PIPELINING: 5-STAGE PIPELINE

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

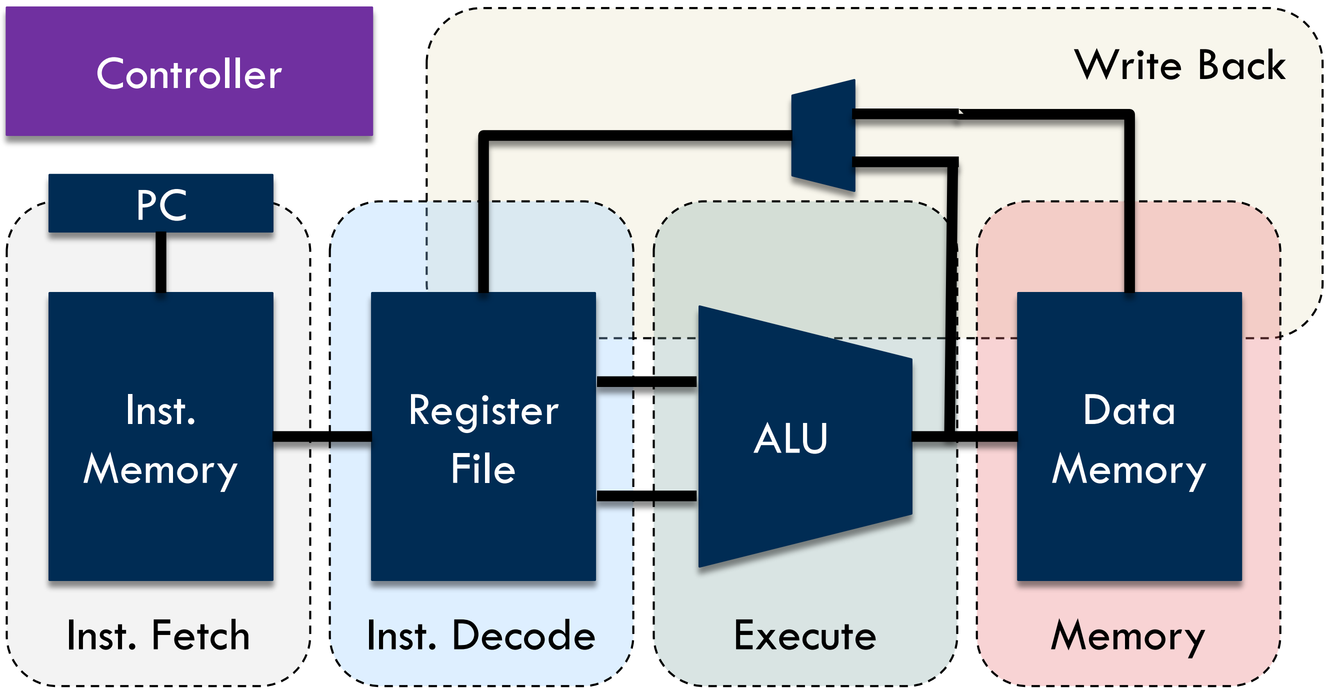
Overview

- Announcement
  - Tonight: Homework 1 deadline (11:59PM)
    - Verify your uploaded files before deadline

- This lecture
  - Impacts of pipelining on performance
  - The MIPS five-stage pipeline
  - Pipeline hazards
    - Structural hazards
    - Data hazards

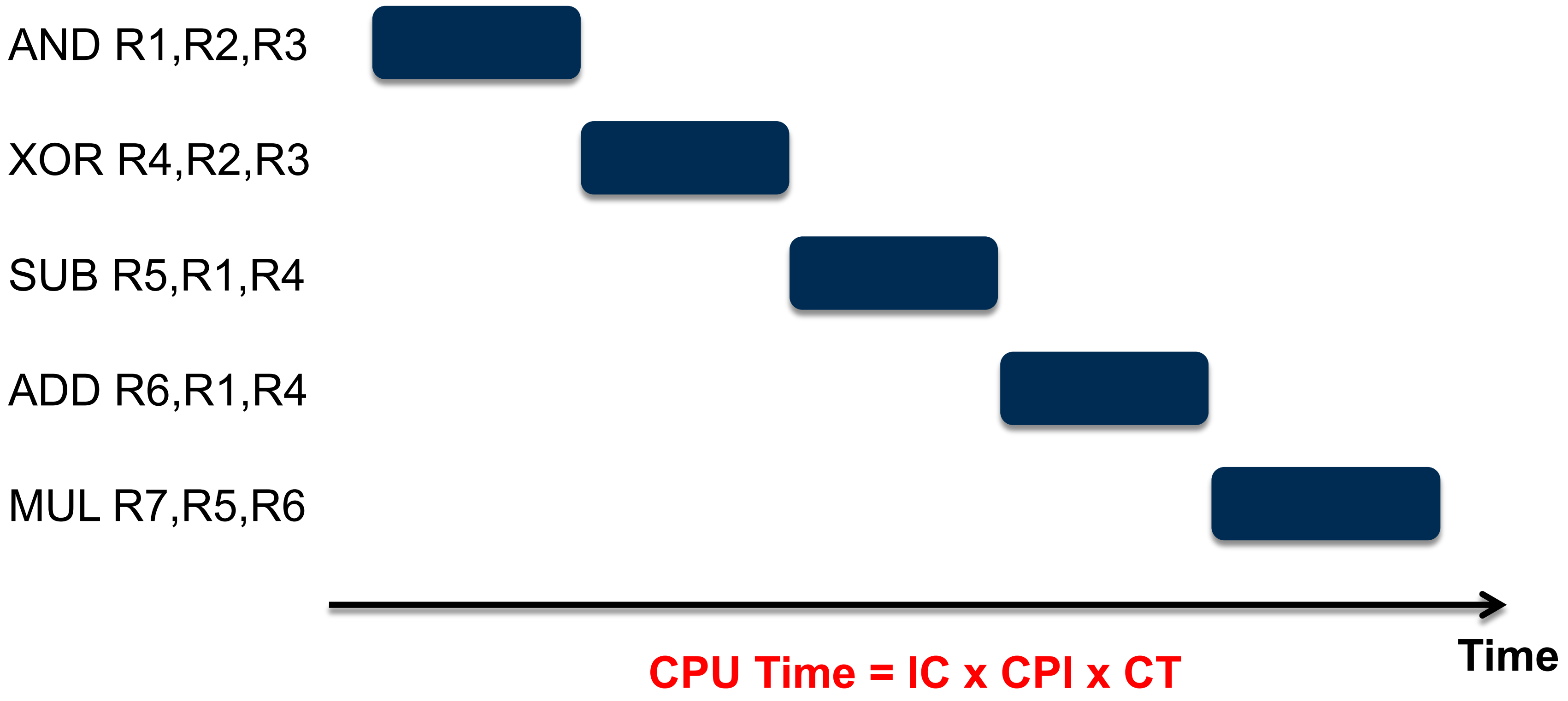
Single-cycle RISC Architecture

- Example: simple MIPS architecture
  - Critical path includes all of the processing steps



Single-cycle RISC Architecture

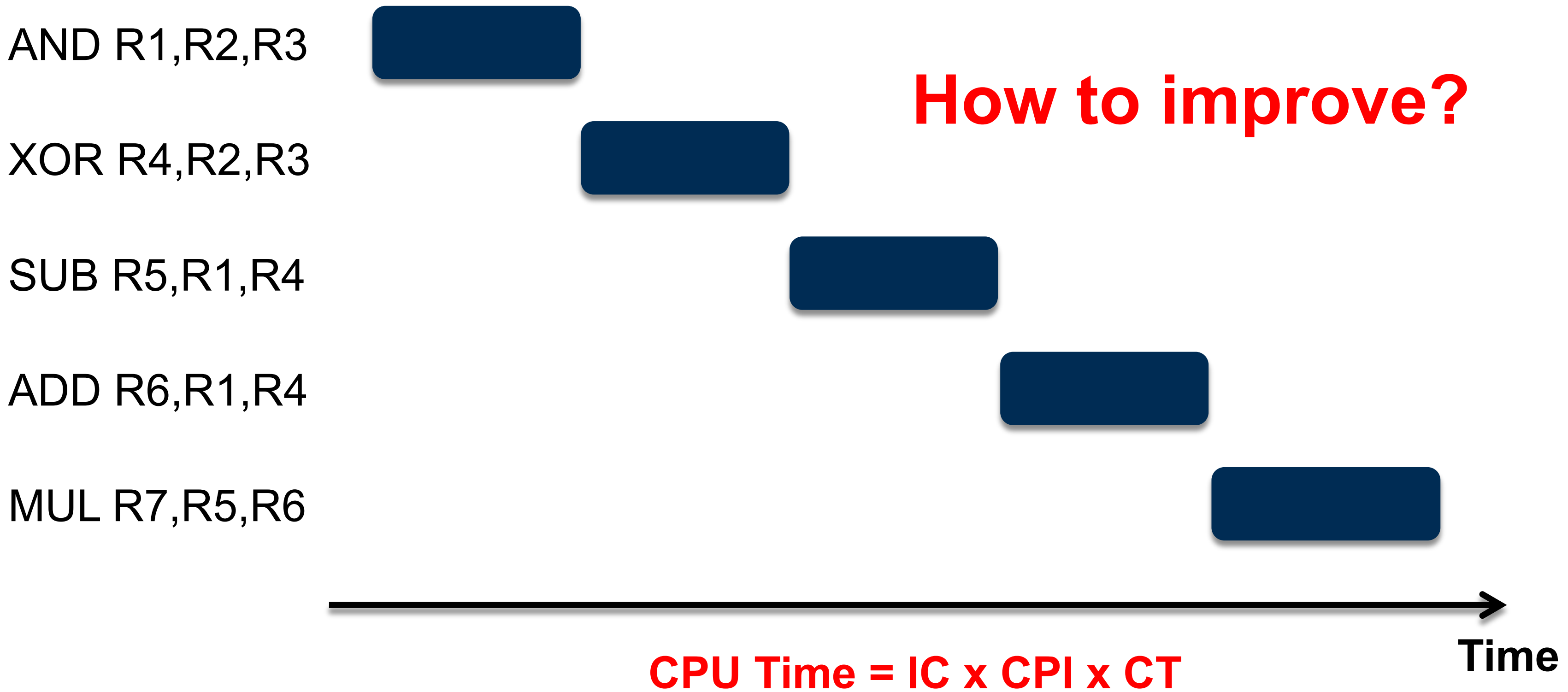
- Example program
  - CT=6ns; CPU Time = ?



CPU Time = IC x CPI x CT

Single-cycle RISC Architecture

- Example program
  - CT=6ns; CPU Time = 5 x 1 x 6ns = 30ns

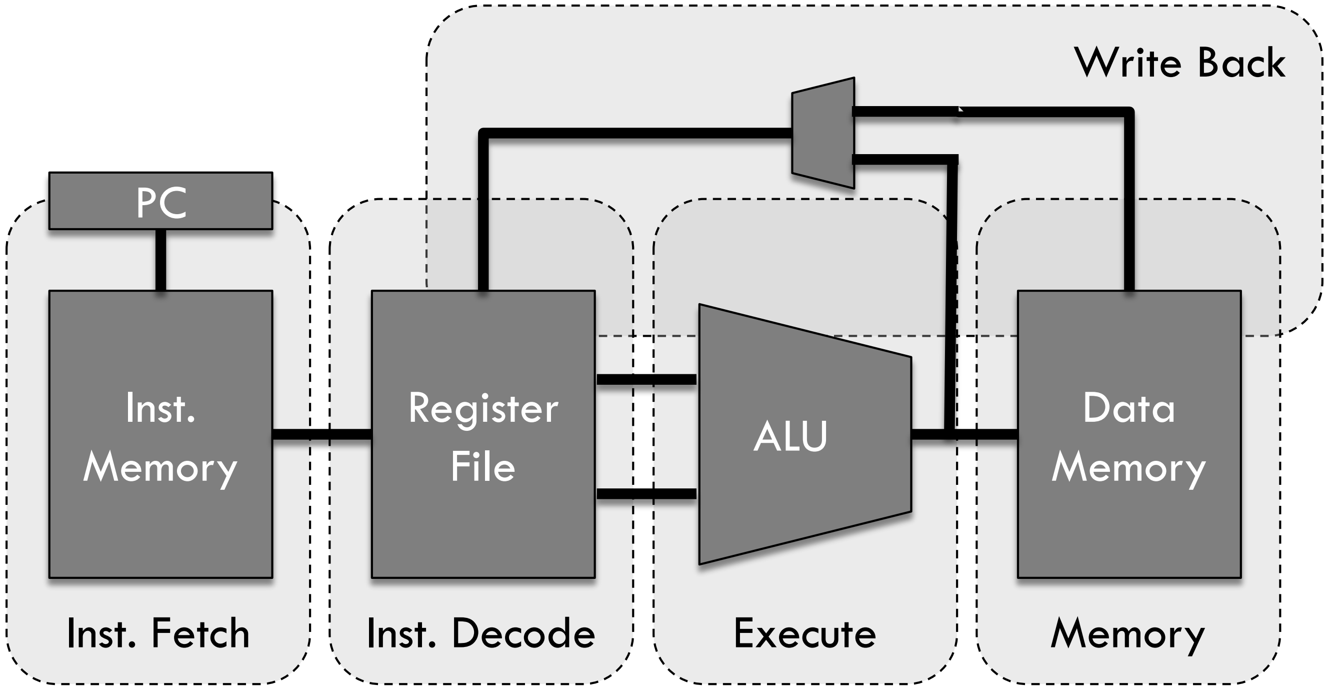


How to improve?

CPU Time = IC x CPI x CT

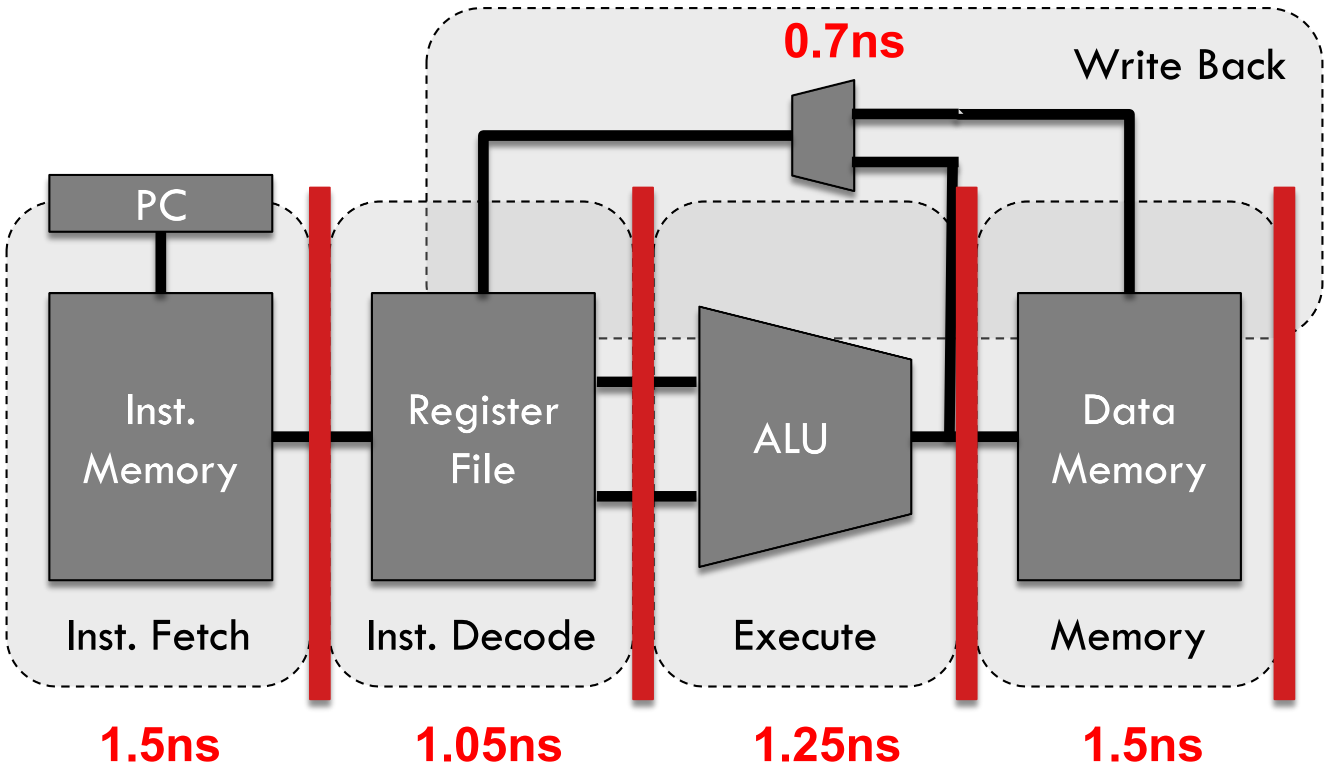
Reusing Idle Resources

- Each processing step finishes in a fraction of a cycle
  - Idle resources can be reused for processing next instructions



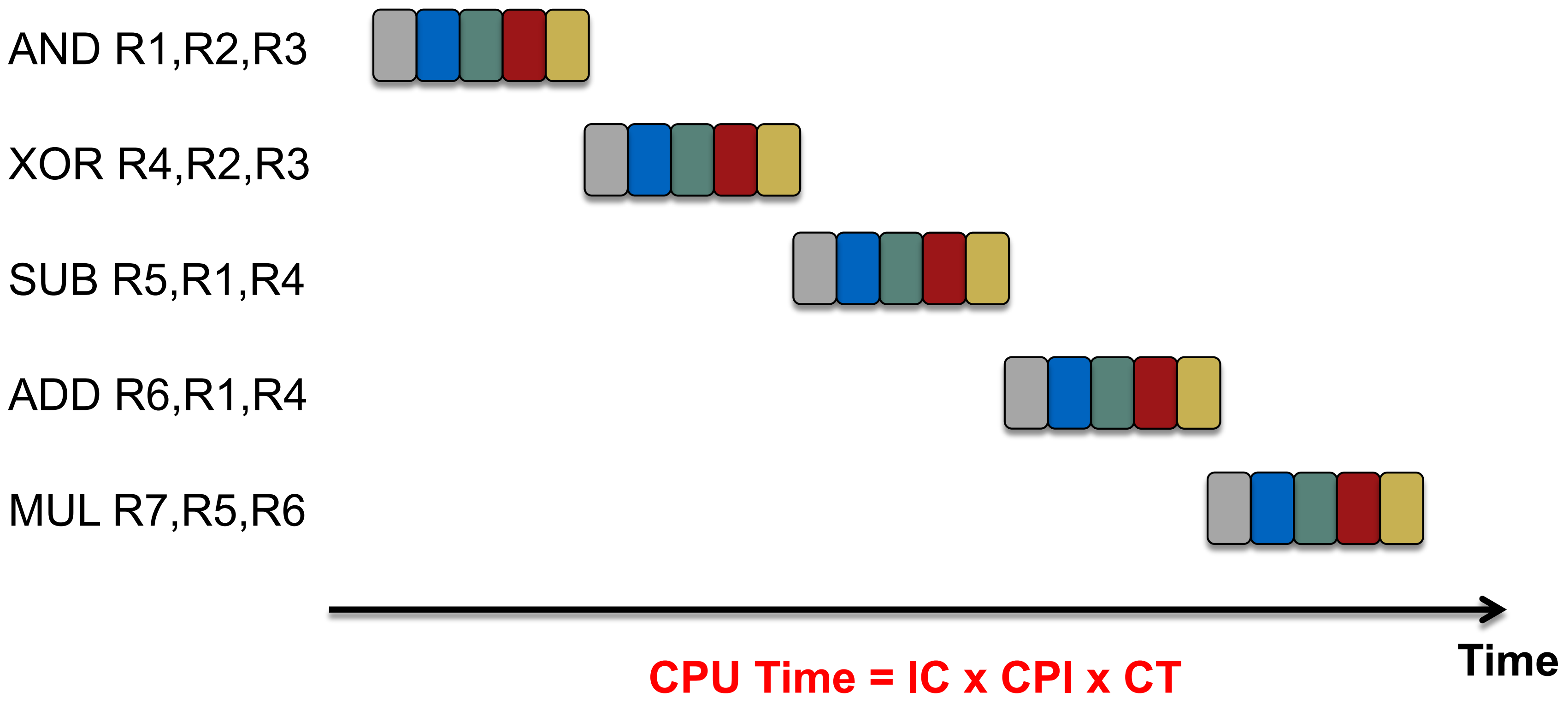
Pipelined Architecture

- Five stage pipeline
  - Critical path determines the cycle time



Pipelined Architecture

- Example program
  - CT=1.5ns; CPU Time = ?

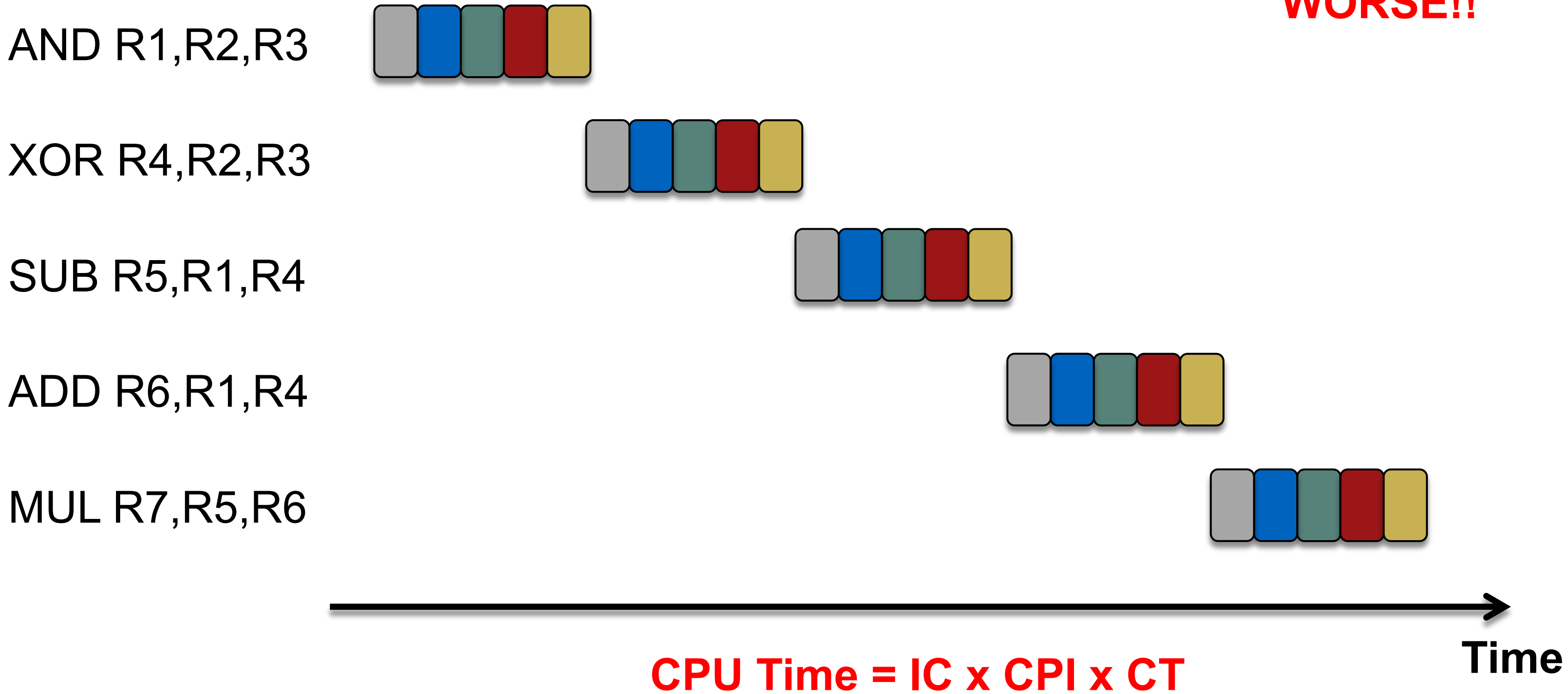


CPU Time = IC x CPI x CT

Pipelined Architecture

- Example program
  - CT=1.5ns; CPU Time = 5 x 5 x 1.5ns = 37.5ns > 30ns

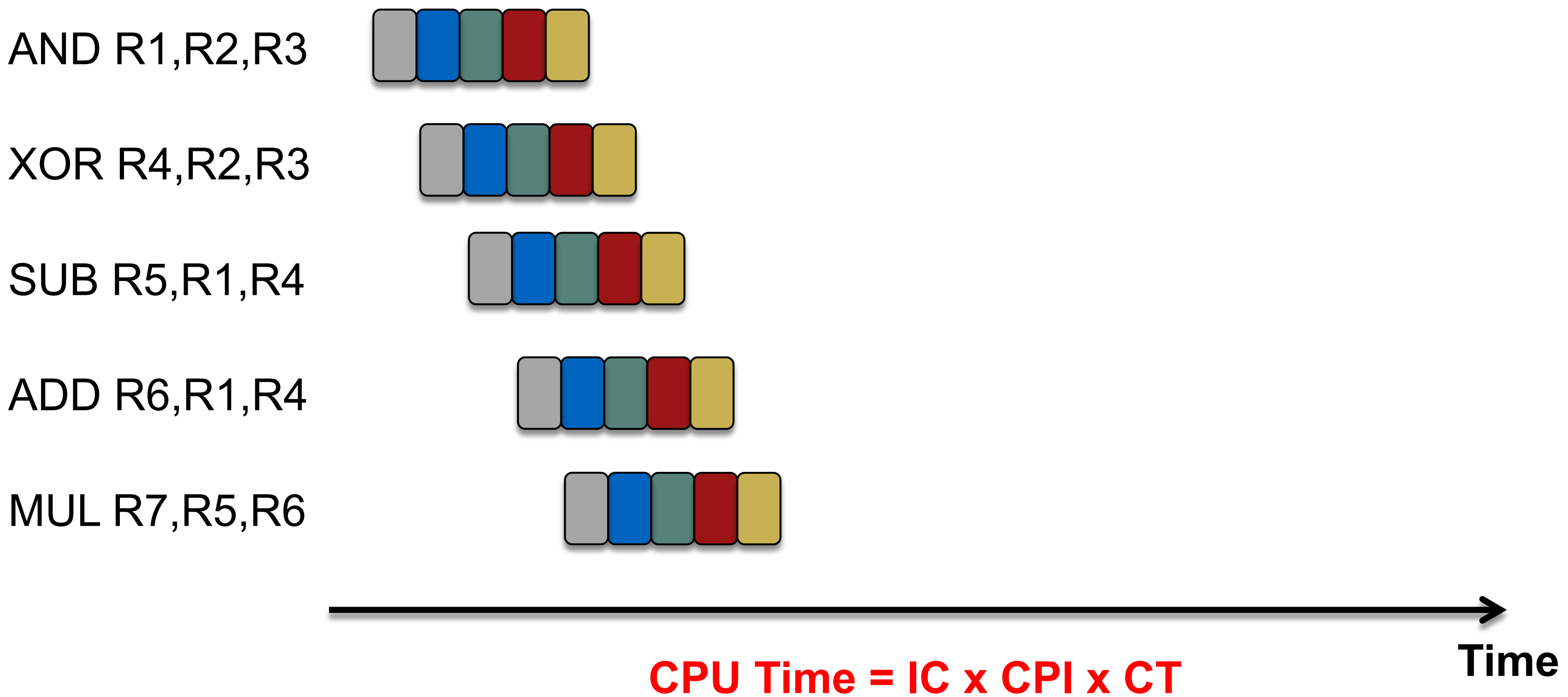
WORSE!!



CPU Time = IC x CPI x CT

Pipelined Architecture

- Example program
  - CT=1.5ns; CPU Time = ?



CPU Time = IC x CPI x CT

Pipelined Architecture

- Example program
  - CT=1.5ns; CPU Time = 9 x 1 x 1.5ns = 13.5ns

What is the cost of pipelining?

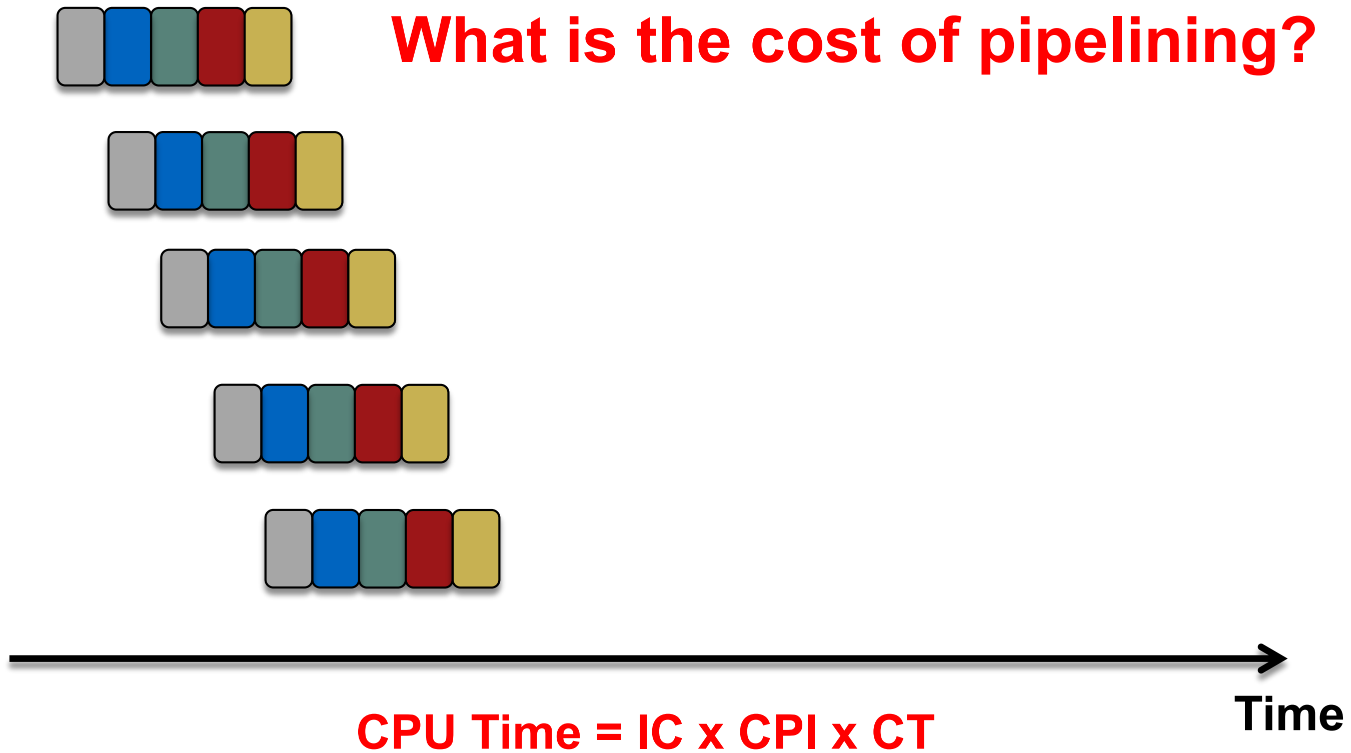
AND R1,R2,R3

XOR R4,R2,R3

SUB R5,R1,R4

ADD R6,R1,R4

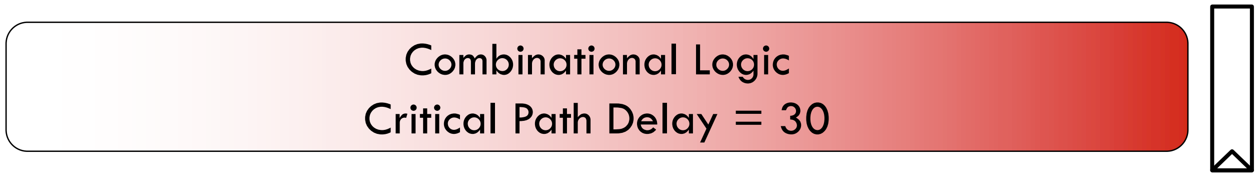
MUL R7,R5,R6



CPU Time = IC x CPI x CT

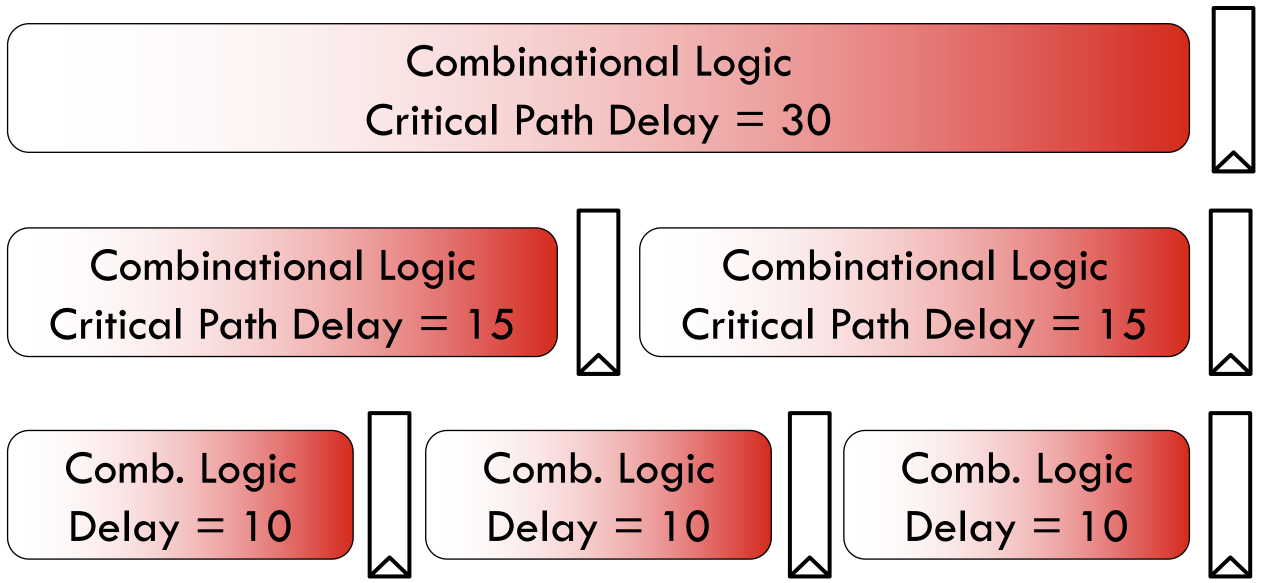
Pipelining Technique

- Improving throughput at the expense of latency
  - Delay: $D = T + n\delta$
  - Throughput: $IPS = n/(T + n\delta)$



Pipelining Technique

- Improving throughput at the expense of latency
  - Delay: $D = T + n\delta$
  - Throughput: $IPS = n/(T + n\delta)$



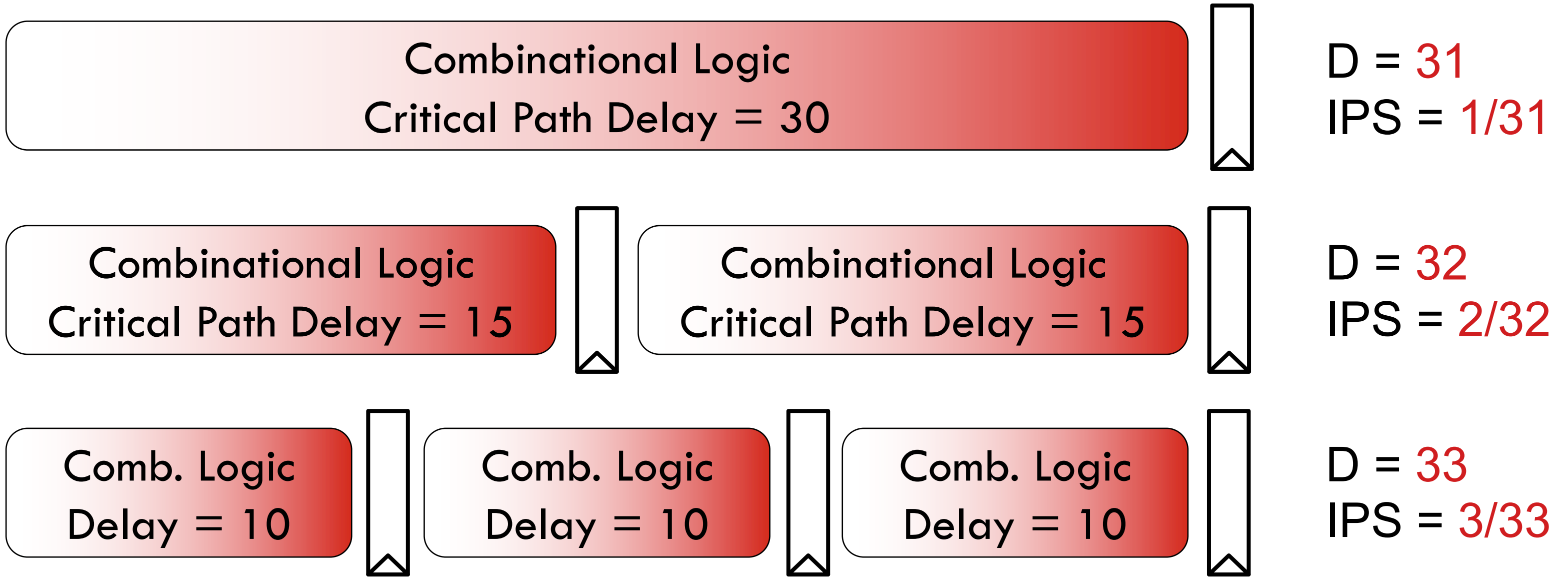
D =
IPS =

D =
IPS =

D =
IPS =

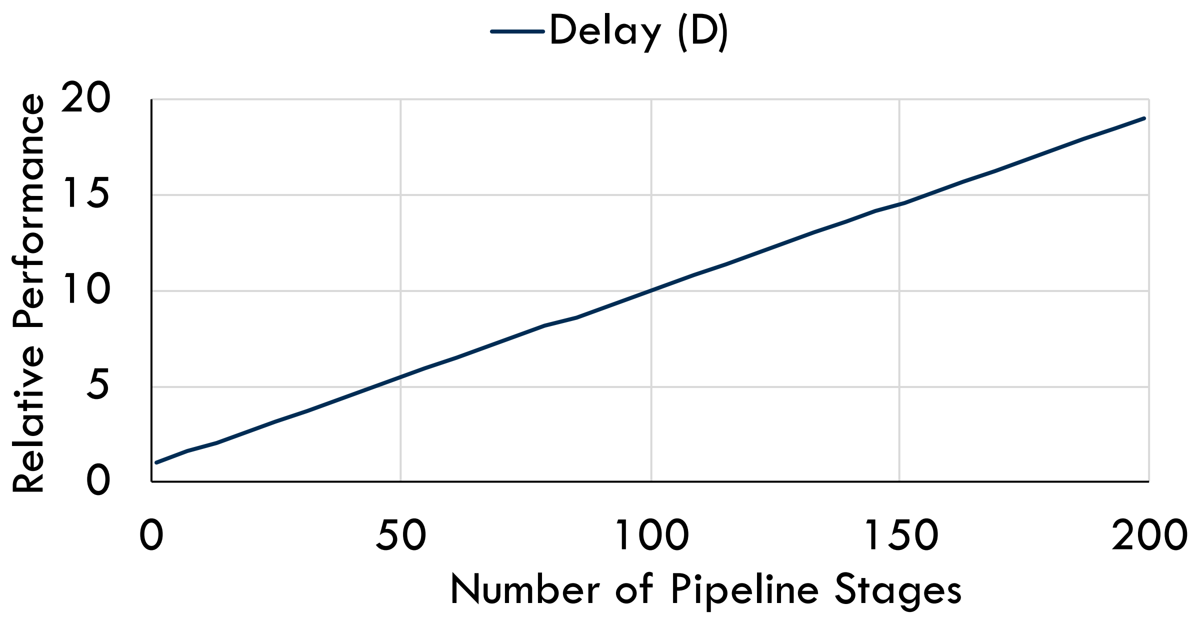
Pipelining Technique

- Improving throughput at the expense of latency
  - Delay: $D = T + n\delta$
  - Throughput: $IPS = n/(T + n\delta)$



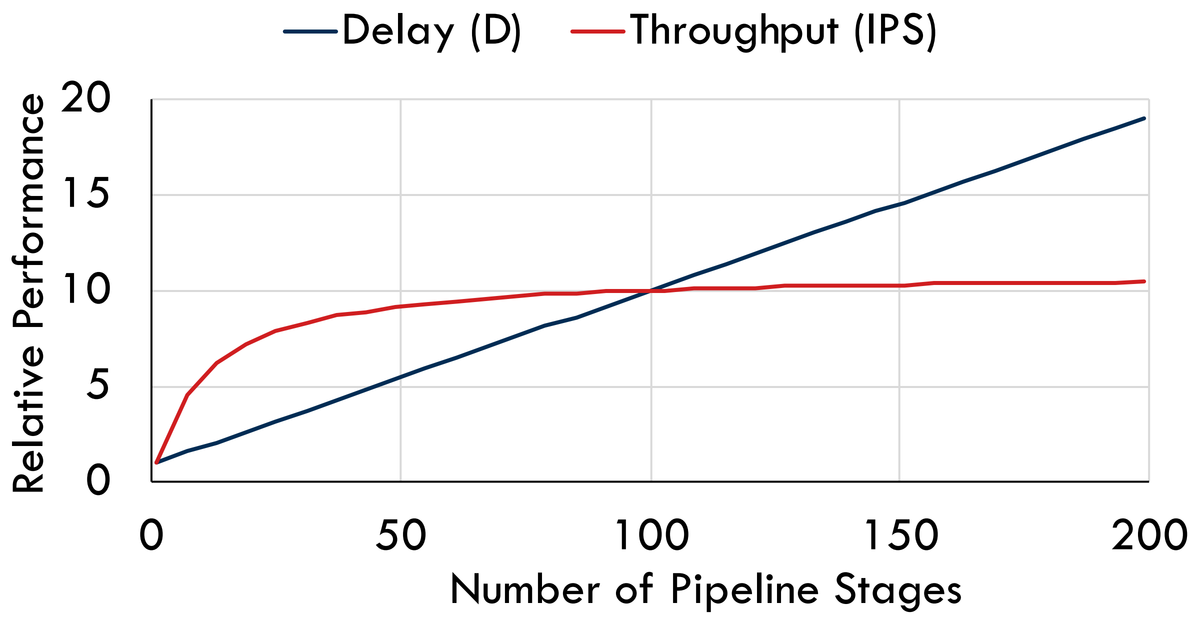
Pipelining Latency vs. Throughput

- Theoretical delay and throughput models for perfect pipelining



Pipelining Latency vs. Throughput

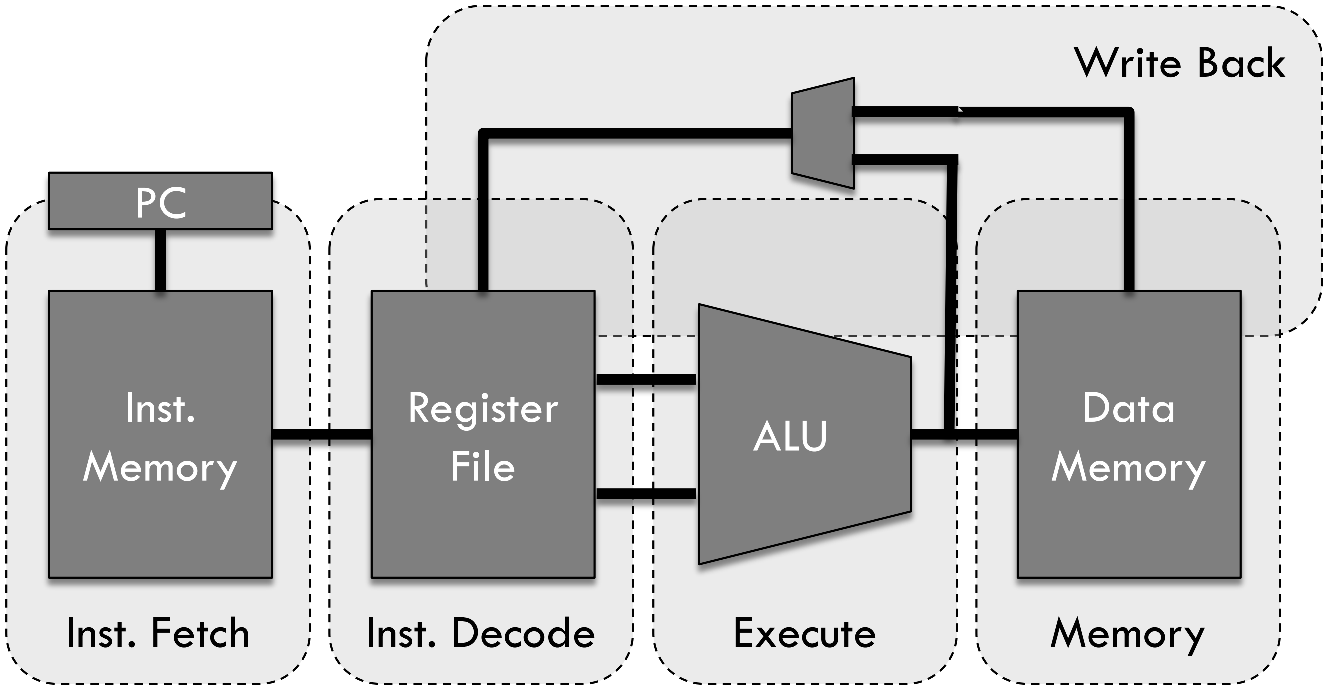
- Theoretical delay and throughput models for perfect pipelining



Five Stage MIPS Pipeline

Simple Five Stage Pipeline

- A pipelined load-store architecture that processes up to one instruction per cycle

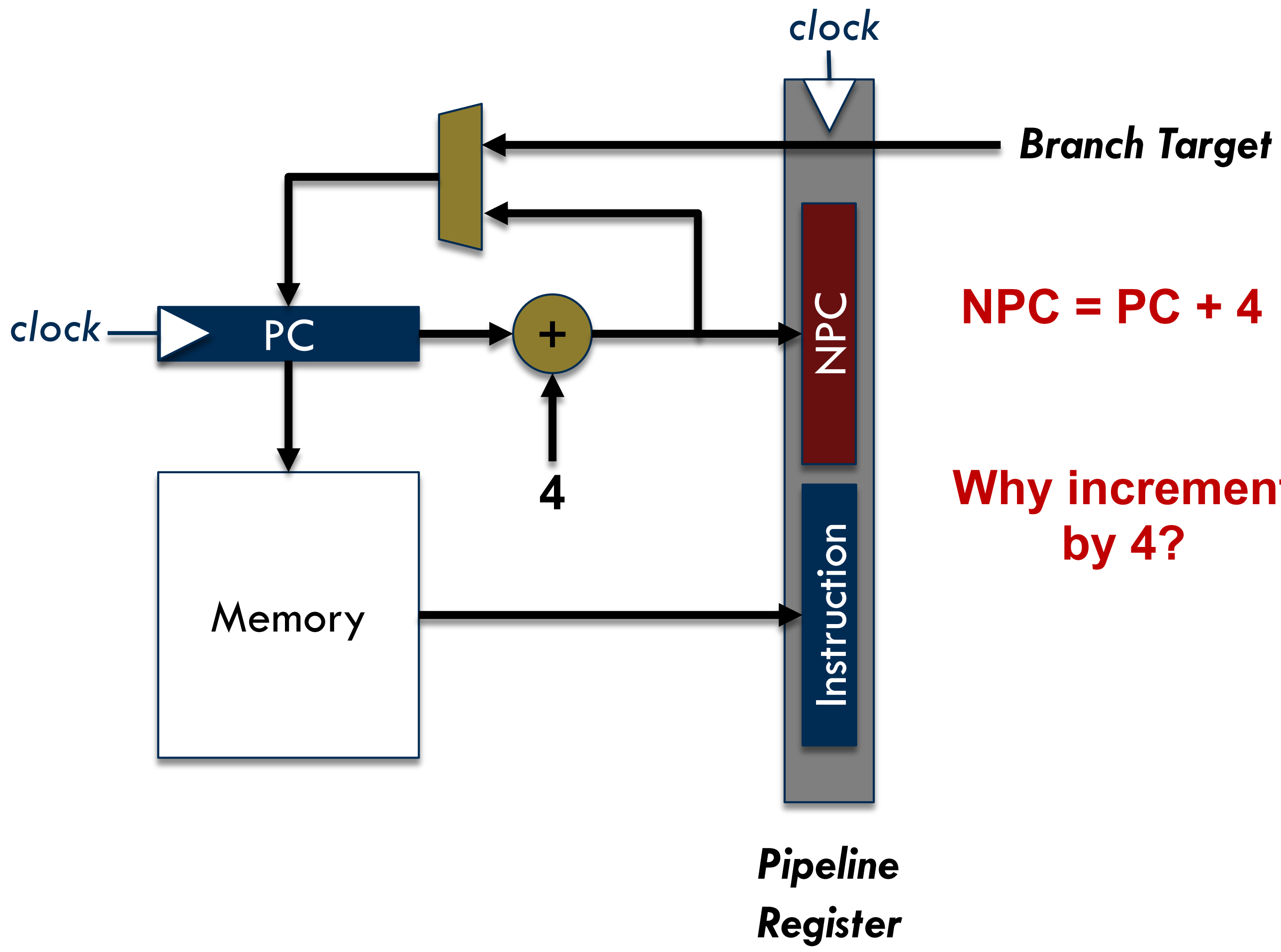


Instruction Fetch

- Read an instruction from memory (I-Memory)
  - Use the program counter (PC) to index into the I-Memory
  - Compute NPC by incrementing current PC
    - What about branches?

- Update pipeline registers
  - Write the instruction into the pipeline registers

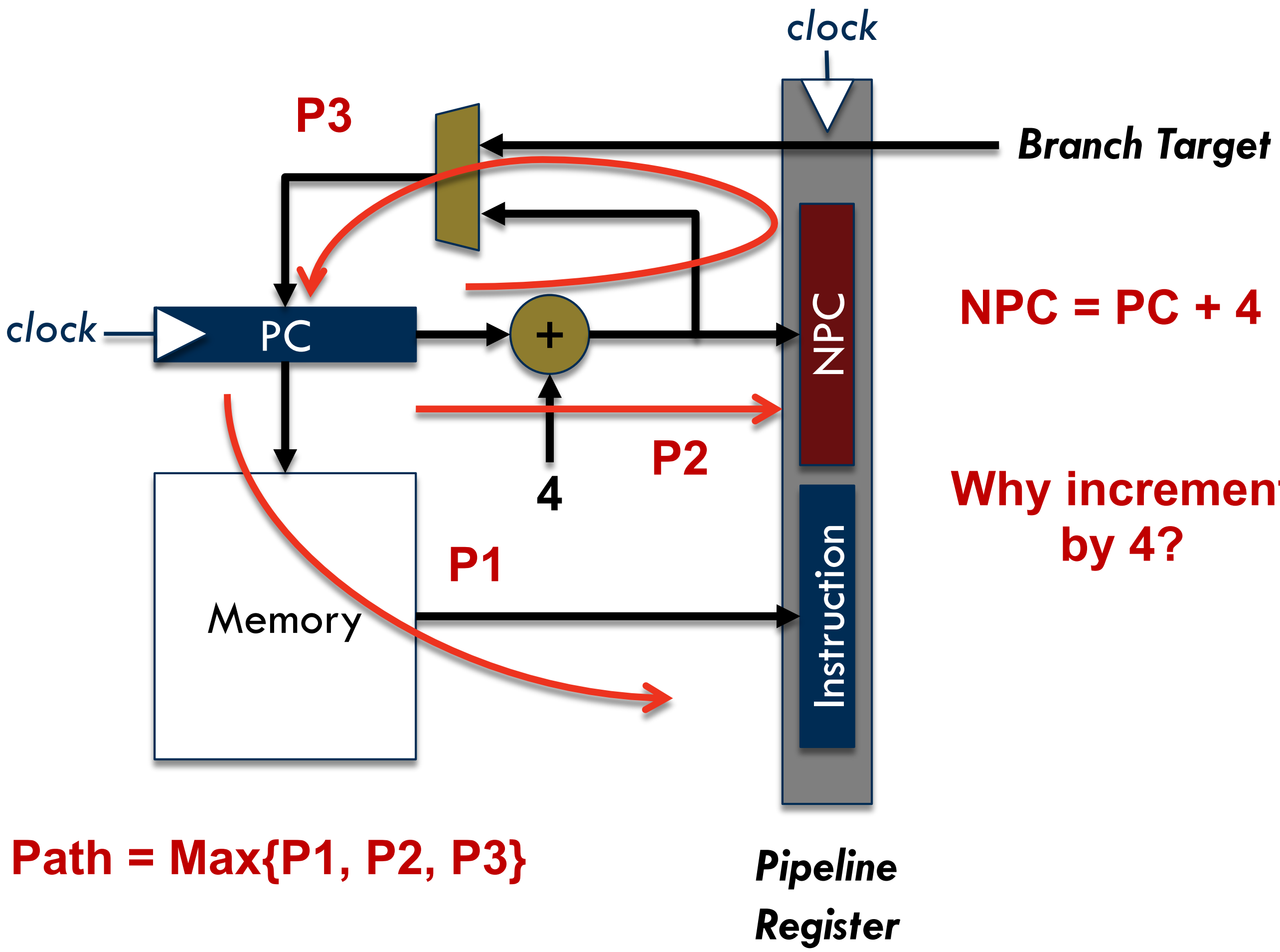
Instruction Fetch



NPC = PC + 4

Why increment by 4?

Instruction Fetch



NPC = PC + 4

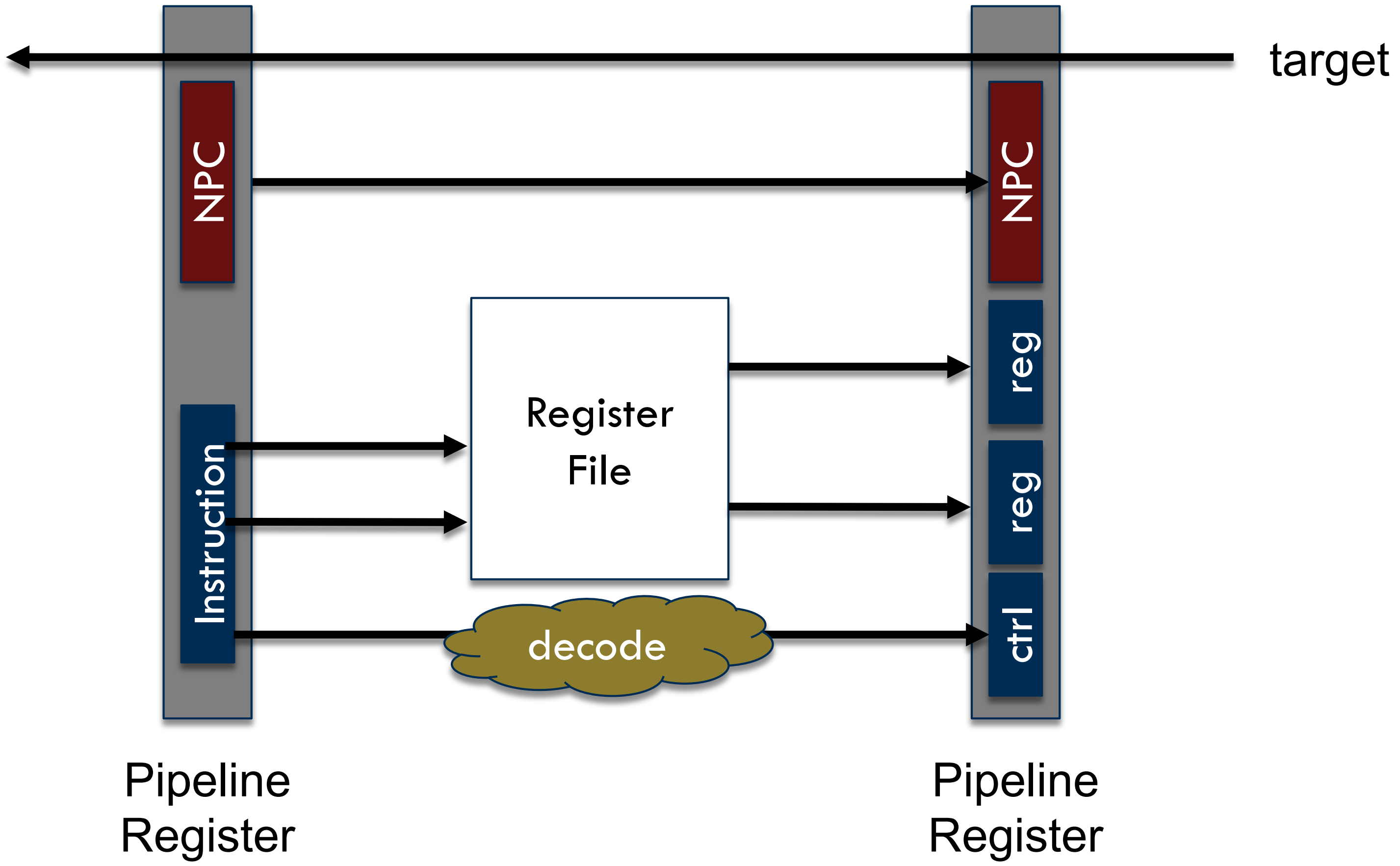
Why increment by 4?

Critical Path = Max{P1, P2, P3}

Instruction Decode

- Generate control signals for the opcode bits
- Read source operands from the register file (RF)
  - Use the specifiers for indexing RF
    - How many read ports are required?
- Update pipeline registers
  - Send the operand and immediate values to next stage
  - Pass control signals and NPC to next stage

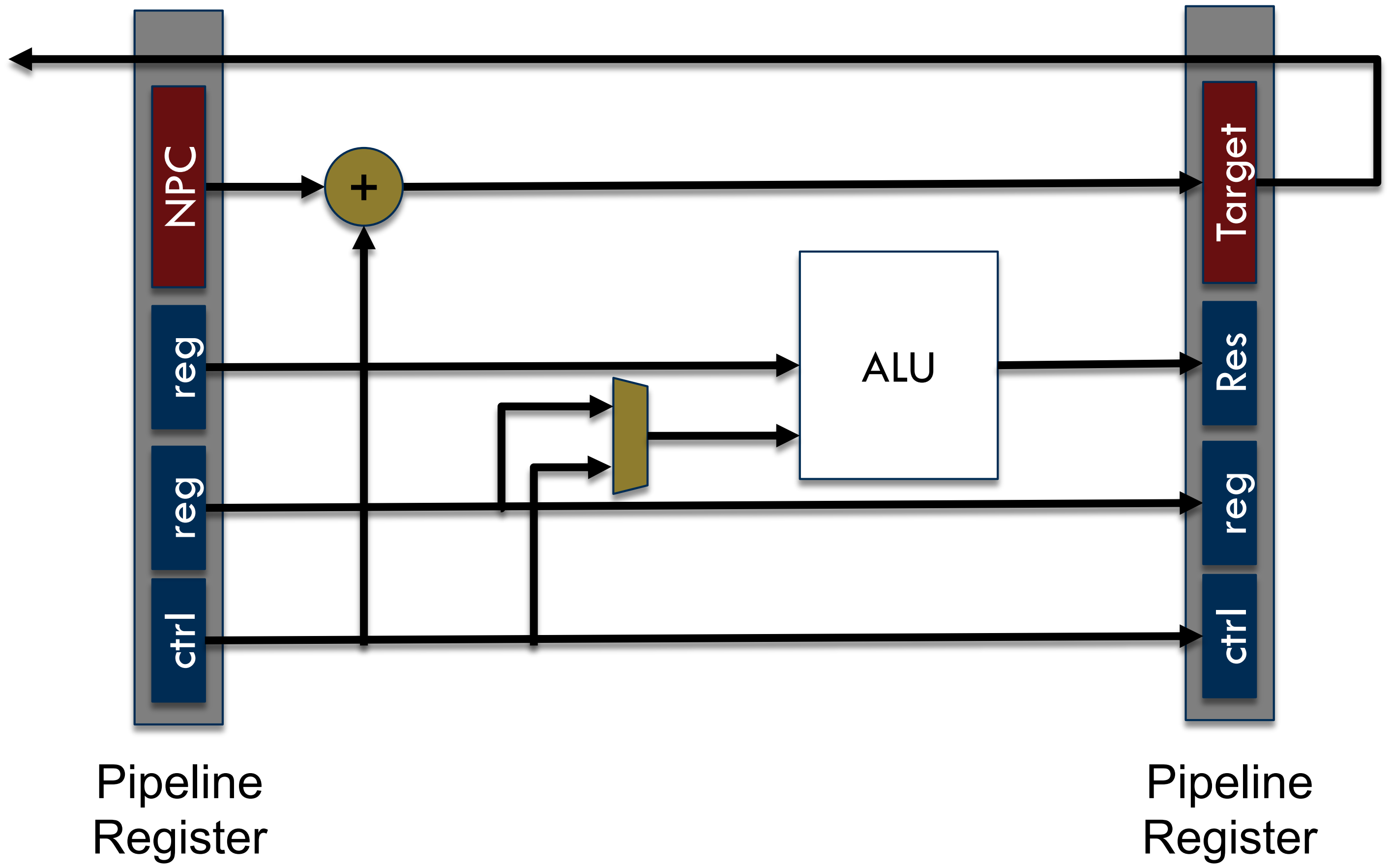
Instruction Decode



Execute Stage

- Perform ALU operation
  - Compute the result of ALU
    - Operation type: control signals
    - First operand: contents of a register
    - Second operand: either a register or the immediate value
  - Compute branch target
    - Target = NPC + immediate
- Update pipeline registers
  - Control signals, branch target, ALU results, and destination

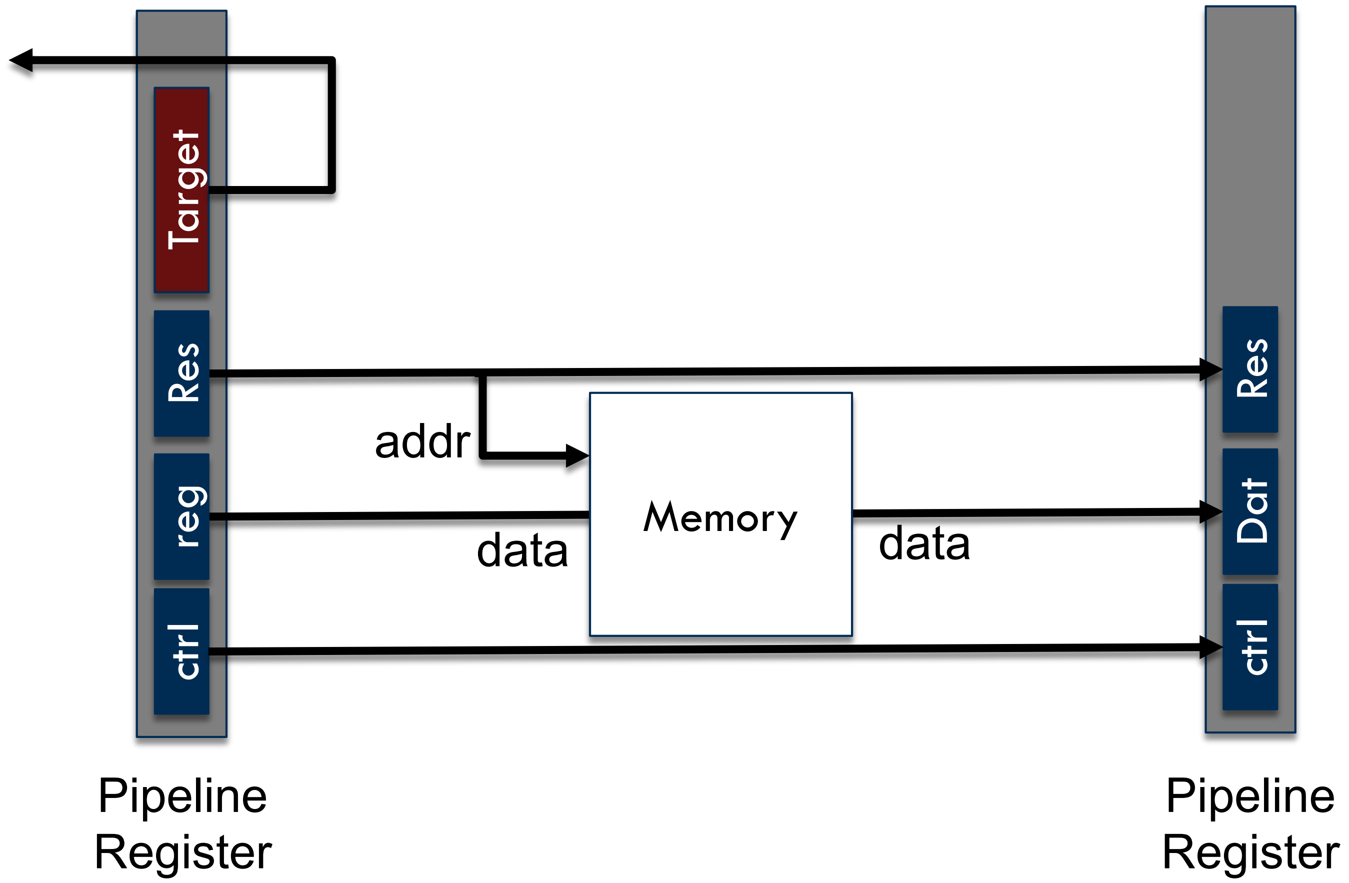
Execute Stage



Memory Access

- Access data memory
  - Load/store address: ALU outcome
  - Control signals determine read or write access
- Update pipeline registers
  - ALU results from execute
  - Loaded data from D-Memory
  - Destination register

Memory Access

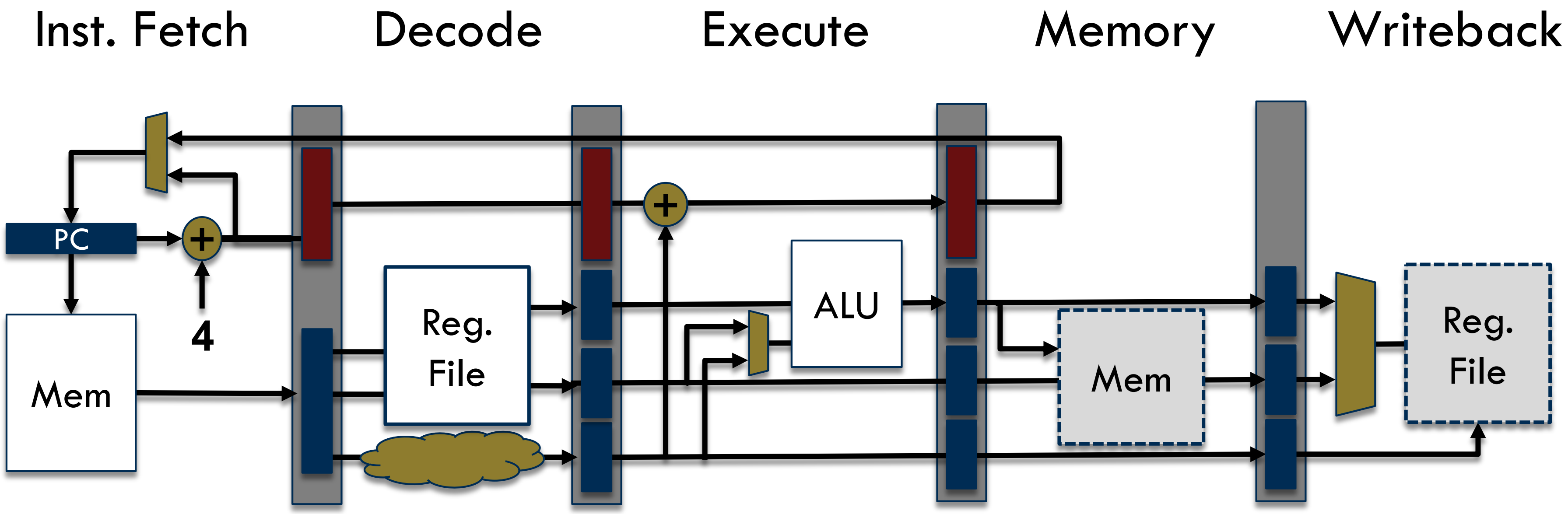


Register Write Back

- Update register file
  - Control signals determine if a register write is needed
  - Only one write port is required
    - Write the ALU result to the destination register, or
    - Write the loaded data into the register file

Five Stage Pipeline

- Ideal pipeline: IPC=1
  - Is there enough resources to keep the pipeline stages busy all the time?



Pipeline Hazards

Pipeline Hazards

- Structural hazards: multiple instructions compete for the same resource
- Data hazards: a dependent instruction cannot proceed because it needs a value that hasn't been produced
- Control hazards: the next instruction cannot be fetched because the outcome of an earlier branch is unknown

Structural Hazards

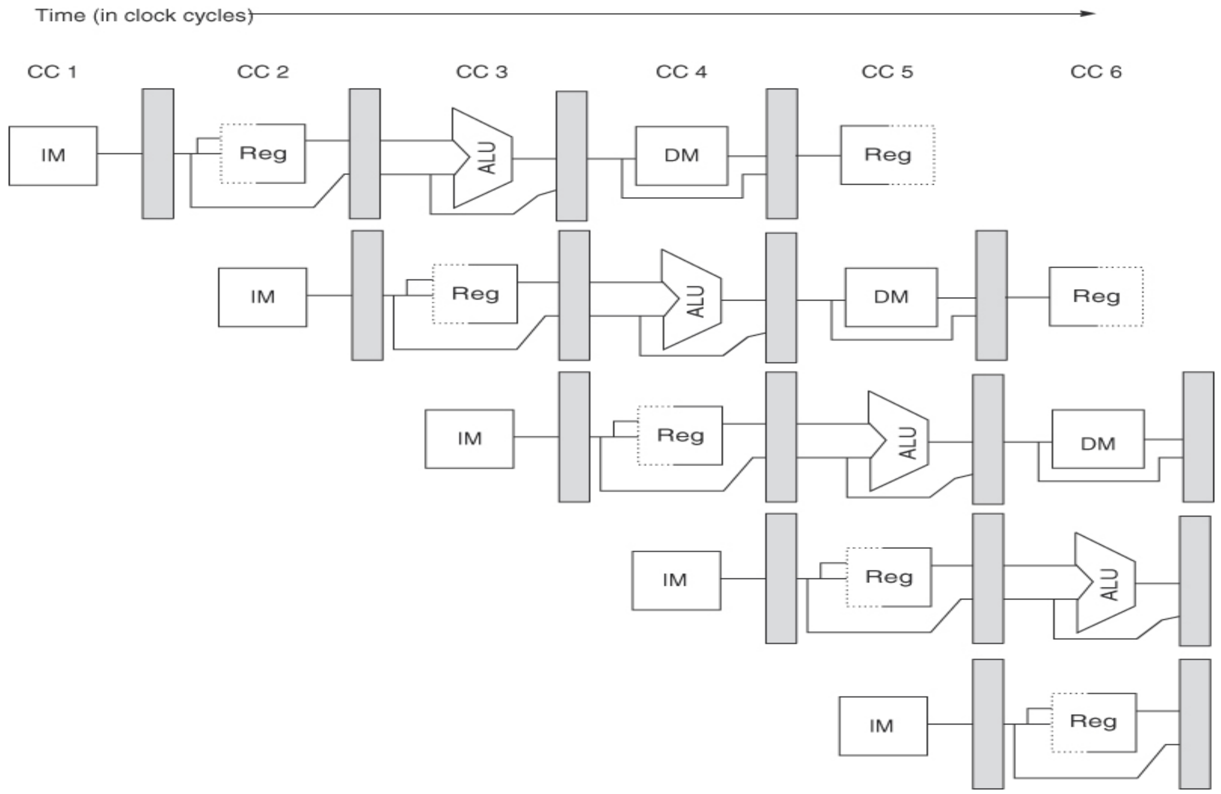
- 1. Unified memory for instruction and data

R1← Mem[R2]

R3← Mem[R20]

R6← R4-R5

R7← R1+R0



Structural Hazards

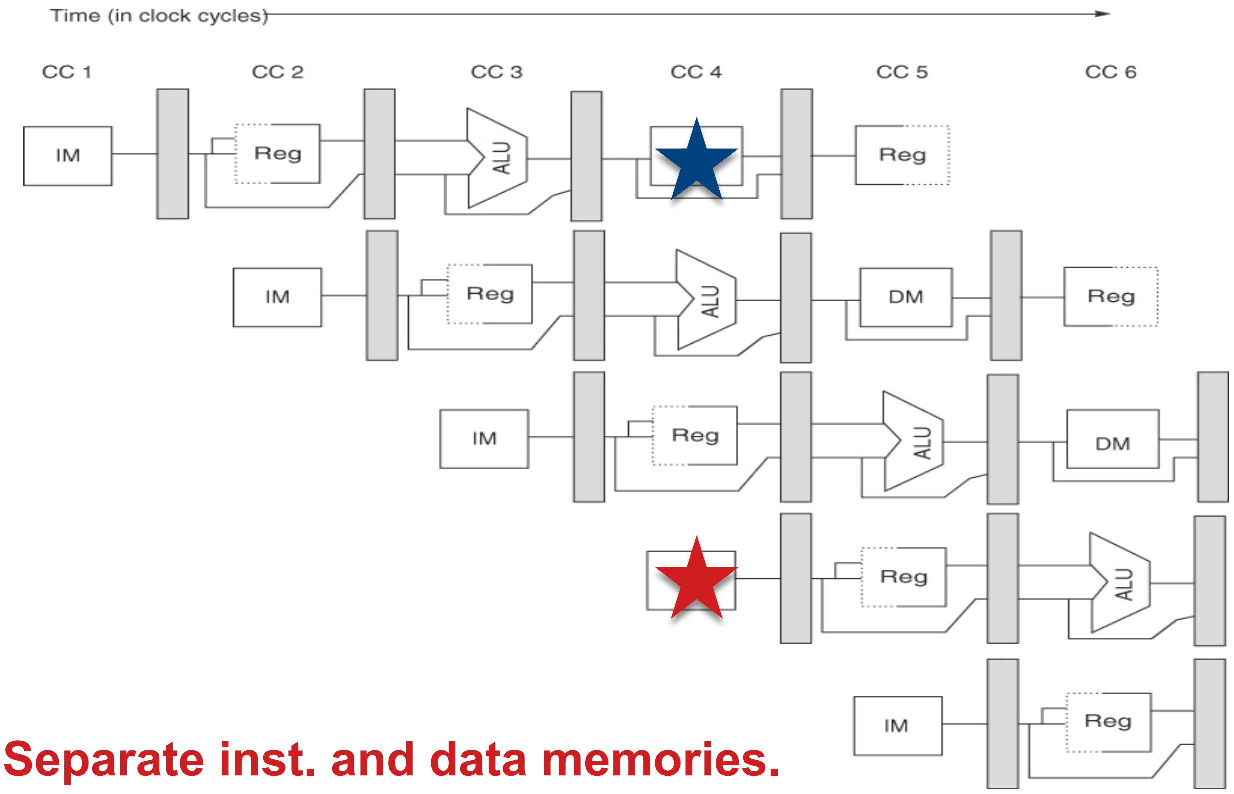
- 1. Unified memory for instruction and data

R1← Mem[R2]

R3← Mem[R20]

R6← R4-R5

R7← R1+R0



Separate inst. and data memories.

Structural Hazards

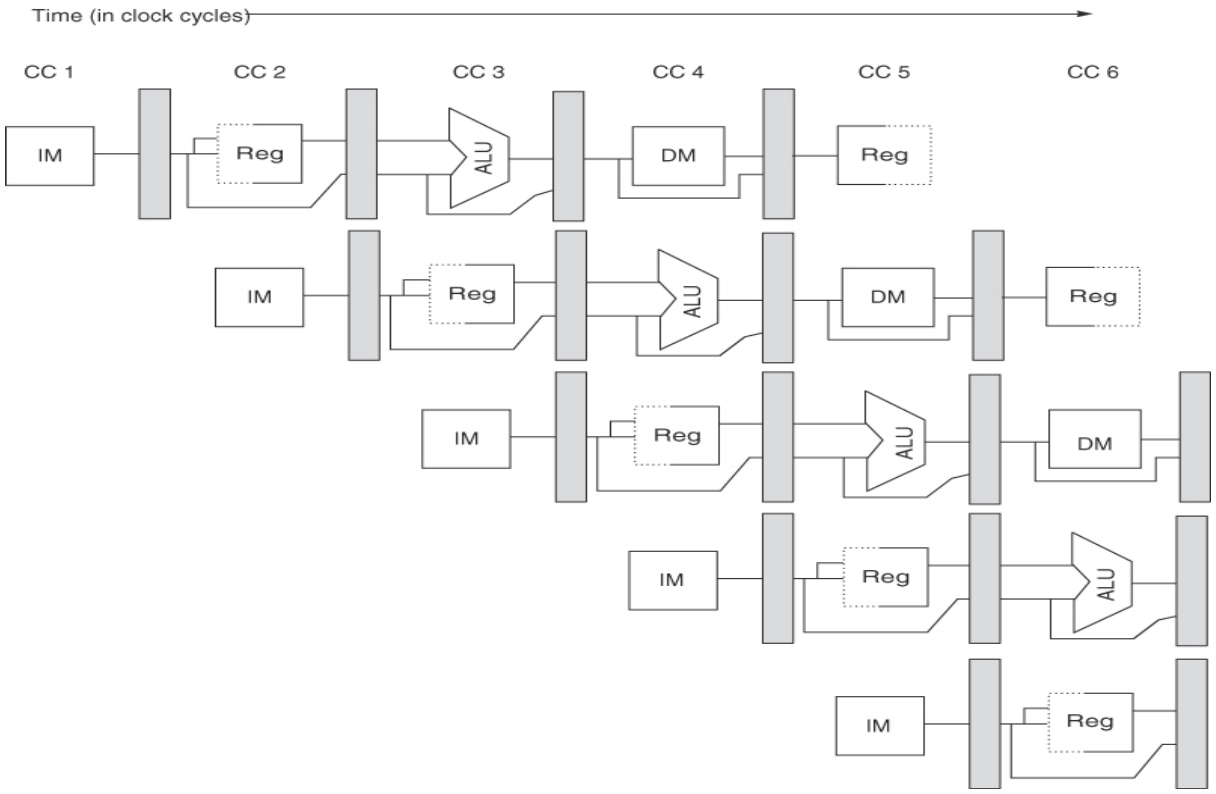
- 1. Unified memory for instruction and data
- 2. Register file with shared read/write access ports

R1← Mem[R2]

R3← Mem[R20]

R6← R4-R5

R7← R1+R0



Structural Hazards

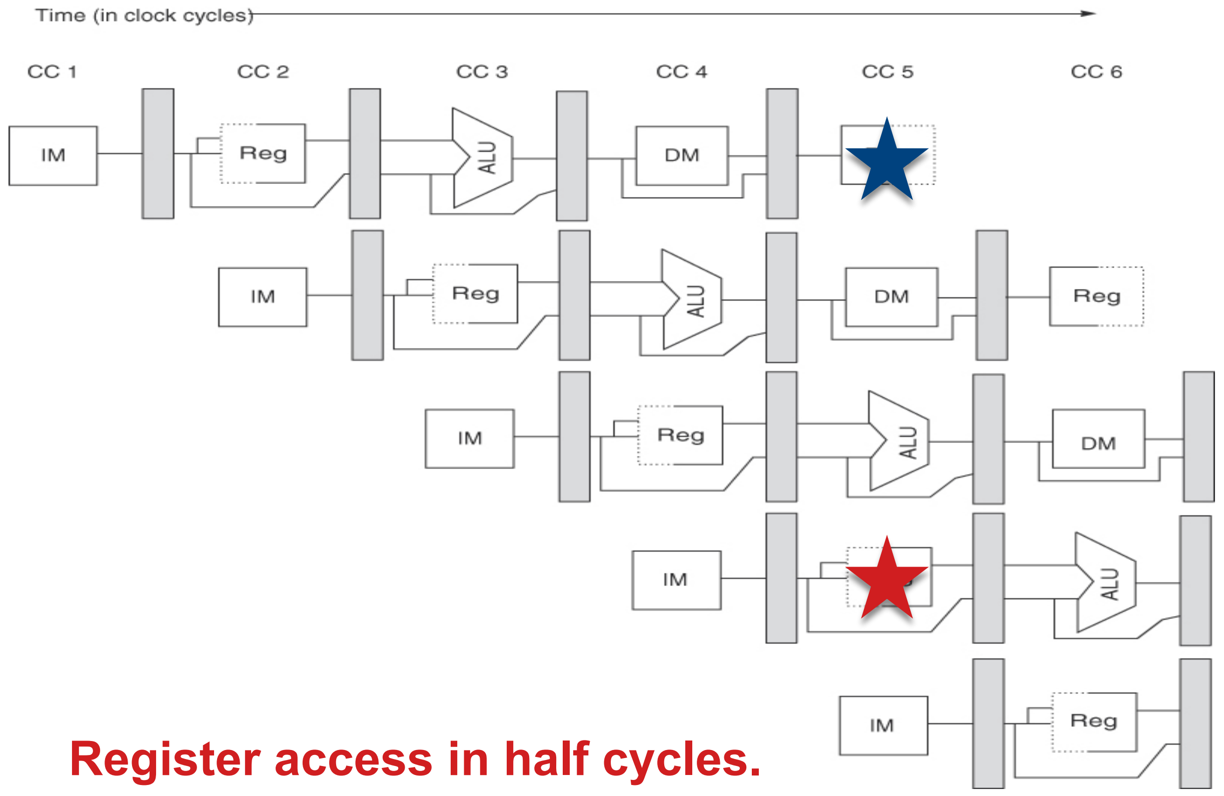
- 1. Unified memory for instruction and data
- 2. Register file with shared read/write access ports

R1← Mem[R2]

R3← Mem[R20]

R6← R4-R5

R7← R1+R0



Register access in half cycles.