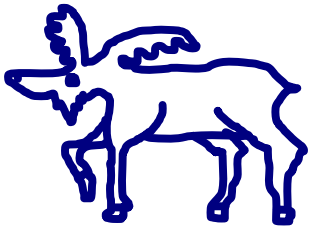
Lecture 1
Introduction and Performance

Byung-gi Kim

School of Computer Science and Engineering

Soongsil University



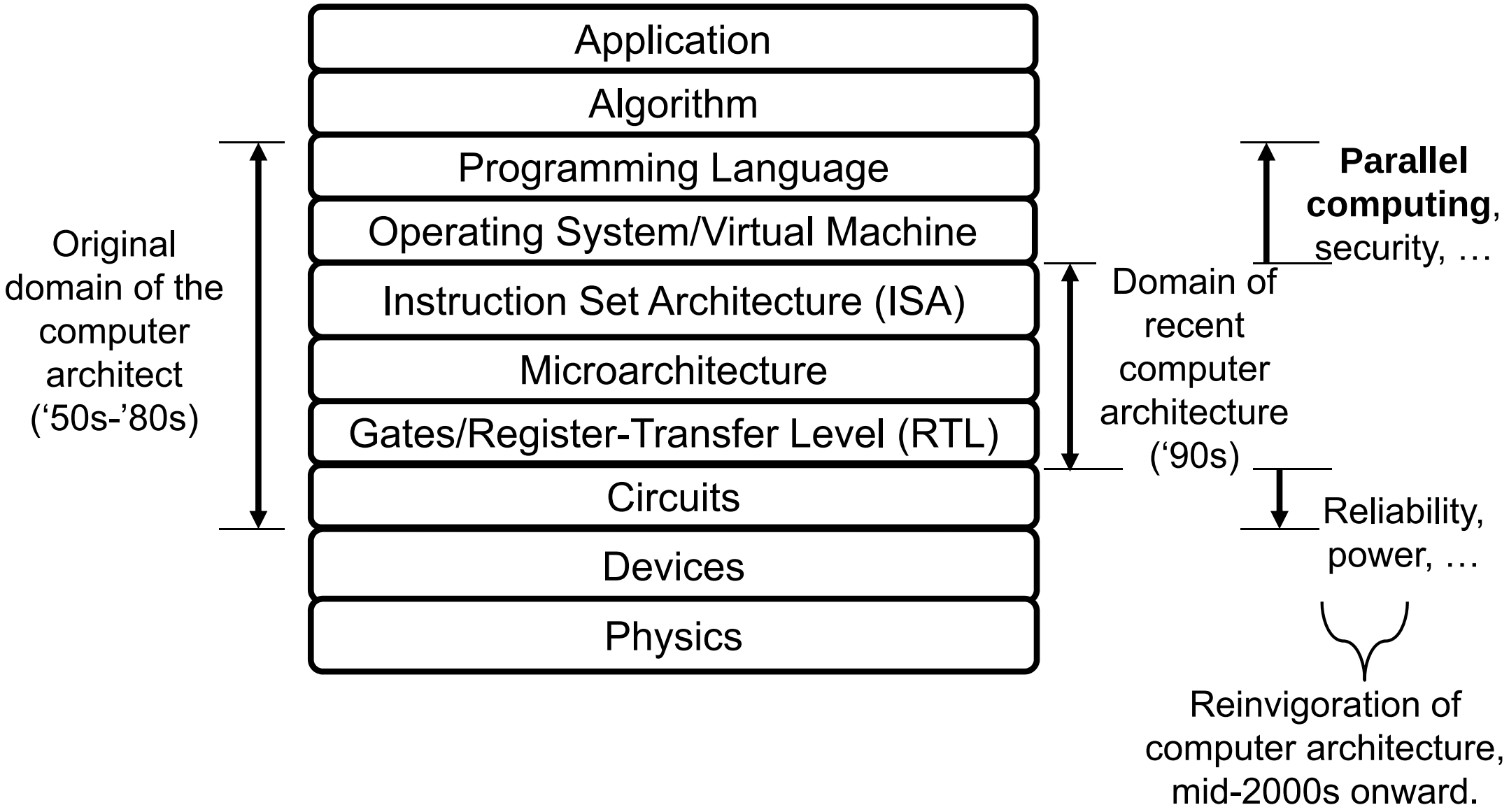
1. Computer Abstractions and Technology

What You Can Learn in This Class

- How programs are translated into the machine language and how the hardware executes them
  - understanding the aspects of both the hardware and software that affect program performance
- The hardware/software interface and how does software instruct the hardware to perform needed functions
- What determines program performance and how it can be improved
- How hardware designers improve performance

  [ref] *Computer Architecture: A Quantitative Approach*
- The reasons for and the consequences of the recent switch from sequential processing to parallel processing
  - "multicore" microprocessors

Abstraction Layers in Modern Systems



Computer Architecture

- = Instruction set architecture (ISA)
- "… the attributes of a [computing] system as seen by the programmer, *i.e.* the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation."
  – Amdahl, Blaauw, and Brooks, 1964
- Old definition of computer architecture = instruction set design
  - Other aspects of computer design called implementation
  - Insinuates implementation is uninteresting or less challenging
- Architect's job much more than instruction set design; technical hurdles today *more* challenging than those in instruction set design

Computer Architecture from Wikipedia

- The way by which the CPU performs internally and accesses memory
- Conceptual design and fundamental operational structure of a computer system
- Science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.
- It forms a blueprint and functional description of requirements and design implementations for the various parts of a computer, focusing largely on the way by which the central processing unit (CPU) performs internally and accesses addresses in memory.

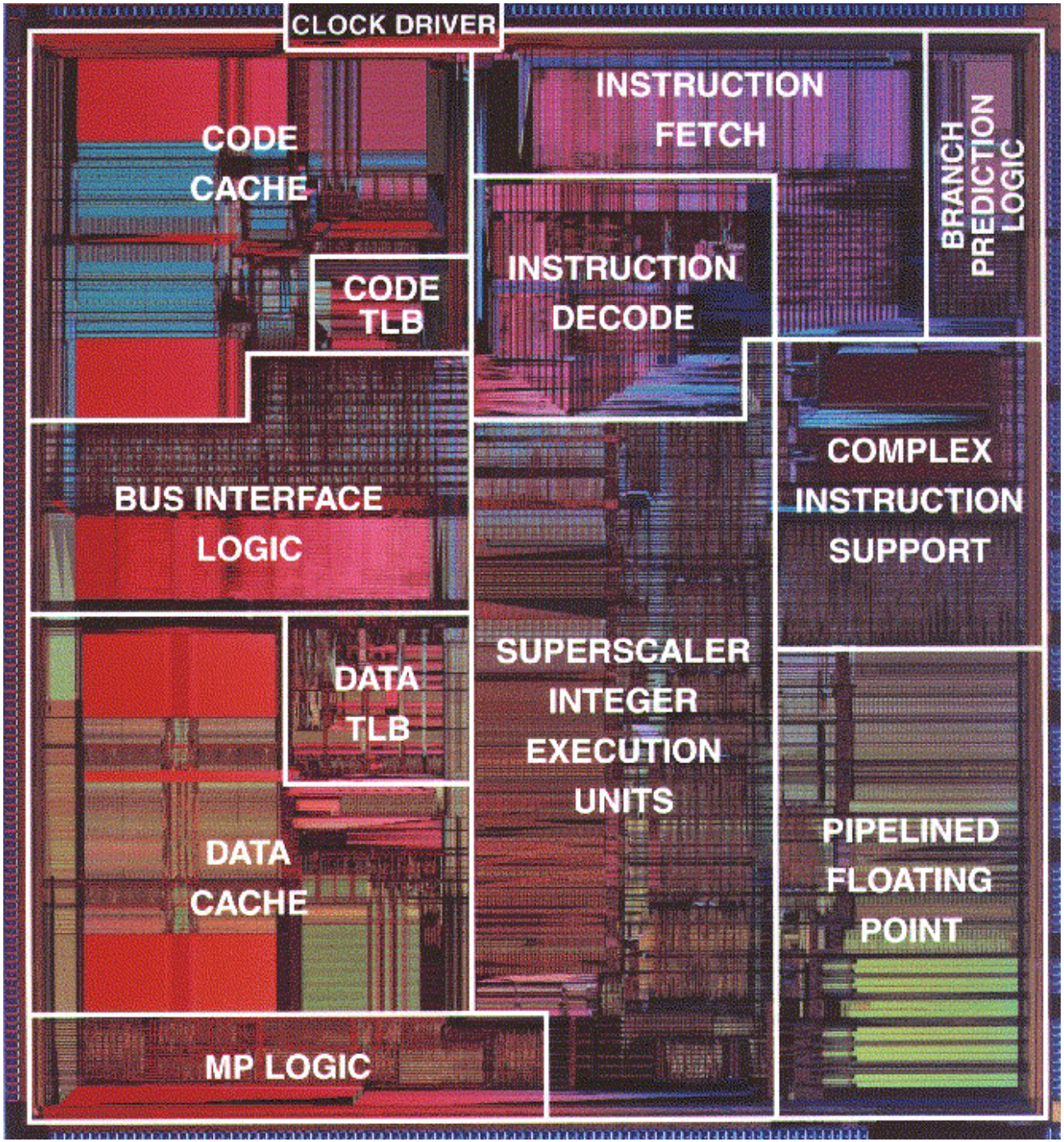
3 Subcategories of Computer Architecture

1. Instruction set architecture (ISA)
   - Abstract image of a computing system that is seen by a machine language (or assembly language) programmer
   - Including the instruction set, word size, memory address modes, processor registers, and address and data formats.
2. Microarchitecture (aka Computer organization)
   - Lower level, more concrete and detailed, description of the system
   - How the constituent parts of the system are interconnected and how they interoperate in order to implement the ISA
   - For example the size of a computer's cache
3. System Design
   - All of the other hardware components within a computing system
   - For example, system interconnects such as computer buses and switches, memory controllers and hierarchies ,CPU off-load mechanisms such as direct memory access (DMA) and issues like multiprocessing

Architecture vs. Implementation

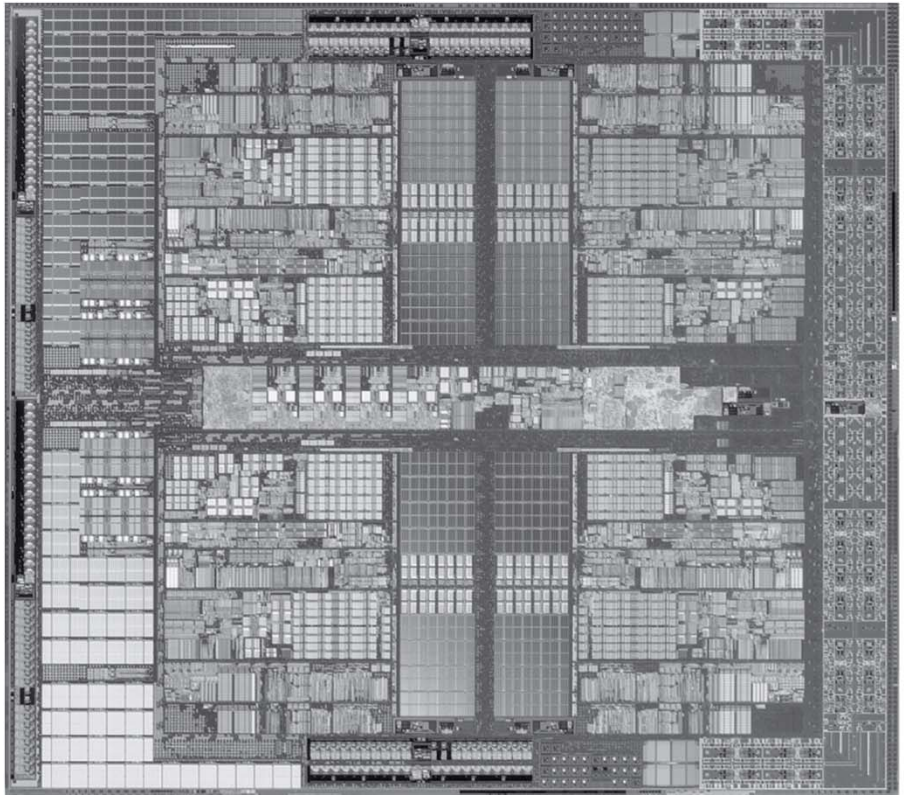
- **Instruction set architecture (=Architecture)**
  - Interface between hardware and lowest-level software
  - Defines what a computer system does in response to a program and a set of data
  - Includes all the information necessary to write a machine language program that will run correctly, including instructions, registers, memory access, I/O, ….
  - MIPS, IA-32, IA-64, Sparc, PowerPC, IBM 390, etc.
- **Implementation (=Organization)**
  - Hardware that obeys the architecture abstraction
  - Defines how a computer does in response to a program and a set of data
  - 8086, 386, 486, Pentium, Pentium II, Pentium 4, etc.

Intel Pentium



AMD Barcelona Processor

- **4 processors or “cores”**



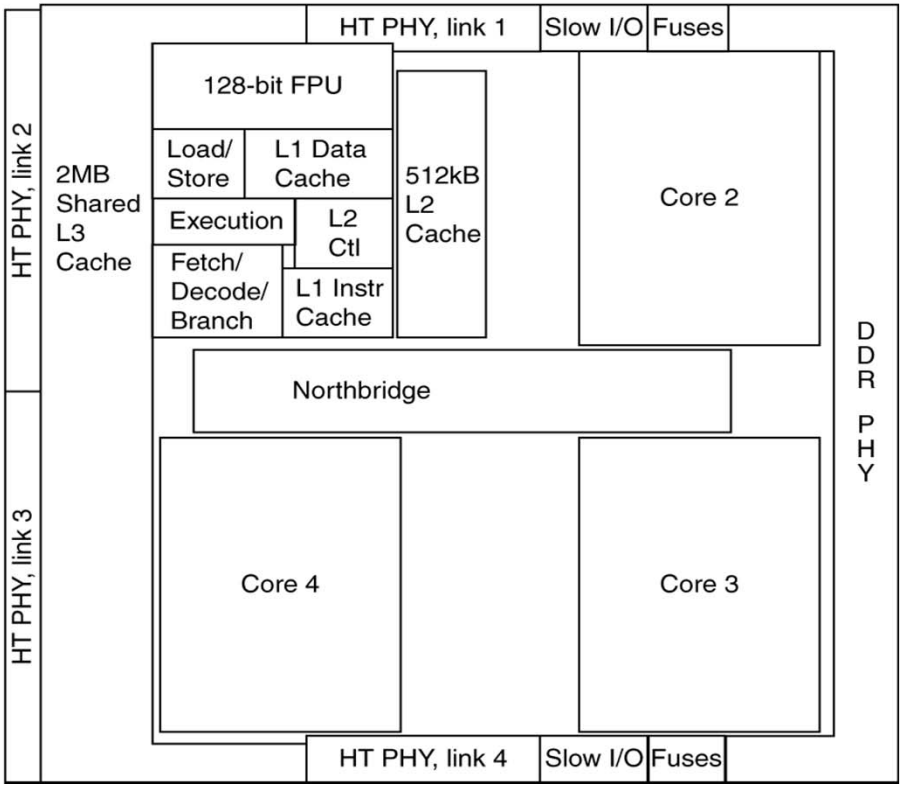


Figure 1.9

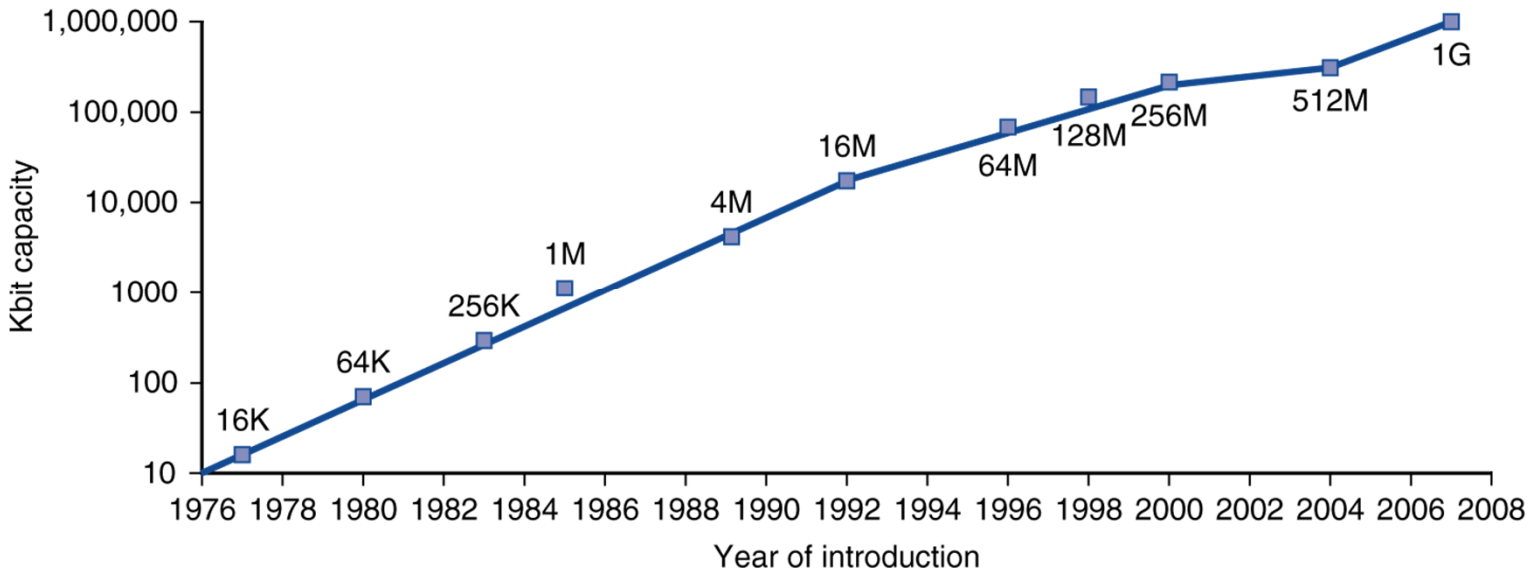
1.4 Performance

Year	Technology	Relative performance / unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	IC	900
1995	VLSI	2,400,000
2005	ULSI	6,200,000,000

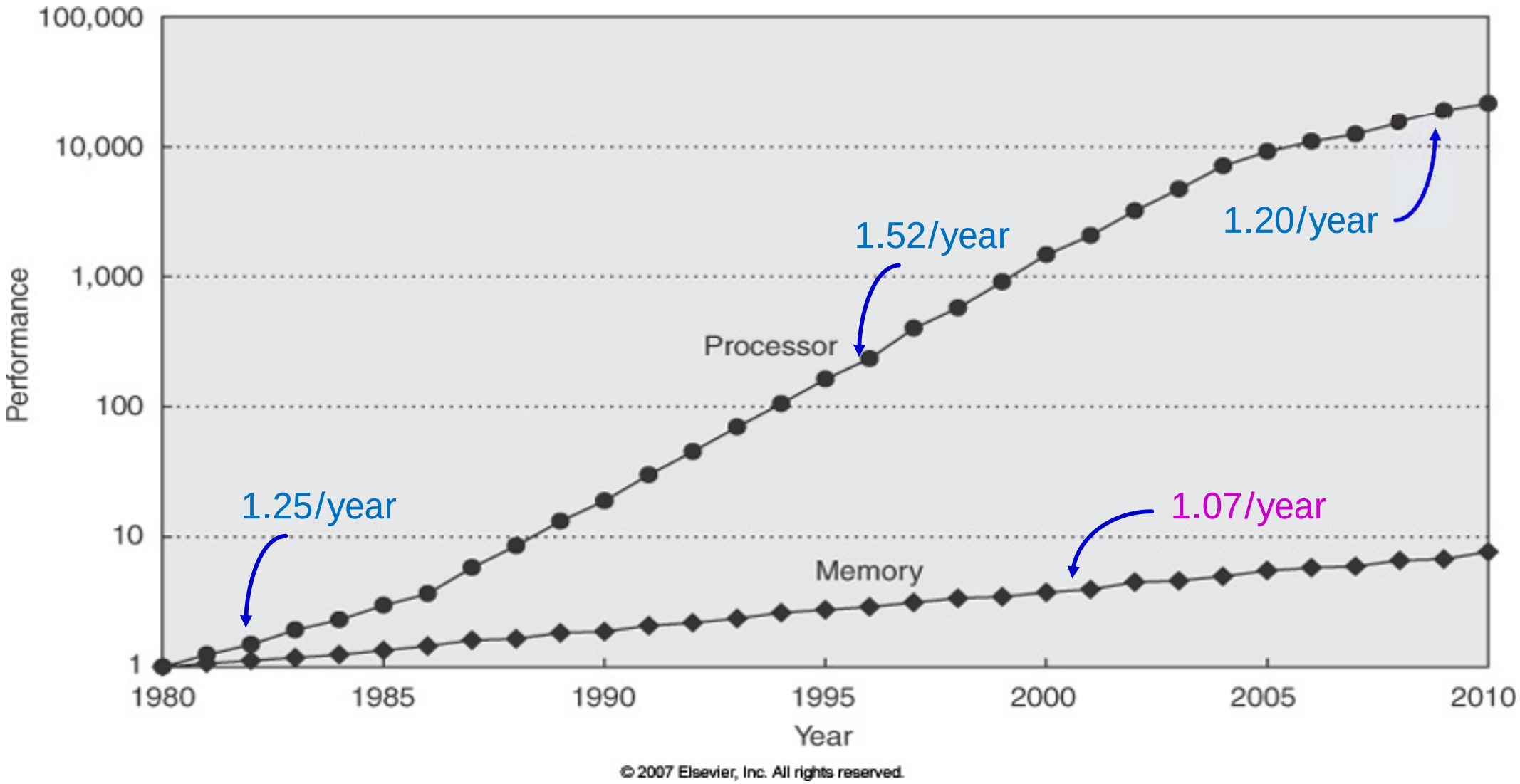
Figure 1.12

- **Moore's law**

Figure 1.11

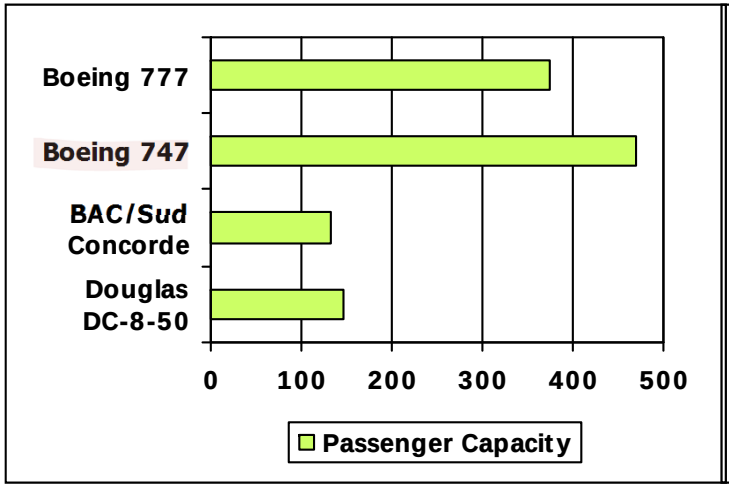


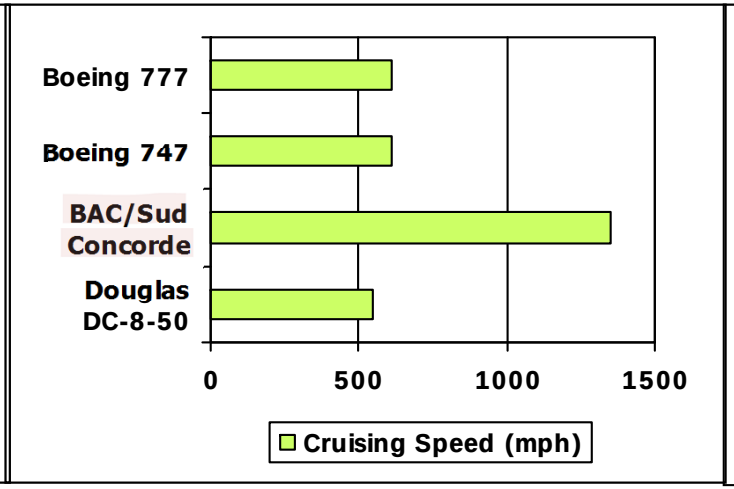
Processor-Memory Performance Gap

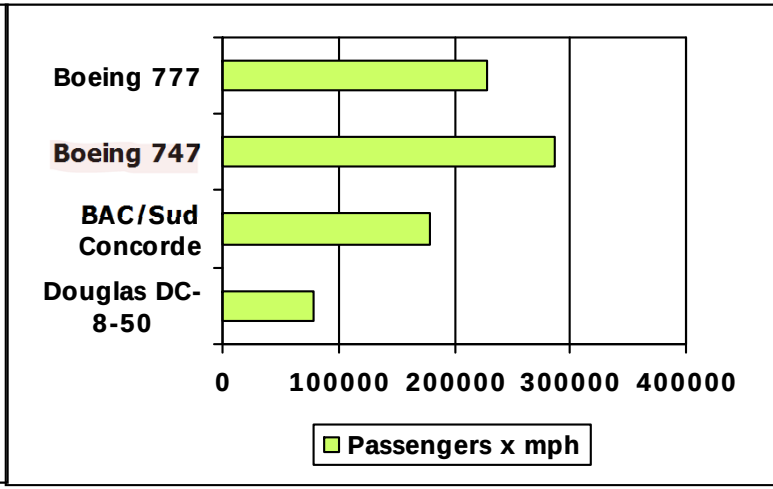


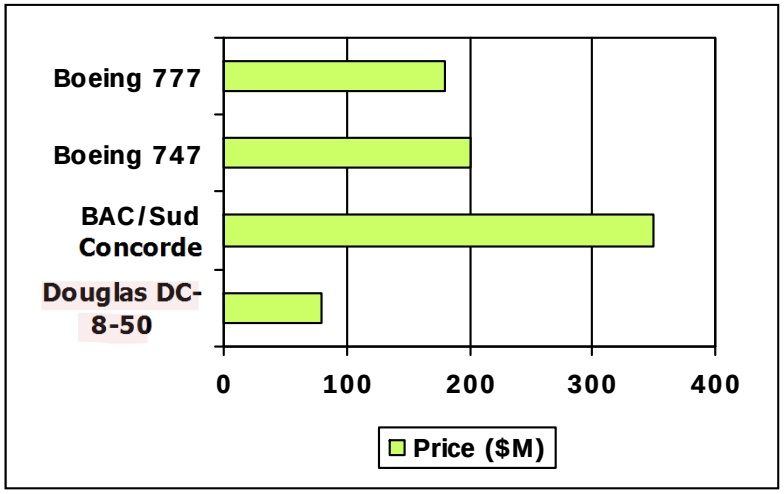
Defining Performance

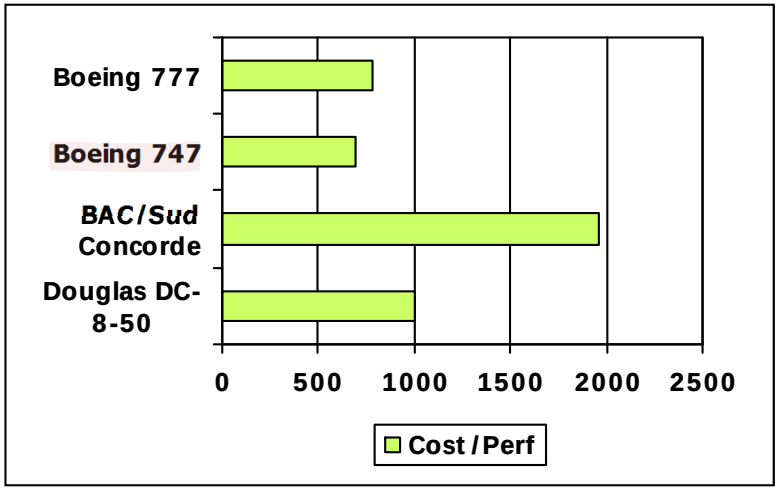
- **Which airplane is the best ?**











Performance of a Computer

- **Response time (= execution time)**
  - The time between the start and completion of a task
- **Throughput (= bandwidth)**
  - The number of tasks completed per unit time
- **Performance and execution time**
  - $\text{Performance}_x = 1 / \text{Execution time}_x$
- **X is n times faster than Y**

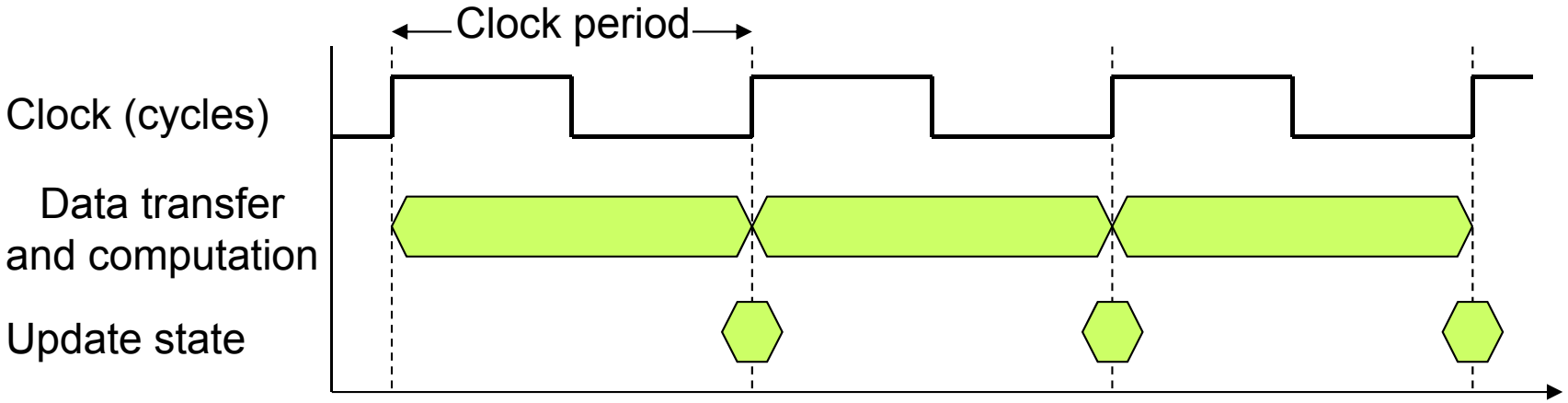
$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Exceution Time}_X} = n$$

Measuring Performance

- **Definitions of time**
  - Wall-clock time = Response time = Elapsed time
    - Total time to complete a task
    - Including disk accesses, memory accesses, I/O activities, OS overhead and etc.
    - Determines system performance
  - CPU execution time = CPU time
    - The time CPU spends computing for this task
    - Not including time spent waiting for I/O or running other programs
    - CPU time = User CPU time + System CPU time
    - Determines CPU performance
- **Definitions of performance**
  - System performance: based on elapsed time
  - CPU performance: based on user CPU time
- **We will focus on CPU performance for now!**

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock
- **Clock period=clock cycle time: duration of a clock cycle**
  - e.g., 250ps = 250×10^{-12}s= 0.25ns = 0.25×10^{-9}s
- **Clock frequency (rate): cycles per second**
  - e.g., 1/ (0.25×10^{-9}s) = 4.0×10^{9}Hz = 4000MHz = 4.0GHz



CPU Performance and Its Factors

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

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

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

Example: Improving Performance

- 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 Performance

Clock Cycles = IC x CPI

- **Instruction Count (IC) for a program**
  - Determined by program, ISA and compiler
- **Average Cycles Per Instruction (CPI)**
  - Determined by CPU hardware
  - If different instructions have different CPI,
    - average CPI affected by instruction mix

Example: Using the Performance Equation

- Same instruction set architecture, same program
- Clock cycle time$_A$ = 250ps, CPI$_A$ = 2.0
- Clock cycle time$_B$ = 500ps, CPI$_B$ = 1.2
- **Which is faster, and by how much ?**

[Answer]

- Let I = instruction count for the program.
- CPU time$_A$ = IC$_A$ x CPI$_A$ x clock cycle time$_A$
  = I x 2.0 x 250 ps = 500 x I ps
- CPU time$_B$ = I x 1.2 x 500 ps = 600 x I ps
- Then

$$\frac{\text{CPU Performanc e}_A}{\text{CPU Performanc e}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times \text{I ps}}{500 \times \text{I ps}} = 1.2$$

- Thus, A is 1.2 times faster than B for this program.

The Classic CPU Performance Equation

$$\text{CPU Time} = \text{IC} \times \text{CPI} \times \text{clock cycle time}$$

$$= (\text{IC} \times \text{CPI}) / \text{clock rate}$$

- **Example: Comparing Code Segments**
  - **Which code sequence executes the most instructions?**
  - **Which will be faster ?**
  - **What is the CPI for each sequence ?**

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

[Answer]

- instruction count$_1$ = 2 + 1 + 2 = 5 and
  instruction count$_2$ = 4 + 1 + 1 = 6
  Thus (1) executes fewer instructions.
- CPU clock cycles$_1$ = 2x1 + 1x2 + 2x3 = 10 and
  CPU clock cycles$_2$ = 4x1 + 1x2 + 1x3 = 9
  Thus (2) is faster.
- CPI$_1$ = CPU clock cycles$_1$ / instruction count$_1$
  = 10 / 5 =2
  CPI$_2$ = 9 / 6 = 1.5
  (2) has lower CPI.

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

Supplement

1.1 Introduction

- **Progress in computer technology**
  - Underpinned by Moore's Law
  - Doubling every 18 months
- **Makes novel applications feasible**
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- **Computers are pervasive**
  - Ubiquitous computing

Classes of Computing Applications and Their Characteristics

1. **Desktop computers**
   - Personal computers
   - Good performance to a single user at low cost
   - Third-party software
2. **Servers**
   - Modern form of mainframes, mini- and supercomputers
   - Usually accessed via a network
   - Expandability and dependability
   - Low-end servers, supercomputers, computer clusters
3. **Embedded computers**
   - A computer inside another device used for running one predetermined application or collection of software
   - Minimum performance with stringent limitations on cost or power

1.5 The Power Wall

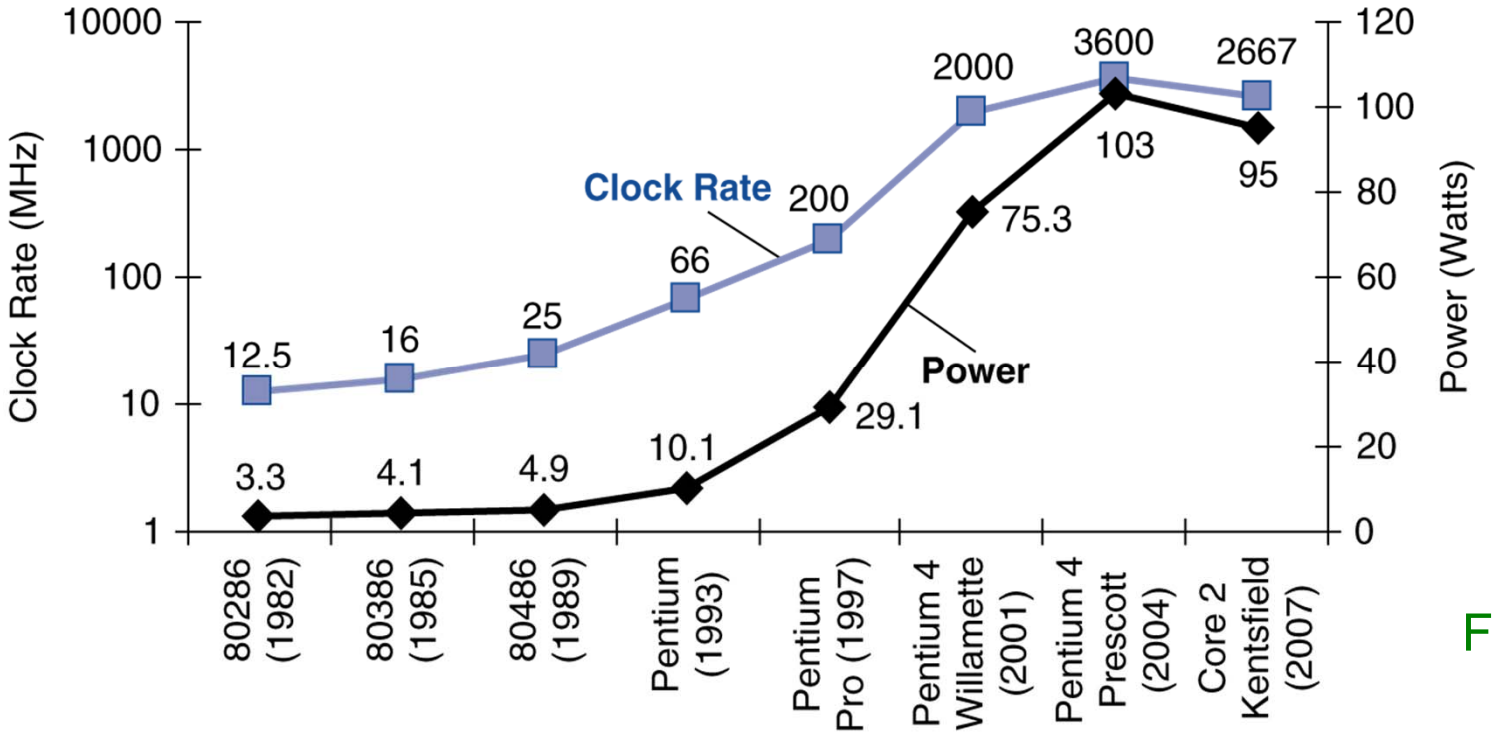


Figure 1.15

- **In CMOS IC technology**

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

$\times 30$ $\quad$ $5V \rightarrow 1V$ $\quad$ $\times 1000$

Example: Relative Power

- **Suppose a new, simpler CPU has**
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction
  - **What is the impact on dynamic power?**

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

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

1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors

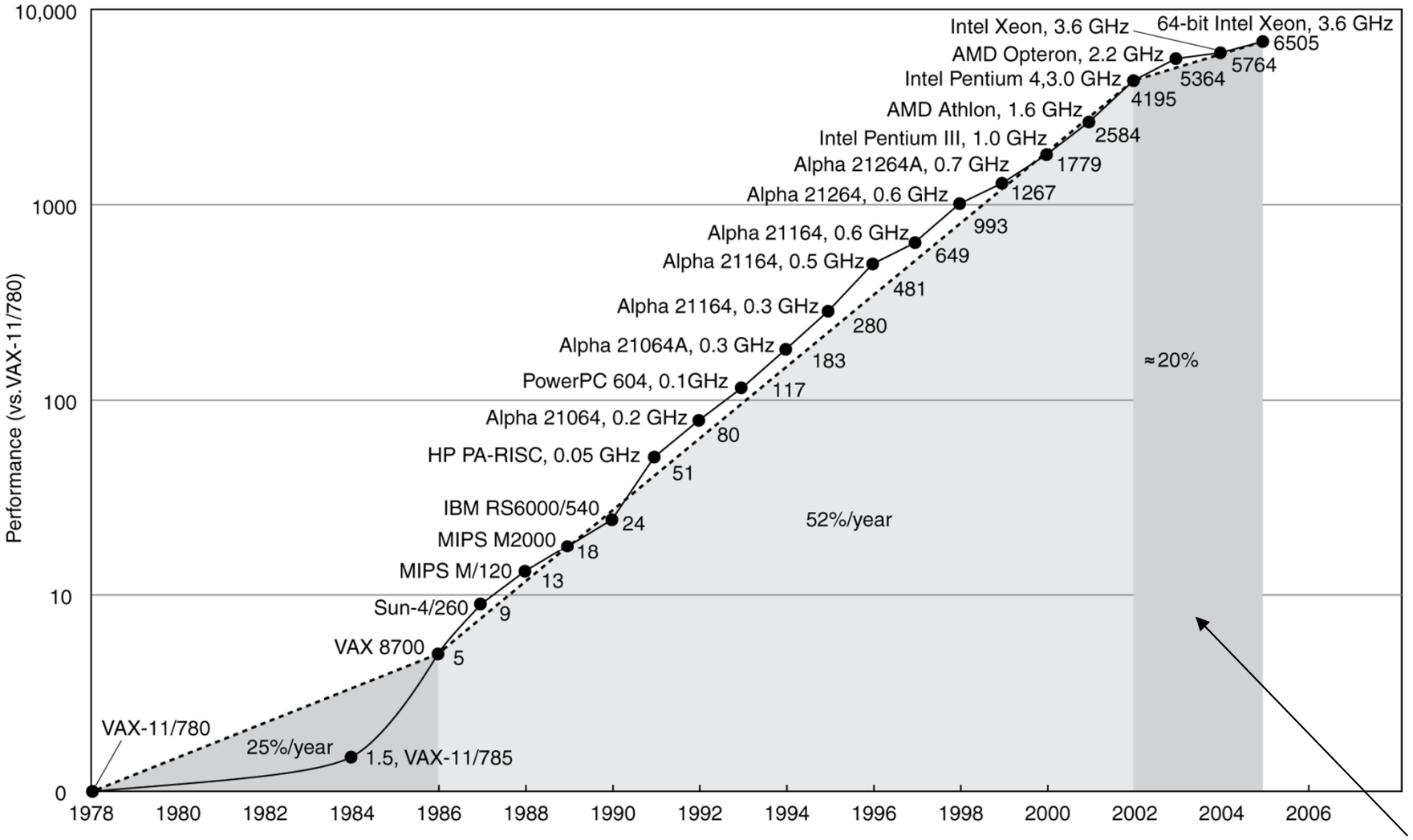


Figure 1.16

Constrained by power, instruction-level parallelism, memory latency

Multicore Microprocessors

- **Multicore**
  - Rather than continuing to decrease the response time of a single program running on the single processor,
  - as of 2006 all desktop and server companies are shipping microprocessors with multiple processors per chip
  - The benefit is often more on throughput than on response time
- **2008 multicore microprocessors (Figure 1.17)**

	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	2.67/2.93/3.2 GHz	4.7 GHz	1.4 GHz
Power	120 W	130 W	~ 100 W ?	94 W

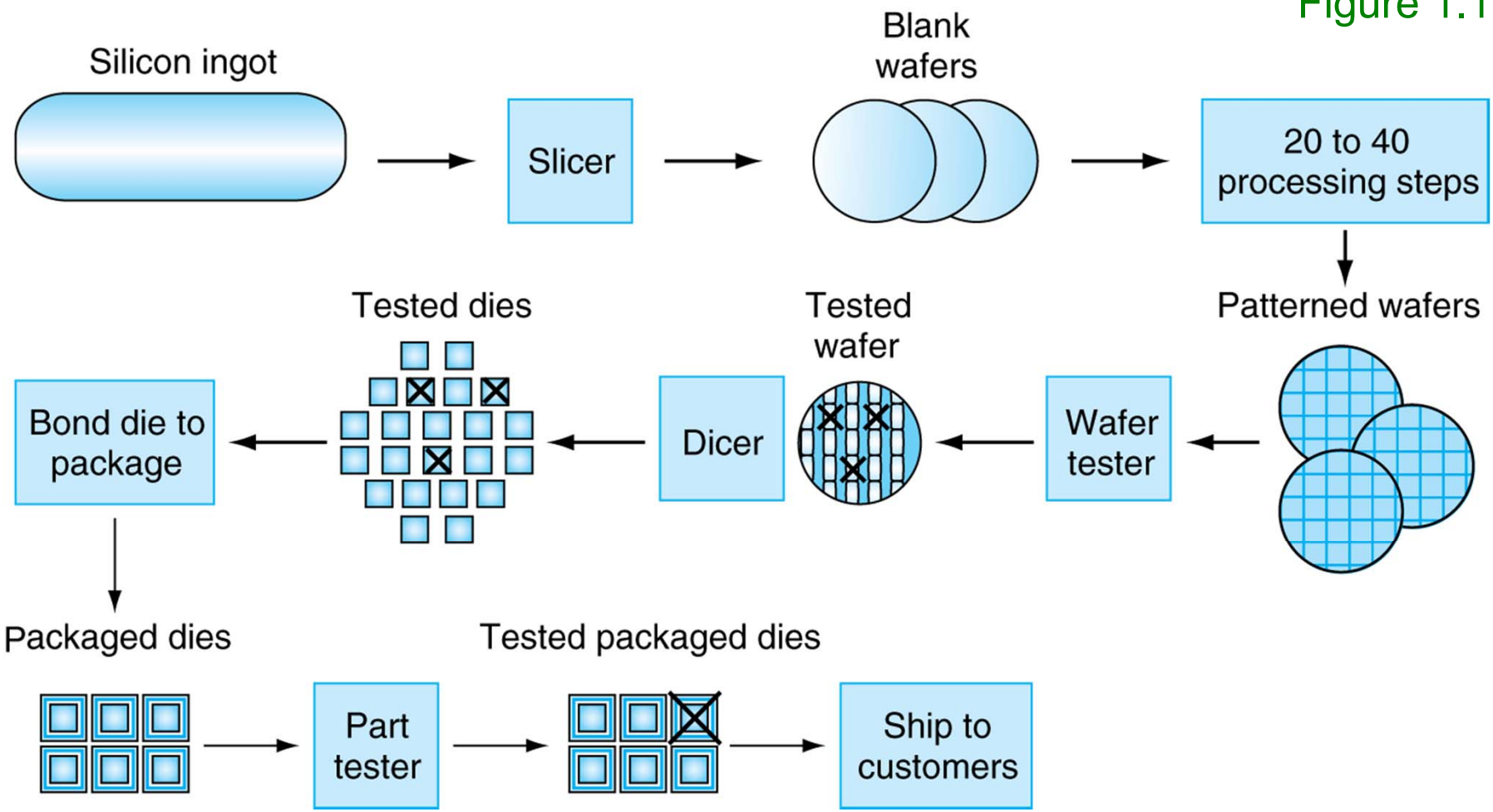
Explicitly Parallel Programming

- **Explicitly parallel programming**
  - Forcing programmers to be aware of the parallel hardware and to explicitly rewrite their programs to be parallel
- **Instruction level parallelism (ILP)**
  - Hardware executes multiple instructions at once
  - Hidden from the programmer
- **Why it has been so hard for programmers**
  1. Programming for performance
     - Parallel programming is by definition performance programming
  2. Need to divide an application
     - Scheduling subtasks
     - Load balancing
     - Optimizing communication and synchronization

1.7 Real Stuff: Manufacturing and Benchmarking the AMD Opteron X4

- **Manufacturing process for ICs**

Figure 1.18



AMD Opteron X2 Wafer

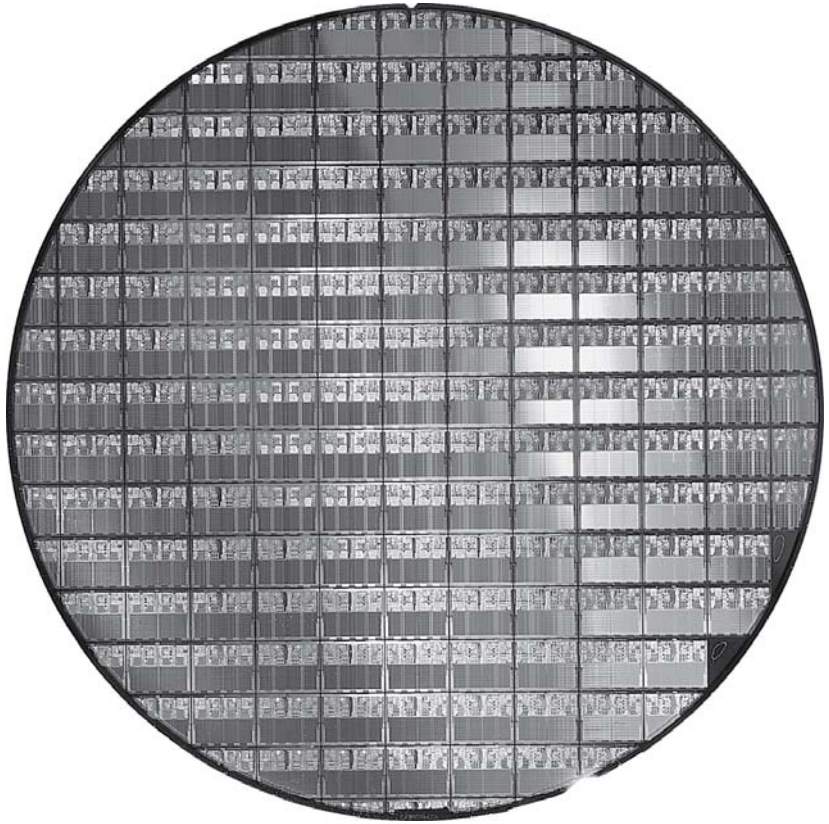


Figure 1.19

- **X2: 300mm wafer, 117 chips, 90nm technology**
- **X4: 45nm technology**

Elaboration

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

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

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

- **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

SPEC CPU Benchmark

- **Benchmark**
  - A program selected for use in comparing computer performance
  - Forming a workload that the user hopes will predict the performance of the actual workload
- **Standard Performance Evaluation Corporation (SPEC)**
  - Founded in 1988
  - Members
- **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)

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							**11.7**

High cache miss rates

Figure 1.20

SPEC Power Benchmark

- **SPECpower**
  - Power consumption of server at different workload levels
  - Divided into 10% increments, over a period of time
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

- **Stochastic Simulation in Java (SSJ)**
  - A Java library for stochastic simulation

SPECpower_ssj2008 for X4

- Running on dual socket 2.3 GHz AMD Opteron X4 2356 (Barcelona) with 16 GB Of DDR2-667 DRAM and one 500 GB disk

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		**493**

Figure 1.21

1.8 Fallacies and Pitfalls

- **Pitfall: Amdahl's Law**
  - Improving an aspect of a computer and expecting a proportional improvement in overall performance

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

- **Example:** multiply accounts for 80s/100s
  - **How much improvement in multiply performance to get 4× overall?**

$$\frac{100}{4} = \frac{80}{n} + 20 \quad \Rightarrow \quad 5 = \frac{80}{n} \quad \Rightarrow \quad n = 16$$

- **Corollary: make the common case fast**

Fallacy: Computers at low utilization use little power

- **Look back at X4 power benchmark**
  - At 100% load: 295W, 50% load: 246W (83%), 10% load: 180W (61%)
- **CPU utilization for servers at Google**
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Server Manufacturer	Microprocessor	Total Cores/ Sockets	Clock Rate	Peak Performance (ssj_ops)	100% Load Power	50% Load Power	50% Load/ 100% Power	10% Load Power	10% Load/ 100% Power	Active Idle Power	Active Idle/ 100% Power
HP	Xeon E5440	8/2	3.0 GHz	308, 022	269 W	227 W	84%	174 W	65%	160 W	59%
Dell	Xeon E5440	8/2	2.8 GHz	305, 413	276 W	230 W	83%	173 W	63%	157 W	57%
Fujitsu Siemens	Xeon X3220	4/1	2.4 GHz	143, 742	132 W	110 W	83%	85 W	65%	80 W	60%

Figure 1.22

Pitfall: Using a subset of the performance equation as a performance metric

- Must predict performance based on clock rate, IC and CPI
- **Millions of Instructions Per Second (MIPS)**

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

$$= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions
  - CPI varies between programs on a given CPU

1.9 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**

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

  - The best performance measure
- **Power is a limiting factor**
  - Use parallelism to improve performance