1.1 Introduction

Welcome to this book! We're delighted to have this opportunity to convey the excitement of the world of computer systems. This is not a dry and dreary field, where progress is glacial and where new ideas atrophy from neglect. No! Computers are the product of the incredibly vibrant information technology industry, all aspects of which are responsible for almost 10% of the gross national product of the United States, and whose economy has become dependent in part on the rapid improvements in information technology promised by Moore's law. This unusual industry embraces innovation at a breathtaking rate. In the last 25 years, there have been a number of new computers whose introduction appeared to revolutionize the computing industry; these revolutions were cut short only because someone else built an even better computer.

This race to innovate has led to unprecedented progress since the inception of electronic computing in the late 1940s. Had the transportation industry kept pace with the computer industry, for example, today we could travel from New York to London in about a second for roughly a few cents. Take just a moment to contemplate how such an improvement would change society—living in Tahiti while working in San Francisco, going to Moscow for an evening at the Bolshoi Ballet—and you can appreciate the implications of such a change.

Computers have led to a third revolution for civilization, with the information revolution taking its place alongside the agricultural and the industrial revolutions. The resulting multiplication of humankind's intellectual strength and reach naturally has affected our everyday lives profoundly and changed the ways in which the search for new knowledge is carried out. There is now a new vein of scientific investigation, with computational scientists joining theoretical and experimental scientists in the exploration of new frontiers in astronomy, biology, chemistry, and physics, among others.

The computer revolution continues. Each time the cost of computing improves by another factor of 10, the opportunities for computers multiply. Applications that were economically infeasible suddenly become practical. In the recent past, the following applications were "computer science fiction."

- *Computers in automobiles:* Until microprocessors improved dramatically in price and performance in the early 1980s, computer control of cars was ludicrous. Today, computers reduce pollution, improve fuel efficiency via engine controls, and increase safety through the prevention of dangerous skids and through the inflation of air bags to protect occupants in a crash.
- *Cell phones:* Who would have dreamed that advances in computer systems would lead to mobile phones, allowing person-to-person communication almost anywhere in the world?
- *Human genome project:* The cost of computer equipment to map and analyze human DNA sequences is hundreds of millions of dollars. It's unlikely that anyone would have considered this project had the computer costs been 10 to 100 times higher, as they would have been 10 to 20 years ago. Moreover, costs continue to drop; you may be able to acquire your own genome, allowing medical care to be tailored to you.
- *World Wide Web:* Not in existence at the time of the first edition of this book, the World Wide Web has transformed our society. For many, the WWW has replaced libraries.
- *Search engines:* As the content of the WWW grew in size and in value, finding relevant information became increasingly important. Today, many people rely on search engines for such a large part of their lives that it would be a hardship to go without them.

Clearly, advances in this technology now affect almost every aspect of our society. Hardware advances have allowed programmers to create wonderfully useful software, which explains why computers are omnipresent. Today's science fiction suggests tomorrow's killer applications: already on their way are virtual worlds, practical speech recognition, and personalized health care.

Classes of Computing Applications and Their Characteristics

Although a common set of hardware technologies (see Sections 1.3 and 1.7) is used in computers ranging from smart home appliances to cell phones to the largest supercomputers, these different applications have different design requirements and employ the core hardware technologies in different ways. Broadly speaking, computers are used in three different classes of applications.

Desktop computers are possibly the best-known form of computing and are characterized by the personal computer, which readers of this book have likely used extensively. Desktop computers emphasize delivery of good performance to single users at low cost and usually execute third-party software. The evolution of many computing technologies is driven by this class of computing, which is only about 30 years old!

Servers are the modern form of what were once mainframes, minicomputers, and supercomputers, and are usually accessed only via a network. Servers are oriented to carrying large workloads, which may consist of either single complex applications—usually a scientific or engineering application—or handling many small jobs, such as would occur in building a large Web server. These applications are usually based on software from another source (such as a database or simulation system), but are often modified or customized for a particular function. Servers are built from the same basic technology as desktop computers, but provide for greater expandability of both computing and input/output capacity. In general, servers also place a greater emphasis on dependability, since a crash is usually more costly than it would be on a single-user desktop computer.

Servers span the widest range in cost and capability. At the low end, a server may be little more than a desktop computer without a screen or keyboard and cost a thousand dollars. These low-end servers are typically used for file storage, small business applications, or simple Web serving (see Section 6.10). At the other extreme are **supercomputers**, which at the present consist of hundreds to thousands of processors and usually **terabytes** of memory and **petabytes** of storage, and cost millions to hundreds of millions of dollars. Supercomputers are usually used for high-end scientific and engineering calculations, such as weather forecasting, oil exploration, protein structure determination, and other large-scale problems. Although such supercomputers represent the peak of computing capability, they represent a relatively small fraction of the servers and a relatively small fraction of the overall computer market in terms of total revenue.

Although not called supercomputers, Internet **datacenters** used by companies like eBay and Google also contain thousands of processors, terabytes of memory, and petabytes of storage. These are usually considered as large clusters of computers (see Chapter 7).

Embedded computers are the largest class of computers and span the widest range of applications and performance. Embedded computers include the

desktop computer A computer designed for use by an individual, usually incorporating a graphics display, a keyboard, and a mouse.

server A computer used for running larger programs for multiple users, often simultaneously, and typically accessed only via a network.

supercomputer A class of computers with the highest performance and cost; they are configured as servers and typically cost millions of dollars.

terabyte Originally 1,099,511,627,776 (2^{40}) bytes, although some communications and secondary storage systems have redefined it to mean 1,000,000,000,000 (10^{12}) bytes.

petabyte Depending on the situation, either 1000 or 1024 terabytes.

datacenter A room or building designed to handle the power, cooling, and networking needs of a large number of servers.

embedded computer A computer inside another device used for running one predetermined application or collection of software.

microprocessors found in your car, the computers in a cell phone, the computers in a video game or television, and the networks of processors that control a modern airplane or cargo ship. Embedded computing systems are designed to run one application or one set of related applications, that are normally integrated with the hardware and delivered as a single system; thus, despite the large number of embedded computers, most users never really see that they are using a computer!

Figure 1.1 shows that during the last several years, the growth in cell phones that rely on embedded computers has been much faster than the growth rate of desktop computers. Note that the embedded computers are also found in digital TVs and set-top boxes, automobiles, digital cameras, music players, video games, and a variety of other such consumer devices, which further increases the gap between the number of embedded computers and desktop computers.

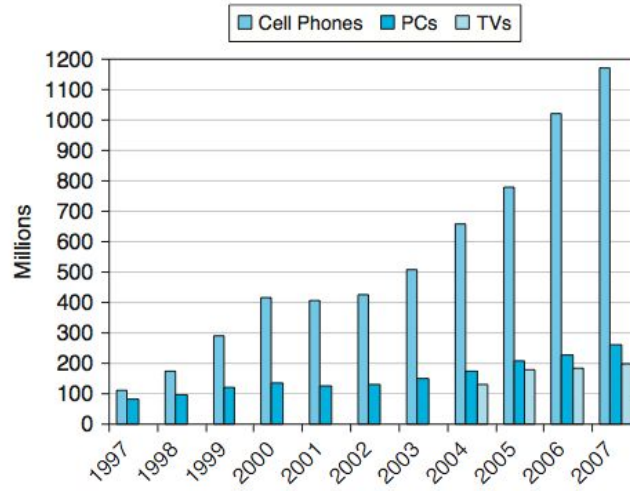

FIGURE 1.1 The number of cell phones, personal computers, and televisions manufactured per year between 1997 and 2007. (We have television data only from 2004.) More than a billion new cell phones were shipped in 2006. Cell phones sales exceeded PCs by only a factor of 1.4 in 1997, but the ratio grew to 4.5 in 2007. The total number in use in 2004 is estimated to be about 2.0B televisions, 1.8B cell phones, and 0.8B PCs. As the world population was about 6.4B in 2004, there were approximately one PC, 2.2 cell phones, and 2.5 televisions for every eight people on the planet. A 2006 survey of U.S. families found that they owned on average 12 gadgets, including three TVs, 2 PCs, and other devices such as game consoles, MP3 players, and cell phones.

Embedded applications often have unique application requirements that combine a minimum performance with stringent limitations on cost or power. For example, consider a music player: the processor need only be as fast as necessary to handle its limited function, and beyond that, minimizing cost and power are the most important objectives. Despite their low cost, embedded computers often have lower tolerance for failure, since the results can vary from upsetting (when your new television crashes) to devastating (such as might occur when the computer in a plane or cargo ship crashes). In consumer-oriented embedded applications, such as a digital home appliance, dependability is achieved primarily through simplicity—the emphasis is on doing one function as perfectly as possible. In large embedded systems, techniques of redundancy from the server world are often employed (see Section 6.9). Although this book focuses on general-purpose computers, most concepts apply directly, or with slight modifications, to embedded computers.

Elaboration: Elaborations are short sections used throughout the text to provide more detail on a particular subject that may be of interest. Disinterested readers may skip over an elaboration, since the subsequent material will never depend on the contents of the elaboration.

Many embedded processors are designed using *processor cores*, a version of a processor written in a hardware description language, such as Verilog or VHDL (see Chapter 4). The core allows a designer to integrate other application-specific hardware with the processor core for fabrication on a single chip.

What You Can Learn in This Book

Successful programmers have always been concerned about the performance of their programs, because getting results to the user quickly is critical in creating successful software. In the 1960s and 1970s, a primary constraint on computer performance was the size of the computer's memory. Thus, programmers often followed a simple credo: minimize memory space to make programs fast. In the last decade, advances in computer design and memory technology have greatly reduced the importance of small memory size in most applications other than those in embedded computing systems.

Programmers interested in performance now need to understand the issues that have replaced the simple memory model of the 1960s: the parallel nature of processors and the hierarchical nature of memories. Programmers who seek to build competitive versions of compilers, operating systems, databases, and even applications will therefore need to increase their knowledge of computer organization.

We are honored to have the opportunity to explain what's inside this revolutionary machine, unraveling the software below your program and the hardware under the covers of your computer. By the time you complete this book, we believe you will be able to answer the following questions:

- How are programs written in a high-level language, such as C or Java, translated into the language of the hardware, and how does the hardware execute the resulting program? Comprehending these concepts forms the basis of understanding the aspects of both the hardware and software that affect program performance.
- What is the interface between the software and the hardware, and how does software instruct the hardware to perform needed functions? These concepts are vital to understanding how to write many kinds of software.
- What determines the performance of a program, and how can a programmer improve the performance? As we will see, this depends on the original program, the software translation of that program into the computer's language, and the effectiveness of the hardware in executing the program.
- What techniques can be used by hardware designers to improve performance? This book will introduce the basic concepts of modern computer design. The interested reader will find much more material on this topic in our advanced book, *Computer Architecture: A Quantitative Approach.*
- What are the reasons for and the consequences of the recent switch from sequential processing to parallel processing? This book gives the motivation, describes the current hardware mechanisms to support parallelism, and surveys the new generation of **"multicore" microprocessors** (see Chapter 7).

multicore microprocessor A microprocessor containing multiple processors ("cores") in a single integrated circuit.

Without understanding the answers to these questions, improving the performance of your program on a modern computer, or evaluating what features might make one computer better than another for a particular application, will be a complex process of trial and error, rather than a scientific procedure driven by insight and analysis.

This first chapter lays the foundation for the rest of the book. It introduces the basic ideas and definitions, places the major components of software and hardware in perspective, shows how to evaluate performance and power, introduces integrated circuits (the technology that fuels the computer revolution), and explains the shift to multicores.

In this chapter and later ones, you will likely see many new words, or words that you may have heard but are not sure what they mean. Don't panic! Yes, there is a lot of special terminology used in describing modern computers, but the terminology actually helps, since it enables us to describe precisely a function or capability. In addition, computer designers (including your authors) *love* using **acronyms**, which are *easy* to understand once you know what the letters stand for! To help you remember and locate terms, we have included a **highlighted** definition of every term in the margins the first time it appears in the text. After a short time of working with the terminology, you will be fluent, and your friends will be impressed as you correctly use acronyms such as BIOS, CPU, DIMM, DRAM, PCIE, SATA, and many others.

acronym A word constructed by taking the initial letters of a string of words. For example: **RAM** is an acronym for Random Access Memory, and **CPU** is an acronym for Central Processing Unit.

To reinforce how the software and hardware systems used to run a program will affect performance, we use a special section, *Understanding Program Performance*, throughout the book to summarize important insights into program performance. The first one appears below.

Understanding Program Performance

The performance of a program depends on a combination of the effectiveness of the algorithms used in the program, the software systems used to create and translate the program into machine instructions, and the effectiveness of the computer in executing those instructions, which may include input/output (I/O) operations. This table summarizes how the hardware and software affect performance.

Hardware or software component	How this component affects performance	Where is this topic covered?
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed	Other books!
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement	Chapters 2 and 3
Processor and memory system	Determines how fast instructions can be executed	Chapters 4, 5, and 7
I/O system (hardware and operating system)	Determines how fast I/O operations may be executed	Chapter 6

Check Yourself

Check Yourself sections are designed to help readers assess whether they comprehend the major concepts introduced in a chapter and understand the implications of those concepts. Some *Check Yourself* questions have simple answers; others are for discussion among a group. Answers to the specific questions can be found at the end of the chapter. *Check Yourself* questions appear only at the end of a section, making it easy to skip them if you are sure you understand the material.

1. Section 1.1 showed that the number of embedded processors sold every year greatly outnumbers the number of desktop processors. Can you confirm or deny this insight based on your own experience? Try to count the number of embedded processors in your home. How does it compare with the number of desktop computers in your home?
2. As mentioned earlier, both the software and hardware affect the performance of a program. Can you think of examples where each of the following is the right place to look for a performance bottleneck?
   - The algorithm chosen
   - The programming language or compiler
   - The operating system
   - The processor
   - The I/O system and devices