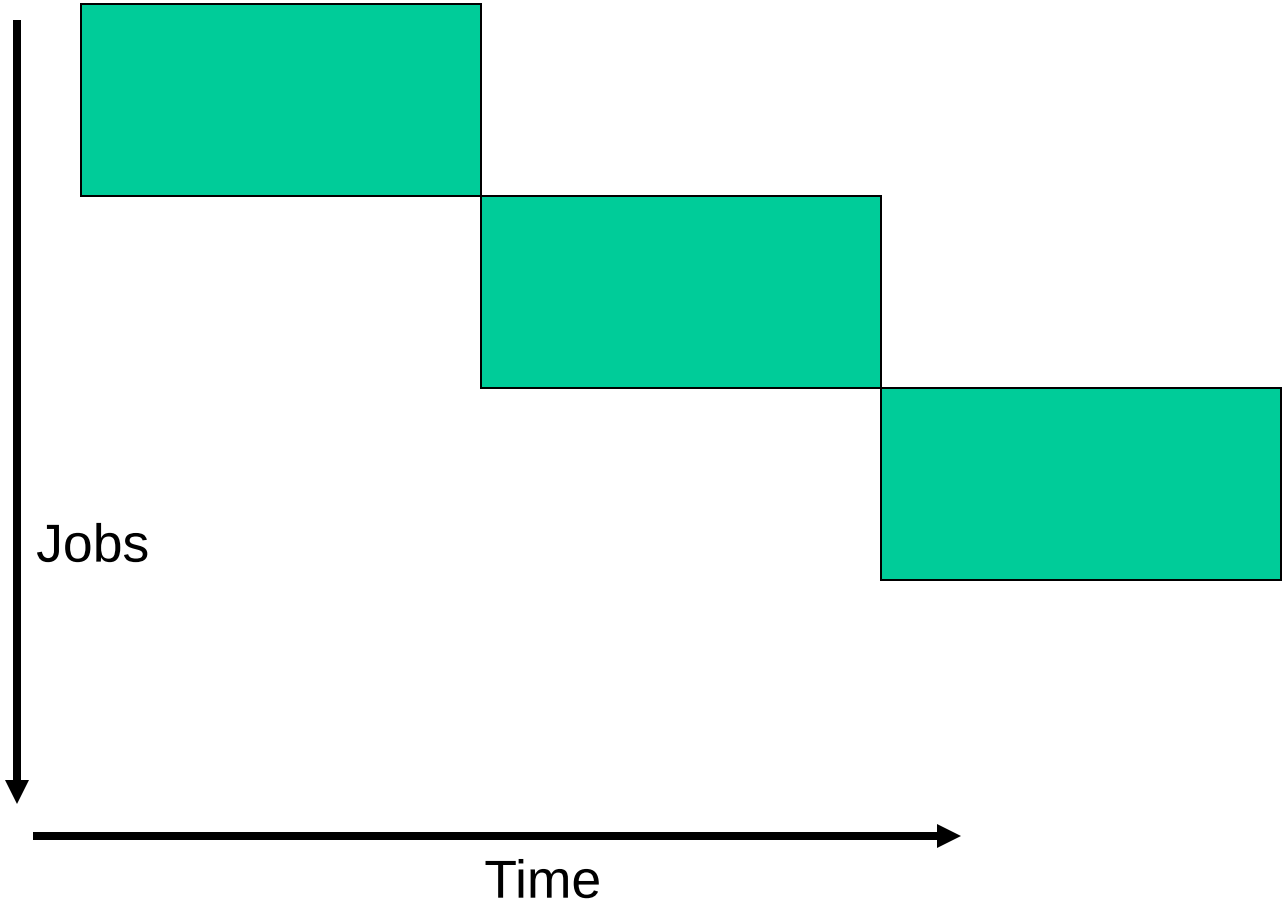
# Lecture: Pipelining Basics

- Topics: Basic pipelining implementation
  - Video 1: What is pipelining?
  - Video 2: Clocks and latches
  - Video 3: An example 5-stage pipeline
  - Video 4: Loads/Stores and RISC/CISC
  - Video 5: Hazards
  - Video 6: Examples of Hazards

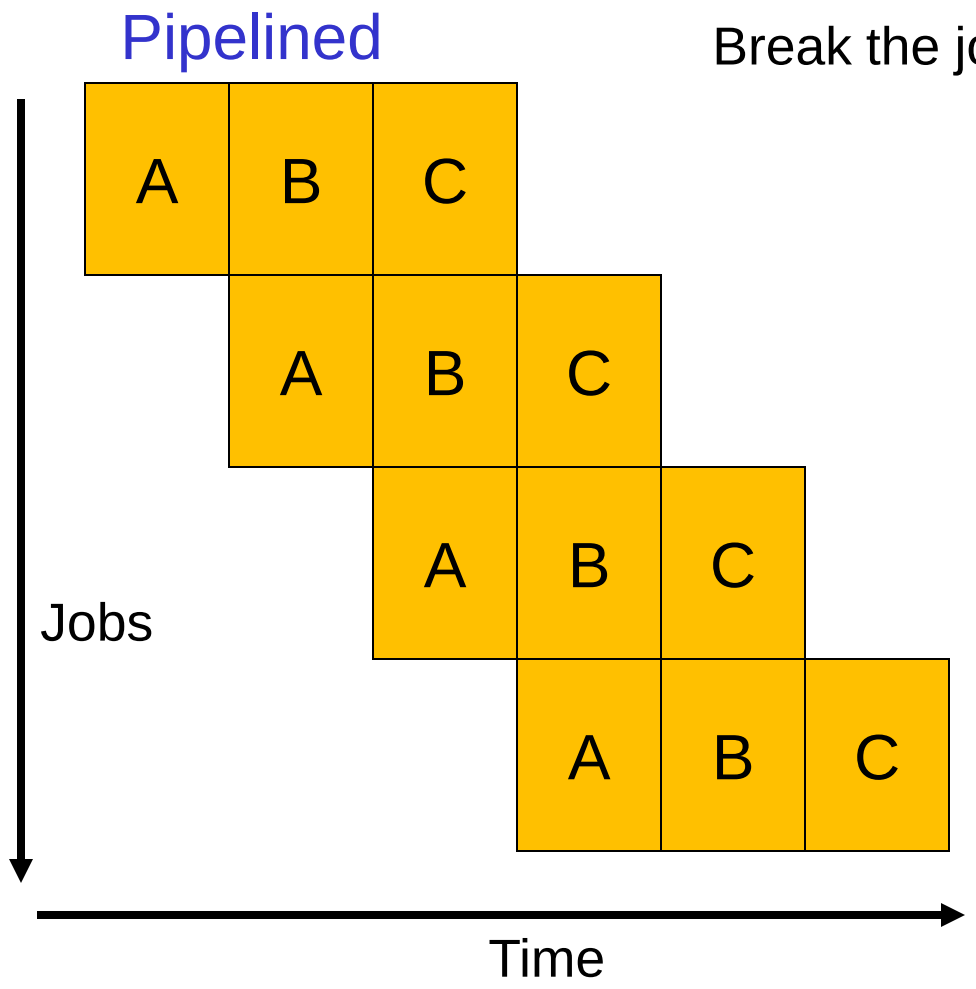
# Building a Car

Unpipelined

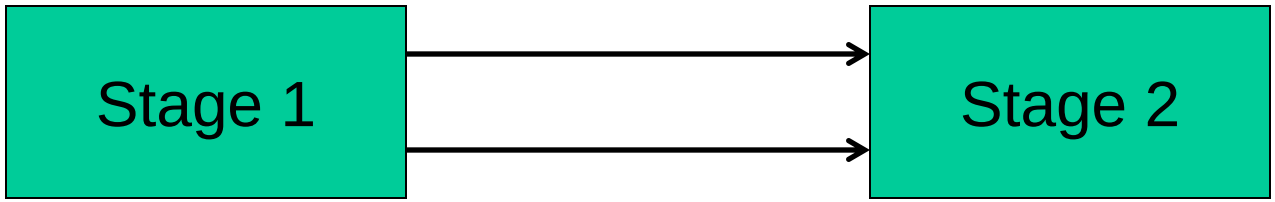
Start and finish a job before moving to the next



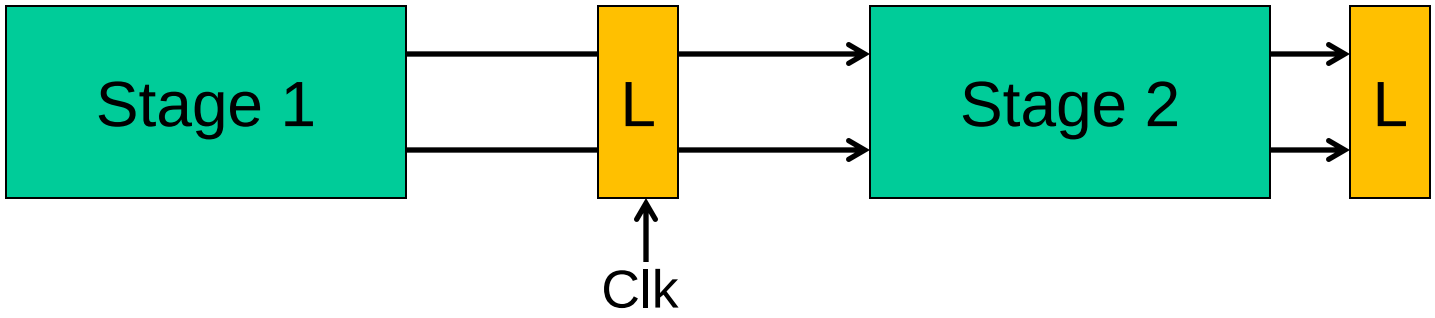
# The Assembly Line



# Clocks and Latches



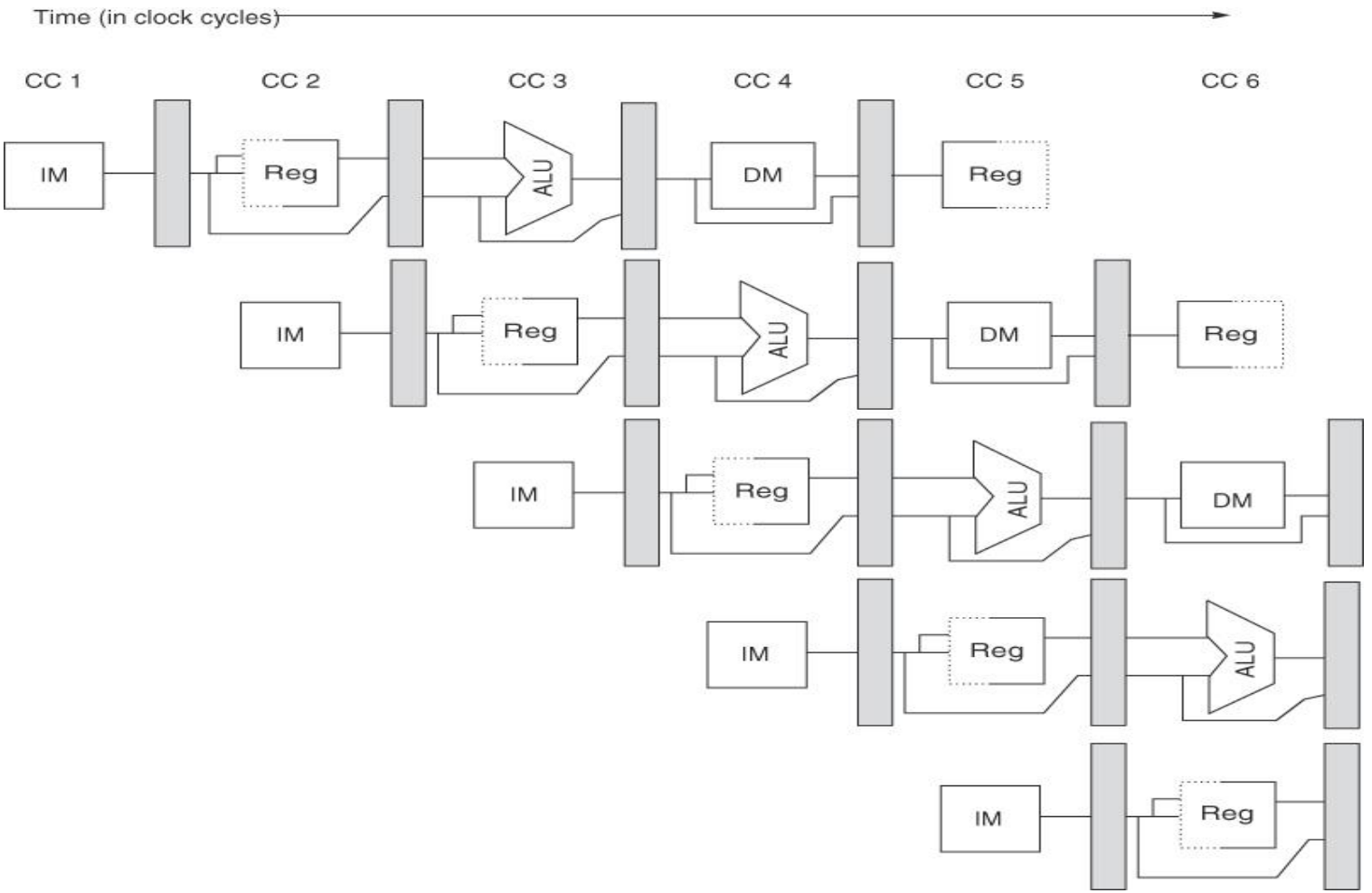
# Clocks and Latches



# 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

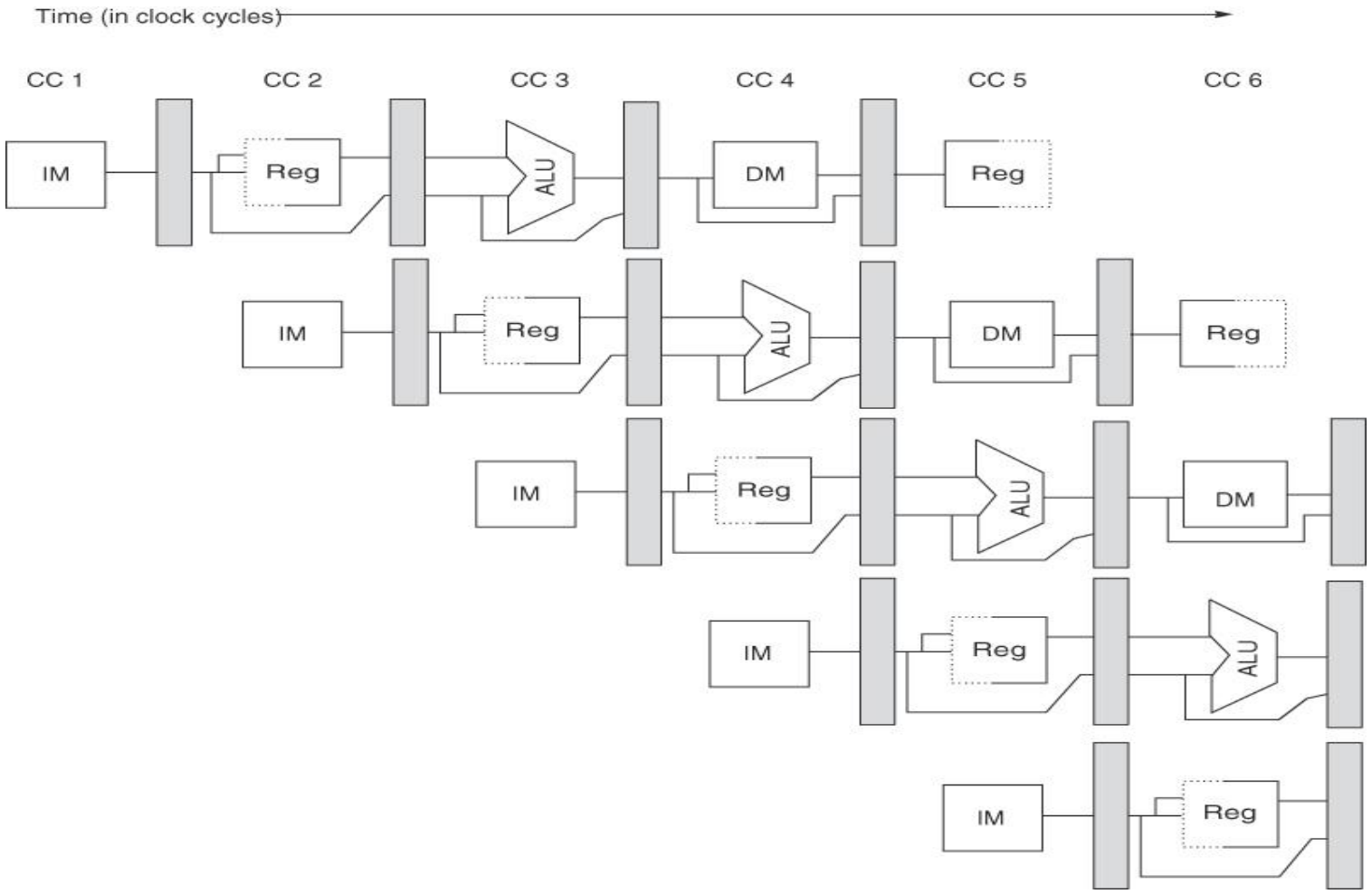
# A 5-Stage Pipeline




Source: H&P textbook

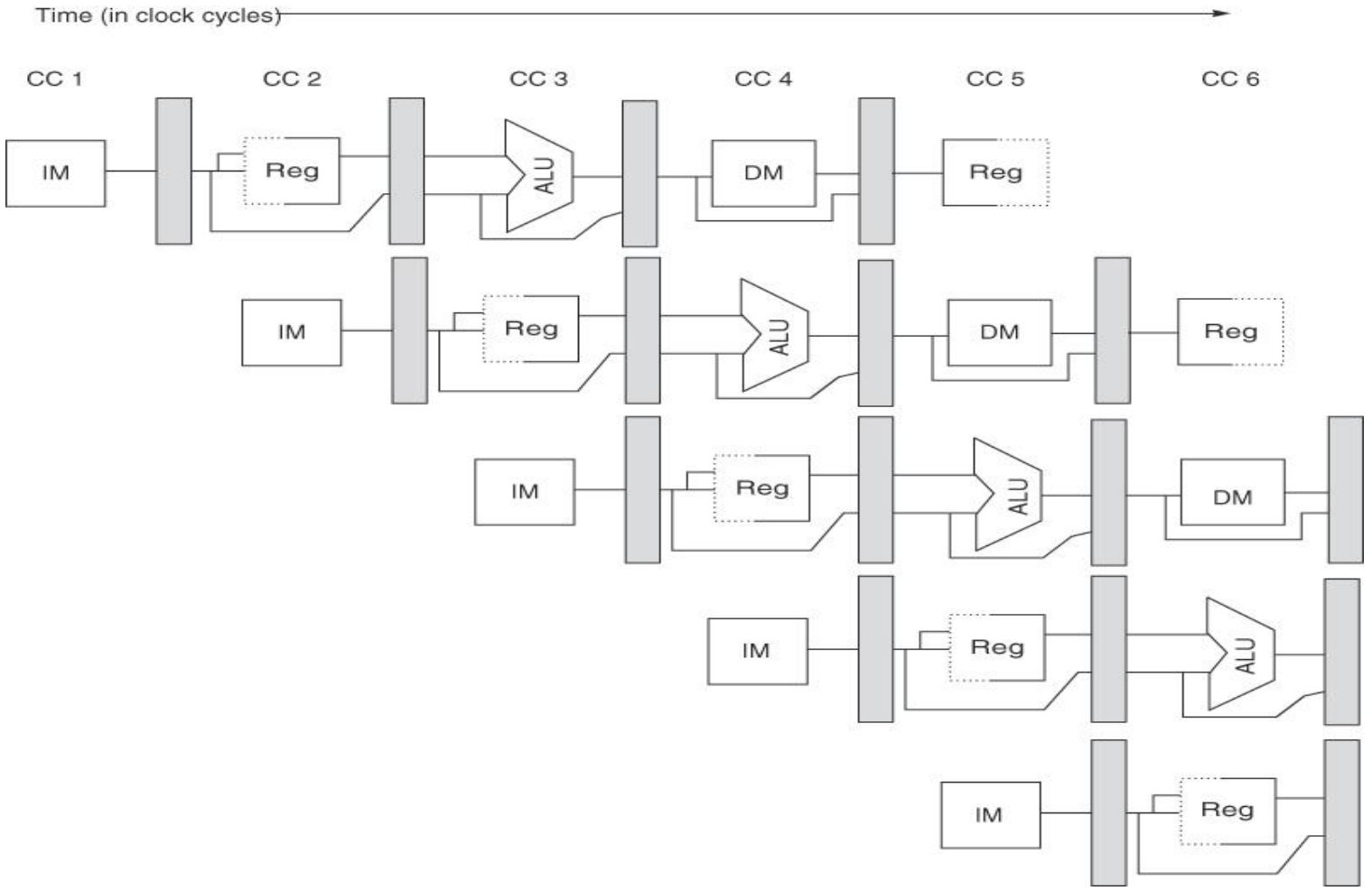
# A 5-Stage Pipeline

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



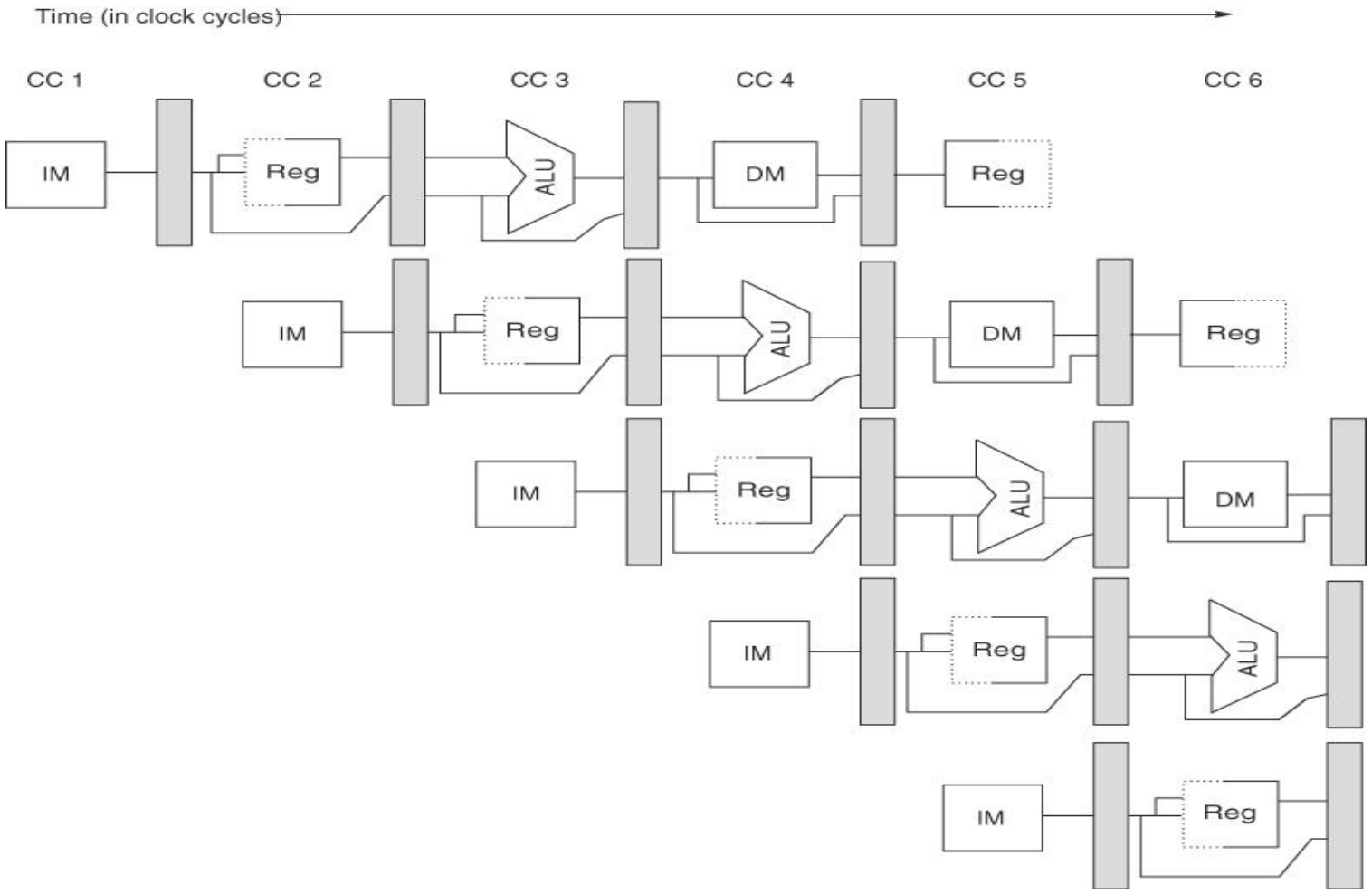
# A 5-Stage Pipeline

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



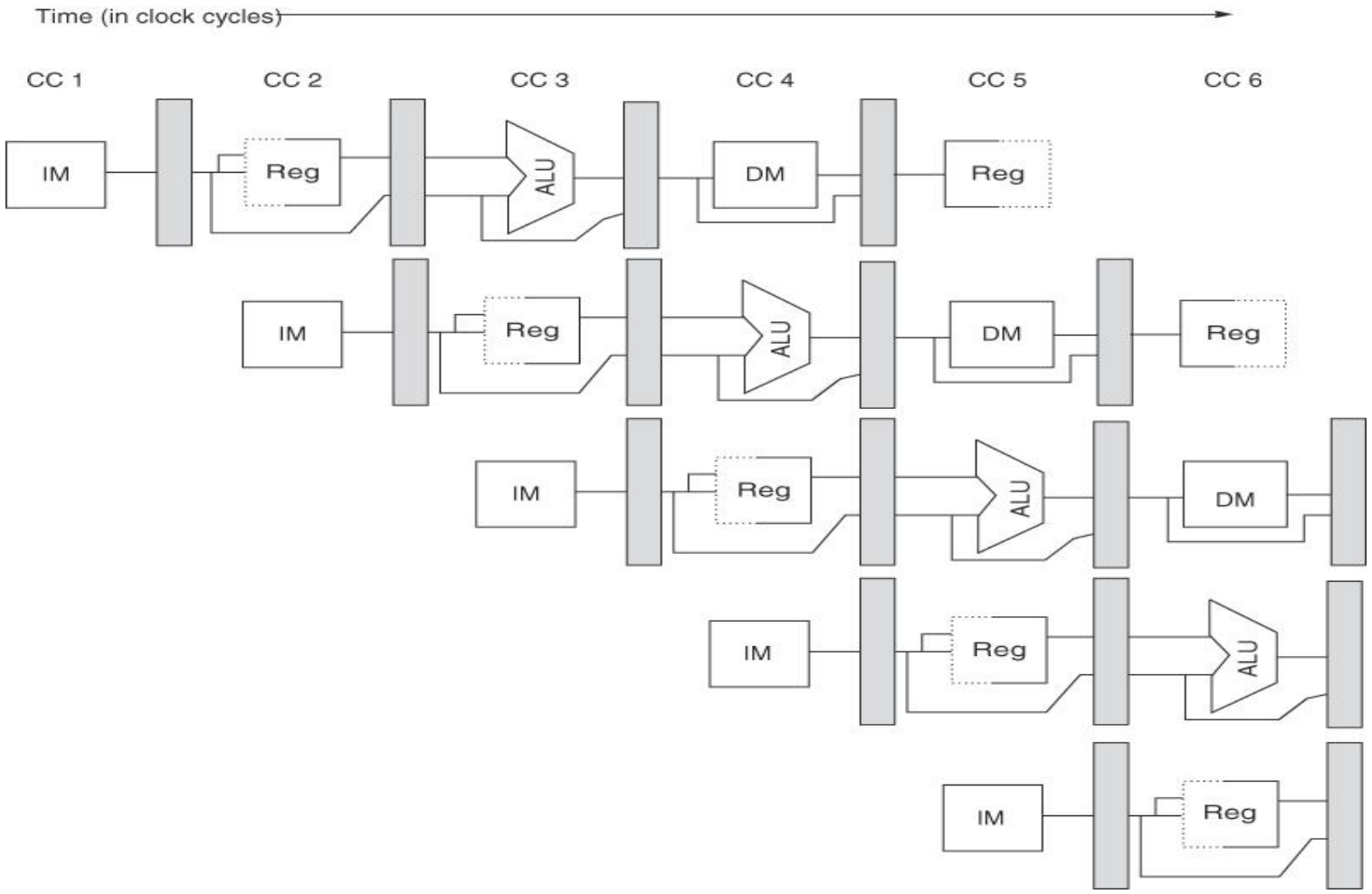
# A 5-Stage Pipeline

ALU computation, effective address computation for load/store



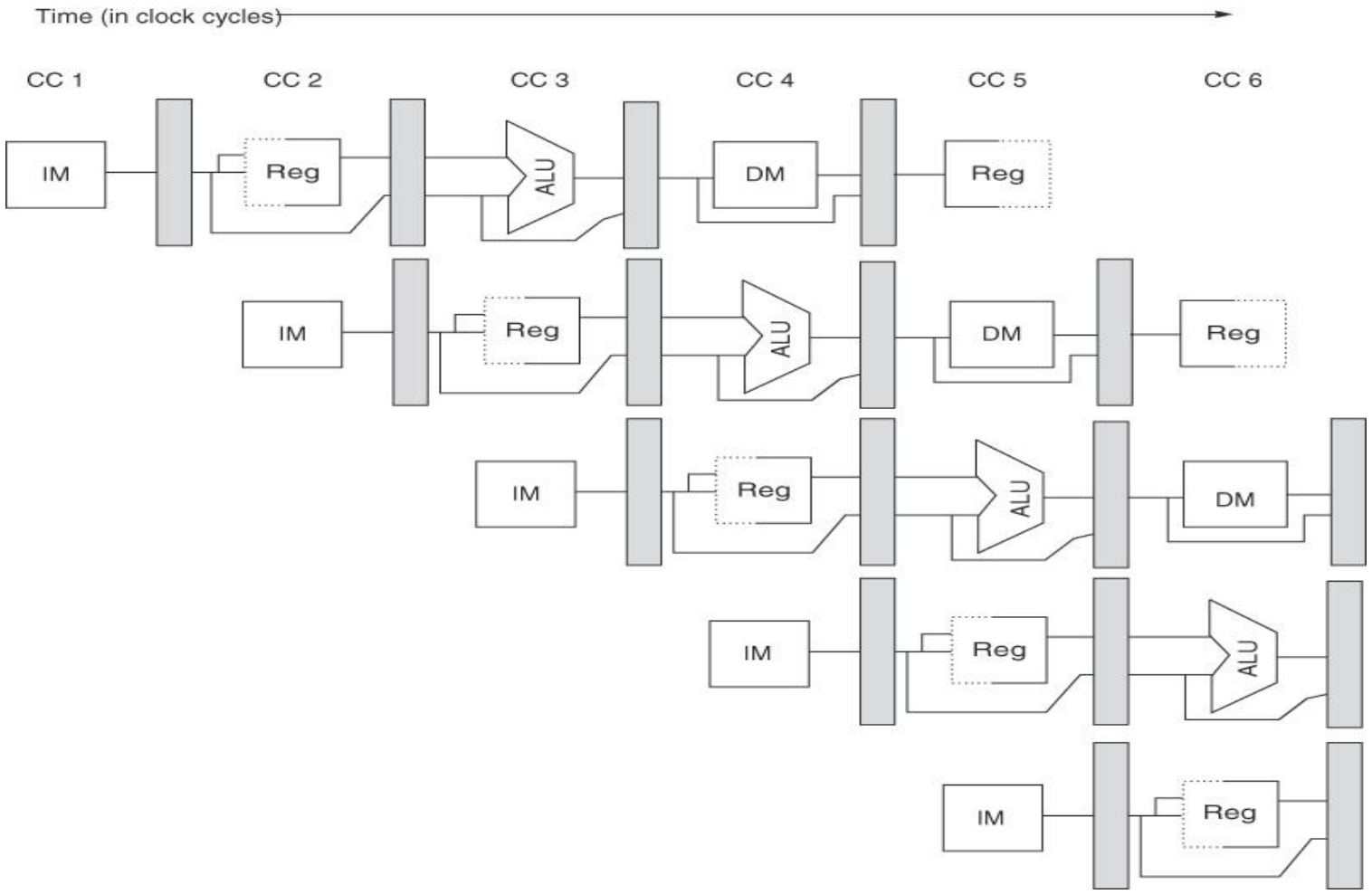
# A 5-Stage Pipeline

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



# RISC/CISC Loads/Stores

# Title

- Bullet