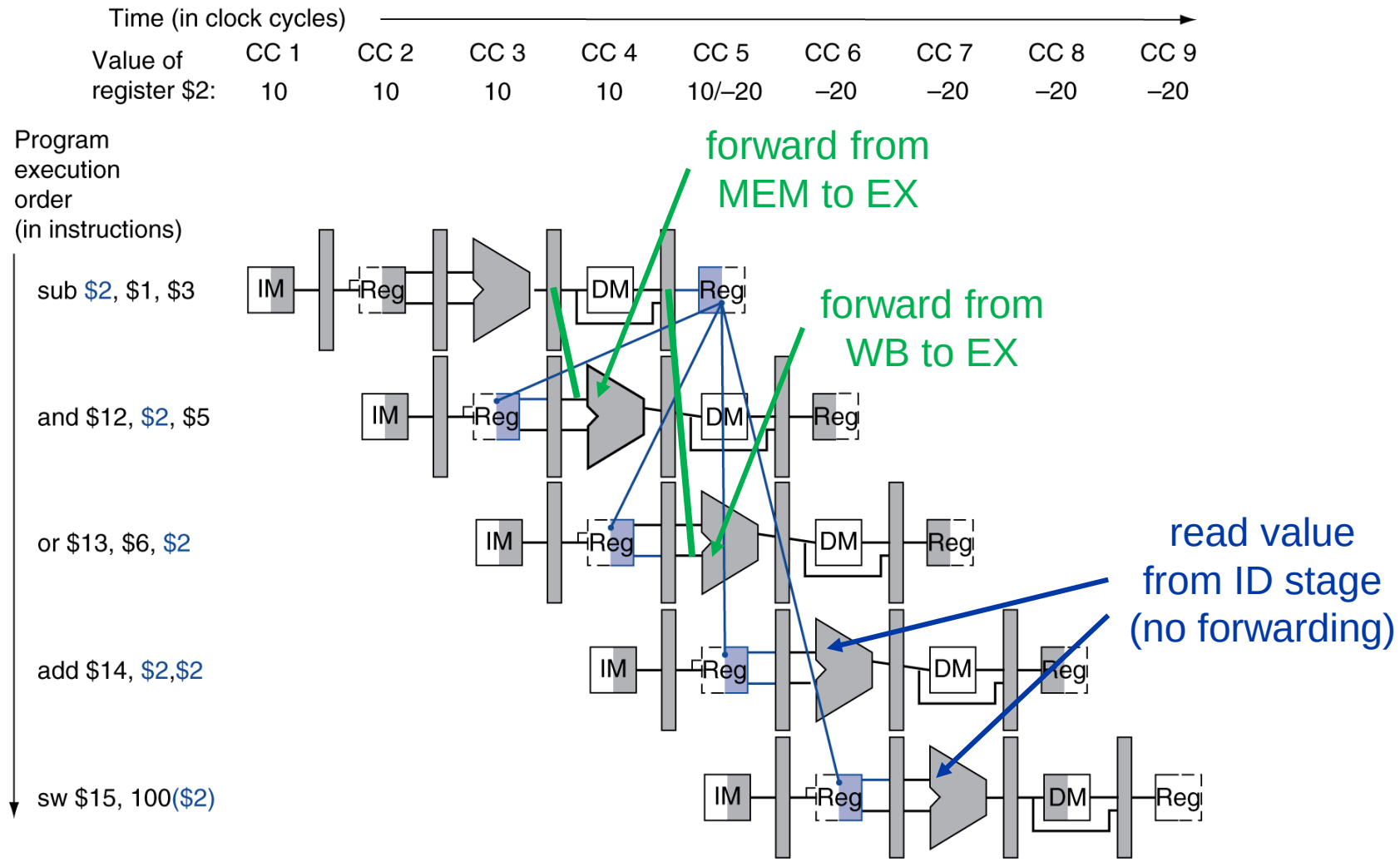
CMPS 221 – Computer Organization and Design

Adapted by Izzat El Hajj from textbook slides for Computer Organization and Design, by David Patterson and John Hennessy, Morgan Kaufmann.

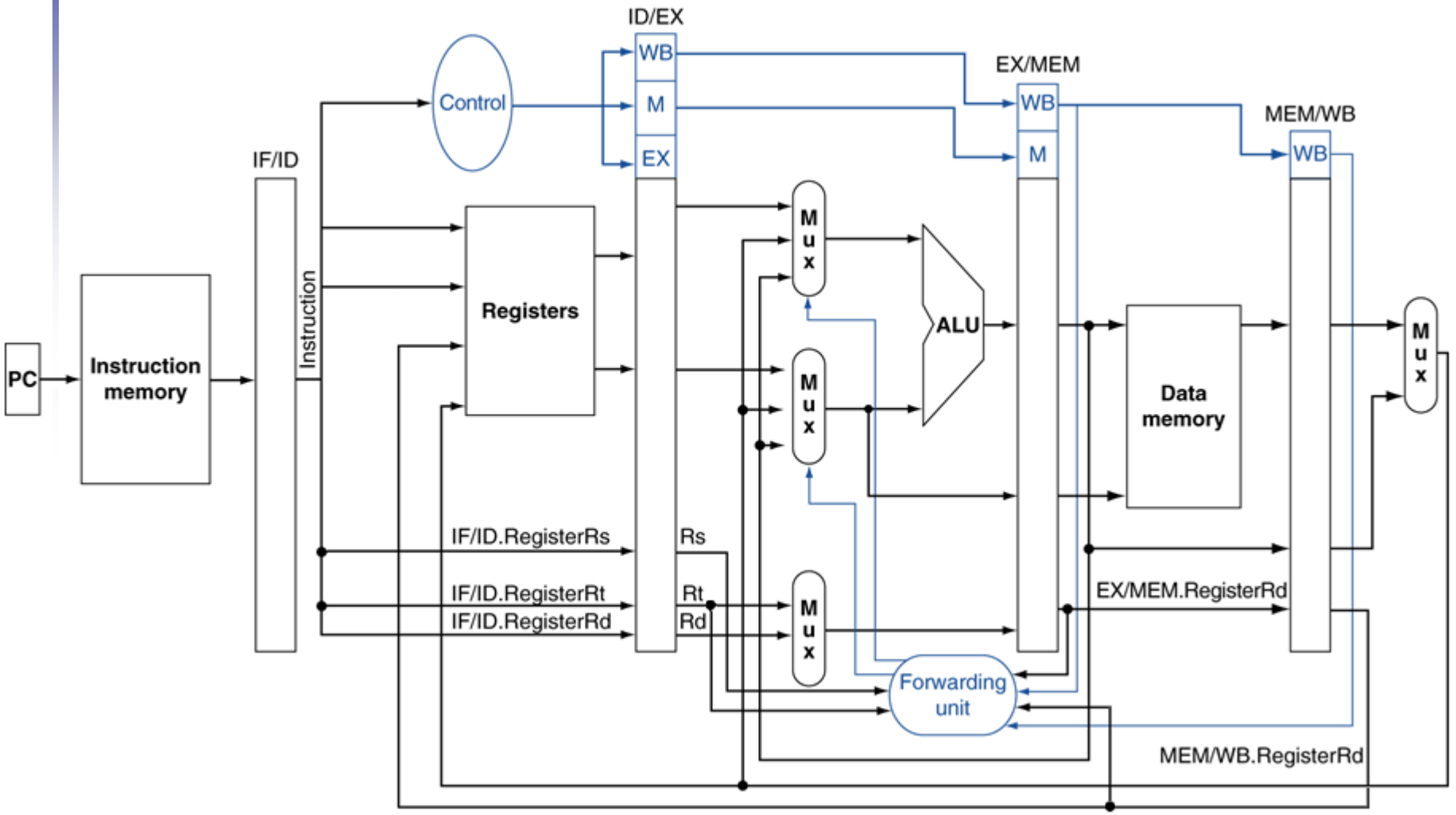
Last Time: Forwarding



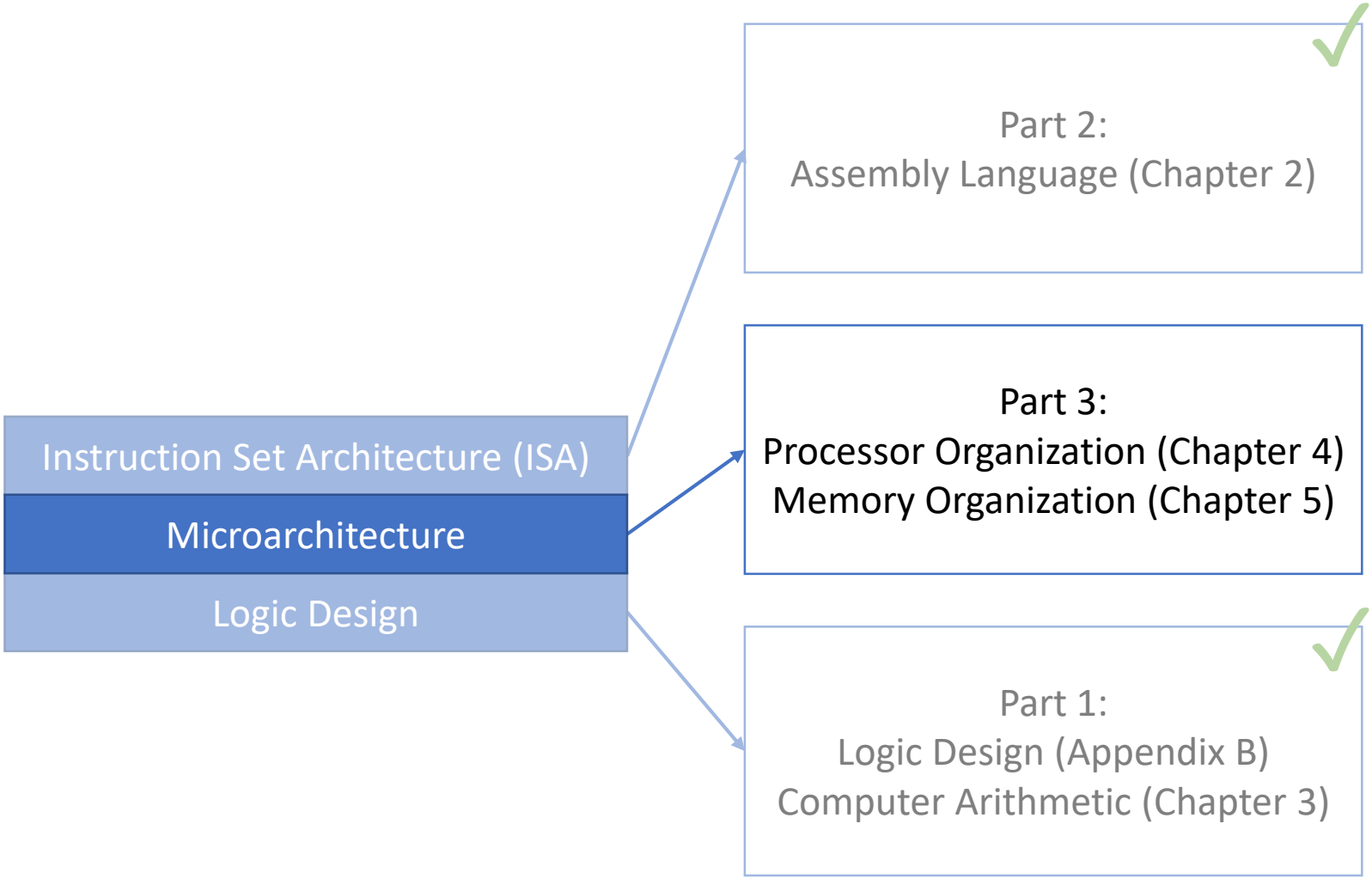
Dependencies & Forwarding



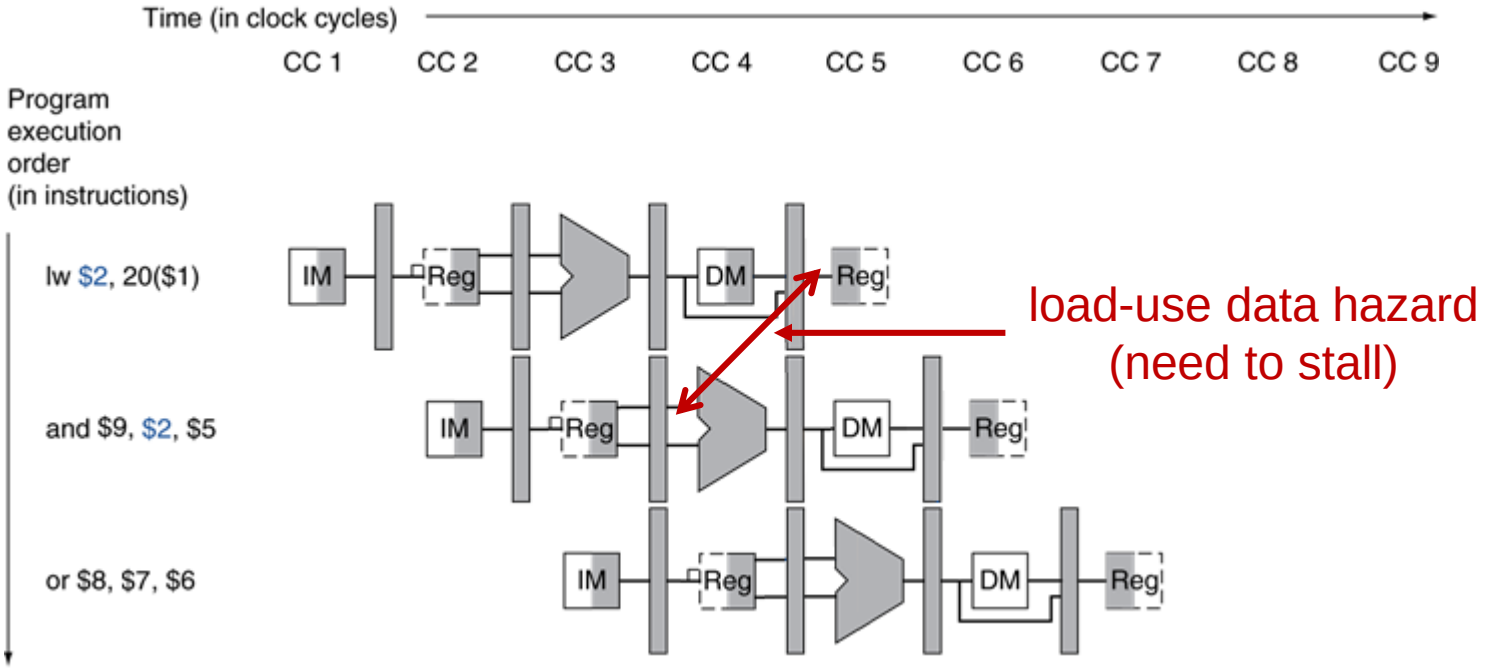
Datapath with Forwarding



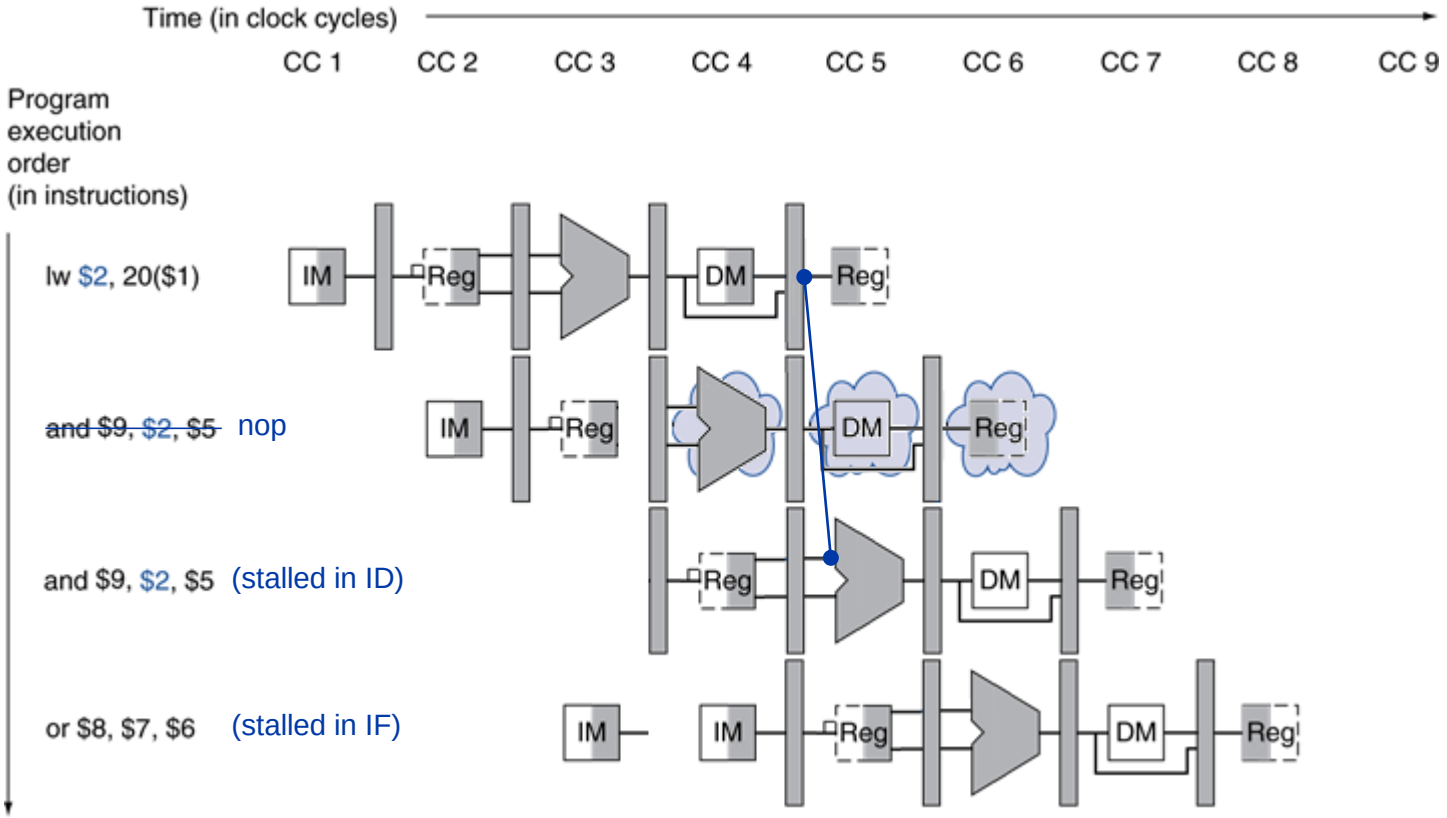
Today: Stalling, Control Hazards



Load-Use Data Hazard

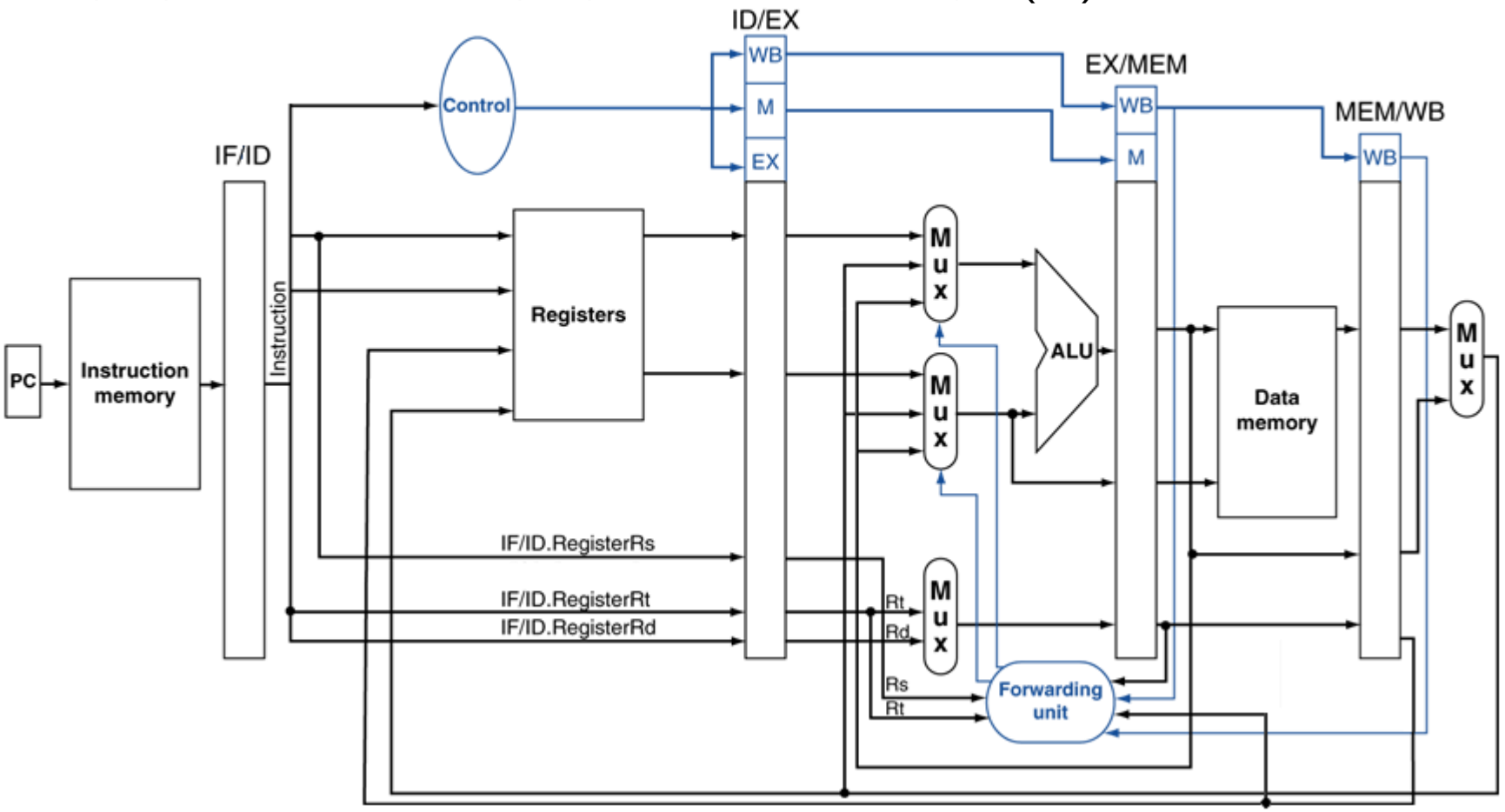


Load-Use Data Hazard



Single Cycle Pipeline Diagram

or $8, $7, $6 and $9, $2, $5 lw $2, 20($1)



Single Cycle Pipeline Diagram

stall or in IF

or $8, $7, $6

stall and in ID

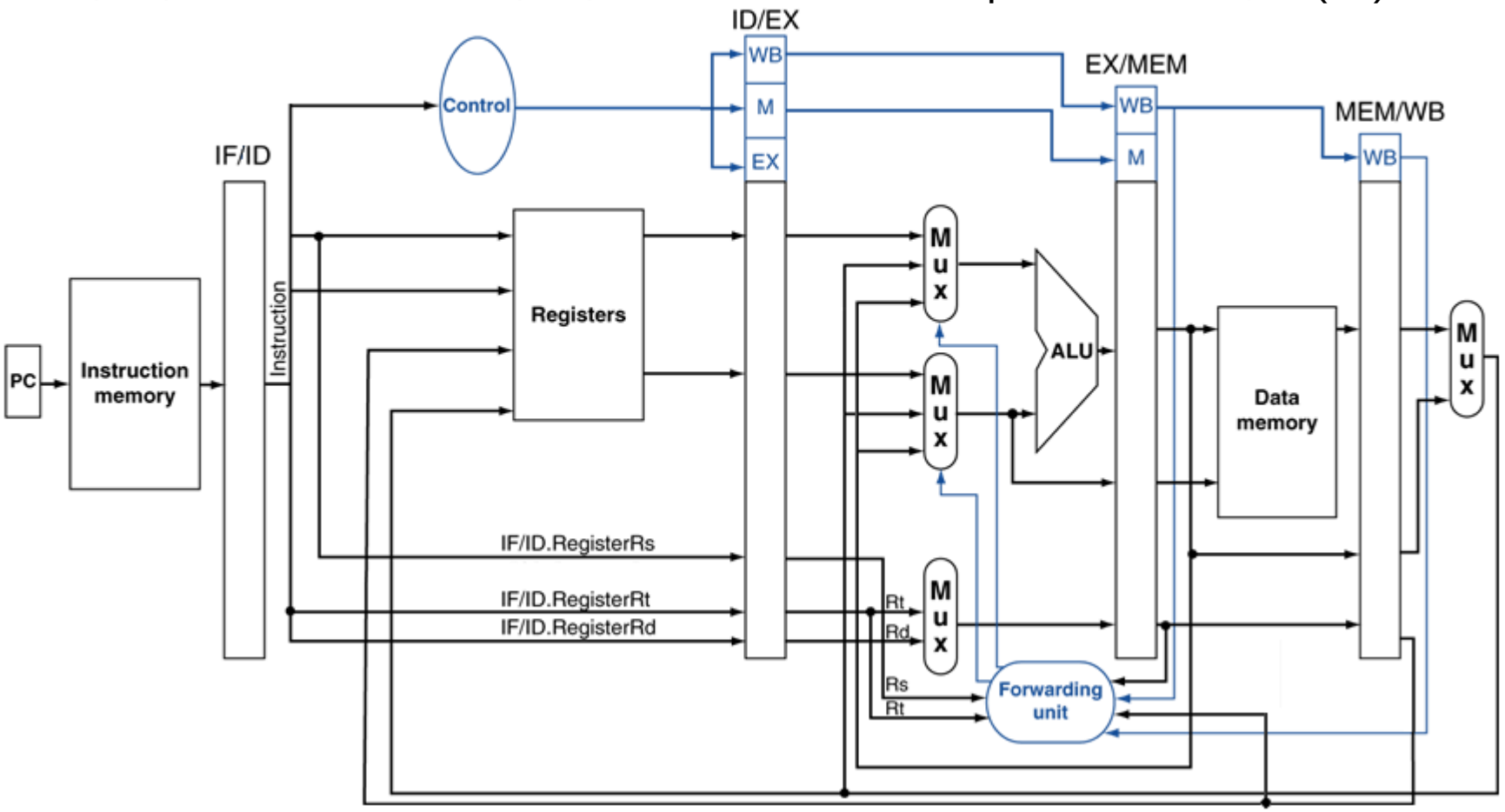
and $9, $2, $5

insert nop in EX stage

nop

allow load to proceed

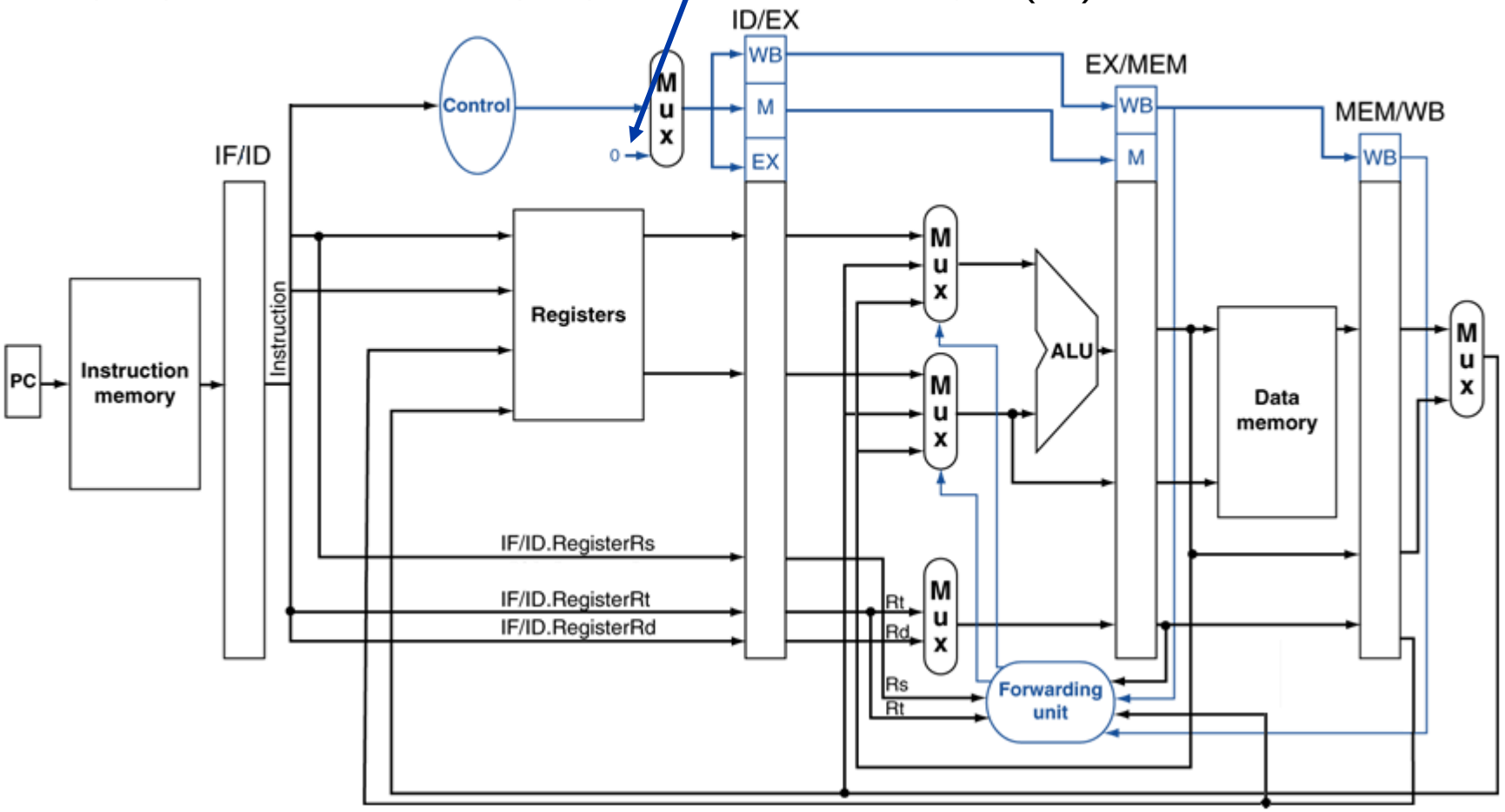
lw $2, 20($1)



Datapath with Hazard Detection

insert nop by setting all control signals to 0

or $8, $7, $6 and $9, $2, $5 lw $2, 20($1)



Datapath with Hazard Detection

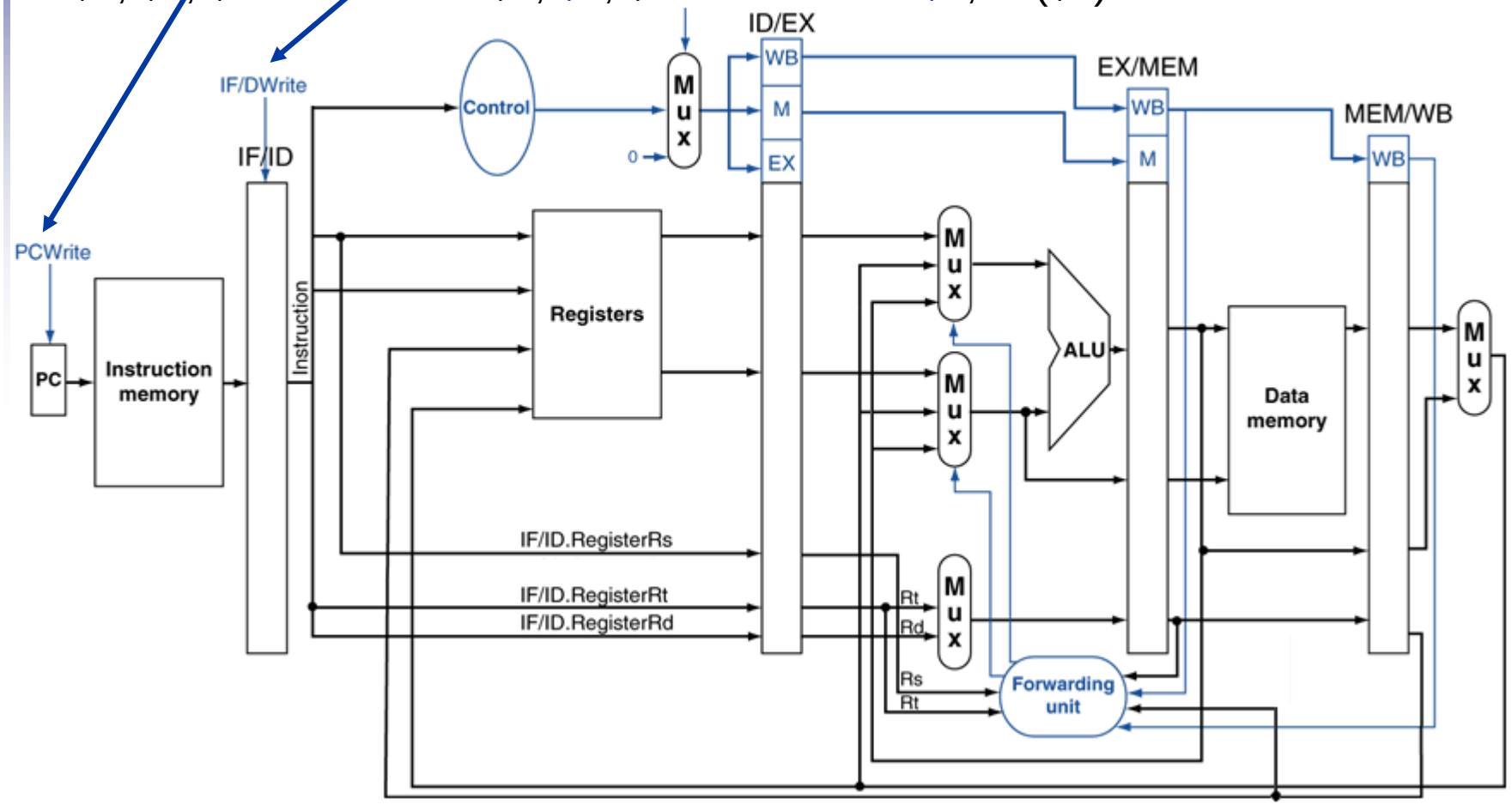
stall or in IF by disabling PCWrite

or $8, $7, $6

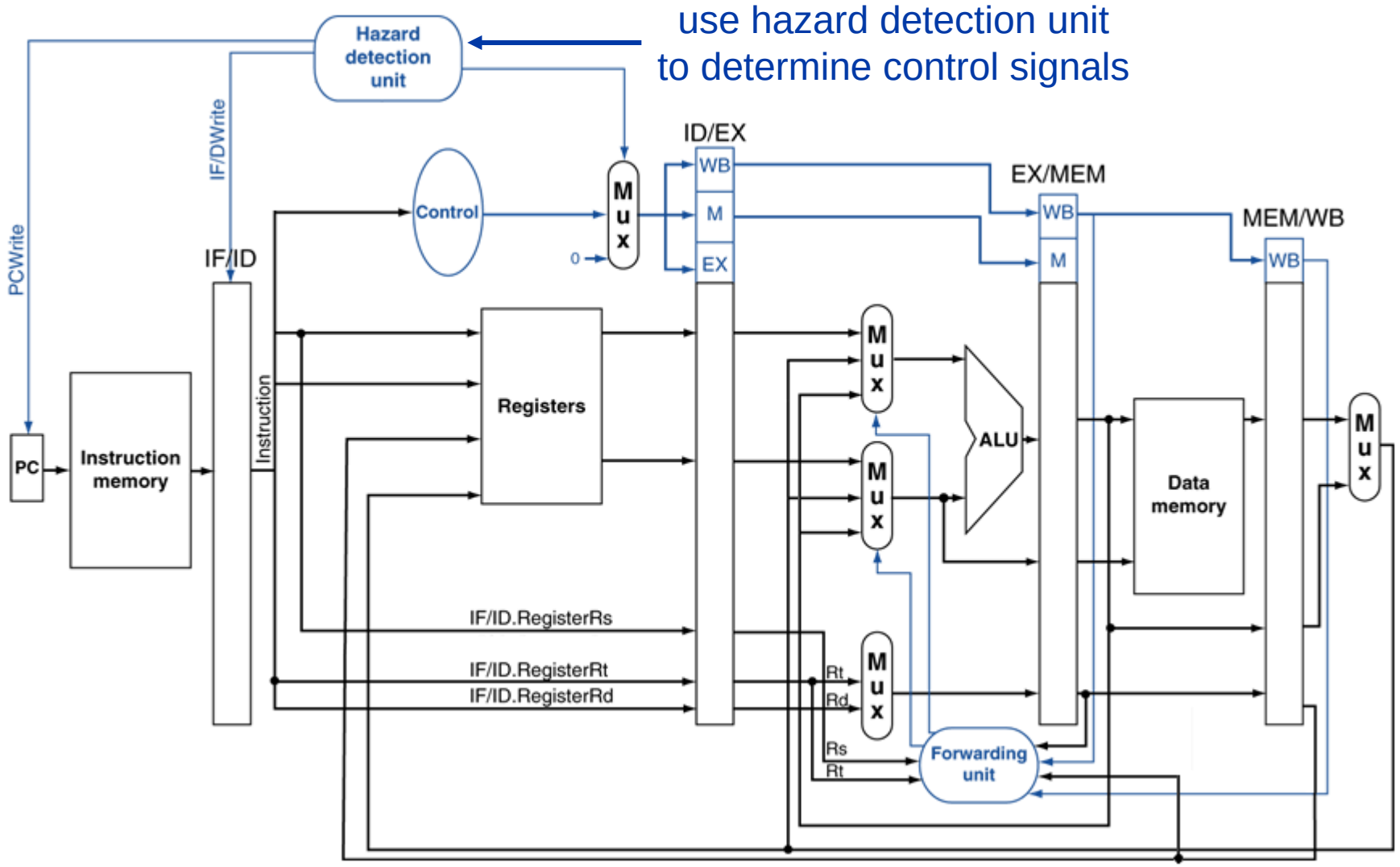
stall and in ID by disabling IF/IDWrite

and $9, $2, $5

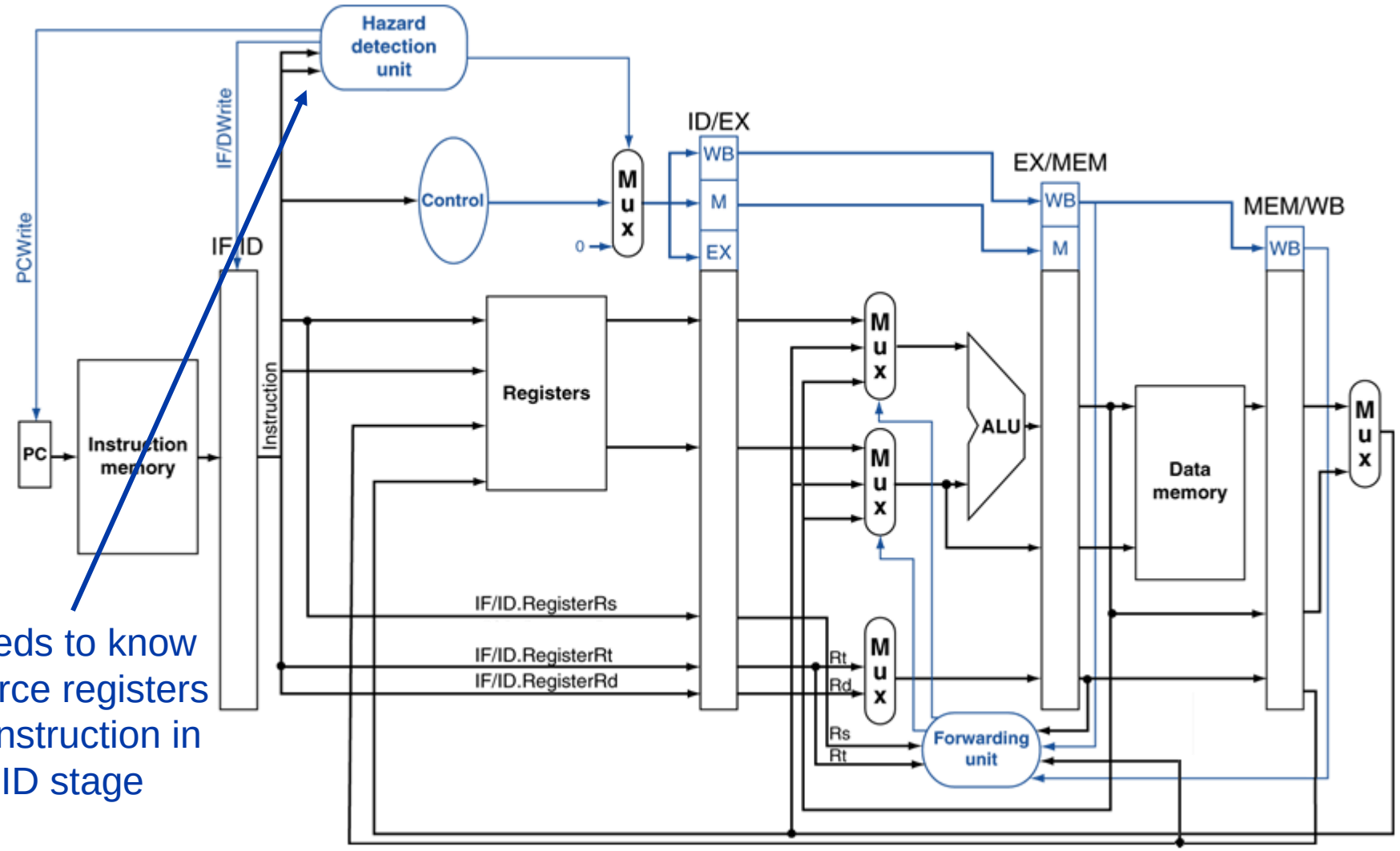
lw $2, 20($1)



Hazard Detection Unit

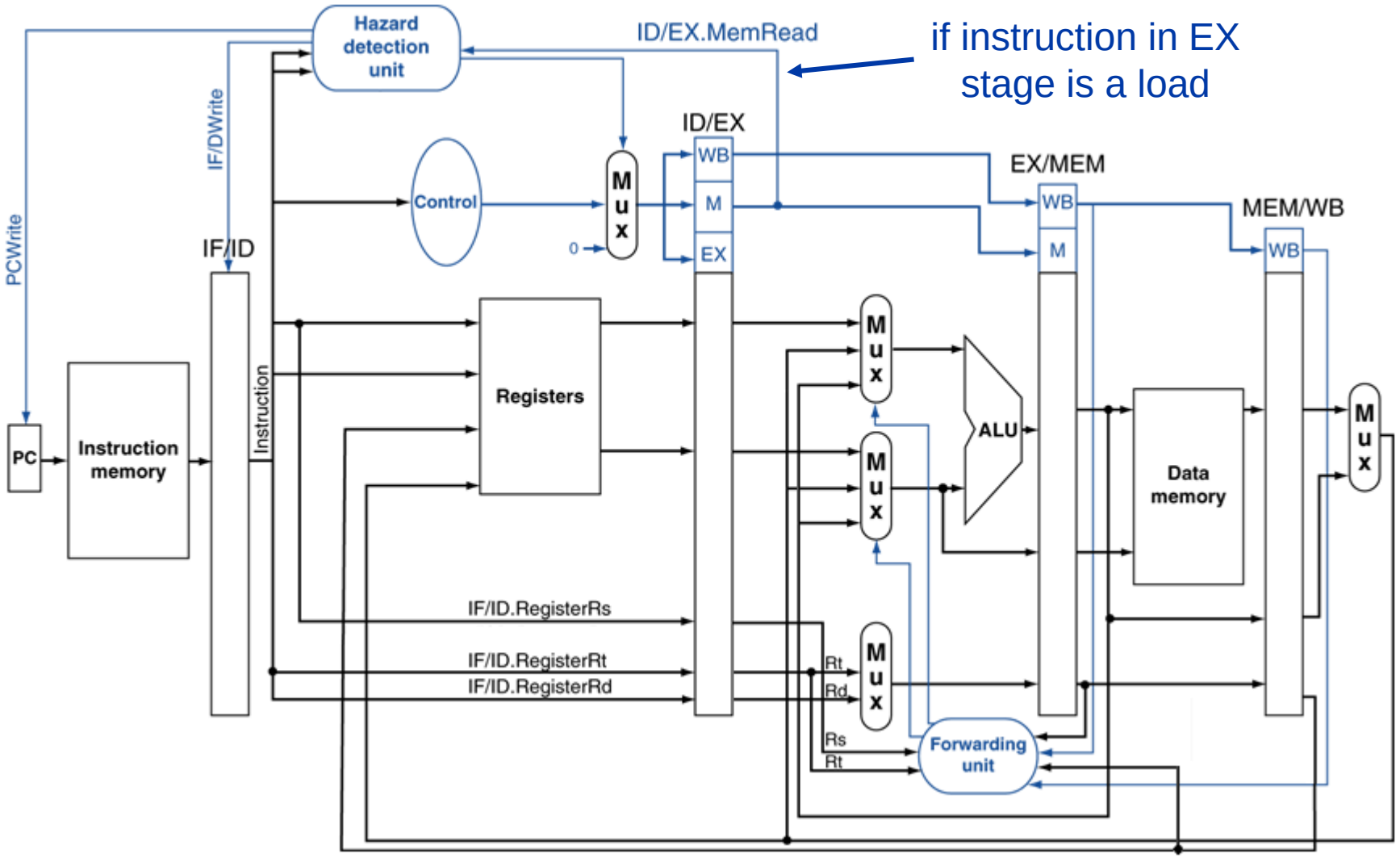


Hazard Detection Unit

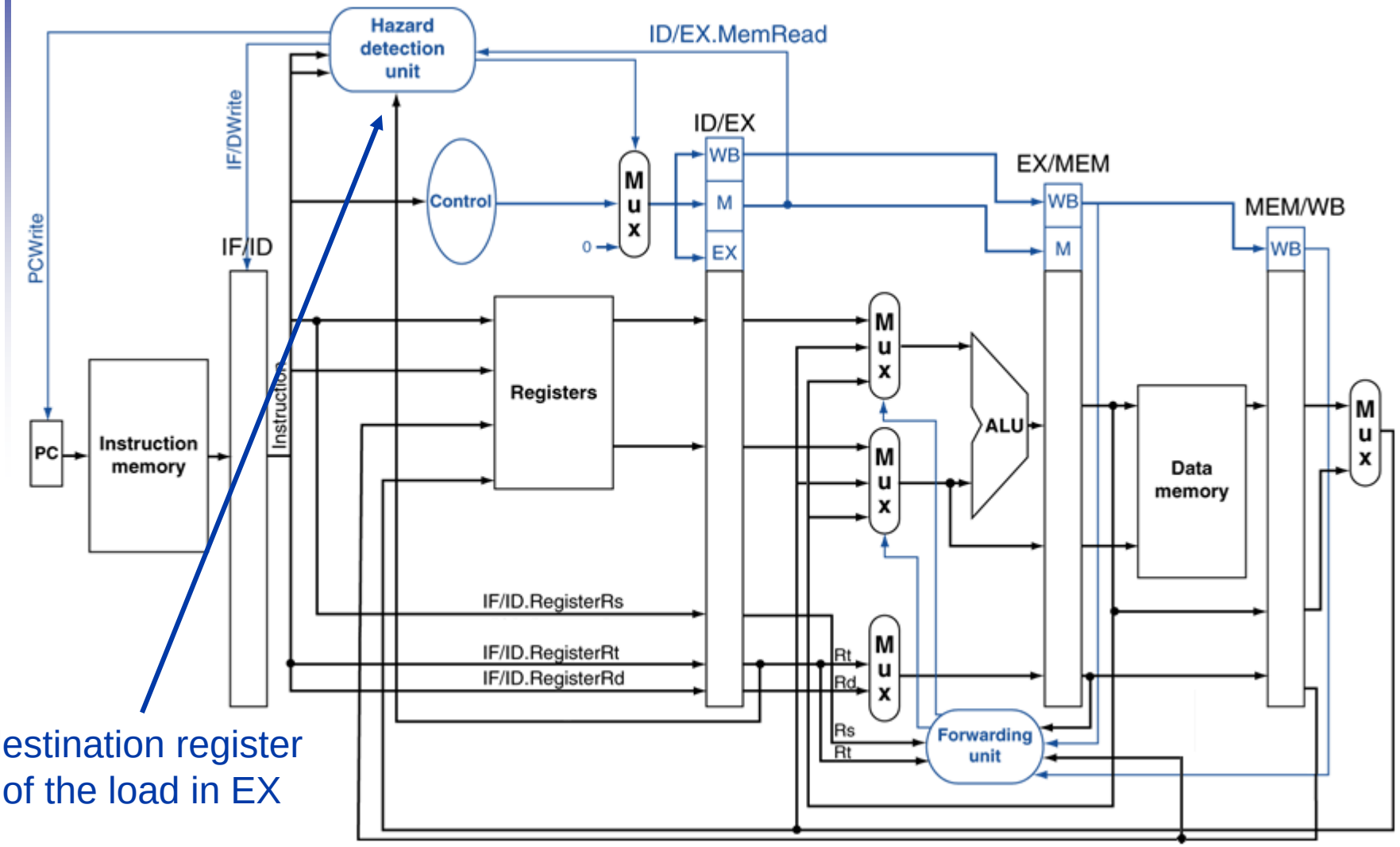


needs to know source registers of instruction in ID stage

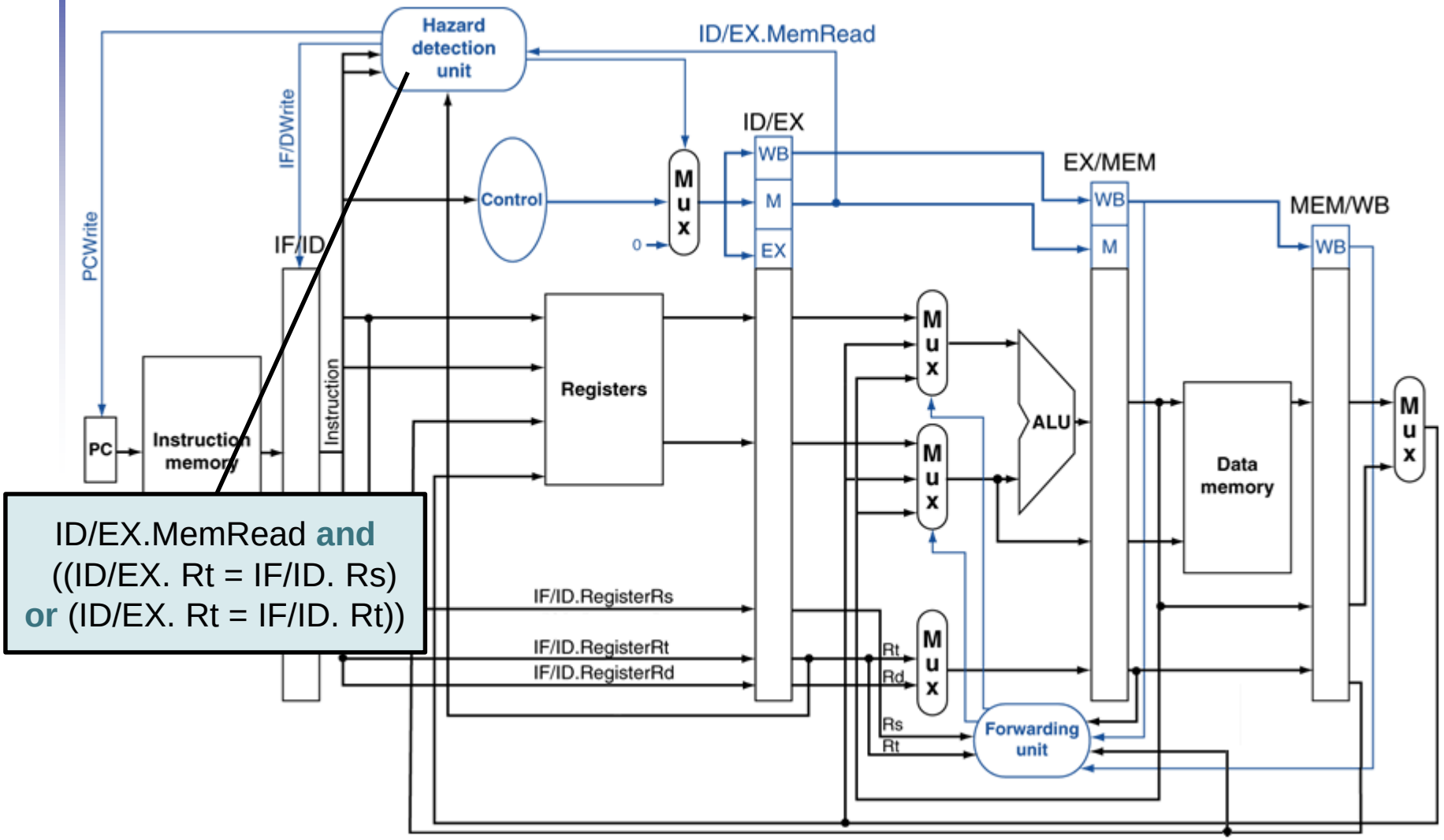
Hazard Detection Unit



Hazard Detection Unit

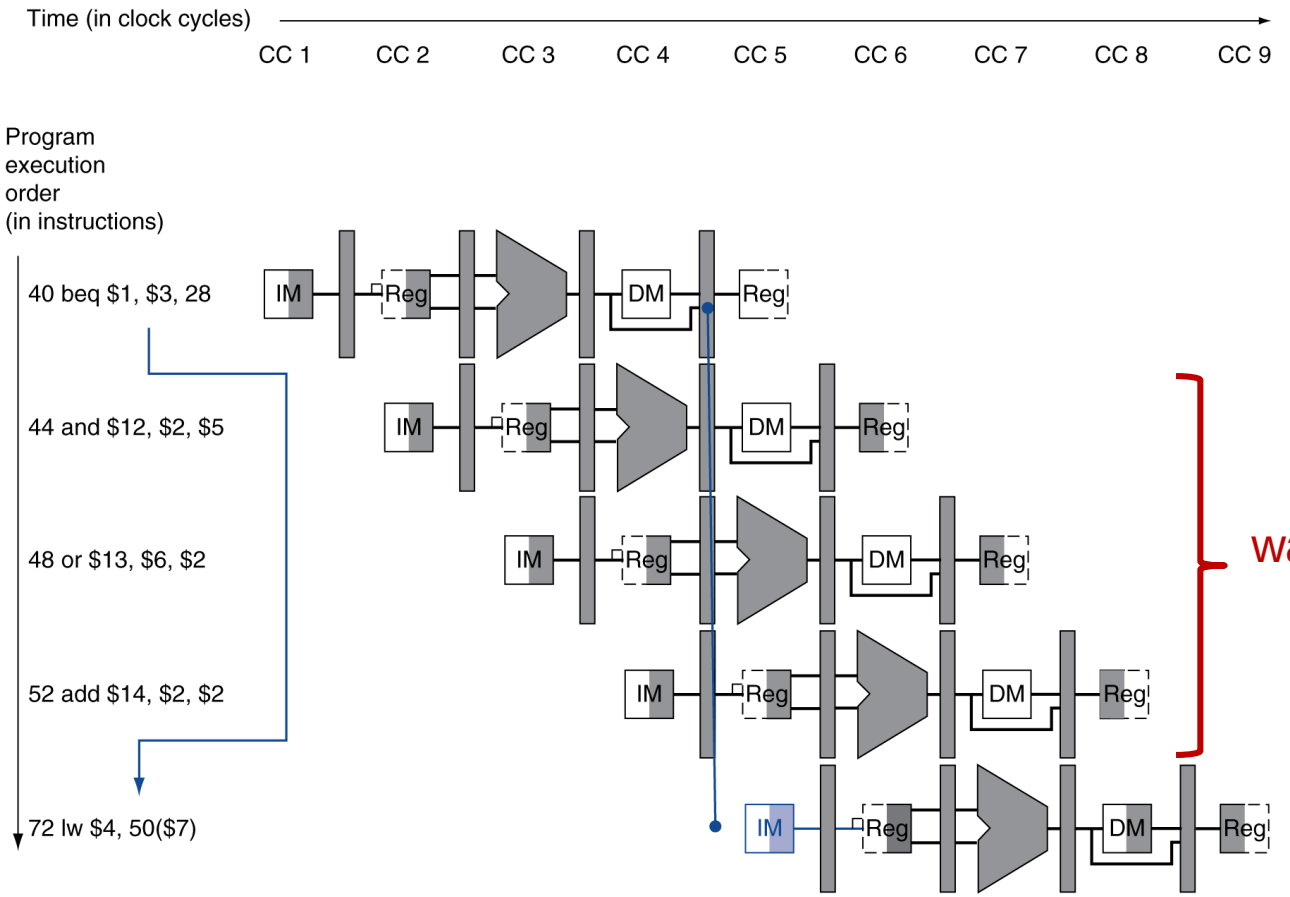


Stalling Condition



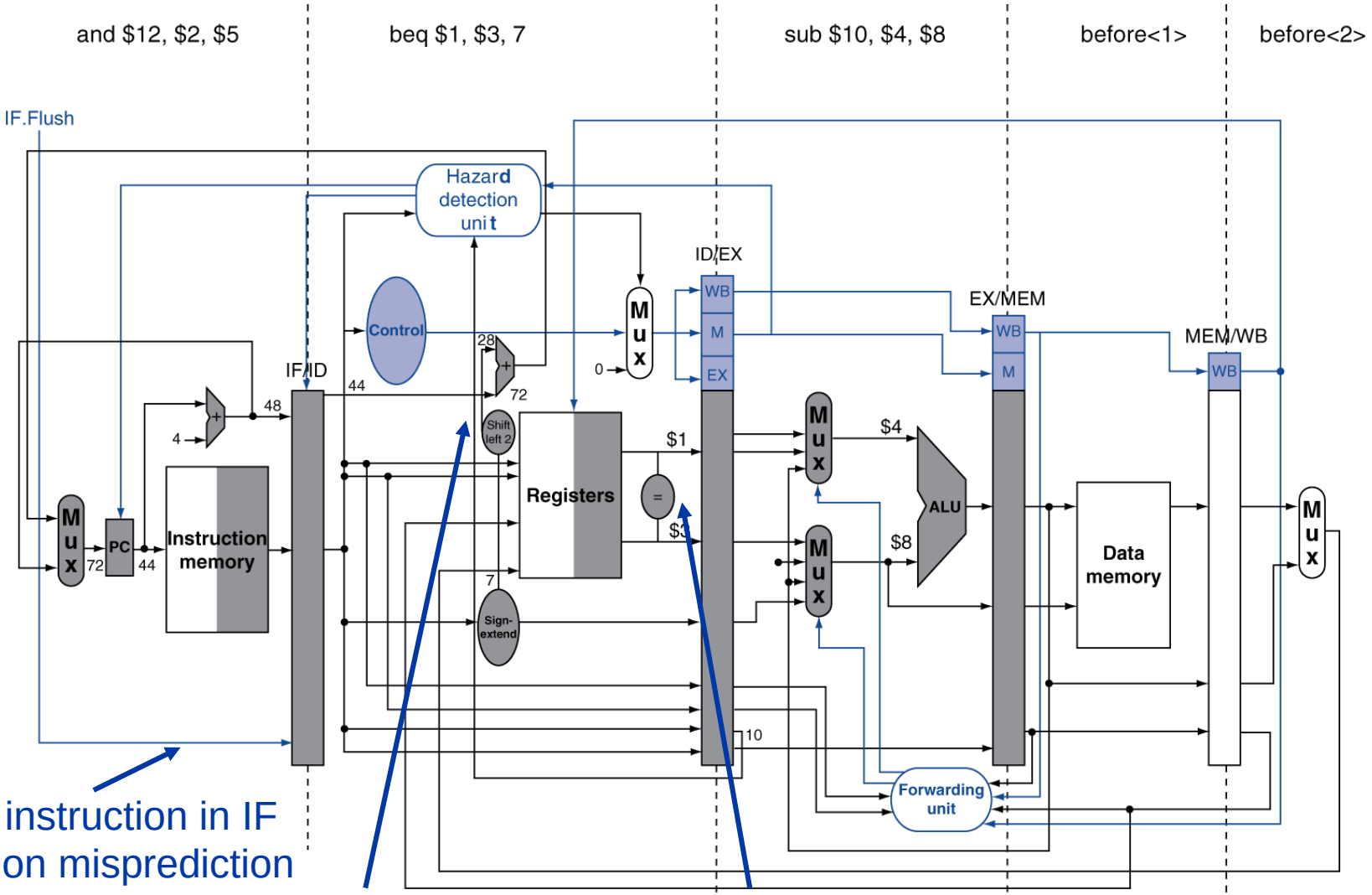
Branch Hazards

- If branch outcome determined in MEM



Recall: Solution was to resolve branch at the end of the ID stage

Reducing Branch Delay



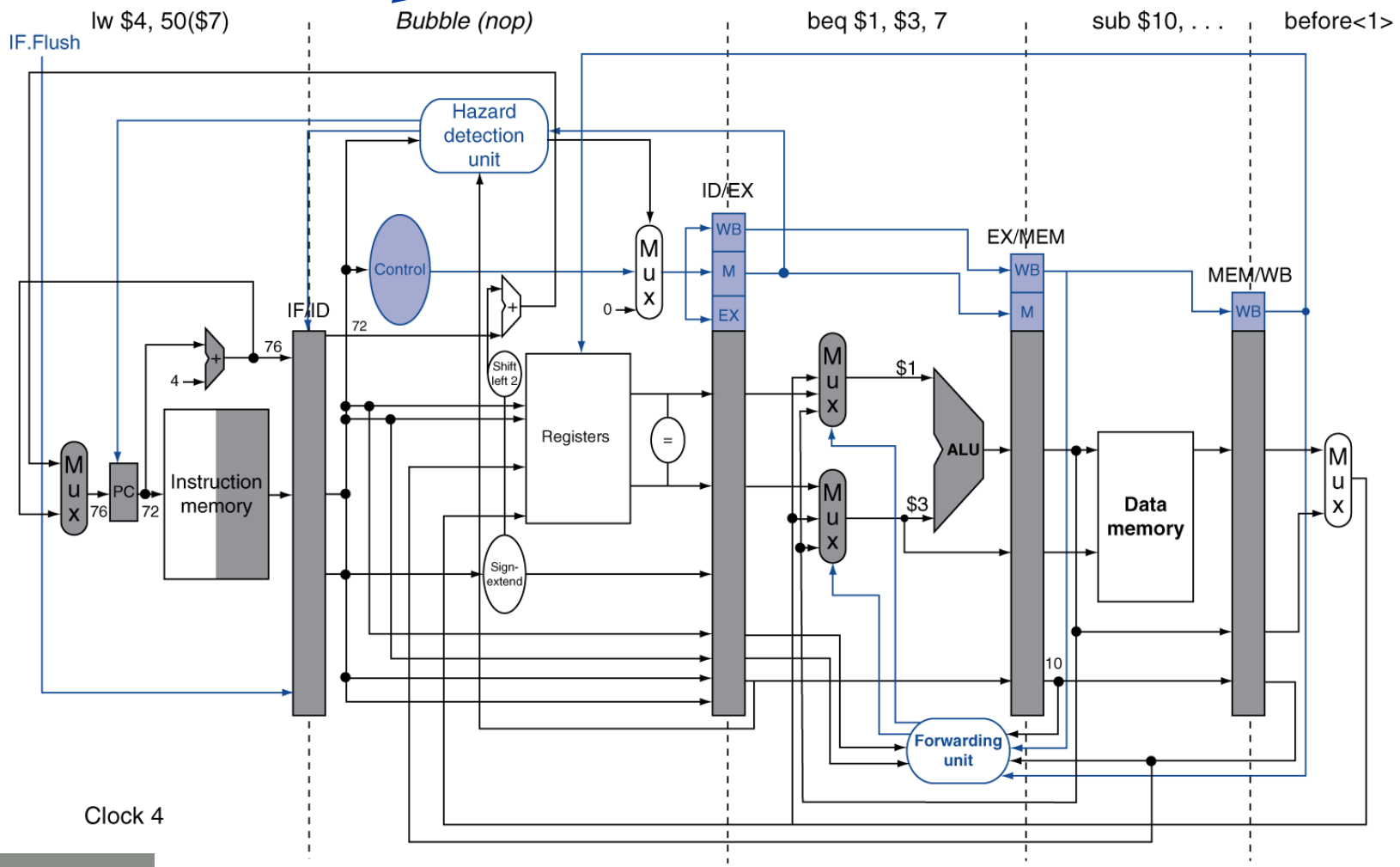
flush instruction in IF stage on misprediction

calculate branch target in ID stage

compare registers in ID stage

Reducing Branch Delay

only waste one cycle on misprediction



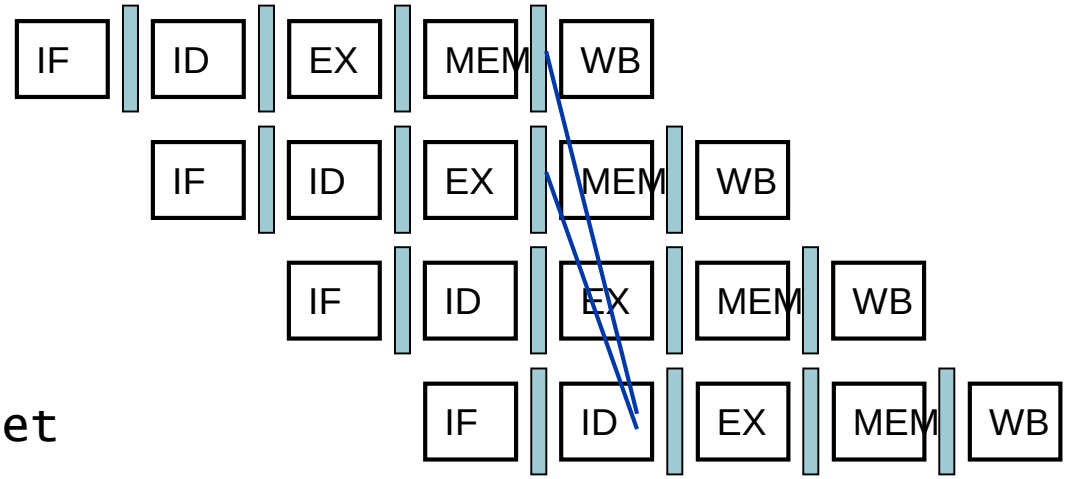
Data Hazards for Branches

- Resolving branches at the end of the ID stage reduces stalls from mispredicted branches…
- But creates **data hazards for branches** resulting in stalls when a branch uses recently computed data

Data Hazards for Branches

- If a comparison register is a destination of 2nd or 3rd preceding ALU instruction

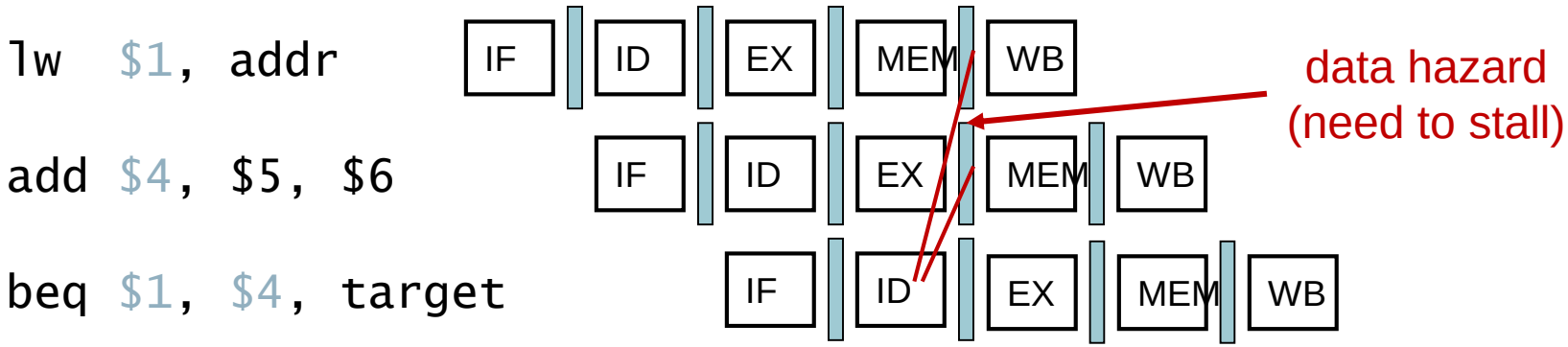
```
add $1, $2, $3
add $4, $5, $6
…
beq $1, $4, target
```



- Can resolve via forwarding to ID stage

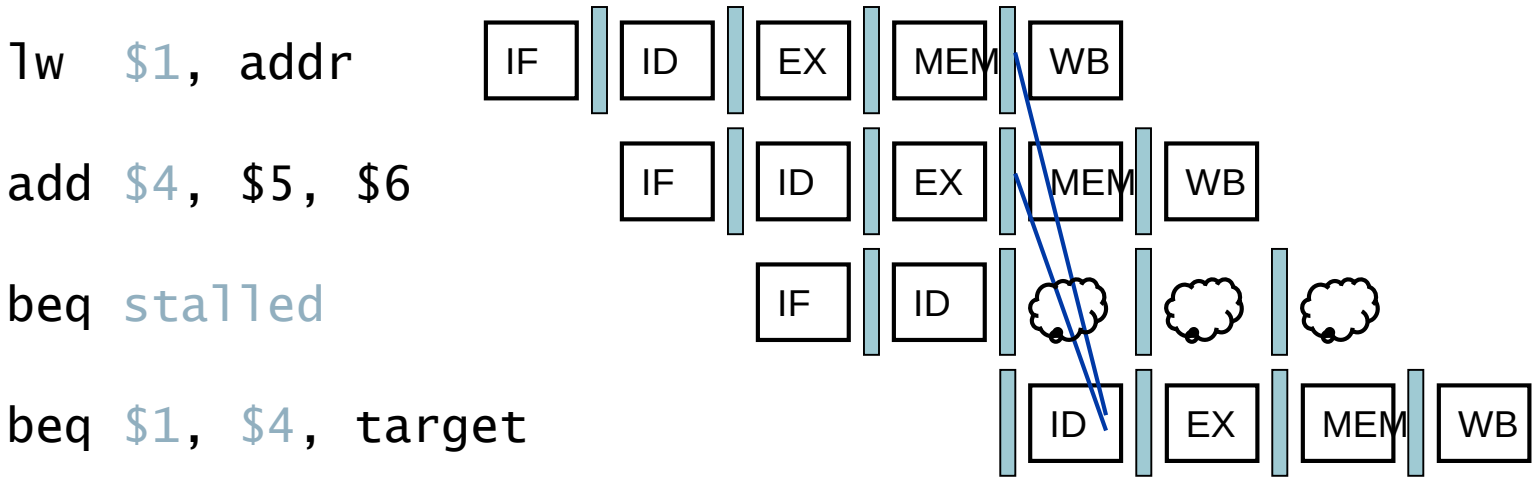
Data Hazards for Branches

- If a comparison register is a destination of preceding ALU instruction or 2nd preceding load instruction



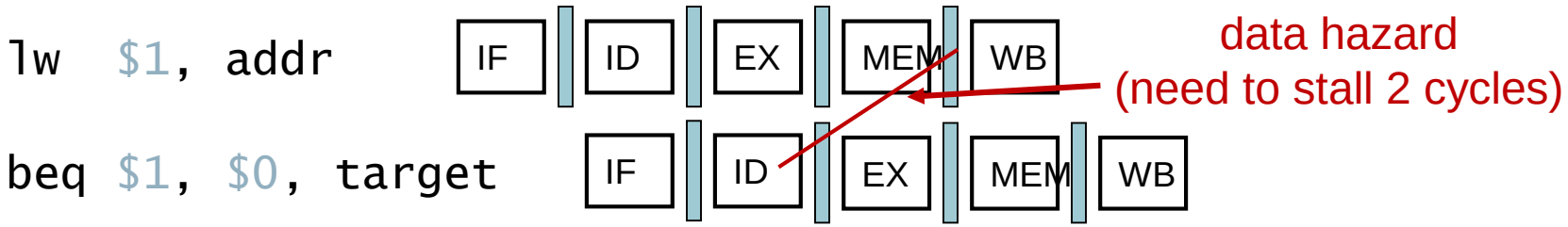
Data Hazards for Branches

- If a comparison register is a destination of preceding ALU instruction or 2nd preceding load instruction



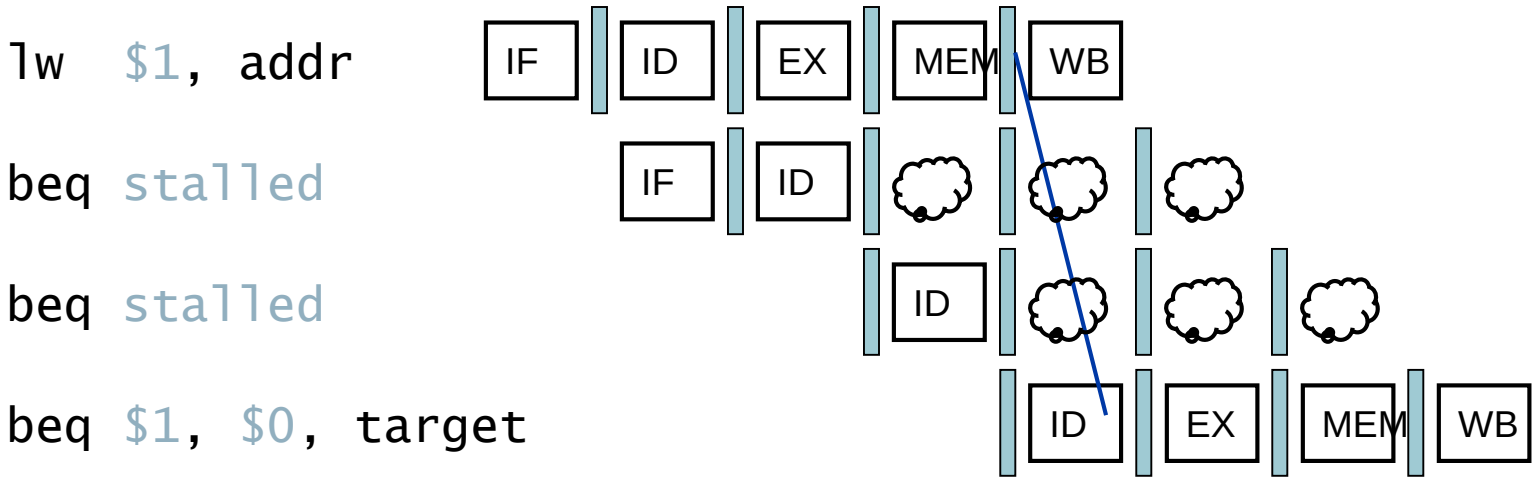
Data Hazards for Branches

- If a comparison register is a destination of immediately preceding load instruction

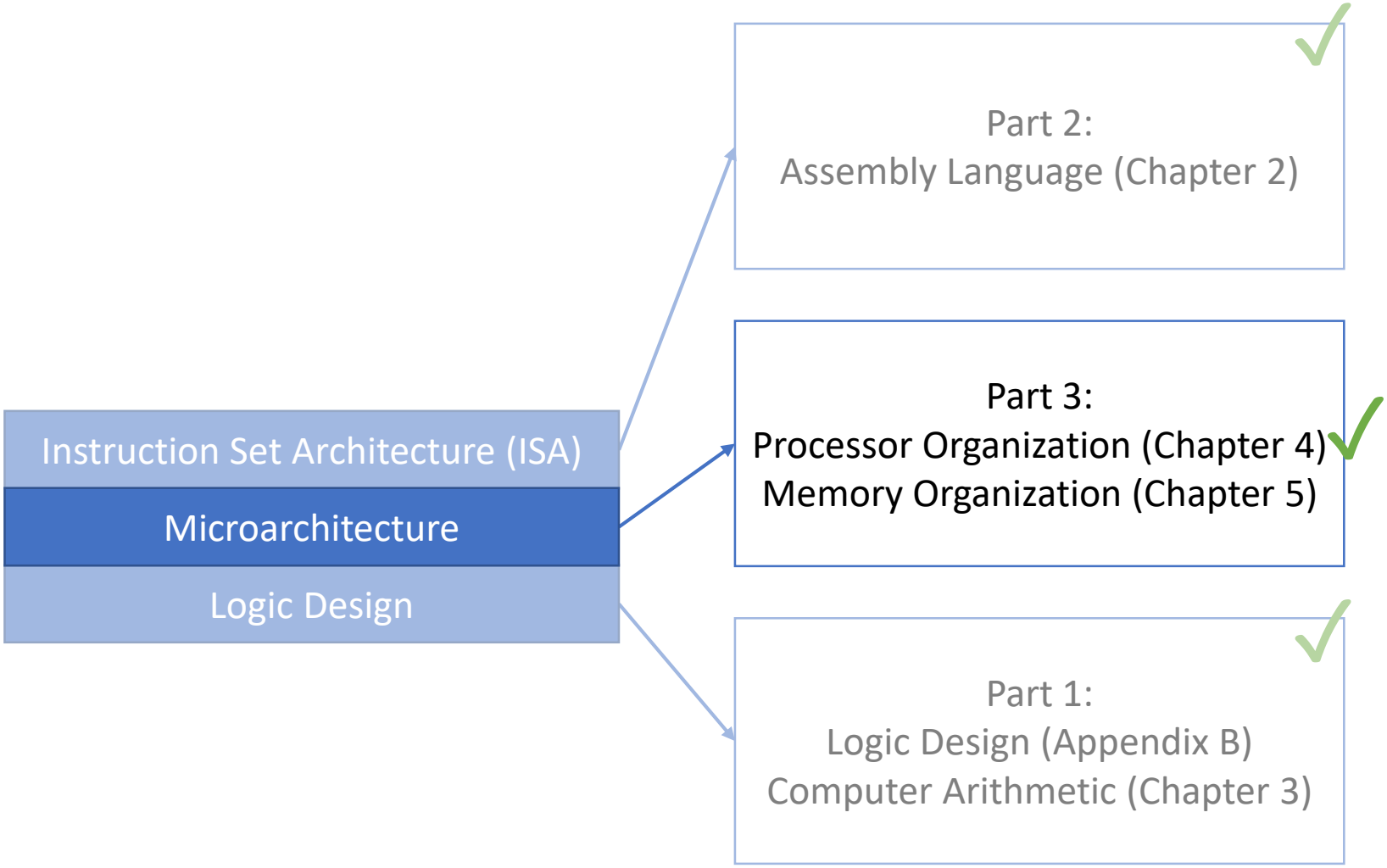


Data Hazards for Branches

- If a comparison register is a destination of immediately preceding load instruction



End of Processor Organization



Textbook Sections

- The content in these slides corresponds to:
  - Textbook:
    - *Computer Organization and Design, 5th Edition by David Patterson and John Hennessy, Morgan Kaufmann, 2014.*
  - Sections:
    - 4.7, 4.8