COMPUTER ABSTRACTIONS AND TECHNOLOGY

Jo, Heeseung

The Computer Revolution

Progress in computer technology

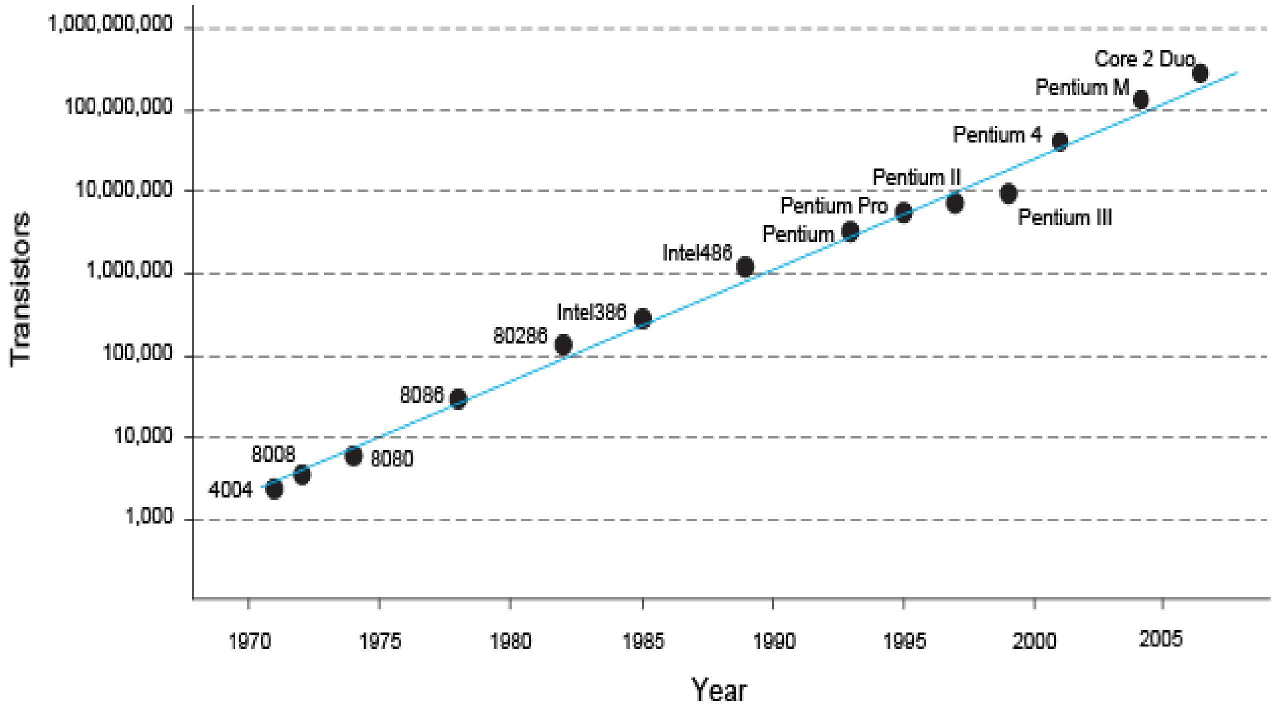
- Underpinned by Moore’s Law

Makes novel applications feasible

- Computers in automobiles
- Cell phones
- Human genome project
- World Wide Web
- Search Engines

Computers are pervasive

Moore's Law



The number of transistors on a computer chip doubles every year (observed in 1965)

Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, 1968.

Classes of Computers

Desktop computers

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

Server computers

- Network based
- High capacity, performance, reliability
- Range from small servers to building sized

Embedded computers

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

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

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

Hardware

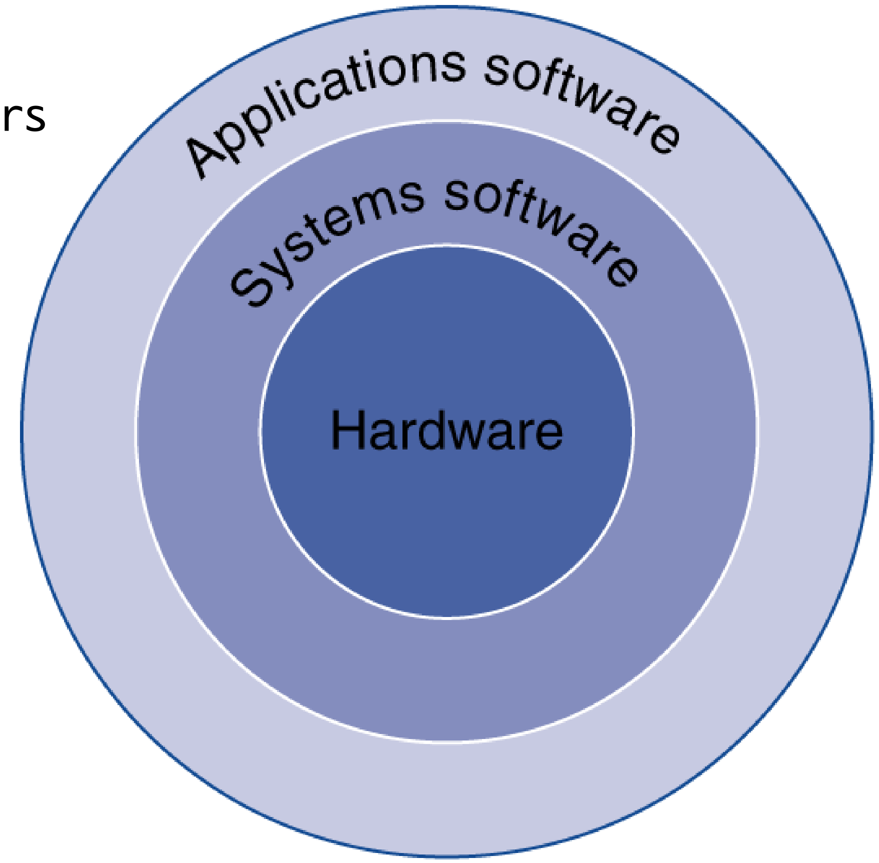
- Processor, memory, I/O controllers

System software

- Compiler: translates HLL code to machine code
- Operating System: service code
  - Handling input/output
  - Managing memory and storage
  - Scheduling tasks & sharing resources

Application software

- Written in high-level language



Levels of Program Code

High-level language

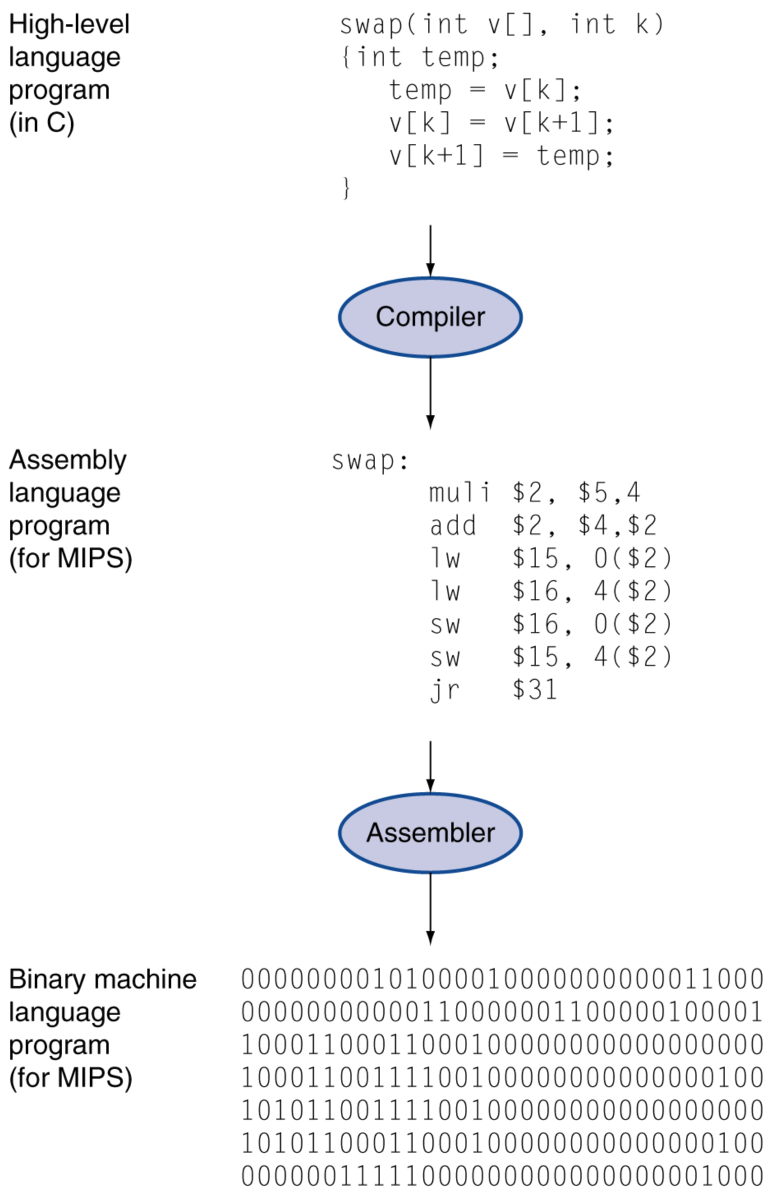
- Level of abstraction closer to problem domain
- Provides for productivity and portability

Assembly language

- Textual representation of instructions

Hardware representation

- Binary digits (bits)
- Encoded instructions and data


High-level language program (in C)
Compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

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

```
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
```

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

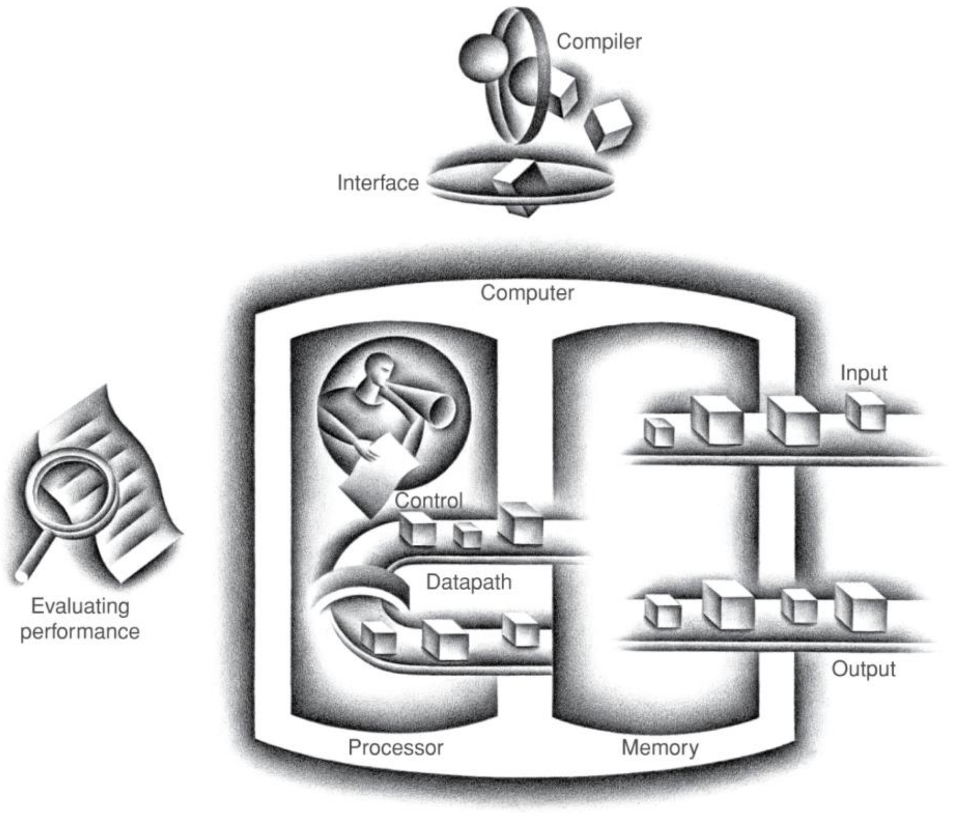
Components of a Computer

Same components for
all kinds of computer

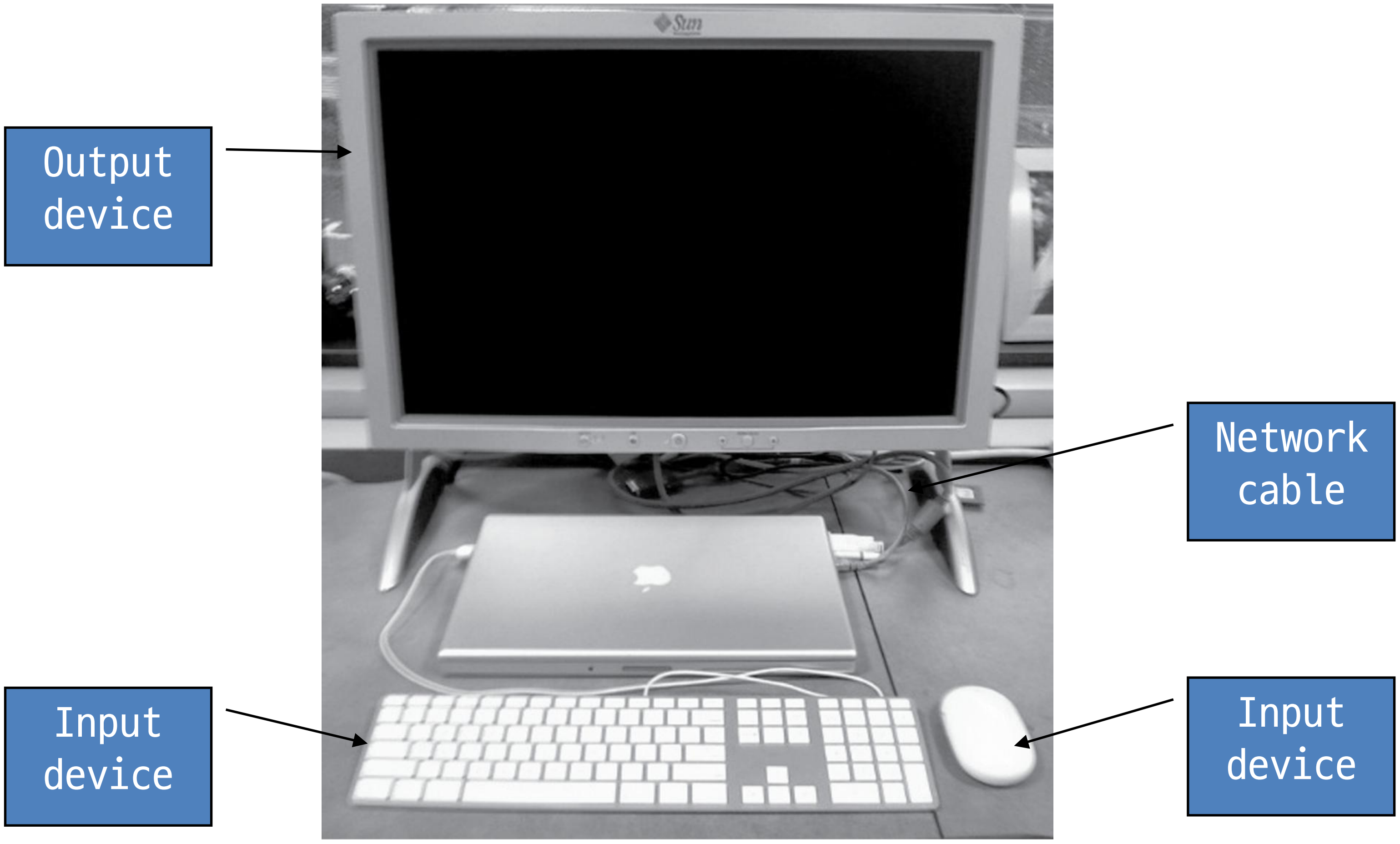
- Desktop, server, embedded

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

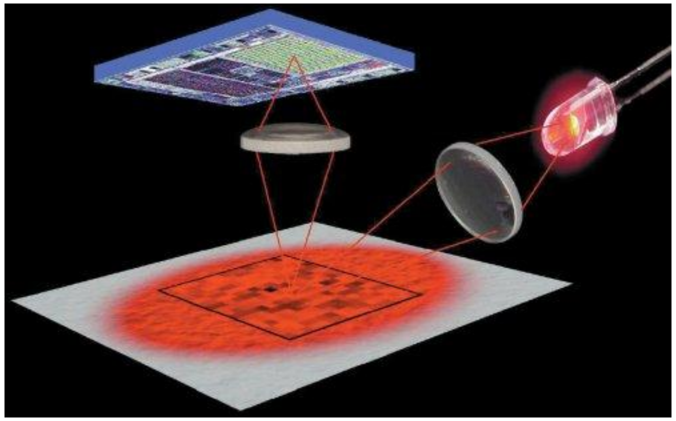


Anatomy of a Mouse

Optical mouse

- Small low-res camera
- Basic image processor
  - Looks for x, y movement
- Buttons & wheel

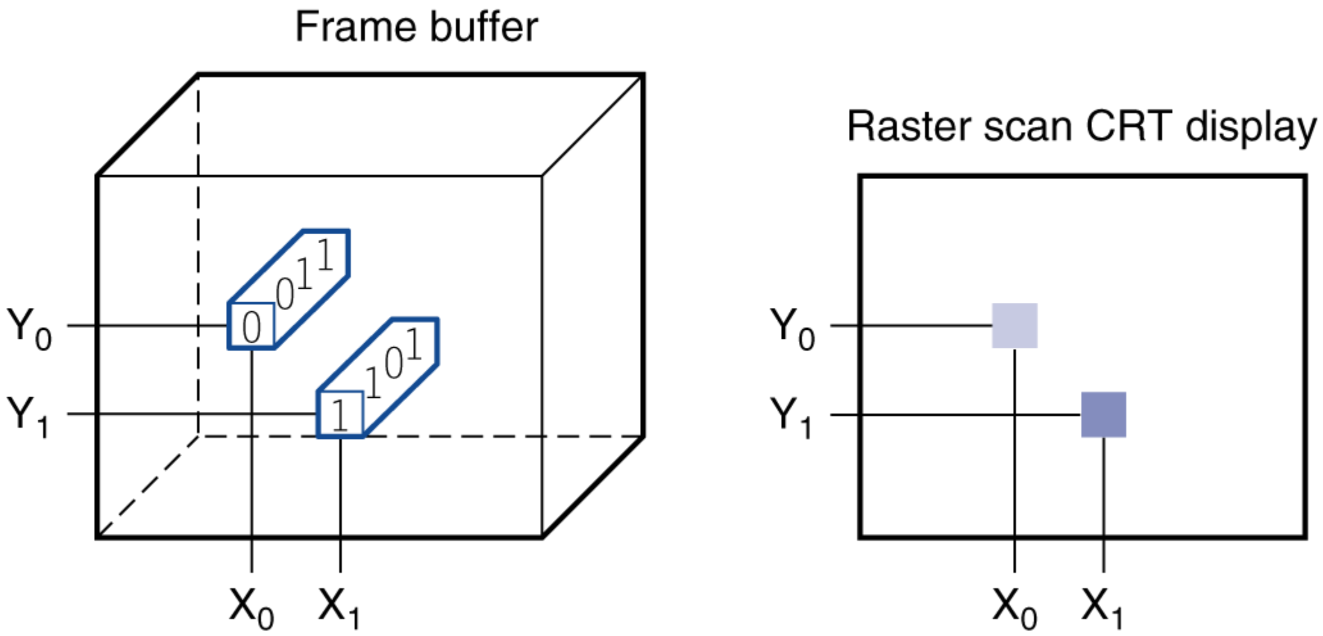
Supersedes roller-ball mechanical mouse



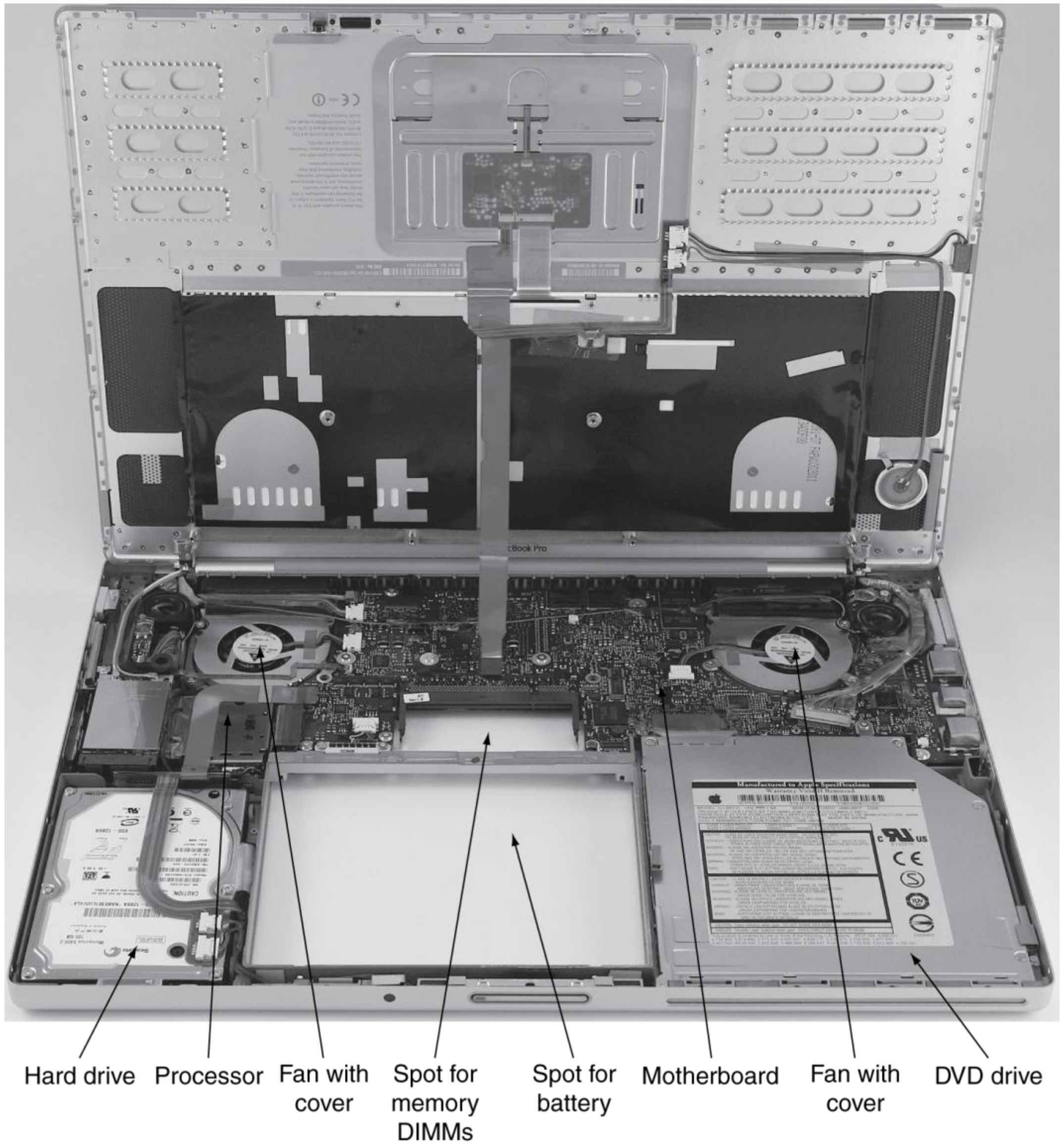
Through the Looking Glass

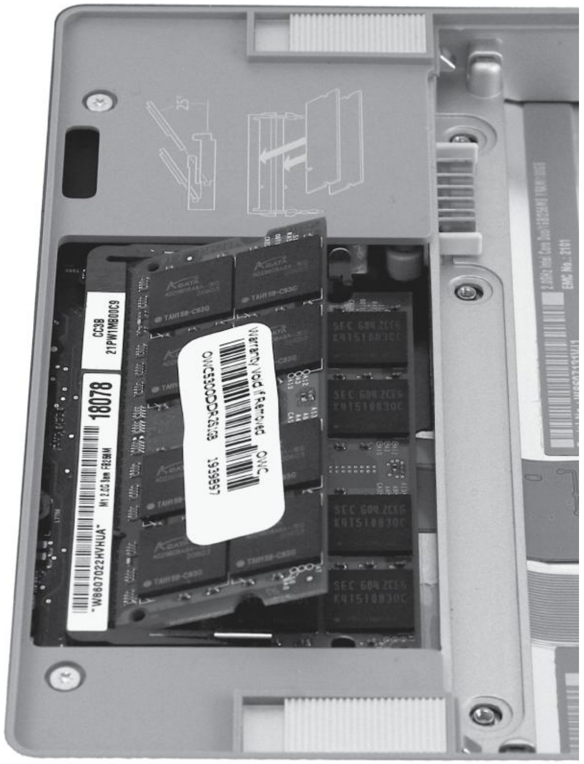
LCD screen: picture elements (pixels)

- Mirrors content of frame buffer memory



Opening the Box





Inside the Processor (CPU)

Datapath: performs operations on data

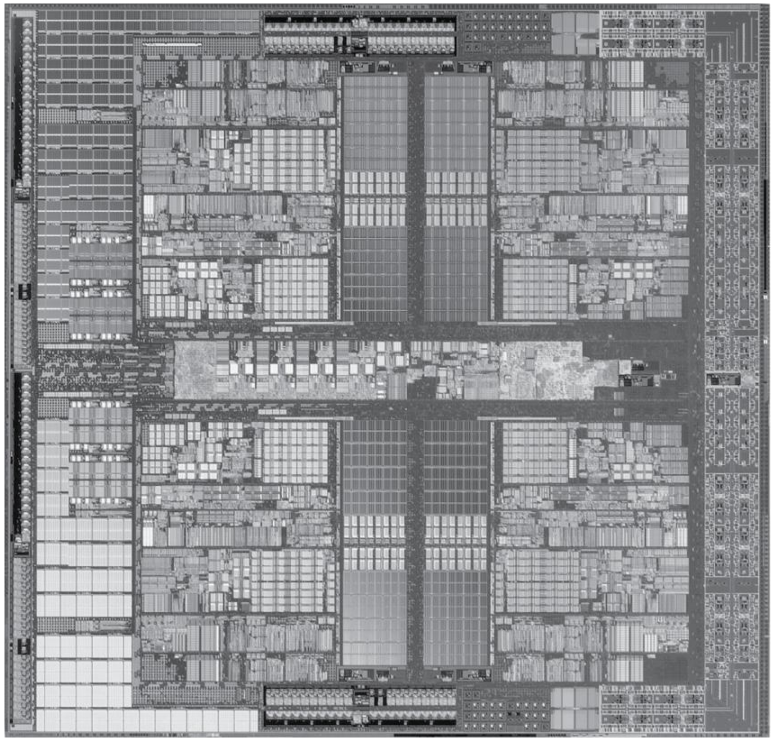
Control: sequences datapath, memory, ...

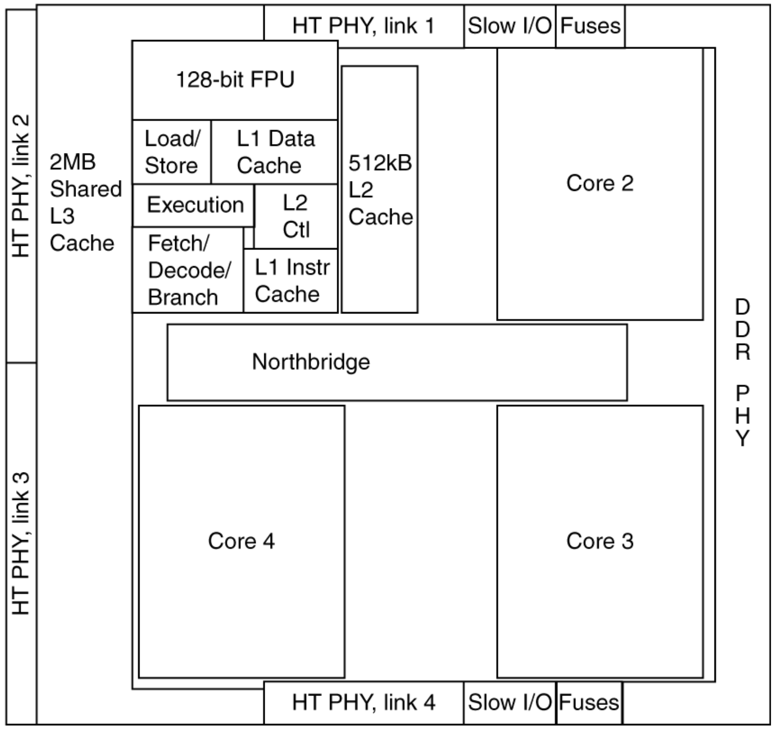
Cache memory

- Small fast SRAM memory for immediate access to data

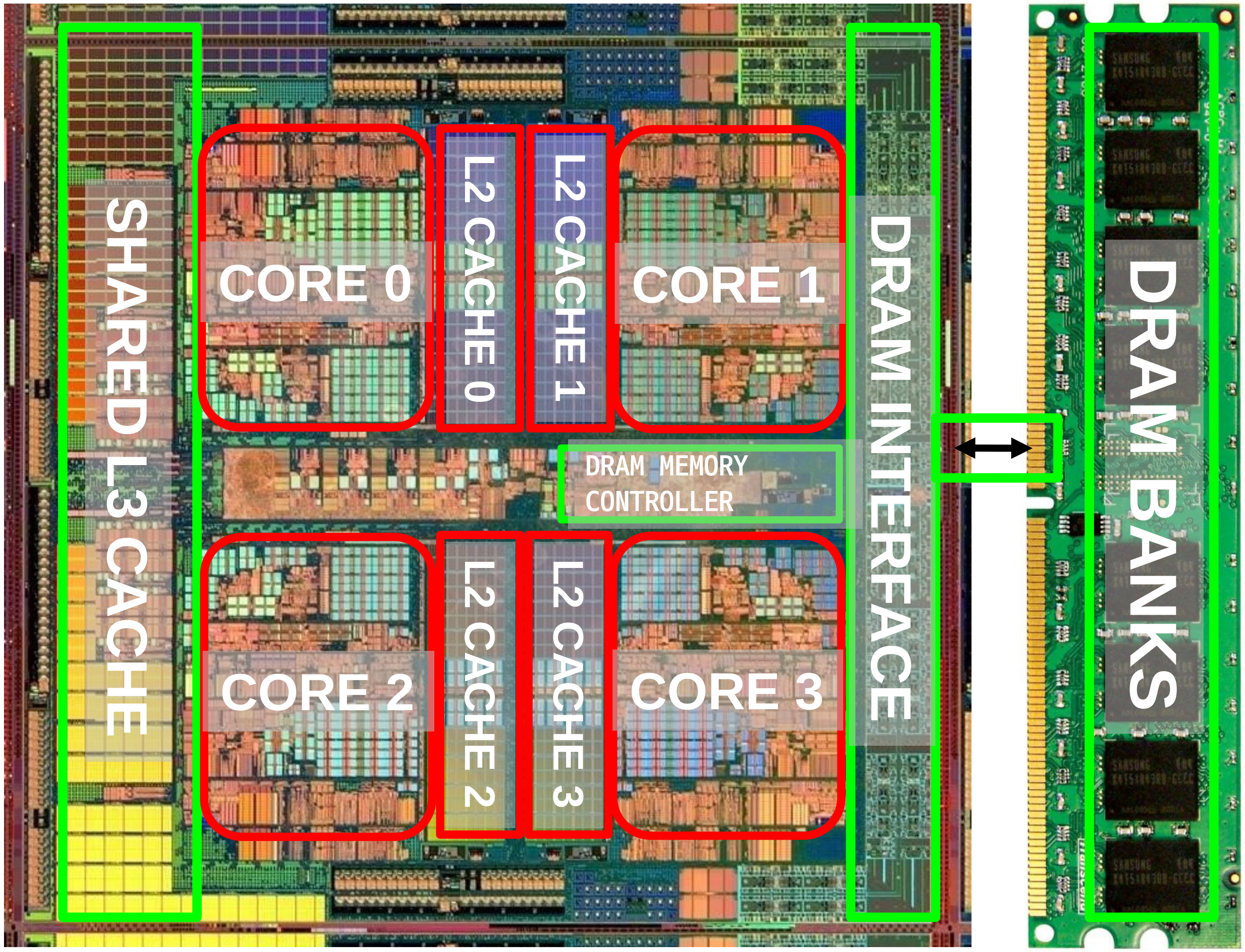
Inside the Processor (CPU)

AMD Barcelona: 4 processor cores





AMD Barcelona: 4 processor cores



*Die photo credit: AMD Barcelona

Abstractions

Abstraction helps us deal with complexity

- Hide lower-level detail

Instruction set architecture (ISA)

- The hardware/software interface

Application binary interface

- The ISA plus system software interface

Implementation

- The details underlying and interface

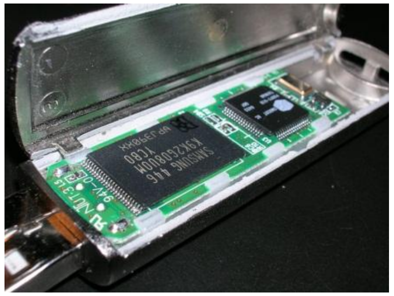
A Safe Place for Data

Volatile main memory

- Loses instructions and data when power off

Non-volatile secondary memory

- Magnetic disk
- Flash memory
- Optical disk (CDROM, DVD)



Networks

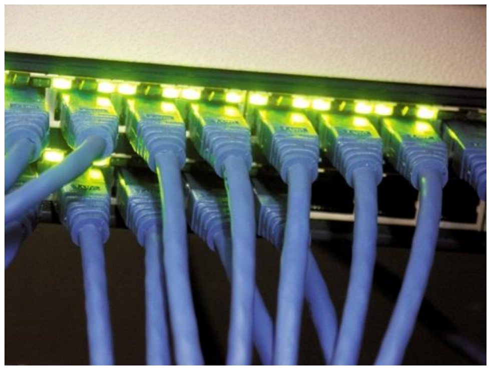
Communication and resource sharing

Local area network (LAN): Ethernet

- Within a building

Wide area network (WAN): the Internet

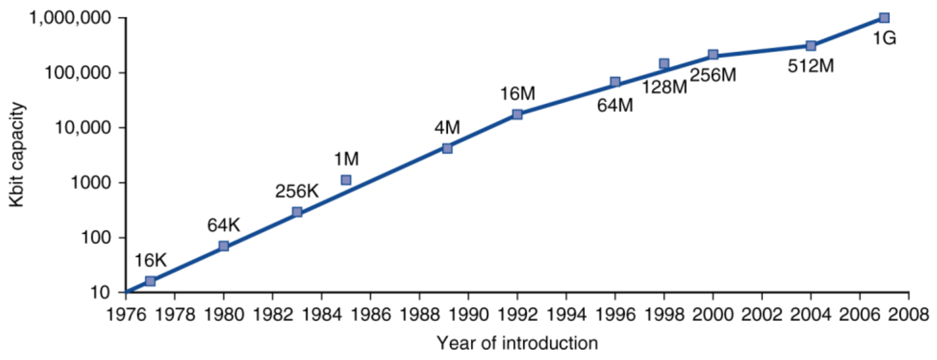
Wireless network: WiFi, Bluetooth



Technology Trends

Electronics technology continues to evolve

- Increased capacity and performance
- Reduced cost

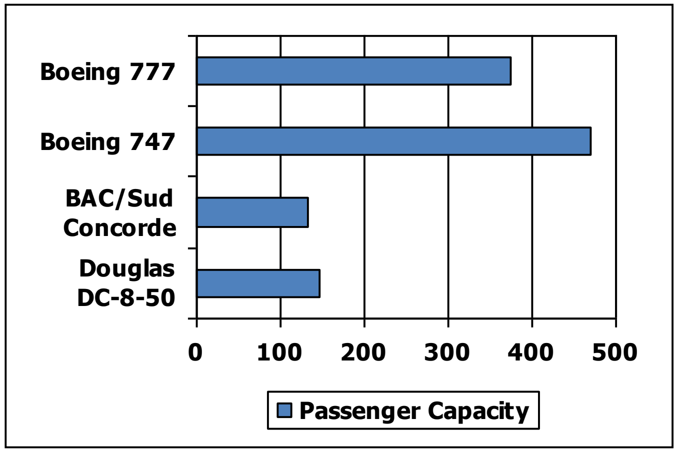


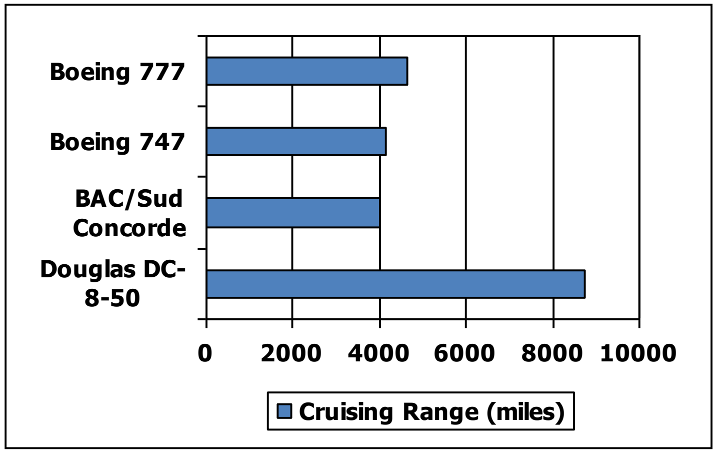
DRAM capacity

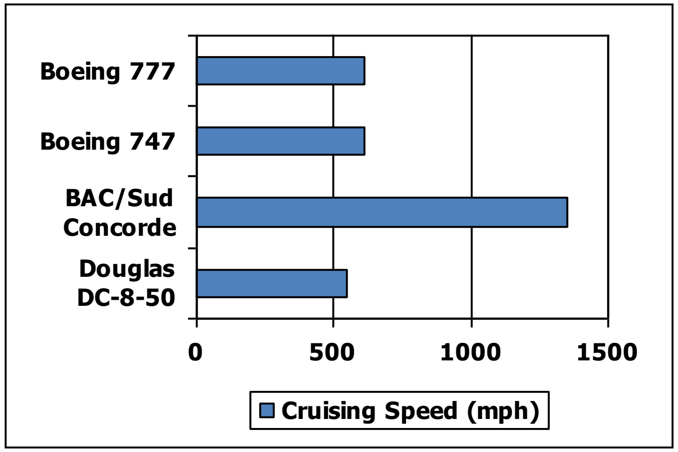
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

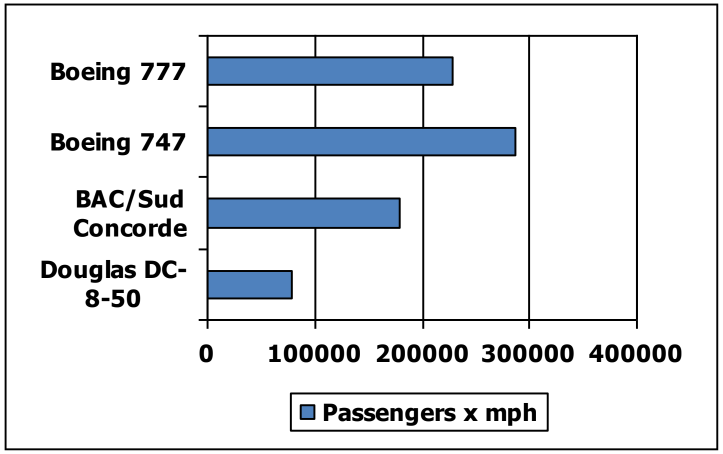
Defining Performance

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 (Wall clock time)

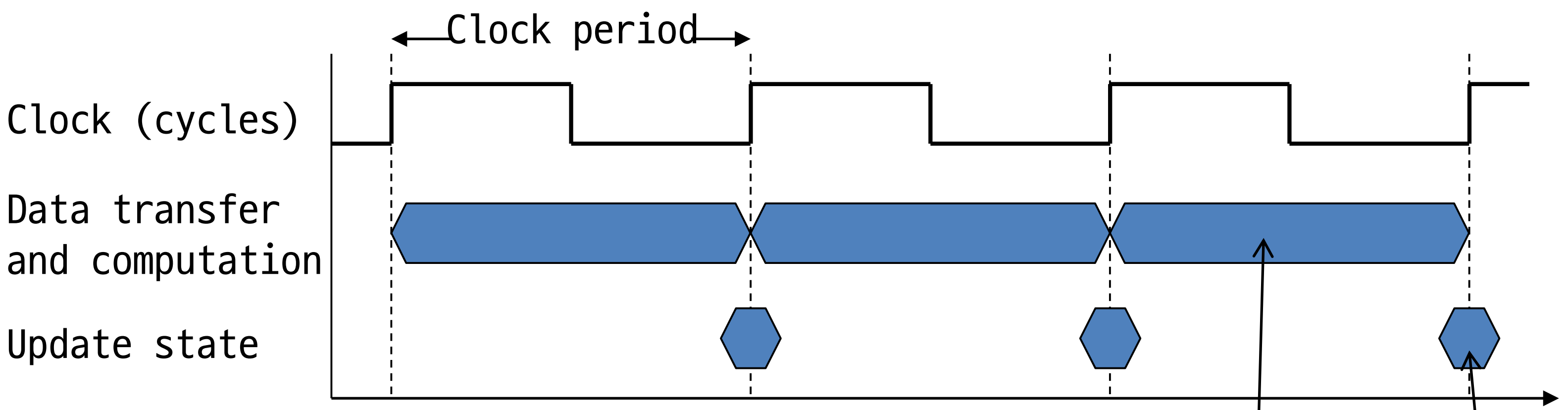
- Total response time, including all aspects
  - Processing, I/O, OS overhead, idle time
- Determines system performance

CPU time

- Time spent processing a given job
  - Discounts I/O time, other jobs' shares
- Total CPU time = user CPU time + 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

ex)

- 1,000,000 CPU clock cycles
- 250ps clock cycle time
- CPU time = 1,000,000 * 250ps = 250,000,000ps
  = 250us = 0.25ms = 0.00025s

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?

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

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

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

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

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

Instruction Count and CPI

Instruction Count for a program

- Determined by program, ISA and compiler

Average cycles per instruction (CPI)

- To execute an instruction ≠ Need a cycle
- Cycles for an instruction are different by instructions
- Determined by CPU hardware
- If different instructions have different CPI
  - Average CPI affected by instruction mix

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

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

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

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

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

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?

$$\text{Clock Cycles} = \text{Instructio n Count} \times \text{Cycles per Instructio n}$$
$$\text{CPU Time} = \text{Instructio n Count} \times \text{CPI} \times \text{Clock Cycle Time}$$
$$= \frac{\text{Instructio n Count} \times \text{CPI}}{\text{Clock Rate}}$$

$$\text{CPU Time}_A = \text{Instructio n Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps}$$

A is faster

$$\text{CPU Time}_B = \text{Instructio n Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$

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

... by this muc

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

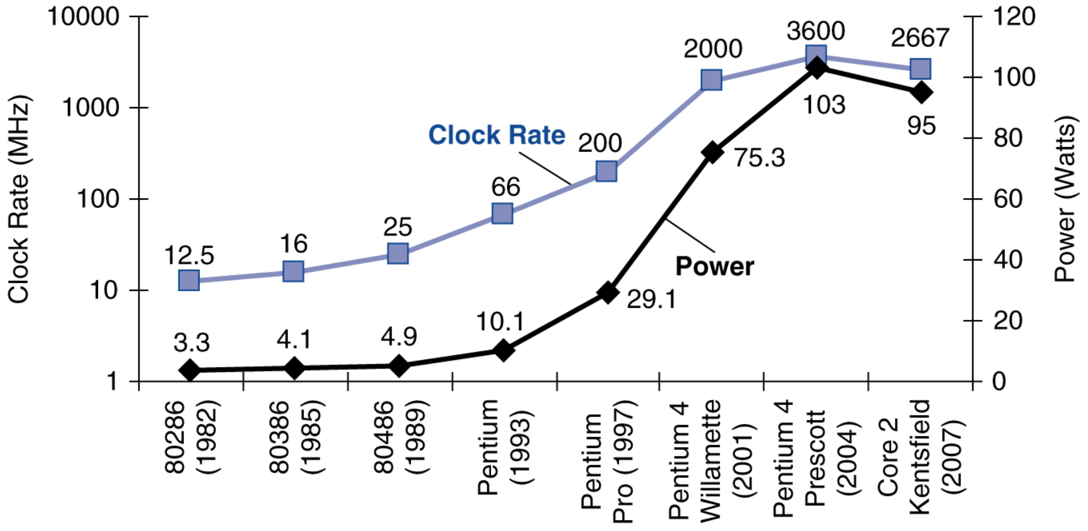
Performance Summary

Performance depends on

- Algorithm affects IC, possibly CPI
- Programming language affects IC, CPI
- Compiler affects IC, CPI
- Instruction set architecture affects IC, CPI, Cycle Time

Power Trends

In CMOS IC technology



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

- Capacitive load (CL) is a type of electrical load that stores electrical energy in a capacitor and releases it back into the circuit

Reducing Power

Suppose a new CPU has

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

- 85% of capacitive load of old CPU
- 15% voltage and 15% frequency reduction

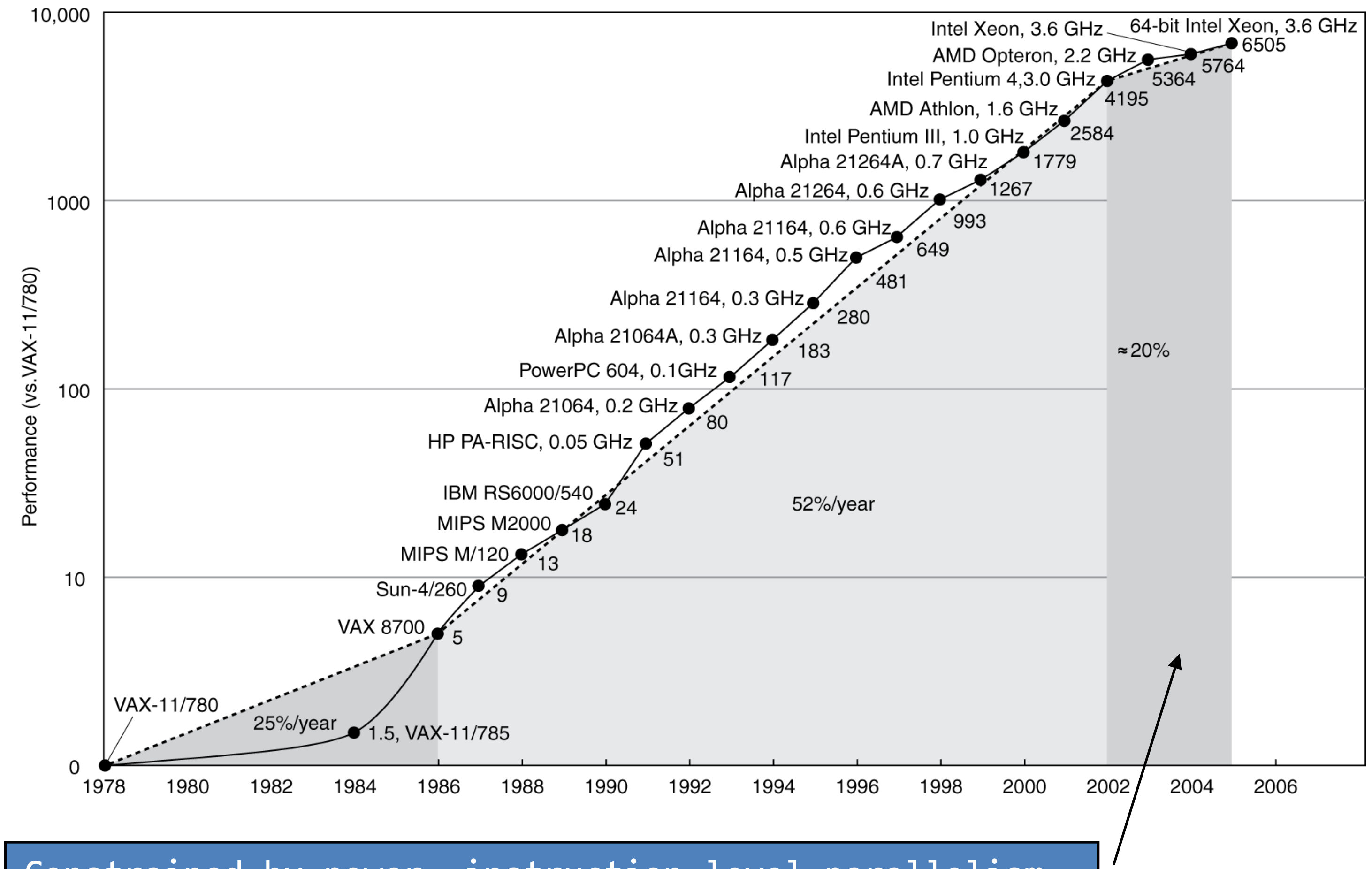
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^2 \times F_{old}} = 0.85^4 = 0.52$$

The power wall

- We can't reduce voltage further
- We can't remove more heat

How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

Multicore microprocessors

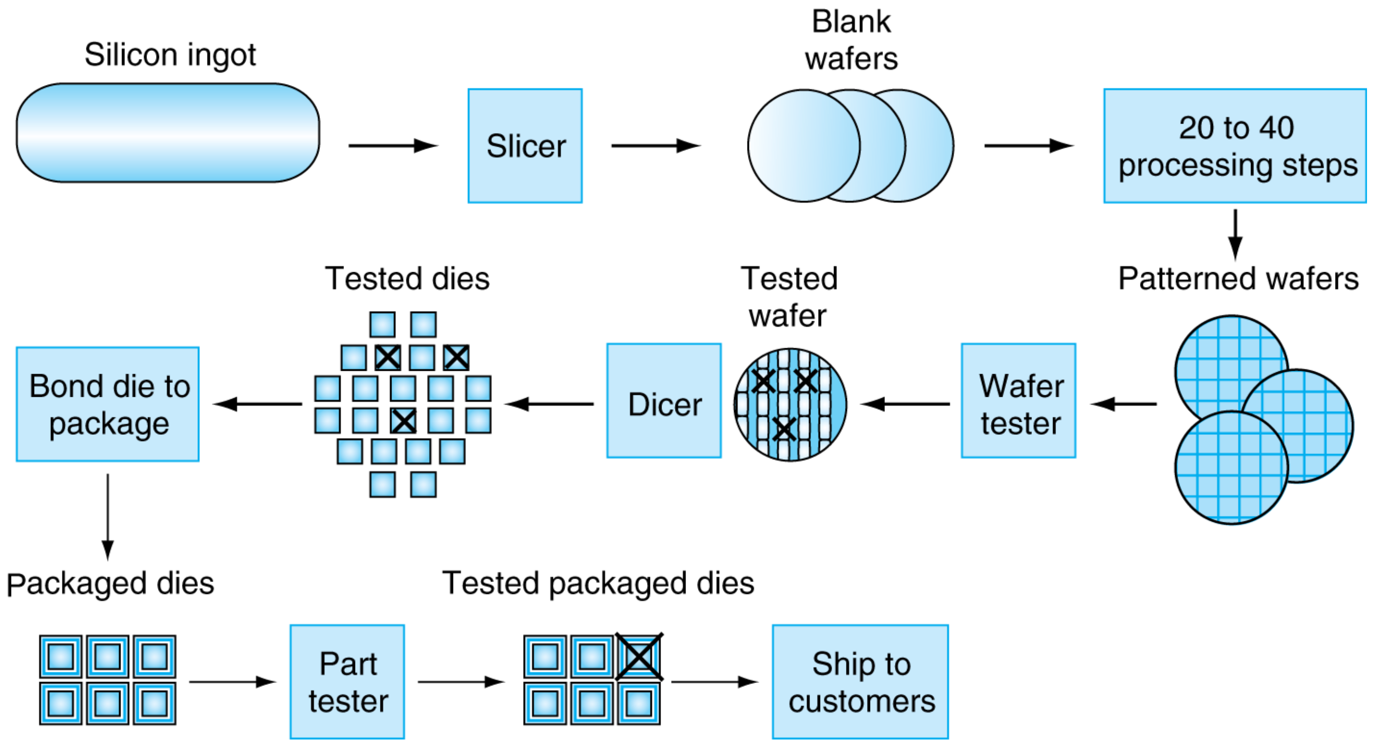
- More than one processor per chip

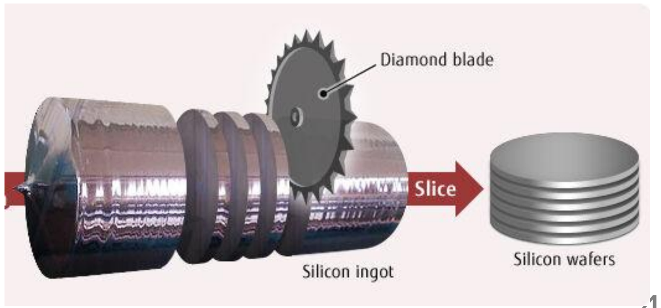
Requires explicitly parallel programming

- 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

Manufacturing ICs

Yield (수율): proportion of working dies per wafer





AMD Opteron X2 Wafer

X2 : 300mm wafer, 117 chips, 90nm technology

X4 : 45nm technology

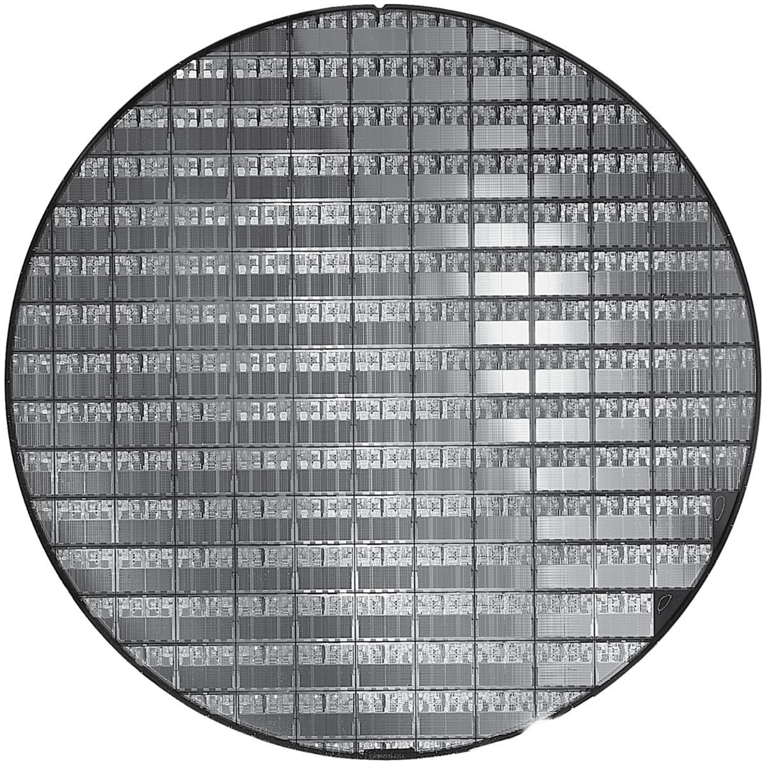
...

...

Intel core 12th : 10nm

Ryzen 4th : 7nm

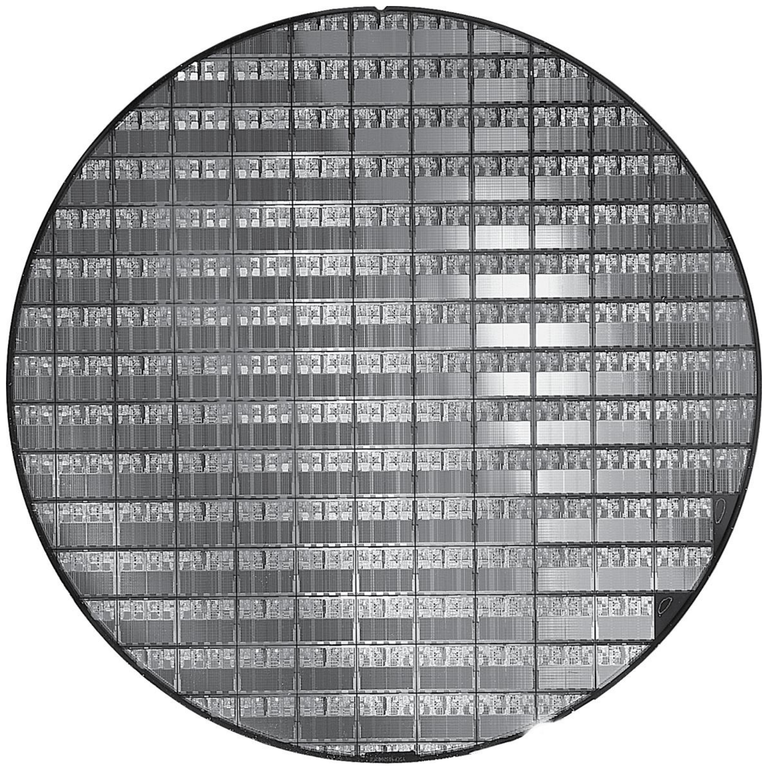
Apple M1 : 5nm



Integrated Circuit Cost

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



Amdahl's Law

프로그램은 병렬처리가 가능한 부분과 불가능한 순차적인 부분으로 구성되므로 프로세서를 아무리 병렬화 시켜도 더 이상 성능이 향상되지 않는 한계가 존재 한다는 법칙

Improving an aspect of a computer cannot guarantee a proportional improvement in overall performance

$$\boxed{T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvemen t factor}} + T_{\text{unaffected}}}$$

Example

- MULTIPLY instruction accounts for 80s of 100s
- ADD instruction accounts for 20s of 100s

Corollary: make the common case fast

Concluding Remarks

Cost/performance is improving

- Due to underlying technology development

Hierarchical layers of abstraction

- In both hardware and software

Instruction set architecture

- The hardware/software interface

Execution time: the best performance measure

Power is a limiting factor

- Use parallelism to improve performance