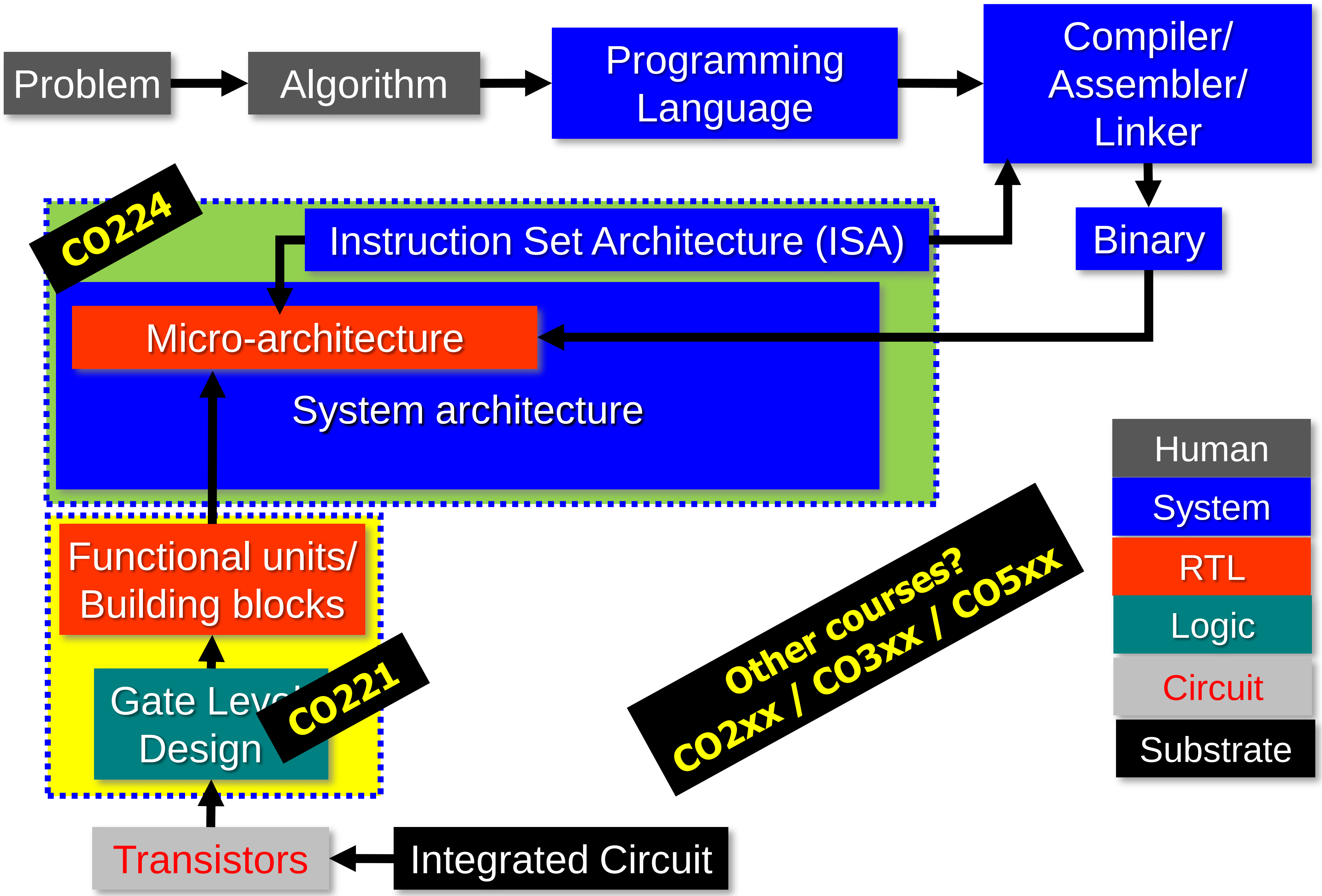
CO224: COMPUTER ARCHITECTURE

Isuru Nawinne

[Adapted from Computer Organization and Design, ARM Edition. Patterson & Hennessy, © 2011, MK]

The BIG Picture



CO224: Outline

Lectures: Computer Organization and Design

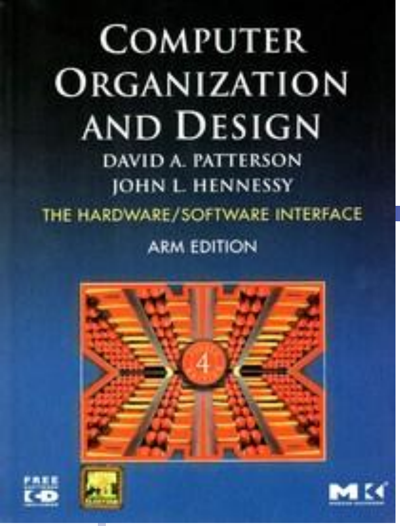
Ch01 – Computer Abstraction and Technology
Ch02 – Instructions – The Language of the Computer
Ch03 – Arithmetic for Computers
Ch04 – The Processor
Ch05 – Memory Hierarchy
Ch06 – Storage and Other IO
Ch07 – Multi-cores, Multi-processors and Clusters

Labs:

(1) Writing Assembly programs for ARM ISA
(2) Micro-architecture & systems design

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



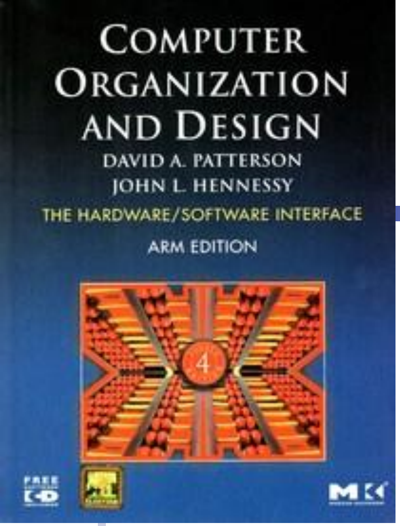
CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

Computer Abstractions

- What is abstraction ?!?
  - ………………..
- Can you figure out the order of these abstractions?
  - Organize them in a meaningful way…



CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

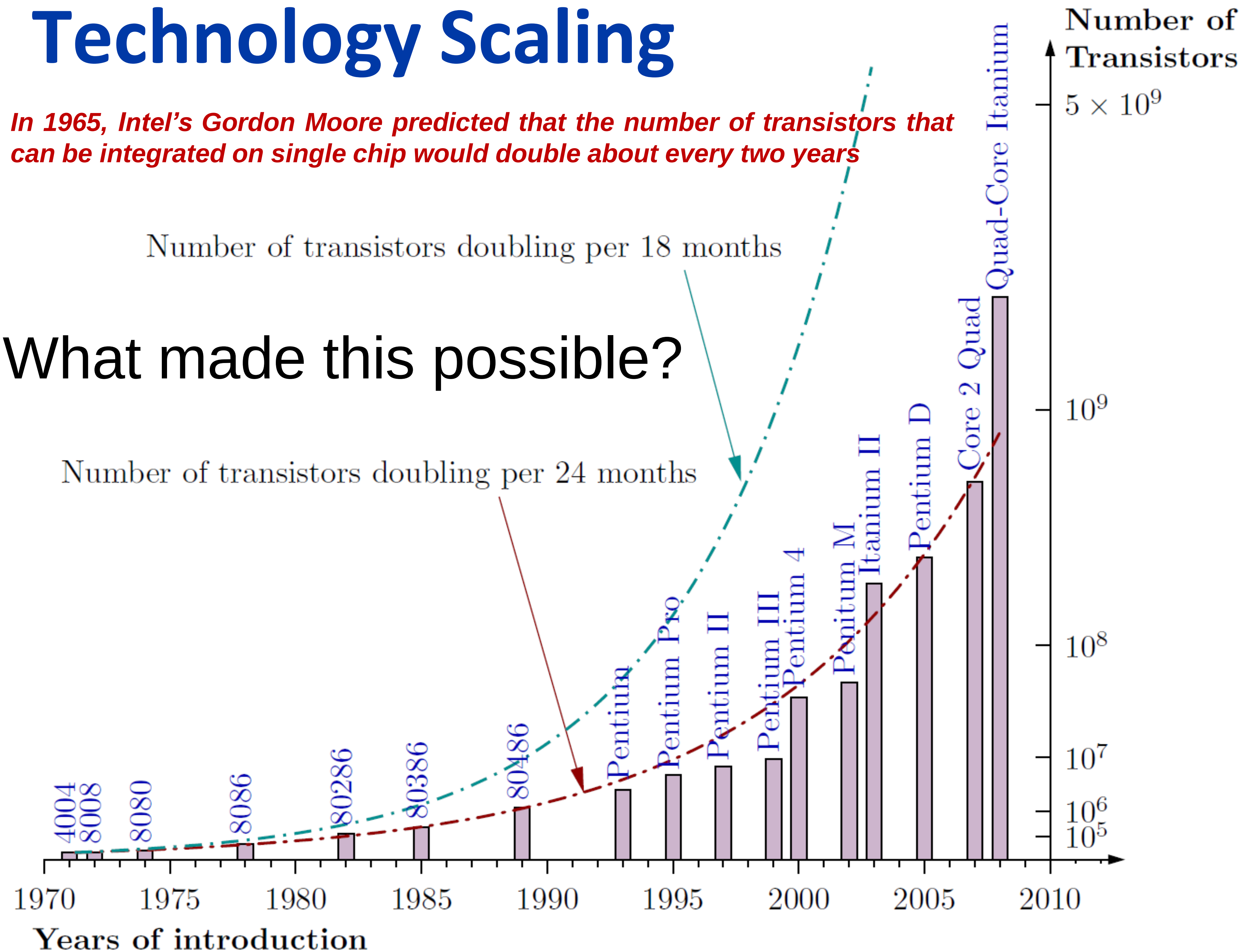
The Computer Revolution

- 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

Technology Scaling

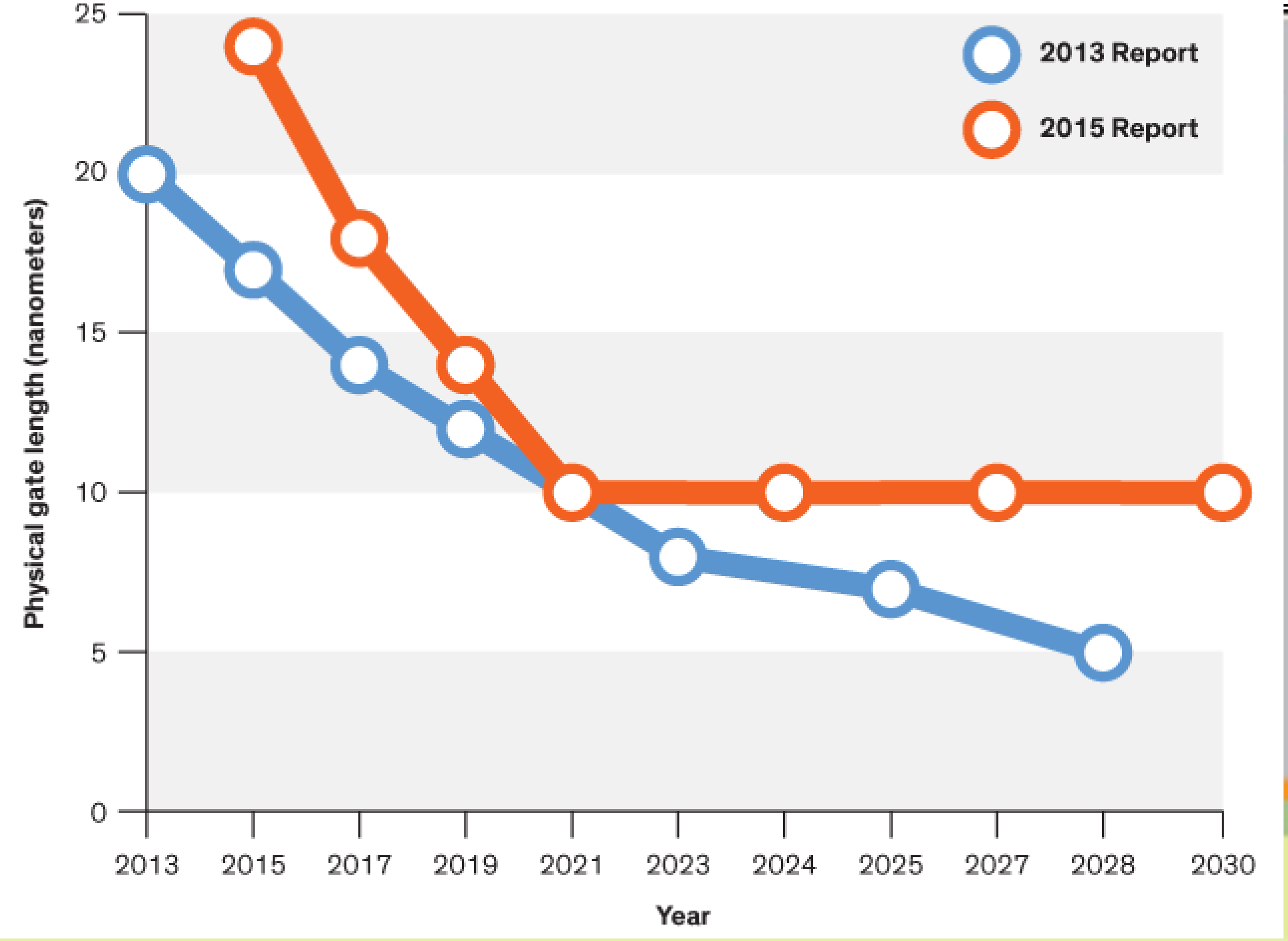
In 1965, Intel's Gordon Moore predicted that the number of transistors that can be integrated on single chip would double about every two years

- What made this possible?

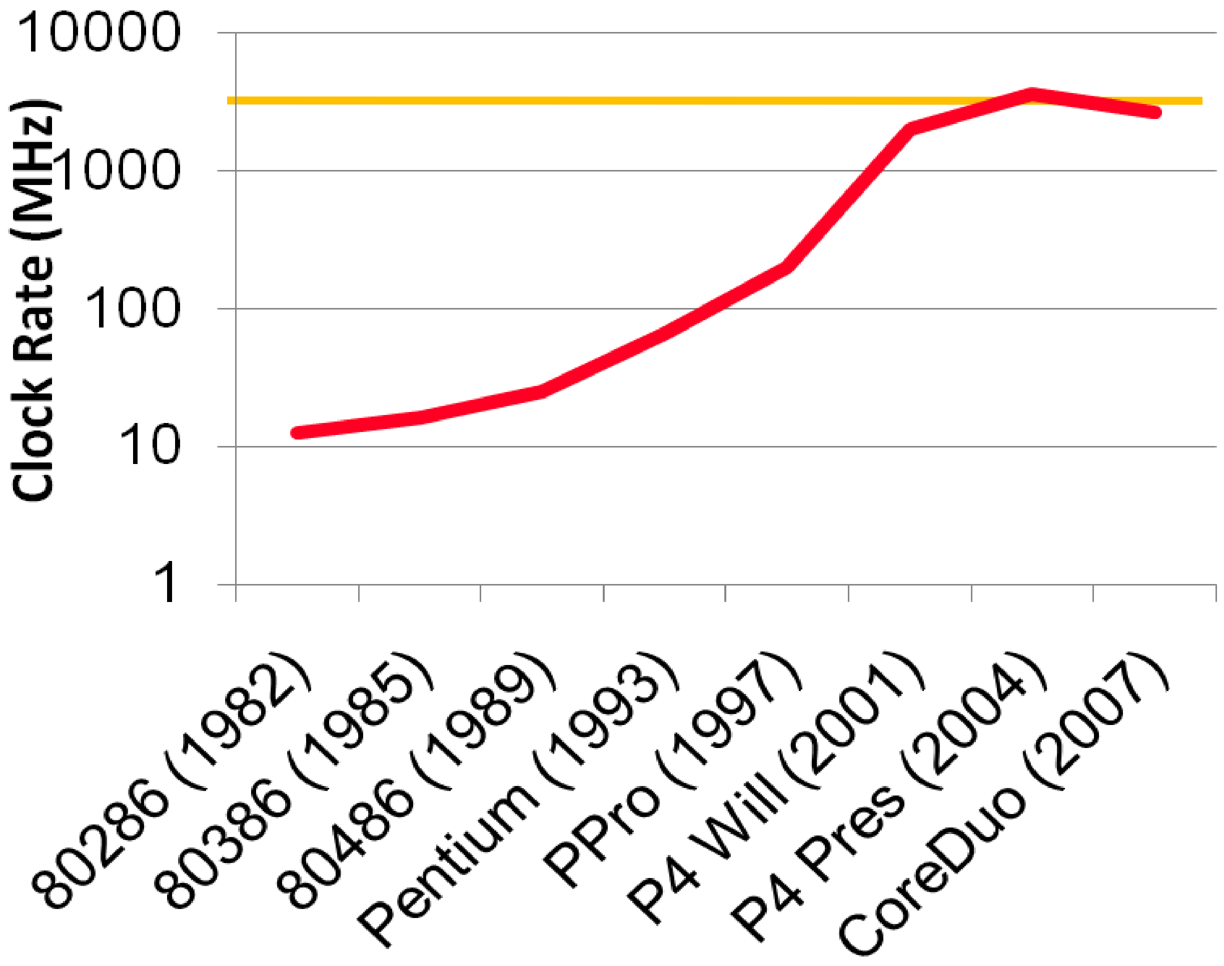


Technology Scaling Road Map (ITRS)

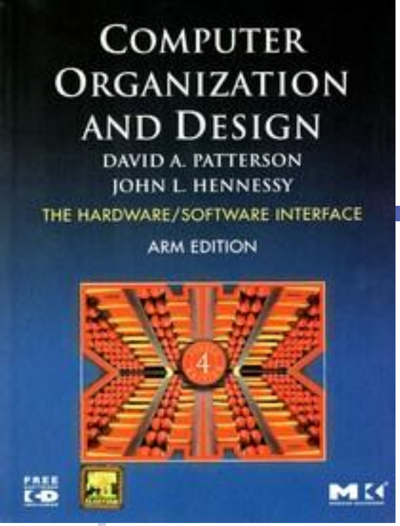
Year	2004	2006	2008	2010	2012
Feature size (nm)	90	65	45	32	22



Performance (Clock Freq) Scaling with Technology



- Clock freq hit a "power wall"

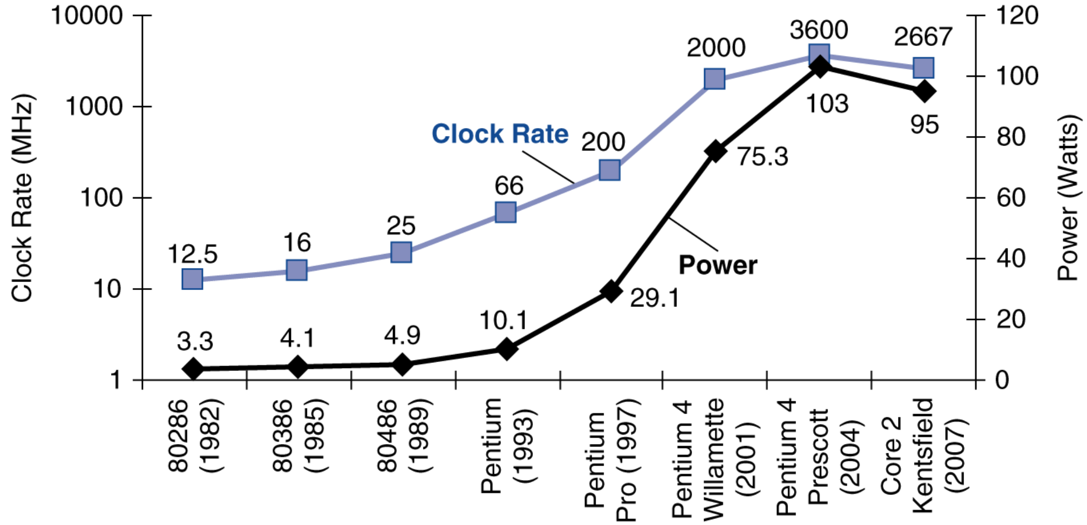


CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

Power Trends (Power Wall)



- Heat dissipation? Over-clocking?

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

×30

5V → 1V

×300

Alternative ways to improve performance

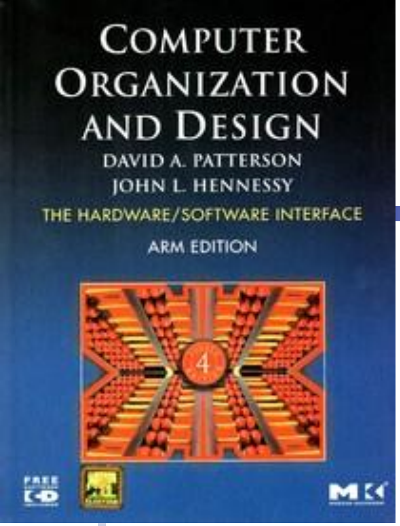
- The power challenge has forced a change in the design of microprocessors
  - Since 2002 the rate of improvement in the response time of programs on desktop computers has slowed from a factor of 1.5 per year to less than a factor of 1.2 per year
- As of 2006 all desktop and server companies are shipping microprocessors with multiple processors – cores – per chip

Product	AMD Barcelona	Intel Nehalem	IBM Power 6	Sun Niagara 2
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~2.5 GHz	4.7 GHz	1.4 GHz
Power	120 W	~100 W	~100 W	94 W

- **Plan of record was to double the number of cores per chip per generation (about every two years). Did that happen?**

Multi-processors

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

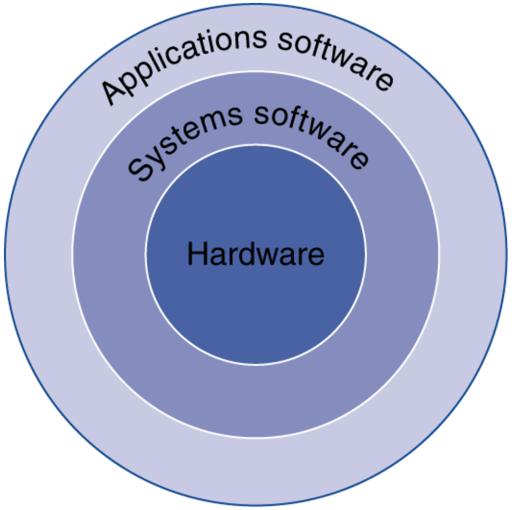


CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

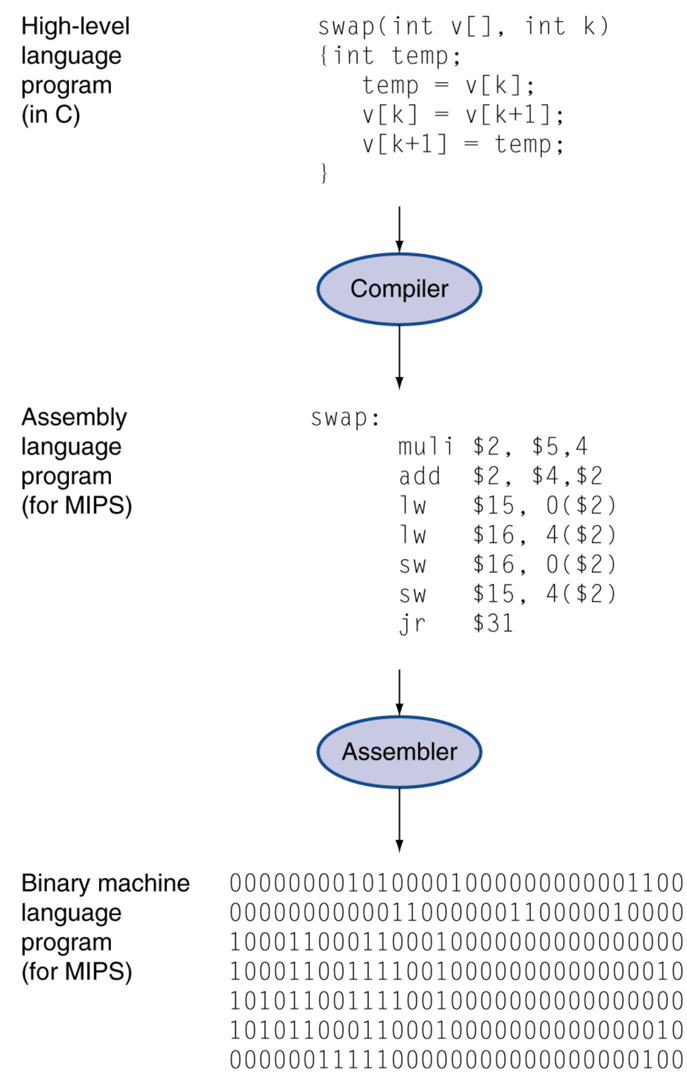
Below Your Program

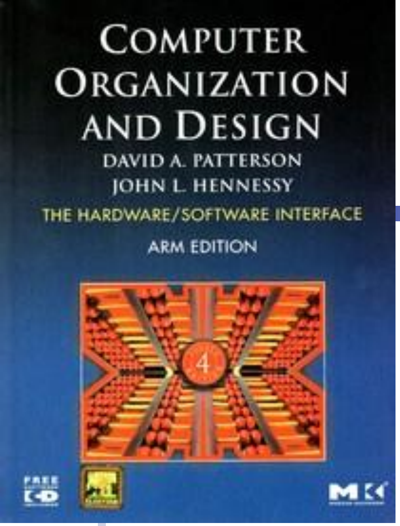


- Application software
  - Written in high-level language
- Systems software
  - Compiler / Assembler: translate HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - CPU, memory, 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



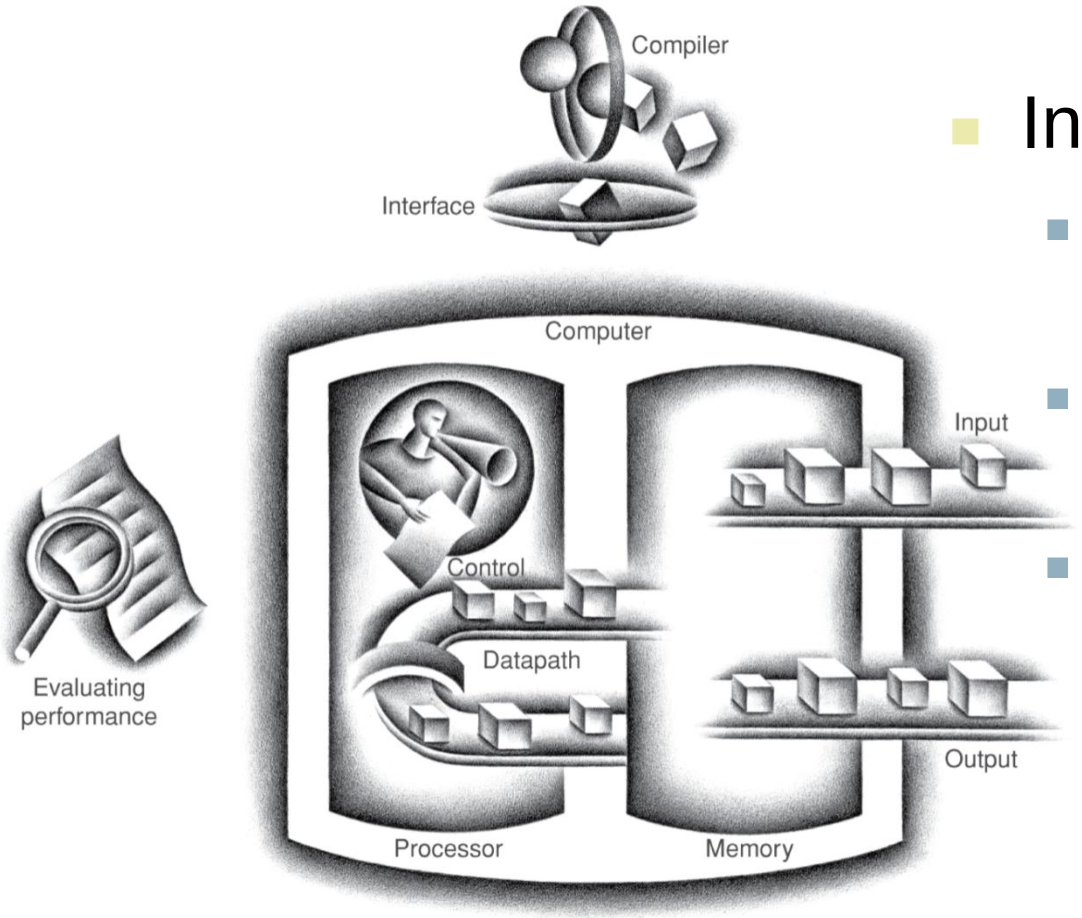


CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

What happens to the machine code…



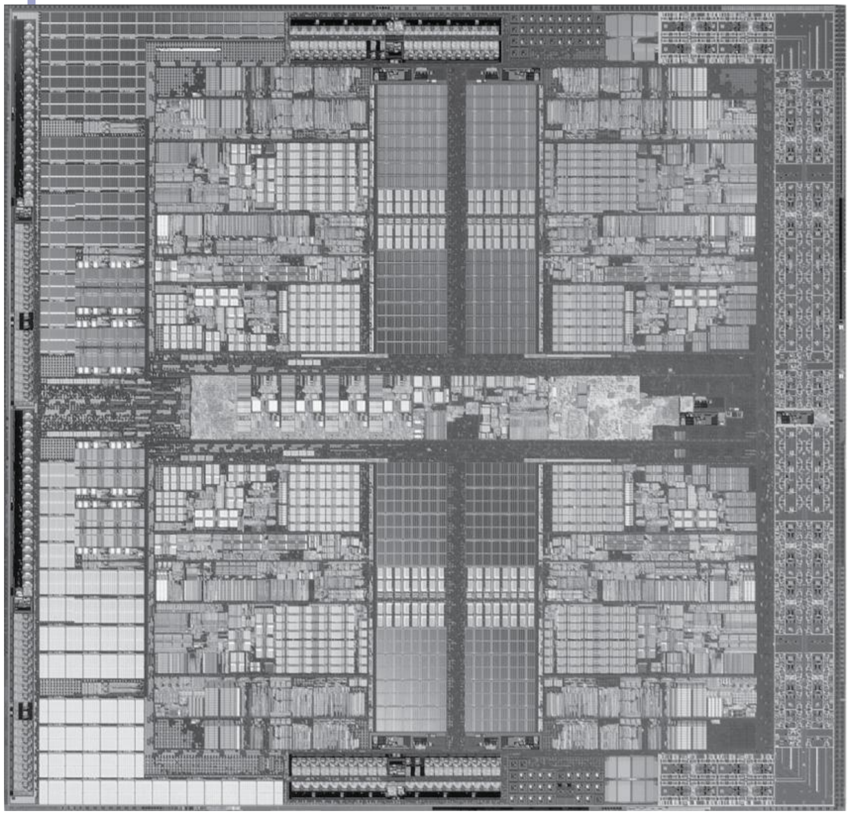
- Inputs/outputs include
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

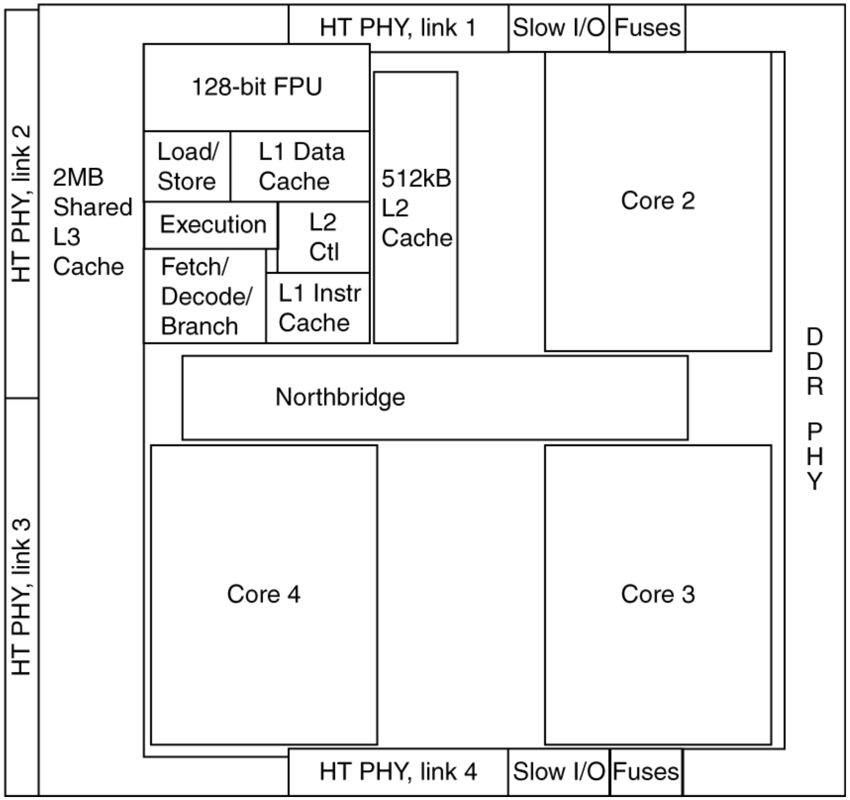
Inside the Processor (CPU)

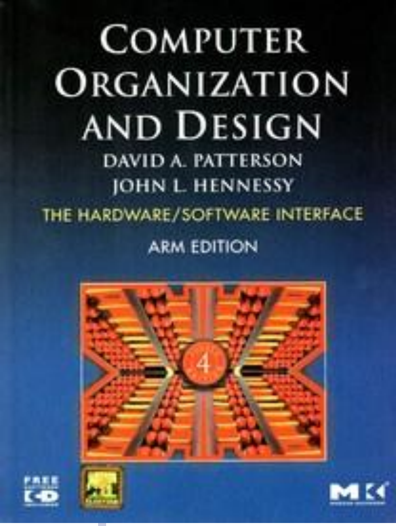
- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

Inside the Processor

- AMD Barcelona: 4 processor cores







CHAPTER 01

COMPUTER ABSTRACTIONS AND TECHNOLOGY

- Computer Abstractions
- Technology Trends
- The Power Wall & Multi-processors
- Below Your Program
- Under the Covers
- Performance

UNDERSTANDING PERFORMANCE

Understanding Performance

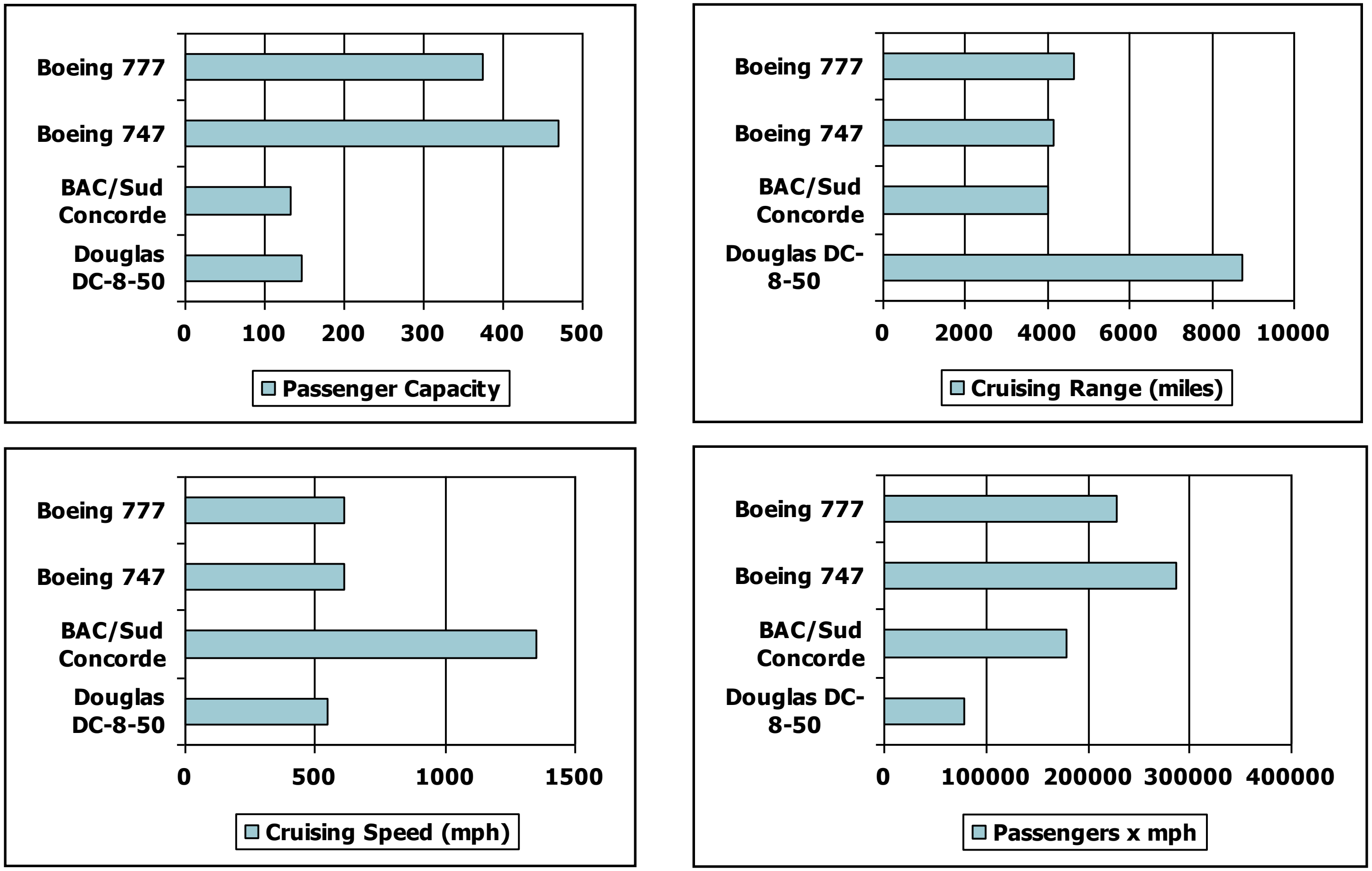
- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast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

Why Know Performance?

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key in understanding the underlying organizational motivation
  - Why is some hardware better than others for different programs?
  - What factors of system performance are hardware related?
    (e.g., Do we need a new machine, or a new operating system?)
  - How does the machine's instruction set affect performance?

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"

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

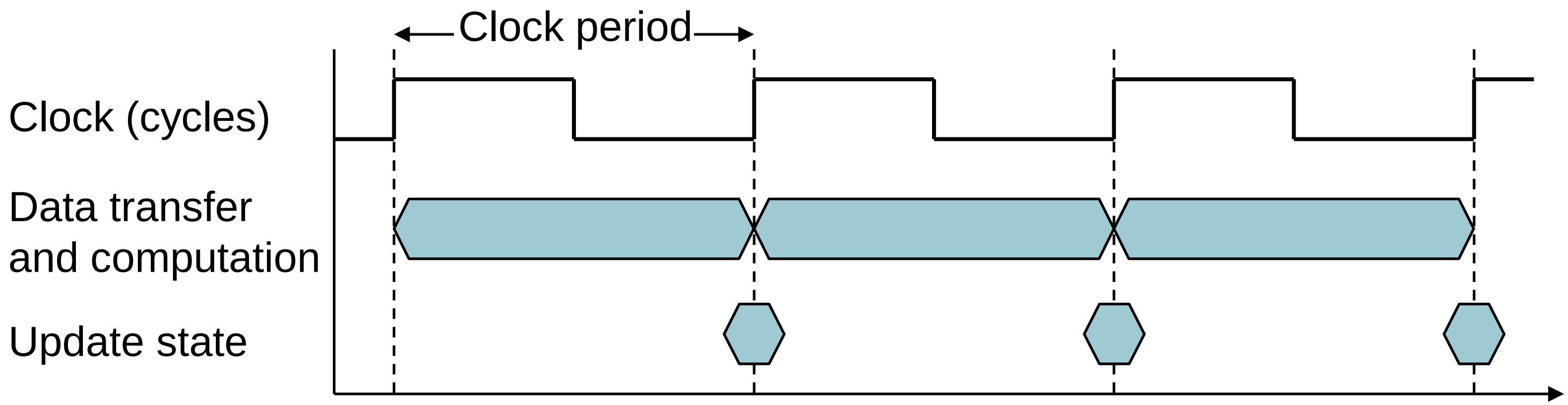
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

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

- A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must a computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

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?

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

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

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

Exercise

- The same set of instructions are being executed on two CPUs A and B. A is faster than B. The reason could be
  1. A is having a complex circuit
  2. The clock frequency of B is less than A
  3. The clock frequency of B is higher than A
  4. The clock rate of A is less than B

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

Using the Performance Equation

- Computers A and B implement the same ISA. Computer A has a clock cycle time of 250ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?

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?

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

A is faster…

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

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

…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

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

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
    $= 2\times1 + 1\times2 + 2\times3$
    $= 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    $= 4\times1 + 1\times2 + 1\times3$
    $= 9$
  - 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

Exercise

- The CPU time of a program **cannot** be given by
  - clock cycles / clock rate
  - (instructions/program) x (clock cycles/instruction) x (seconds/clock cycle)
  - instruction count * CPI * clock cycle time
  - instruction count * Average CPI / clock cycle time

Exercise

- A program runs on *computer1* with a 4GHz clock in 5.0 seconds. Calculate the clock rate of *computer2* that can finish this program in 2.5 seconds. Unfortunately, to accomplish this, *computer2* will require 1.5 times as many clock cycles as *computer1* to run the program.
  - 8 GHz
  - 12 GHz
  - The correct clock rate is not provided in answer a and b
  - Not enough data are given to calculate the clock rate

Exercise

- P_1, P_2 are two different hardware implementations of the same ISA. P_1 and P_2 have 4 GHz and 6 GHz clock rates respectively.

- If the peak performance is defined as the fastest rate that a computer can execute any instruction sequence, what are the peak performance of P_1 and P_2

Class	CPI on P1	CPI on P2
A	1	2
B	2	2
C	3	2
D	4	4
E	3	4

- If the instructions of a certain program P compiles into equally among the classes of instructions given, calculate how much faster it would be to execute this program in P2 than in P1.

Exercise

- Which of the following is **incorrect** about the performance computation of a CPU
  - If you have two CPUs with you, the best way to do a performance comparison is by measuring and comparing the execution time of appropriate applications on the CPUs.
  - The instructions of a CPU are divided into classes based on their CPI.
  - The CPI is calculated as an average number as individual instructions take a different number of clock cycles depending on which part of the program they are used.
  - A computer architect uses formulas and simulations to check the performance of a CPU as she does not have access to the CPUs they are building until they build them.

Pitfall: 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

Exercise

- According to Amdahl's law, the improvement in overall performance of a system can be given by
  - T(improved) = (T(unaffected) / improvement factor) + T (affected)
  - T(improved) = (T(affected) / improvement factor) + T (unaffected)
  - T(improved) = (T(affected) * improvement factor) + T (unaffected)
  - T(improved) = (T(affected) / improvement factor) - T (unaffected)

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