Welcome to CS 2323

Computer Architecture

Sadhana Jha
PhD: IIT Kharagpur

Course Logistics

- Classes will be conducted on Google meet
- Will post
  - Course material, lecture videos, Assignments, quizzes, etc will be posted in Google classroom
- Be active in the classroom group
- Help each other, ask questions

Course Logistics

- Pre-requisites:
  - CS1353: Introduction to Data structure
  - ID1303: Introduction to Programming

- Mostly: Computer Organization and Design, The hardware/Software Interface 5th Edition (MIPS)
  - Authors:
    - David A. Patterson
    - Joh L. Hennessy

Course Component

- Assignments:
  - 2- Programming assignments
- Periodic quizzes/Tutorials during class hours
- Tentative scoring policy
  - Attendance: 10% (if issues in joining, please drop a mail before the class, they will mark it)
  - Coding assignments: 60% (30% + 30%), Quizzes/Tutorial: 30%
  - No final exam

Course topics

- Computer abstraction and technology
- Instructions: Language of the computer
- Arithmetic for Computers
- The processor
- Large and Fast: Exploiting Memory Hierarchy
- Parallel processors from Client to Cloud

MK®

COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

FOURTH EDITION 4

Chapter 1

Computer Abstractions and Technology

Thanks to Authors & MK for the slides

Classes of Computers

- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff

Intel i3-35K, i5-50K, i7-80K (Rs)

- Server computers
  - Network based
  - Super computers: High capacity, performance, reliability
  - Cloud computing: Range from 10’s of servers to 100K servers (E.g., DCs built by Google, Amazon..)

Super computer: >100 crores, 1000’s of processors, Terabytes of memory

- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

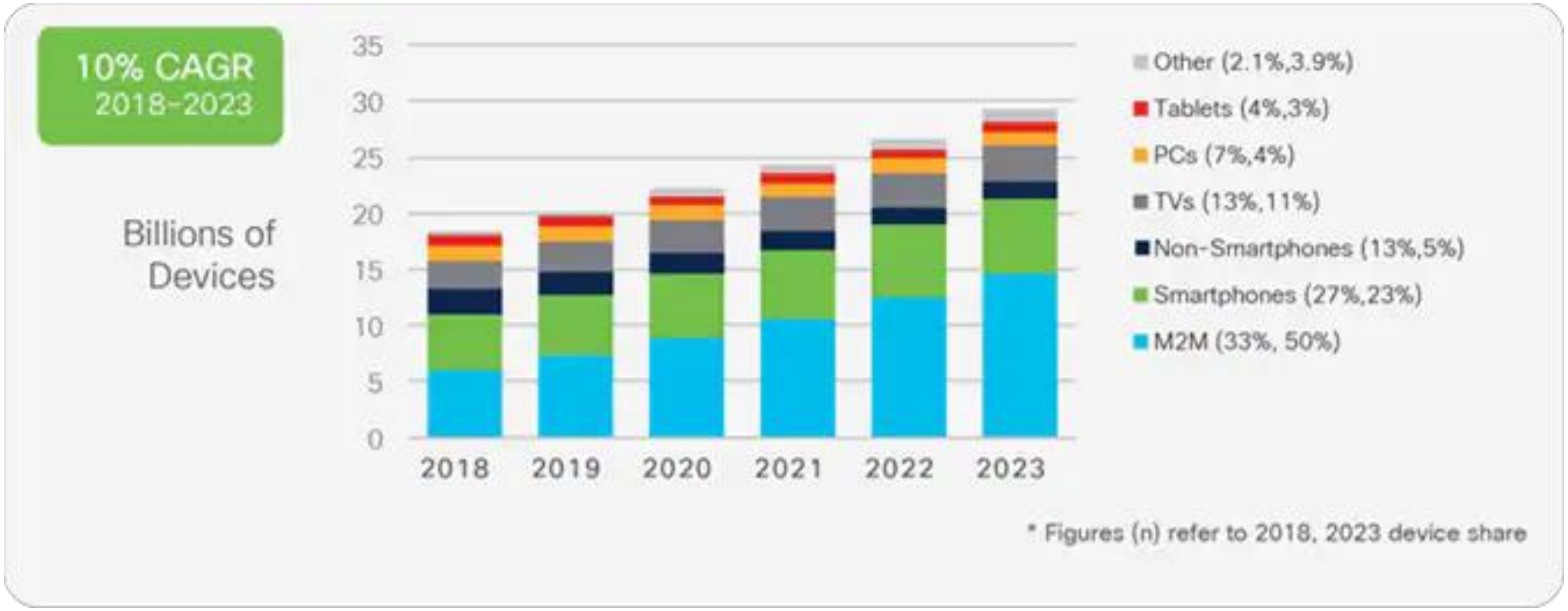
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

What You Will Learn

- How programs are translated into the machine language
    - And how the hardware executes them
- The hardware/software interface
- What determines program performance
    - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

The Processor Market

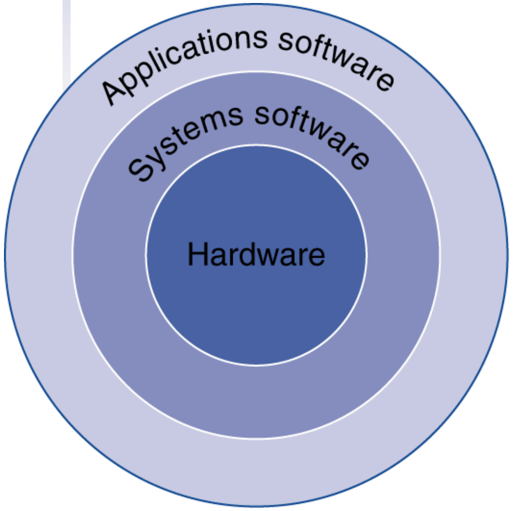


Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Below Your Program

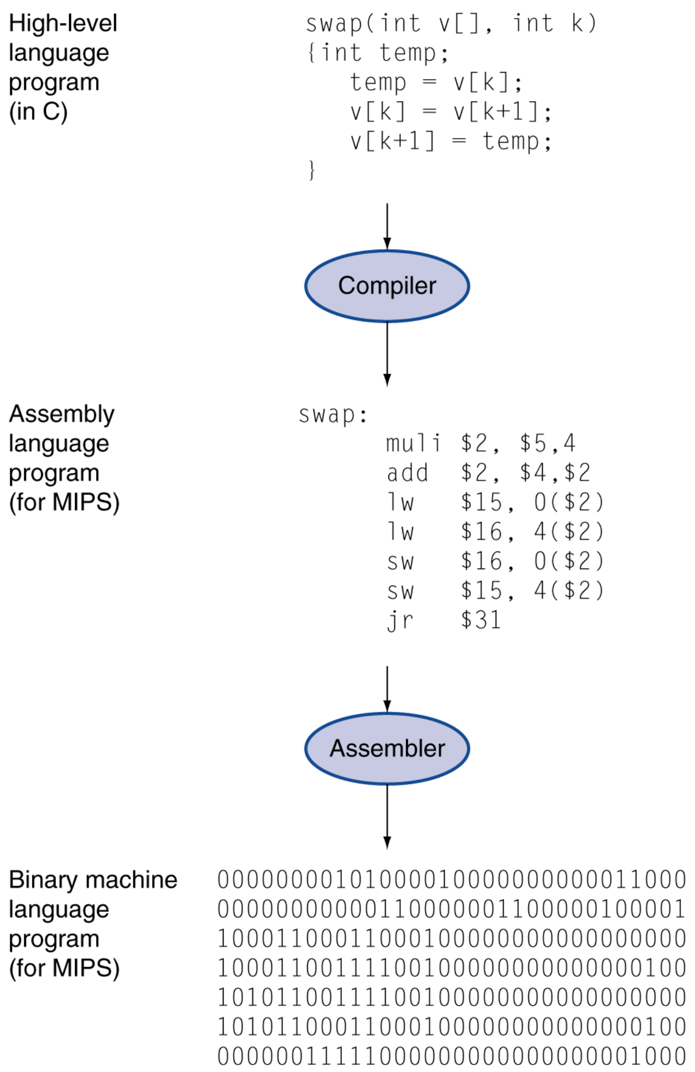
§1.2 Below Your Program



- Application software
  - Written in high-level language
  - Java, Python, C, C++
- System software
  - Compiler: translates HLL code to machine code
    - Ex: GCC
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
    - Ex: Linux, Windows
- Hardware
  - Processor, memory, I/O controllers
  - Ex: Intel i3-7, ARM, DRAM

Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

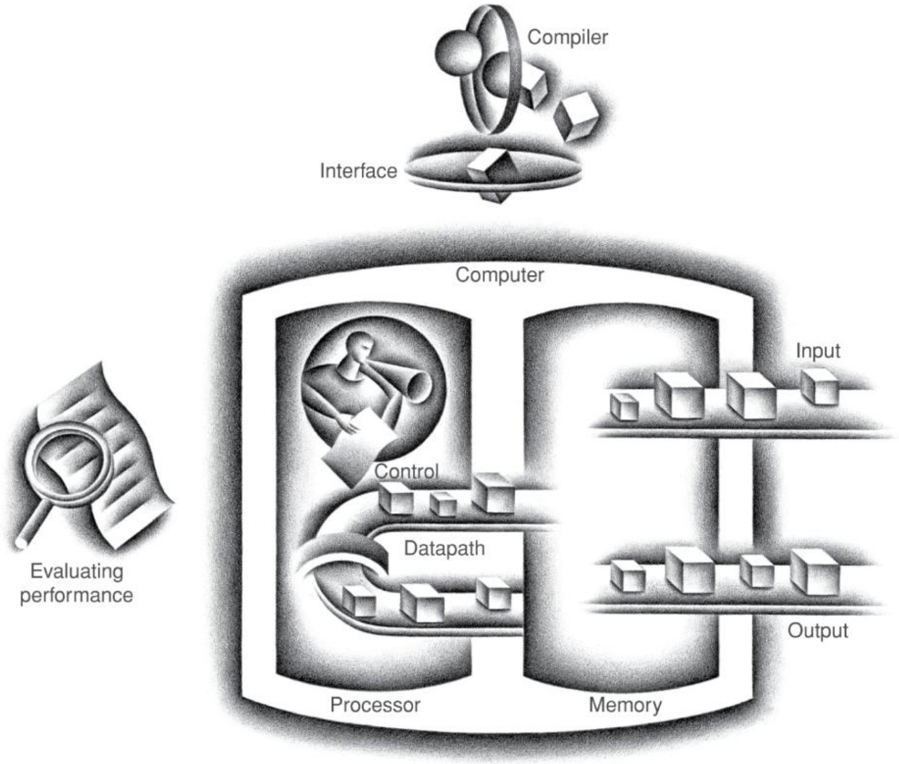
Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Components of a Computer

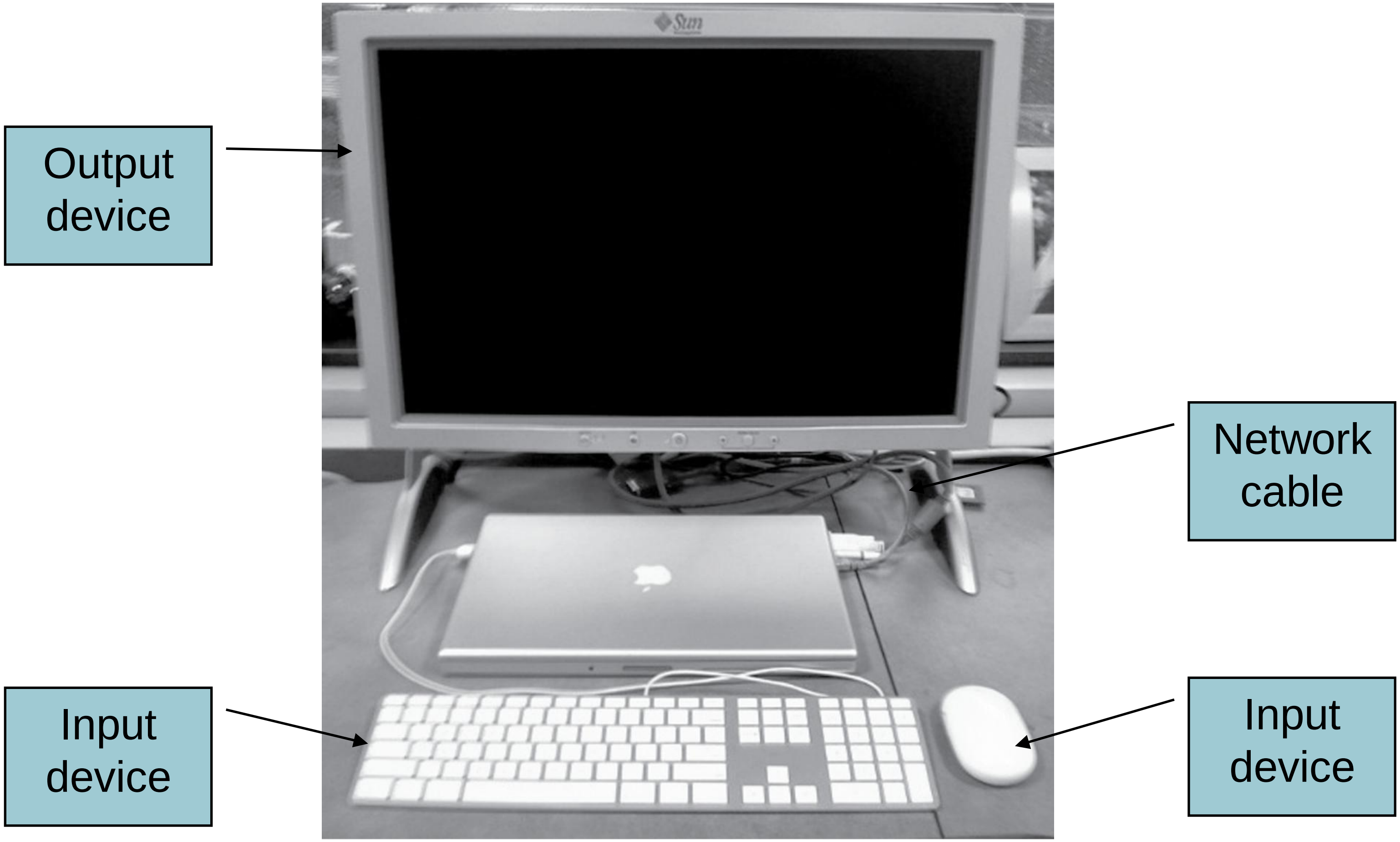
§1.3 Under the Covers

The BIG Picture



- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Anatomy of a Computer



Output device

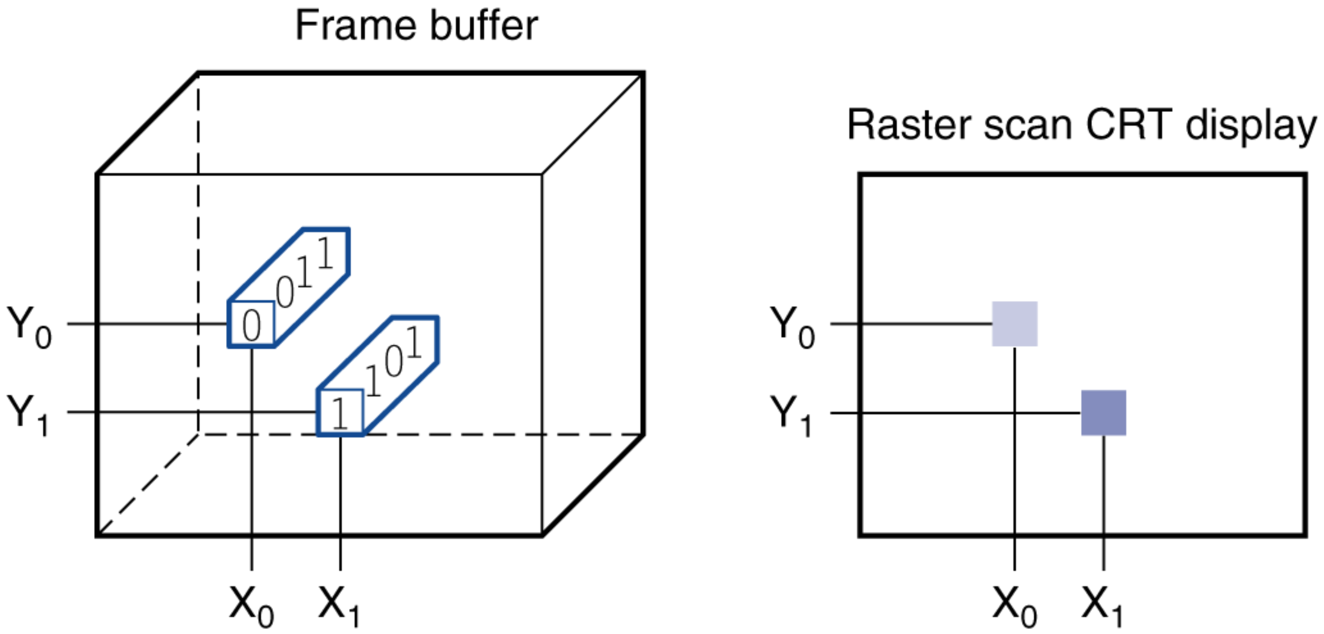
Network cable

Input device

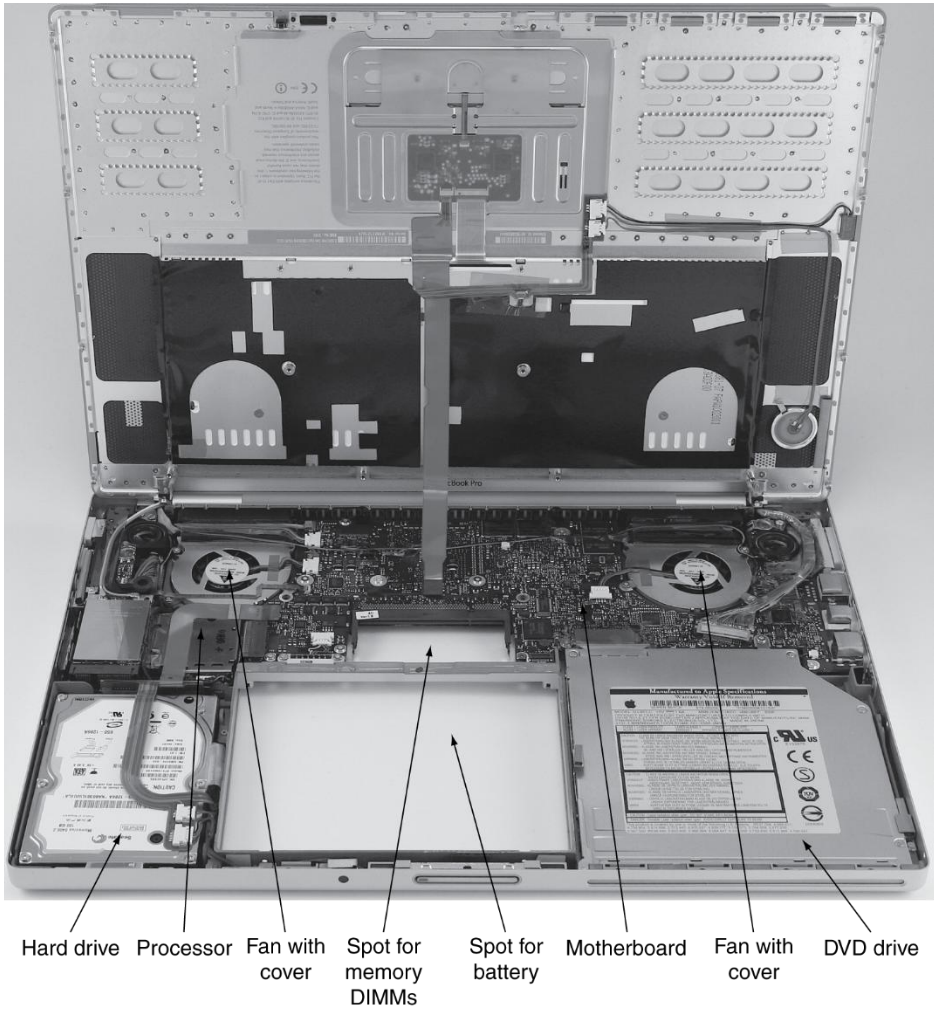
Input device

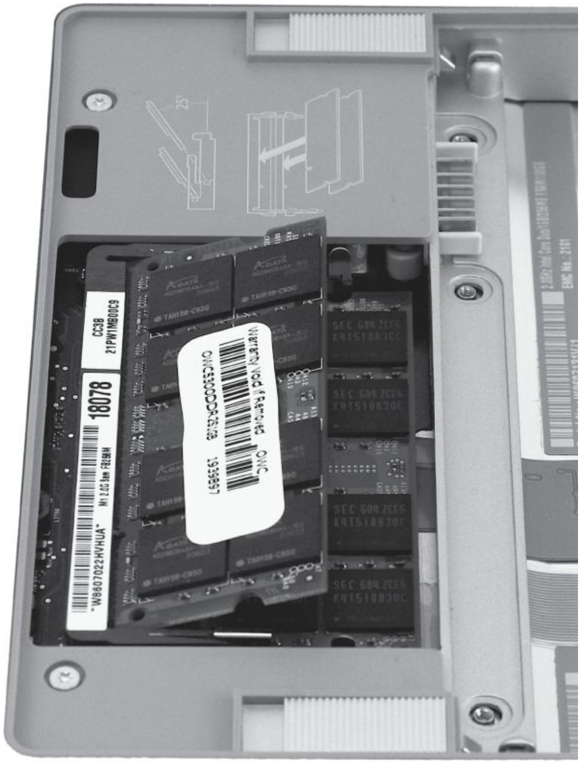
Through the Looking Glass

- LCD screen: picture elements (pixels)
  - Mirrors content of frame buffer memory



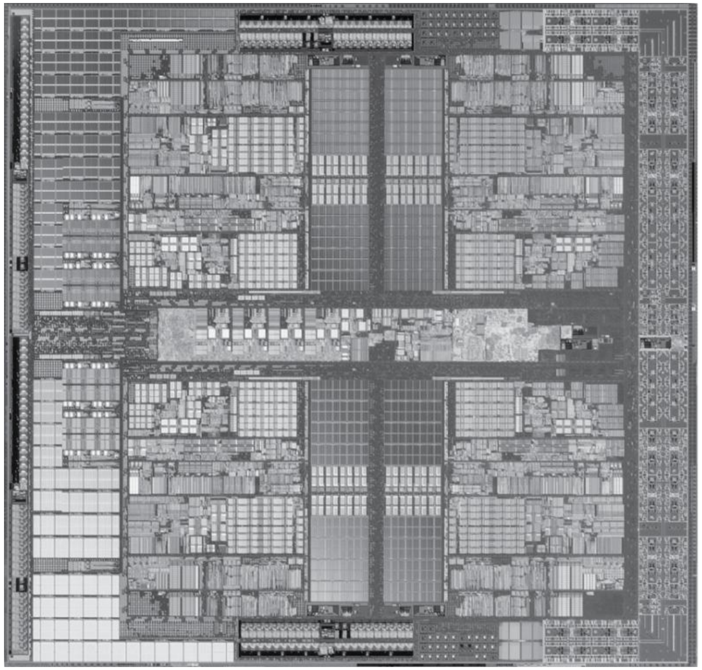
Opening the Box

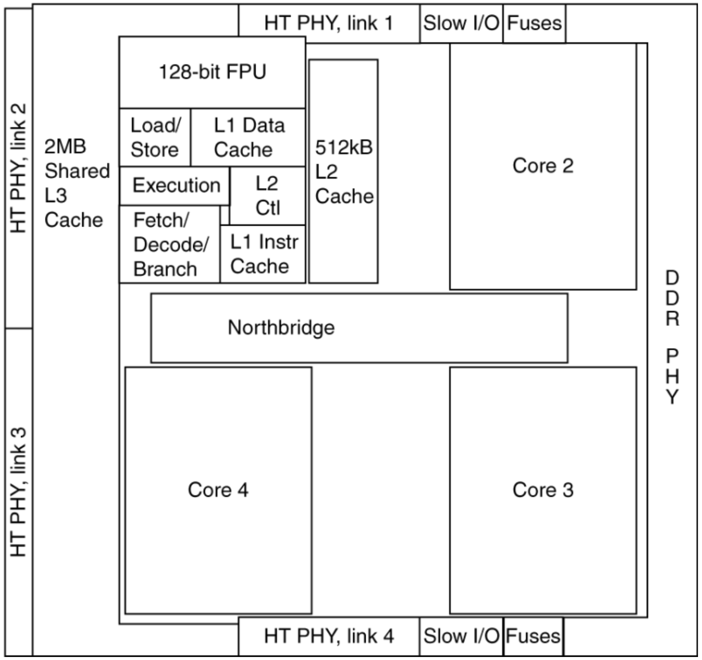




Inside the Processor

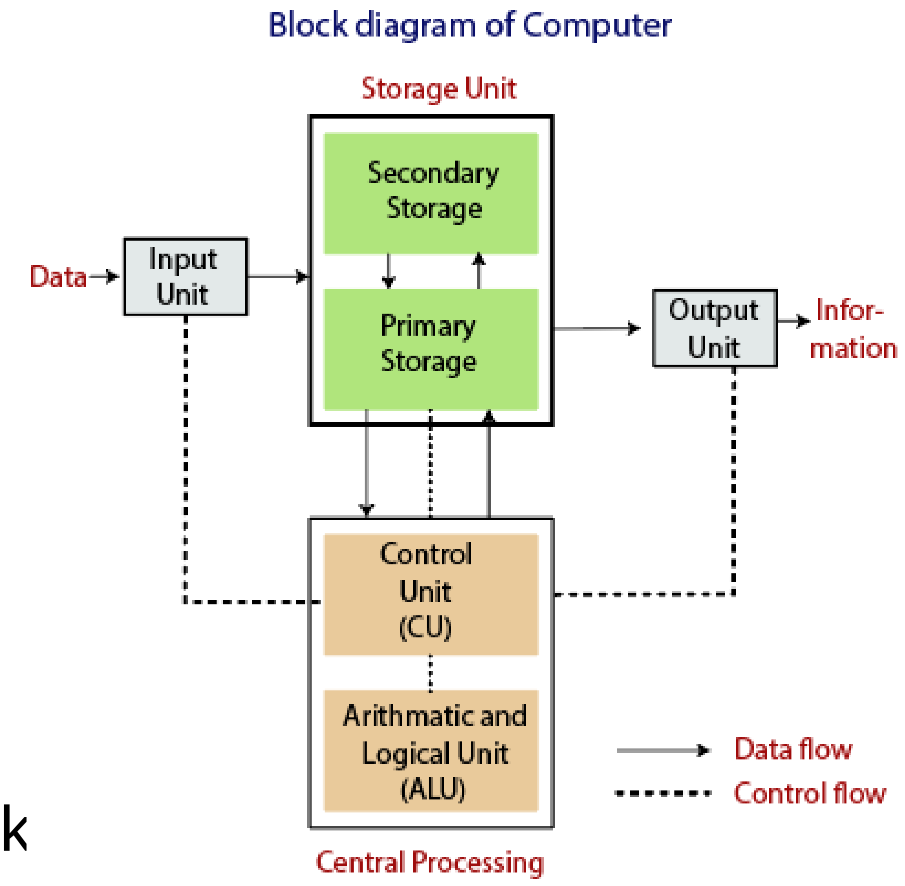
- AMD Barcelona: 4 processor cores





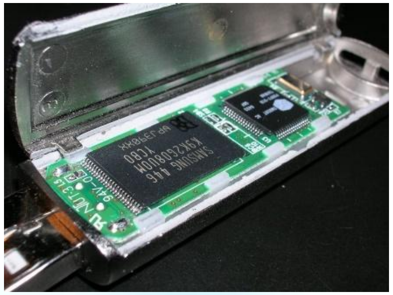
Block diagram of computer

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data
- Primary storage – DRAM
  - Large
- Secondary storage – Hard disk



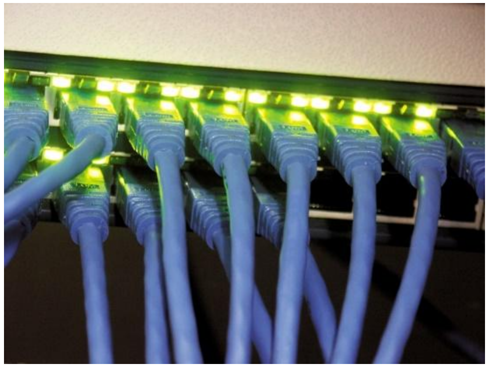
A Safe Place for Data

- Volatile main/primary memory
  - Loses instructions and data when power off
  - Cache (SRAM), DRAM
- Non-volatile secondary memory
  - Magnetic disk (also hard disk)
  - Flash memory (also SSD)
  - Optical disk (CDROM, DVD)



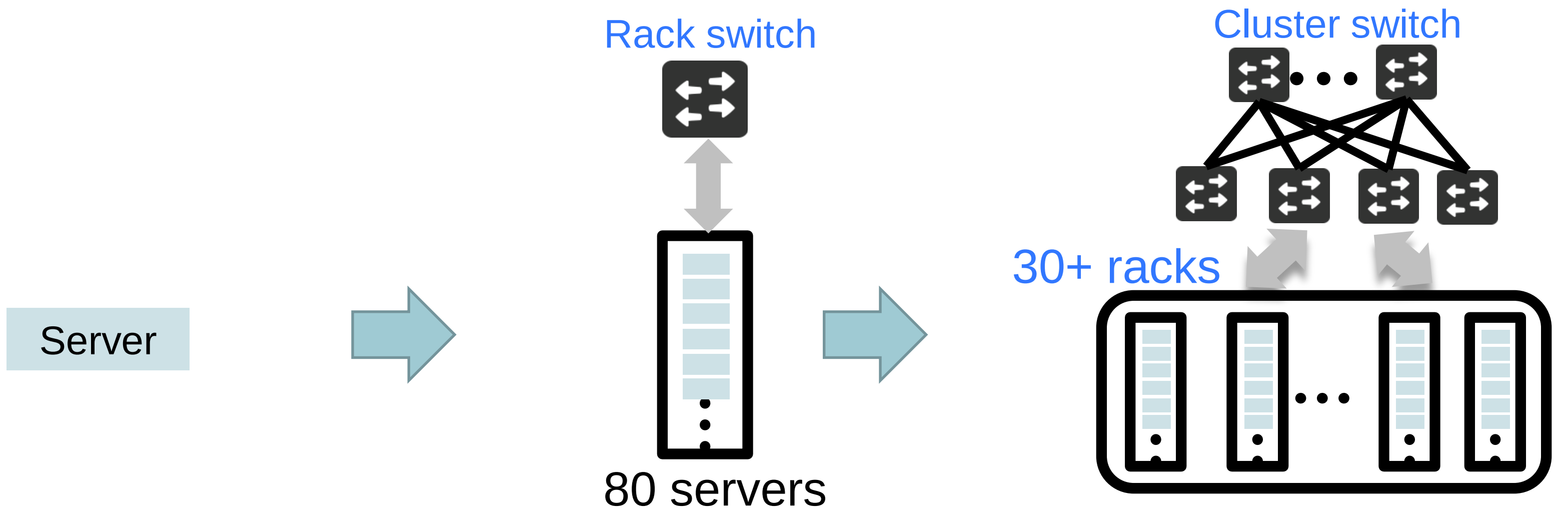
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN: the Internet)
- Wireless network: WiFi, Bluetooth



Networks enable compute and storage at massive scales

The network has thousands of switches and millions of links
(Datacenters built by Google, facebook, Amazon, MS..)



- NIC: 10Gbps
- CPUs : 64
- DRAM: 16GB, 100ns
- Disk: 2TB, 10ms

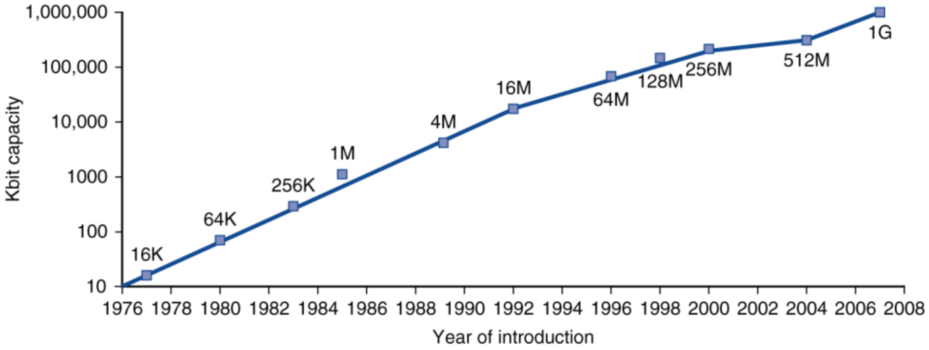
- Bandwidth: 800Gbps
- CPUs: 5K
- DRAM: 1TB, 300us
- Disk: 160TB, 11ms

- Bandwidth: 24Tbps
- CPUs : 153K
- DRAM: 30TB, 500us
- Disk: 4.80PB, 12ms

Reference: Jeff Dean. Designs, Lessons and Advice from Building Large Distributed Systems.

Technology Trends

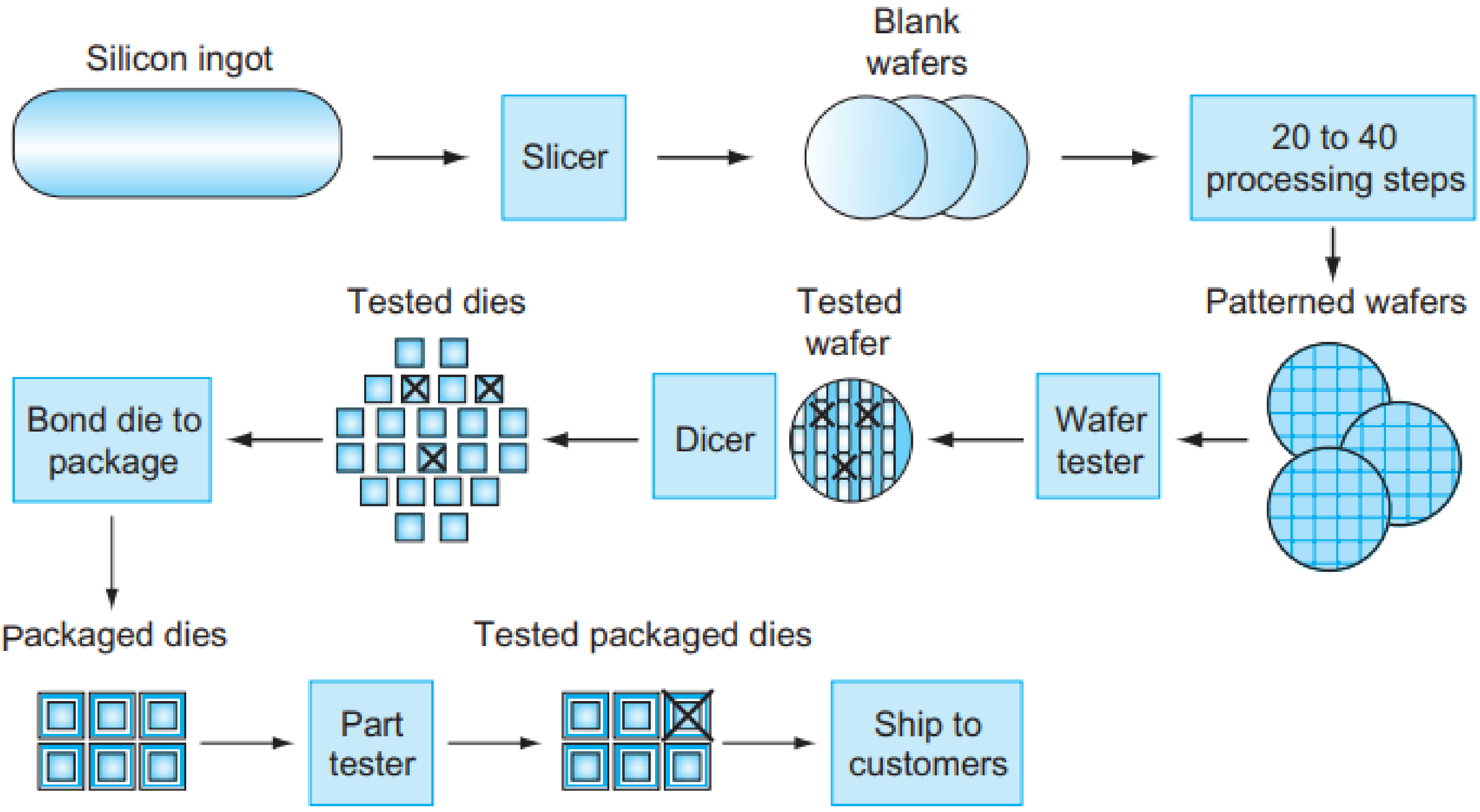
- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



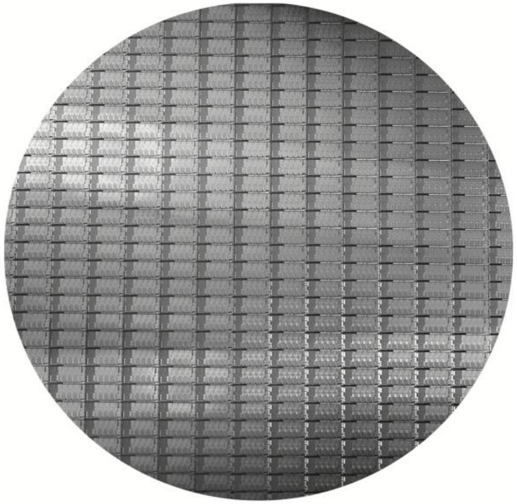
DRAM capacity

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

Chip Manufacturing Process



Intel core i7



- 300mm (12 inch) wafer
- 100% yields = 280 chips, 32nm technology

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

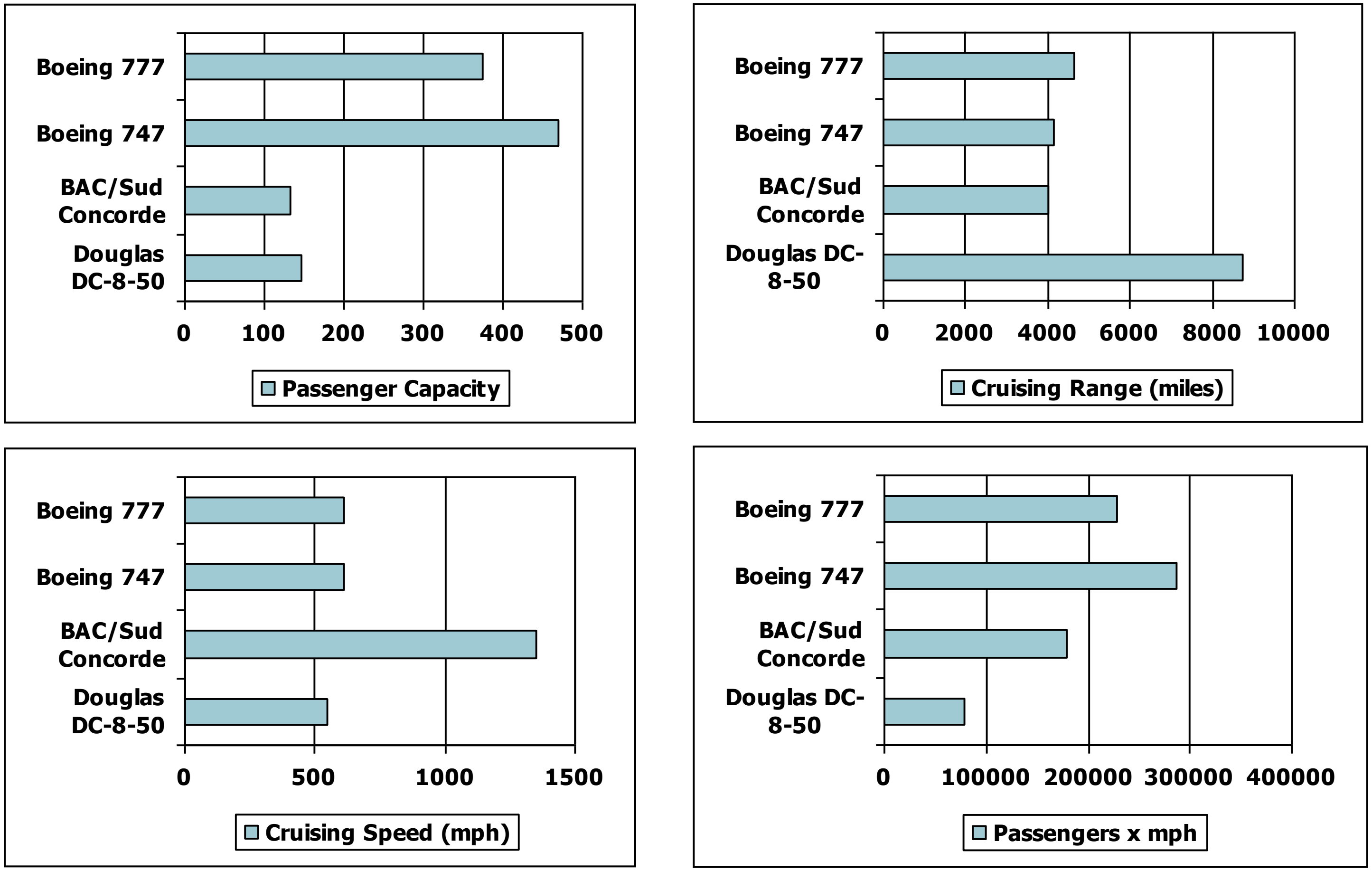
$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

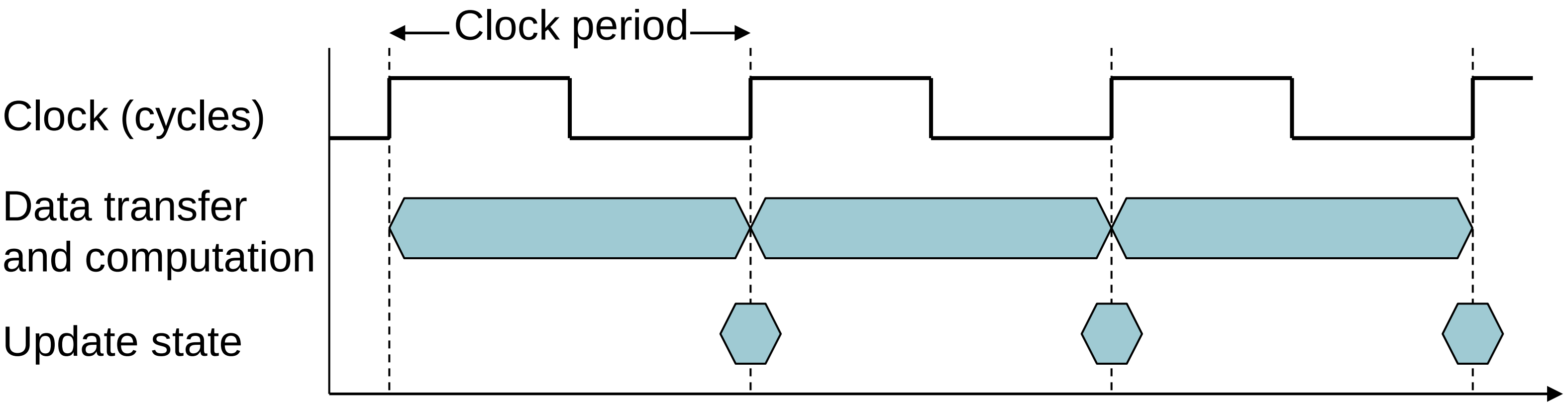
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10$^{–12}$s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$
\begin{aligned}
\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\
&= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps}
\end{aligned}
$$

A is faster…

$$
\begin{aligned}
\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\
&= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}
\end{aligned}
$$

$$
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{I} \times 600\text{ps}}{\text{I} \times 500\text{ps}} = 1.2
$$

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

  - Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
= 2×1 + 1×2 + 2×3
= 10
  - Avg. CPI = 10/5 = 2.0

- Sequence 2: IC = 6
  - Clock Cycles
= 4×1 + 1×2 + 1×3
= 9
  - Avg. CPI = 9/6 = 1.5

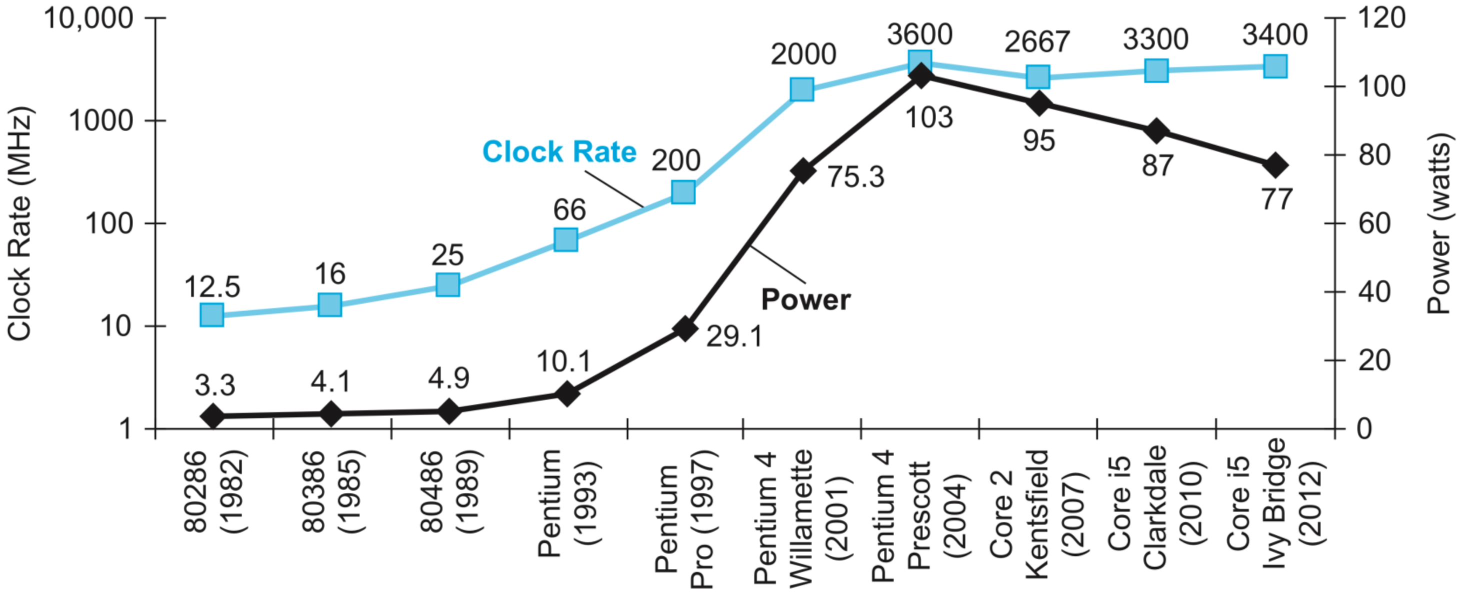
Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30

5V → 1V

×1000

Reducing Power

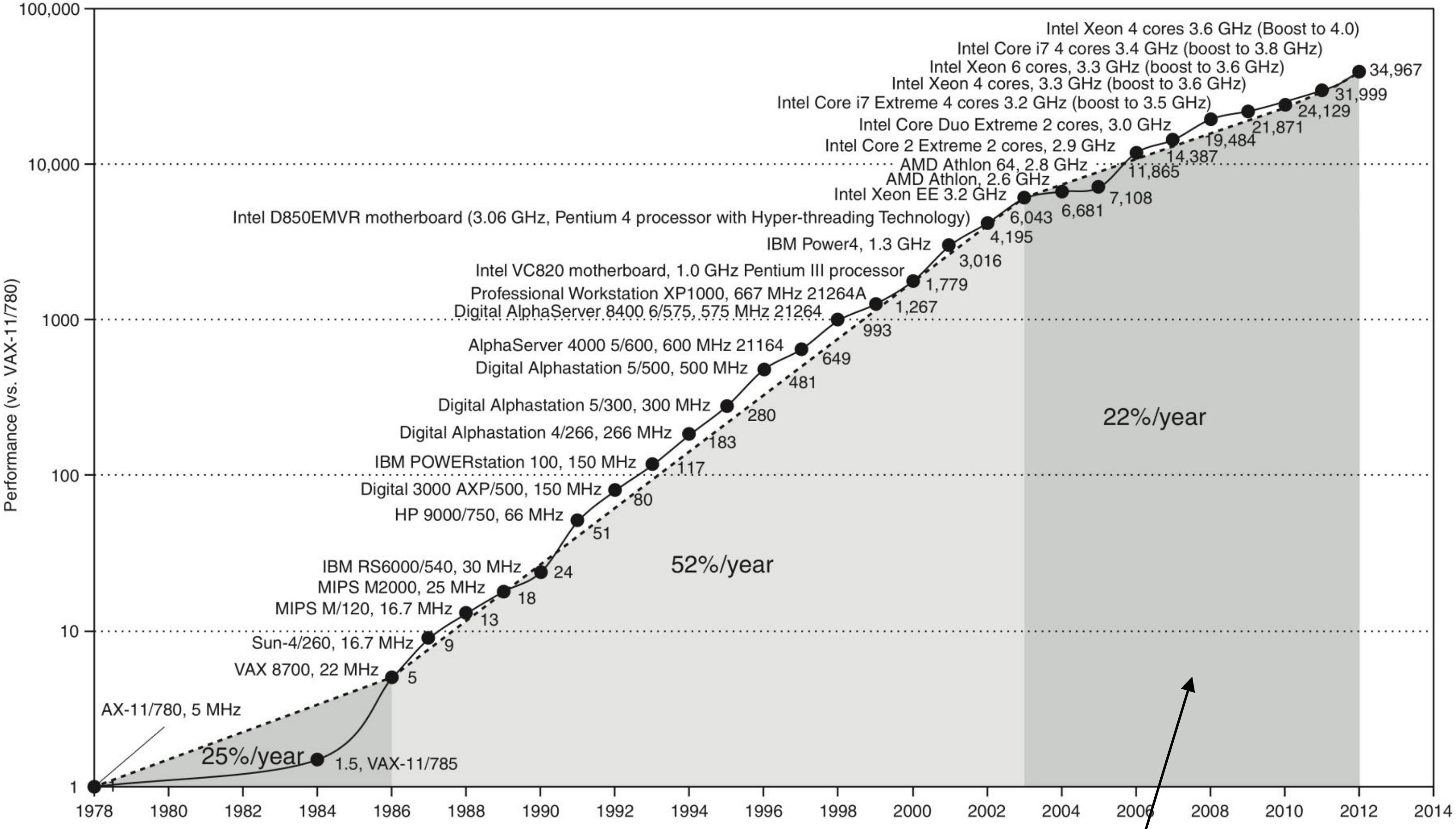
- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^{\;2} \times F_{\text{old}}} = 0.85^4 = 0.52$$

- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat
- How else can we improve performance?

Uniprocessor Performance

§1.6 The Sea Change: The Switch to Multiprocessors

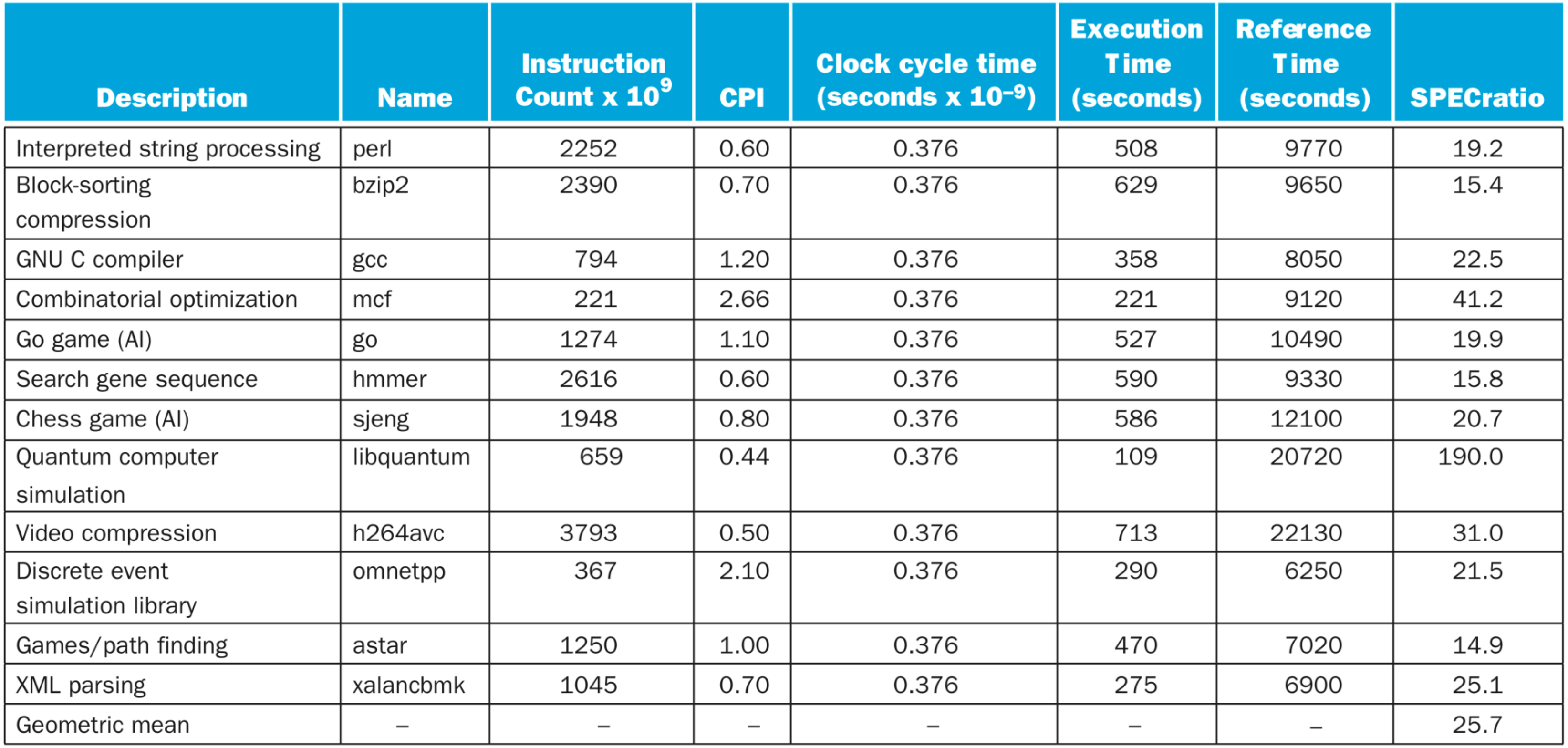


Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Intel core i7 2.66GHz performance



Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Concluding Remarks

§1.9 Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Execution time: the best performance measure (IC x CPI x Clock period)
- Power is a limiting factor
  - Use parallelism to improve performance