Computer Abstractions

Dr. Igor Ivkovic

iivkovic@uwaterloo.ca

[with material from “Computer Organization and Design” by Patterson and Hennessy, Published by Morgan Kaufmann]

Objectives

- How programs are translated into the machine language
- The hardware/software interface
- What determines program performance
- And how it can be improved
- How hardware designers improve performance
- What is parallel processing

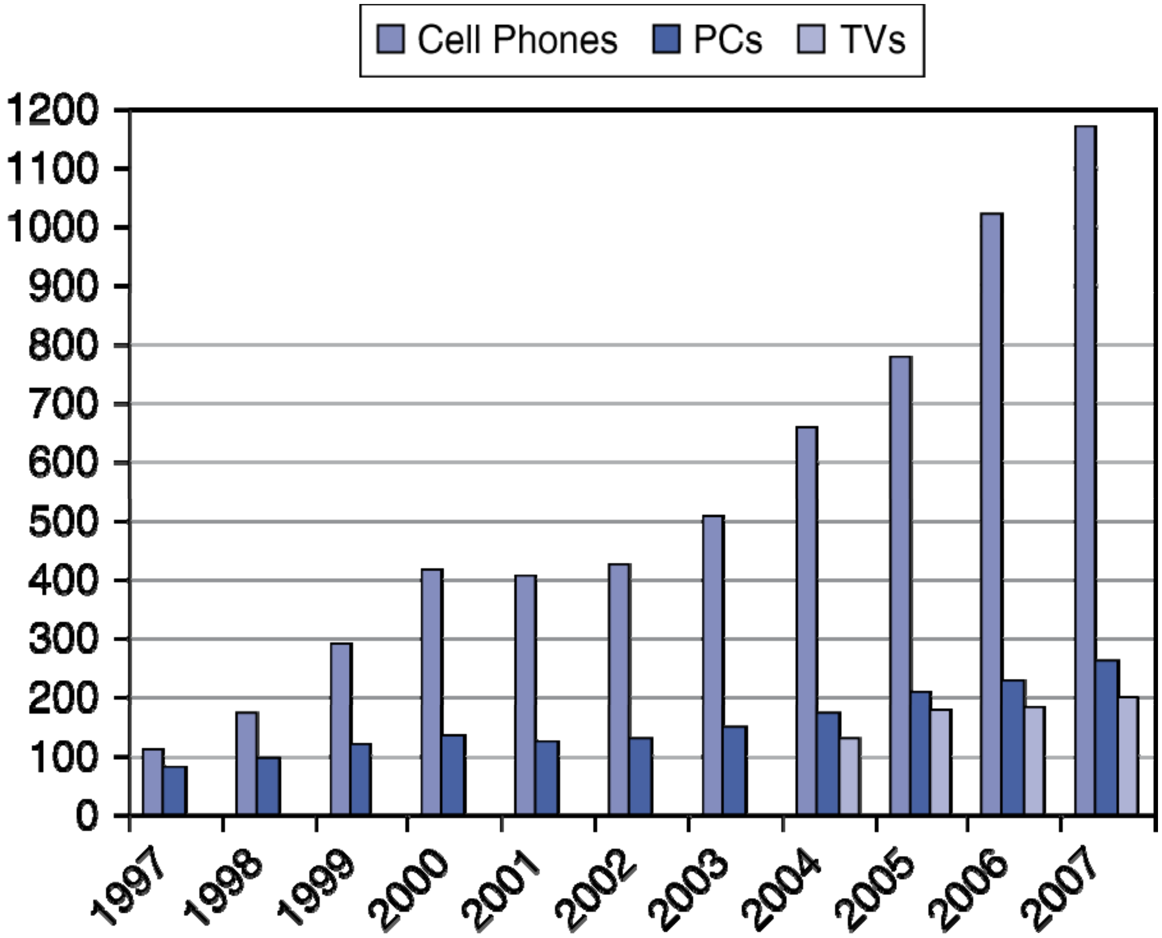
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
  - The number of transistors and ICs on a chip approximately doubles every two years (e.g., Intel's tri-gate 22nm transistor)
- **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

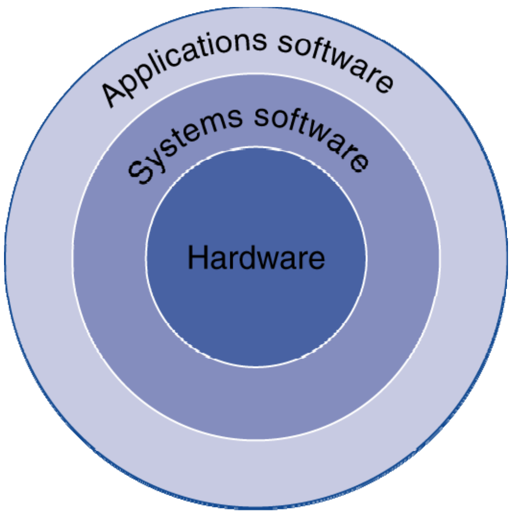
The Processor Market



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

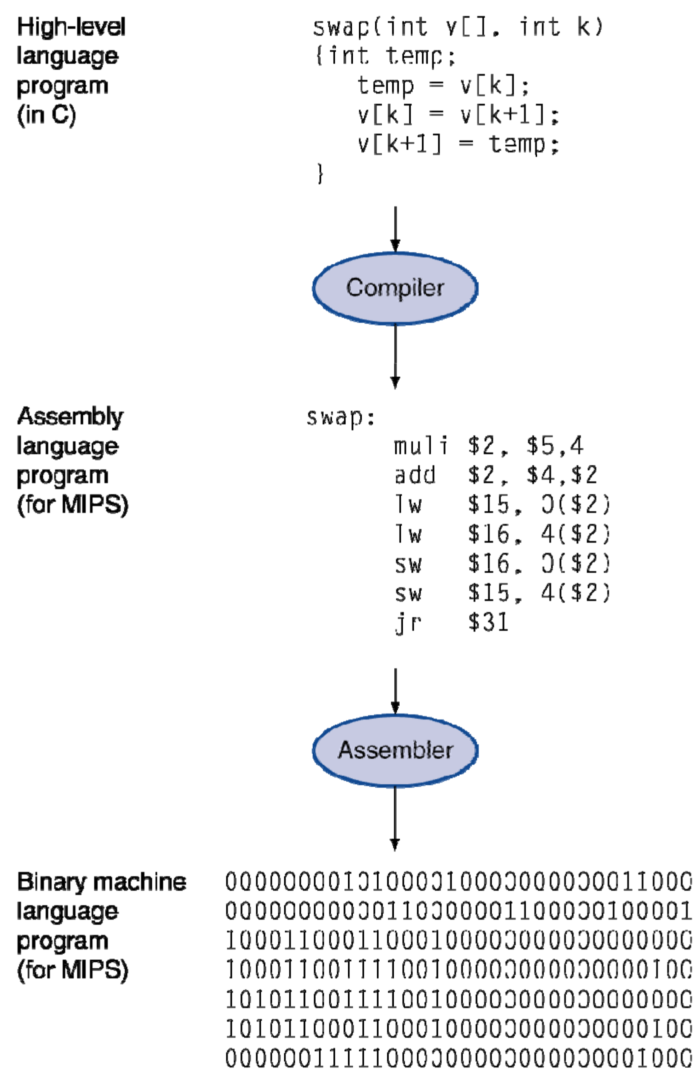
Behind the Curtain



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

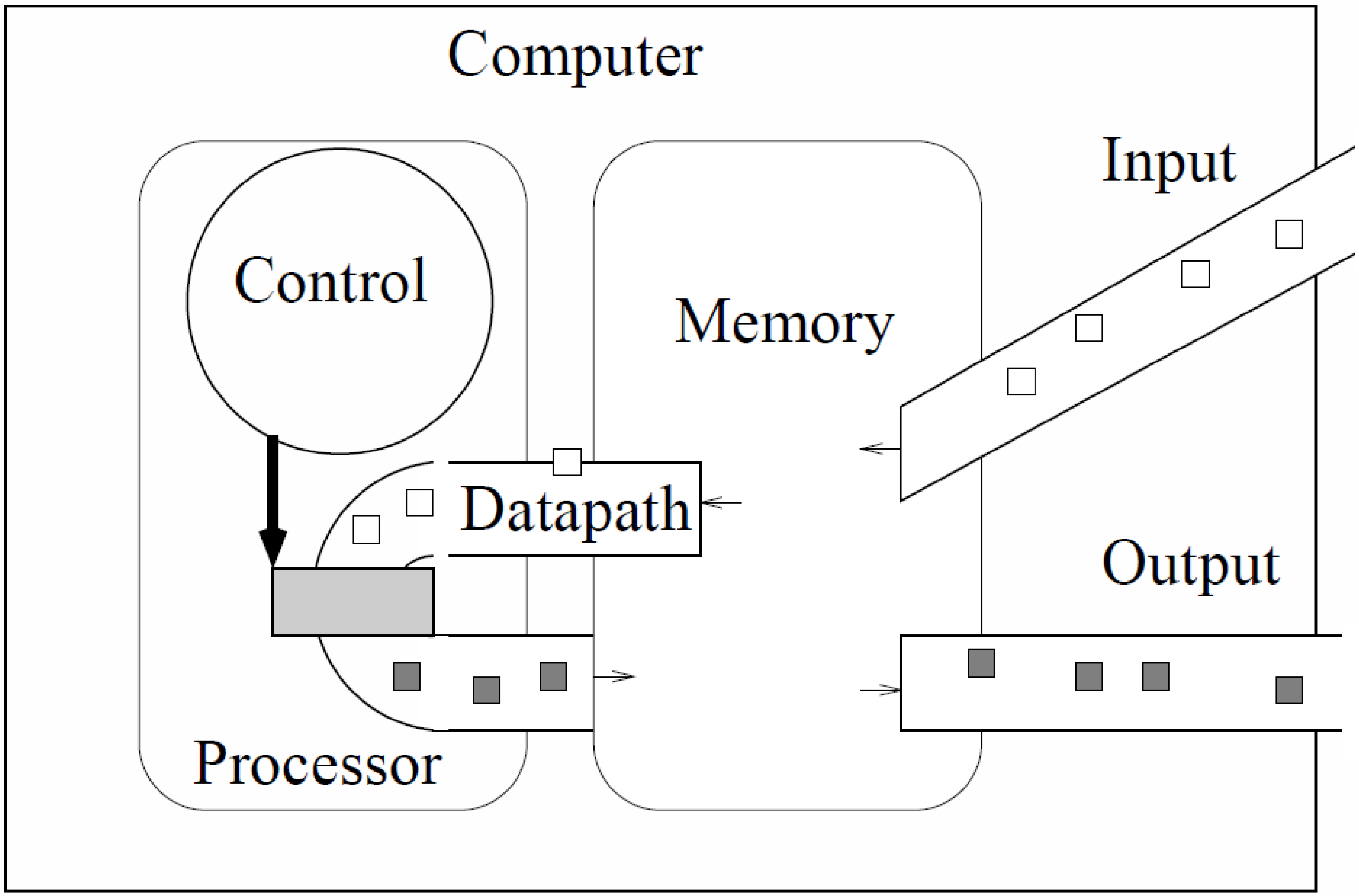
Levels of Program Code

- 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



Components of a Computer

- Same components for all kinds of computer
  - Desktop, server, embedded

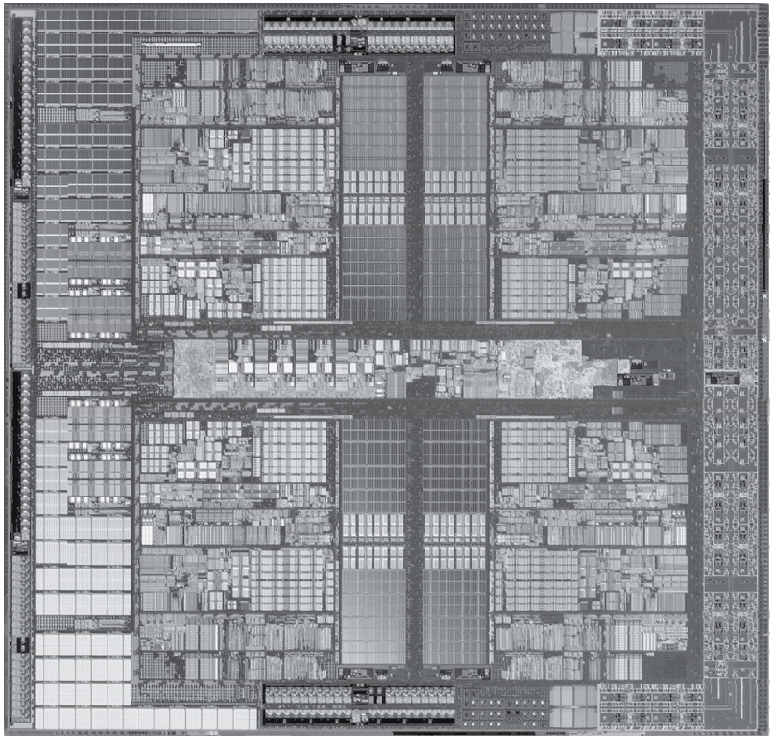


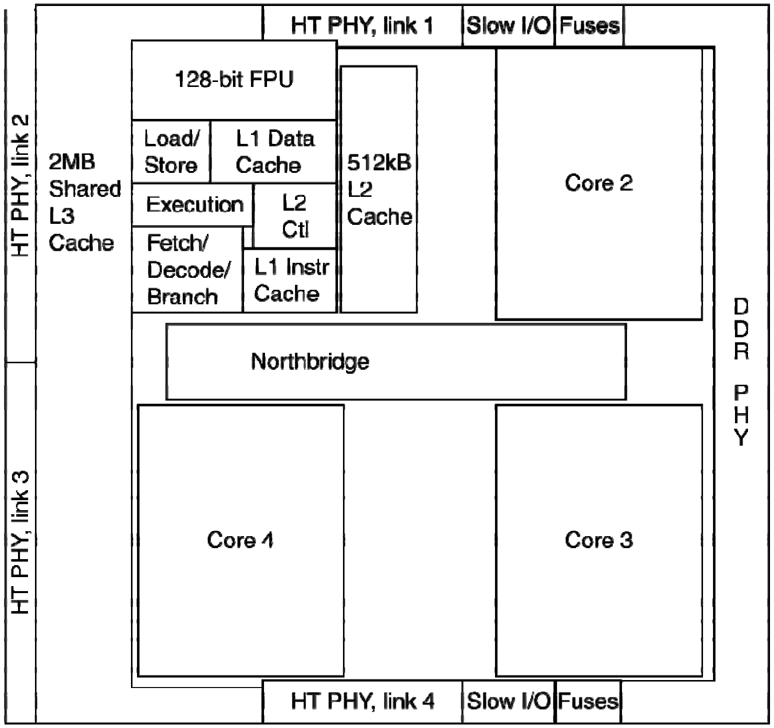
Inside the Processor (CPU)

- **Datapath:**
  - Performs operations on data
- **Control:**
  - Sequences datapath, memory, ...
- **Cache memory**
  - Small and fast static random-access memory (SRAM) for immediate access to data

Inside the Processor

- AMD Barcelona: 4 processor cores



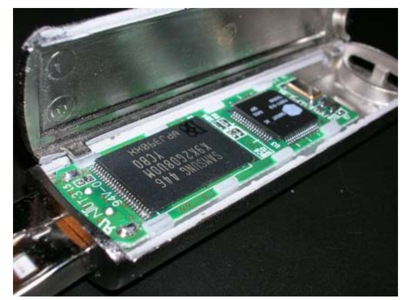


Abstractions

- **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

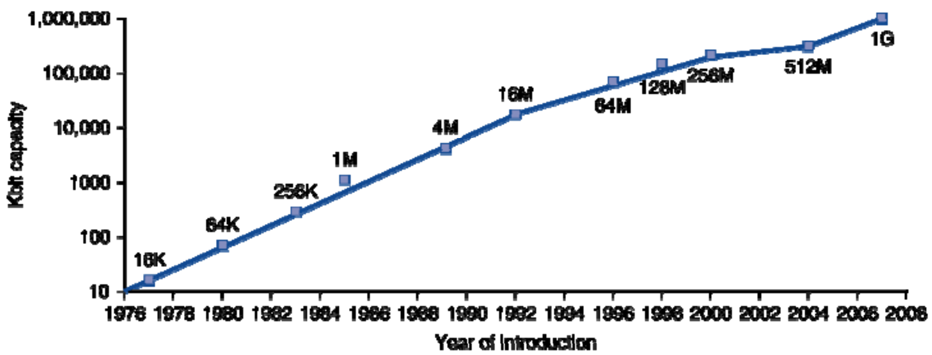
A Safe Place for Data

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



Technology Trends

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

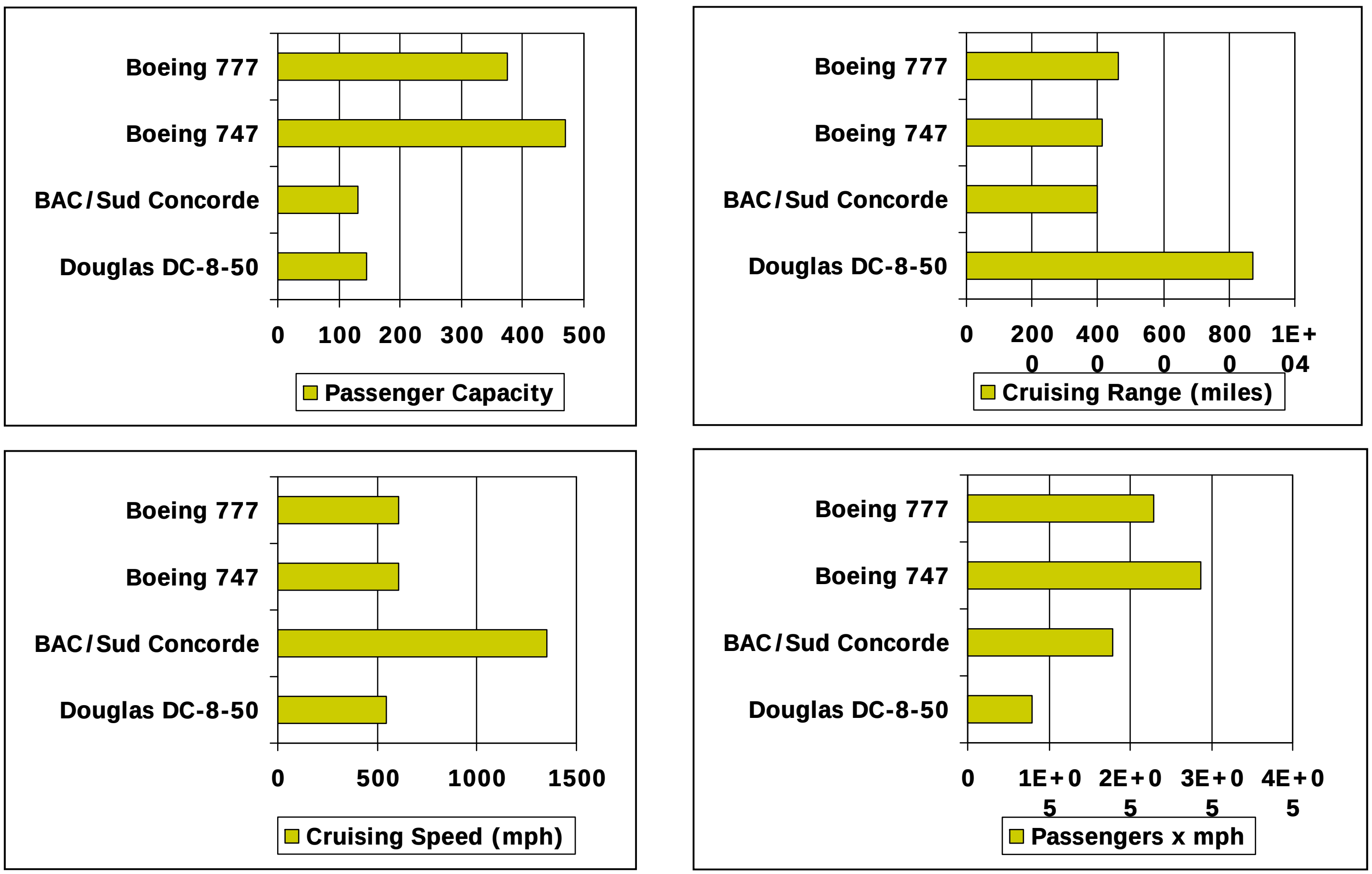


DRAM capacity

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 or transactions per second
- 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 times faster than Y"

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

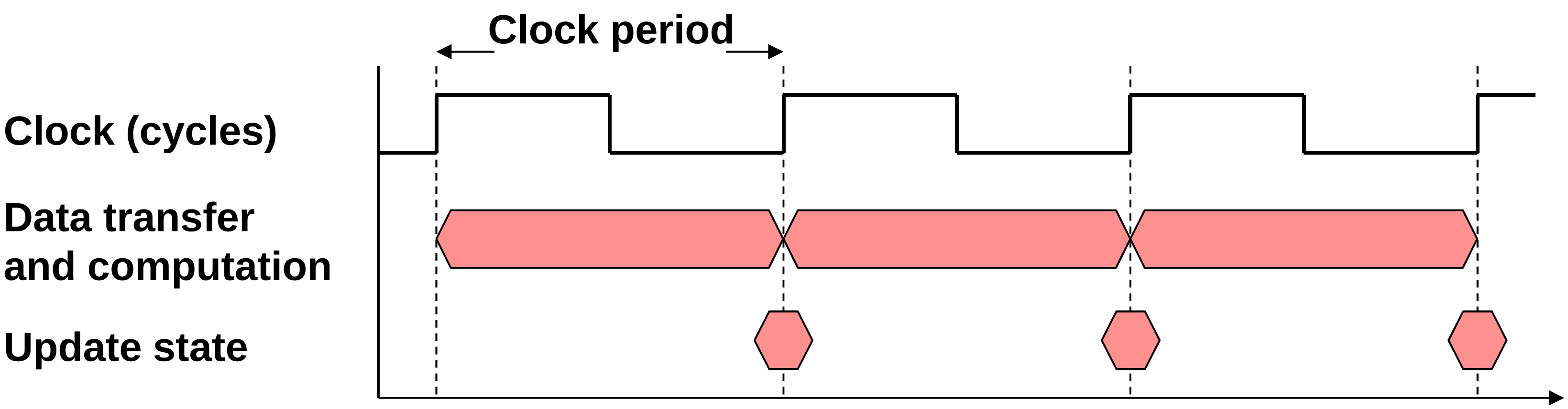
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

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
  - **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

$$\begin{aligned}\text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

- **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 × clock cycles
- How fast must Computer B clock be?

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

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

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

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

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

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

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

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

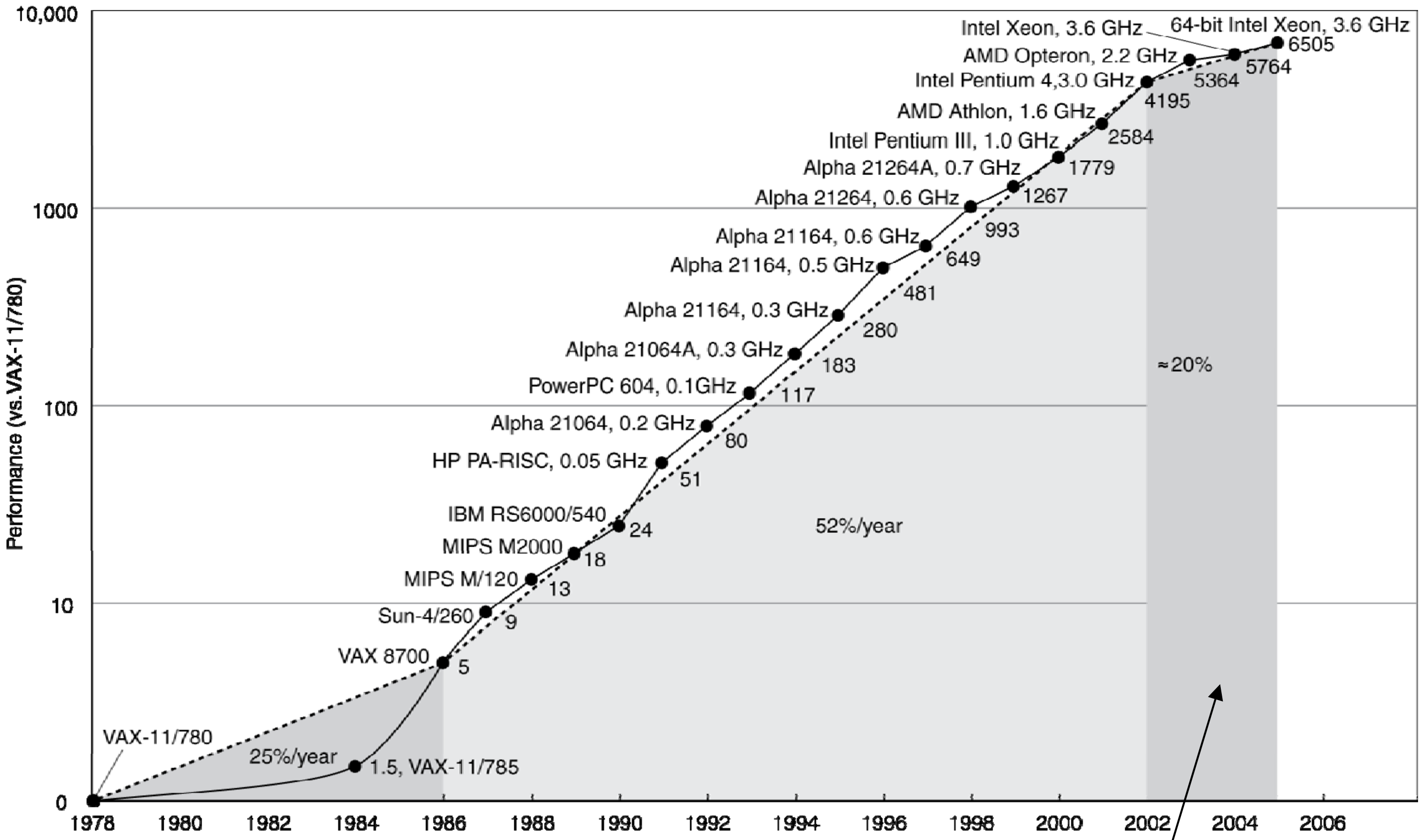
- Sequence 1: IC = 5
  - Clock Cycles
    $= 2\times1 + 1\times2 + 2\times3$
    $= 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    $= 4\times1 + 1\times2 + 1\times3$
    $= 9$
  - Avg. CPI = 9/6 = 1.5

Performance Summary

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

- 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

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

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

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

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

CPI varies between programs on a given CPU

Food for Thought

- Download and Read Assignment #0 Specifications
  - **Assignment #0 is intended as a simple introduction to the MIPS assembly language that we will use later in the term**
  - This assignment is optional, but if you complete it you can earn up to 1% bonus mark (the bonus will not apply if your final course grade is 100%)
- **Read:**
  - Chapter 1 of the course textbook
    - Review the material discussed in the lecture notes in more detail
    - Complete at least a few exercises from each chapter