Computer Architecture: Introduction & Basic Concepts

Hossein Asadi (asadi@sharif.edu)
Department of Computer Engineering
Sharif University of Technology
Spring 2025

Copyright Notice

- Some Parts (text & figures) of this Lecture adopted from following:
  - D.A. Patterson and J.L. Hennessy, “Computer Organization and Design: the Hardware/Software Interface” (MIPS), 6th Edition, 2020.
  - “Intro to Computer Architecture” handouts, by Prof. Hoe, CMU, Spring 2009.
  - “Computer Architecture & Engineering” handouts, by Prof. Kubiatowicz, UC Berkeley, Spring 2004.
  - “Intro to Computer Architecture” handouts, by Prof. Hoe, UWisc, Spring 2021.
  - “Computer Arch I” handouts, by Prof. Garzarán, UIUC, Spring 2009.
  - “Intro to Computer Organization” handouts, by Prof. Mahlke & Prof. Narayanasamy, Winter 2008.

Our Lectur Today

Topics Covered Today

- Syllabus & Objectives
- Textbook
- Assignments
- Class Policy
- Reminder from “Logic Design” & “CSL” Course
- Introduction to Computer Architecture

Course Introduction

- Instructor: Hossein Asadi
- Classes
  - Sat. & Mon.: 15:00~16:30
  - Attend class on time
    - Starts at 15:00 and usually ends by 16:10

Course Introduction (cont.)

- Office Hours
  - I am usually online and can be reached email
    - asadi@sharif.edu
  - In my Room: Sat. through Wed.: 8:30AM ~ 6PM
  - Room # 610
- TA Classes
  - TBD

Course Introduction (cont.)

- Course Webpage on CW
  - Check this webpage on regular basis
    - At least on Sun, Tue, Thur
    - Q&A only using CW forums
  - Everything will be posted on CW
    - Announcements, handouts, assignments, grades, quiz and exam notices, simulators, …
  - Handouts
    - Will be posted a day before class
      - Print it & bring it to class
    - But I may update it a day after class
      - Check out submission date of handouts

Teaching Assistants

- Comments, Suggestions, & Objections
  - Ali Sedaghatgoo (TA Chair)
- TAs
  - – Already listed in CW!

Few Notes on Assignments

- Post All your Questions on CW Forums
  - Check forum history before posting any question
- Be Respectful to your Classmates and TAs
- Harsh Cheating Penalty

Course Introduction (cont.)

- Course Webpage
  - Sharif CW webpage, http://cw.sharif.edu
  - Make sure to have an account on CW
  - Check this webpage on regular basis
    - At least on Sun, Tue, Thur
  - Everything will be posted online
    - Announcements, assignments, and toolsets
  - Handouts (in pdfs)
    - Print it & bring it to class
    - I may update it a day after class
      - Check out submission date of handouts

Course Introduction (cont.)

- Textbook
  - D.A. Patterson and J.L. Hennessy, "Computer Organization and Design: the Hardware/Software Interface" (MIPS), 6th Edition, 2020.

Syllabus

- **Review**
  - Combinational & sequential logic design
  - Design abstractions
  - Computer/CPU history
  - Computer organization
  - Addressing modes
  - Instruction Set Architecture (ISA)
- **Number Representation**
  - Fixed-point
  - IEEE 754 Floating-point standard
    - Single precision and double precision

Syllabus (cont.)

- **Performance Evaluation**
  - Performance
  - Important factors in performance
  - Benchmarks
- **Data-Path and Control-Path Design**
  - Register Transfer Logic (RTL)
  - Data-path components
  - Control unit design and hardwired controller
  - MIPS data-path
  - Interrupt and I/O polling

Syllabus (cont.)

- Micro-Programmed Controller
  - Pros & cons compared to hardwired
- Multi-Cycle Architecture
- Introduction to Pipeline Architecture
- I/O Approaches
  - I/O handshaking
- Introduction to Multi-Core Systems
- Introduction to Parallel Computing

Syllabus (cont.)

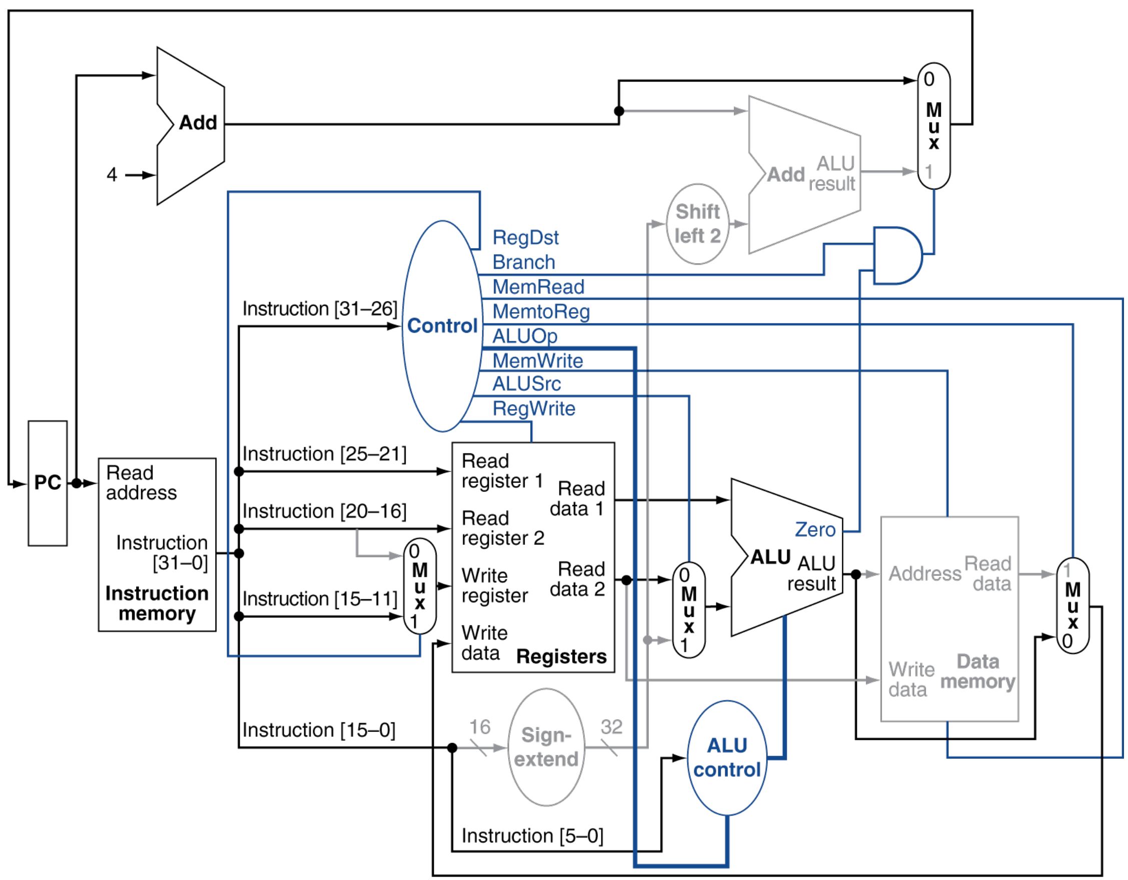
- Memory System
  - Types of memory
  - Memory hierarchy
  - Cache memory and cache configurations
- Arithmetic Algorithms
  - Addition, subtraction, multiplication, division
  - Arithmetic architectures
  - Booth and array multiplication

Topics Covered in This Course (I)

- Review of Performance Metrics & ISA in Computer Systems
  - Response time
  - Throughput
  - CPI (Clocks Per Instruction)
  - Benchmarking
  - RISC vs. CISC

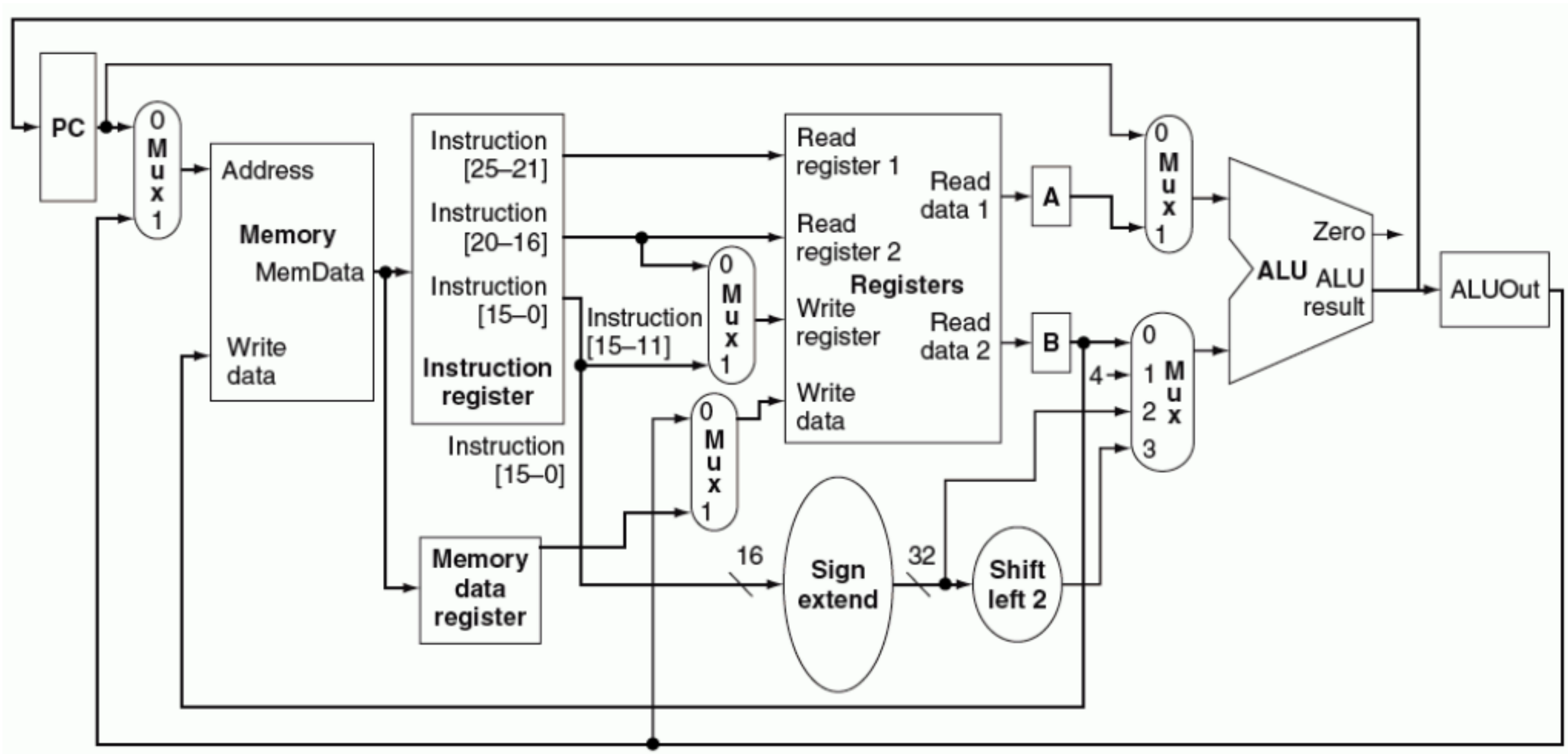
Topics Covered in This Course (II)

- MIPS Microarchitecture
  - Single cycle datapath
  - Control unit design



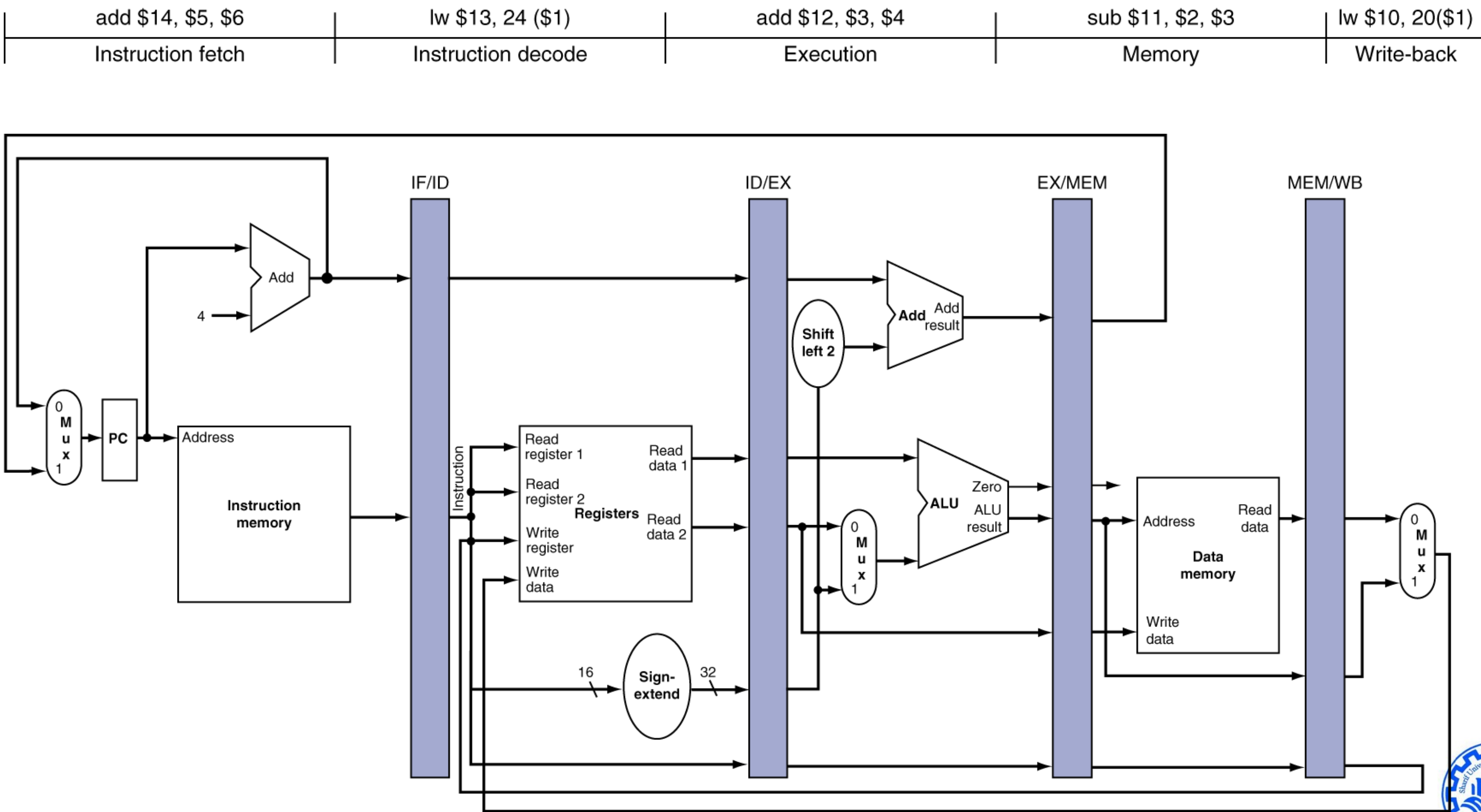
Topics Covered in This Course (III)

- MIPS Microarchitecture
  - Multi-cycle datapath



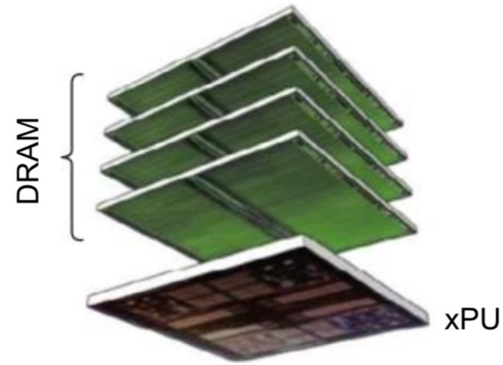
Topics Covered in This Course (IV)

- MIPS Microarchitecture
  - Pipelined data path

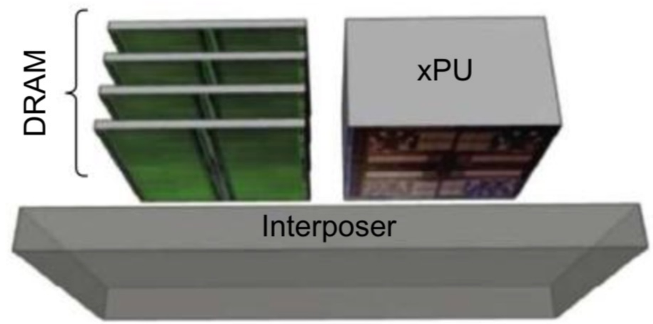


Topics Covered in This Course (V)

- Memory Internals in Advanced Computer Systems

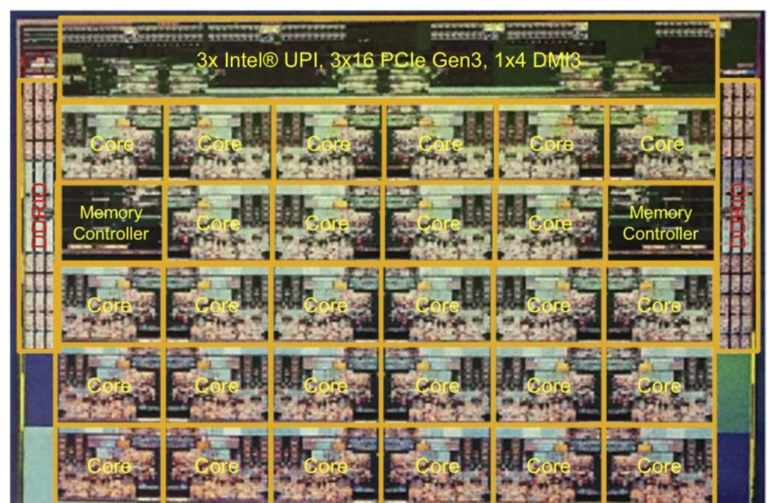


Vertical stacking (3D)



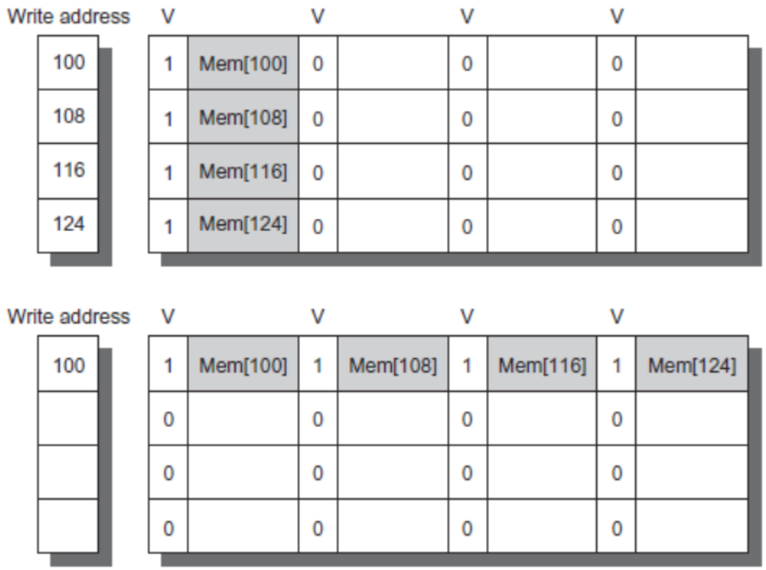
Interposer stacking (2.5D)

- Major Functional Units in Advanced Processors



Topics Covered in This Course (VI)

- Advanced Optimizations in Memory Hierarchy
  - Cache organization
  - Multi-level cache design
  - Write buffers
  - Stored buffer
  - Compiler prefetching
  - Hardware prefetching



Objective

- By the end of semester, you should be able to answer these questions:
  - What is functionality of main components of a RISC processor?
  - Why standard benchmarks used for performance evaluation?
  - What are pros and cons of single-cycle, multi-cycle, and pipelined data-paths?
  - Difference between micro-programmed controller and hardwired controller?

Objective (cont.)

- By the end of semester, you should be able to answer these questions:
  - Tradeoffs of small vs. large L1 caches?
  - How many levels in a cache hierarchy?
  - What are pros and cons of direct-mapped, set-associative, and fully-associative cache configurations?
  - What are pros and cons of different adder implementations (RC, CSA, CLA)?
    - Ripple-carry, carry-select, carry look-ahead adder

Grading

- Midterm Exam: 20%~25%
  - Ordibehesht 1st
- Final Exam: ~30% (date posted in EDU)
- Quiz (1&2): 10%~15%
  - First quiz: Esfand 20th
  - Second quiz: Ordibehesht 29th
  - Up to five additional unscheduled quizzes
- Assignments & Project: ~30-40%
  - Bonus points for outstanding projects
- Exams: Topics of this Class and TA Classes

Class Policy

- Ask Questions Anytime
  - Don’t hesitate to ask even stupid questions!!!
- Cell Phones Off or on Silent
- Absence
  - Only three sessions allowed
- Food No, Drink yes!
- Feel Free to Pass Me Your Feedbacks
  - Anything related to this course
- Will have random in-class quizzes

Assignments

- ~10 Assignments
  - ~4 analytical assignments
  - ~6 design & simulation assignments
    - Altera (Intel) Quartus or similar toolset ©
    - SimpleScalar/Gem5 toolset ©
  - Spend enough time on assignments as they will be covered in midterm and final exams

Assignments (cont.)

- Assignment Policy
  - Three late assignments will be accepted!
    - Only two days late!
    - Fourth and next late assignment (two-day late)
      - HW will be graded out of 50%
  - Discussions encouraged!
  - But do your own handwriting!
  - Zero score for copied assignments!
    - Second time zero score for 30% share!

What You Learned So Far

- Logic Design
  - Simple logical & arithmetic logic design
    - Addition and subtraction units
    - Multiplexer and tri-state buffer
    - Latch and flip-flop
    - Sequential logic, registers, shifters, counters

- Computer Structure & Language
  - Computer organization
  - Instruction Set Architecture (ISA)
  - Assembly programming

- Now “Computer Architecture”
  - What is “Computer Architecture”?

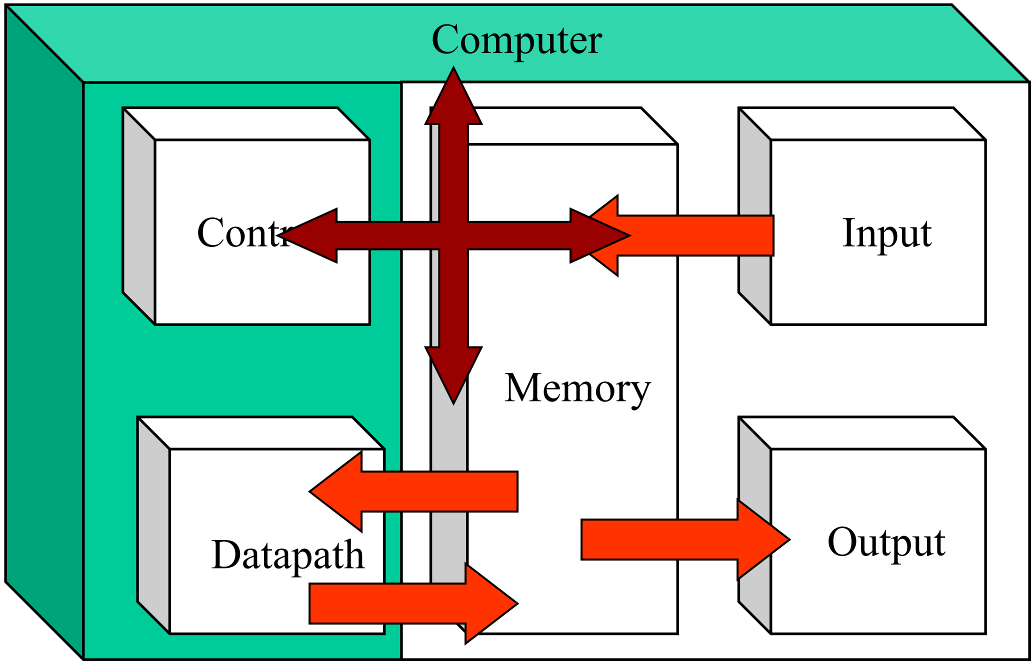
Computer Architecture: Basic Concepts

Reminder: Computer Systems

- A computer system consists of hardware and software that are combined to provide a tool to solve problems (with best performance)
  - Hardware may include:
    - CPU, memory, disks, printers, screen, keyboard, mouse, …
  - Software may include:
    - System software
      - A general environment to create specific applications
    - Application software
      - A tool to solve a specific problem

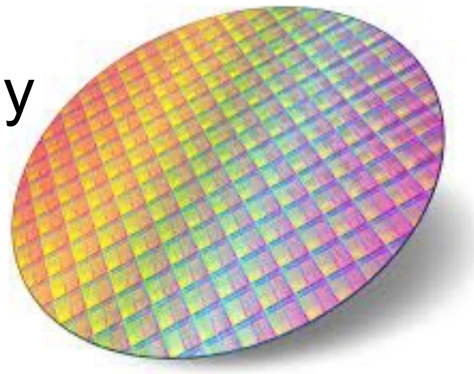
Reminder: Computer Organization

- Computer Components
  - Input, output, memory, control unit, & datapath

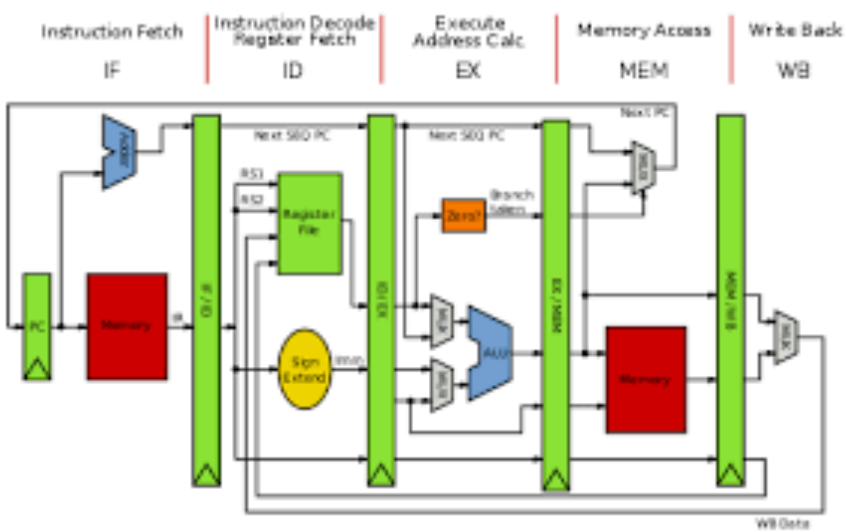


Computer Technology Trend in Past

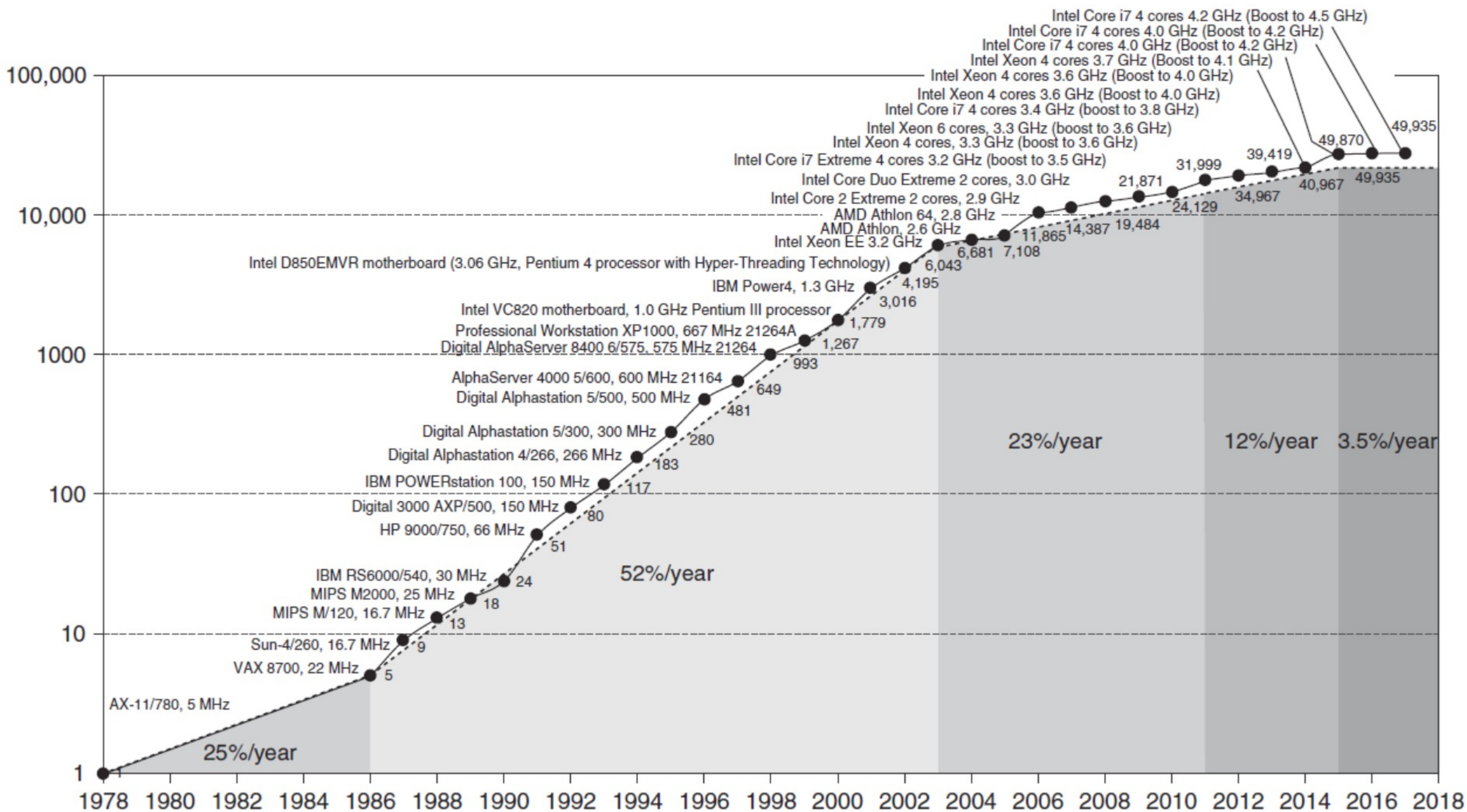
- **Performance** Improvements:
  - Improvements in semiconductor technology
    - Feature size, clock speed



  - Improvements in computer architectures
    - Enabled by HLL, compilers, Operating Systems
    - Lead to RISC architectures



Single Processor Performance



Current Trends in Architecture

- Cannot Continue to Leverage Instruction-Level Parallelism (ILP)
  - E.g., pipelining, out-of-order execution, speculative execution, & superscalar architectures
  - Single processor performance improvement ended in 2003

- New Models for Performance:
  - Data-Level Parallelism (DLP)
  - Thread-Level Parallelism (TLP)
  - Request-Level Parallelism (RLP)

- These require explicit restructuring of **application**

Parallelism

- Classes of Parallelism in Applications
  - Data-Level Parallelism (DLP)
    - Multiple data items can be operated at the same time
  - Task-Level Parallelism (TLP)
    - Multiple tasks of a work can operate independently and largely in parallel

Parallelism (cont.)

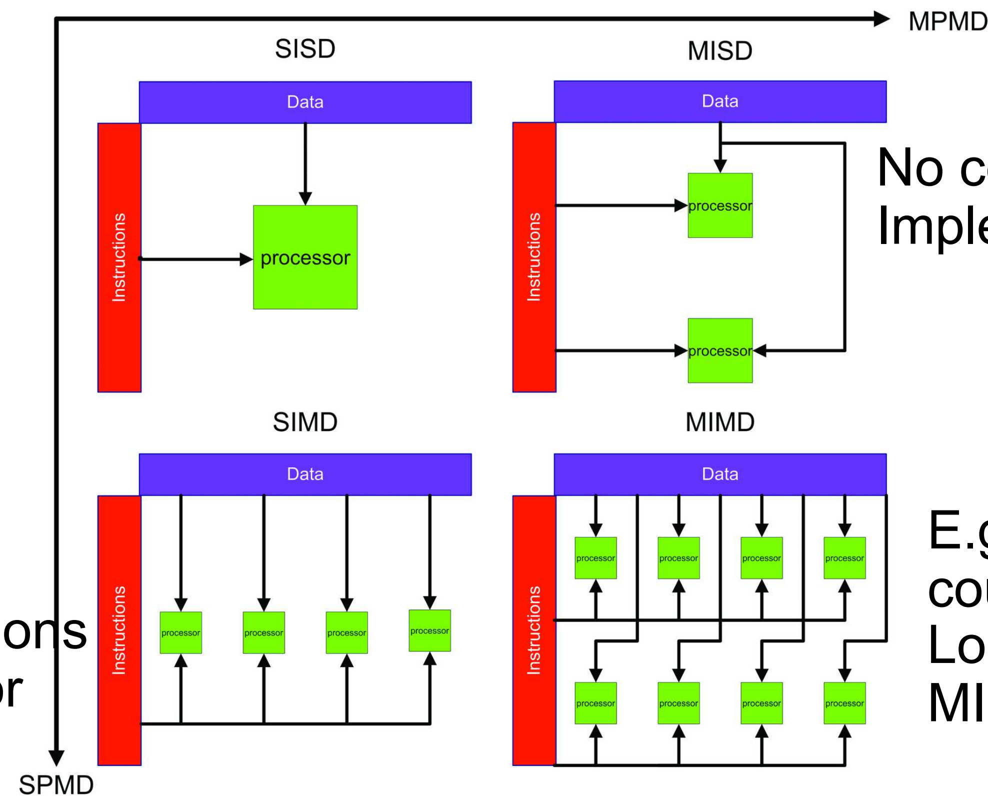
- Classes of Architectural Parallelism
  - Instruction-Level Parallelism (ILP)
    - Pipelining, out-of-order execution, speculative execution
  - Vector architectures/Graphic Processor Units (GPUs)
    - Exploits data-level parallelism by applying a single instruction to a collection of data in parallel
  - Thread-Level Parallelism
    - Exploits either data-level or task-level parallelism
    - Used in a tightly coupled hardware model that allows for interaction between parallel threads
  - Request-Level Parallelism
    - Parallelism among largely decoupled tasks specified either by programmer or OS

Flynn's Taxonomy for Parallelism

Single Instruction stream, Single Data stream (SISD)

Multiple Instruction stream, Single Data stream (MISD)

E.g., Uniprocessors



No commercial Implementation

E.g., Vector architectures
Multimedia extensions
Graphics processor units

E.g., Tightly-coupled MIMD, Loosely-coupled MIMD

Single Instruction stream, Multiple Data stream (SIMD)

Multiple Instruction stream, Multiple Data stream (MIMD)

Classes of Computers (I)

Personal Mobile Device (PMD)

- Smart phones, tablet computers
- Emphasis on energy efficiency and real-time

Desktop Computing

- Price-performance

Servers

- Emphasis on availability, scalability, & throughput

Classes of Computers (II)

Clusters / Warehouse Scale Computers

- Used for “Software as a Service (SaaS)”
- Emphasis on availability and price-performance
- Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks

Internet of Things (IoT) / Embedded Computers

- Emphasis on price, power, energy, reliability

Why Availability Important in Clusters?

Higher Cost of Downtime ➔

Higher Importance of Availability

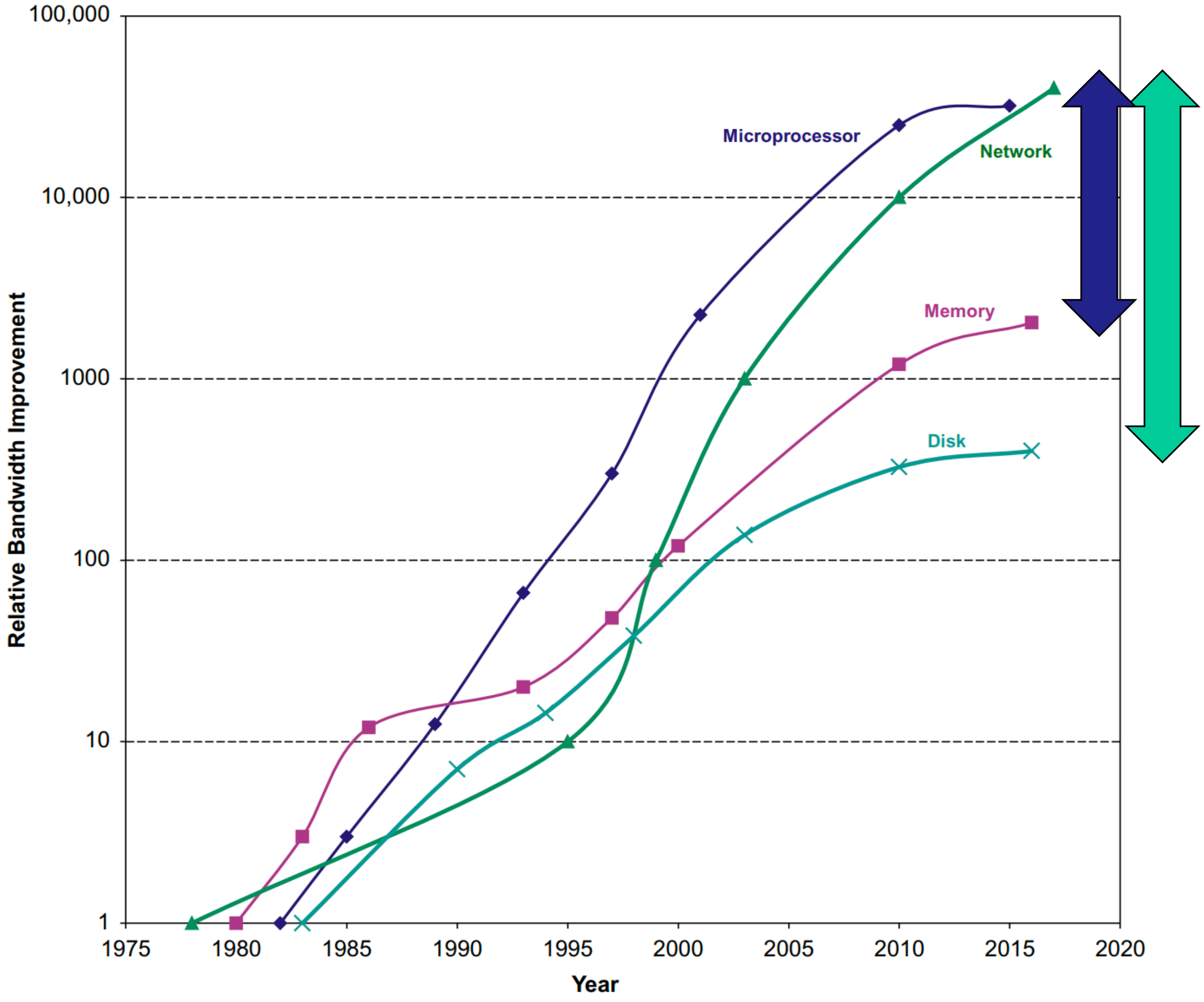
Application	Cost of downtime per hour	Annual losses with downtime of 1% (87.6 h/year)	0.5% (43.8 h/year)	0.1% (8.8 h/year)
Brokerage service	$4,000,000	$350,400,000	$175,200,000	$35,000,000
Energy	$1,750,000	$153,300,000	$76,700,000	$15,300,000
Telecom	$1,250,000	$109,500,000	$54,800,000	$11,000,000
Manufacturing	$1,000,000	$87,600,000	$43,800,000	$8,800,000
Retail	$650,000	$56,900,000	$28,500,000	$5,700,000
Health care	$400,000	$35,000,000	$17,500,000	$3,500,000
Media	$50,000	$4,400,000	$2,200,000	$400,000

$$Availability = \frac{Uptime}{Total\ Time} \qquad Unavailability = \frac{Downtime}{Total\ Time}$$

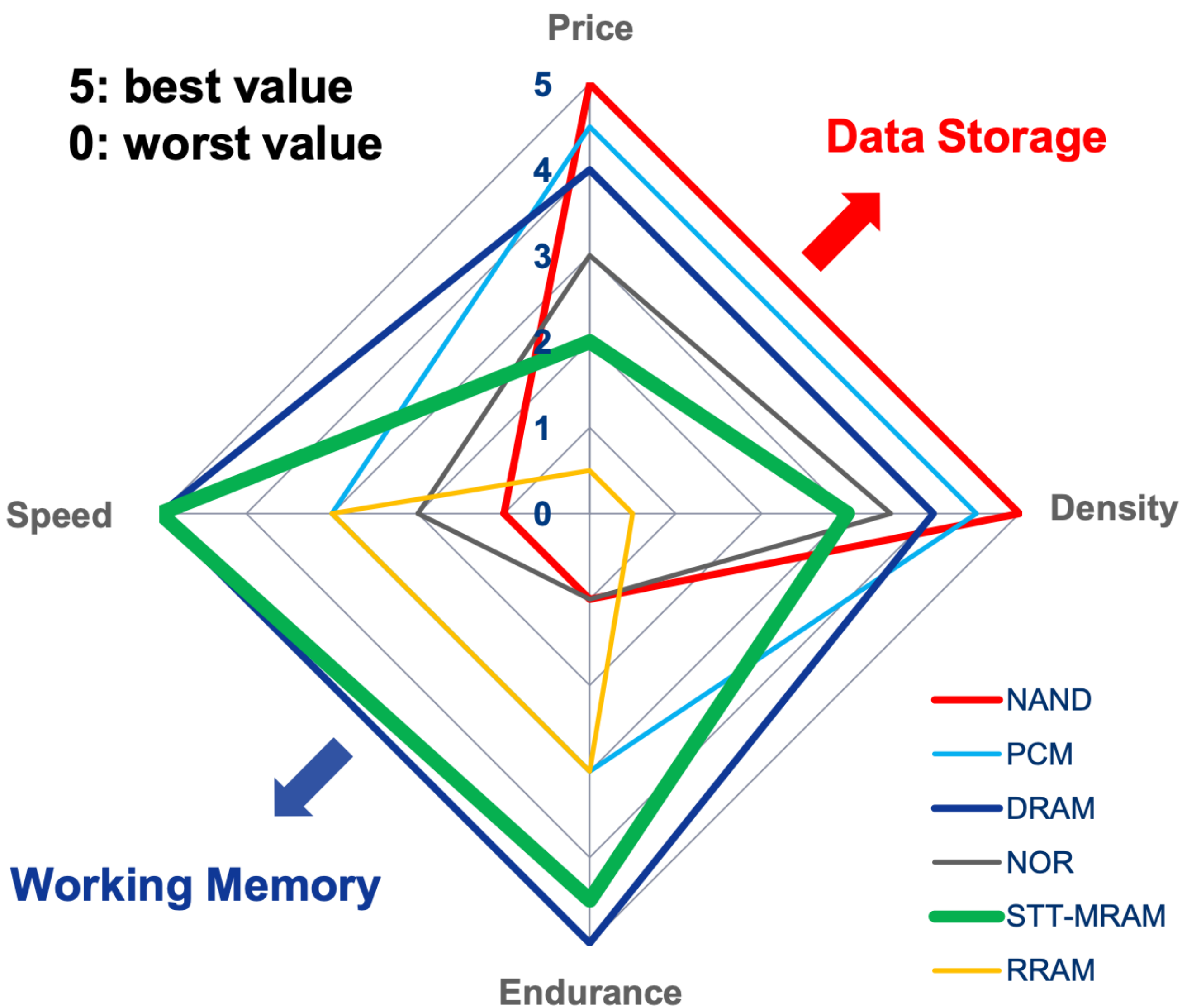
Trends in Technology: Key Parameters

- Cost ➔ Integrated Circuit Technology (Moore's Law)
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
- Capacity
  - DRAM capacity: 25-40%/year (slowing)
  - Flash capacity: 50-60%/year
    - 8-10X cheaper/bit than DRAM
  - Magnetic disk capacity
    - 8-10X cheaper/bit than Flash
    - 200-300X cheaper/bit than DRAM
- Performance
  - DRAM > Flash > HDD
- Endurance
  - Flash limited, HDD and DRAM unlimited

Trend in Bandwidth Improvement



Memory Technology Comparison



Computer Systems Abstractions

Applications	User and problems
Compilers	Prog. Languages
Operating System (OS)	Resources / virtualization
Architecture (ISA)	HW/SW interface
Micro-architecture	Datapath
Digital Design	Registers, ALU
Circuit	Digital logic
Device	Transistors, signals
	Atoms, electrons

Micro-Architecture (uArch)

- Definition from Wiki
  - A way a given ISA is implemented on a processor
- ISA
  - Can be implemented with different uArch
  - Why different implementation?
    - Different goals (performance, power, cost, …)
- Computer Architecture?
  - ISA + uArch?

Defining Computer Architecture

- "Old" View of Computer Architecture:
  - Instruction Set Architecture (ISA) design
  - i.e. decisions regarding:
    - Registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- In Reality, Computer Architecture Means:
  - Specific requirements of target machine
  - Design to maximize performance within constraints: cost, power, availability, and reliability
  - Includes ISA, microarchitecture, hardware, even logic design, and somehow OS

Computer Systems Abstractions (cont.)

High Level Language Program (e.g., C)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

Compiler

Assembly Language Program (e.g.,MIPS)

```
lw   $t0, 0($s2)
lw   $t1, 4($s2)
sw   $t1, 0($s2)
sw   $t0, 4($s2)
```

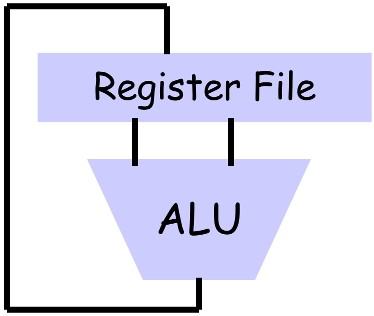
Assembler

Machine Language Program (MIPS)

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Machine Interpretation

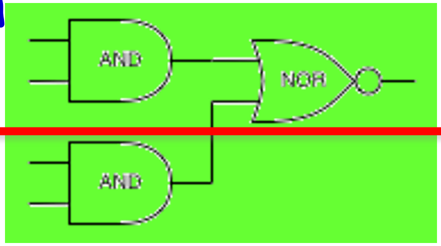
Hardware Architecture Description



Architecture Implementation

Logic Circuit Description

(Circuit Schematic Diagrams)



Reminder: ISA

- Instruction Set Architecture (ISA)
  - A set of instructions used by a machine to run programs
  - Interface between hardware & software
  - Provides an abstraction of hardware implementation
    - Hardware implementation decides what and how instructions are implemented
  - ISA specifies
    - Instructions, Registers, Memory access, Input/output

Reminder: ISA (cont.)

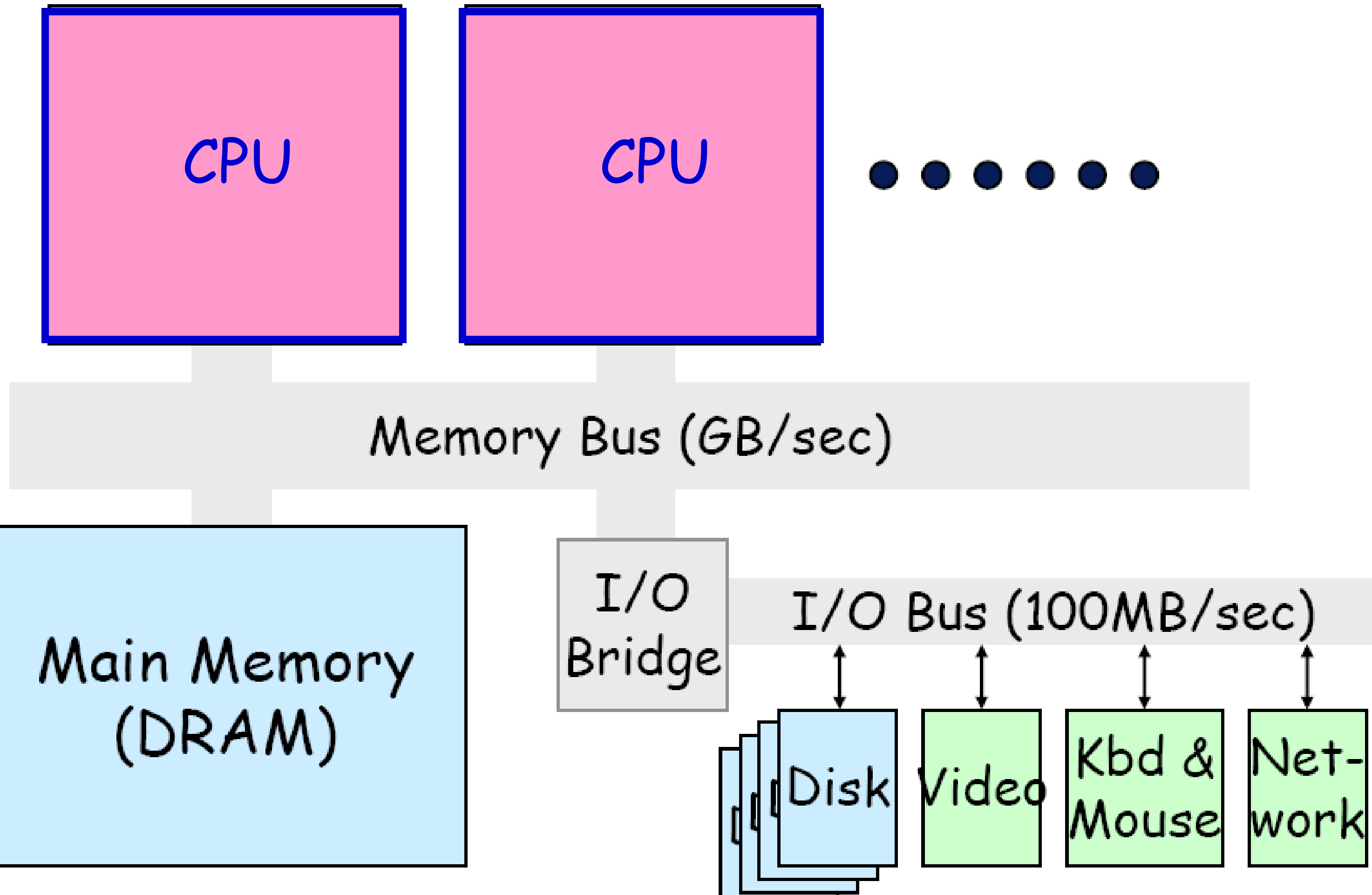
- Key ISA Decisions
  - Instruction length?
  - How many registers?
  - Where operands reside?
    - Which instructions can access memory?
  - Instruction format?
  - Operand format?
    - How many? How big?

Reminder: Addressing Modes

- Addressing Modes
  - Immediate addressing
  - Register addressing
  - Base or displacement addressing
  - PC-relative addressing
  - Pseudo-direct addressing
  - Register indirect
  - Direct
  - Memory indirect
  - Scaled
  - Auto-increment / Auto-decrement
  - Indexed

Reminder: Computer Organization

- Computer Components
  - Input, output, memory, control unit, & datapath



Reminder: Typical ISA

- Data Transfer Instructions
  - CPU ⇔ Memory
  - CPU ⇔ I/O
- Arithmetic & Logical Instructions
- Control Instruction
  - Conditional branch
  - Unconditional branch

Reminder: ISA (cont.)

- Computer ISA
  - How many instructions needed?
  - What functionalities required?
    - Load
    - Store
    - Control (such as compare & jump)
    - Arithmetic & logical operations
- Example ISAs
  - ISA_{ARMX}={$add_{8/r/r}$, $sub_{8/r/r}$, or, jmp, load, store}
  - ISA_{IntelY}={$add_{8/r/r}$, $add_{16r/M}$, $add_{c16/M/M}$, $mul_{8/r/r}$, $mul_{16r/M}$, $mul_{32M/M}$, $div_{8/r/r}$, $div_{16r/M}$, $div_{32M/M}$, ….,}

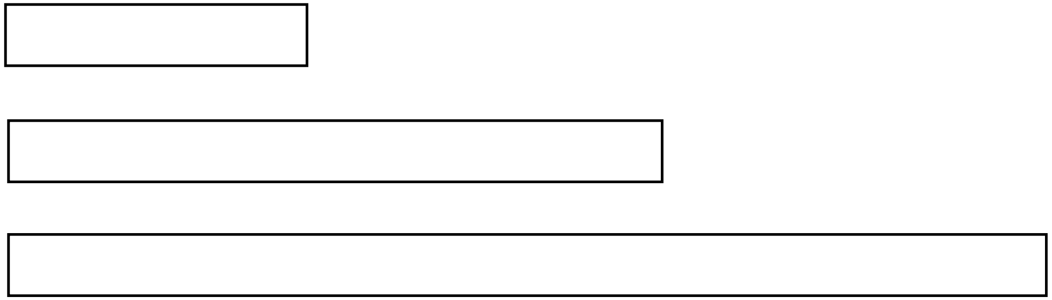
Reminder: ISA (cont.)

- Key ISA Decisions
  - Instruction length?
  - How many registers?
  - Types, size, & location of operands
  - Memory addressing?
  - Which instructions can access memory?
  - How to access operands? (Addressing modes)
    - Register, immediate, displacement (base+offset)
    - Or, autoincrement, indexed, PC-relative
  - Instruction format?
  - Operand format?
    - How many? How big?

Reminder: ISA (cont.)

- Instruction length?

Variable:



x86 – Instructions vary from 1 to 17 Bytes long

VAX – from 1 to 54 Bytes

Fixed:

MIPS, PowerPC:

all instruction are 4 Bytes long

Reminder: ISA (cont.)

- Instruction length?
  - Variable-length instructions
    - Require multi-step fetch and decode
    - Allow for a more flexible and compact instruction set
    - CISC processors like x86 & VAX
      - Complex Instruction Set Computing
  - Fixed-length instructions
    - Allow easy fetch and decode
    - Simplify pipelining and parallelism
    - RISC processors like MIPS & PowerPC
      - Reduced Instruction Set Computing

Reminder: RISC vs. CISC

- Early Trend:
  - Adding more instructions to next generation CPUs to do more complicated operations
  - VAX arch had an instruction to multiply polynomials!
- CISC Philosophy
  - Limited main memory + immature compilers
  - ➔ More dense instructions, highly encoded, variable length instructions
  - ➔ Data loading as well as calculation

Reminder: RISC vs. CISC (cont.)

- RISC Philosophy
  - Keep ISA small and simple
    - ➔ Makes it easier to build faster hardware
  - Let SW do complicated operations by composing simpler instructions
- Note on “Reduced" in RISC Phrase:
  - Amount of work any single instruction accomplishes reduced
    - Single ALU operation, single memory access, …

Reminder: RISC vs. CISC (cont.)

- CISC Problems
  - Performance tuning challenging
    - Complex/high-level instructions rarely used
  - Slower clock rates
  - Longer time-to-market
    - Due to prolonged design time
- CISC Features
  - Ease compiler implementation
    - HW supports all kind of addressing modes

Reminder: RISC vs. CISC (cont.)

- RISC Features
  - Low complexity
    - Less error-prone HW implementation
  - Implementation advantages
    - Less transistors
    - Extra space: more registers, cache
  - Marketing
    - Reduced design time
- Hybrid Solution?
  - RISC core & CISC interface
  - Taking advantage of both architectures

Reminder: ISA: How Many Registers?

- Advantages of a Small # of Registers
  - Requires fewer bits to specify which one
  - Less hardware
  - Faster access
    - Shorter wires
    - Faster index decoding (fewer logic levels)
  - Faster context switch
    - When all registers need saving
- Advantages of a Larger # of Registers
  - Fewer <u>loads</u> and <u>stores</u> needed
    - Less data transfer between CPU & memory
  - Easier to do several operations at once

Reminder: ISA Classes

Stack Machine

- "0-operand" ISA
  - ALU operations don't need any operands
- "Push": Loads mem into top of stack
- "Pop" : Does reverse
- "Add", "Sub", "Mul", and etc.
  - Combines contents of first two regs

Accumulator Machine

- "1-operand" ISA
- Only 1 register
  - Stores intermediate arithmetic & logic results
- Instructions include
  - "Store"
  - "Load"
  - "acc $\leftarrow$ acc + mem"

Load-Store Machine

- Aka, Register-Register Machine
- Operands
  - Only registers
- Arithmetic & logical instructions can only access registers

Register-Memory Machine

- Operands
  - Register or memory
- Arithmetic instructions can use data in registers and/or memory

Reminder: ISA Classes: Example

Code sequence for **C = A + B**

Stack	Accumulator	Load-Store	Register-Memory
`Push A`	`Load A`	`Load R1,A`	`Add C, A, B`
`Push B`	`Add B`	`Load R2,B`	
`Add`	`Store C`	`Add R3,R1,R2`	
`Pop C`		`Store C,R3`	

ISA: Load-Store Machine

- RISC-V registers
  - 32 g.p., 32 f.p.

Register	Name	Use	Saver
x0	zero	constant 0	n/a
x1	ra	return addr	caller
x2	sp	stack ptr	callee
x3	gp	gbl ptr	
x4	tp	thread ptr	
x5-x7	t0-t2	temporaries	caller
x8	s0/fp	saved/ frame ptr	callee

Register	Name	Use	Saver
x9	s1	saved	callee
x10-x17	a0-a7	arguments	caller
x18-x27	s2-s11	saved	callee
x28-x31	t3-t6	temporaries	caller
f0-f7	ft0-ft7	FP temps	caller
f8-f9	fs0-fs1	FP saved	callee
f10-f17	fa0-fa7	FP arguments	callee
f18-f27	fs2-fs21	FP saved	callee
f28-f31	ft8-ft11	FP temps	caller

ISA: Load-Store Machine

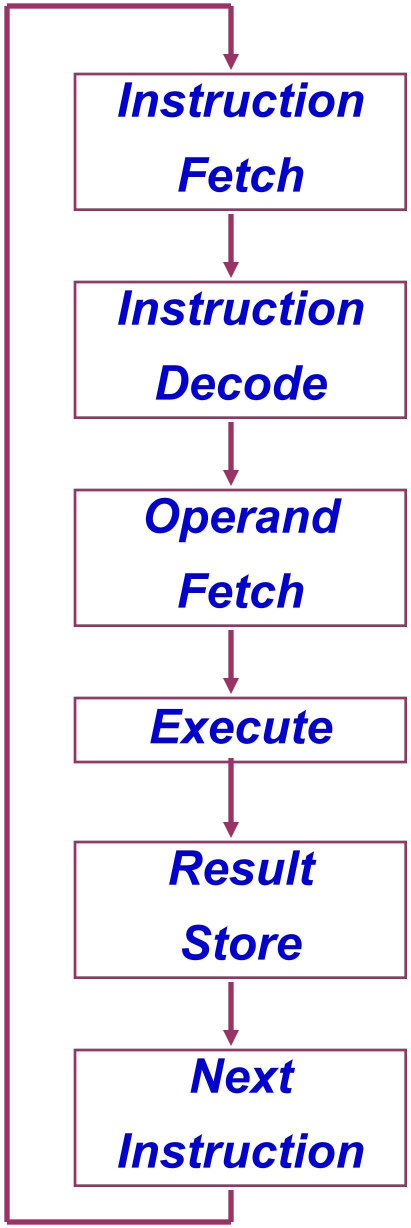
- MIPS Registers

Register Number	Mnemonic Name	Conventional Use	Register Number	Mnemonic Name	Conventional Use
$0	$zero	Permanently 0	$24, $25	$t8, $t9	Temporary
$1	$at	Assembler Temporary (reserved)	$26, $27	$k0, $k1	Kernel (reserved for OS)
$2, $3	$v0, $v1	Value returned by a subroutine	$28	$gp	Global Pointer
$4–$7	$a0–$a3	Arguments to a subroutine	$29	$sp	Stack Pointer
$8–$15	$t0–$t7	Temporary (not preserved across a function call)	$30	$fp	Frame Pointer
$16–$23	$s0–$s7	Saved registers (preserved across a function call)	$31	$ra	Return Address

Reminder: Von-Neumann Model

- Stored Program
  - Instructions stored in a linear memory array
- Sequential Instruction Processing
  1. Program counter identifies current instruction
  2. Instructions fetched one by one from memory
  3. Once fetched, instruction is executed
  4. Results stored in memory
  5. Program counter incremented
  6. Return to step 1

Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Micro-Architecture

- BIG Picture
  - Basic blocks
    - Components need to execute Von-Neumann algorithm

Micro-Architecture

- Basic Blocks of a Micro-Architecture
  - A high-speed unit to keep code & data
    - CPU runs very fast but memory is slow
    - Cache memory (instruction & data cache)
  - A unit to fetch instructions from cache
    - Instruction fetch unit (IFU)
    - Instructions transferred from I-cache to IFU
  - A unit to decode instructions after fetch process
    - Instruction decoder unit

Micro-Architecture (cont.)

- Basic Blocks of a Micro-Architecture
  - A unit to execute instructions
    - Execution unit
  - A unit to do arithmetic/logical operations
    - ALU
  - A unit to execute branch instruction
    - Branch unit
  - A unit to execute load/store instructions
    - Load/store unit
    - LSU ⇔ D-cache

Micro-Architecture (cont.)

- Basic Blocks of a Micro-Architecture
  - A unit to save temporary results within processor
    - Register file
  - A unit to locate next instruction
    - Program counter
  - A unit to schedule all data movements
    - Control unit


Thanks for Your Attention!