250P: Computer Systems Architecture

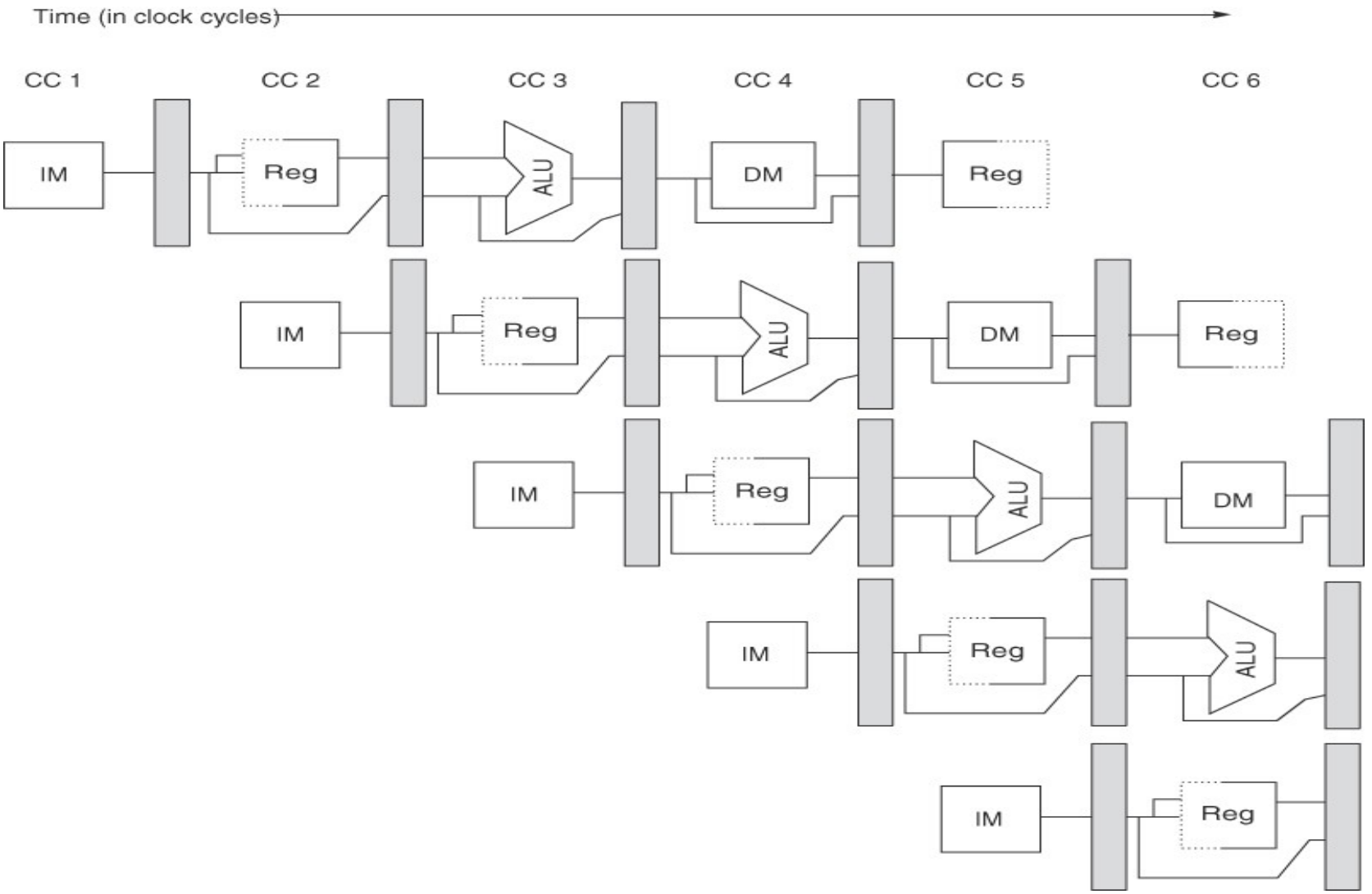
Lecture 5: Pipelining and pipelining hazards

Anton Burtsev
April, 2021

Some Equations

- Unpipelined: time to execute one instruction = $T + T_{ovh}$
- For an N-stage pipeline, time per stage = $T/N + T_{ovh}$
- Total time per instruction = $N (T/N + T_{ovh}) = T + N\ T_{ovh}$
- Clock cycle time = $T/N + T_{ovh}$
- Clock speed = $1 / (T/N + T_{ovh})$
- Ideal speedup = $(T + T_{ovh}) / (T/N + T_{ovh})$
- Cycles to complete one instruction = N
- Average CPI (cycles per instr) = 1

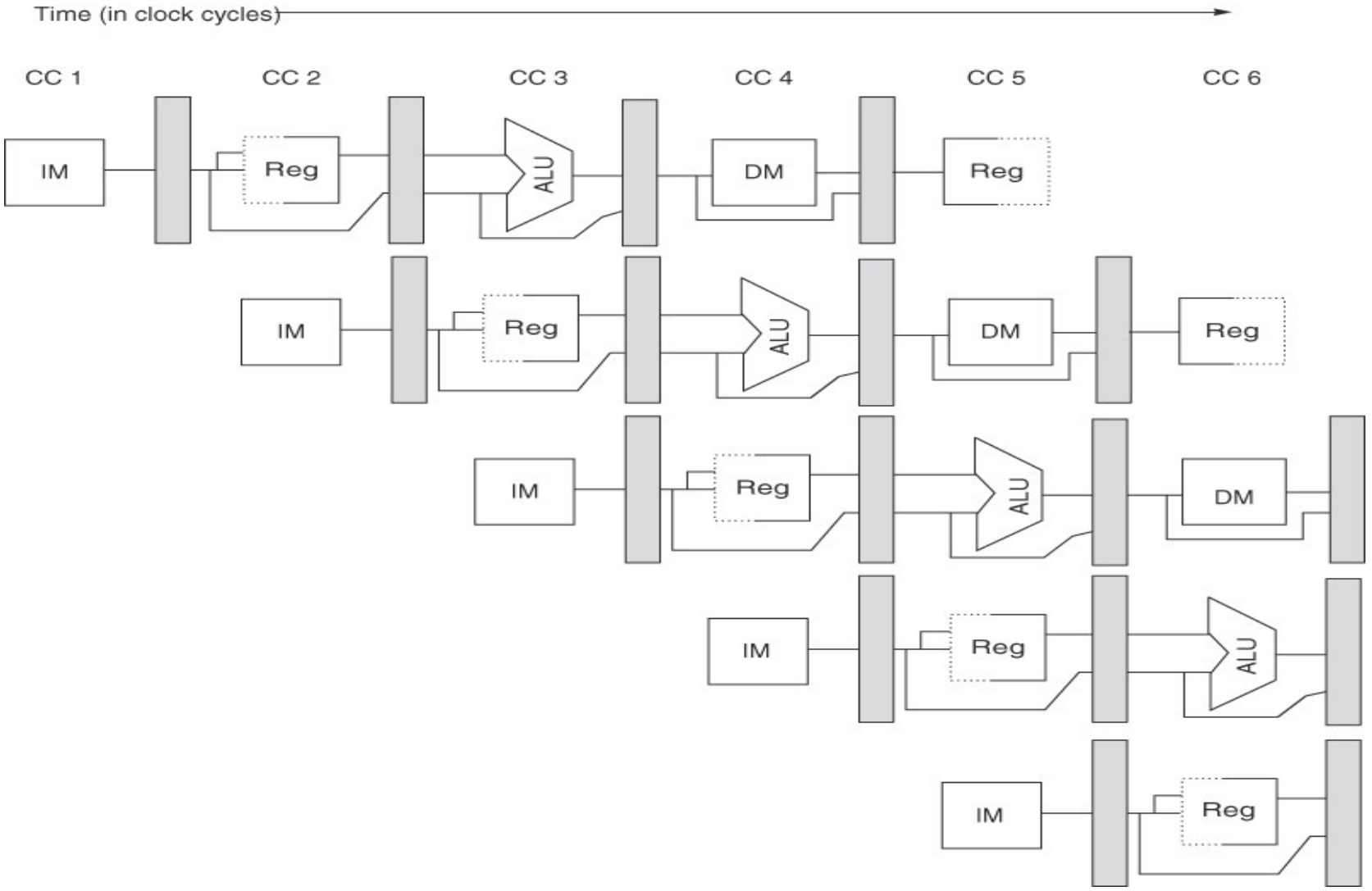
A 5-Stage Pipeline



Source: H&P textbook

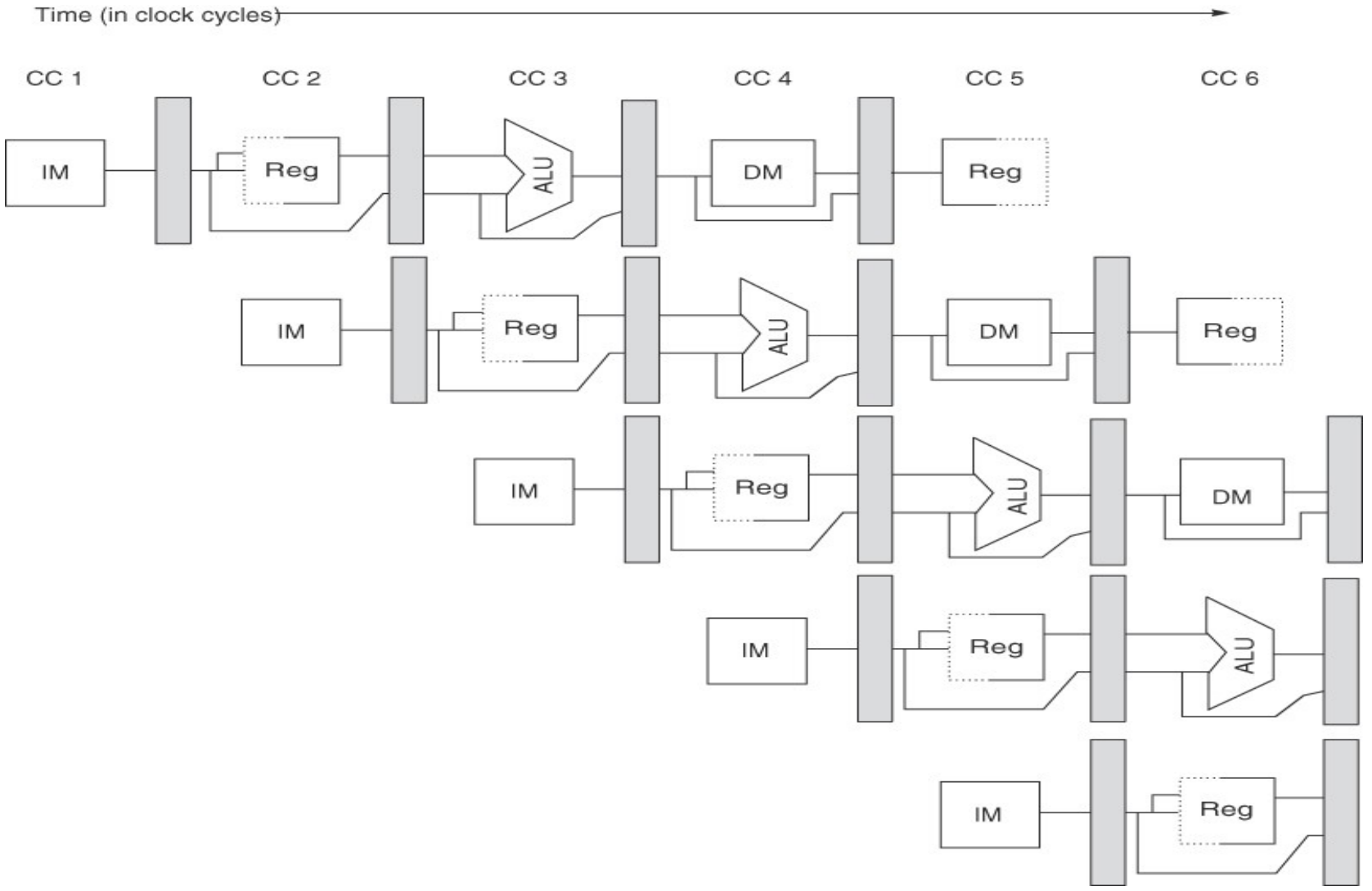
A 5-Stage Pipeline

Use the PC to access the I-cache and increment PC by 4



A 5-Stage Pipeline

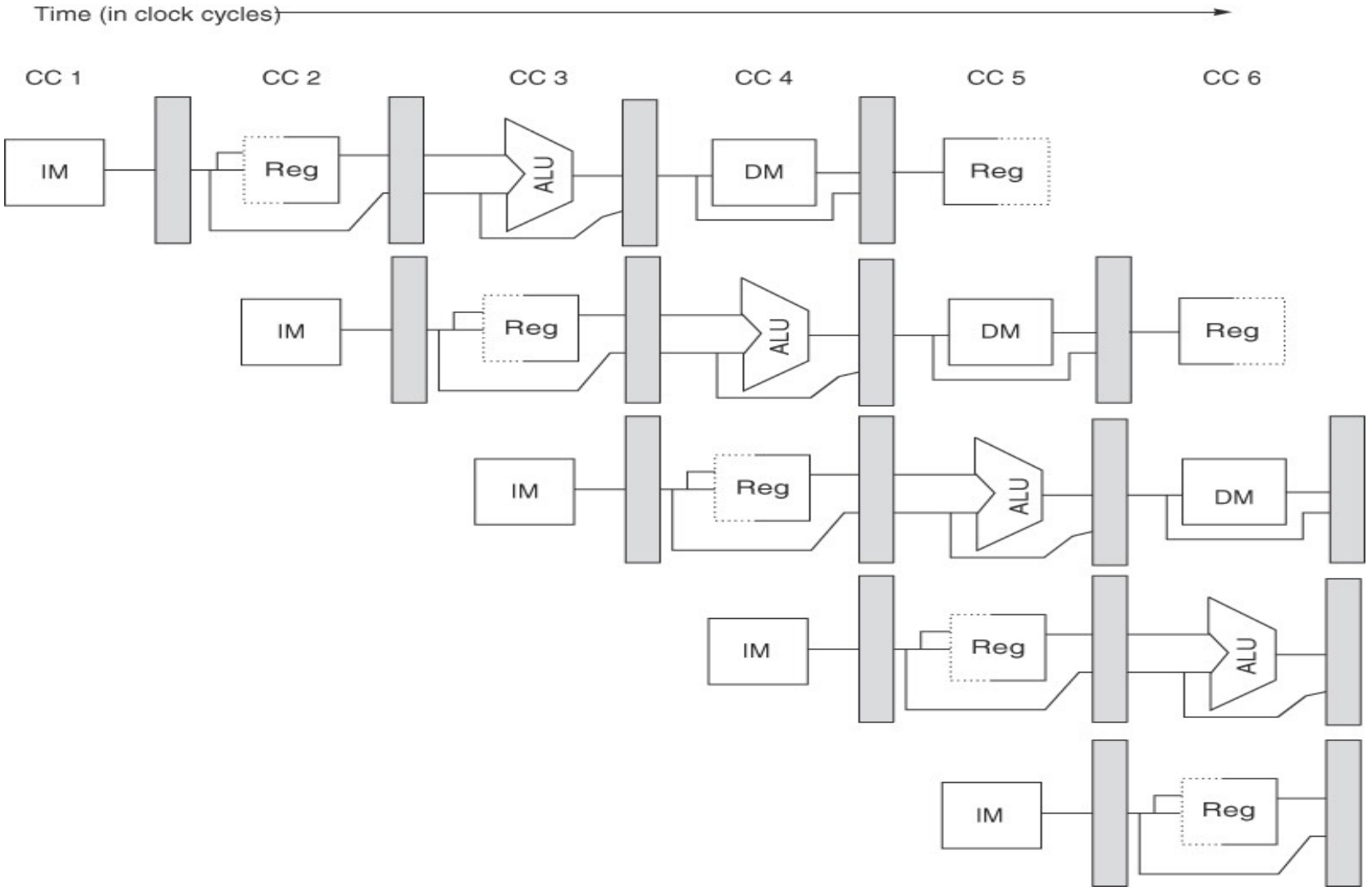
Read registers, compare registers, compute branch target; for now, assume branches take 2 cyc (there is enough work that branches can easily take more)



RISC/CISC Loads/Stores

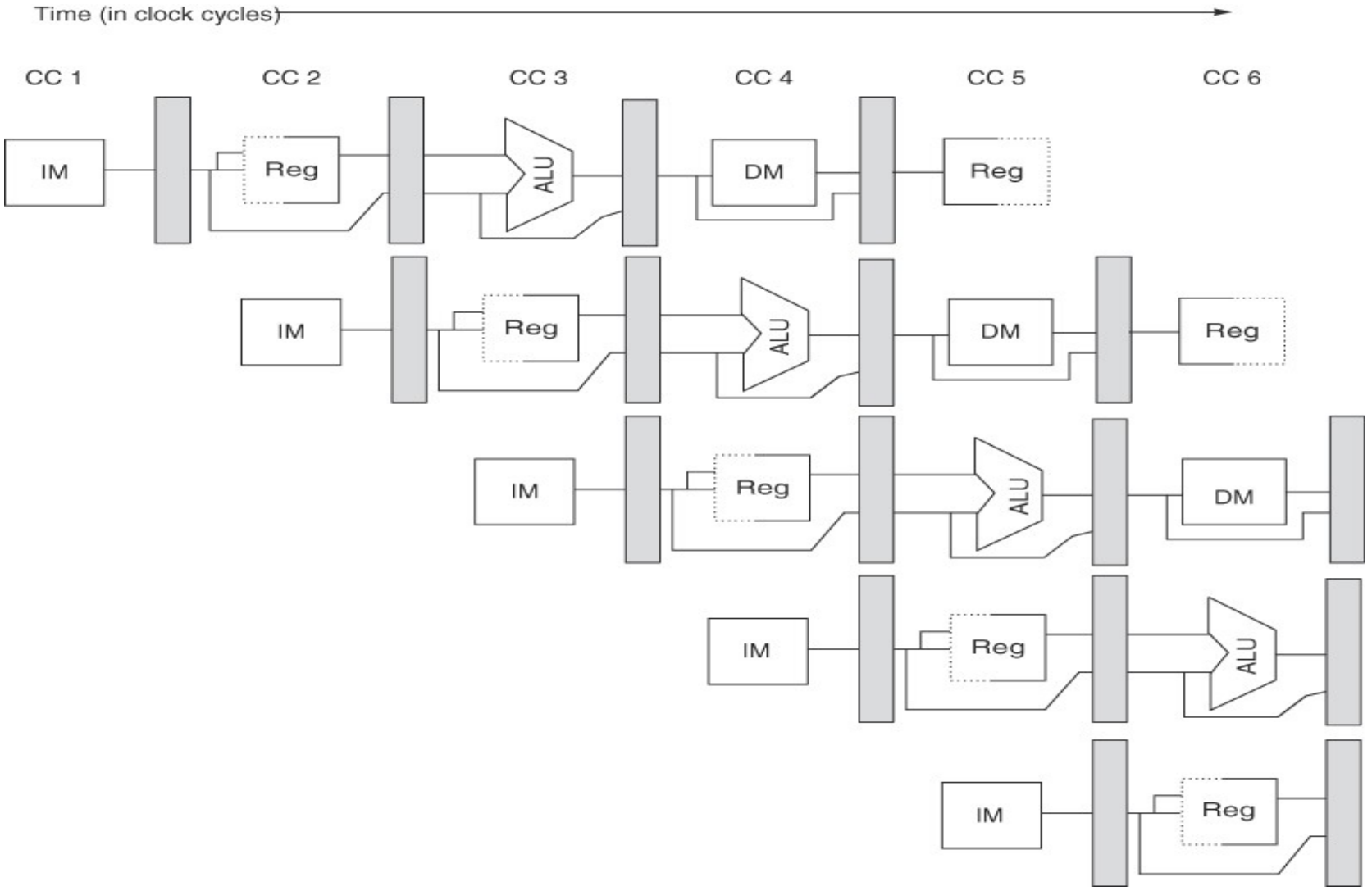
A 5-Stage Pipeline

ALU computation, effective address computation for load/store



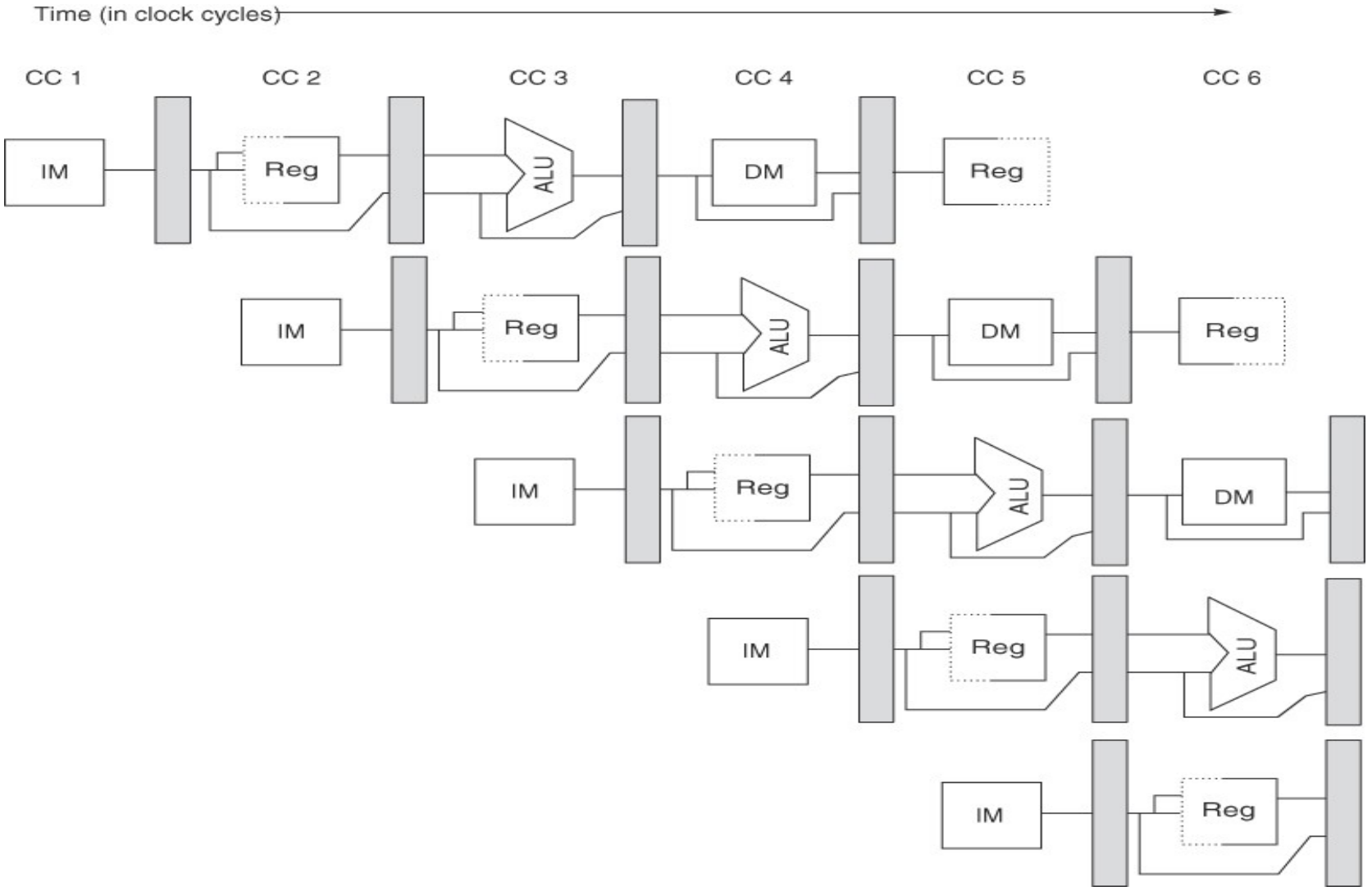
A 5-Stage Pipeline

Memory access to/from data cache, stores finish in 4 cycles



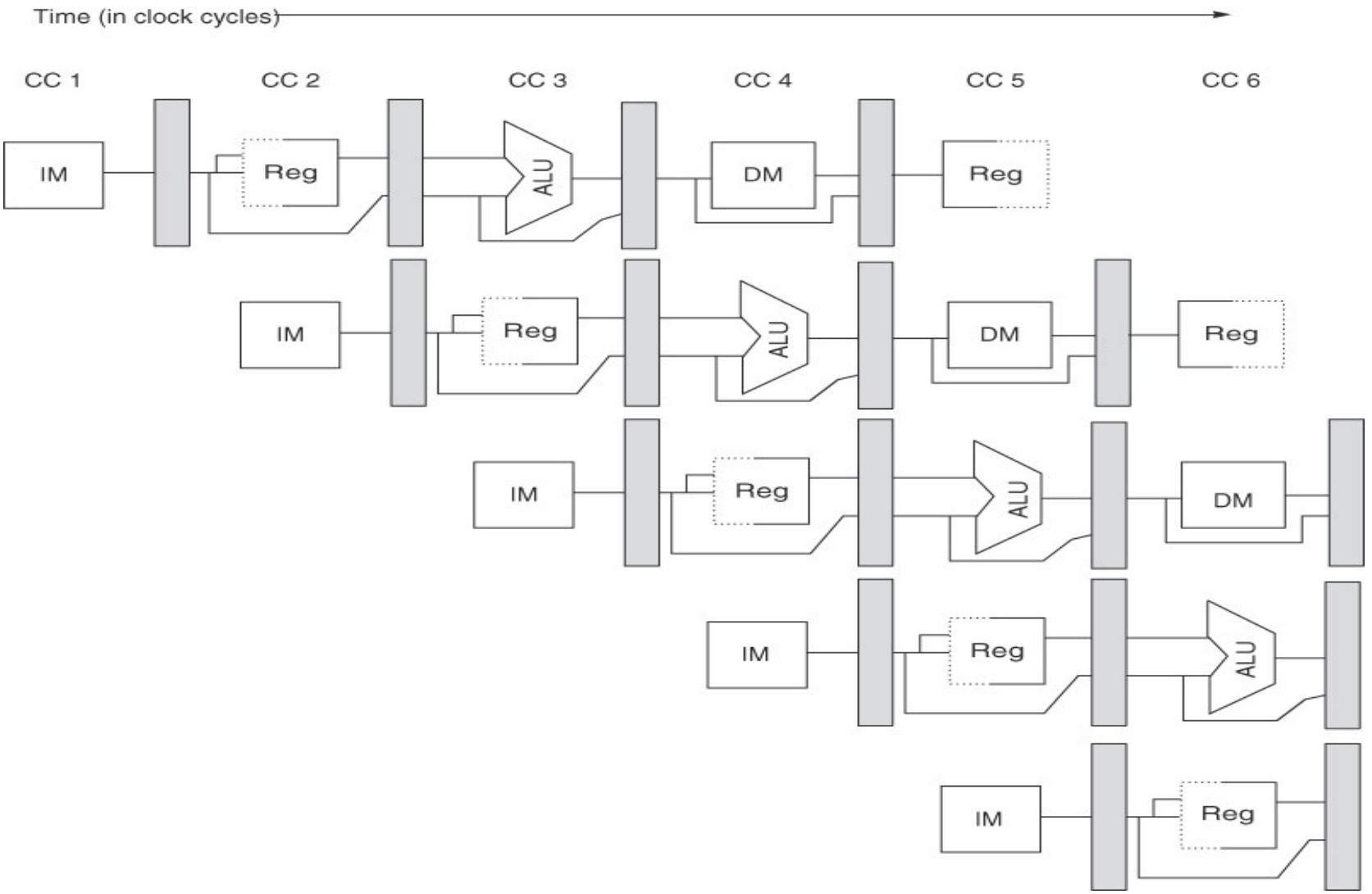
A 5-Stage Pipeline

Write result of ALU computation or load into register file



Pipelining Hazards

A 5-Stage Pipeline



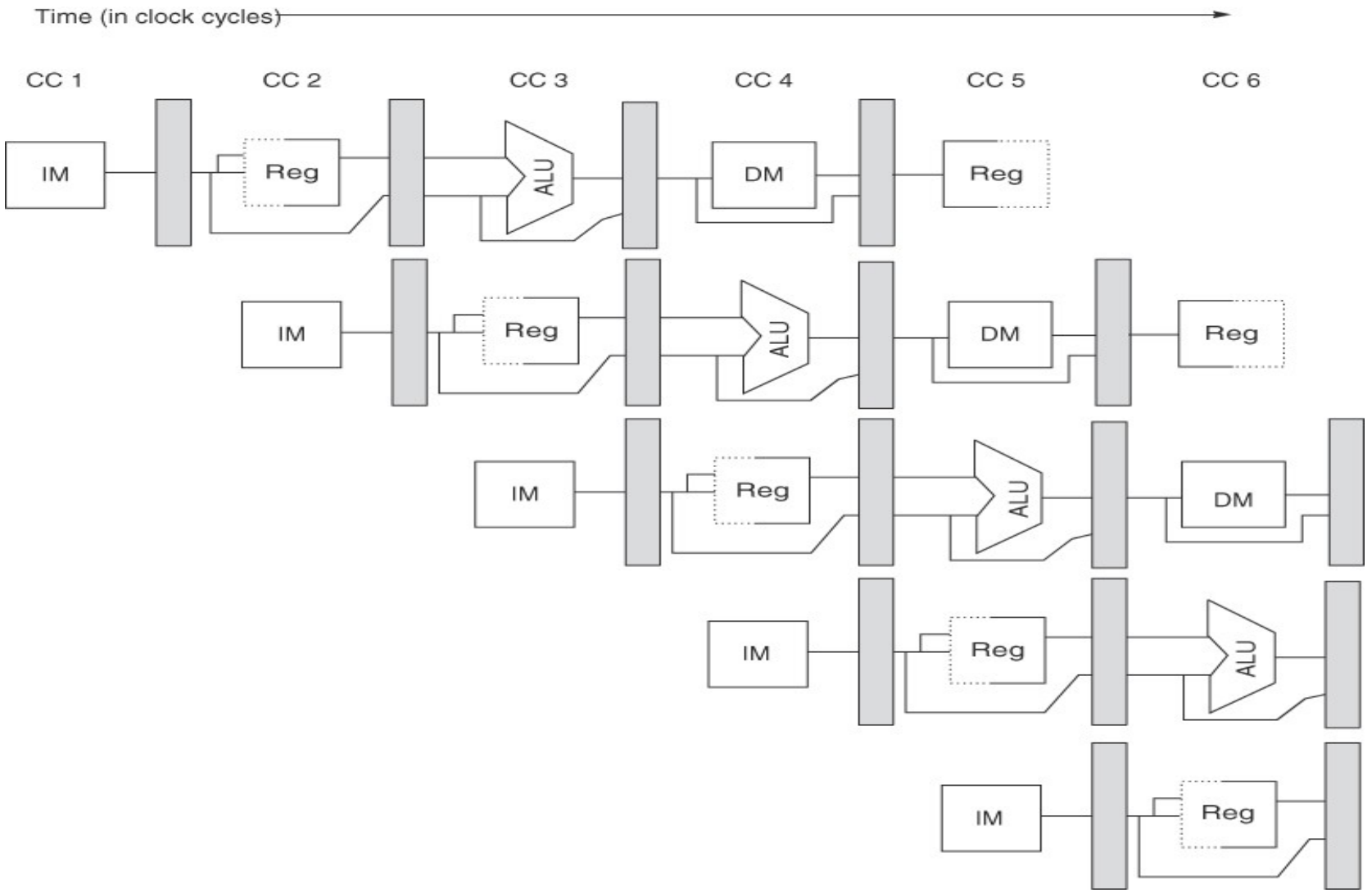
Source: H&P textbook

Hazards

- Structural hazards: different instructions in different stages (or the same stage) conflicting for the same resource
- Data hazards: an instruction cannot continue because it needs a value that has not yet been generated by an earlier instruction
- Control hazard: fetch cannot continue because it does not know the outcome of an earlier branch – special case of a data hazard – separate category because they are treated in different ways

Structural hazards

A 5-Stage Pipeline



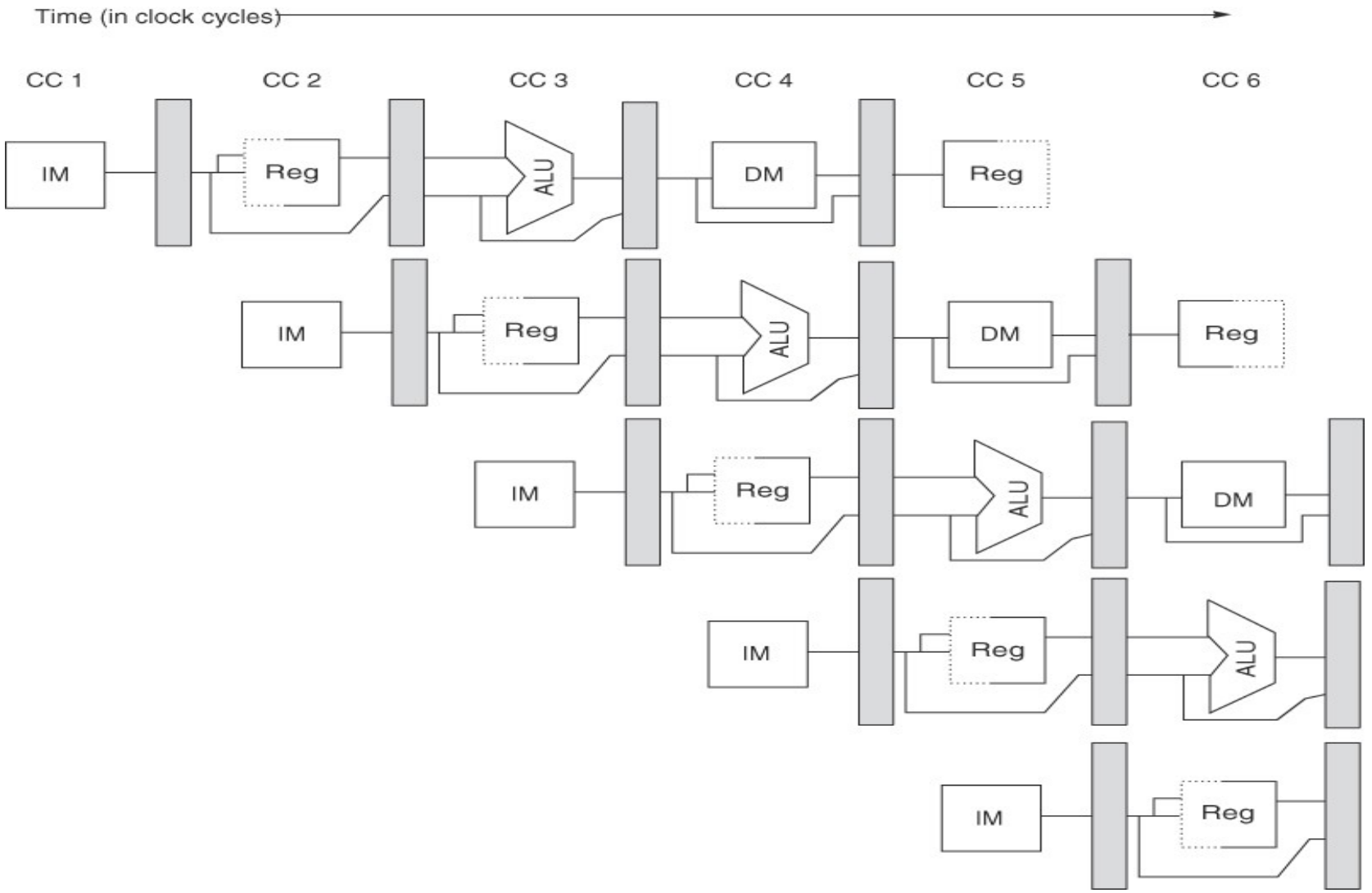
Source: H&P textbook

Structural Hazards

- Example: a unified instruction and data cache → stage 4 (MEM) and stage 1 (IF) can never coincide
- The later instruction and all its successors are delayed until a cycle is found when the resource is free → these are pipeline bubbles
- Structural hazards are easy to eliminate – increase the number of resources (for example, implement a separate instruction and data cache)

Data hazards

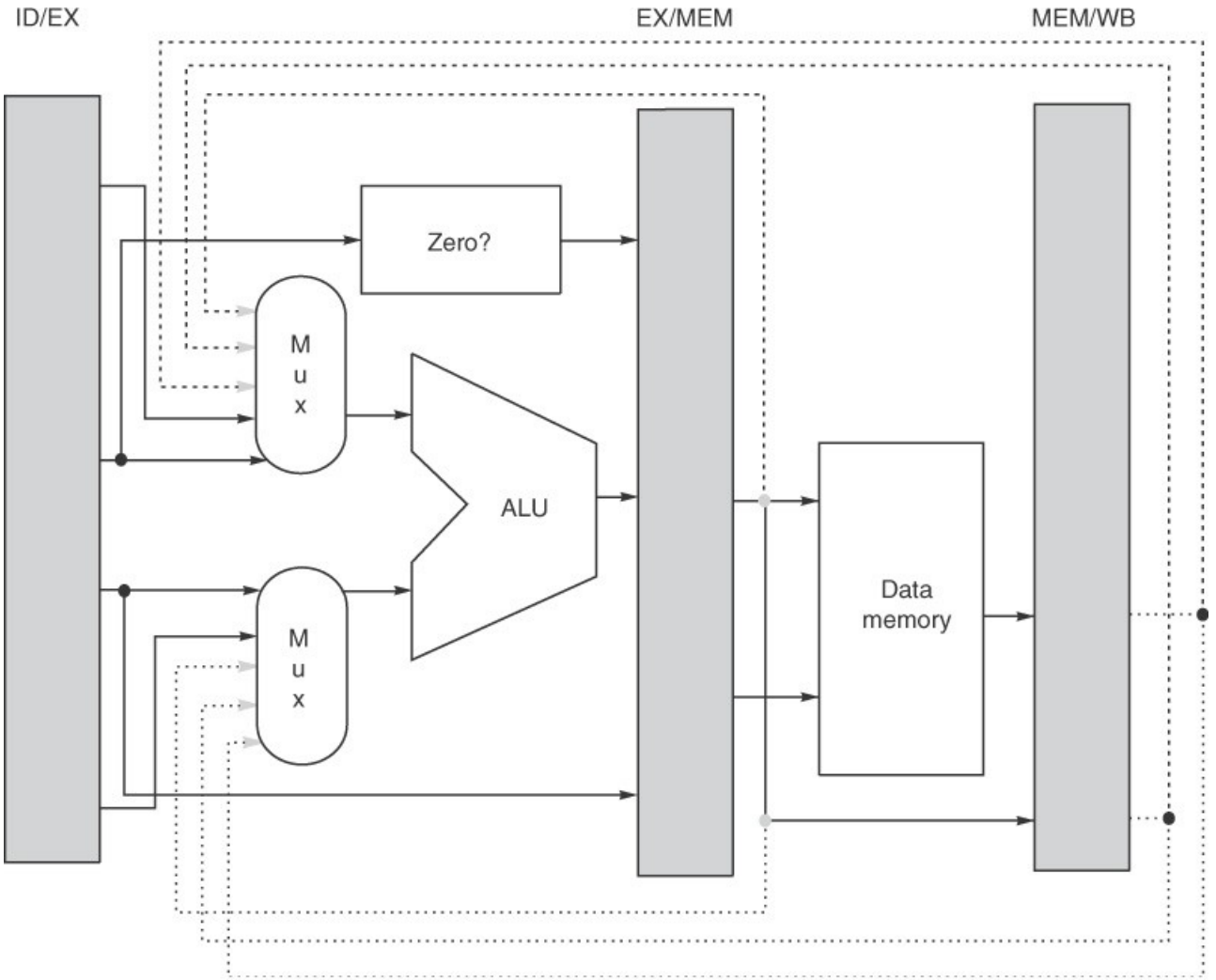
A 5-Stage Pipeline



Source: H&P textbook

Pipeline Implementation

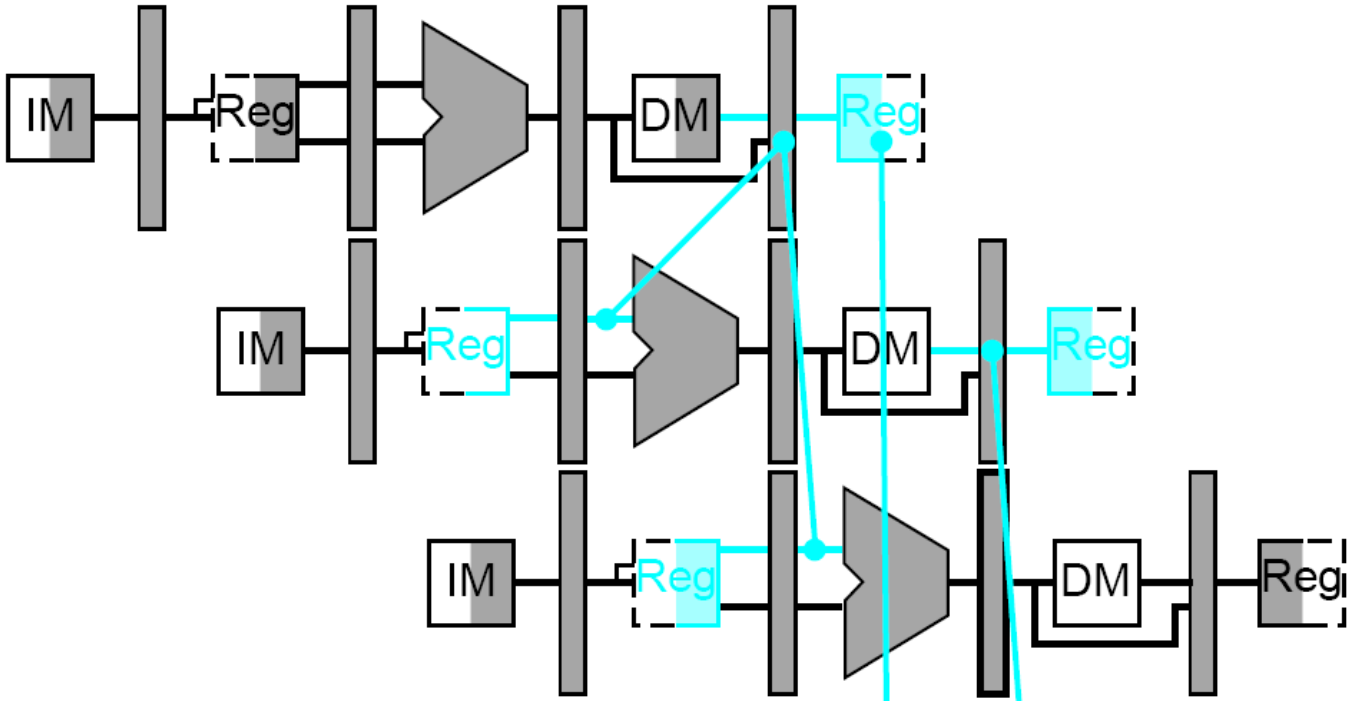
- Signals for the muxes have to be generated – some of this can happen during ID
- Need look-up tables to identify situations that merit bypassing/stalling – the number of inputs to the muxes goes up



Example

add R1, R2, R3

lw R4, 8(R1)

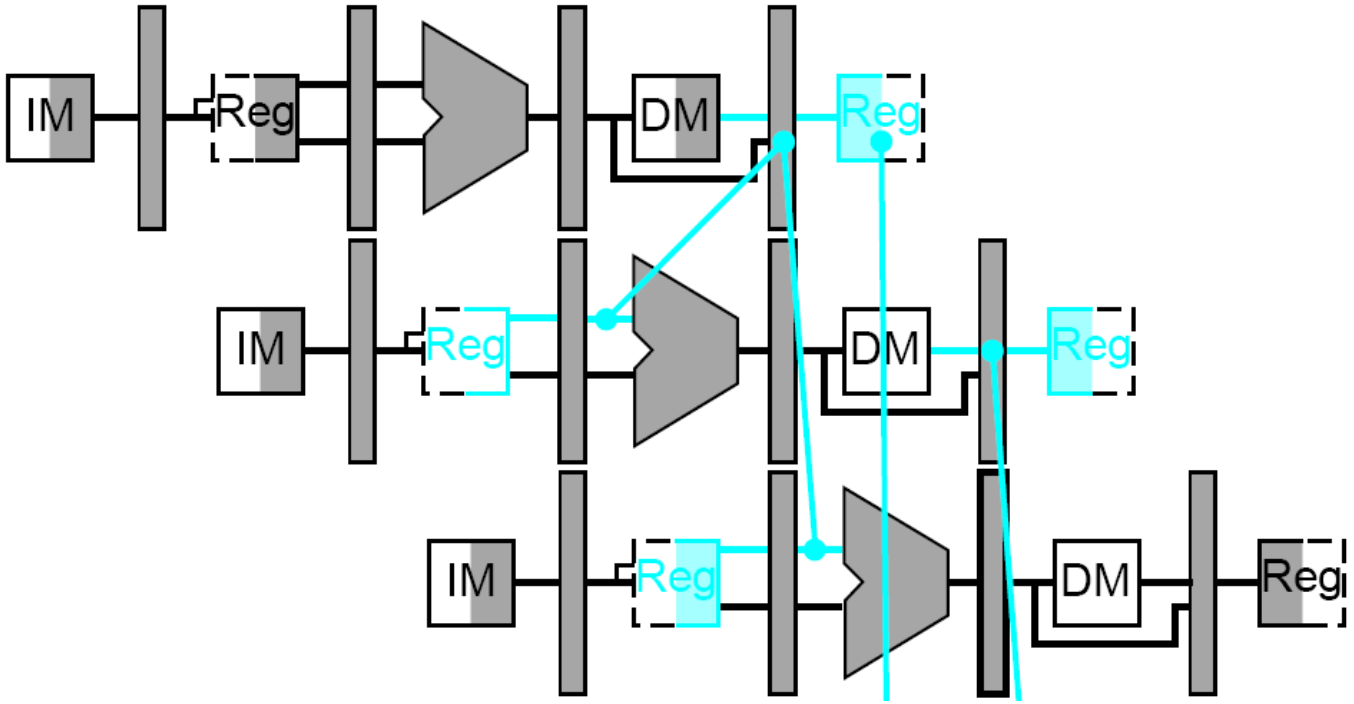


Source: H&P textbook

Example

lw R1, 8(R2)

lw R4, 8(R1)

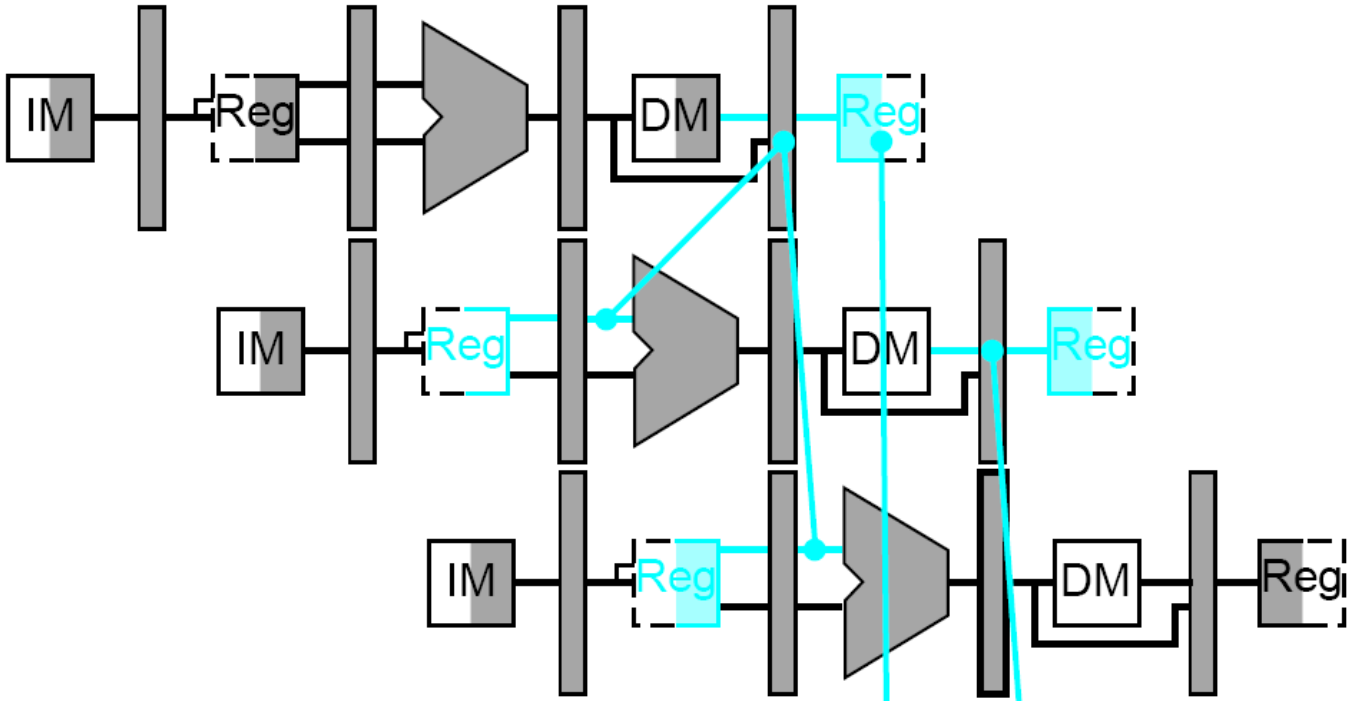


Source: H&P textbook

Example

lw R1, 8(R2)

sw R1, 8(R3)



Source: H&P textbook

Summary

- For the 5-stage pipeline, bypassing can eliminate delays between the following example pairs of instructions:

```
add/sub          R1, R2, R3
add/sub/lw/sw    R4, R1, R5

lw       R1, 8(R2)
sw       R1, 4(R3)
```

- The following pairs of instructions will have intermediate stalls:

```
lw               R1, 8(R2)
add/sub/lw       R3, R1, R4      or   sw   R3, 8(R1)

fmul       F1, F2, F3
fadd       F5, F1, F4
```

Control hazards

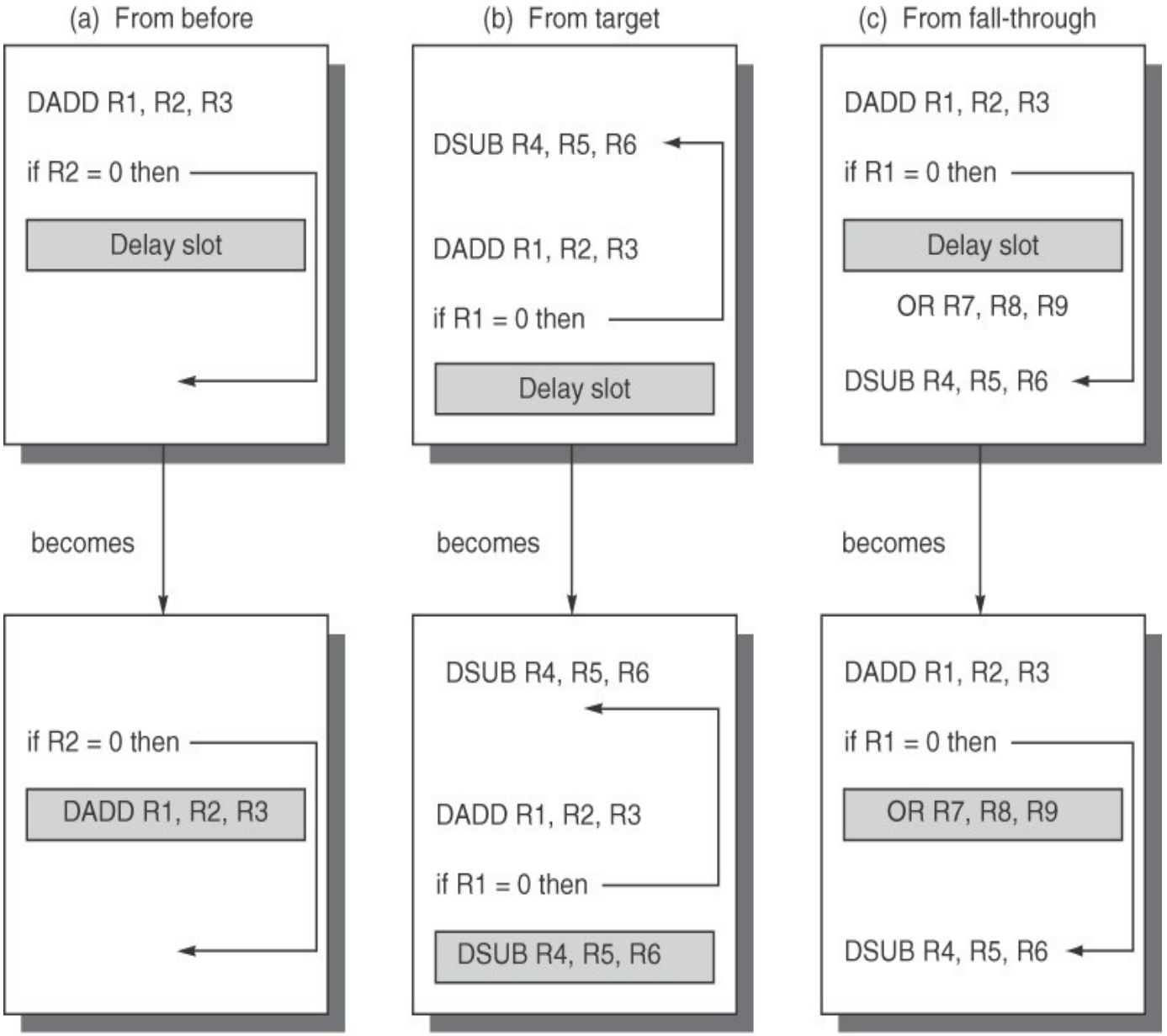
Hazards

- Structural hazards
- Data hazards
- Control hazards

Control Hazards

- Simple techniques to handle control hazard stalls:
  - for every branch, introduce a stall cycle (note: every 6th instruction is a branch on average!)
  - assume the branch is not taken and start fetching the next instruction – if the branch is taken, need hardware to cancel the effect of the wrong-path instructions
  - predict the next PC and fetch that instr – if the prediction is wrong, cancel the effect of the wrong-path instructions
  - fetch the next instruction (branch delay slot) and execute it anyway – if the instruction turns out to be on the correct path, useful work was done – if the instruction turns out to be on the wrong path, hopefully program state is not lost

Branch delay slot



Thank you!