

Building the Infinite Brain

COMP 590/790

Raghavendra Pradyumna Pothukuchi



THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

✉ raghav@cs.unc.edu

Quick Review

1-page write-up due now!
Project proposals due 9/3

What is computer architecture?

Historically, the ISA; but now encompasses organization

What are the goals?

Mnemonic: Simple Timely Efficient Adaptable Dependable Yummy

How to estimate impact of fixing bottlenecks?

Amdahl's law

How to fix bottlenecks?

Algorithms, adding a fast path

How to improve efficiency?

Technology, approximation, locality, regroup



For Today

- Quick review
- **Pipelining**

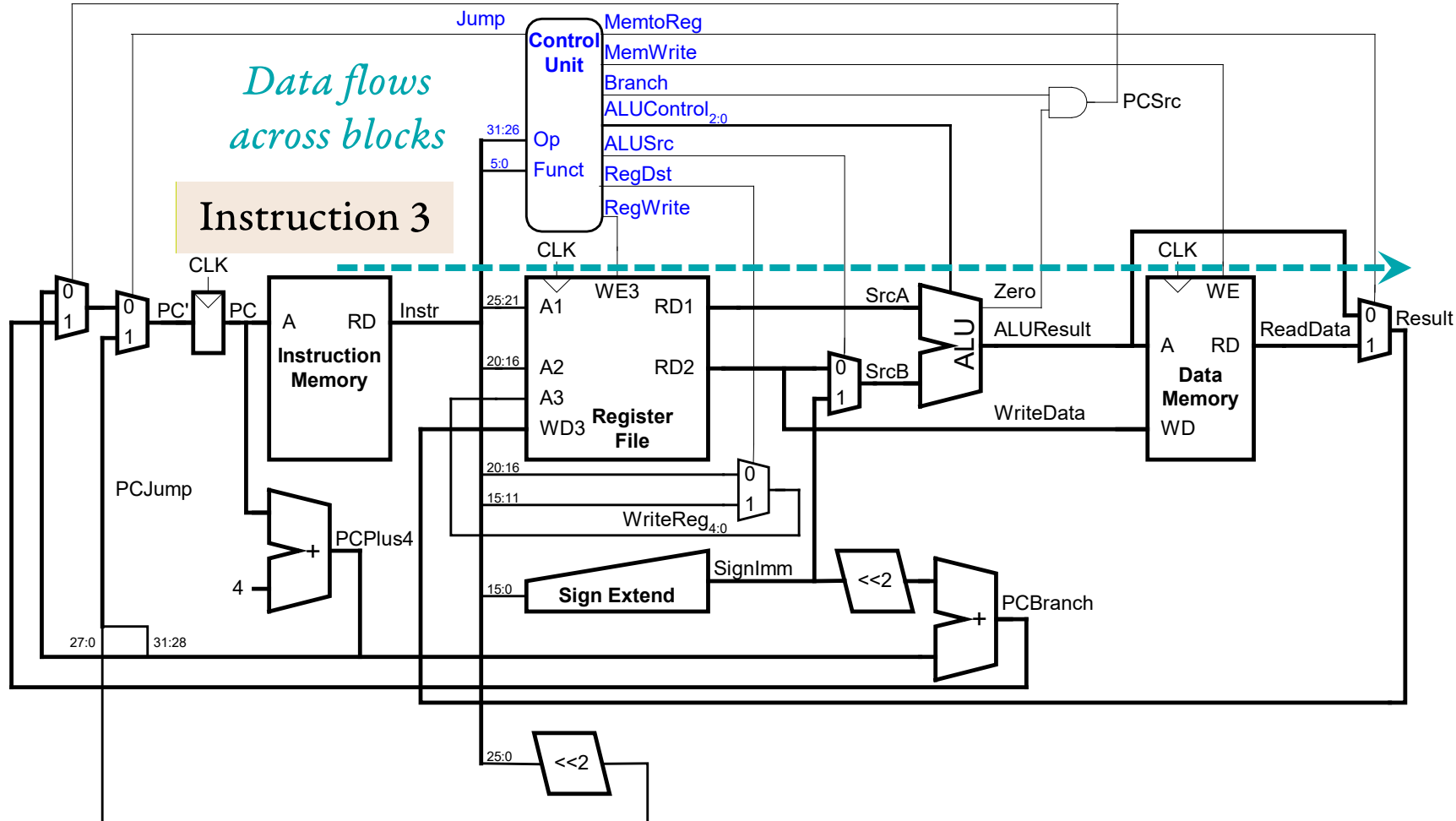


Motivation

Is all the hardware fully being used all the time, i.e., is there resource efficiency?

Latency: τ

Throughput: $\frac{1}{\tau}$



Fetch→Decode→Execute→Memory→Write-back

Regroup: Pipelining

$\tau_{\text{pipeline}} \neq \tau_{\text{original}}$

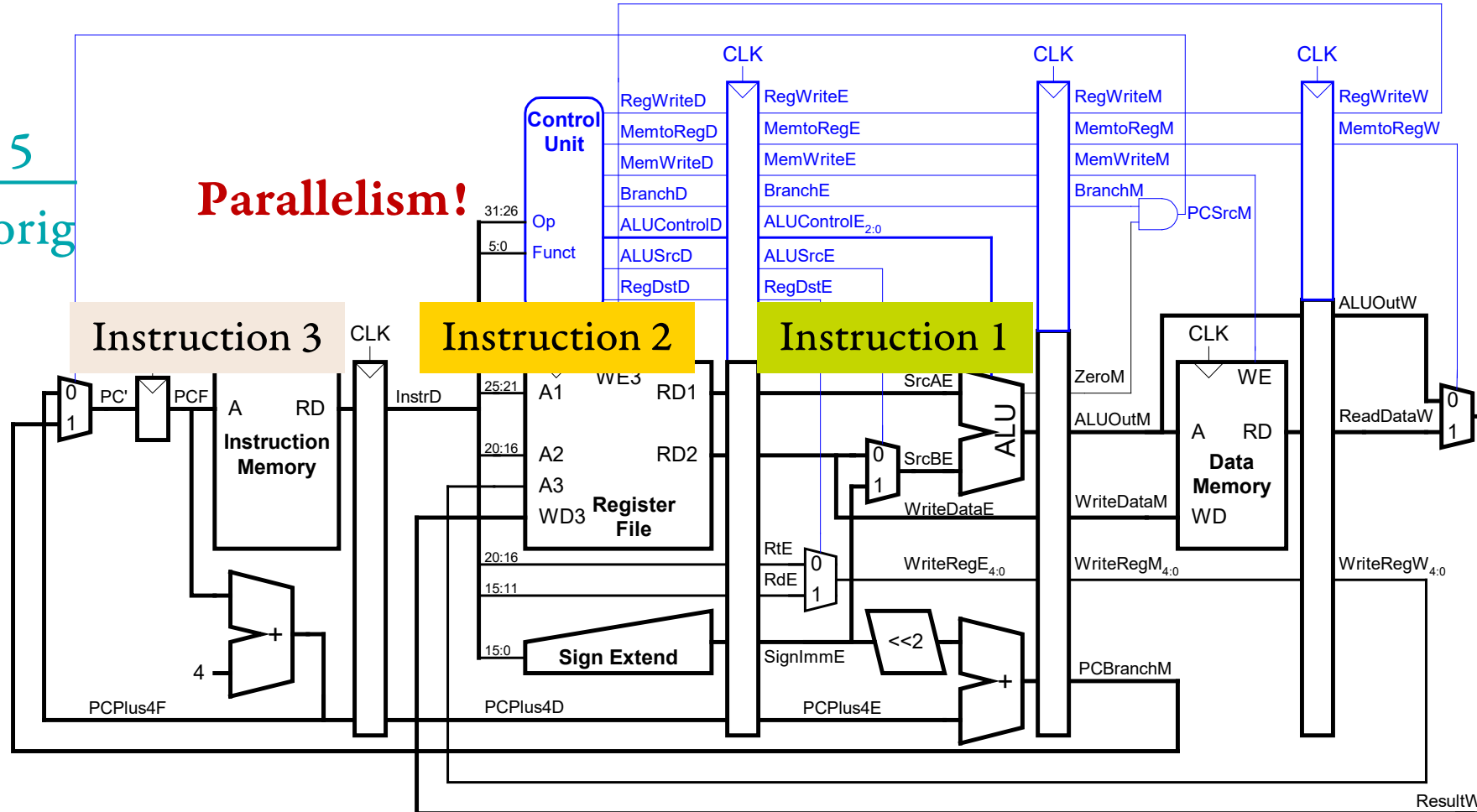


Divide the resource into stages

Latency: τ_{orig}

Throughput: $\frac{5}{\tau_{\text{orig}}}$

Parallelism!



Fetch (IF)

Decode (ID)

Execute (EX) Memory (MEM) Write-back (WB)



Pipelining: Assessing Goals

Improve throughput, utilization

Sacrifice latency, simplicity, power



Pipeline Performance

$$\sum \tau_i = \tau$$

$$\text{Latency: } n \times \max(\tau_i) \geq \tau$$

$$\text{Throughput: } \frac{1}{\max(\tau_i)} \leq \frac{n}{\tau}$$

“Optimal Pipelining in Supercomputers”, Kunkel and Smith, ISCA’86

“The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays”, Hrishikesh et al., ISCA’02



Exercise

$\tau_i: 8, 12, 10$ $\Delta_{\text{register}}: 2$

Latency, throughput, speedup?

$$\text{Execution time} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{time}}{\text{cycle}}$$



Pipeline Hazards

Structural

Multiple instructions need to use the same resource (execution unit, memory port etc.)

Data

Subsequent instructions need to wait for a prior instruction to complete

ADD R1, R2, R3
ADD R4, R5, R1

Control

Subsequent instructions need to be determined by a prior instruction

→BEQ R1, label
ADD R7, R6, R7

$$CPI_{\text{pipeline}} = CPI_{\text{ideal}} (=1) + \text{Stall CPI}$$

$$\text{Pipeline speedup} = \frac{\text{Time}_{\text{original}}}{\text{Time}_{\text{pipeline}}} = \frac{CPI_{\text{original}} \times \text{Period}_{\text{original}}}{CPI_{\text{pipeline}} \times \text{Period}_{\text{pipeline}}} = \frac{\text{Pipeline depth}}{1 + \text{Stall CPI}}$$



Exercise: Examining Throughput With Stalls

Total gate delay per instruction: **D**

Overhead per stage: **Δ**

Average stalls per instruction per stage: **S**

Number of stages : **n**

$$\text{CPI}_{\text{pipeline}} = 1 + S \times n$$

$$\text{Period} = \frac{D}{n} + \Delta$$

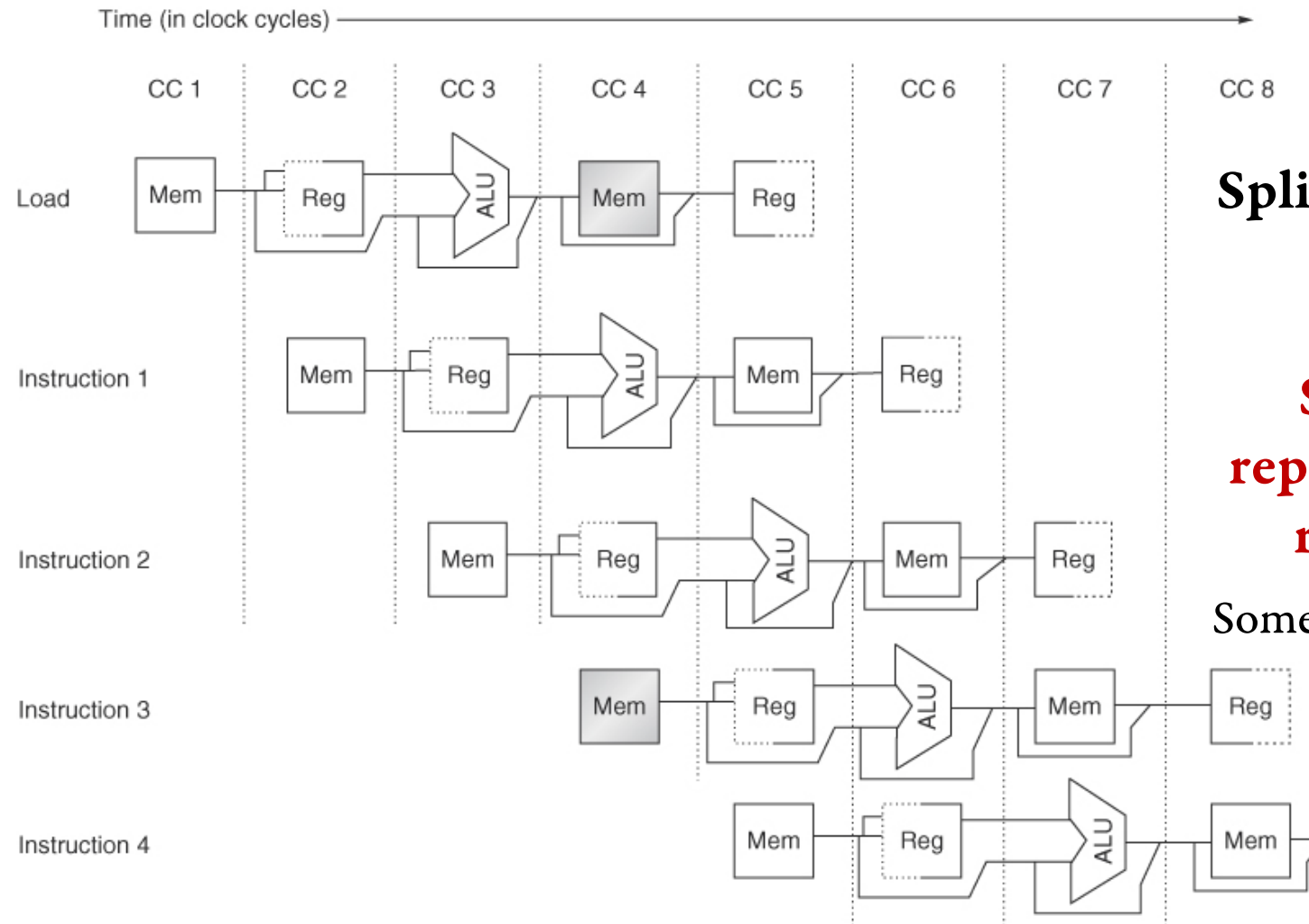
$$\text{Throughput: } \frac{\text{Instructions}}{\text{second}} = \text{IPC} \times \text{Frequency}$$

For some choices of D (about 100), Δ (about 1), S (about 0.05–0.2), plot throughput vs n



Pipeline Hazards: Structural

Recall: “Independent” “Divide and conquer”



Split I and D caches!

**Split, pipeline,
replicate, streamline
resource access**

Some instructions may take
multiple cycles

Pipeline Hazards: Data

Read after a Write (RAW) – data dependence

ADD R1 \leftarrow R1, 1

ADD R2 \leftarrow R1, R3

Write after a Write (WAW) – output dependence

Write after a Read (WAR) – anti dependence

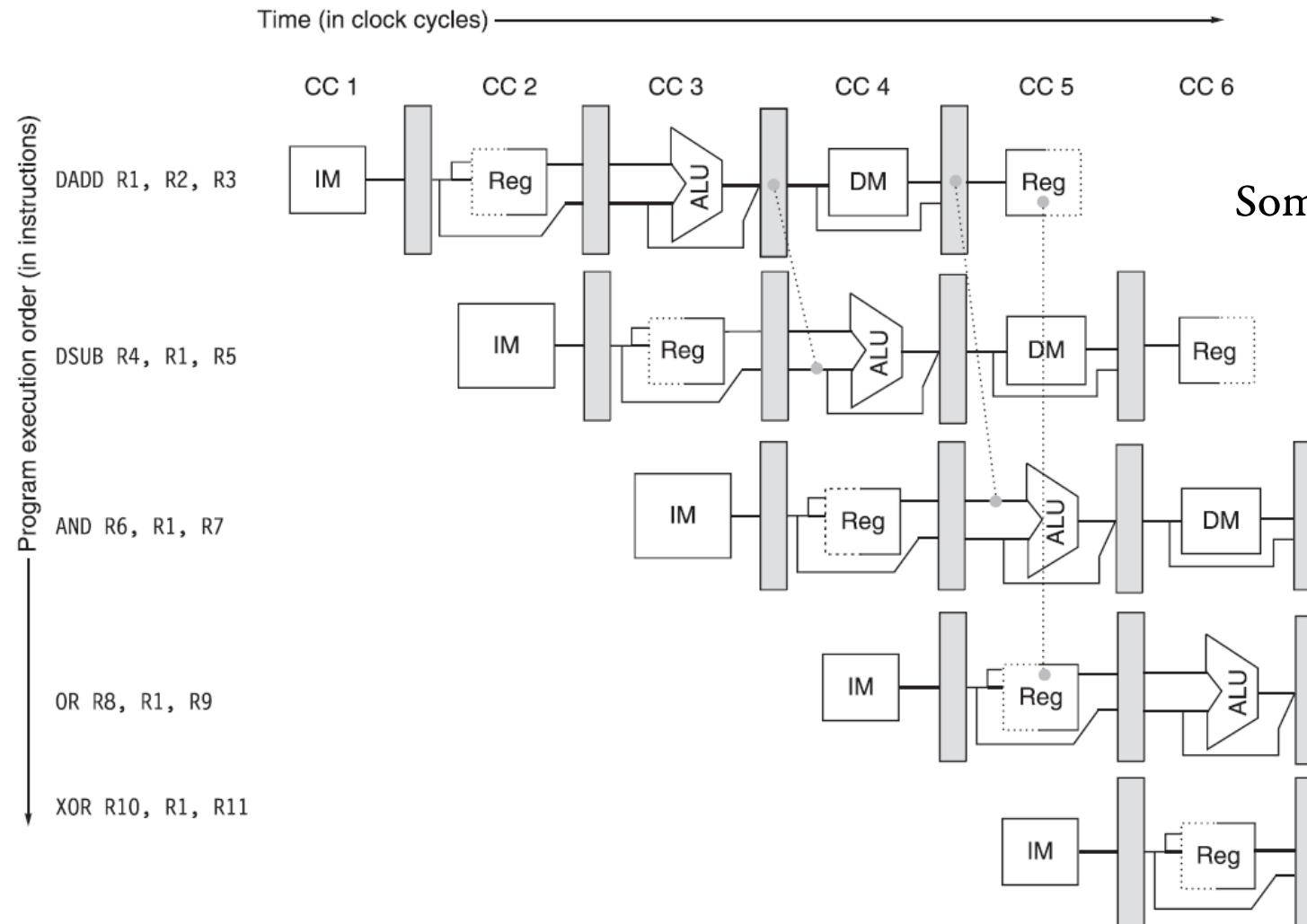
When do these matter?



Data Hazard Example

How to solve?

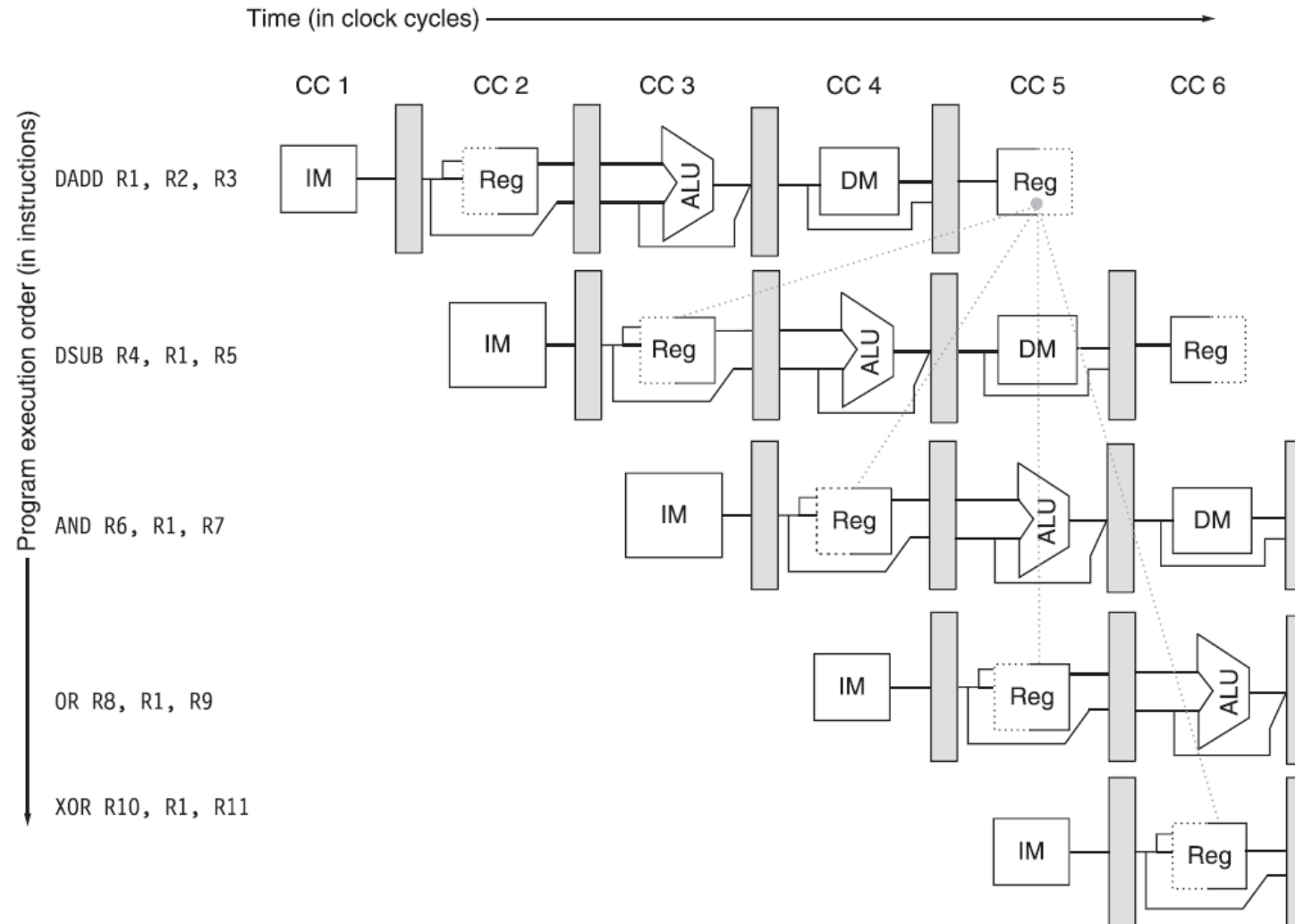
“Eager”: Bypassing/forwarding/“short circuiting”



Some instructions may take multiple cycles

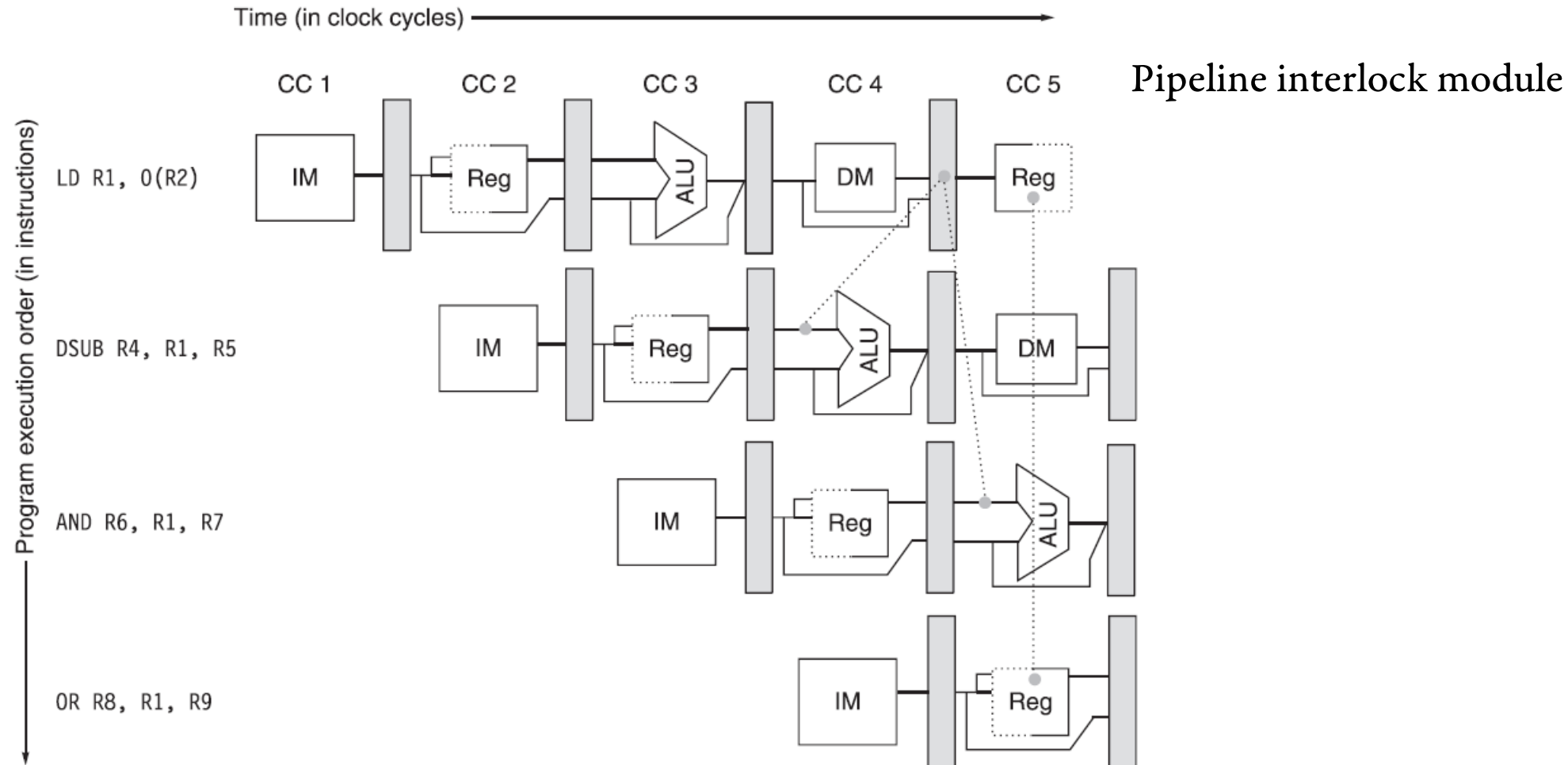
Data Hazard Example

How to solve?



What About This Example?

Unavoidable stall!



Another solution when feasible: compiler reordering

Pipeline Hazards: Control

Branch instruction	IF	ID	EX	MEM	WB		
Branch successor		IF	IF	ID	EX	MEM	WB
Branch successor + 1				IF	ID	EX	MEM
Branch successor + 2					IF	ID	EX

Branch stall costs: Branch frequency \times Branch penalty

How to solve?

“Eager”: Delay slot (make use of the next instruction that gets fetched)

“Speculate”: Branch prediction (check if PC is PC+1 or the branch target)

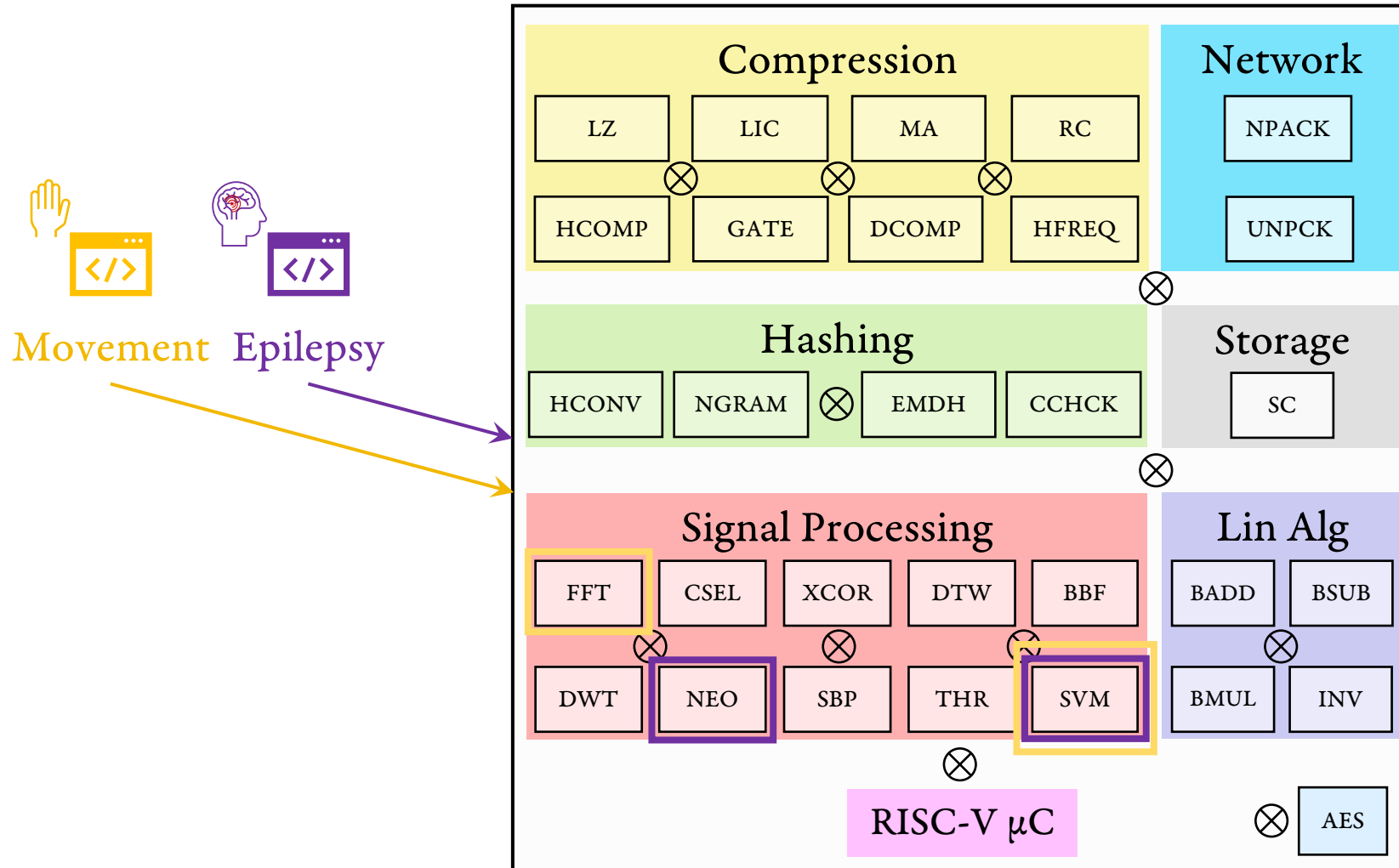
2-bit predictors, tournament predictors, machine learning etc.

<https://ericrotenberg.wordpress.ncsu.edu/cbp2025-workshop-program/>



Example from a BCI Processor (SCALO)

A different kind of pipeline: Dataflow—work flows through a directed acyclic graph (DAG)



No mitigation for any hazards!

Software managed pipelining

Simplicity vs Efficiency

Pipelining was conceived to meet hard design goals—**IBM Stretch, ILLIAC etc.**
BCI processing creates a similar need!

Takeaways

2-page project proposals due 9/3

What is pipelining?

Splitting work into many components executed independently, and passed in a chain

Why pipelining?

Increases efficiency via parallelism

What are the challenges in pipelining?

Hazards: structural, data, control

How to fix hazards?

Concurrency (structural), eager (forwarding data), eager or speculative (branch delays, prediction)

What systems design principles does pipelining touch?

Tradeoff simplicity for efficiency, leveraging parallelism through regrouping



Image Credits (Educational, Fair Use)

- Title image: VLADGRIN, https://www.istockphoto.com/vector/human_-machine-gm147409511-16840728 (Educational fair use)
- Infinite brain: Science wonder stories, May 1930, Illustrator: Frank R Paul, Editor: Hugo Gernsback
- Brain color, ICs, cloud server, black rat: No attribution required (Hiclipart)
- Hand with spoon: public domain freepng
- Signals: <https://www.nature.com/articles/nrn3724>
- Thought clouds: F. Willett et al./*Nature* 2021/Erika Woodrum, <https://med.stanford.edu/neurosurgery/news/2022/bci-award>. <https://www.the-scientist.com/news-opinion/brain-computer-interface-user-types-90-characters-per-minute-with-mind-68762>
- Picture of scientists: <https://www.cs.auckland.ac.nz/~brian/rutherford8.html> (original: Pierre de Latil), Bush (Carnegie Science), Others (Wikipedia and National Academies)
- Flowchart: Pause08 – flaticon.com
- Digital brain: Smashicons – flaticon.com
- Server rack: upklyak – freepik.com
- Quantum processor icons created by Paul J. - Flaticon
- Arm, Lotus: Adobe stock
- Quantum processor: Rigetti computing
- Images of implanted users: Top: Case Western Reserve University (<https://thedaily.case.edu/man-quadruplegia-employs-injury-bridging-technologies-move-just-thinking/>), Bottom: Jan Scheuermann (University of Pittsburgh/UPMC; <https://www.upmc.com/media/news/bci-press-release-chocolate>)
- Images of wearable BCIs: Cognixion, NextMind
- Types of BCIs: “Brain–computer interfaces for communication and rehabilitation,
- Illustrative BCI: Neuralink
- Electrodes: “Electrochemical and electrophysiological considerations for clinical high channel count neural interfaces”, Vatsyayan et al.
- Form factors: Neuropace, Medtronic, Bloomberg, “Fully Implanted Brain–Computer Interface in a Locked-In Patient with ALS” by Vansteensel et al., Blackrock Neurotech
- Jose Delgado’s video: Online, various sources (CNN, Youtube)
- Video of Kennedy and Ramsey: Online, various sources (Youtube, Neural signals)

Logos, trademarks are all properties of respective owners

Not to be shared outside the course

