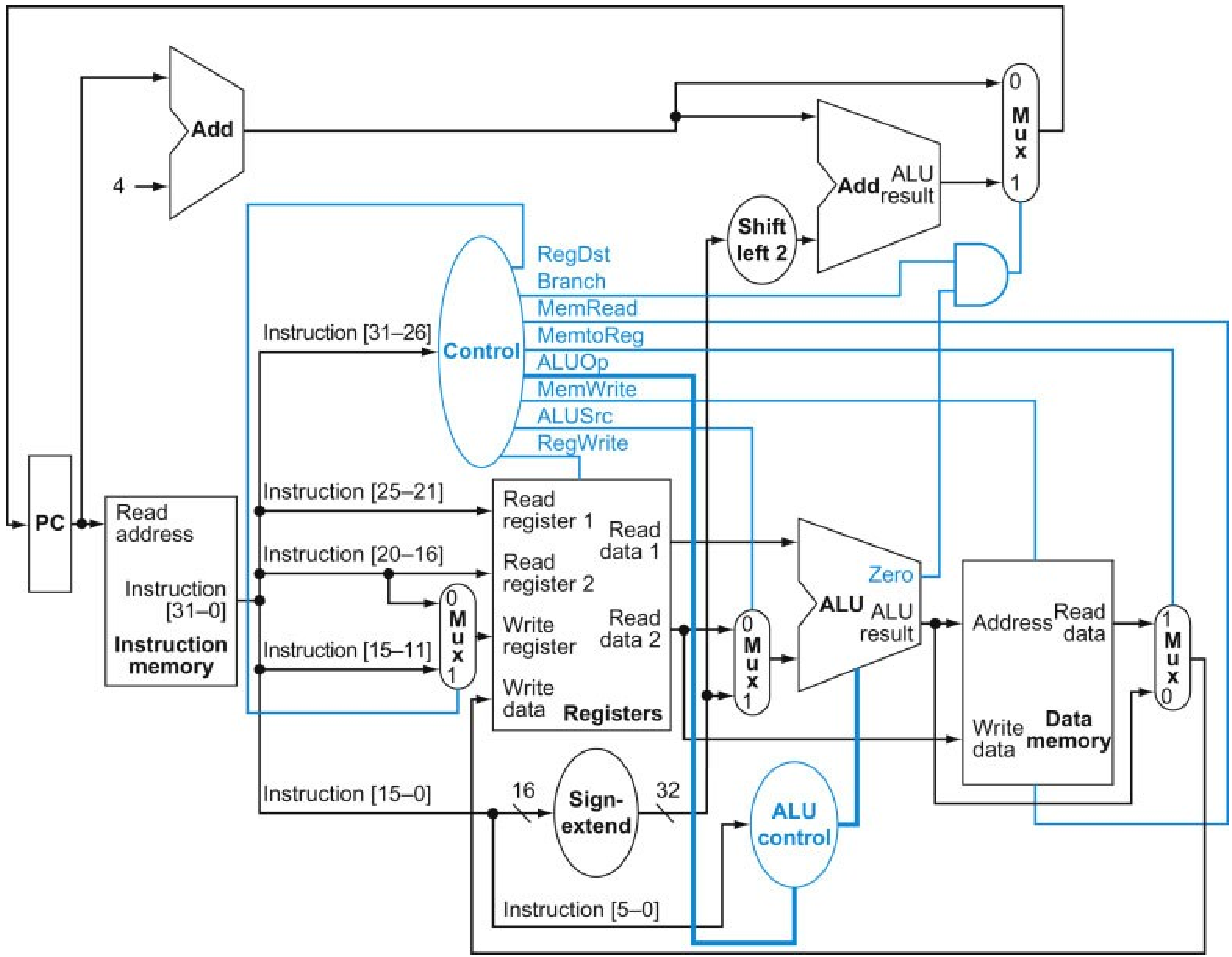
250P: Computer Systems Architecture

Lecture 4: Basics of pipelining

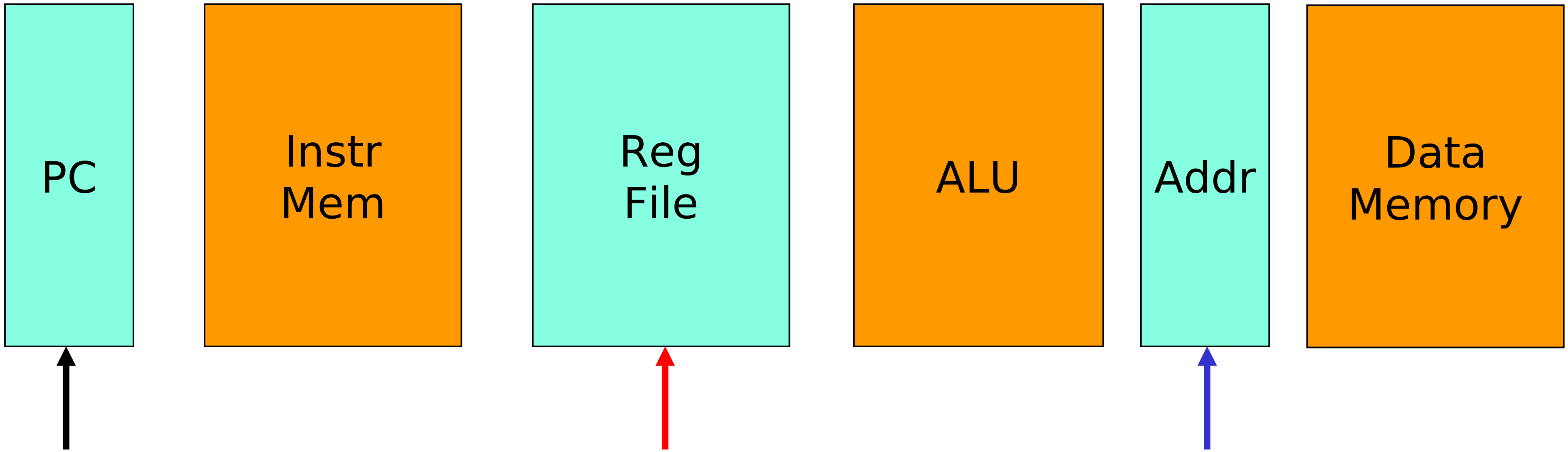
Anton Burtsev
April, 2021

View from 5,000 Feet



Source: H&P textbook

Latches and Clocks in a Single-Cycle Design



The entire instruction executes in a single cycle

Green blocks are latches

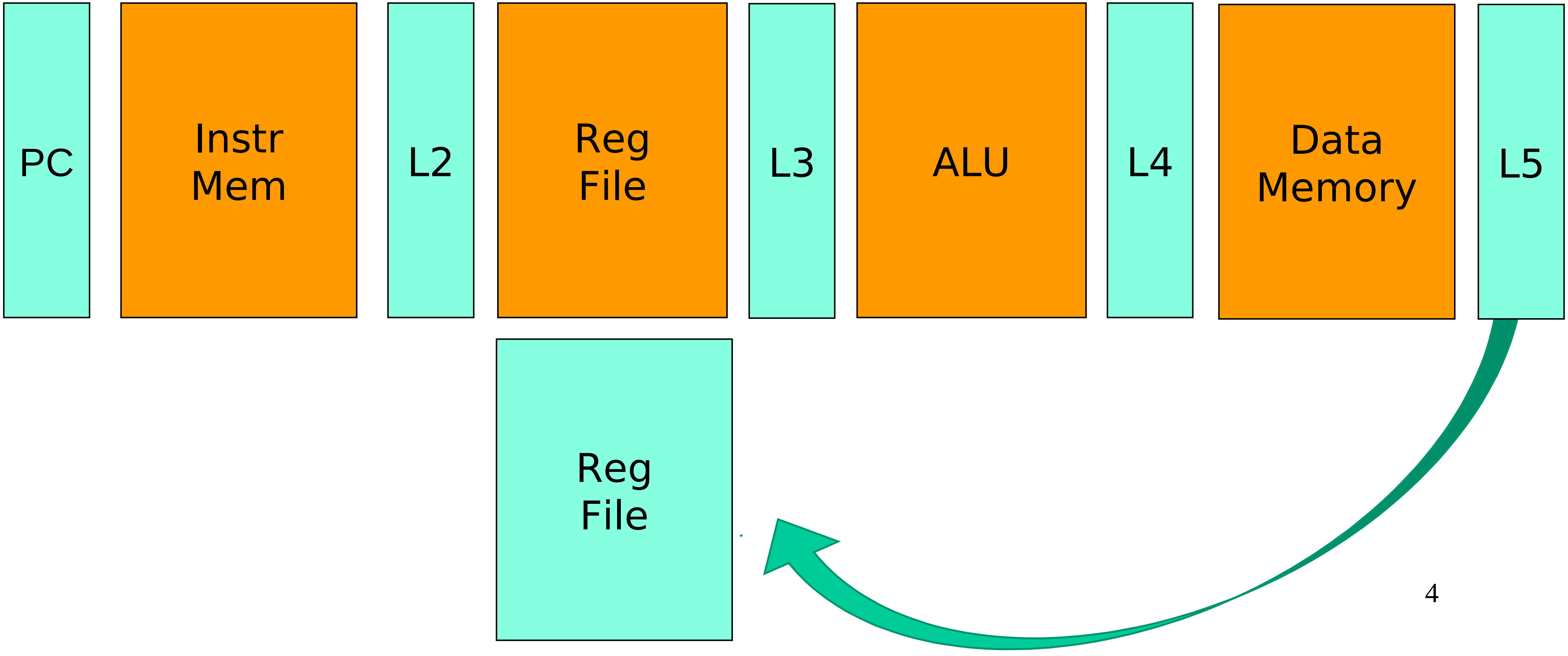
At the rising edge, a new PC is recorded

At the rising edge, the result of the previous cycle is recorded

At the falling edge, the address of LW/SW is recorded so we can access the data memory in the 2nd half of the cycle

Multi-Stage Circuit

- Instead of executing the entire instruction in a single cycle (a single stage), let's break up the execution into multiple stages, each separated by a latch



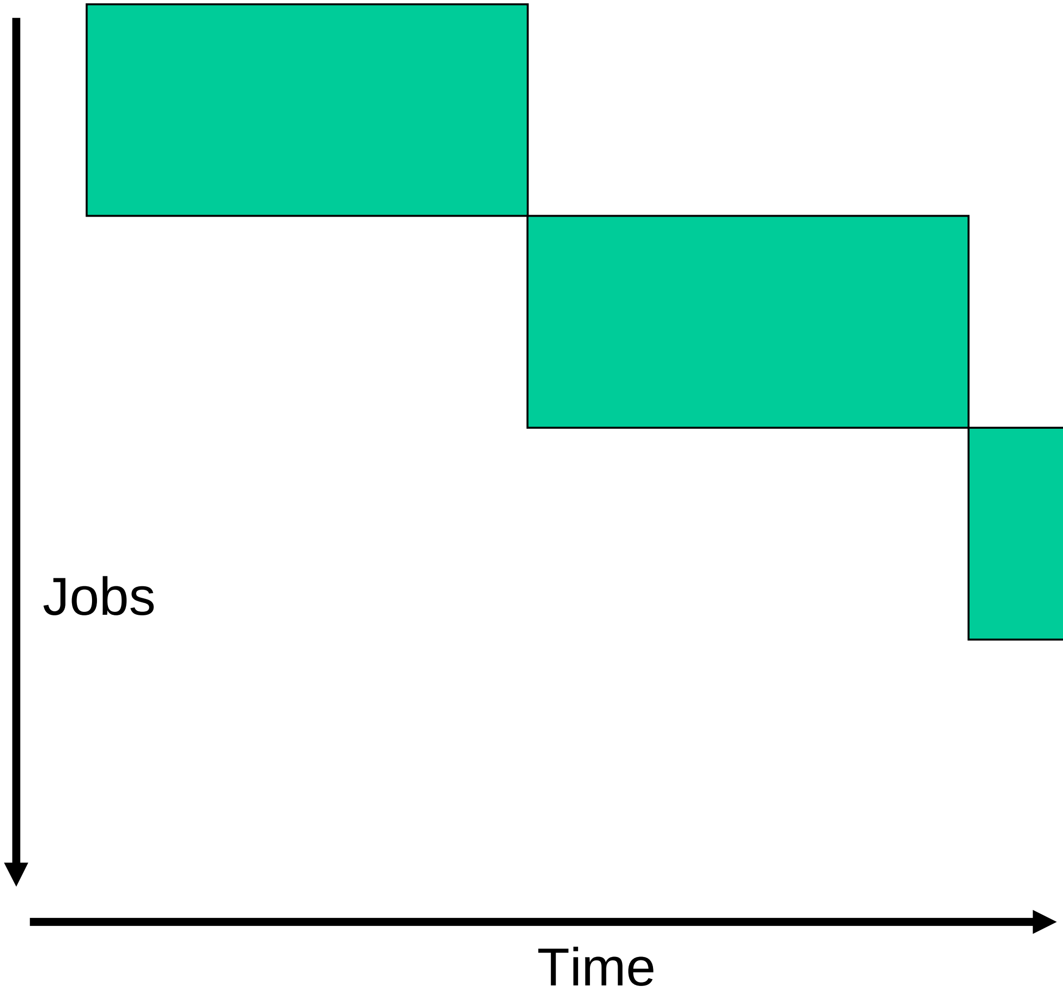
Building a Car

Time

Building a Car

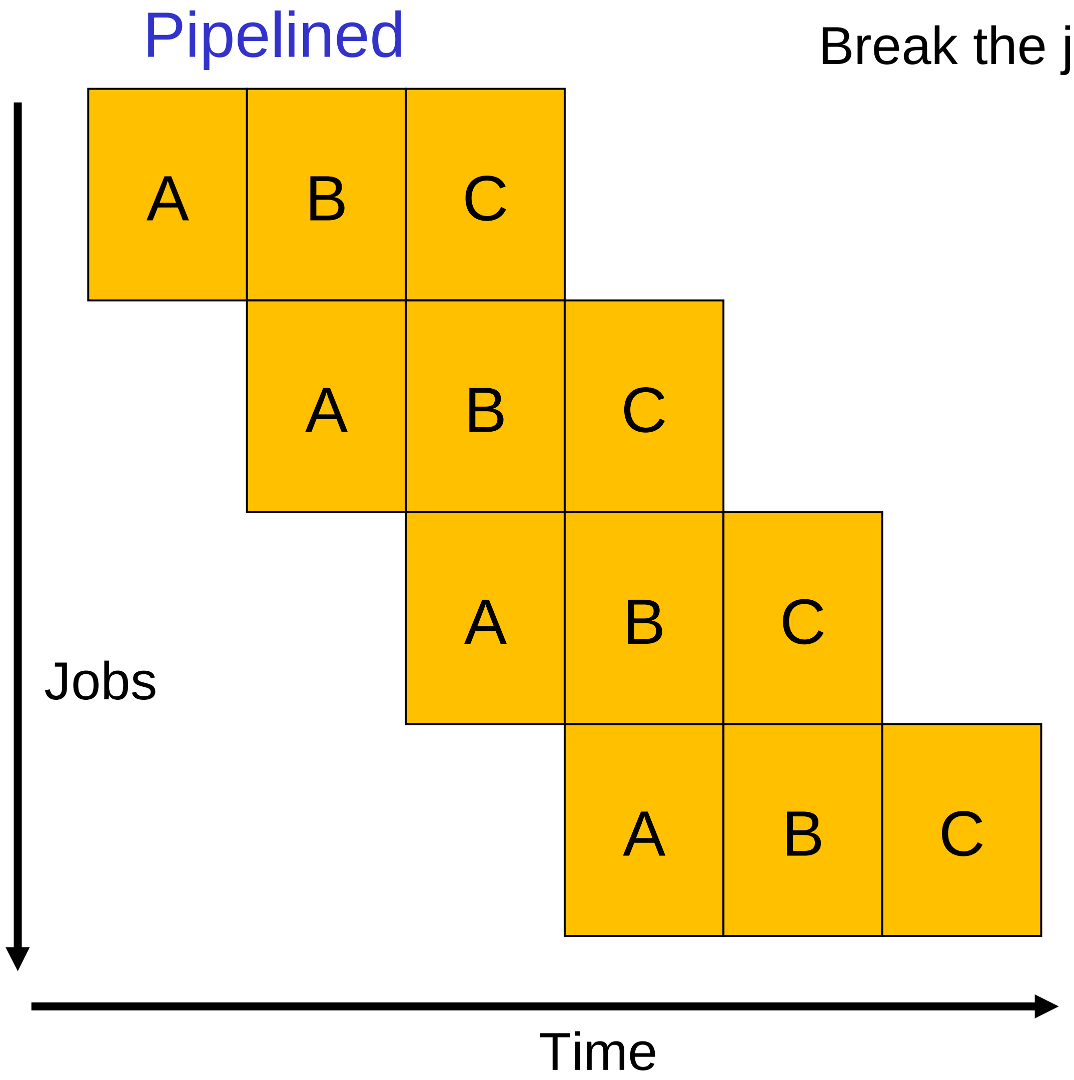
Unpipelined

Start and finish a job before moving to the next



The Assembly Line

Break the job into smaller stages



Performance Improvements?

Does it take longer to finish each individual job?

Does it take shorter to finish a series of jobs?

What assumptions were made while answering these questions?

Is a 10-stage pipeline better than a 5-stage pipeline?

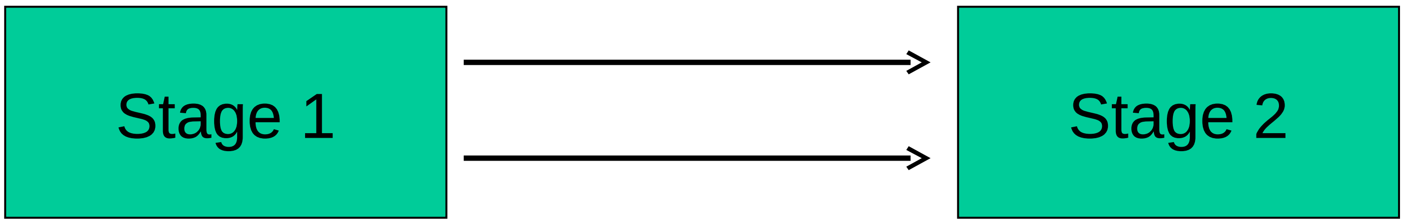
Quantitative Effects

- As a result of pipelining:
  - Time in ns per instruction goes up
  - Each instruction takes more cycles to execute
  - But… average CPI remains roughly the same
  - Clock speed goes up
  - Total execution time goes down, resulting in lower average time per instruction
  - Under ideal conditions, speedup

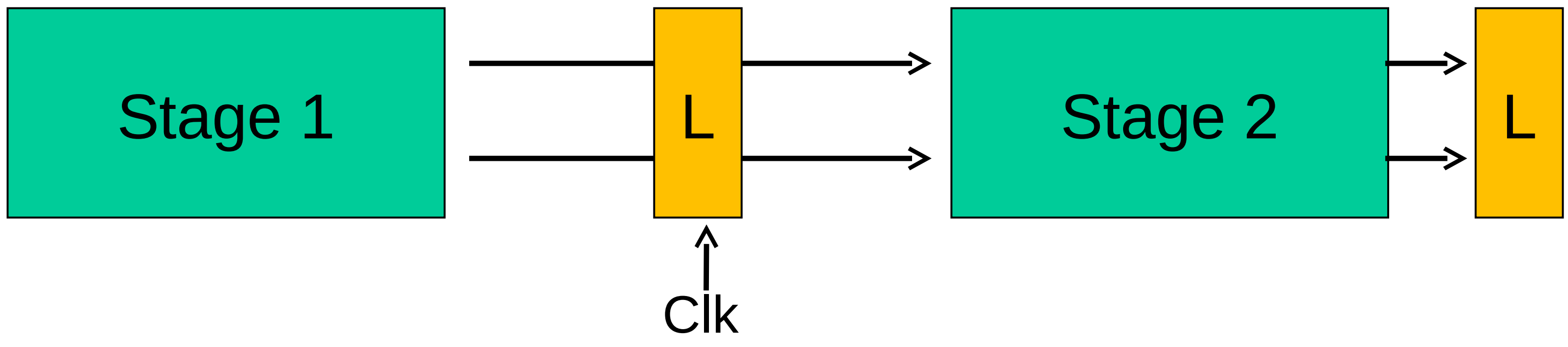
    = ratio of elapsed times between successive instruction completions

    = number of pipeline stages = increase in clock speed

Clocks and Latches



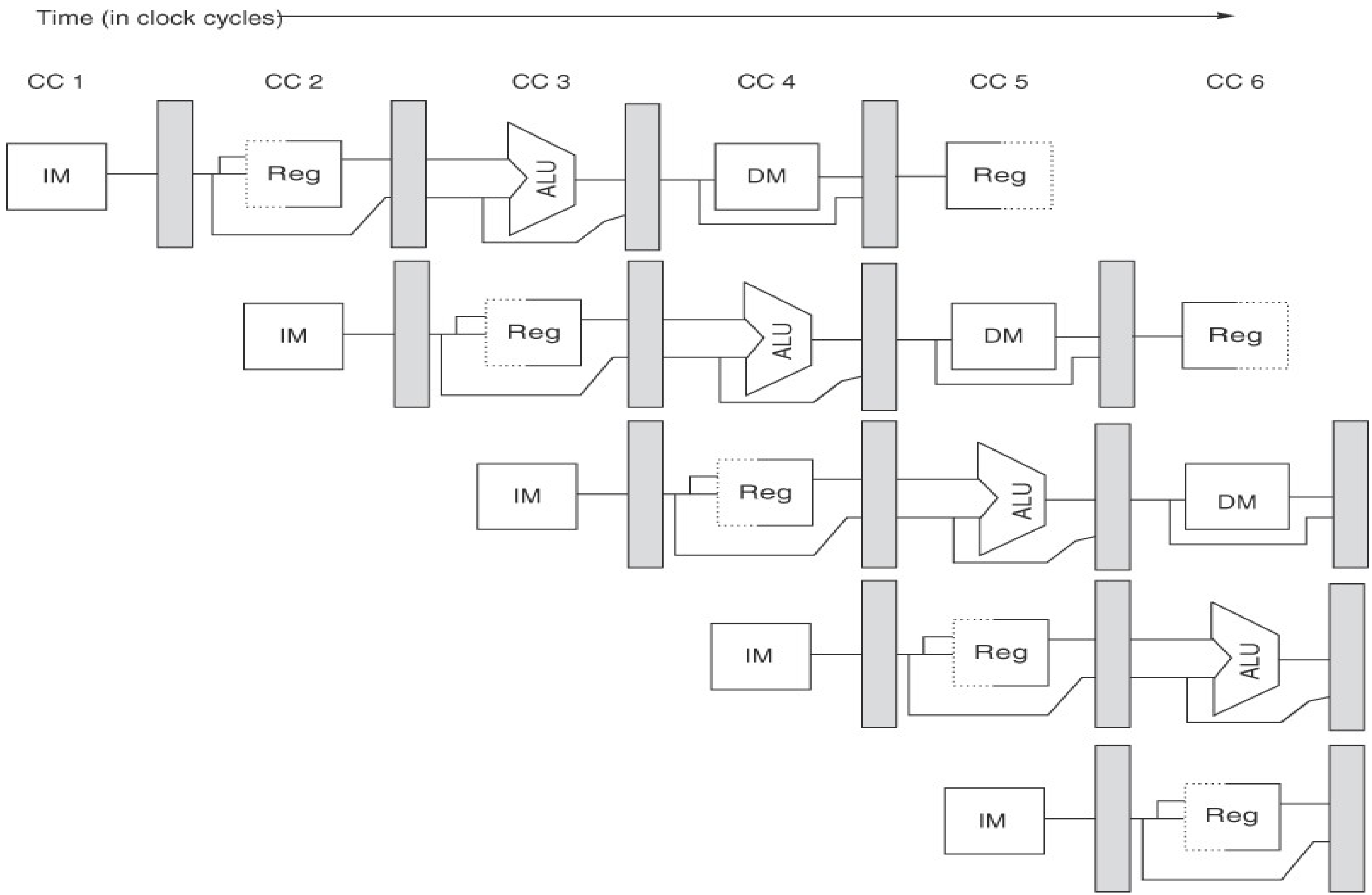
Clocks and Latches



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

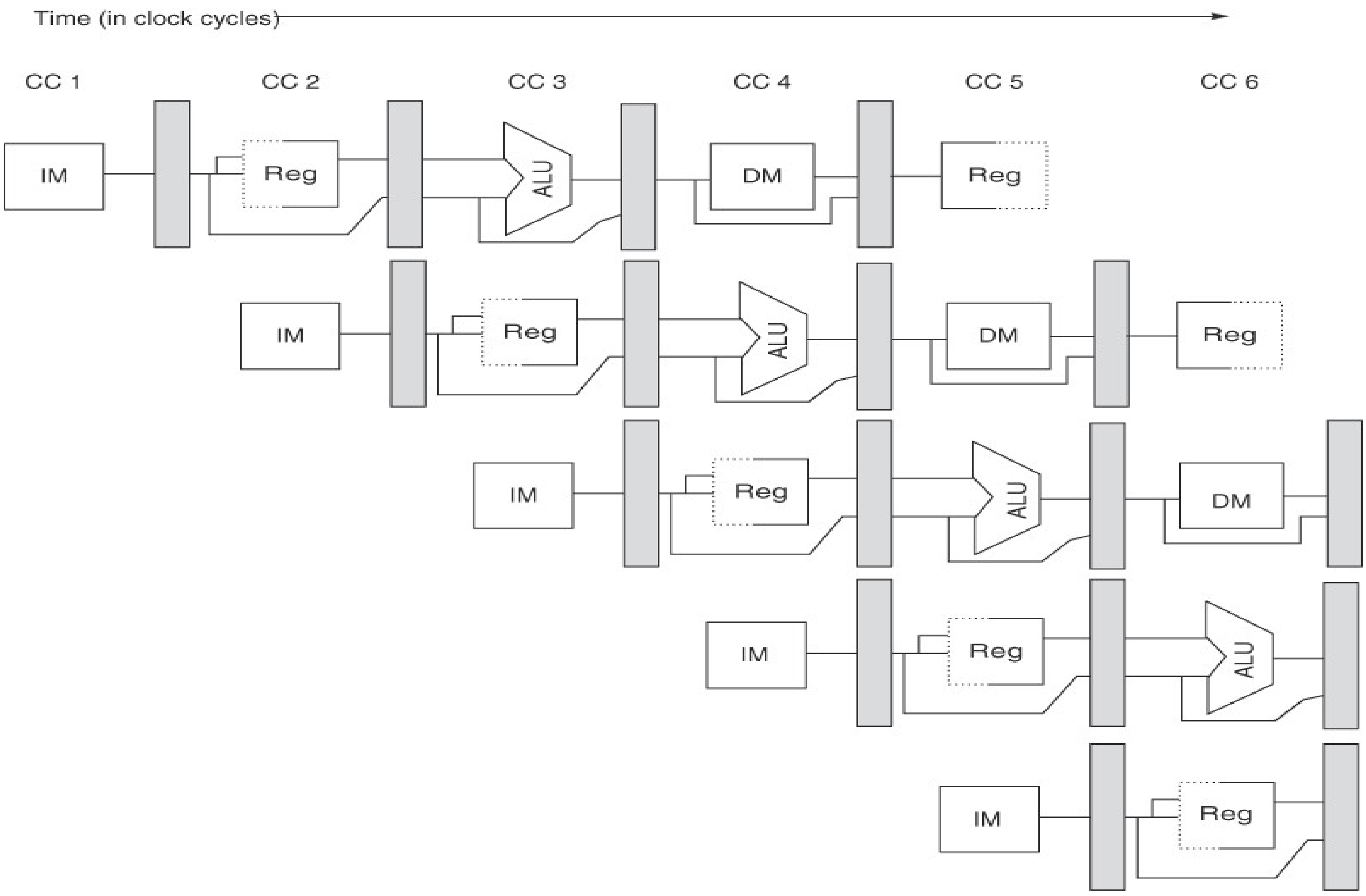
A 5-Stage Pipeline



Source: H&P textbook

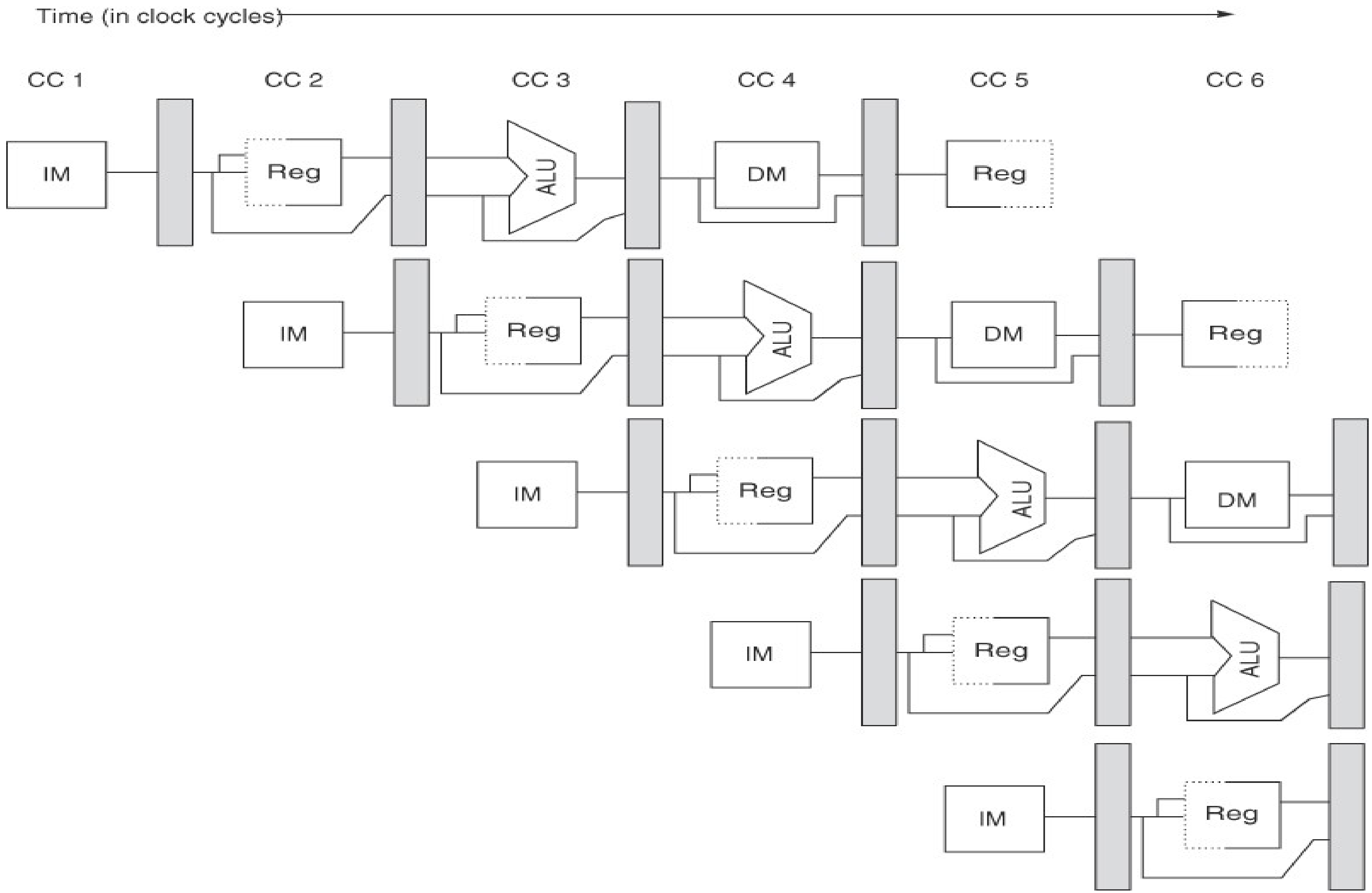
A 5-Stage Pipeline

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



A 5-Stage Pipeline

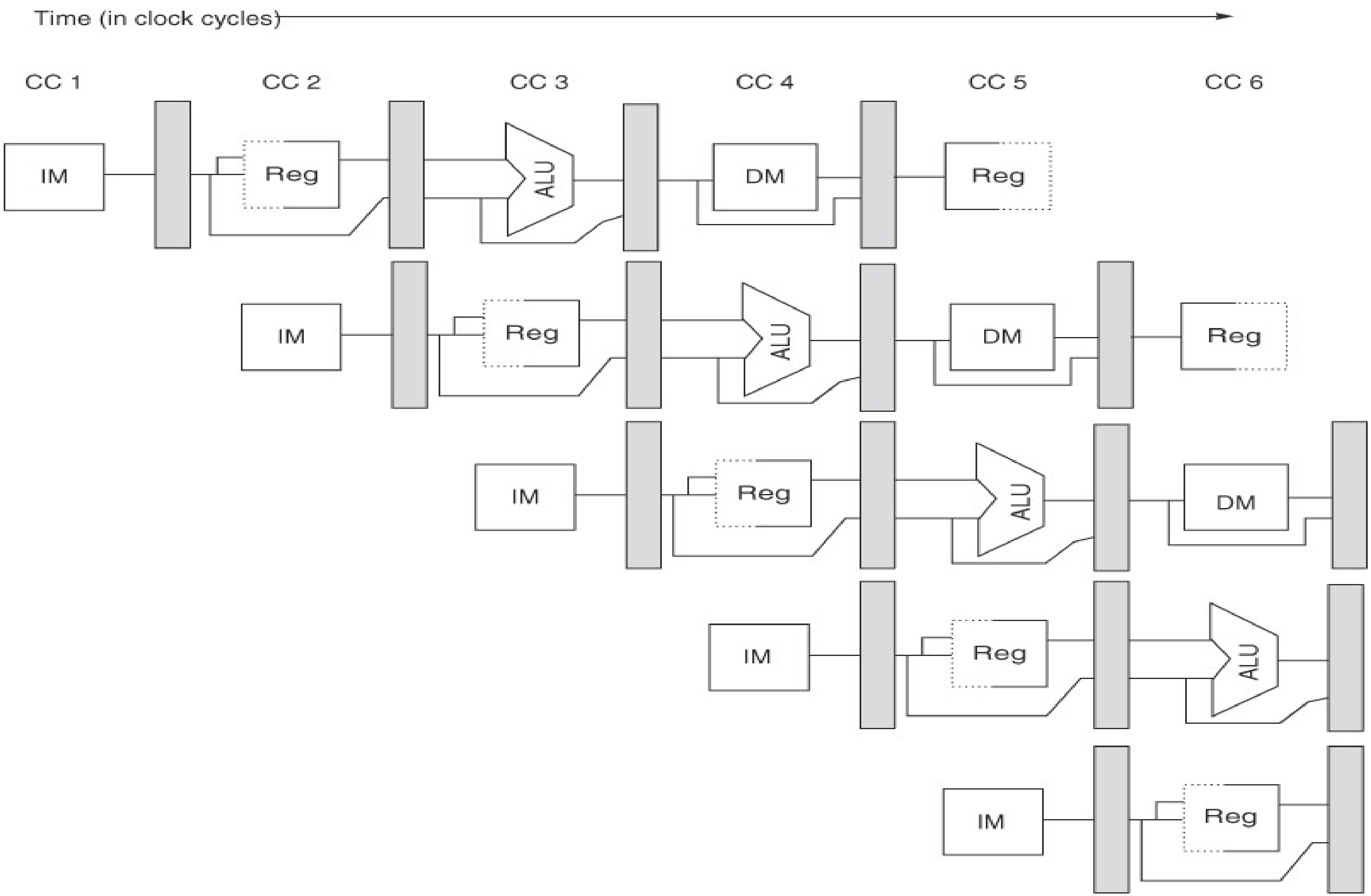
Read registers, compare registers, compute branch target; for now, assume branches take 2 cyc (there is enough work that branches can easily take more)



RISC/CISC Loads/Stores

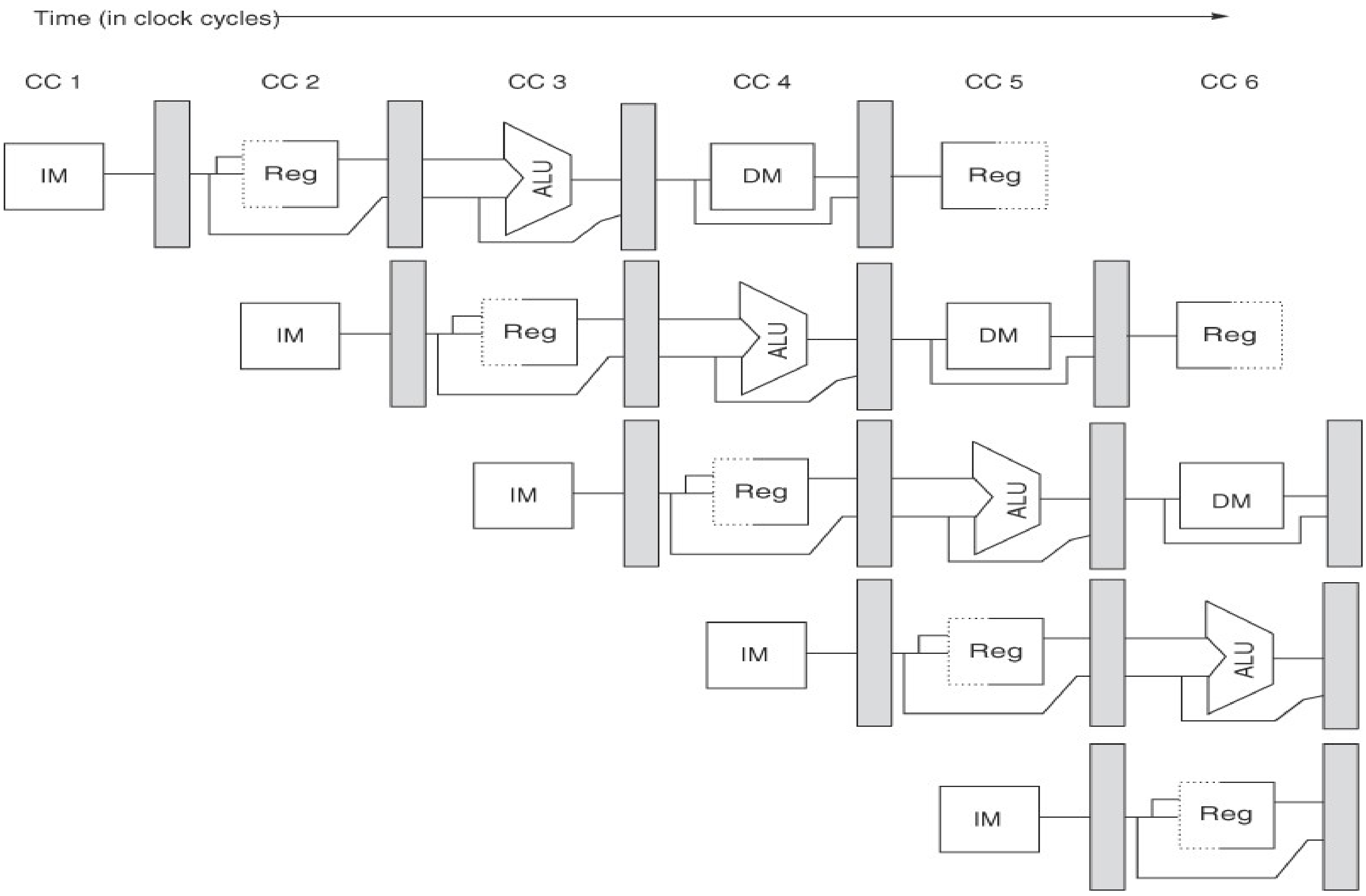
A 5-Stage Pipeline

ALU computation, effective address computation for load/store



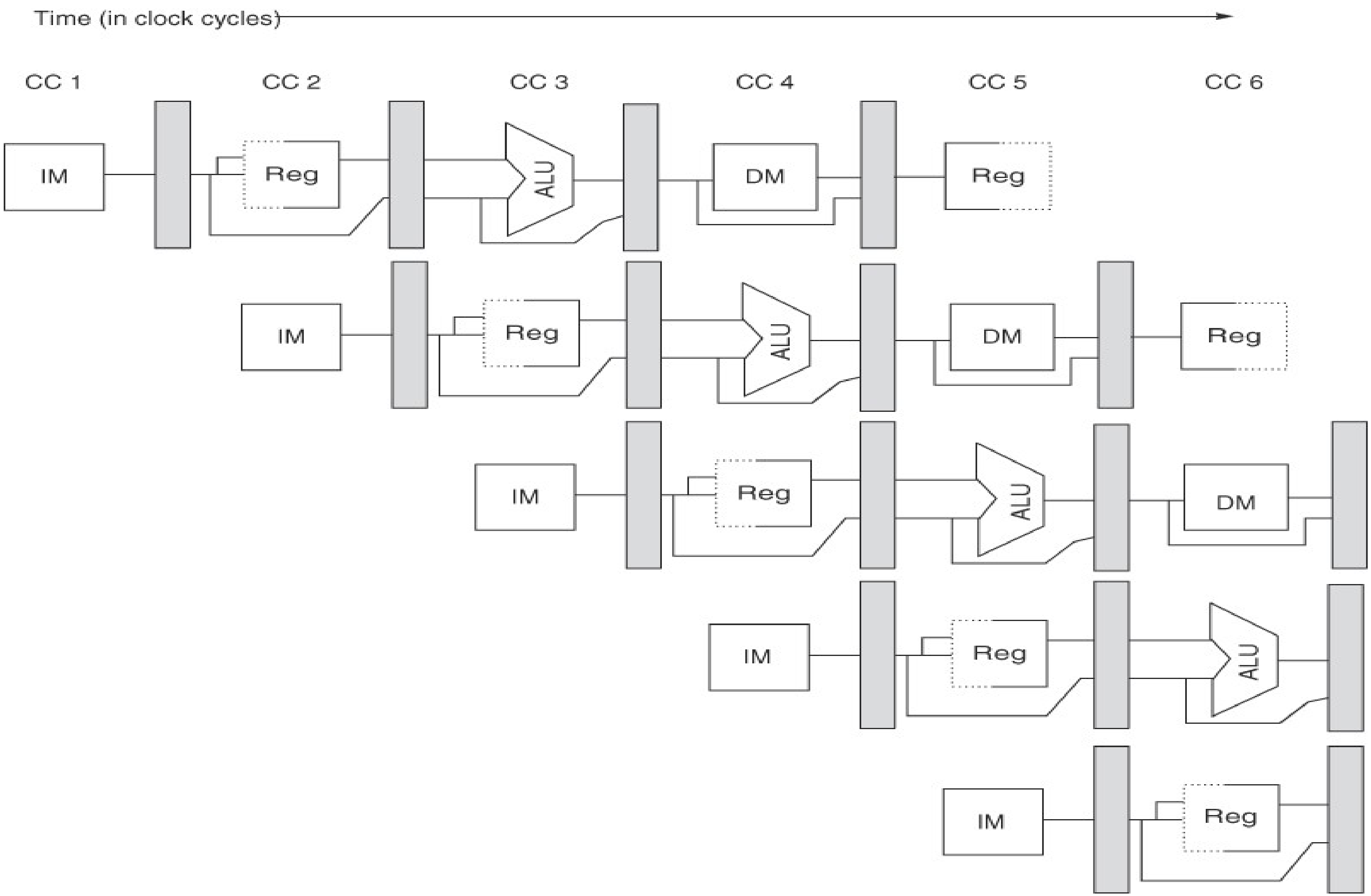
A 5-Stage Pipeline

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



Thank you!

AM vs. GM

- GM of IPCs = 1 / GM of CPIs
- AM of IPCs represents thruput for a workload where each program runs sequentially for 1 cycle each; but high-IPC programs contribute more to the AM
- GM of IPCs does not represent run-time for any real workload (what does it mean to multiply instructions?); but every program's IPC contributes equally to the final measure

Speedup Vs. Percentage

- “Speedup” is a ratio = old exec time / new exec time
- “Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline
  = (new perf – old perf) / old perf
- A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  - What is the speedup?
  - What is the percentage increase in performance?
  - What is the reduction in execution time?