

Lecture 1 &2

Computer Abstractions and
Technology

Chapter 1

- Syllabus

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

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

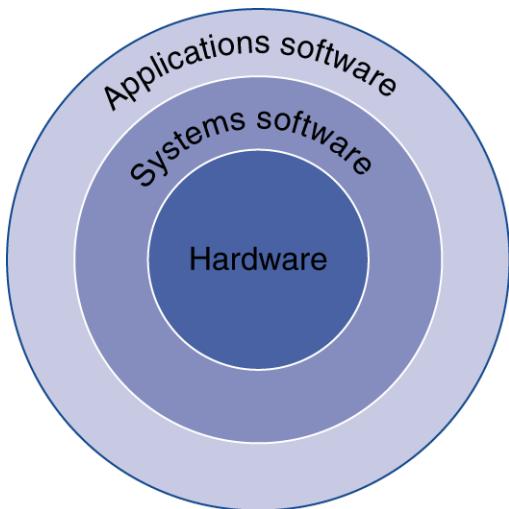
What You Will Learn

- 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



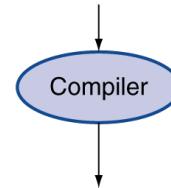
- Application software
 - Written in high-level language (HLL)
- System software
 - Compiler: translates HLL code to machine code
 - Operating System:
 - Handling input/output
 - Managing memory and storage
 - Scheduling tasks & sharing resources
- Hardware
 - Processor, memory, I/O controllers

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;
}
```

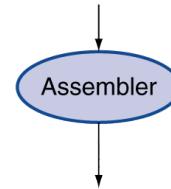


Assembly language program (for MIPS)

```

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

```

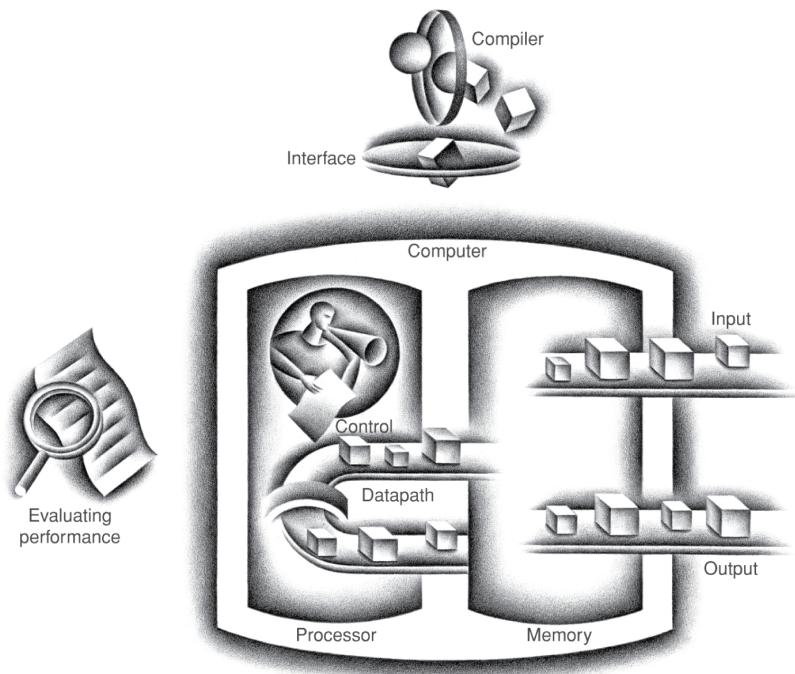


Binary machine
language
program
(for MIPS)

```
00000000010100001000000000000011000  
000000000000110000001100000100001  
1000110001100010000000000000000000  
100011001111001000000000000000000100  
101011001111001000000000000000000000  
101011000110001000000000000000000100  
0000001111100000000000000000000000100
```

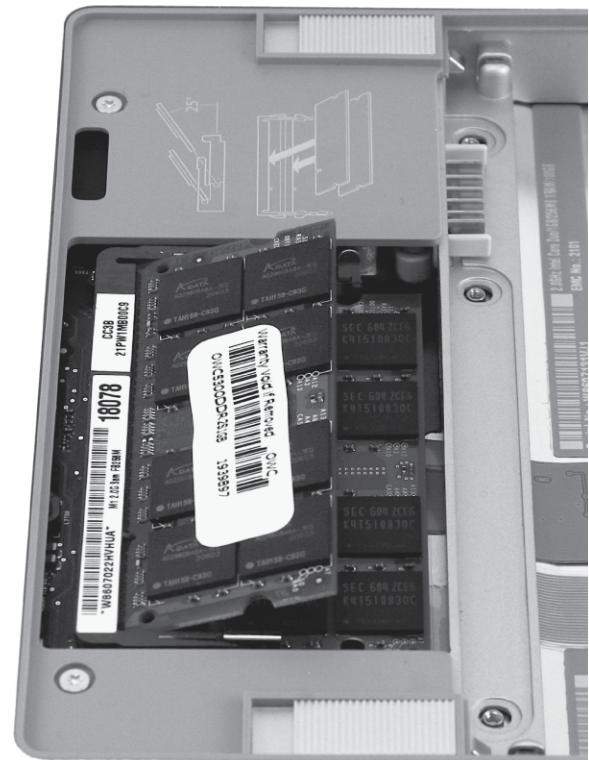
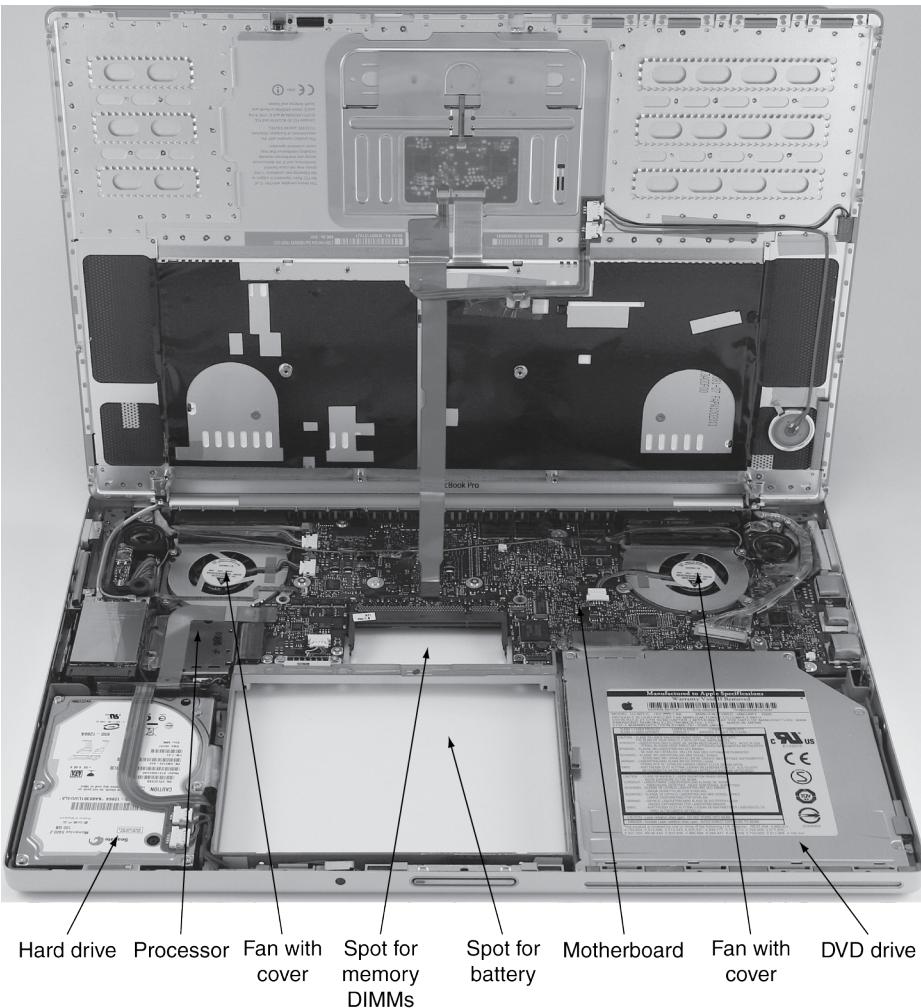
Components of a Computer

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

Opening the Box

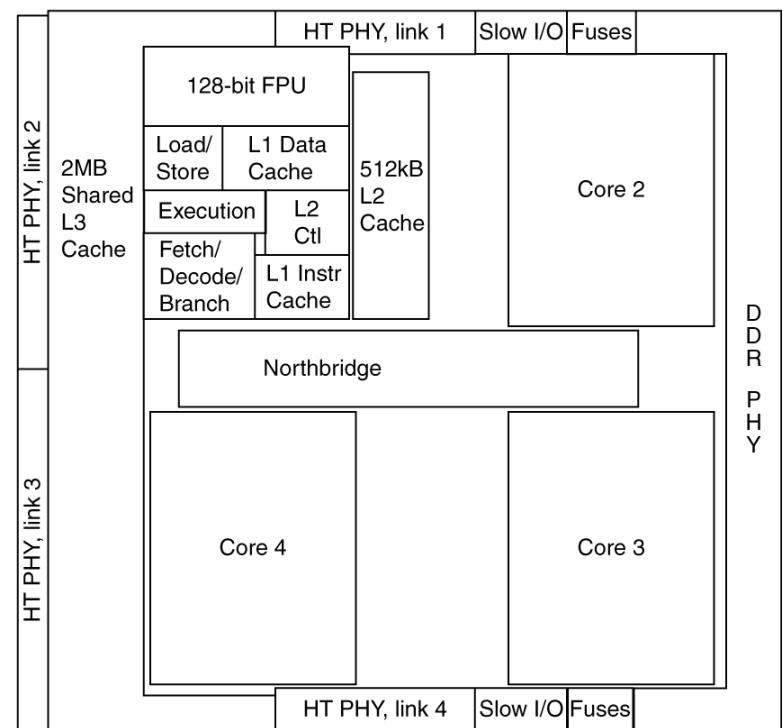
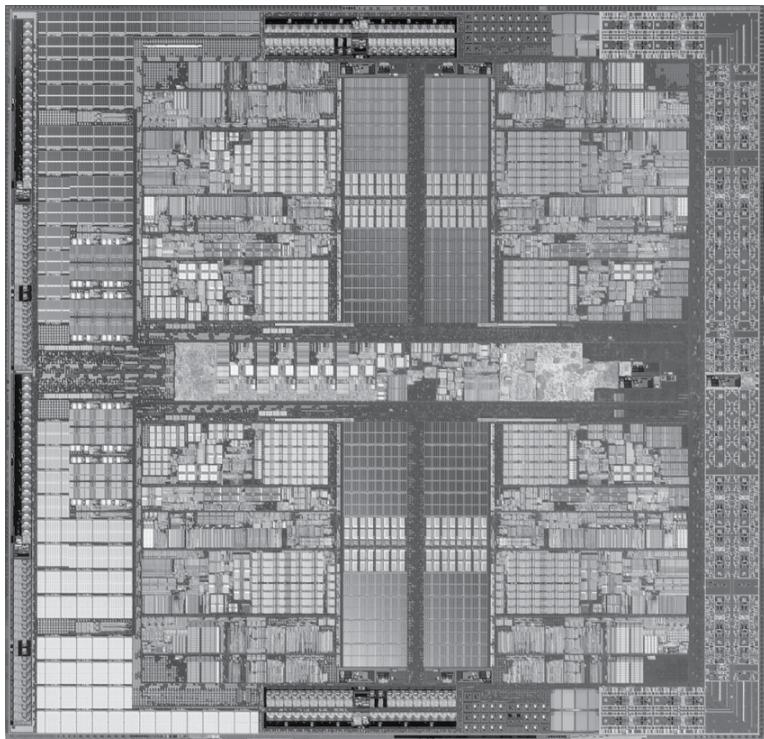


Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
 - Small fast SRAM memory for immediate access to data

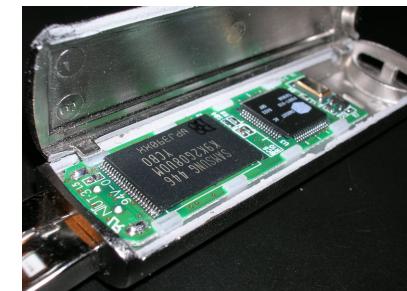
Inside the Processor

- AMD: 4 processor cores



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)



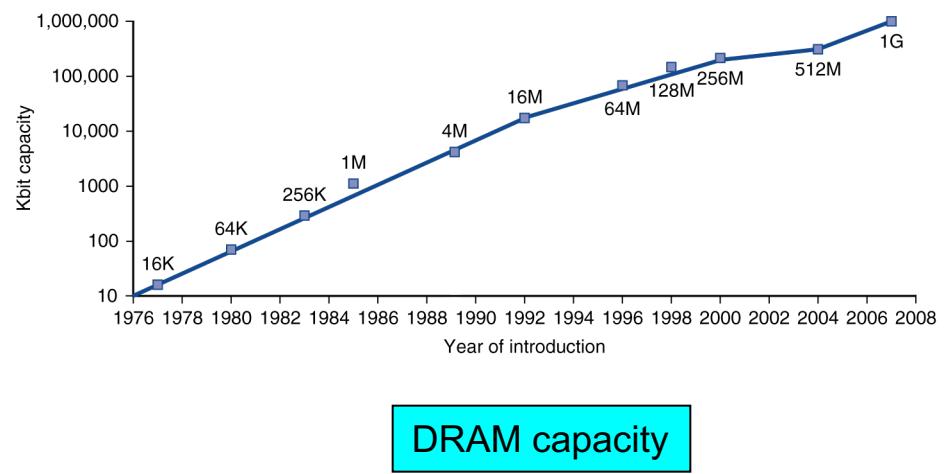
Networks

- 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



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

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”

$$\begin{aligned}\text{Performance}_x / \text{Performance}_y \\ = \text{Execution time}_y / \text{Execution time}_x = n\end{aligned}$$

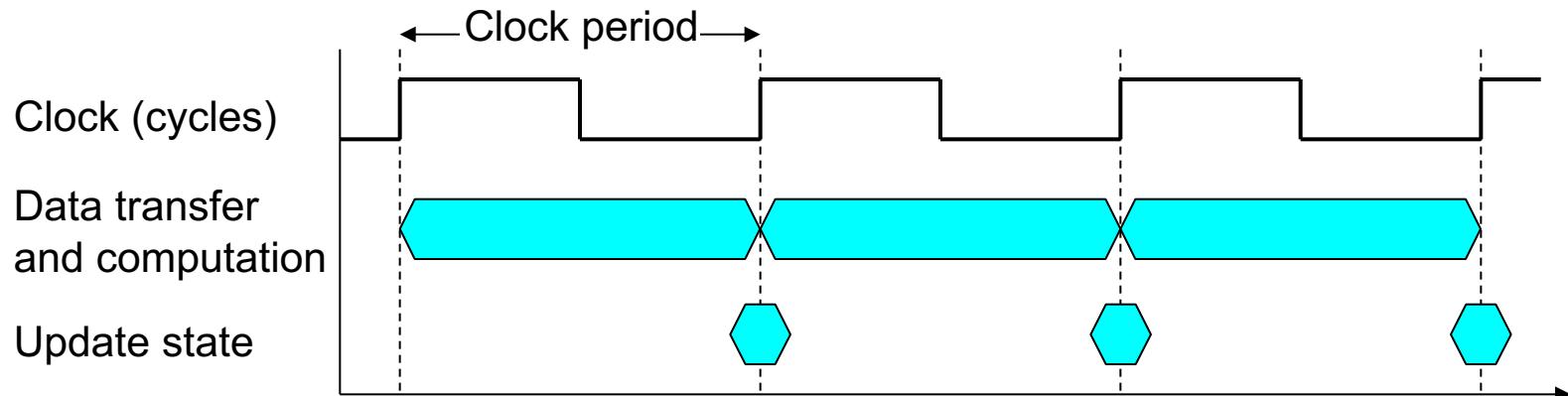
- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A$
 $= 15s / 10s = 1.5$
 - So A is 1.5 times faster than B

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
 - 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., $250\text{ps} = 0.25\text{ns} = 250 \times 10^{-12}\text{s}$
- Clock frequency (rate): cycles per second
 - e.g., $4.0\text{GHz} = 4000\text{MHz} = 4.0 \times 10^9\text{Hz}$

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 $1.2 \times$ clock cycles
- How fast must Computer B clock be?

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

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

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

Instruction Count and CPI

Clock Cycles = Instruction Count \times Cycles per Instruction

CPU Time = Instruction Count \times CPI \times 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?

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

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

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

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

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



Relative
frequency

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
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
 - Avg. CPI = $10/5 = 2.0$
- Sequence 2: IC = 6
 - Clock Cycles
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$
 $= 9$
 - 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

Structure

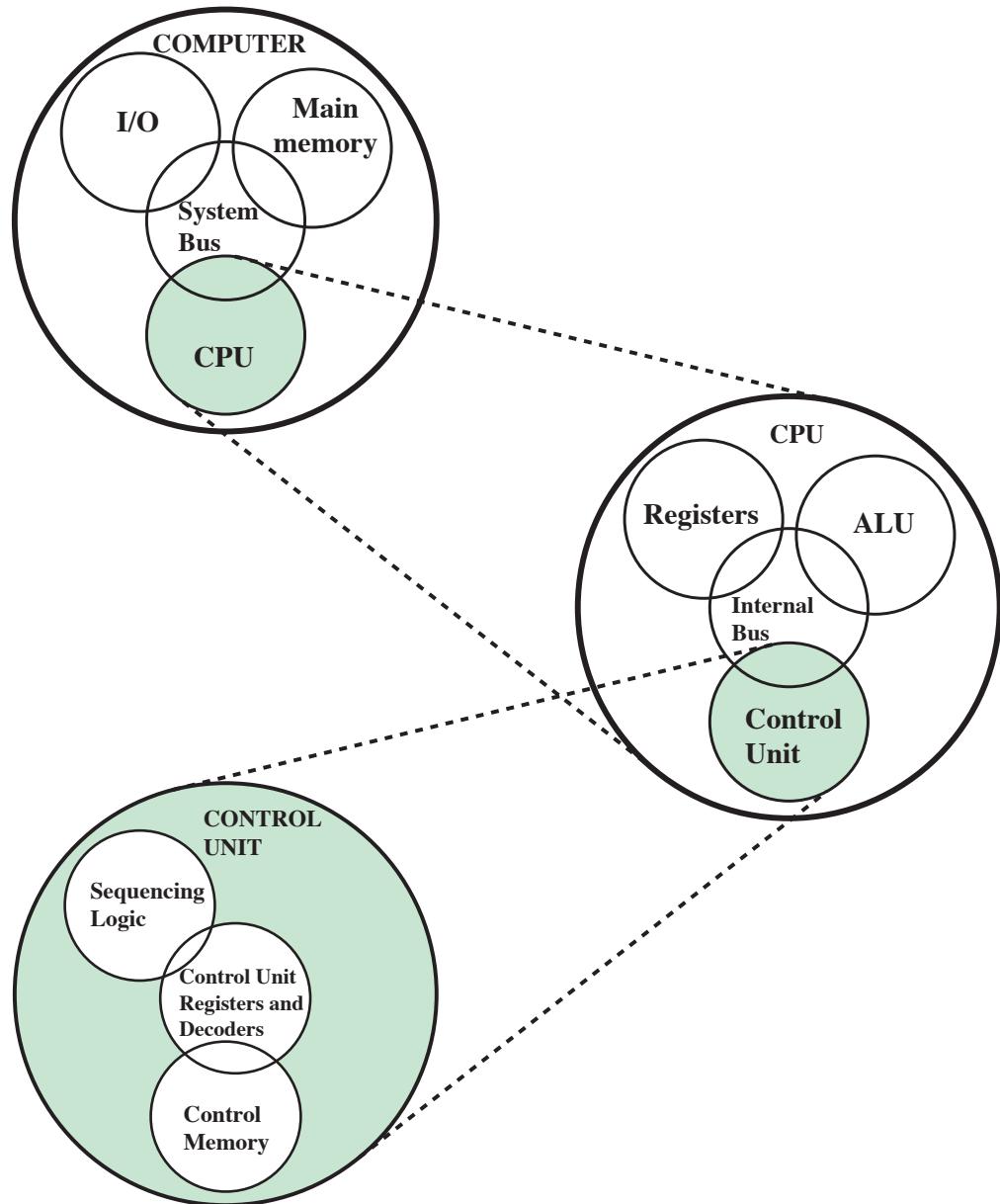
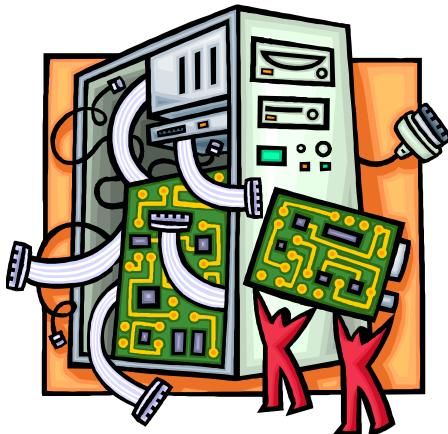


Figure 1.1 A Top-Down View of a Computer

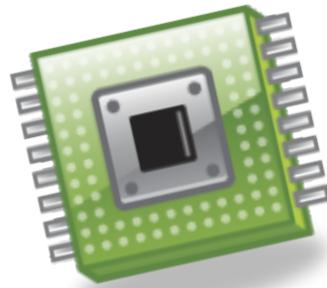
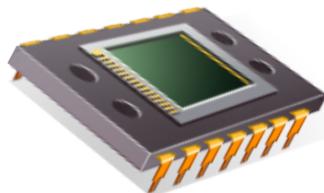
There are four main structural components of the computer:



- ◆ CPU – controls the operation of the computer and performs its data processing functions
- ◆ Main Memory – stores data
- ◆ I/O – moves data between the computer and its external environment
- ◆ System Interconnection – some mechanism that provides for communication among CPU, main memory, and I/O

CPU

Major structural components:



- Control Unit
 - Controls the operation of the CPU and hence the computer
- Arithmetic and Logic Unit (ALU)
 - Performs the computer's data processing function
- Registers
 - Provide storage internal to the CPU
- CPU Interconnection
 - Some mechanism that provides for communication among the control unit, ALU, and registers

Cache Memory

- Multiple layers of memory between the processor and main memory
- Is smaller and faster than main memory
- Used to speed up memory access by placing in the cache data from main memory that is likely to be used in the near future
- A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, etc.) progressively farther from the core

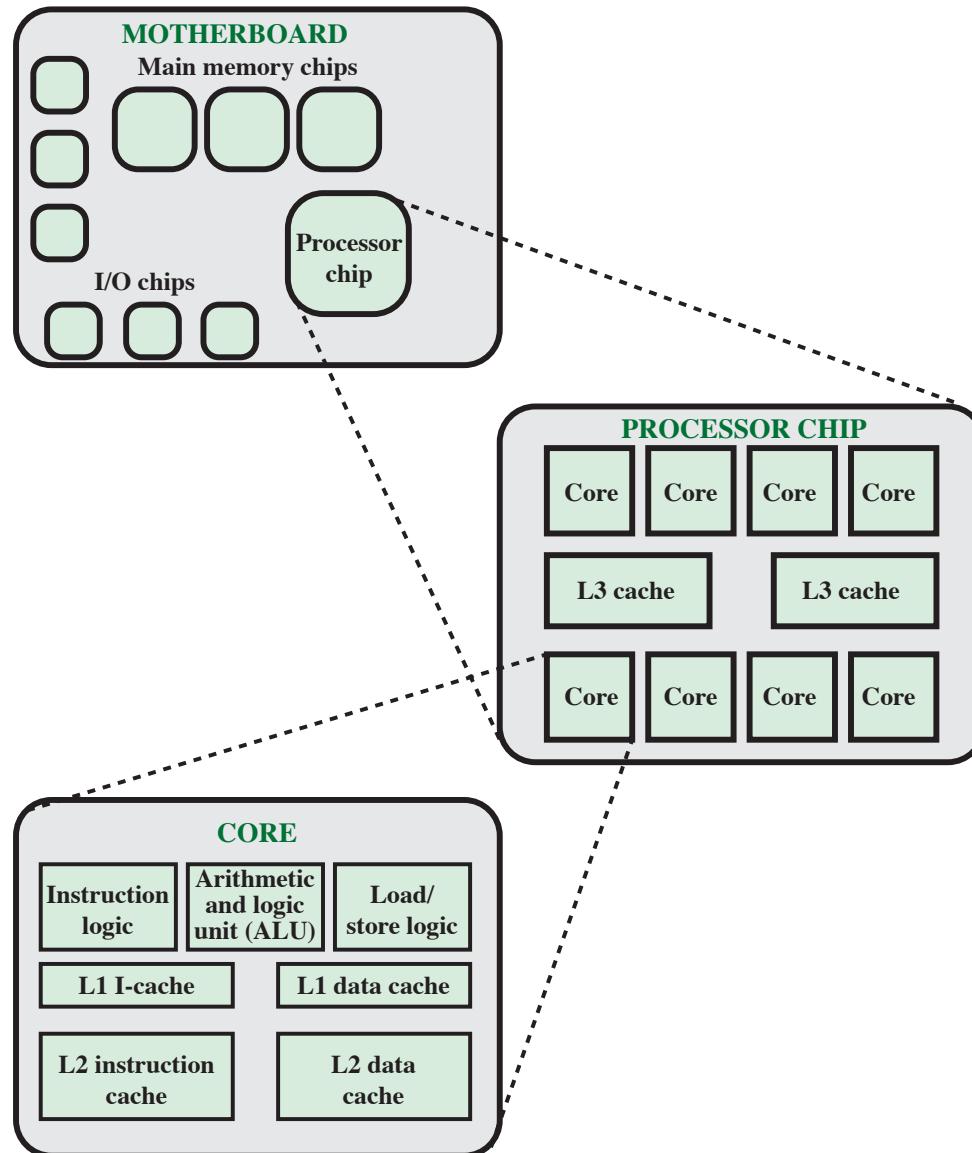


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

Moore's Law

1965; Gordon Moore – co-founder of Intel

Observed number of transistors that could be put on a single chip was doubling every year

The pace slowed to a doubling every 18 months in the 1970's but has sustained that rate ever since

Consequences of Moore's law:

The cost of computer logic and memory circuitry has fallen at a dramatic rate

The electrical path length is shortened, increasing operating speed

Computer becomes smaller and is more convenient to use in a variety of environments

Reduction in power and cooling requirements

Fewer interchip connections

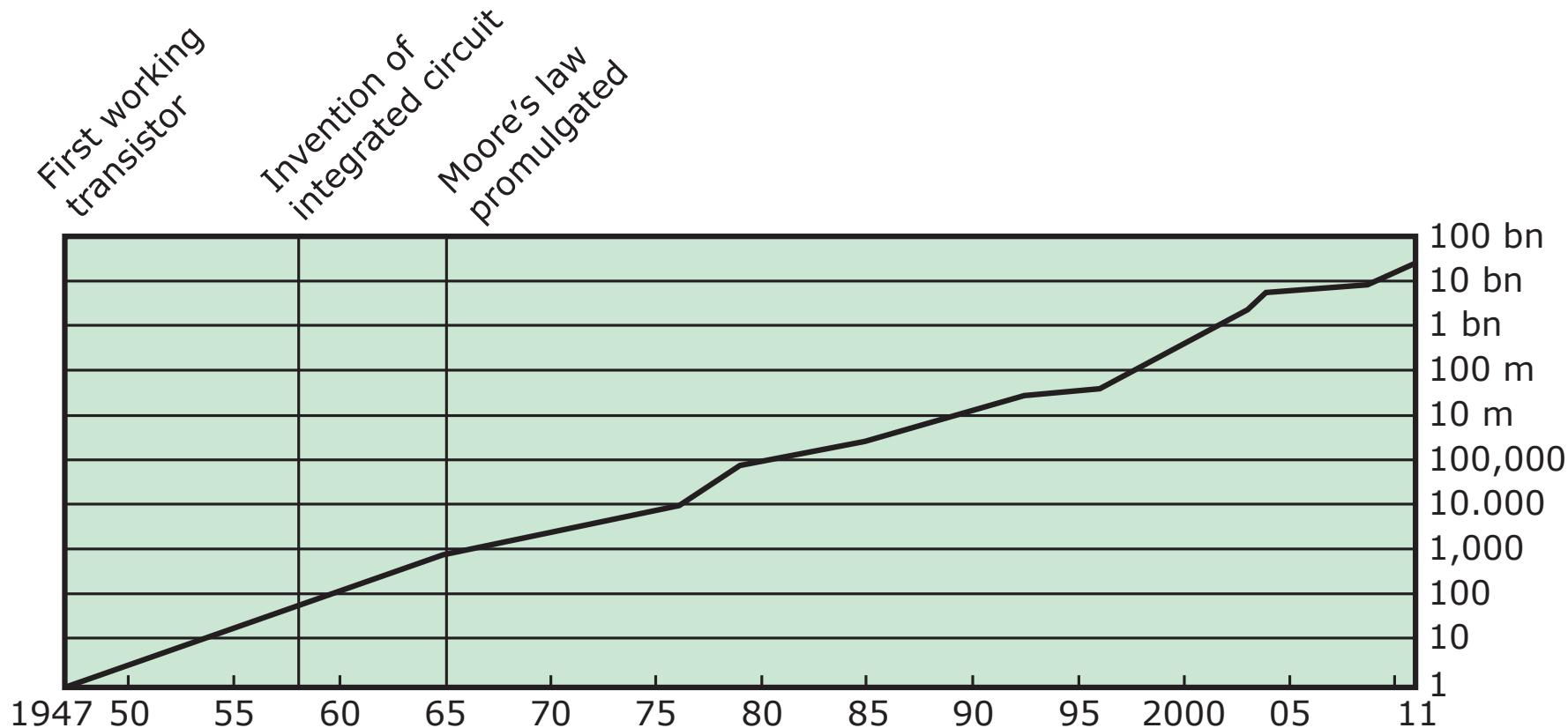
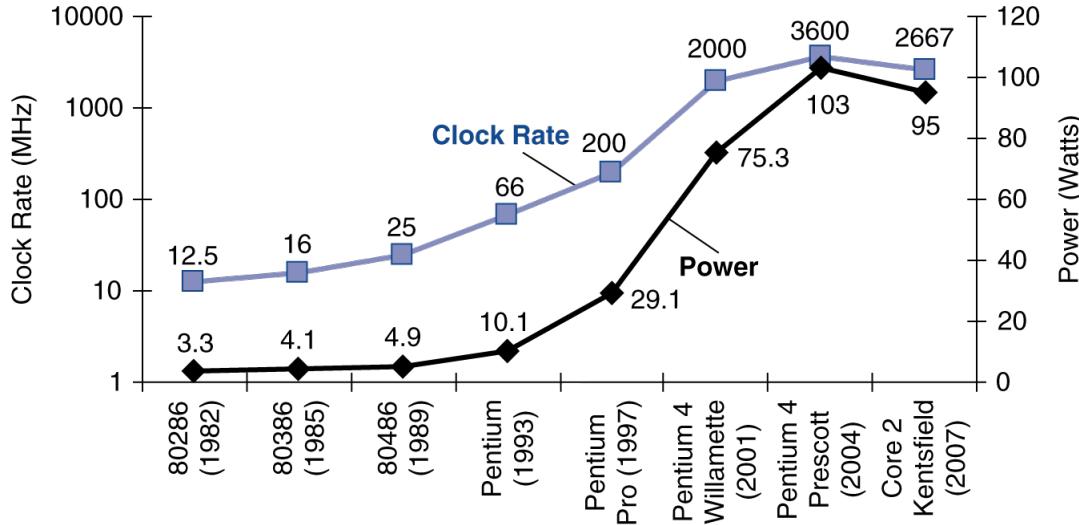


Figure 1.12 Growth in Transistor Count on Integrated Circuits (DRAM memory)

Power Trends



- In CMOS IC technology

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

×30

5V → 1V

×1000

Problems with Clock Speed and Login Density

- Power
 - Power density increases with density of logic and clock speed
 - Dissipating heat
- RC delay
 - Speed at which electrons flow limited by resistance and capacitance of metal wires connecting them
 - Delay increases as the RC product increases
 - As components on the chip decrease in size, the wire interconnects become thinner, increasing resistance
 - Also, the wires are closer together, increasing capacitance
- Memory latency
 - Memory speeds lag processor speeds

Reducing Power

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

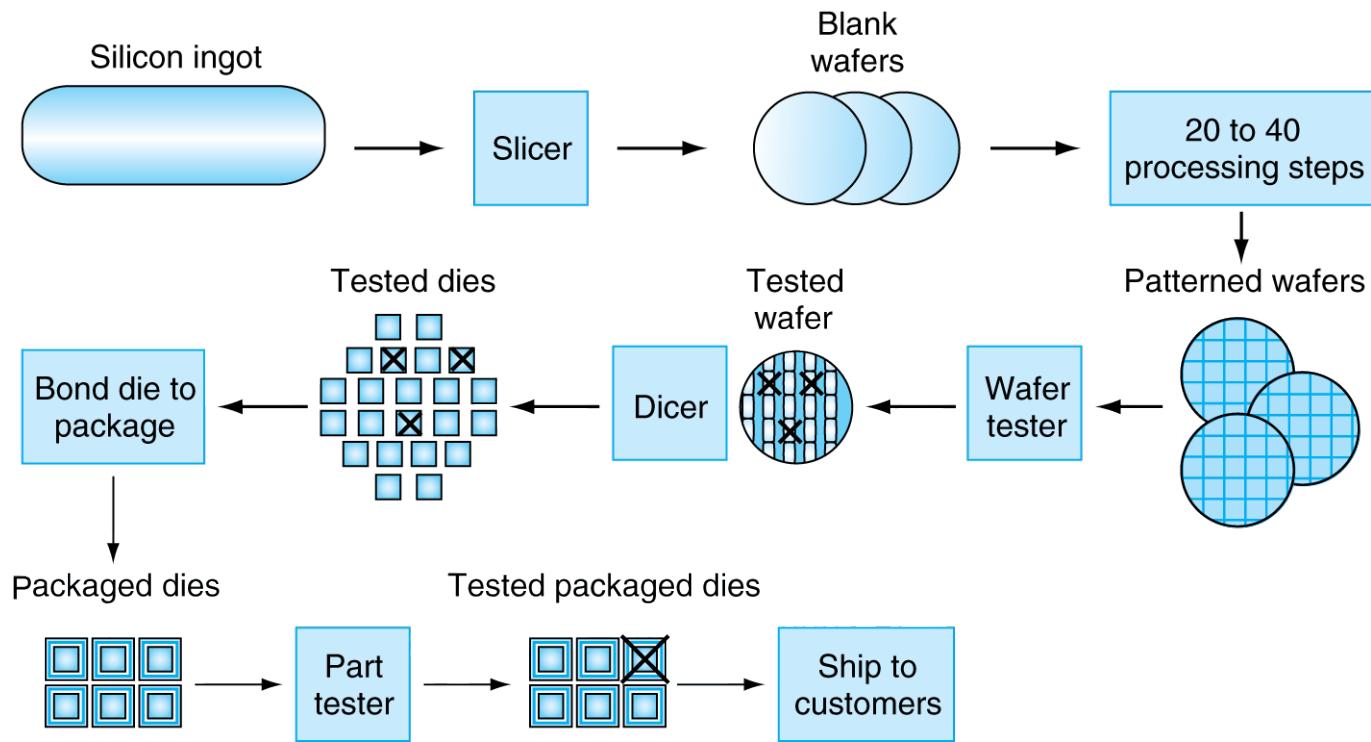
$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

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

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: proportion of working dies per wafer

Wafer

- A thin **wafer** of silicon is divided into a matrix of small areas, each a few millimeters square.
- The identical circuit pattern is fabricated in each area, and the wafer is broken up into **chips**.
- Each chip consists of many gates and/or memory cells plus a number of input and output attachment points.
- This chip is then packaged in housing that protects it and provides pins for attachment to devices beyond the chip.
- A number of these packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.

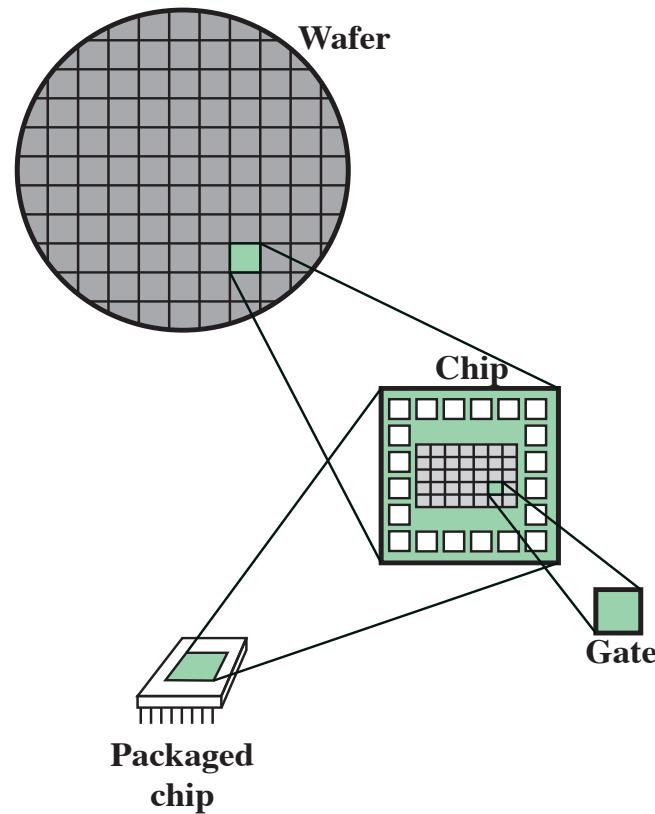


Figure 1.11 Relationship Among Wafer, Chip, and Gate

Benchmark Principles

- The same set of programs can be run on different machines and the execution times compared.
- Benchmarks provide guidance to customers trying to decide which system to buy and can be useful to vendors and designers in determining how to design systems to meet benchmark goals.
- Desirable characteristics of a benchmark program:
 1. It is written in a high-level language, making it portable across different machines
 2. It is representative of a particular kind of programming domain or paradigm, such as systems programming, numerical programming, or commercial programming
 3. It can be measured easily
 4. It has wide distribution



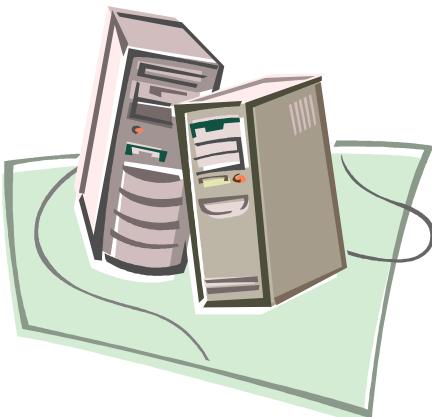
Standard Performance Evaluation Corporation (SPEC)

- Programs used to measure performance
 - Supposedly typical of actual workload
 - Develops benchmarks for CPU, I/O, Web, ...
 - Performance measurements are widely used for comparison and research purposes
- SPEC CPU2006
 - Elapsed time to execute a selection of programs
 - Negligible I/O, so focuses on CPU performance
 - Normalize relative to reference machine
 - Summarize as geometric mean of performance ratios
 - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^n \text{Execution time ratio}_i}$$

Standard Performance Evaluation Corporation SPEC CPU2006

- Best known SPEC benchmark suite
- Industry standard suite for processor intensive applications
- Appropriate for measuring performance for applications that spend most of their time doing computation rather than I/O
- Consists of 17 floating point programs written in C, C++, and Fortran and 12 integer programs written in C and C++
- Contains over 3 million lines of code
- Fifth generation of processor intensive suites from SPEC



Amdahl's Law

- Gene Amdahl
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
 - Software must be adapted to a highly parallel execution environment to exploit the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system

Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

- Can't be done!

- Corollary: make the common case fast

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

- CPI varies between programs on a given CPU