Topic 1

Introduction to Computer

The Computer Revolution

- Progress in computer technology
  - Moore’s Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Biomedical project
  - Internet+
  - Security
- Computers are pervasive

Classes of Computers

- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Embedded computers
  - Hidden as components of systems
  - Subject to power/performance/cost constraints
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

Where can we find Computers

- Desktop, Laptop, hand held PC, …
- Automotive
  - Automatic Ignition Systems, Cruise, ABS, traction control, airbag release system…
- Consumer Electronics
  - TV, PDA, appliances, toys, cell phones, camera …
- Industrial Control
  - robotics, control systems …
- Medical
  - Infusion Pumps, Dialysis Machines, Prosthetic Devices, Cardiac Monitors, …
- Networking
  - wired and wireless routers, hubs, …
- Office Automation
  - fax, photocopiers, printers, scanners, …
- Aerospace applications
  - Flight-control systems, engine controllers, auto-pilots and passenger in-flight entertainment systems…
- Defense systems
  - Radar systems, fighter aircraft flight-control systems, radio systems, and missile guidance systems…

Product: Hunter Programmable Digital Thermostat.

Microprocessor: 4-bit

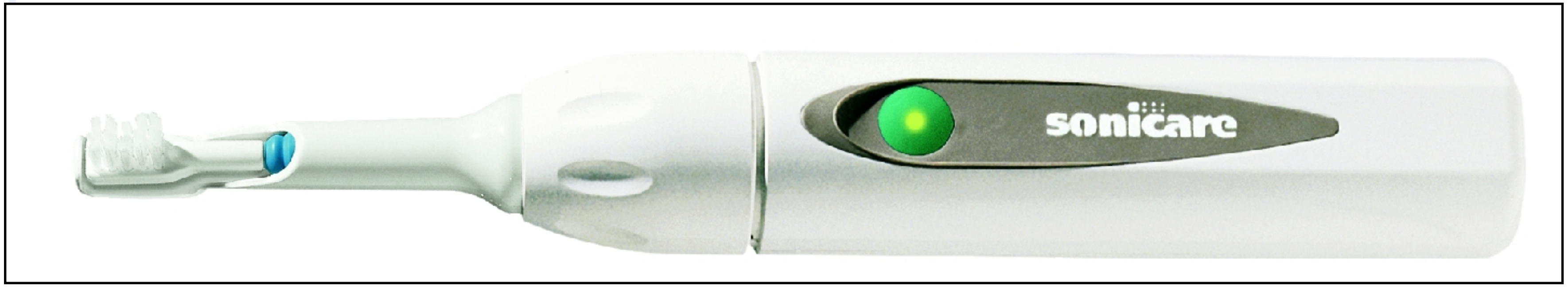
by Daniel W. Lewis

Product:Vendo V-MAX 720 vending machine.

Microprocessor: 8-bit Motorola 68HC11.

by Daniel W. Lewis

Product: Sonicare Plus toothbrush. Microprocessor: 8-bit Zilog Z8.

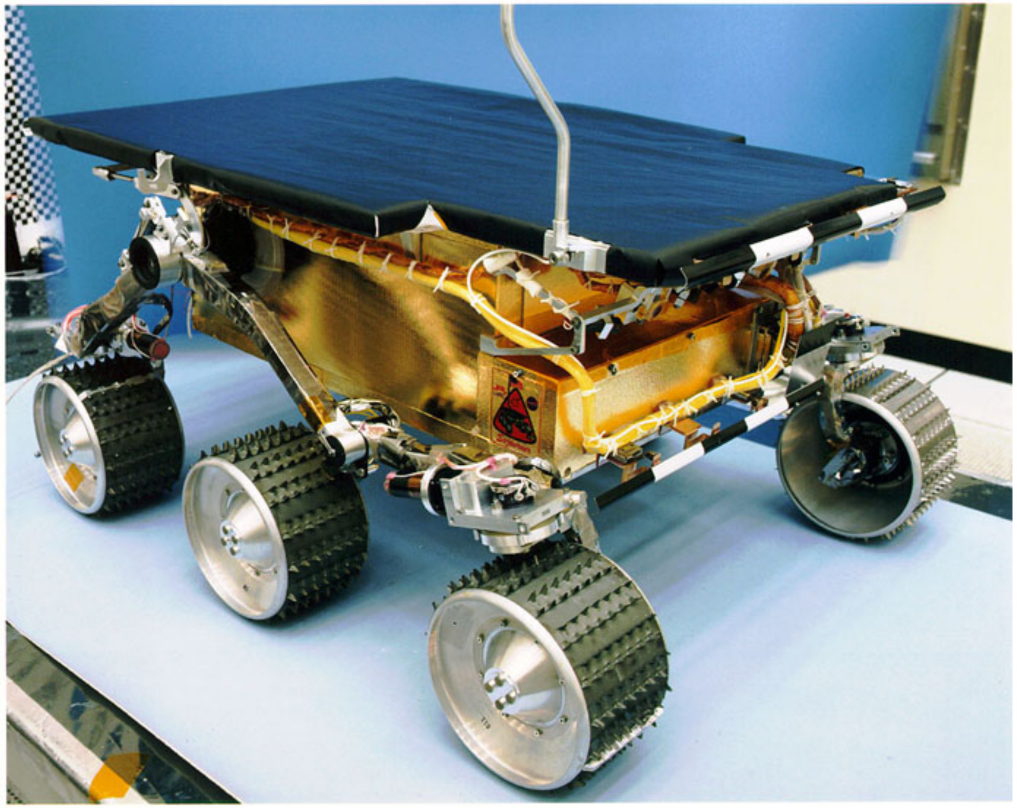


by Daniel W. Lewis

Product: Miele dishwashers.

Microprocessor: 8-bit Motorola 68HC05.

by Daniel W. Lewis



Product: NASA's Mars Sojourner Rover.

Microprocessor: 8-bit Intel 80C85.

by Daniel W. Lewis

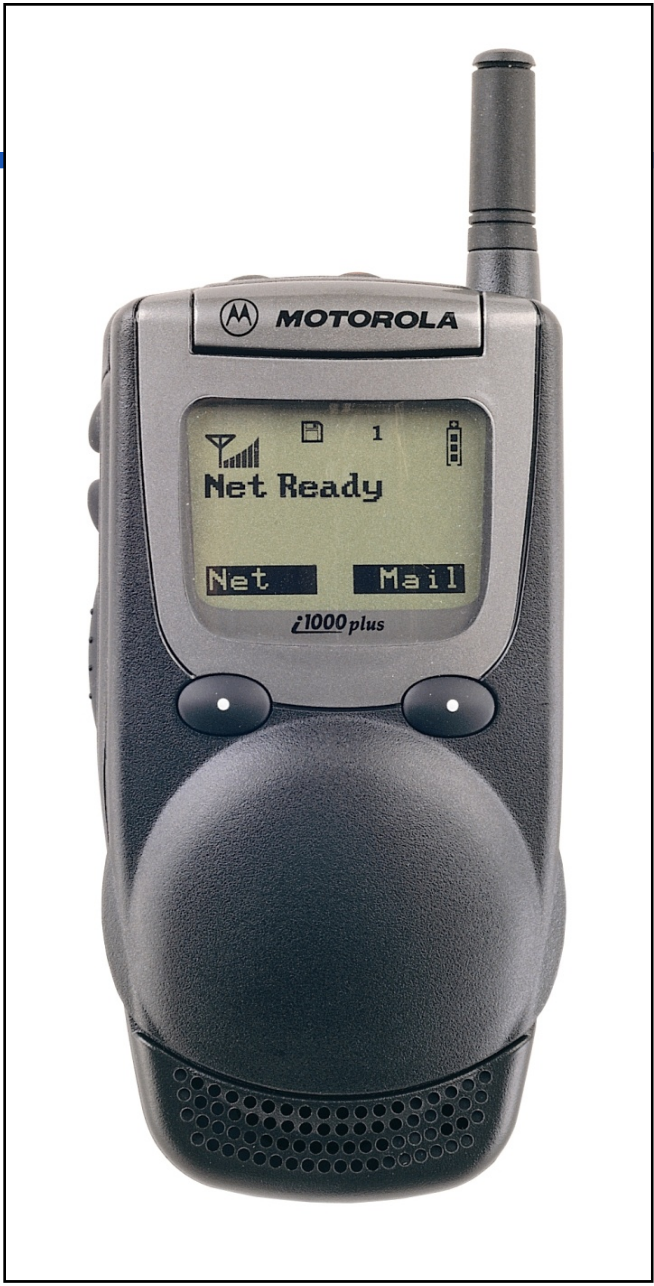
Product: CoinCo USQ-712 coin changer.

Microprocessor: 8-bit Motorola 68HC912.

by Daniel W. Lewis

Product: Garmin Nüvi GPS Receiver

Microprocessor: 32-bit.



Product: Motorola i1000plus iDEN Multi-Service Digital Phone.

Microprocessor: Motorola 32-bit MCORE.

by Daniel W. Lewis

Product: Nintendo Wii Controller
Microprocessor: IBM 32-bit Power RISC

Product: Apple iPad

Microprocessor: Apple A4, a 32-bit ARM RISC.

Product: Apple iWatch

Microprocessor: 32-bit Apple A6 and M7 Coprocessor

by Daniel W. Lewis

Product: iPhone 7

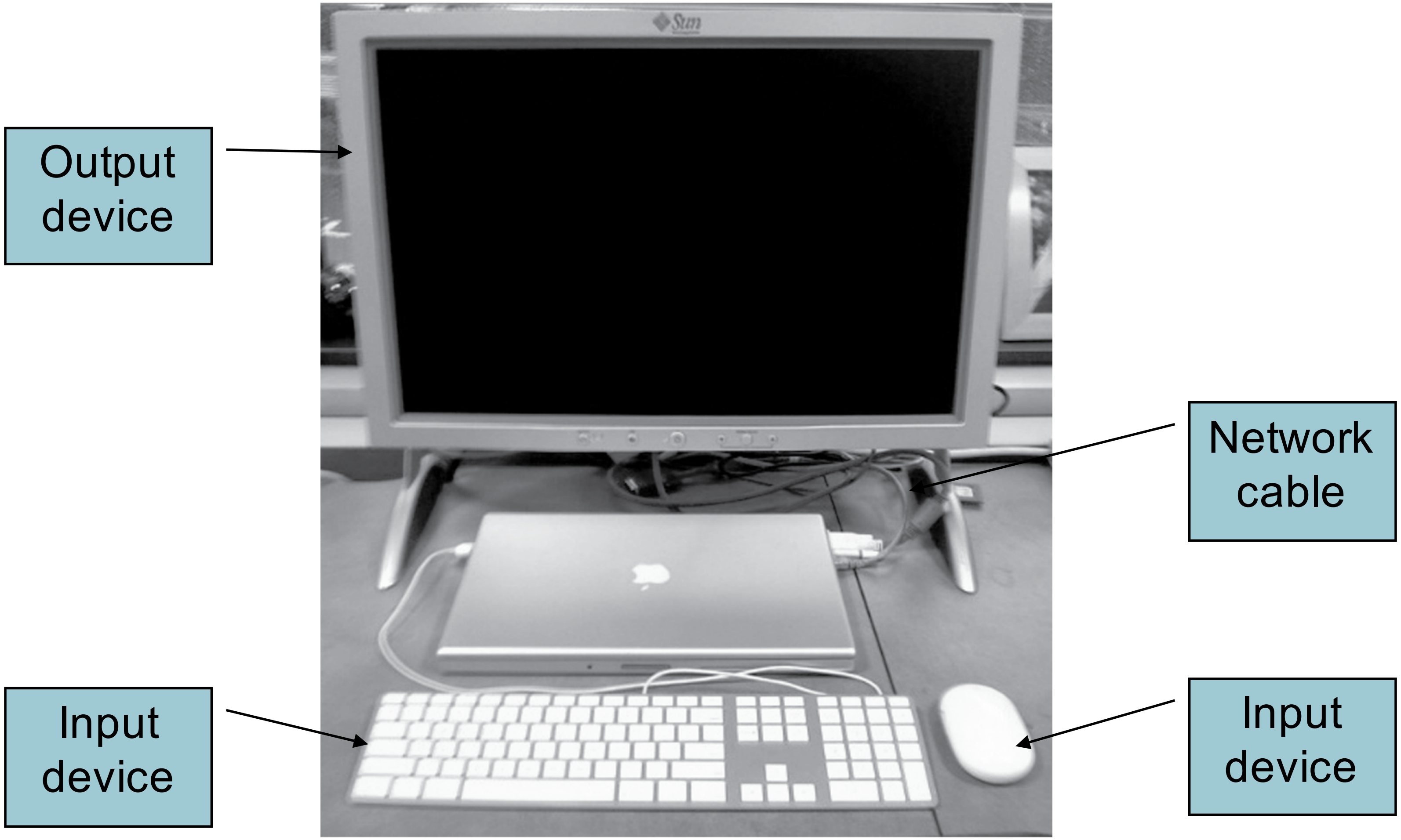
Microprocessor: Apple A10, based on 64-bit quad-core ARMv8-A

Product: Sony Aibo ERS-110 Robotic Dog.

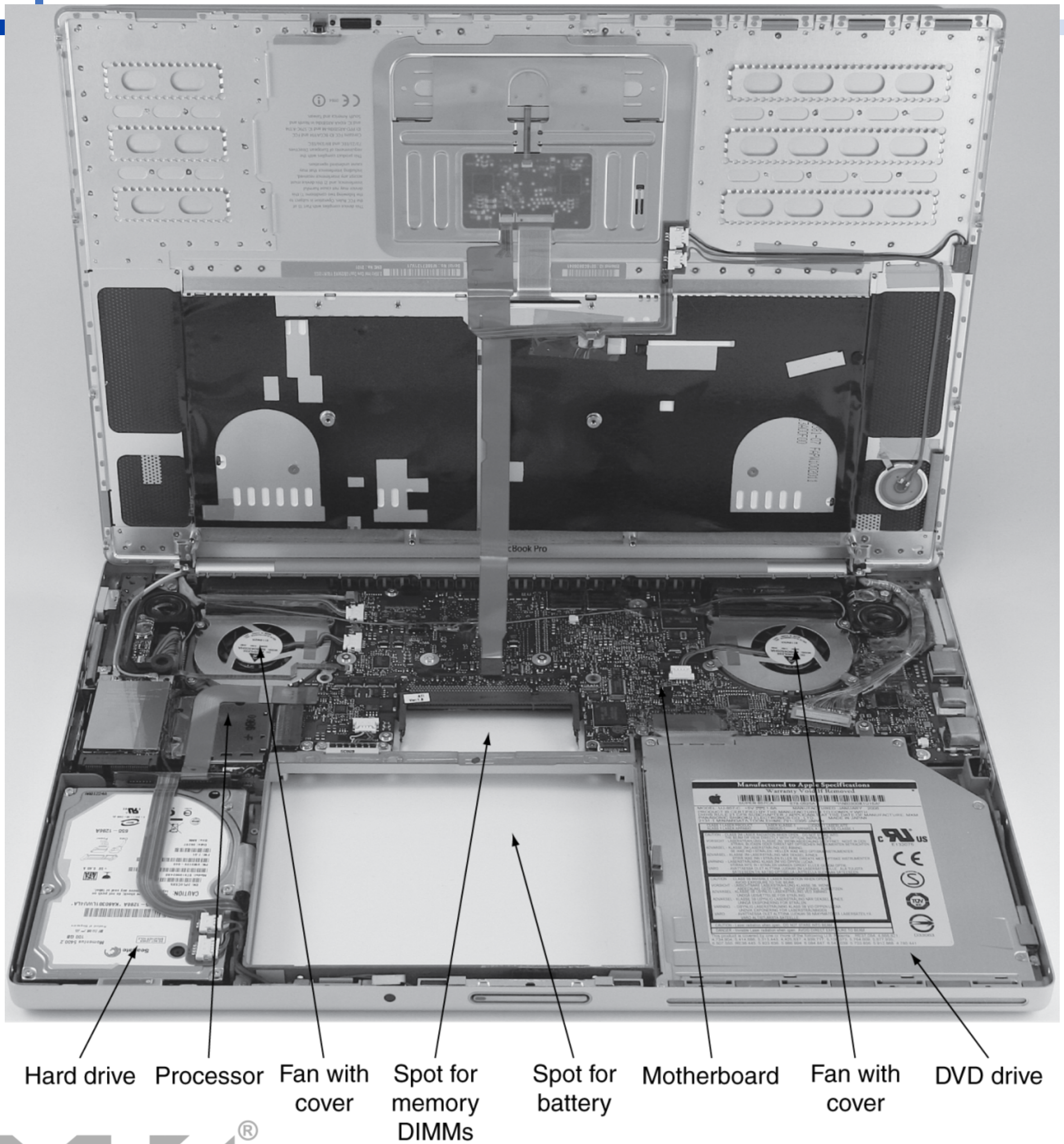
Microprocessor: 64-bit MIPS RISC.

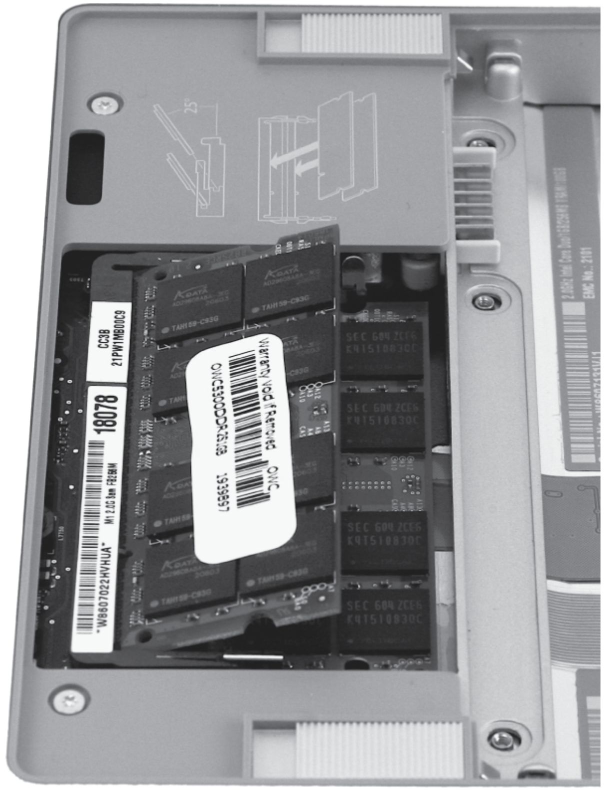
by Daniel W. Lewis

Anatomy of a Computer



Opening the Box



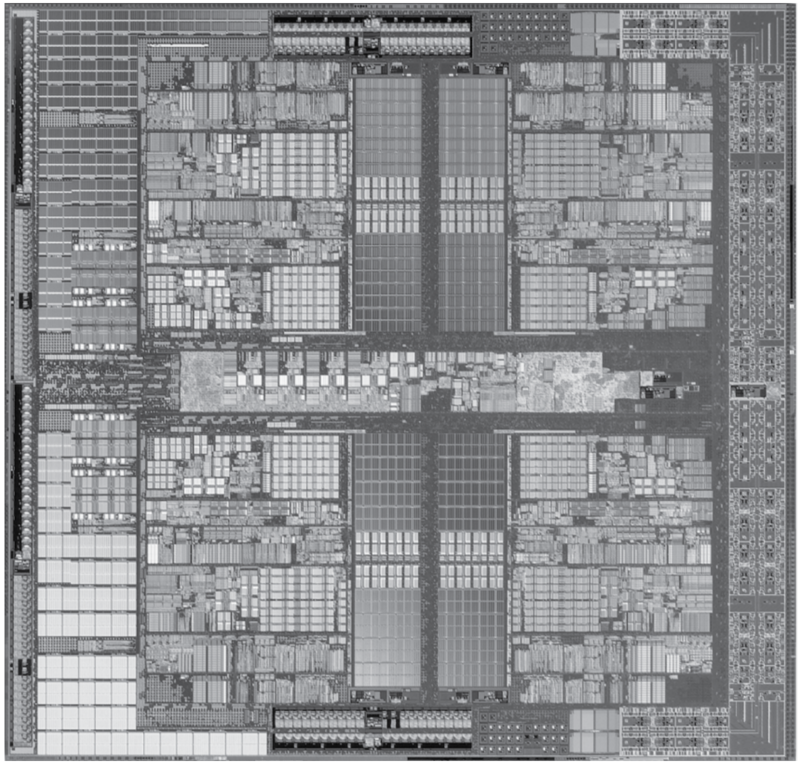


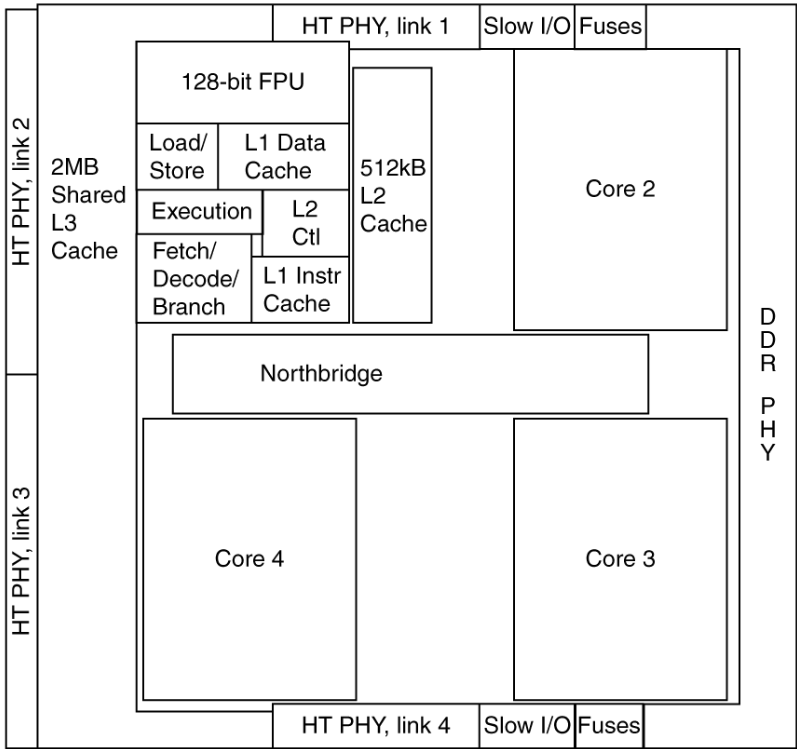
Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: controls how data flows
- Cache memory
  - Small fast SRAM memory for immediate access to data

Inside the Processor

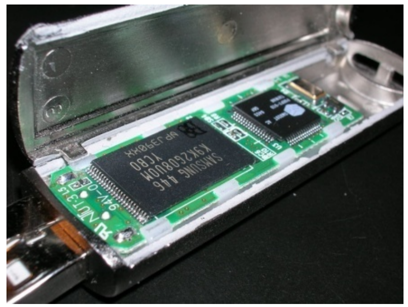
- AMD Barcelona: 4 processor cores





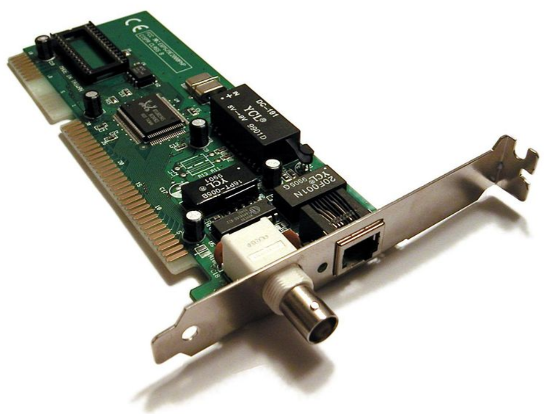
Peripheral – Memory

- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



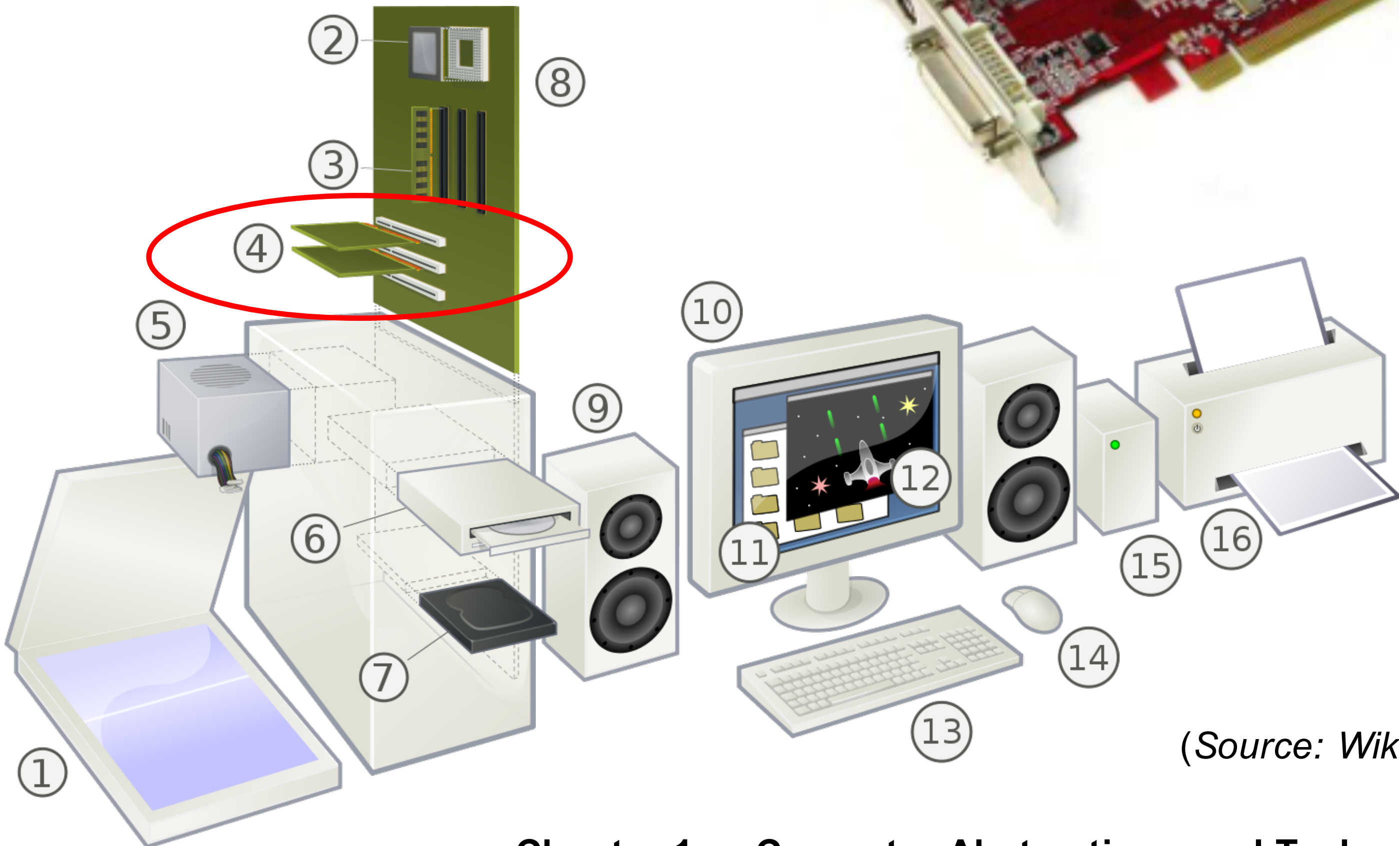
Peripheral – Networks

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



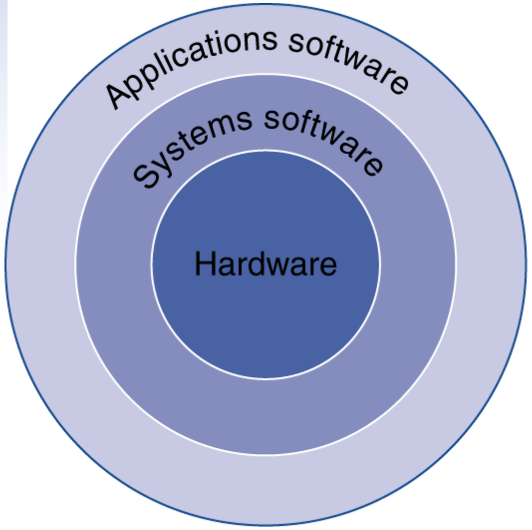
Peripheral – Display

- Screen and VGA card



(*Source: Wikipedia.org*)

Together with Software



- Application software
  - Written in high-level language (HLL)
- System software
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
  - Written in C and assembly
- Hardware
  - Processor, memory, I/O controllers

Levels of Program Code

- **High-level language**
  - What we use
- **Assembly language**
  - What both we and computers can use
- **Machine instruction**
  - What computers use

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;
}
```

Compiler

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
```

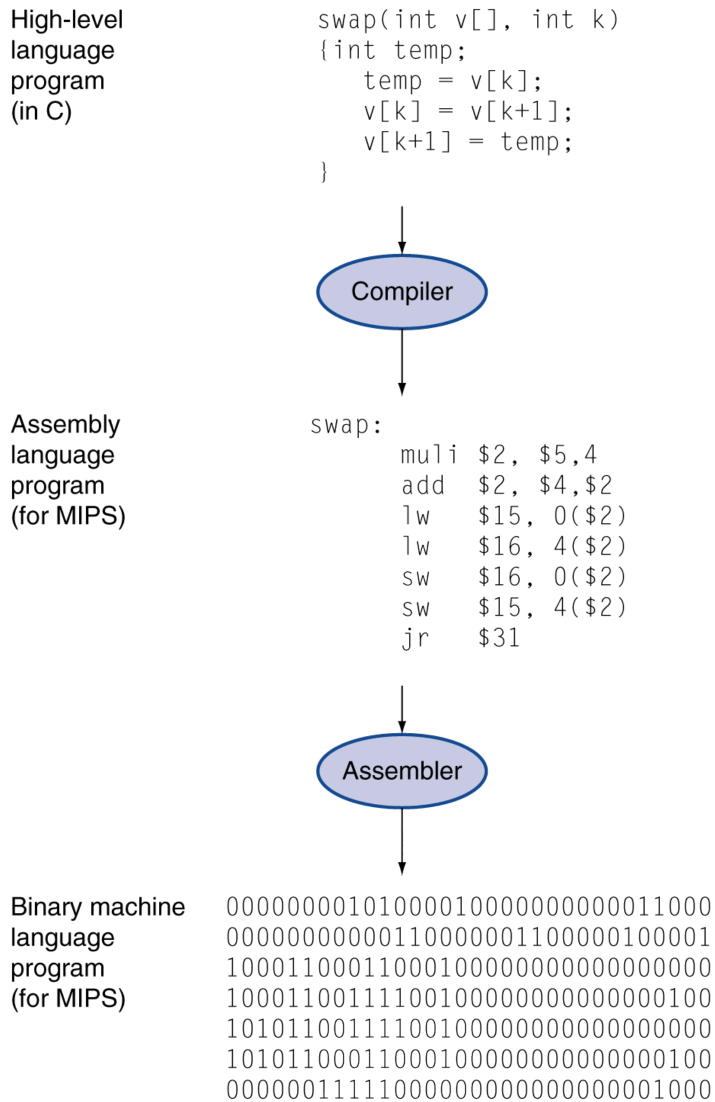
Assembler

Binary machine language program (for MIPS)

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

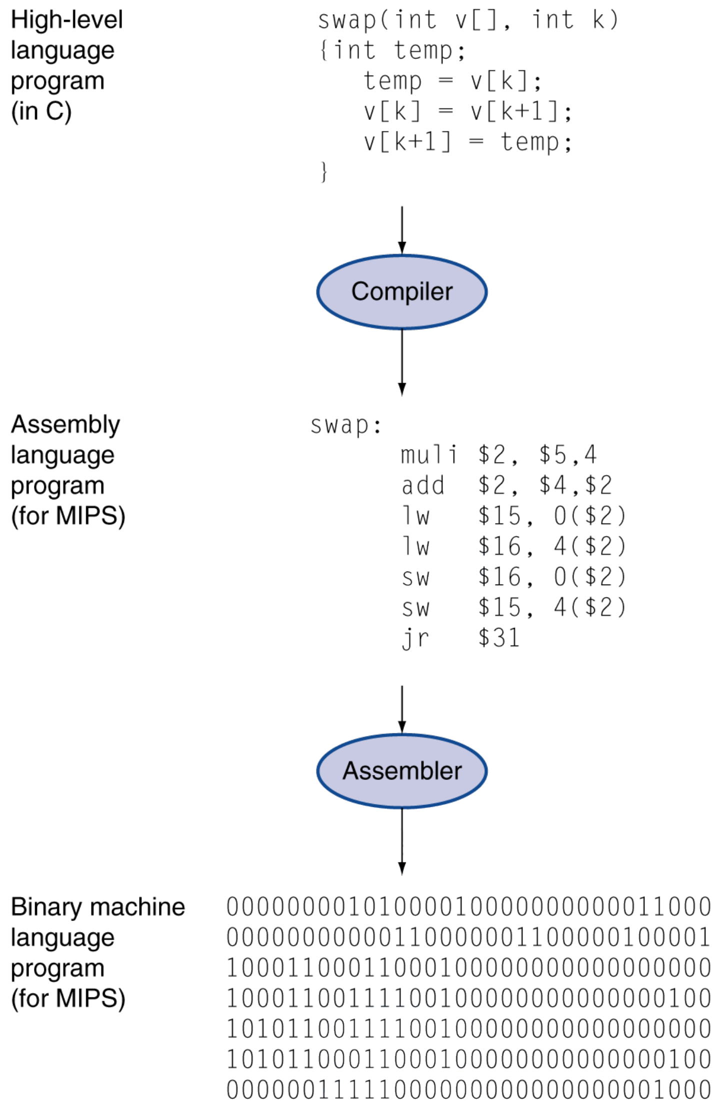
Levels of Program Code

- **High-level language**
  - Syntax is similar to English
  - A translator is required to translate the program – compile
  - Allows the user to work on the program logic at higher level



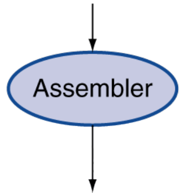
Levels of Program Code

- **Assembly language**
  - Composed of assembly ***instructions***
  - An assembly instruction is a mnemonic representation of a machine instruction
  - Assembly instruction must be translated before it can be executed – assembler
  - Programmers need to work on the program logic at a very low level, hard to achieve high productivity.



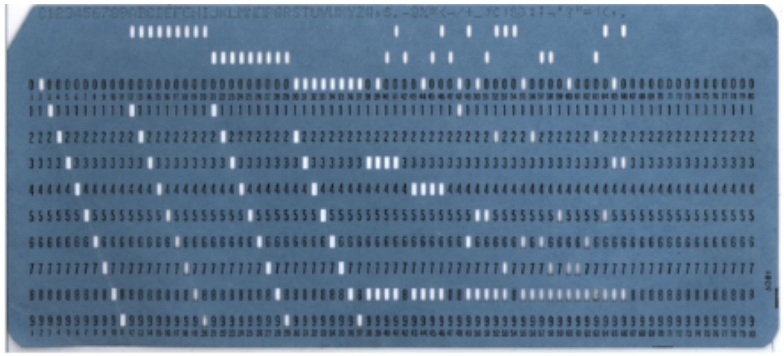
Levels of Program Code

- **Machine instruction**
  - A sequence of binary digits which can be executed by the processor
  - Hard to understand, program, and debug for human being



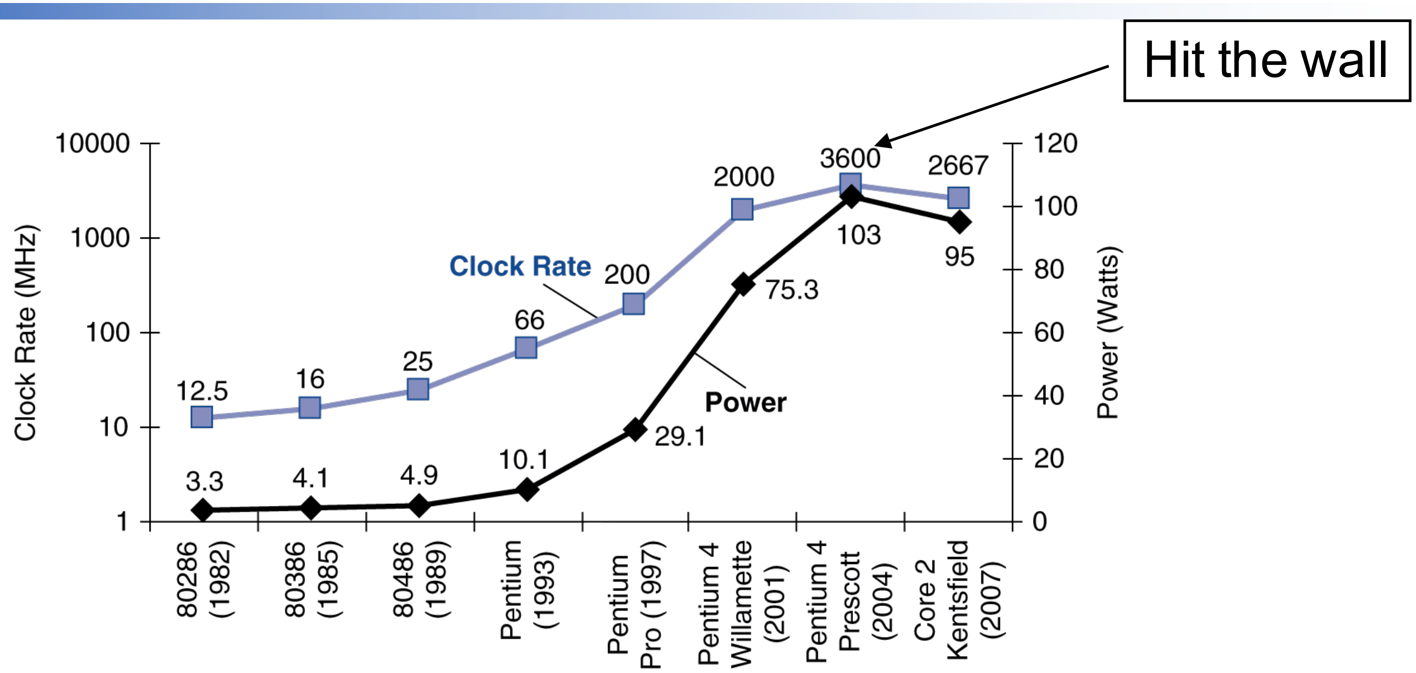
Binary machine language program (for MIPS)

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



From Wikipedia

Power Trends



- In CMOS IC technology

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

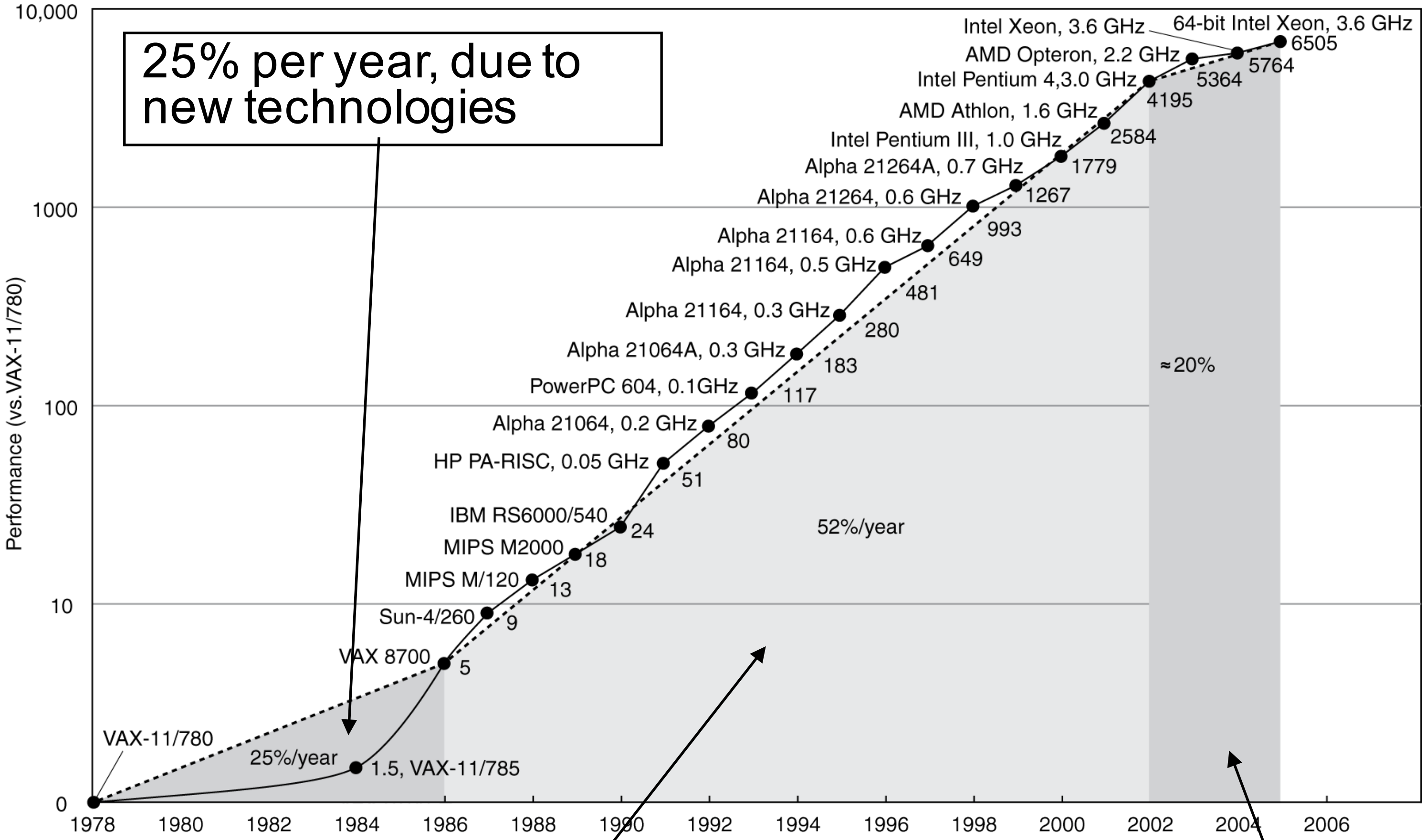
$\times 30$

5V→1V in 15y,
15% off/generation

$\times 1000$

Uniprocessor Performance

25% per year, due to new technologies

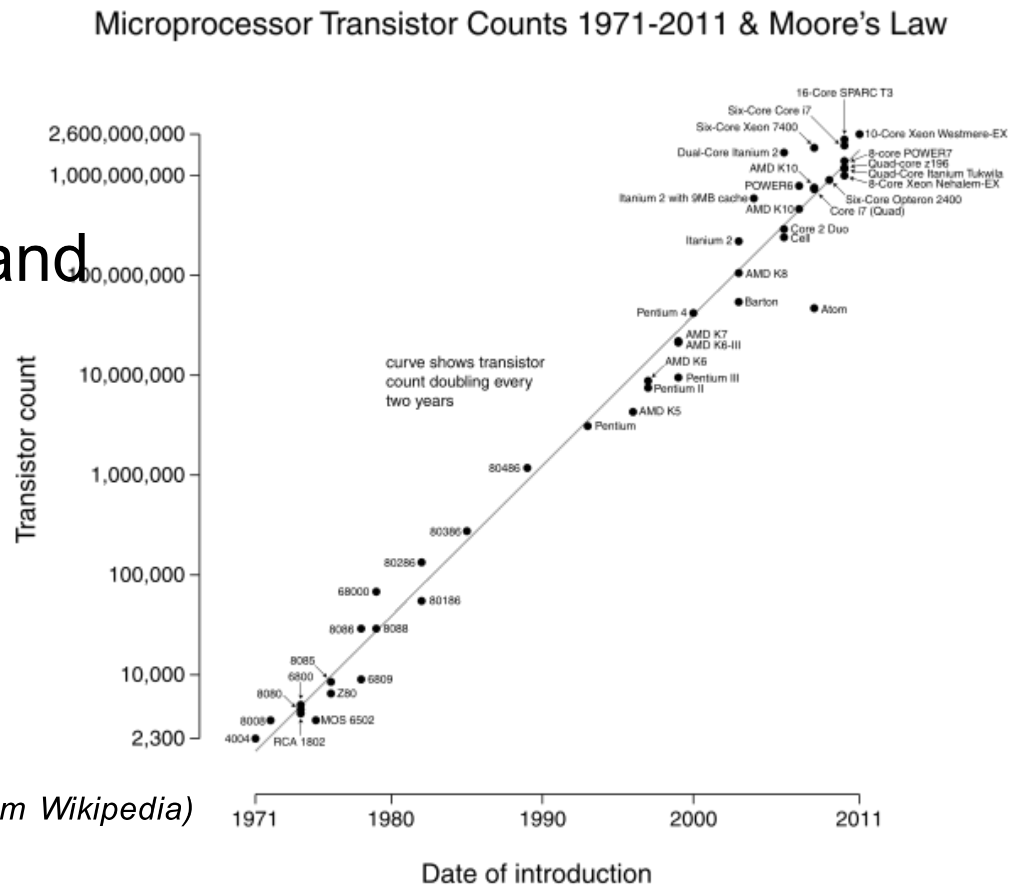


52% per year, benefited from architecture/organization advances

20% per year, limited by power, instruction-level parallelism, memory latency

Technology Trends

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



(From Wikipedia)

How to Measure Computer Performance?

- Execution (response) time
  - How long it takes to do a task
  - Measure for PCs and embedded computers
- Throughput
  - Total work done per unit time
  - Servers
- We'll focus on execution time for now

Measuring Performance - Time

- Elapsed time – for System Performance
  - Total execution time to complete a task, including
    - Processing, I/O, OS overhead, idle time, everything
  - But doesn't completely reflect computer's performance if it focuses on better throughput
- CPU time – for CPU Performance
  - CPU execution time processing a task
    - Exclude I/O time, time spent for other tasks
  - Comprises **user CPU time** and **system CPU time**
    - Hard to differentiate
  - Different programs run with different CPU performance and system performance
    - User must know what's the desired performance metric

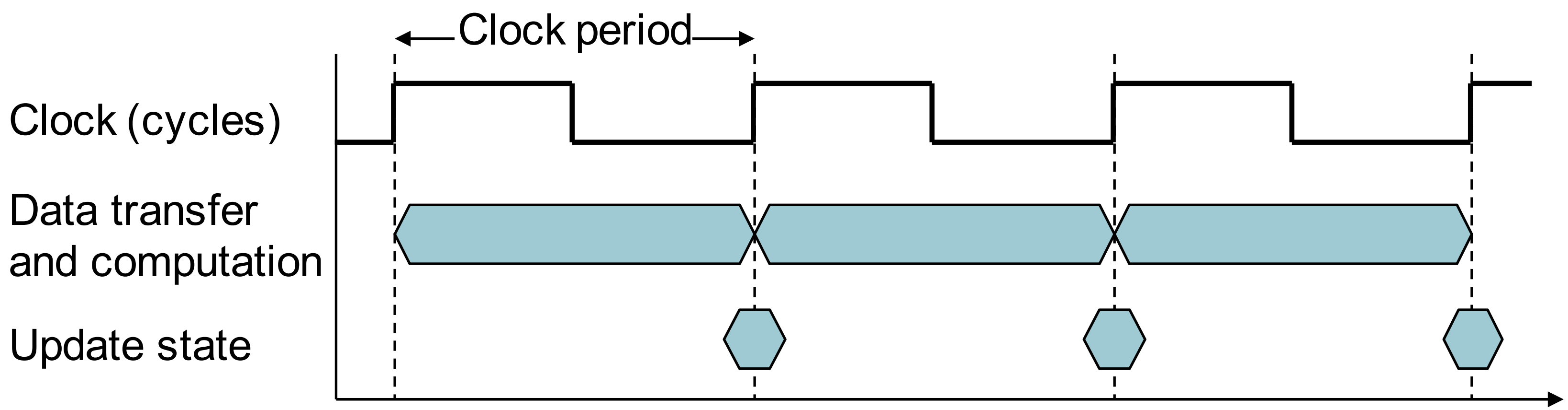
Understanding Performance

Performance-related factors:

- 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

CPU Clocking

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



CPU Time

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

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate

$\left.\right\}$ Tradeoff we will face when designing the CPU

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 $_A$
- 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 (IC) and Clock cycle Per Instruction (CPI)

- How instructions related to performance?

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

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

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

Classic CPU performance equation

- Instruction Count for a program
  - Determined by program, Instruction Set Architecture (ISA), and compiler
- 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?

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

A is faster…

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

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

…by this much

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

Concluding Remarks

- Classes and applications of computer
- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance