REVISED FOURTH EDITION

Computer Organization and Design

THE HARDWARE/SOFTWARE INTERFACE

David A. Patterson
University of California, Berkeley

John L. Hennessy
Stanford University

With contributions by
Perry Alexander
The University of Kansas

Peter J. Ashenden
Ashenden Designs Pty Ltd

Javier Bruguera
Universidade de Santiago de Compostela

Jichuan Chang
Hewlett-Packard

Matthew Farrens
University of California, Davis

David Kaeli
Northeastern University

Nicole Kaiyan
University of Adelaide

David Kirk
NVIDIA

James R. Larus
Microsoft Research

Jacob Leverich
Hewlett-Packard

Kevin Lim
Hewlett-Packard

John Nickolls
NVIDIA

John Oliver
Cal Poly, San Luis Obispo

Milos Prvulovic
Georgia Tech

Partha Ranganathan
Hewlett-Packard

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Morgan Kaufmann is an imprint of Elsevier

In Paris they simply stared when I spoke to them in French; I never did succeed in making those idiots understand their own language.

Mark Twain, *The Innocents Abroad*, 1869

1.2 Below Your Program

A typical application, such as a word processor or a large database system, may consist of millions of lines of code and rely on sophisticated software libraries that implement complex functions in support of the application. As we will see, the hardware in a computer can only execute extremely simple low-level instructions. To go from a complex application to the simple instructions involves several layers of software that interpret or translate high-level operations into simple computer instructions.

Figure 1.2 shows that these layers of software are organized primarily in a hierarchical fashion, with applications being the outermost ring and a variety of **systems software** sitting between the hardware and applications software.

systems software Software that provides services that are commonly useful, including operating systems, compilers, loaders, and assemblers.

There are many types of systems software, but two types of systems software are central to every computer system today: an operating system and a compiler. An **operating system** interfaces between a user's program and the hardware and provides a variety of services and supervisory functions. Among the most important functions are

operating system Supervising program that manages the resources of a computer for the benefit of the programs that run on that computer.

- Handling basic input and output operations
- Allocating storage and memory
- Providing for protected sharing of the computer among multiple applications using it simultaneously.

Examples of operating systems in use today are Linux, MacOS, and Windows.

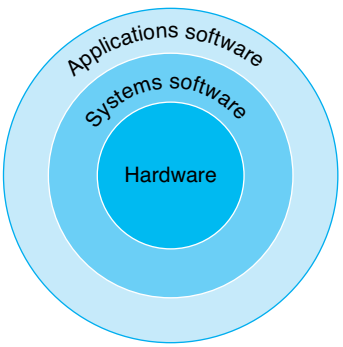

FIGURE 1.2 A simplified view of hardware and software as hierarchical layers, shown as concentric circles with hardware in the center and applications software outermost. In complex applications, there are often multiple layers of application software as well. For example, a database system may run on top of the systems software hosting an application, which in turn runs on top of the database.

Compilers perform another vital function: the translation of a program written in a high-level language, such as C, C++, Java, or Visual Basic into instructions that the hardware can execute. Given the sophistication of modern programming languages and the simplicity of the instructions executed by the hardware, the translation from a high-level language program to hardware instructions is complex. We give a brief overview of the process here and then go into more depth in Chapter 2 and Appendix B.

compiler A program that translates high-level language statements into assembly language statements.

From a High-Level Language to the Language of Hardware

To actually speak to electronic hardware, you need to send electrical signals. The easiest signals for computers to understand are *on* and *off*, and so the computer alphabet is just two letters. Just as the 26 letters of the English alphabet do not limit how much can be written, the two letters of the computer alphabet do not limit what computers can do. The two symbols for these two letters are the numbers 0 and 1, and we commonly think of the computer language as numbers in base 2, or *binary numbers.* We refer to each "letter" as a **binary digit** or **bit**. Computers are slaves to our commands, which are called **instructions**. Instructions, which are just collections of bits that the computer understands and obeys, can be thought of as numbers. For example, the bits

binary digit Also called a **bit**. One of the two numbers in base 2 (0 or 1) that are the components of information.

instruction A command that computer hardware understands and obeys.

```
1000110010100000
```

tell one computer to add two numbers. Chapter 2 explains why we use numbers for instructions *and* data; we don't want to steal that chapter's thunder, but using numbers for both instructions and data is a foundation of computing.

The first programmers communicated to computers in binary numbers, but this was so tedious that they quickly invented new notations that were closer to the way humans think. At first, these notations were translated to binary by hand, but this process was still tiresome. Using the computer to help program the computer, the pioneers invented programs to translate from symbolic notation to binary. The first of these programs was named an **assembler**. This program translates a symbolic version of an instruction into the binary version. For example, the programmer would write

assembler A program that translates a symbolic version of instructions into the binary version.

```
add A,B
```

and the assembler would translate this notation into

```
1000110010100000
```

This instruction tells the computer to add the two numbers `A` and `B`. The name coined for this symbolic language, still used today, is **assembly language**. In contrast, the binary language that the machine understands is the **machine language**.

assembly language A symbolic representation of machine instructions.

machine language A binary representation of machine instructions.

Although a tremendous improvement, assembly language is still far from the notations a scientist might like to use to simulate fluid flow or that an accountant might use to balance the books. Assembly language requires the programmer

to write one line for every instruction that the computer will follow, forcing the programmer to think like the computer.

The recognition that a program could be written to translate a more powerful language into computer instructions was one of the great breakthroughs in the early days of computing. Programmers today owe their productivity—and their sanity—to the creation of **high-level programming languages** and compilers that translate programs in such languages into instructions. Figure 1.3 shows the relationships among these programs and languages.

high-level programming language A portable language such as C, C++, Java, or Visual Basic that is composed of words and algebraic notation that can be translated by a compiler into assembly language.

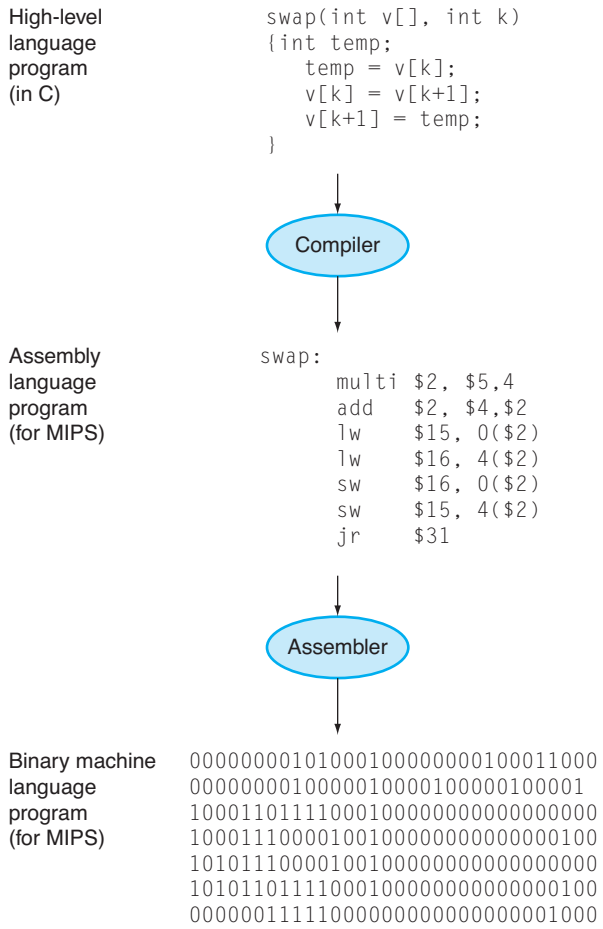

FIGURE 1.3 C program compiled into assembly language and then assembled into binary machine language. Although the translation from high-level language to binary machine language is shown in two steps, some compilers cut out the middleman and produce binary machine language directly. These languages and this program are examined in more detail in Chapter 2.

A compiler enables a programmer to write this high-level language expression:

```
A + B
```

The compiler would compile it into this assembly language statement:

```
add A,B
```

As shown above, the assembler would translate this statement into the binary instructions that tell the computer to add the two numbers `A` and `B`.

High-level programming languages offer several important benefits. First, they allow the programmer to think in a more natural language, using English words and algebraic notation, resulting in programs that look much more like text than like tables of cryptic symbols (see Figure 1.3). Moreover, they allow languages to be designed according to their intended use. Hence, Fortran was designed for scientific computation, Cobol for business data processing, Lisp for symbol manipulation, and so on. There are also domain-specific languages for even narrower groups of users, such as those interested in simulation of fluids, for example.

The second advantage of programming languages is improved programmer productivity. One of the few areas of widespread agreement in software development is that it takes less time to develop programs when they are written in languages that require fewer lines to express an idea. Conciseness is a clear advantage of high-level languages over assembly language.

The final advantage is that programming languages allow programs to be independent of the computer on which they were developed, since compilers and assemblers can translate high-level language programs to the binary instructions of any computer. These three advantages are so strong that today little programming is done in assembly language.

1.3 Under the Covers

Now that we have looked below your program to uncover the underlying software, let's open the covers of your computer to learn about the underlying hardware. The underlying hardware in any computer performs the same basic functions: inputting data, outputting data, processing data, and storing data. How these functions are performed is the primary topic of this book, and subsequent chapters deal with different parts of these four tasks.

When we come to an important point in this book, a point so important that we hope you will remember it forever, we emphasize it by identifying it as a *Big Picture* item. We have about a dozen Big Pictures in this book, the first being

the five components of a computer that perform the tasks of inputting, outputting, processing, and storing data.

The BIG Picture

The five classic components of a computer are input, output, memory, datapath, and control, with the last two sometimes combined and called the processor. Figure 1.4 shows the standard organization of a computer. This organization is independent of hardware technology: you can place every piece of every computer, past and present, into one of these five categories. To help you keep all this in perspective, the five components of a computer are shown on the front page of each of the following chapters, with the portion of interest to that chapter highlighted.

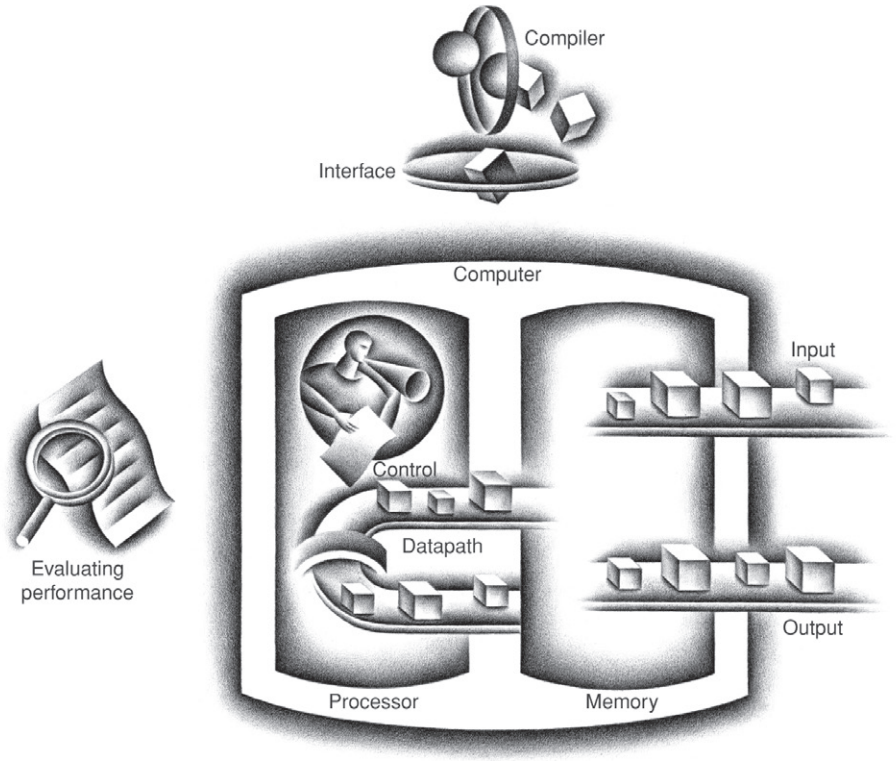

FIGURE 1.4 The organization of a computer, showing the five classic components. The processor gets instructions and data from memory. Input writes data to memory, and output reads data from memory. Control sends the signals that determine the operations of the datapath, memory, input, and output.

FIGURE 1.5 A desktop computer. The liquid crystal display (LCD) screen is the primary output device, and the keyboard and mouse are the primary input devices. On the right side is an Ethernet cable that connected the laptop to the network and the Web. The laptop contains the processor, memory, and additional I/O devices. This system is a Macbook Pro 15" laptop connected to an external display.

Figure 1.5 shows a computer with keyboard, wireless mouse, and screen. This photograph reveals two of the key components of computers: **input devices**, such as the keyboard and mouse, and **output devices**, such as the screen. As the names suggest, input feeds the computer, and output is the result of computation sent to the user. Some devices, such as networks and disks, provide both input and output to the computer.

Chapter 6 describes input/output (I/O) devices in more detail, but let's take an introductory tour through the computer hardware, starting with the external I/O devices.

input device
A mechanism through which the computer is fed information, such as the keyboard or mouse.

output device
A mechanism that conveys the result of a computation to a user or another computer.

I got the idea for the mouse while attending a talk at a computer conference. The speaker was so boring that I started daydreaming and hit upon the idea.

Doug Engelbart

Anatomy of a Mouse

Although many users now take mice for granted, the idea of a pointing device such as a mouse was first shown by Doug Engelbart using a research prototype in 1967. The Alto, which was the inspiration for all workstations as well as for the Macintosh and Windows OS, included a mouse as its pointing device in 1973. By the 1990s, all desktop computers included this device, and new user interfaces based on graphics displays and mice became the norm.

The original mouse was electromechanical and used a large ball that when rolled across a surface would cause an x and y counter to be incremented. The amount of increase in each counter told how far the mouse had been moved.

The electromechanical mouse has largely been replaced by the newer all-optical mouse. The optical mouse is actually a miniature optical processor including an LED to provide lighting, a tiny black-and-white camera, and a simple optical processor. The LED illuminates the surface underneath the mouse; the camera takes 1500 sample pictures a second under the illumination. Successive pictures are sent to a simple optical processor that compares the images and determines whether the mouse has moved and how far. The replacement of the electromechanical mouse by the electro-optical mouse is an illustration of a common phenomenon where the decreasing costs and higher reliability of electronics cause an electronic solution to replace the older electromechanical technology. On page 22 we'll see another example: flash memory.

Through computer displays I have landed an airplane on the deck of a moving carrier, observed a nuclear particle hit a potential well, flown in a rocket at nearly the speed of light and watched a computer reveal its innermost workings.

Ivan Sutherland, the "father" of computer graphics, *Scientific American*, 1984

Through the Looking Glass

The most fascinating I/O device is probably the graphics display. All laptop and handheld computers, calculators, cellular phones, and almost all desktop computers now use **liquid crystal displays (LCDs)** to get a thin, low-power display. The LCD is not the source of light; instead, it controls the transmission of light. A typical LCD includes rod-shaped molecules in a liquid that form a twisting helix that bends light entering the display, from either a light source behind the display or less often from reflected light. The rods straighten out when a current is applied and no longer bend the light. Since the liquid crystal material is between two screens polarized at 90 degrees, the light cannot pass through unless it is bent. Today, most LCD displays use an **active matrix** that has a tiny transistor switch at each pixel to precisely control current and make sharper images. A red-green-blue mask associated with each dot on the display determines the intensity of the three color components in the final image; in a color active matrix LCD, there are three transistor switches at each point.

The image is composed of a matrix of picture elements, or **pixels**, which can be represented as a matrix of bits, called a *bit map*. Depending on the size of the screen and the resolution, the display matrix ranges in size from 640×480 to 2560×1600 pixels in 2008. A color display might use 8 bits for each of the three colors (red, blue, and green), for 24 bits per pixel, permitting millions of different colors to be displayed.

liquid crystal display A display technology using a thin layer of liquid polymers that can be used to transmit or block light according to whether a charge is applied.

active matrix display A liquid crystal display using a transistor to control the transmission of light at each individual pixel.

pixel The smallest individual picture element. Screens are composed of hundreds of thousands to millions of pixels, organized in a matrix.

The computer hardware support for graphics consists mainly of a *raster refresh buffer*, or *frame buffer*, to store the bit map. The image to be represented onscreen is stored in the frame buffer, and the bit pattern per pixel is read out to the graphics display at the refresh rate. Figure 1.6 shows a frame buffer with a simplified design of just 4 bits per pixel.

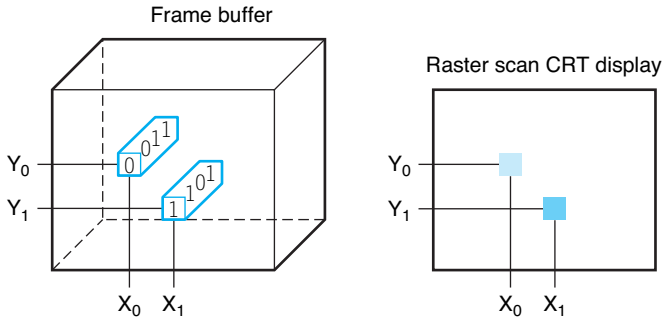

FIGURE 1.6 Each coordinate in the frame buffer on the left determines the shade of the corresponding coordinate for the raster scan CRT display on the right. Pixel (X_0, Y_0) contains the bit pattern 0011, which is a lighter shade on the screen than the bit pattern 1101 in pixel (X_1, Y_1).

The goal of the bit map is to faithfully represent what is on the screen. The challenges in graphics systems arise because the human eye is very good at detecting even subtle changes on the screen.

Opening the Box

If we open the box containing the computer, we see a fascinating board of thin plastic, covered with dozens of small gray or black rectangles. Figure 1.7 shows the contents of the laptop computer in Figure 1.5. The **motherboard** is shown in the upper part of the photo. Two disk drives are in front—the hard drive on the left and a DVD drive on the right. The hole in the middle is for the laptop battery.

The small rectangles on the motherboard contain the devices that drive our advancing technology, called **integrated circuits** and nicknamed **chips**. The board is composed of three pieces: the piece connecting to the I/O devices mentioned earlier, the memory, and the processor.

The **memory** is where the programs are kept when they are running; it also contains the data needed by the running programs. Figure 1.8 shows that memory is found on the two small boards, and each small memory board contains eight integrated circuits. The memory in Figure 1.8 is built from DRAM chips. *DRAM*

motherboard A plastic board containing packages of integrated circuits or chips, including processor, cache, memory, and connectors for I/O devices such as networks and disks.

integrated circuit Also called a **chip**. A device combining dozens to millions of transistors.

memory The storage area in which programs are kept when they are running and that contains the data needed by the running programs.

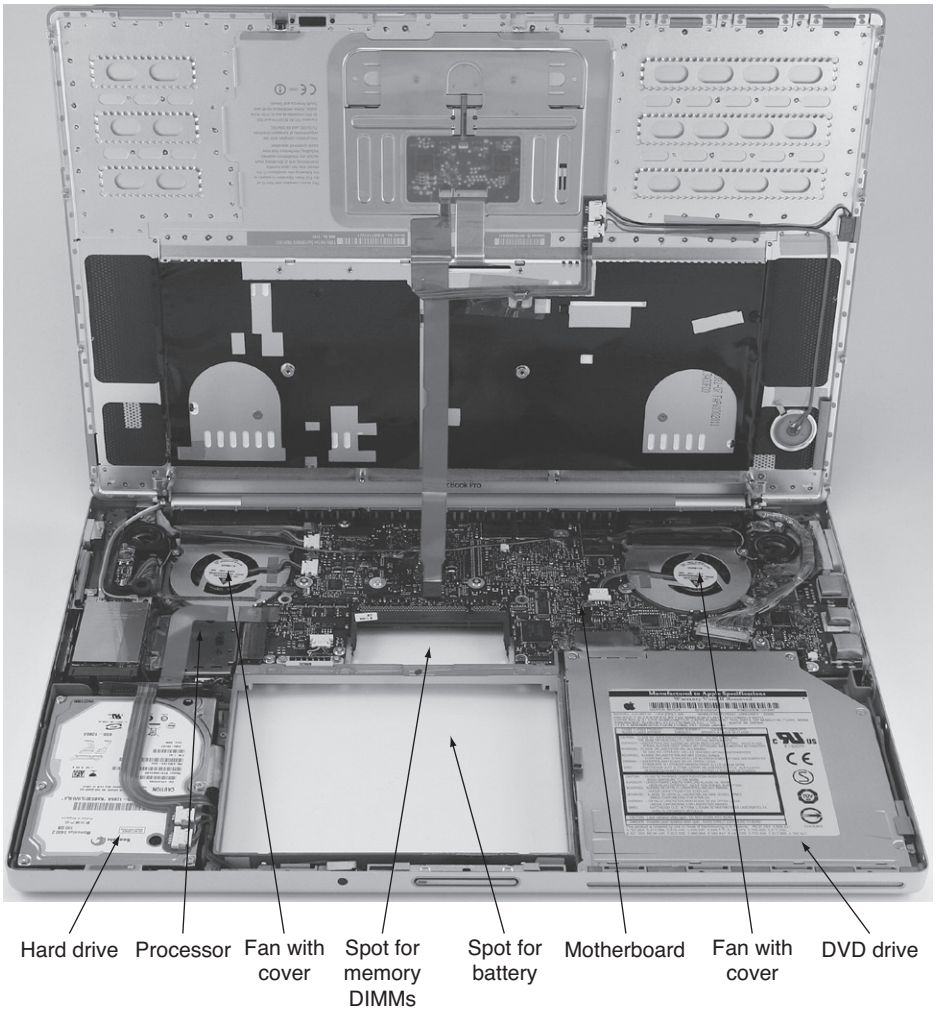

FIGURE 1.7 Inside the laptop computer of Figure 1.5. The shiny box with the white label on the lower left is a 100 GB SATA hard disk drive, and the shiny metal box on the lower right side is the DVD drive. The hole between them is where the laptop battery would be located. The small hole above the battery hole is for memory DIMMs. Figure 1.8 is a close-up of the DIMMs, which are inserted from the bottom in this laptop. Above the battery hole and DVD drive is a printed circuit board (PC board), called the *motherboard*, which contains most of the electronics of the computer. The two shiny circles in the upper half of the picture are two fans with covers. The processor is the large raised rectangle just below the left fan. Photo courtesy of OtherWorldComputing.com.

stands for **dynamic random access memory**. Several DRAMs are used together to contain the instructions and data of a program. In contrast to sequential access memories, such as magnetic tapes, the *RAM* portion of the term DRAM means that memory accesses take basically the same amount of time no matter what portion of the memory is read.

dynamic random access memory (DRAM) Memory built as an integrated circuit; it provides random access to any location.

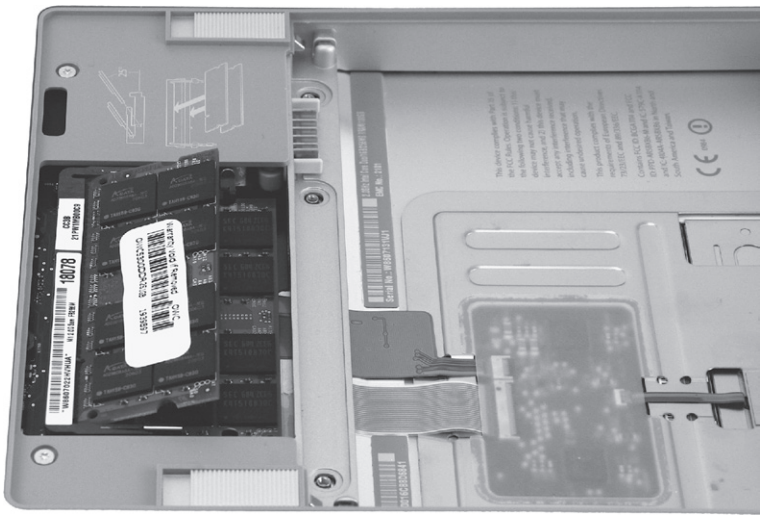

FIGURE 1.8 Close-up of the bottom of the laptop reveals the memory. The main memory is contained on one or more small boards shown on the left. The hole for the battery is to the right. The DRAM chips are mounted on these boards (called **DIMMs**, for dual inline memory modules) and then plugged into the connectors. Photo courtesy of OtherWorldComputing.com.

dual inline memory module (DIMM) A small board that contains DRAM chips on both sides. (SIMMs have DRAMs on only one side.)

central processor unit (CPU) Also called processor. The active part of the computer, which contains the datapath and control and which adds numbers, tests numbers, signals I/O devices to activate, and so on.

datapath The component of the processor that performs arithmetic operations

control The component of the processor that commands the datapath, memory, and I/O devices according to the instructions of the program.

The *processor* is the active part of the board, following the instructions of a program to the letter. It adds numbers, tests numbers, signals I/O devices to activate, and so on. The processor is under the fan and covered by a heat sink on the left side of Figure 1.7. Occasionally, people call the processor the **CPU**, for the more bureaucratic-sounding **central processor unit**.

Descending even lower into the hardware, Figure 1.9 reveals details of a microprocessor. The processor logically comprises two main components: datapath and control, the respective brawn and brain of the processor. The **datapath** performs the arithmetic operations, and **control** tells the datapath, memory, and I/O devices what to do according to the wishes of the instructions of the program. Chapter 4 explains the datapath and control for a higher-performance design.

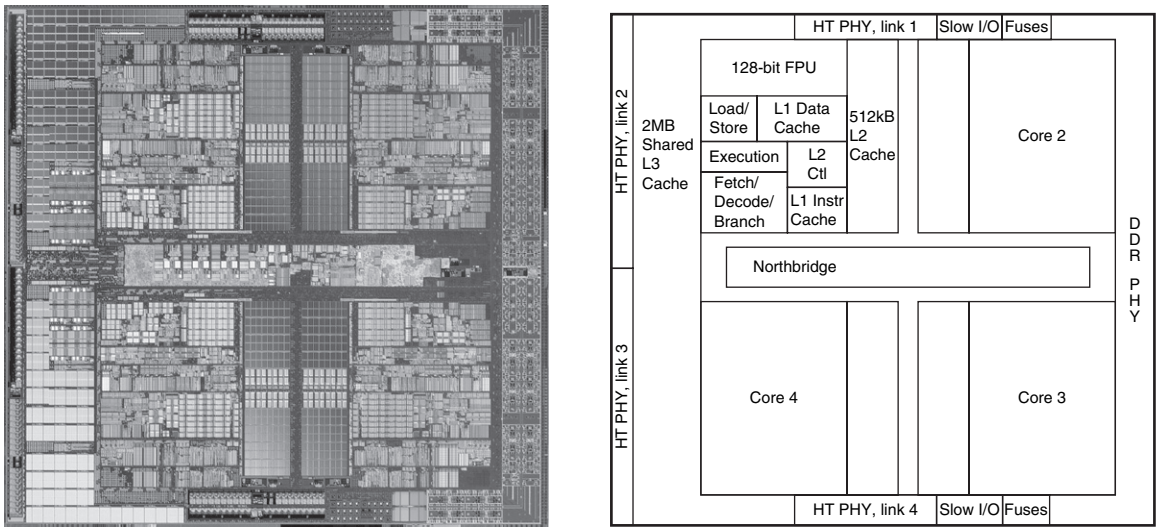

FIGURE 1.9 Inside the AMD Barcelona microprocessor. The left-hand side is a microphotograph of the AMD Barcelona processor chip, and the right-hand side shows the major blocks in the processor. This chip has four processors or "cores". The microprocessor in the laptop in Figure 1.7 has two cores per chip, called an Intel Core 2 Duo.

cache memory A small, fast memory that acts as a buffer for a slower, larger memory.

static random access memory (SRAM) Also memory built as an integrated circuit, but faster and less dense than DRAM.

abstraction A model that renders lower-level details of computer systems temporarily invisible to facilitate design of sophisticated systems.

Descending into the depths of any component of the hardware reveals insights into the computer. Inside the processor is another type of memory—cache memory. **Cache memory** consists of a small, fast memory that acts as a buffer for the DRAM memory. (The nontechnical definition of *cache* is a safe place for hiding things.) Cache is built using a different memory technology, **static random access memory** (**SRAM**). SRAM is faster but less dense, and hence more expensive, than DRAM (see Chapter 5).

You may have noticed a common theme in both the software and the hardware descriptions: delving into the depths of hardware or software reveals more information or, conversely, lower-level details are hidden to offer a simpler model at higher levels. The use of such layers, or **abstractions**, is a principal technique for designing very sophisticated computer systems.

One of the most important abstractions is the interface between the hardware and the lowest-level software. Because of its importance, it is given a special

name: the **instruction set architecture**, or simply **architecture**, of a computer. The instruction set architecture includes anything programmers need to know to make a binary machine language program work correctly, including instructions, I/O devices, and so on. Typically, the operating system will encapsulate the details of doing I/O, allocating memory, and other low-level system functions so that application programmers do not need to worry about such details. The combination of the basic instruction set and the operating system interface provided for application programmers is called the **application binary interface** (**ABI**).

An instruction set architecture allows computer designers to talk about functions independently from the hardware that performs them. For example, we can talk about the functions of a digital clock (keeping time, displaying the time, setting the alarm) independently from the clock hardware (quartz crystal, LED displays, plastic buttons). Computer designers distinguish architecture from an **implementation** of an architecture along the same lines: an implementation is hardware that obeys the architecture abstraction. These ideas bring us to another Big Picture.

instruction set architecture Also called **architecture**. An abstract interface between the hardware and the lowest-level software that encompasses all the information necessary to write a machine language program that will run correctly, including instructions, registers, memory access, I/O, ...

application binary interface (ABI) The user portion of the instruction set plus the operating system interfaces used by application programmers. Defines a standard for binary portability across computers.

implementation Hardware that obeys the architecture abstraction.

The BIG Picture

Both hardware and software consist of hierarchical layers, with each lower layer hiding details from the level above. This principle of *abstraction* is the way both hardware designers and software designers cope with the complexity of computer systems. One key interface between the levels of abstraction is the *instruction set architecture*—the interface between the hardware and low-level software. This abstract interface enables many *implementations* of varying cost and performance to run identical software.

A Safe Place for Data

Thus far, we have seen how to input data, compute using the data, and display data. If we were to lose power to the computer, however, everything would be lost because the memory inside the computer is **volatile**—that is, when it loses power, it forgets. In contrast, a DVD doesn't forget the recorded film when you turn off the power to the DVD player and is thus a **nonvolatile memory** technology.

To distinguish between the volatile memory used to hold data and programs while they are running and this nonvolatile memory used to store data and programs between runs, the term **main memory** or **primary memory** is used for the

volatile memory Storage, such as DRAM, that retains data only if it is receiving power.

nonvolatile memory A form of memory that retains data even in the absence of a power source and that is used to store programs between runs. Magnetic disk is nonvolatile.

main memory Also called **primary memory**. Memory used to hold programs while they are running; typically consists of DRAM in today's computers.

secondary memory Nonvolatile memory used to store programs and data between runs; typically consists of magnetic disks in today's computers.

magnetic disk Also called **hard disk**. A form of nonvolatile secondary memory composed of rotating platters coated with a magnetic recording material.

flash memory A nonvolatile semiconductor memory. It is cheaper and slower than DRAM but more expensive and faster than magnetic disks.

former, and **secondary memory** for the latter. DRAMs have dominated main memory since 1975, but **magnetic disks** have dominated secondary memory since 1965. The primary nonvolatile storage used in all server computers and workstations is the magnetic **hard disk**. **Flash memory**, a nonvolatile semiconductor memory, is used instead of disks in mobile devices such as cell phones and is increasingly replacing disks in music players and even laptops.

As Figure 1.10 shows, a magnetic hard disk consists of a collection of platters, which rotate on a spindle at 5400 to 15,000 revolutions per minute. The metal platters are covered with magnetic recording material on both sides, similar to the material found on a cassette or videotape. To read and write information on a hard disk, a movable *arm* containing a small electromagnetic coil called a *read-write head* is located just above each surface. The entire drive is permanently sealed to control the environment inside the drive, which, in turn, allows the disk heads to be much closer to the drive surface.

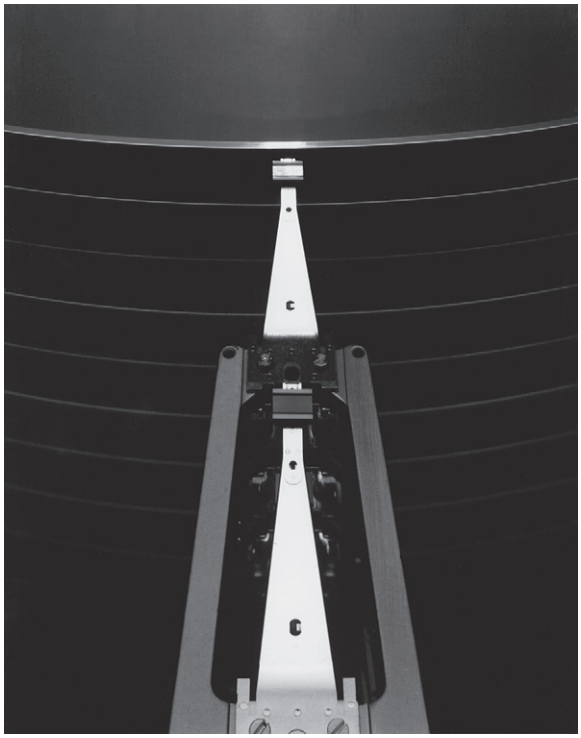

FIGURE 1.10 A disk showing 10 disk platters and the read/write heads.

Diameters of hard disks vary by more than a factor of 3 today, from 1 inch to 3.5 inches, and have been shrunk over the years to fit into new products; workstation servers, personal computers, laptops, palmtops, and digital cameras have all inspired new disk form factors. Traditionally, the widest disks have the highest performance and the smallest disks have the lowest unit cost. The best cost per **gigabyte** varies. Although most hard drives appear inside computers, as in Figure 1.7, hard drives can also be attached using external interfaces such as universal serial bus (USB).

gigabyte Traditionally 1,073,741,824 (2^{30}) bytes, although some communications and secondary storage systems have redefined it to mean 1,000,000,000 (10^9) bytes. Similarly, depending on the context, megabyte is either 2^{20} or 10^6 bytes.

The use of mechanical components means that access times for magnetic disks are much slower than for DRAMs: disks typically take 5–20 milliseconds, while DRAMs take 50–70 nanoseconds—making DRAMs about 100,000 times faster. Yet disks have much lower costs than DRAM for the same storage capacity, because the production costs for a given amount of disk storage are lower than for the same amount of integrated circuit. In 2008, the cost per gigabyte of disk is 30 to 100 times less expensive than DRAM.

Thus, there are three primary differences between magnetic disks and main memory: disks are nonvolatile because they are magnetic; they have a slower access time because they are mechanical devices; and they are cheaper per gigabyte because they have very high storage capacity at a modest cost.

Many have tried to invent a technology cheaper than DRAM but faster than disk to fill that gap, but many have failed. Challengers have never had a product to market at the right time. By the time a new product would ship, DRAMs and disks had continued to make rapid advances, costs had dropped accordingly, and the challenging product was immediately obsolete.

Flash memory, however, is a serious challenger. This semiconductor memory is nonvolatile like disks and has about the same bandwidth, but latency is 100 to 1000 times faster than disk. Flash is popular in cameras and portable music players because it comes in much smaller capacities, it is more rugged, and it is more power efficient than disks, despite the cost per gigabyte in 2008 being about 6 to 10 times higher than disk. Unlike disks and DRAM, flash memory bits wear out after 100,000 to 1,000,000 writes. Thus, file systems must keep track of the number of writes and have a strategy to avoid wearing out storage, such as by moving popular data. Chapter 6 describes flash in more detail.

Although hard drives are not removable, there are several storage technologies in use that include the following:

- Optical disks, including both compact disks (CDs) and digital video disks (DVDs), constitute the most common form of removable storage. The Blu-Ray (BD) optical disk standard is the heir-apparent to DVD.
- Flash-based removable memory cards typically attach to a USB connection and are often used to transfer files.
- Magnetic tape provides only slow serial access and has been used to back up disks, a role now often replaced by duplicate hard drives.

Optical disk technology works differently than magnetic disk technology. In a CD, data is recorded in a spiral fashion, with individual bits being recorded by burning small pits—approximately 1 micron (10^{-6} meters) in diameter—into the disk surface. The disk is read by shining a laser at the CD surface and determining by examining the reflected light whether there is a pit or flat (reflective) surface. DVDs use the same approach of bouncing a laser beam off a series of pits and flat surfaces. In addition, there are multiple layers that the laser beam can focus on, and the size of each bit is much smaller, which together increase capacity significantly. Blu-Ray uses shorter wavelength lasers that shrink the size of the bits and thereby increase capacity.

Optical disk writers in personal computers use a laser to make the pits in the recording layer on the CD or DVD surface. This writing process is relatively slow, taking from minutes (for a full CD) to tens of minutes (for a full DVD). Thus, for large quantities a different technique called *pressing* is used, which costs only pennies per optical disk.

Rewritable CDs and DVDs use a different recording surface that has a crystalline, reflective material; pits are formed that are not reflective in a manner similar to that for a write-once CD or DVD. To erase the CD or DVD, the surface is heated and cooled slowly, allowing an annealing process to restore the surface recording layer to its crystalline structure. These rewritable disks are the most expensive, with write-once being cheaper; for read-only disks—used to distribute software, music, or movies—both the disk cost and recording cost are much lower.

Communicating with Other Computers

We've explained how we can input, compute, display, and save data, but there is still one missing item found in today's computers: computer networks. Just as the processor shown in Figure 1.4 is connected to memory and I/O devices, networks interconnect whole computers, allowing computer users to extend the power of computing by including communication. Networks have become so popular that they are the backbone of current computer systems; a new computer without an optional network interface would be ridiculed. Networked computers have several major advantages:

- *Communication:* Information is exchanged between computers at high speeds.
- *Resource sharing:* Rather than each computer having its own I/O devices, devices can be shared by computers on the network.
- *Nonlocal access:* By connecting computers over long distances, users need not be near the computer they are using.

Networks vary in length and performance, with the cost of communication increasing according to both the speed of communication and the distance that information travels. Perhaps the most popular type of network is *Ethernet*. It can be up to a kilometer long and transfer at upto 10 gigabits per second. Its length and

speed make Ethernet useful to connect computers on the same floor of a building; hence, it is an example of what is generically called a **local area network**. Local area networks are interconnected with switches that can also provide routing services and security. **Wide area networks** cross continents and are the backbone of the Internet, which supports the World Wide Web. They are typically based on optical fibers and are leased from telecommunication companies.

local area network (LAN) A network designed to carry data within a geographically confined area, typically within a single building.

wide area network (WAN) A network extended over hundreds of kilometers that can span a continent.

Networks have changed the face of computing in the last 25 years, both by becoming much more ubiquitous and by making dramatic increases in performance. In the 1970s, very few individuals had access to electronic mail, the Internet and Web did not exist, and physically mailing magnetic tapes was the primary way to transfer large amounts of data between two locations. Local area networks were almost nonexistent, and the few existing wide area networks had limited capacity and restricted access.

As networking technology improved, it became much cheaper and had a much higher capacity. For example, the first standardized local area network technology, developed about 25 years ago, was a version of Ethernet that had a maximum capacity (also called bandwidth) of 10 million bits per second, typically shared by tens of, if not a hundred, computers. Today, local area network technology offers a capacity of from 100 million bits per second to 10 gigabits per second, usually shared by at most a few computers. Optical communications technology has allowed similar growth in the capacity of wide area networks, from hundreds of kilobits to gigabits and from hundreds of computers connected to a worldwide network to millions of computers connected. This combination of dramatic rise in deployment of networking combined with increases in capacity have made network technology central to the information revolution of the last 25 years.

For the last decade another innovation in networking is reshaping the way computers communicate. Wireless technology is widespread, and laptops now incorporate this technology. The ability to make a radio in the same low-cost semiconductor technology (CMOS) used for memory and microprocessors enabled a significant improvement in price, leading to an explosion in deployment. Currently available wireless technologies, called by the IEEE standard name 802.11, allow for transmission rates from 1 to nearly 100 million bits per second. Wireless technology is quite a bit different from wire-based networks, since all users in an immediate area share the airwaves.

Check Yourself

- Semiconductor DRAM and disk storage differ significantly. Describe the fundamental difference for each of the following: volatility, access time, and cost.

Technologies for Building Processors and Memory

Processors and memory have improved at an incredible rate, because computer designers have long embraced the latest in electronic technology to try to win the race to design a better computer. Figure 1.11 shows the technologies that have been

used over time, with an estimate of the relative performance per unit cost for each technology. Section 1.7 explores the technology that has fueled the computer industry since 1975 and will continue to do so for the foreseeable future. Since this technology shapes what computers will be able to do and how quickly they will evolve, we believe all computer professionals should be familiar with the basics of integrated circuits.

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2005	Ultra large-scale integrated circuit	6,200,000,000

FIGURE 1.11 Relative performance per unit cost of technologies used in computers over time. Source: Computer Museum, Boston, with 2005 extrapolated by the authors. See Section 1.10 on the CD.

vacuum tube An electronic component, predecessor of the transistor, that consists of a hollow glass tube about 5 to 10 cm long from which as much air has been removed as possible and that uses an electron beam to transfer data.

transistor An on/off switch controlled by an electric signal.

very large-scale integrated (VLSI) circuit A device containing hundreds of thousands to millions of transistors.

A **transistor** is simply an on/off switch controlled by electricity. The *integrated circuit* (IC) combined dozens to hundreds of transistors into a single chip. To describe the tremendous increase in the number of transistors from hundreds to millions, the adjective *very large scale* is added to the term, creating the abbreviation *VLSI*, for **very large-scale integrated circuit**.

This rate of increasing integration has been remarkably stable. Figure 1.12 shows the growth in DRAM capacity since 1977. For 20 years, the industry has consistently quadrupled capacity every 3 years, resulting in an increase in excess of 16,000 times! This increase in transistor count for an integrated circuit is popularly known as Moore's law, which states that transistor capacity doubles every 18–24 months. Moore's law resulted from a prediction of such growth in IC capacity made by Gordon Moore, one of the founders of Intel during the 1960s.

Sustaining this rate of progress for almost 40 years has required incredible innovation in manufacturing techniques. In Section 1.7, we discuss how to manufacture integrated circuits.

1.4 Performance

Assessing the performance of computers can be quite challenging. The scale and intricacy of modern software systems, together with the wide range of performance improvement techniques employed by hardware designers, have made performance assessment much more difficult.

When trying to choose among different computers, performance is an important attribute. Accurately measuring and comparing different computers is critical to