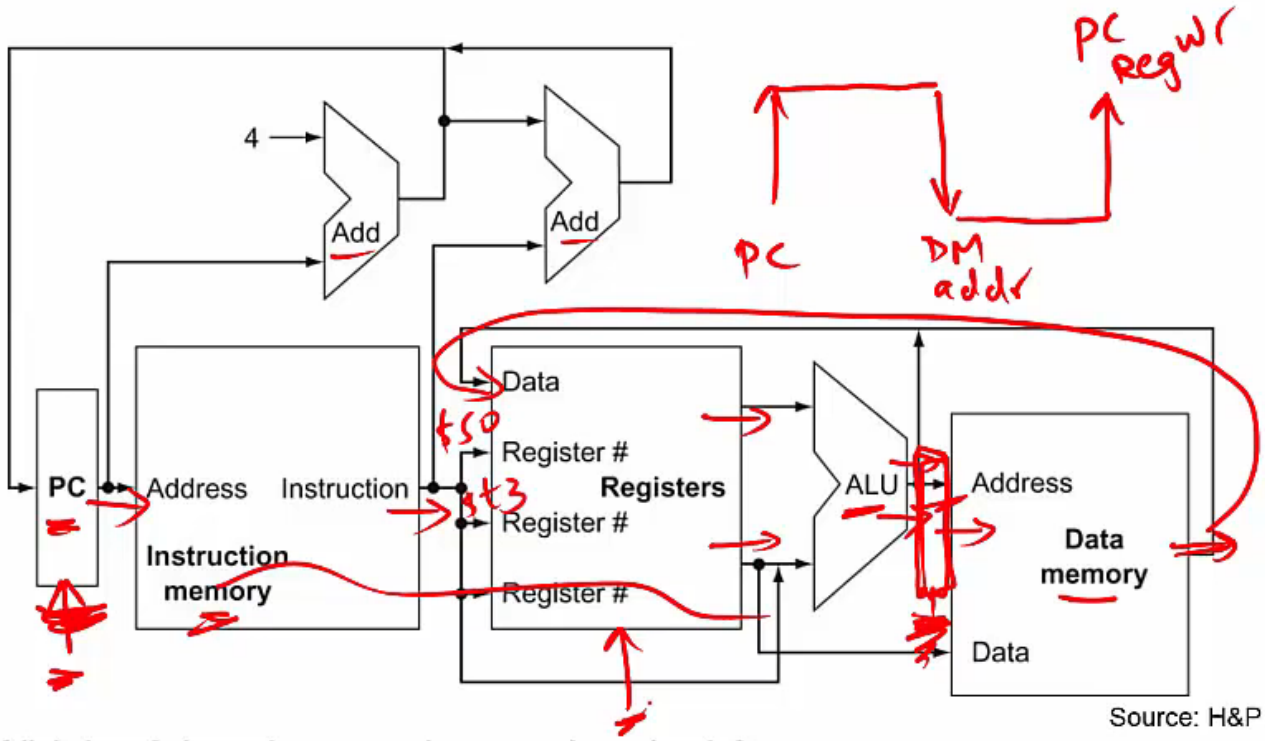
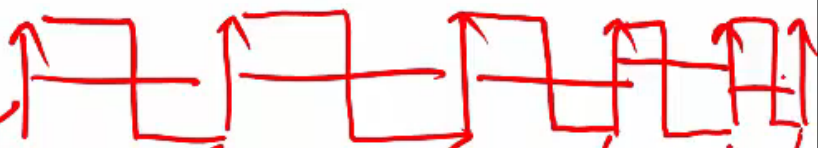
# Clocking Methodology



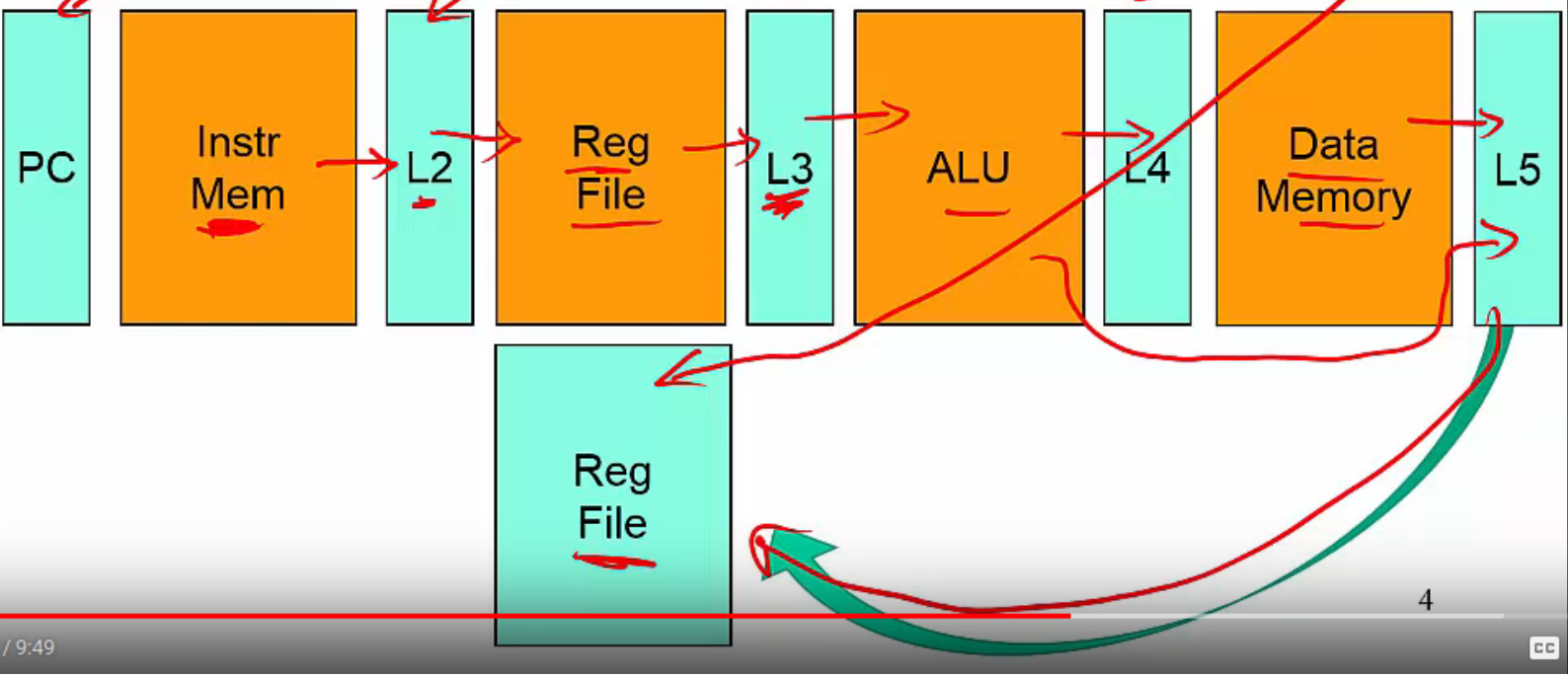

Source: H&P textbook

- Which of the above units need a clock?
- What is being saved (latched) on the rising edge of the clock? Keep in mind that the latched value remains there for an entire 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



# The Assembly Line

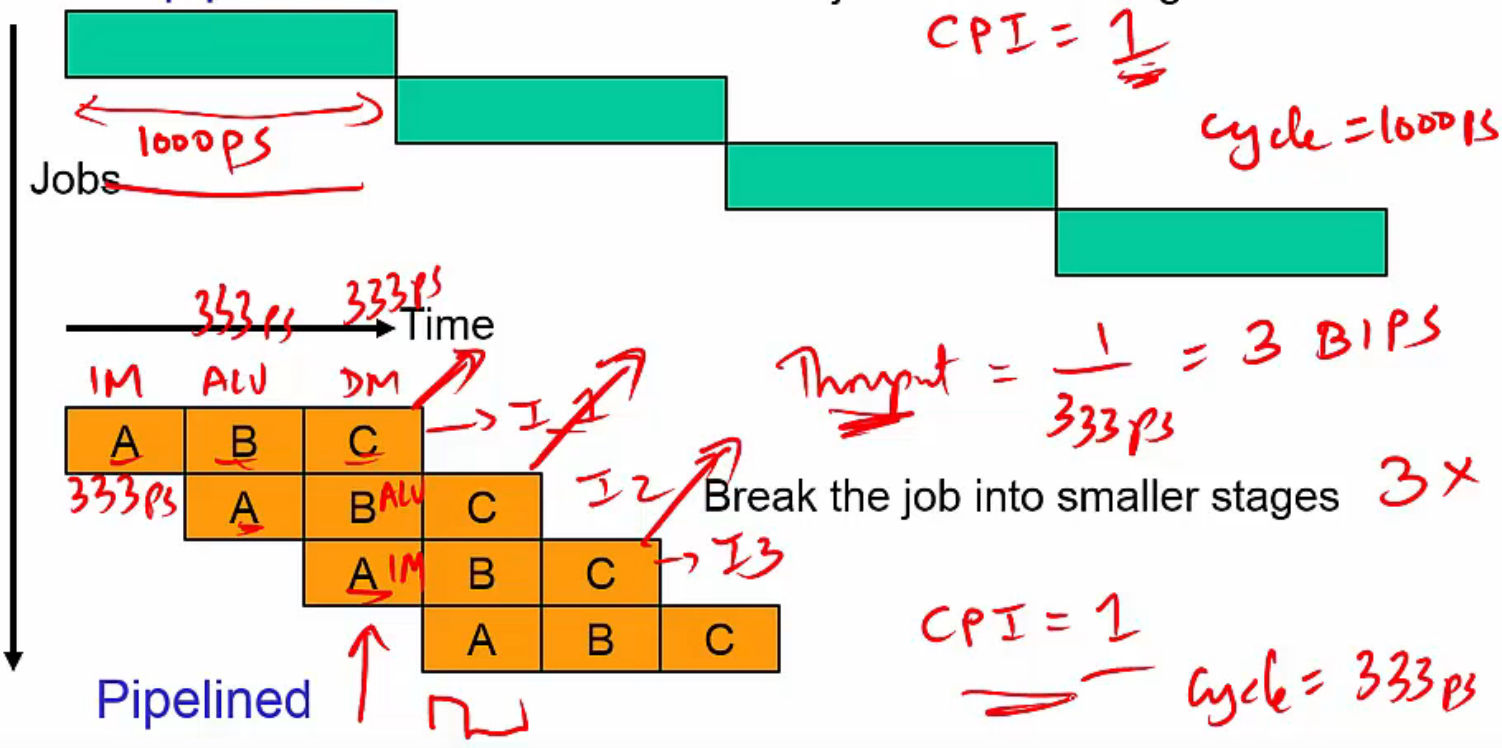
Thruput = $\frac{1}{1000ps}$ = 1 BIPS

Unpipelined

Start and finish a job before moving to the next

CPI = 1

cycle = 1000ps




Thruput = $\frac{1}{333ps}$ = 3 BIPS

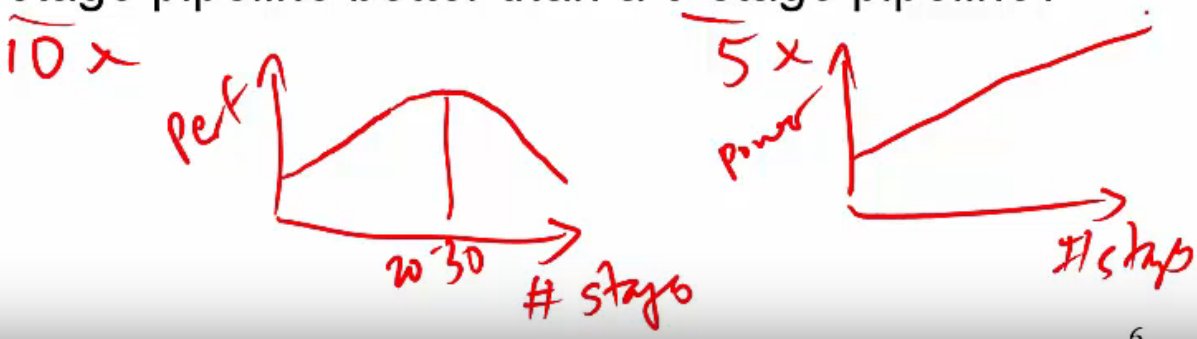
Break the job into smaller stages 3×

CPI = 1

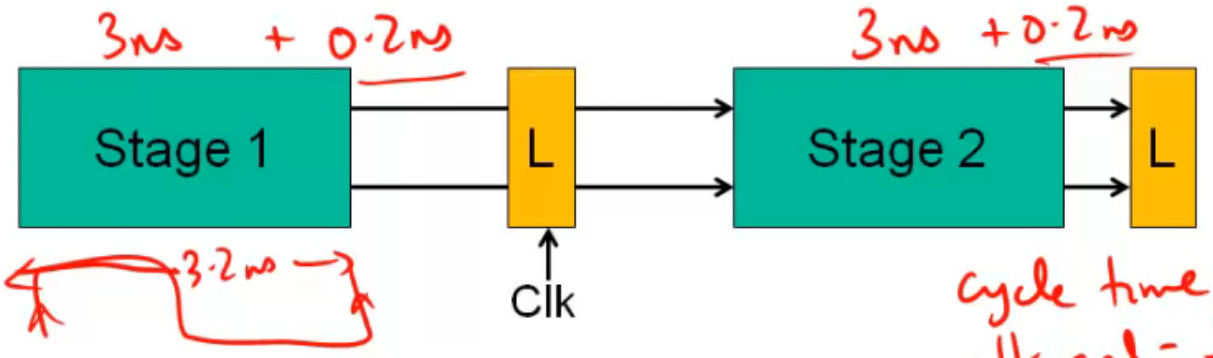
cycle = 333ps

# 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?

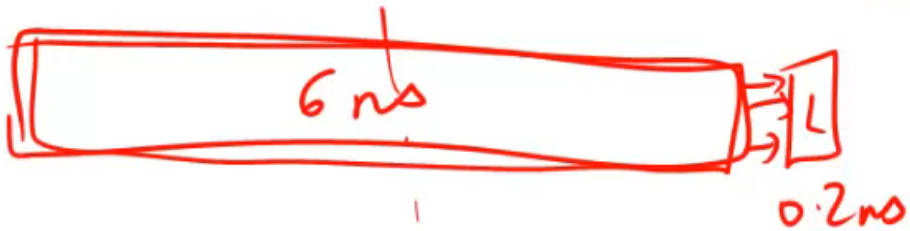


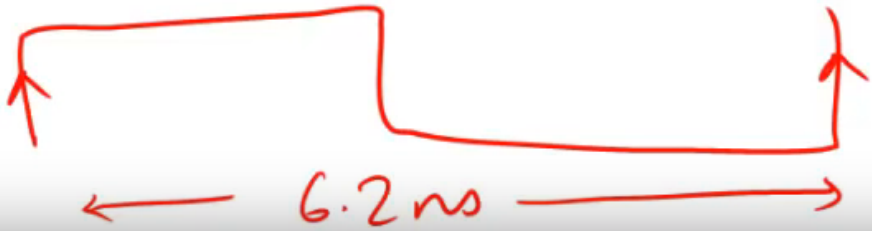
# Clocks and Latches




cycle time = 3.2ns

clk spd = $\frac{1}{3.2\text{ns}}$ = 312.5MHz



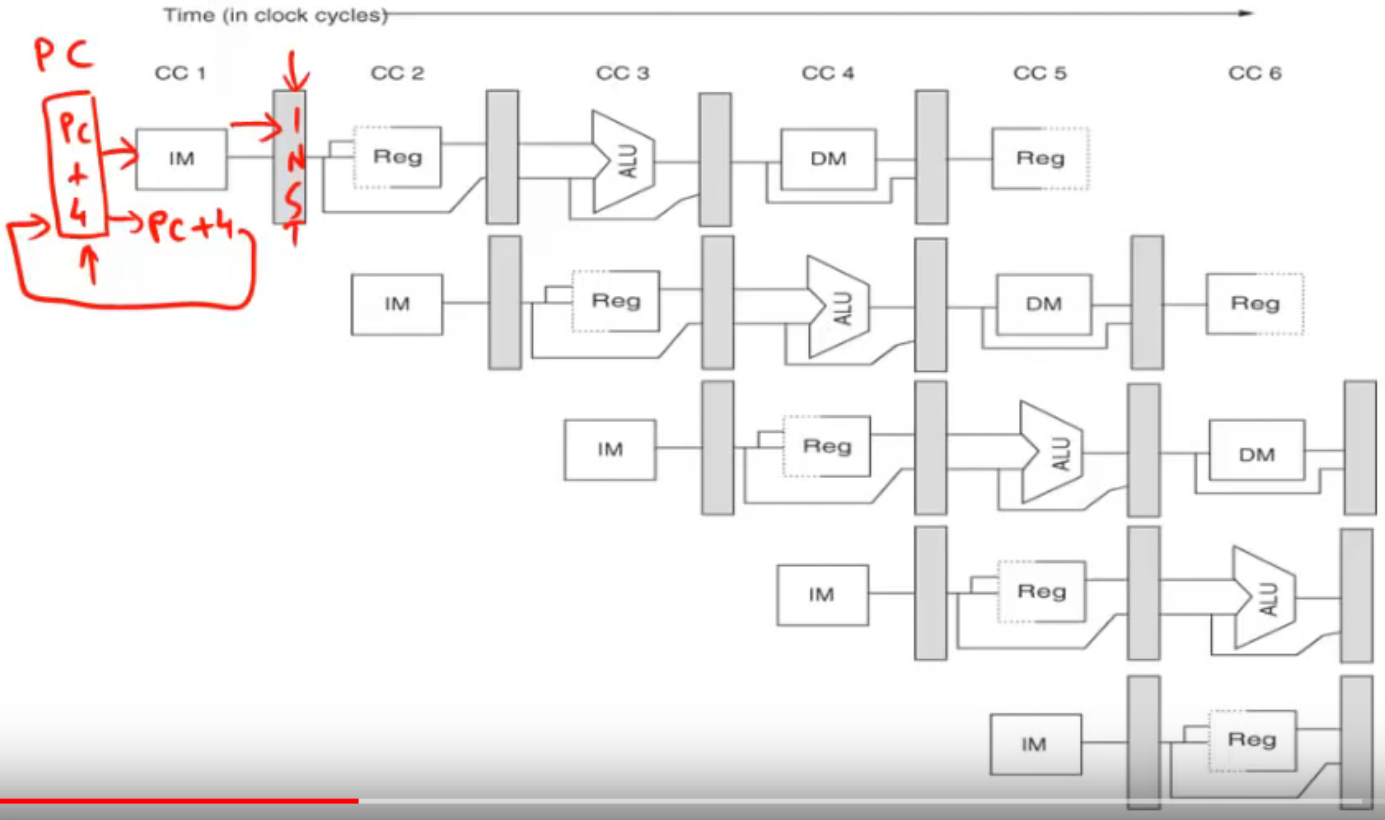




cycle time = 6.2ns

clk spd = $\frac{1}{6.2\text{ns}}$ = 160MHz

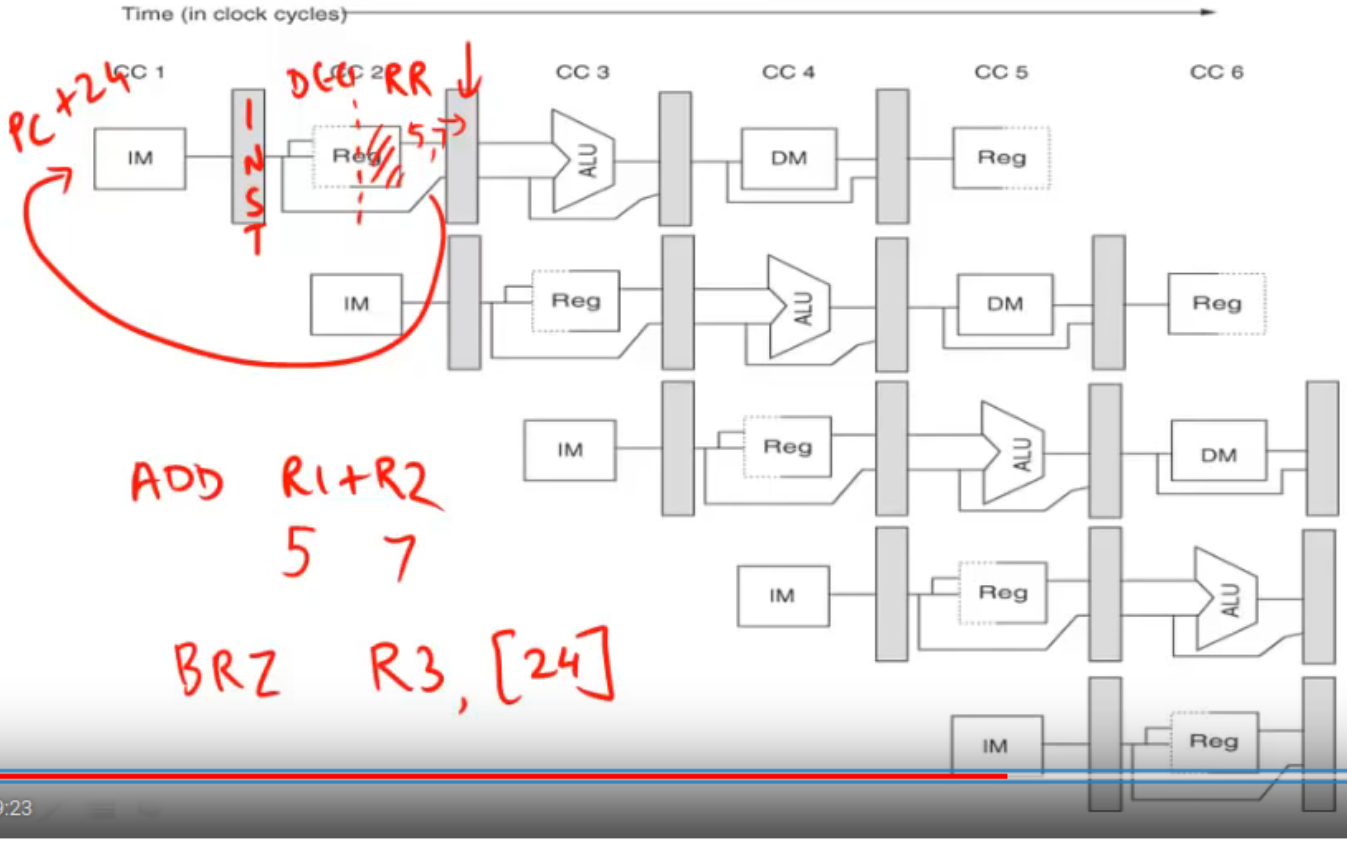
# A 5-Stage Pipeline

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



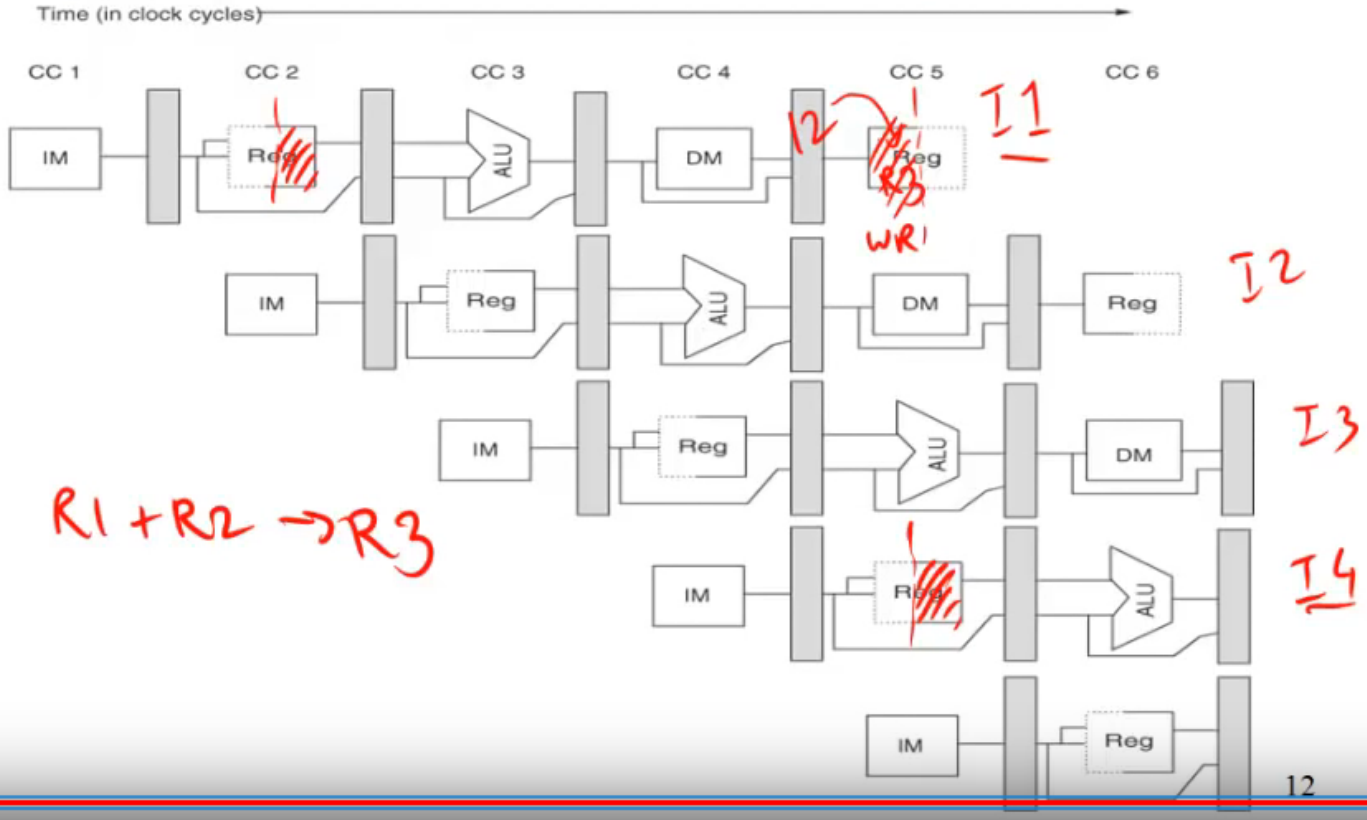
# A 5-Stage Pipeline

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

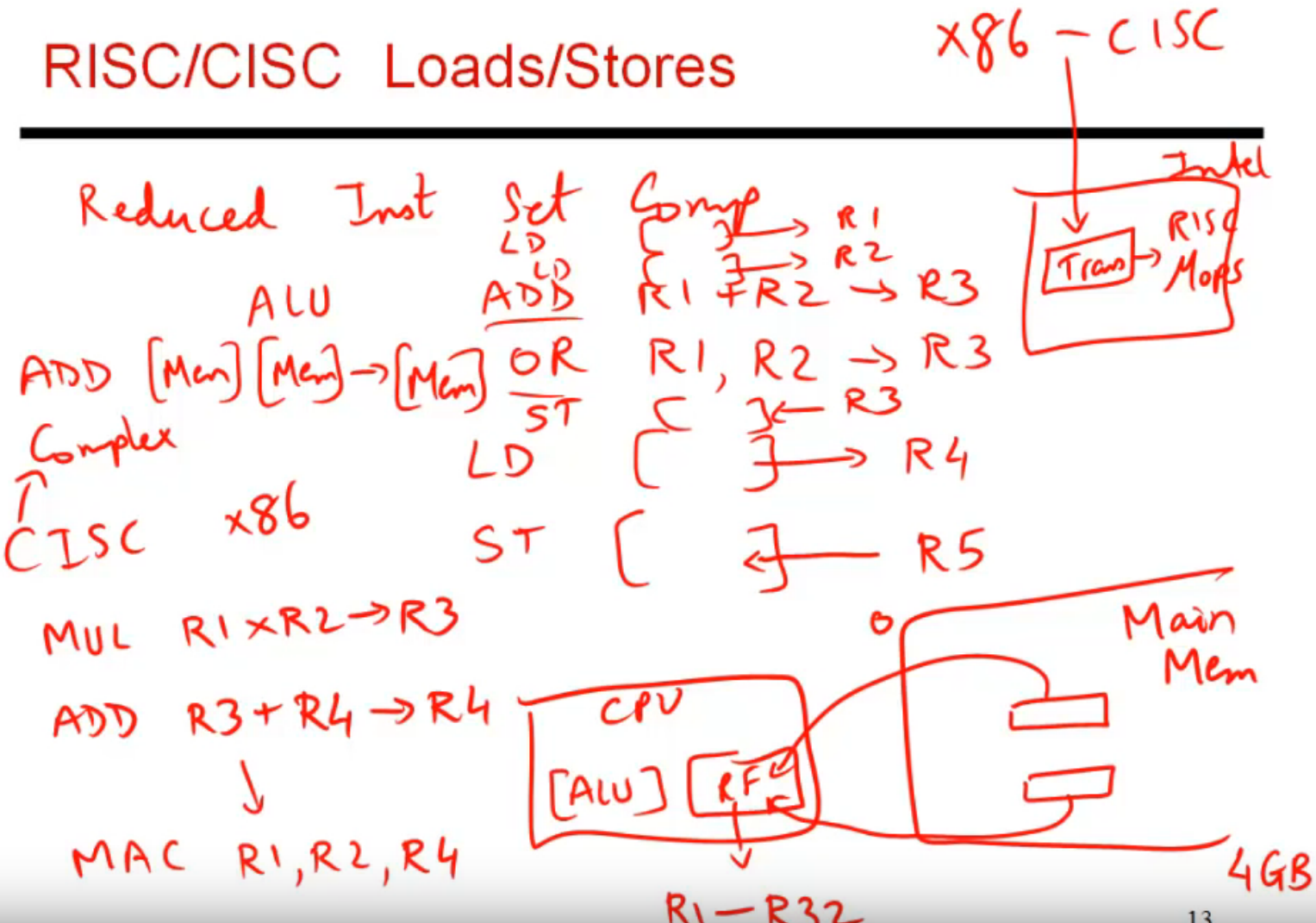


# A 5-Stage Pipeline

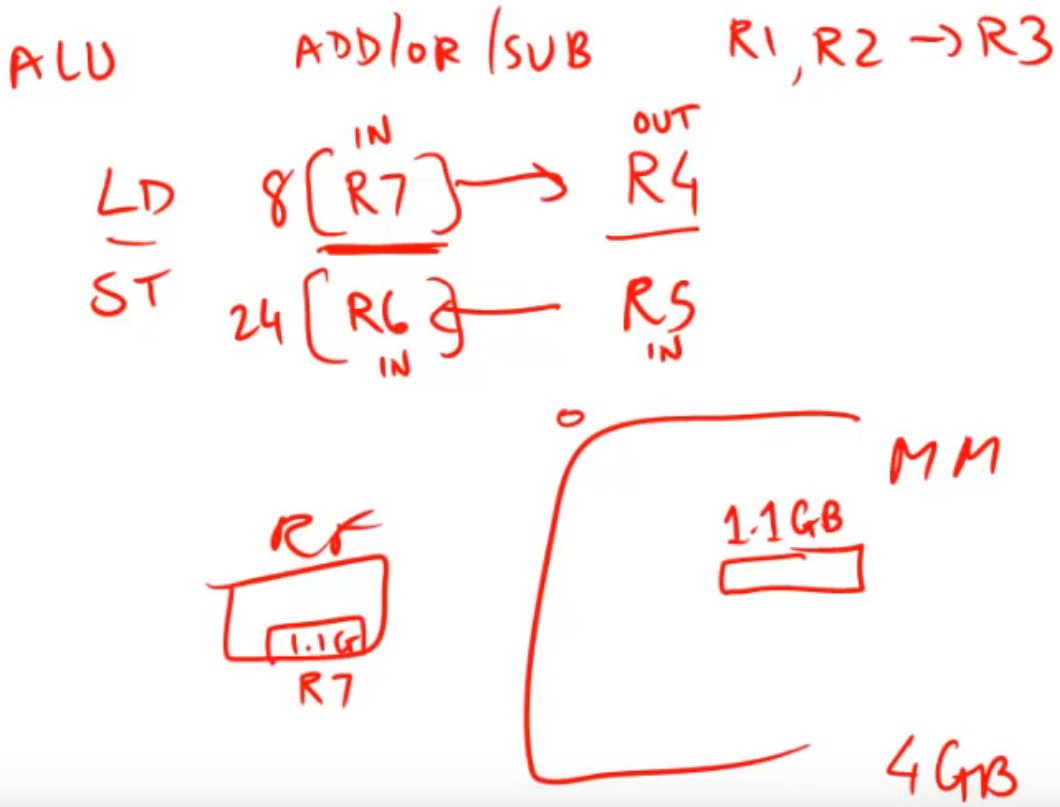
Write result of ALU computation or load into register file



# RISC/CISC Loads/Stores



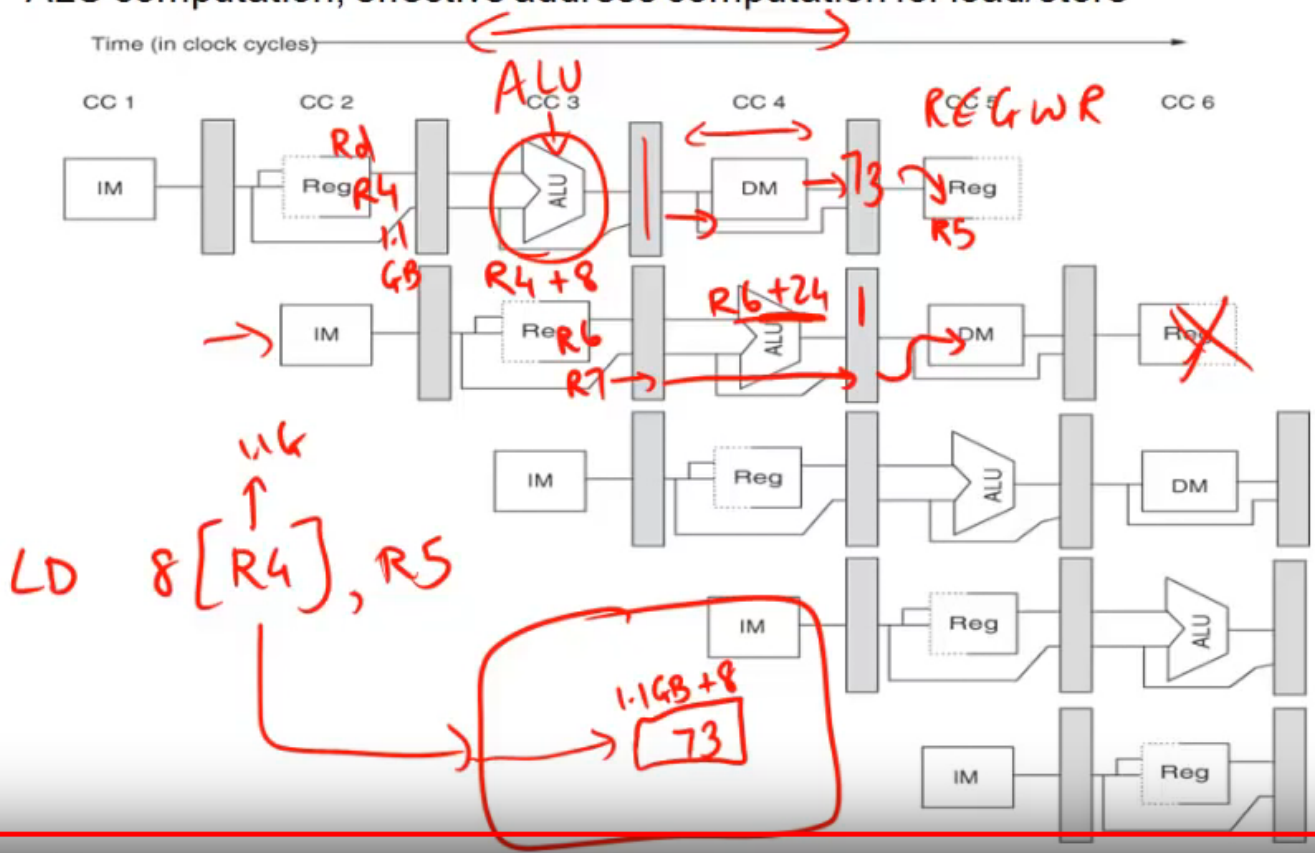
# RISC/CISC Loads/Stores



# A 5-Stage Pipeline

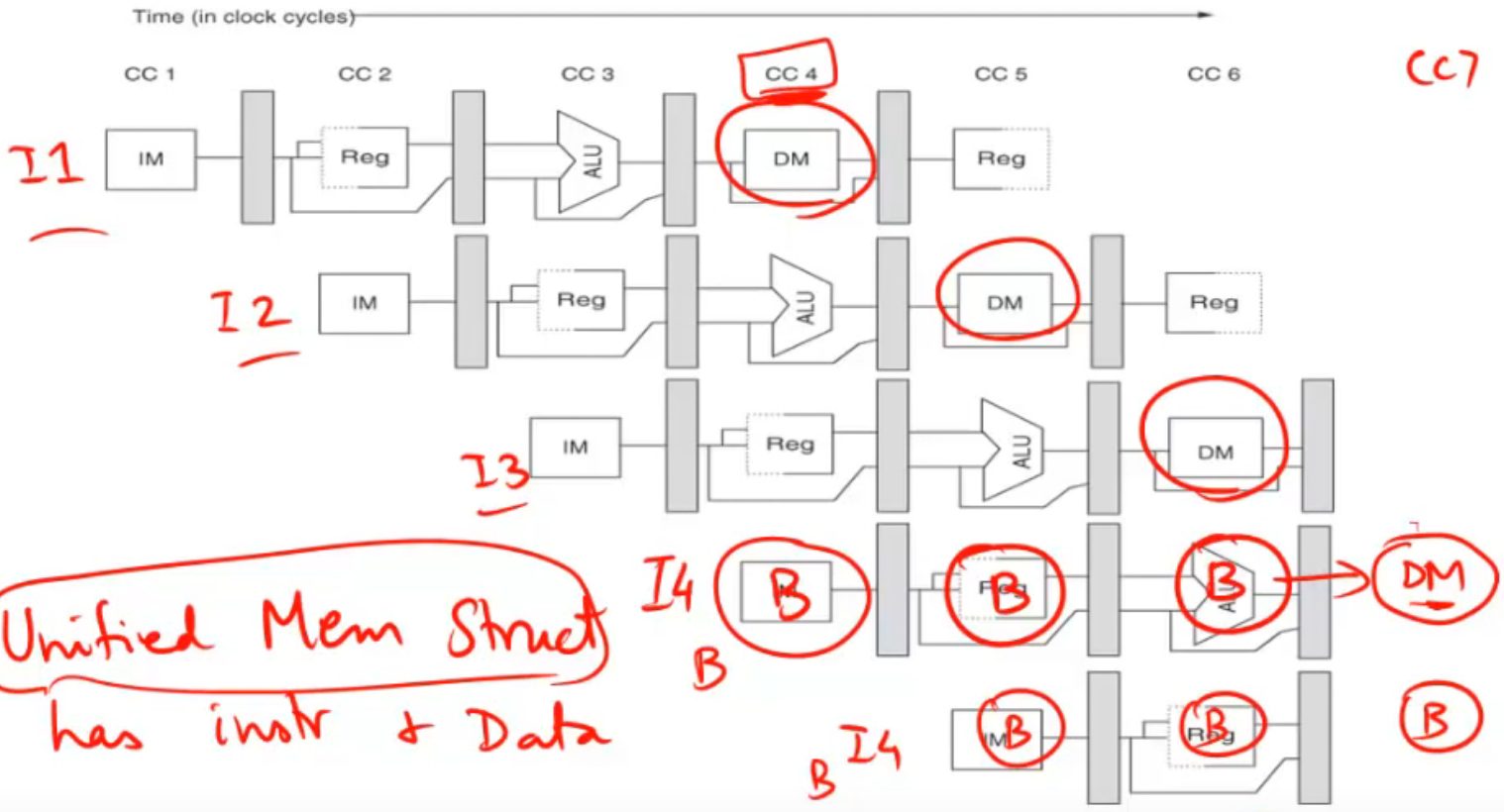
ST 24[R6], R7

ALU computation, effective address computation for load/store



# A 5-Stage Pipeline

IPC of 1 → 0.5




Source: Hennessy textbook