Computer Systems Architecture

Spring 2016

Lecture 01: Introduction

Shuai Wang

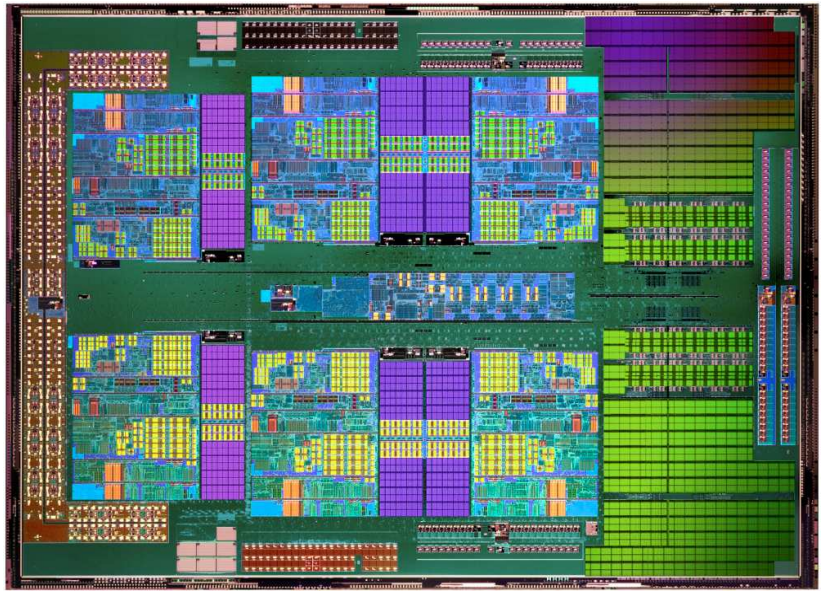
Department of Computer Science and Technology

Nanjing University

[Adapted from Computer Architecture: A Quantitative Approach, J. L. Hennessy and D. A. Patterson, 5rd Edition]

What is Computer Architecture?

What is Computer Architecture?



Computer Architecture

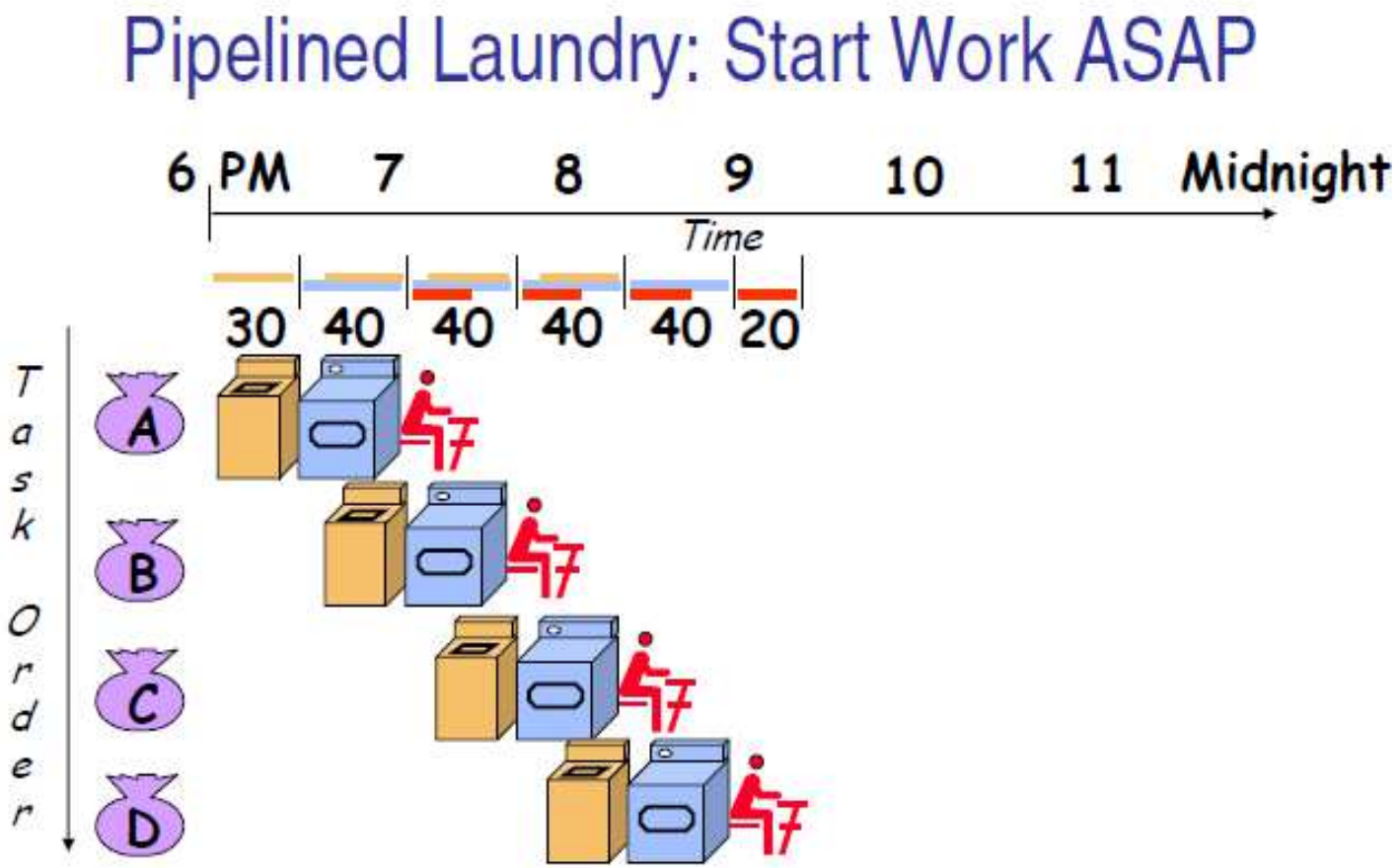
- Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints.

---Frederick P.Brooks Jr, *Planning a Computer System: Project Stretch*, 1962

Critical Issues in Computer Architecture

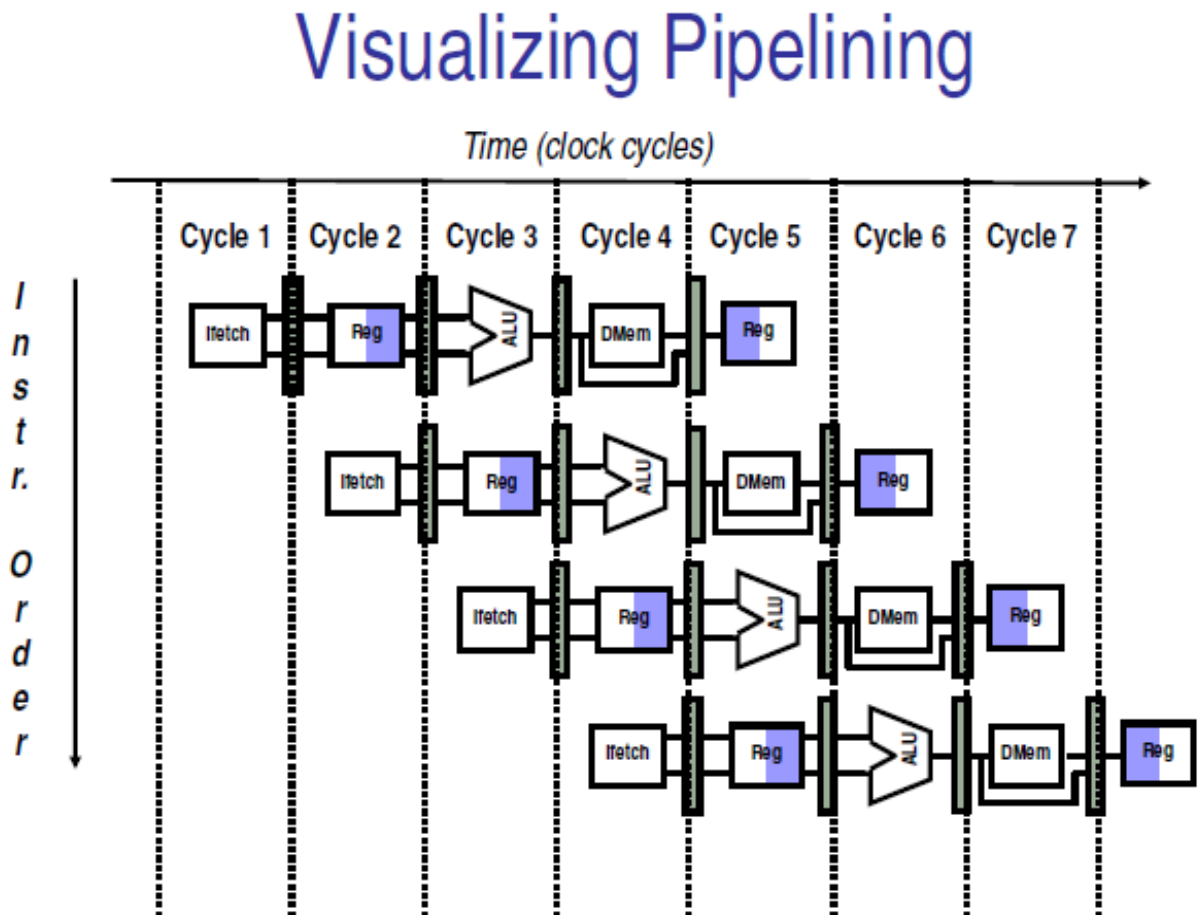
- Pipeline
- Out-of-Order (OoO) Execution
- Superscalar
- Multithreading and Multiprocessor
- Caches
- Memories
- Power
- Reliability
- ...etc.

Pipeline



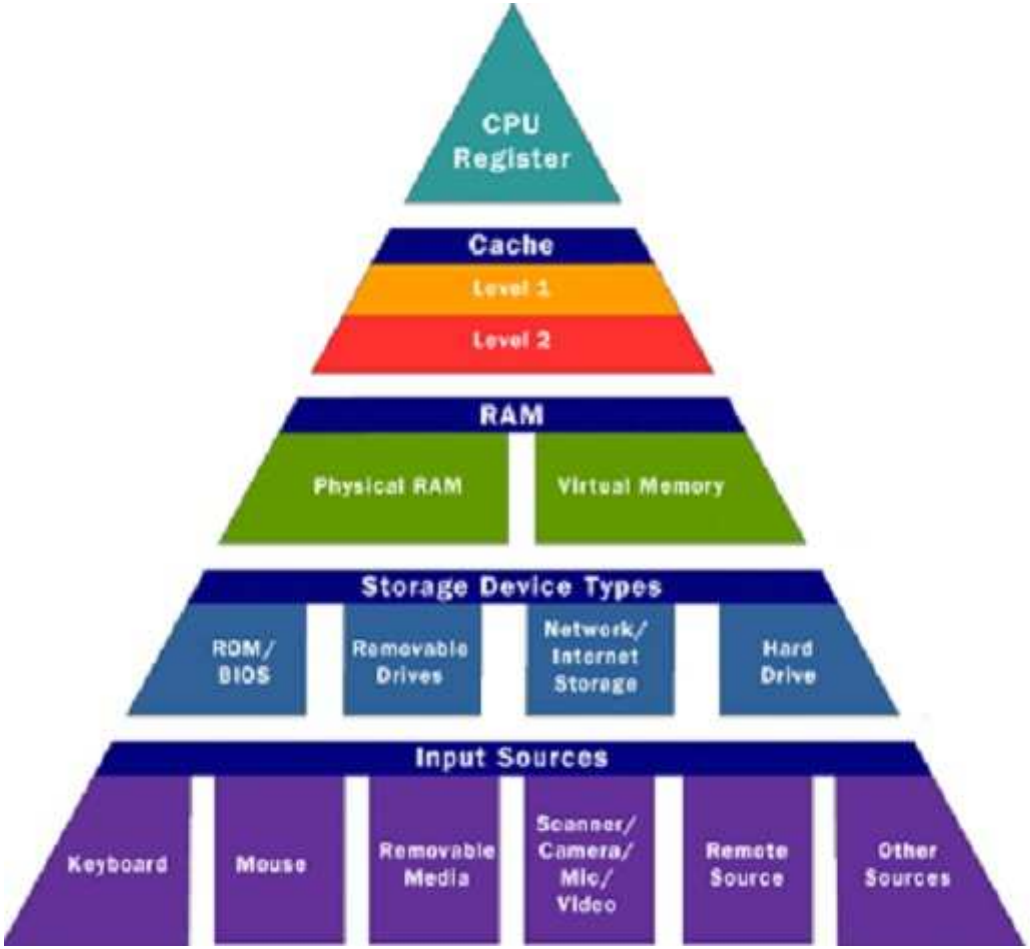
- Pipelined laundry takes 3.5 hours for 4 loads

Pipeline


Visualizing Pipelining
Time (clock cycles)
Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Cycle 6
Cycle 7
Instr. Order
Ifetch
Reg
ALU
DMem
Reg

Caches

Speed Increasing



Cost Increasing

Caches

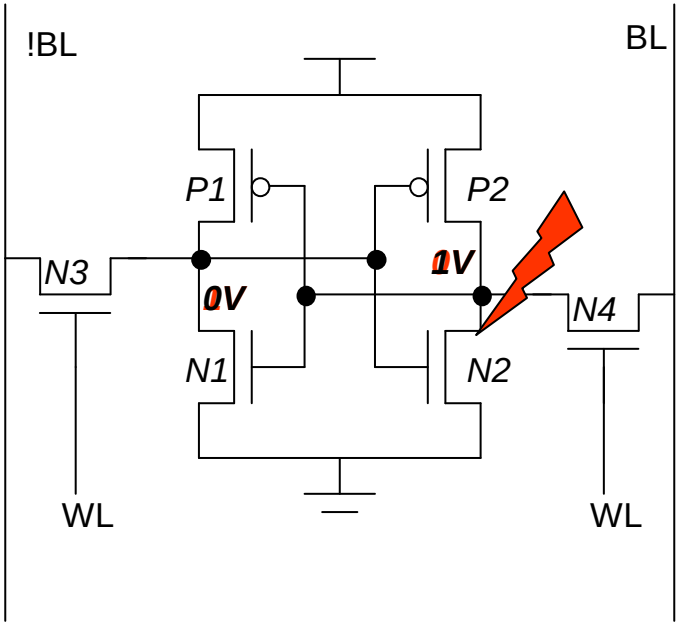
- Why do we need caches?
  - Latency vs. Cost
- Why do caches work?
  - Principle of Locality
    - Temporal locality
    - Spatial locality

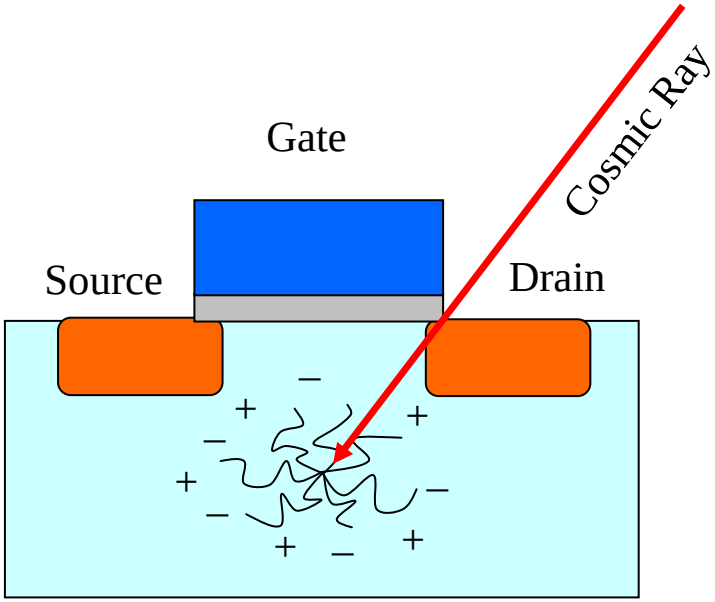
Hot Areas

- Performance
  - Moore's Law
  - Multi-core / Many-core Processor
- Power
  - Voltage/Frequency
  - Thermal
- Reliability
  - Manufacture
  - Aging
  - Soft Errors

Soft Errors

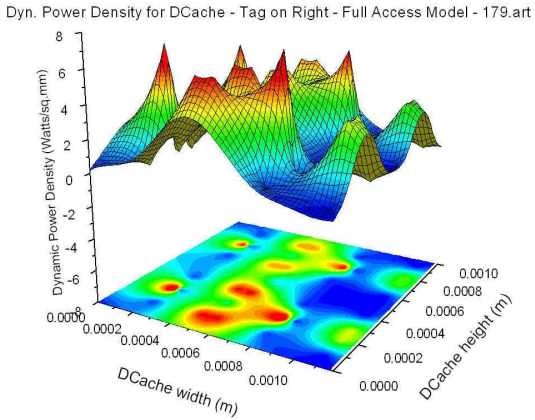
- Reliability in Multi-/Many- core Processors (Register files, Caches, and Memories).

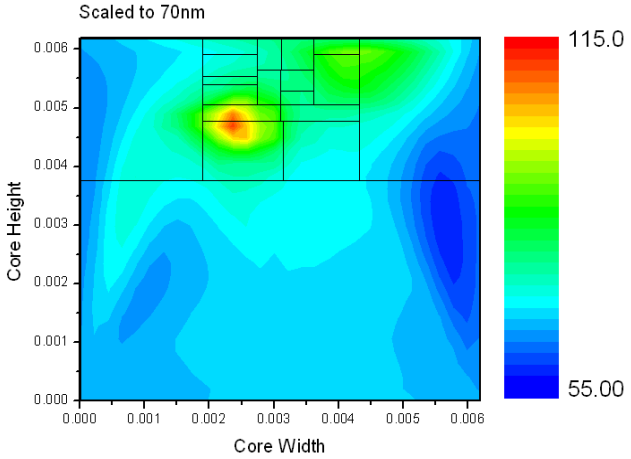




Power/Thermal Aware Design

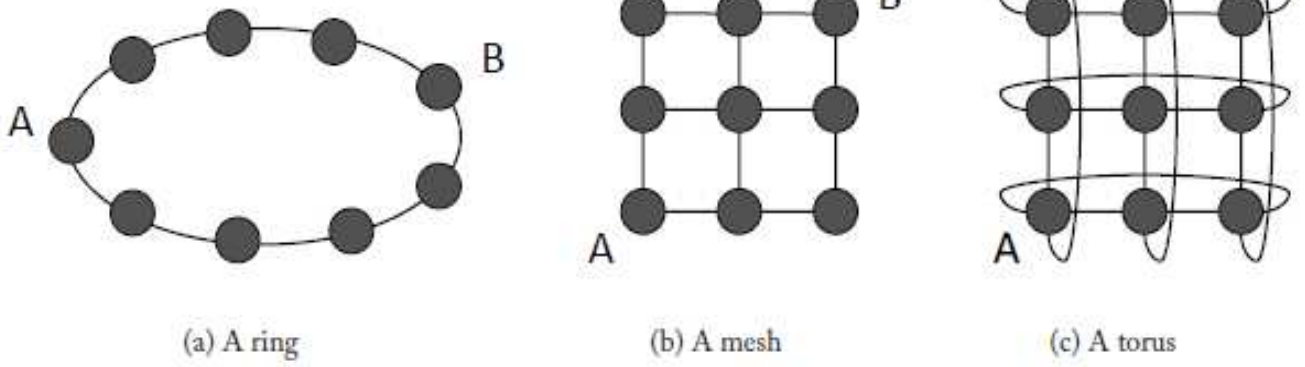
- Low power design
  - Especially in Embedded Systems
- Thermal-aware design
  - Temperature Control





Networks on Chip (NoC)

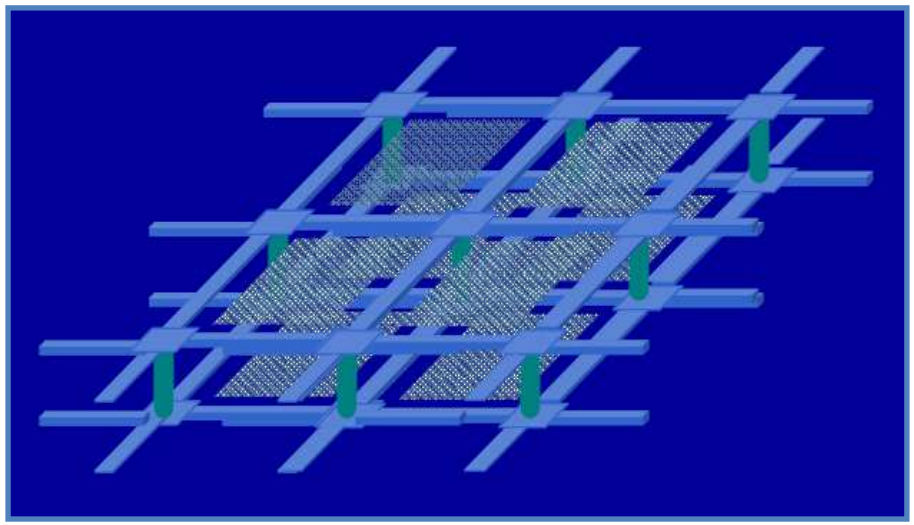
- Many-Core Processors
  - Interconnection

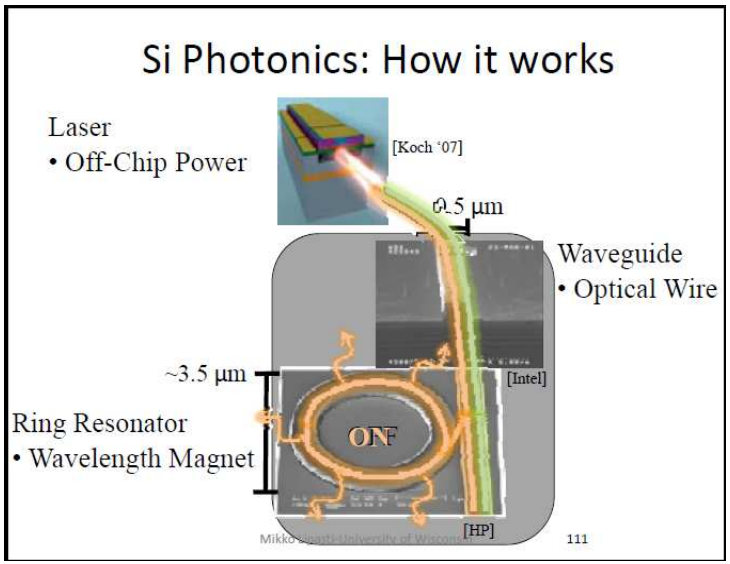


(a) A ring (b) A mesh (c) A torus

Networks on Chip (NoC)

- 3D Networks On Chip
- Photonic Networks On Chip





Course Content

- A quantitative approach to computer design & analysis
  - Techniques of quantitative analysis and evaluation of modern computing systems, with an emphasis on major component subsystems of high performance computers: pipelining, instruction level parallelism, memory hierarchies, input/output, and network-oriented interconnections.

Course Administration

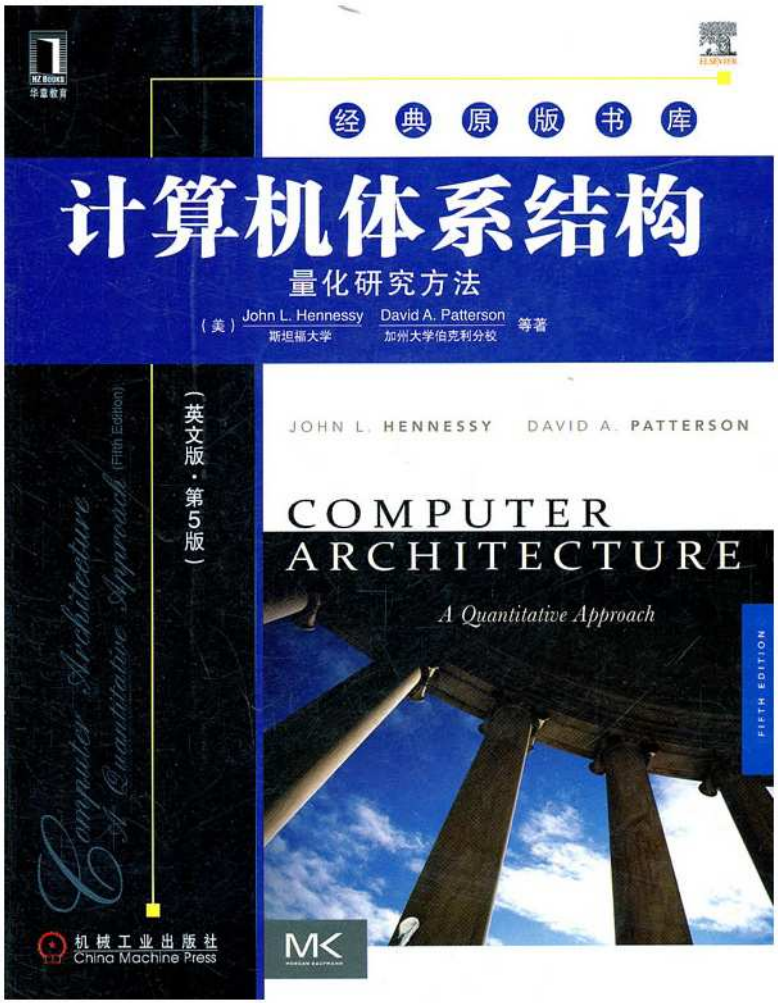
Instructor: Shuai Wang (swang@nju.edu.cn)
http://cs.nju.edu.cn/swang
Office: 416 CS Building

URL: http://cs.nju.edu.cn/swang/CA_16S/

Textbook: *Computer Architecture: A Quantitative Approach*, by John L. Hennessy & David A. Patterson

Slides: Posted after the lecture

Textbook


经典原版书库
计算机体系结构
量化研究方法
（美）John L. Hennessy 斯坦福大学 David A. Patterson 加州大学伯克利分校 等著
（英文版·第5版）
JOHN L. HENNESSY DAVID A. PATTERSON
COMPUTER ARCHITECTURE
A Quantitative Approach
FIFTH EDITION
机械工业出版社
China Machine Press

Grading Policy

- Class Participation： 10%
- Homework Assignments： 15%
- Labs and Projects: 40%
- Final Exam： 35%

Course Structure

Lecture outline

- Fundamentals of Computer Architecture, Pipelining, Performance, Caches, Virtual Memory, Cost
- Instruction Sets, DSPs, SIMD, Vector Processors
- Dynamic Execution
- Static Execution
- Memory Hierarchy
- Multiprocessors
- Input/Output and Storage
- Networks and Clusters
- Power Consumption
- Reliability

Today's Lecture

Review of the following topics

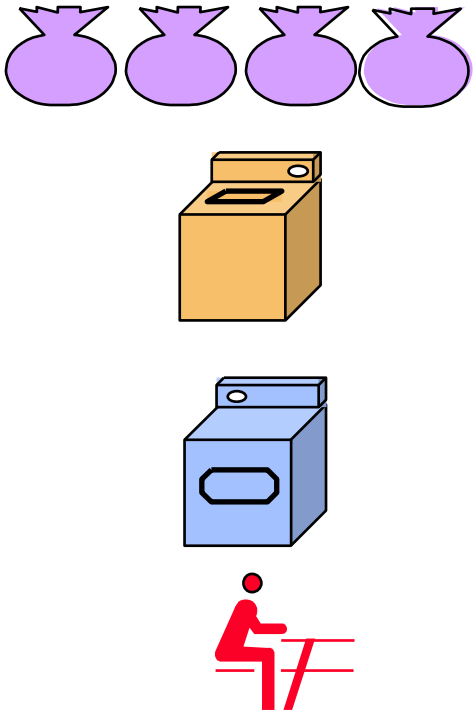
- Pipelining
- Caches
- Virtual memory

Fundamentals of computer design

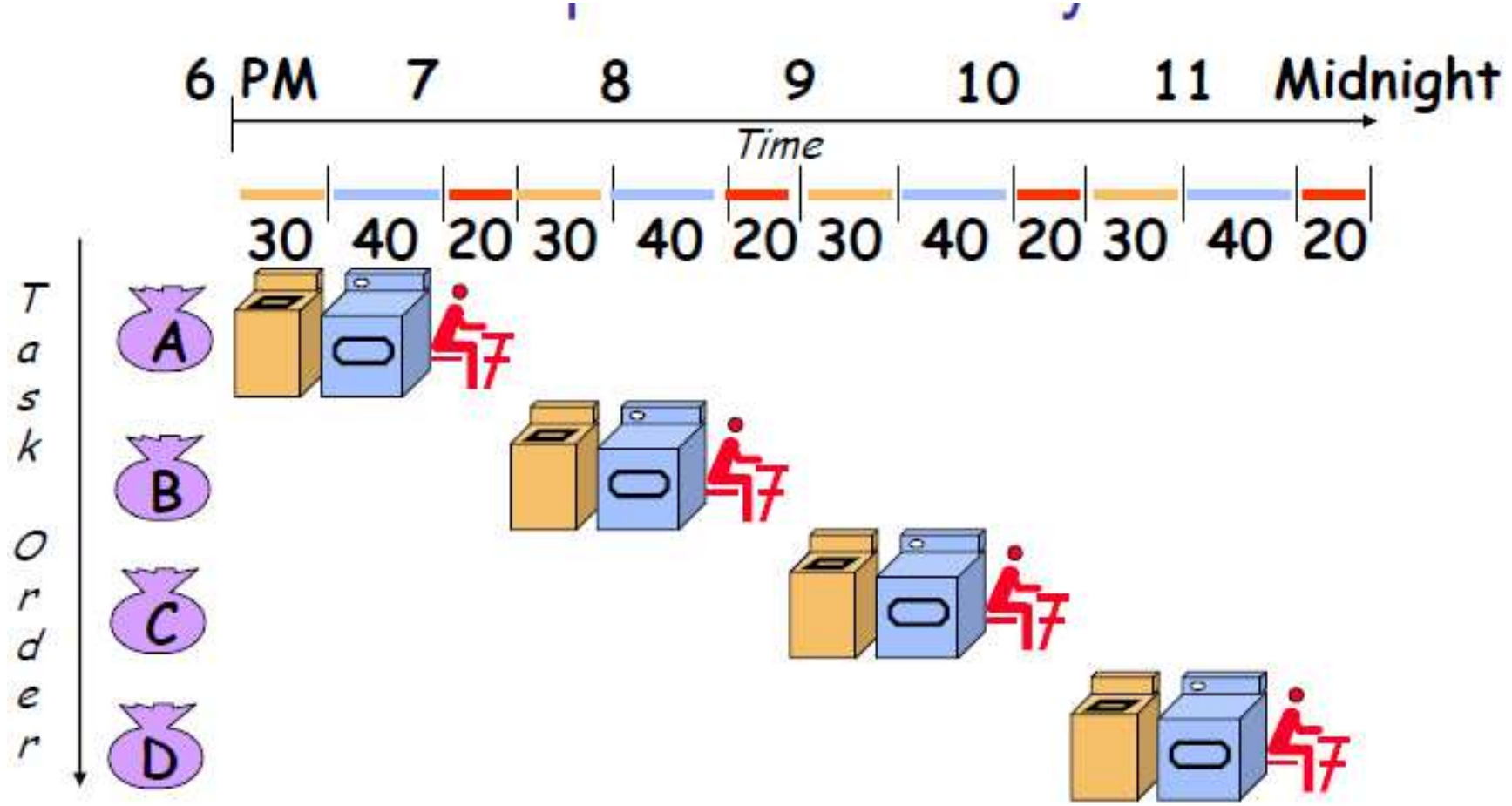
- Performance
- Cost
- Technology

Pipelining: An Example

- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- “Washer” takes 30 minutes
- “Dryer” takes 40 minutes
- “Folder” takes 20 minutes

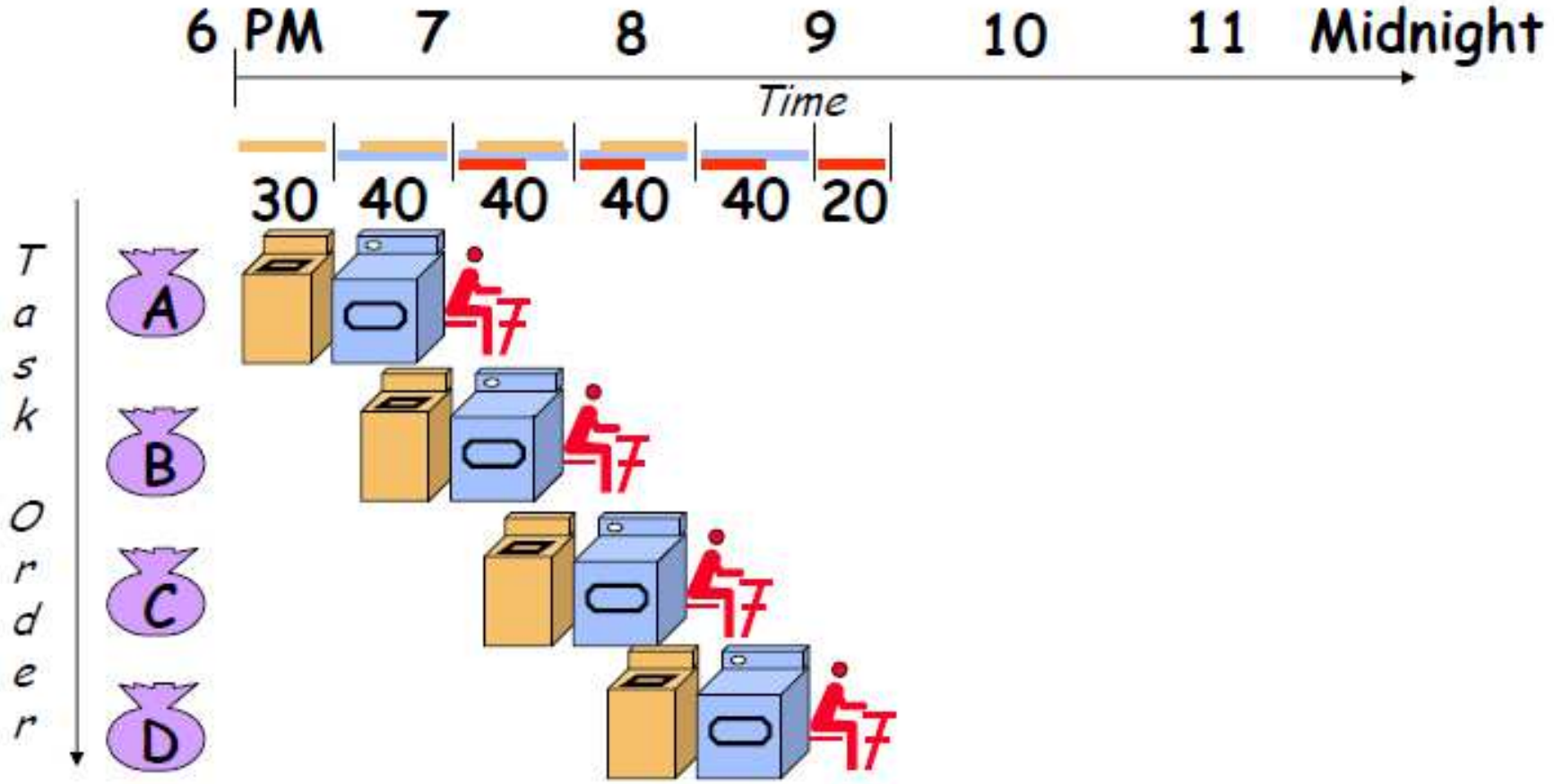


Sequential Laundry



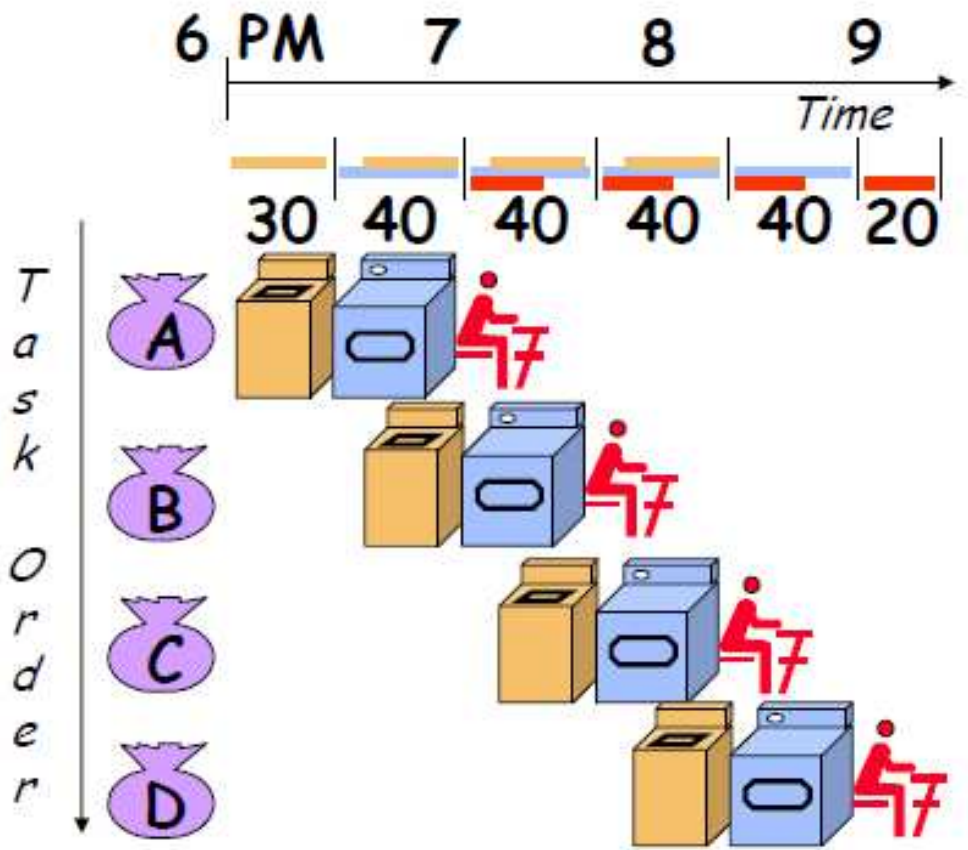
Sequential laundry takes 6 hours for 4 loads

Pipelined Laundry: Start Work ASAP



Pipelined laundry takes 3.5 hours for 4 loads

Pipelining Observations



- **Pipelining doesn't help latency of single task, it helps throughput of entire workload.**
- **Pipeline rate limited by slowest pipeline stage.**
- **Multiple tasks operating simultaneously.**
- **Potential speedup = Number pipe stages.**
- **Unbalanced lengths of pipe stages reduces speedup.**
- **Time to "fill" pipeline and time to "drain" it reduces speedup.**

Computer Pipelines

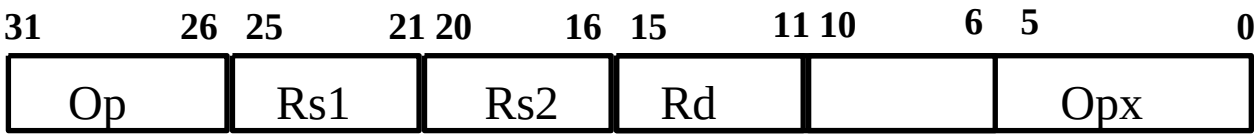
- Execute billions of instructions, throughput is what matters
- Pipelining exploits parallelism among sequential instruction stream
- What is desirable in instruction set architectures for pipelining?
  - Variable length instructions vs. all instructions same length?
  - Memory operands part of any operation vs. memory operands only in loads or stores?
  - Register operand many places in instruction format vs. registers located in same place?

A “Typical” RISC ISA

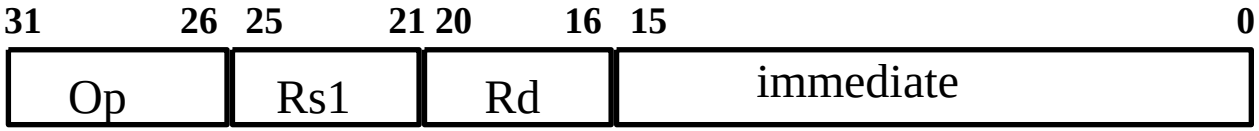
- 32-bit fixed format instruction (3 formats)
- Memory access only via load/store instructions
- 32 32-bit GPR (R0 contains zero)
- 3-address, register-register arithmetic instruction; registers in same place
- Single address mode for load/store:
  base + displacement
  - no indirection
- Simple branch conditions
- Delayed branch

Example: MIPS (Note register location)

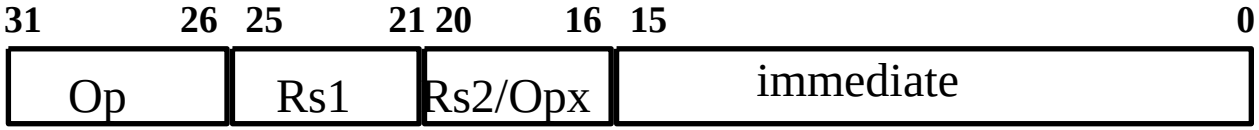
Register-Register



Register-Immediate

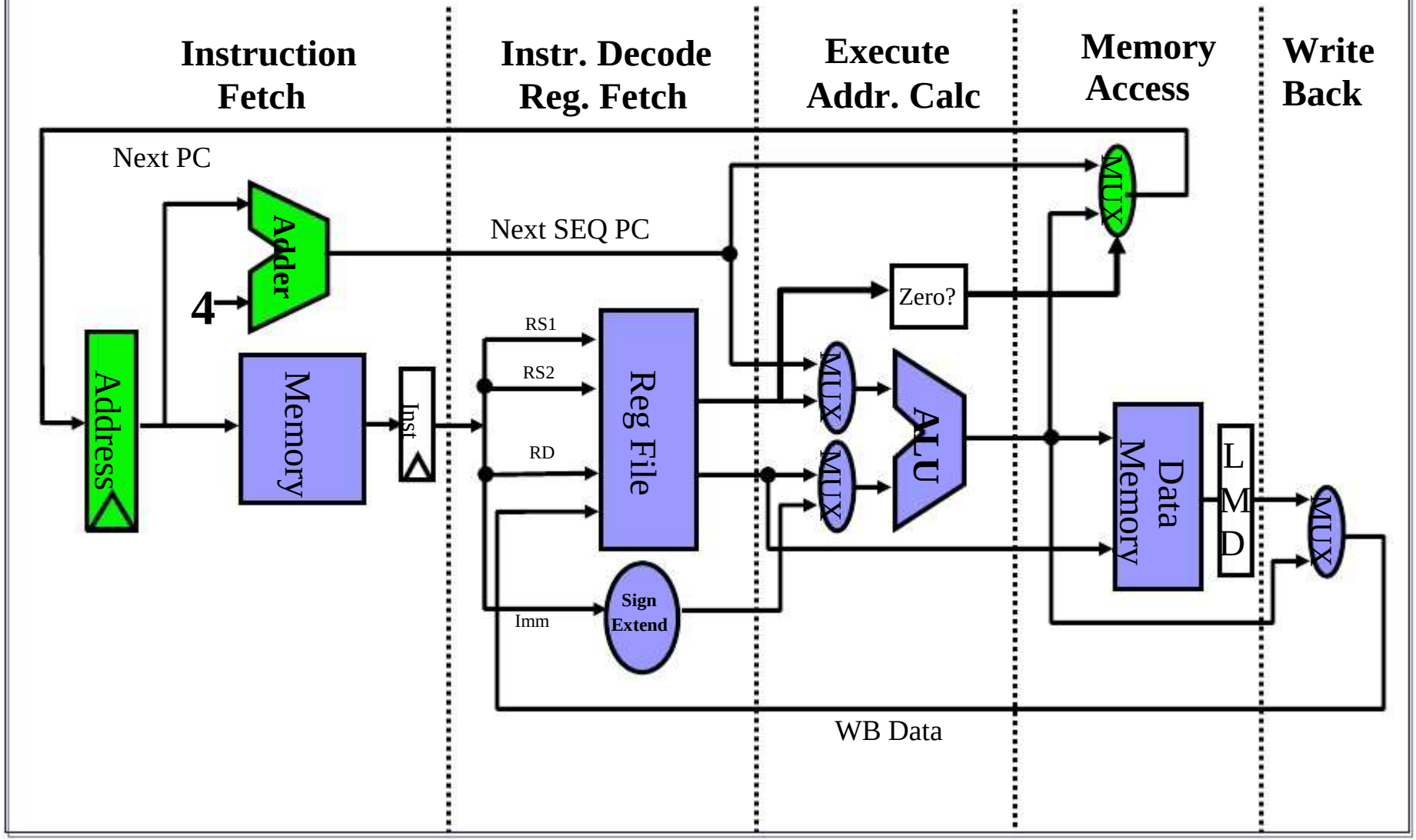


Branch

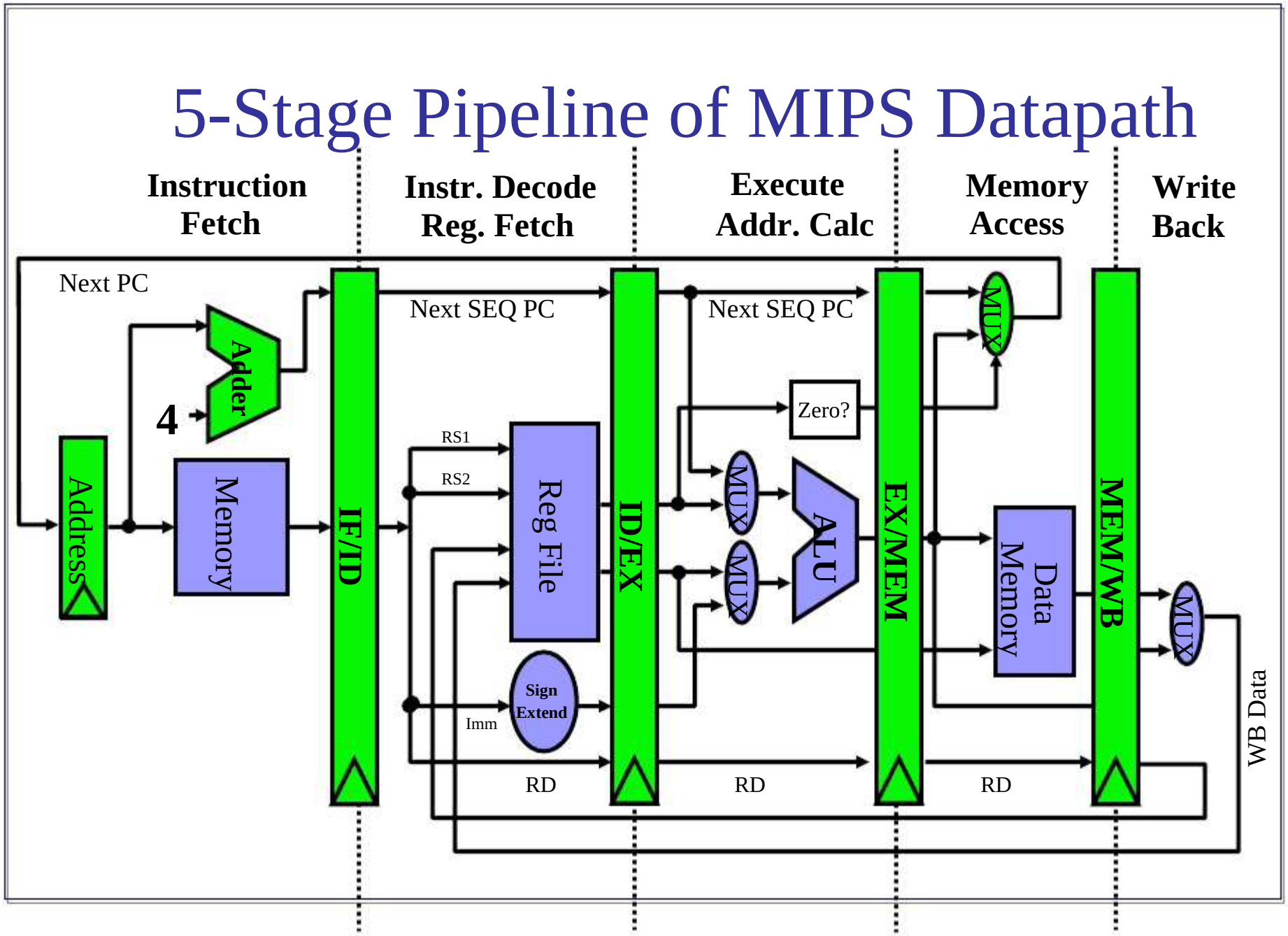


Jump / Call

5 Steps of MIPS Datapath



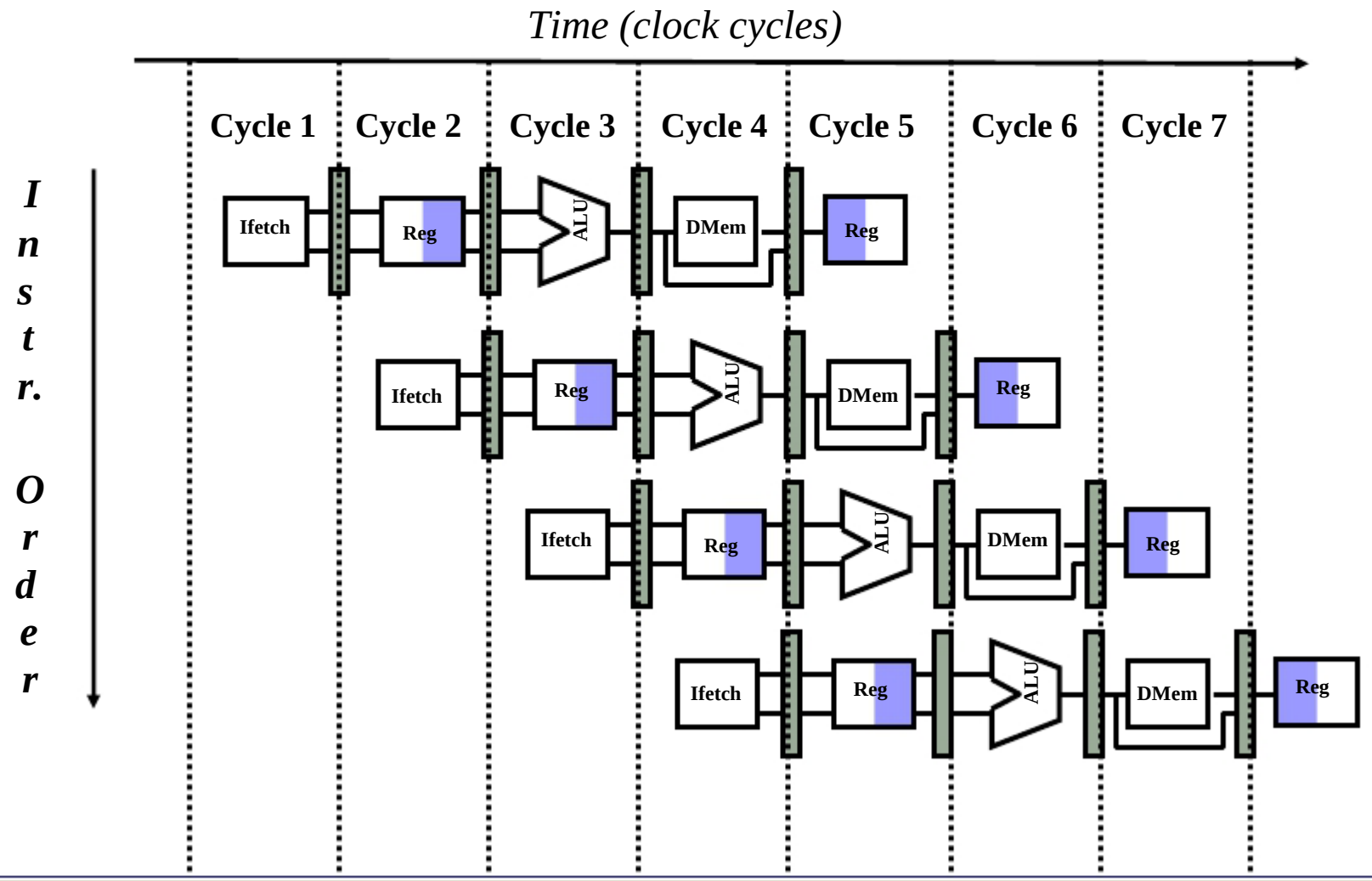
5-Stage Pipeline of MIPS Datapath



Visualizing Pipelining

	Clock number								
Instruction #	1	2	3	4	5	6	7	8	9
Inst. i	**IF**	**ID**	**EX**	**MEM**	**WB**				
Inst. i+1		**IF**	**ID**	**EX**	**MEM**	**WB**			
Inst. i+2			**IF**	**ID**	**EX**	**MEM**	**WB**		
Inst. i+3				**IF**	**ID**	**EX**	**MEM**	**WB**	
Inst. i+4					**IF**	**ID**	**EX**	**MEM**	**WB**

Visualizing Pipelining



Pipeline Hazards

- Major hurdle to pipelining: **hazards** prevent the next instruction from executing during its designated clock cycle
  - **Structural hazards:** hardware cannot support all possible combinations of instructions simultaneously
  - **Data hazards:** instruction depends on result of a previous instruction still in the pipeline
  - **Control hazards:** caused by delay between the fetching of instructions and decisions about changes in control flow