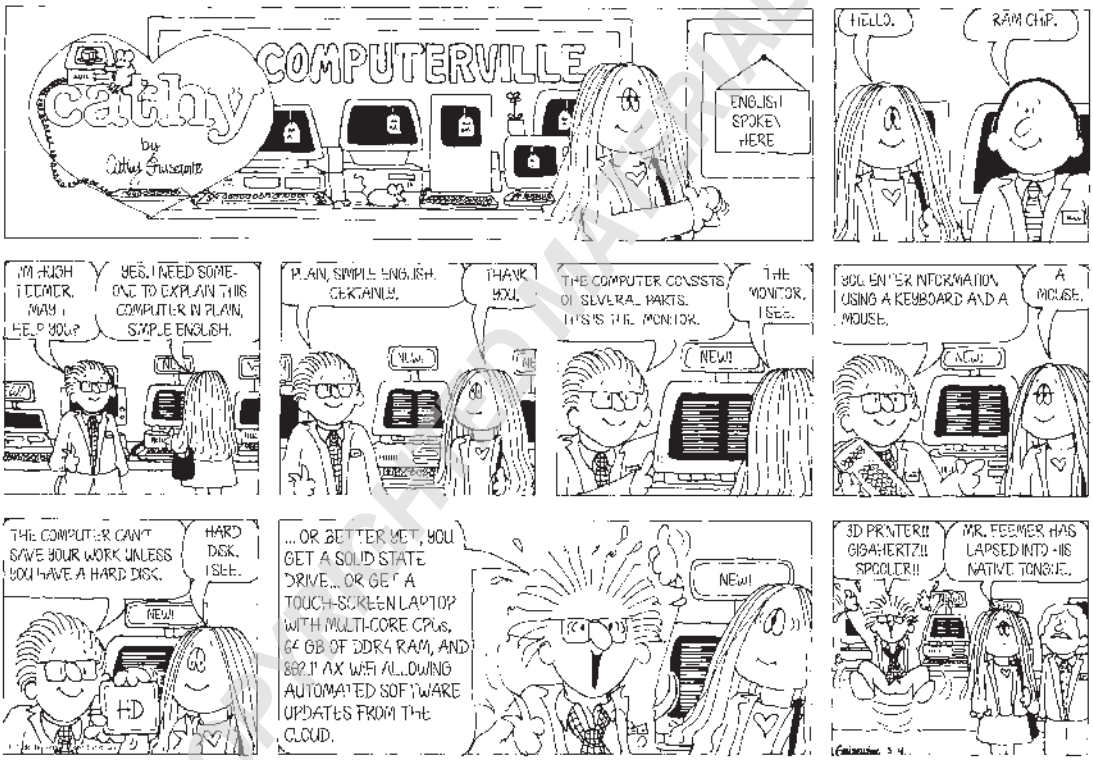
1 Computers and Systems



1.0 Introduction

Welcome to the wonderful modern world of computers. Where the technology changes daily. Or does it?

It's true that the computer of today looks nothing like the computer of even five or ten years ago. It is nearly impossible today to escape the immediate and ubiquitous reach of computers and computer-based systems and devices. There is probably a smartphone in your pocket or on your desk. For many of you, your laptop or tablet is sitting nearby as you read this paragraph. (Or, maybe you're even *reading* this paragraph on your tablet or e-book.) Furthermore, your smartphone probably has more computing power than most of the computers on the market ten years ago. It fits easily in your pocket or purse. It weighs less than 1/100 of that old desktop computer and has at least ten times as much memory!

And that's not all. Your car has several embedded computers controlling various automotive functions, and perhaps a touch screen for hands-free telephone, navigation, the radio, Internet access, and more. Which is almost unnecessary, because you can probably tell it what you want verbally anyway. Even your microwave oven and the machine that launders your clothes depend on computers to function. As you are likely aware, most of these machines can talk to each other, using Bluetooth, the Internet, or some other networking technology. (Think IoT, the Internet of Things, as just one example.) Just for fun, Figure 1.1 shows pictures typical of a recent laptop, a 2020 smartphone, and a current embedded computer that controls many functions in your car.

Although the focus of this book is on IT systems, our discussion of computer hardware, software, and networking applies equally well to workplace computers, tablets, smartphones, and, even computers embedded in other devices and equipment. In this figure, we have three seemingly very different looking pieces of equipment working on different types of applications. And yet, it's hopefully obvious to you that these three systems share a lot in common. They are all computer based. All contain at least one central processing unit (CPU, some contain more) and memory. All provide a facility for interacting with long-term storage and other devices and with users. What may be less obvious to you is that the programs that they run are also essentially similar, differing mostly in the details required by the different components of the particular system and by the nature of the applications. For example, systems may have different amounts of memory, different types of displays, different I/O devices, and different operating systems, as well as running different types of applications and serving different purposes.

In fact, a modern IT system may contain elements of many different types of systems, with networking that ties everything together.

When creating an IT system, our concerns are whether the various components provide the features and performance that the users require. To be an effective designer and user, you have to understand the specifications, their importance and their meaning; the terminology; and the jargon. Which features are important to the users? Is this the right combination

Global American, Inc.

FIGURE 1.1
Computer Devices, Old and New

of features that you need in your computer to have the computer perform the work that you wish to get done? Are there features missing that we need? Perhaps we are paying too much for the performance that we need. Or maybe we need more. What other information about this system would allow you to make a more informed decision? There is obviously no need to understand the inner workings of most modern computer-based systems to operate them adequately. Indeed, in many cases, the presence of the computer is hidden from us, or **embedded**, and its operation is invisible to us as users. We don't need to know how a computer works to read an e-book.

Even as experienced users, we can run standard software packages on a personal computer or apps on a smartphone without understanding exactly how they work; we can program a computer in a high-level or a scripting language without understanding the details of how the machine executes the individual instructions; we can design and implement Web pages without understanding how the Web browser gets its pages from a Web server or how the Web server creates those pages; we can purchase a tablet or a laptop computer from a salesperson without understanding the specifications of the system.

And yet, there is something missing. Perhaps the software doesn't do exactly what we want, and we don't understand the machine well enough to risk fooling around with the software's options. Perhaps if we understood the system better, we might have written and configured the program to be faster and more efficient. Perhaps we could create Web pages that load faster and work better. Perhaps the salesperson did not sell us the optimum system for our job. Or perhaps it's nothing more than a sense of excitement that's missing. But that's important, too!

You are reading this book because you are a student studying to become a computer professional, or maybe you are simply a user wanting a deeper understanding of what the computer is all about. In either case, you know you'll be interacting with computer systems in some form or other for the rest of your life. It's nice (as well as useful) to know something about the tools of the trade. More importantly, understanding the computer system's operations has an immediate benefit: it will allow you to use the machine more effectively.

As a user, you will be more aware of the capabilities, strengths, and limitations of a computer system. You will have a better understanding of the commands that you use. You will understand what is taking place during the operation of the program applications that you use. You will be able to make better informed decisions about your computer equipment and application programs. You will understand more clearly what an operating system is, and how to use it effectively and to your advantage. You will know when it is preferable to do a job manually, and when the computer should be used. You will understand the most efficient way to "go online", and what benefits might be gained from your home network. You will improve your ability to communicate with system analysts, programmers, and other computer specialists. You might even save money! Do you really need to pay for 50 GB of download data on your mobile phone every month? And what is that 50 GB being used for?

As a programmer, it will allow you to write better programs. You will be able to use the characteristics of the machine to make your programs operate more effectively. For example, choosing the appropriate data type for a variable can result in significantly faster performance. Soon you will know why this is so, and how to make the appropriate choices.

You will discover that some computers will process nested loops much more quickly if the index variables are reversed. A rather surprising idea, perhaps, and you'll understand why this is true.

You will understand why programs written in a compiled language like C++ usually run much faster than those written in interpreted program languages like BASIC or scripting languages like JavaScript. Similarly, you'll see why the basic layout of a program can have a major impact on the program's run-time efficiency.

As a systems architect or system analyst, you will be responsible for the design and implementation of systems that meet an organization's information technology (IT) needs, recognizing that the differences in the cost and capabilities of the components that you select may have significant impact on the organization. With the knowledge gained here you will be in a better position to determine and justify the set of computer system components and the system architecture that are appropriate for a particular job and to determine the trade-offs with other possible system architectures.

You'll be able to assist management in making intelligent decisions about system strategy: should the company adopt a large mainframe/virtual machine system approach for its Web servers, or would a system consisting of a network of off-the-shelf blade servers provide better performance at lower cost? You'll be better prepared to analyze the best way to provide appropriate facilities to meet the needs of your users. In an era of fast-changing technology, you'll be more able to differentiate between simple technological obsolescence that does not affect the organization's requirements significantly and major advances that suggest a real need to replace older equipment. You will understand the trade-offs inherent in the use of cloud and other remote services.

When selecting computers, you would like to purchase the computers and services that best meet the needs of the organization's applications and the users. You must be able to read and understand the technical specifications in order to compare different alternatives and to match the system to the users' needs. This book will teach you what you need to know to specify and purchase a system intelligently. You'll know the differences between various CPU technologies and the advantages and disadvantages of each. You will learn what peripheral hardware is appropriate for your organization's files and the trade-offs between different file system formats, what is required to build an intranet, and what the speed, size, and performance limitations of a particular system are. You'll be able to compare the features of macOS, Windows, Chrome OS, and Linux knowledgeably and decide which ones are important to you. You'll be able to apply your basic understanding of computers to new technologies and concepts such as mobile IT, new network protocols, virtual machines and cloud services as they appear. You'll learn to understand the jargon used by computer salespeople and judge the validity of their sales claims.

As a networking professional, you are responsible for the design, maintenance, support, and management of the networks that connect your computer systems together and provide the required interfaces to the outside world. You must be able to specify network layouts that optimize your equipment and network resources. Your understanding of basic network configurations and protocols will allow you to control and provide sufficient and appropriate access to your users in an efficient manner. This text offers the basic tools as a starting point to prepare for a career in networking.

As a Web services designer, you must be able to make intelligent decisions to optimize your Web system configurations, page designs, data formatting and scripting language choices, security features, and operating systems to optimize customer accessibility to your Web services.

As a system administrator or manager, your job is to maximize the availability and efficiency of your systems. You will need to understand the reports generated by your systems and be able to use the information in those reports to make changes to the systems that will optimize system performance. You will need to know when additional resources are required, and be able to specify appropriate choices. You will need to specify and configure operating system parameters, set up file systems, select cloud services, manage system and user PC upgrades in a fast-changing environment, reconfigure networks, provide and ensure the robustness of system security and data backup, and perform many other system management tasks. The configuration of large systems can be very challenging. This text will give you an understanding of operating system tools that is essential to the effective management of systems.

In brief, when you complete this book, you will understand what computer hardware and software are and how programs and data interact with the computer system. You will understand the computer hardware, software, and communication components that are required to make up a computer system and what the role of each component in the system is.

You will have a better understanding of what is happening inside the computer when you interact with the computer as a user. You will be able to write programs that are more efficient. You will be able to understand the function of the different components of the computer system and to specify the computer equipment and resources you need in a meaningful way. You will understand the options that you have as a system administrator or Web services or network designer.

In an era in which technology changes extremely rapidly, the architecture of the computer system rests on a solid foundation that has changed only slightly and gradually over the last sixty years. Understanding the foundations of computer system architecture makes it possible to flow comfortably with technological change and to understand changes in the context of the improvements that they make and the needs that they meet. In fact, interviews with former students and with IT executives and other IT professionals clearly indicate that a deep understanding of the basic concepts presented here is fundamental to long-term survival and growth in the field of information technology and IT management.

This type of understanding is at the very foundation of being a competent and successful system analyst, system architect, system administrator, or programmer. It may not be necessary to understand the workings of an automobile engine in order to drive a car, but you can bet that a top-notch race car driver knows his or her engine thoroughly and can use it to win races. Like the professional race car driver, it is our intention to help you to use your computer engine effectively to succeed in using your computer in a winning way. The typical end user might not care about how their computer system works, but you do.

These are the goals of this book. So let's get started!

1.1 The Starting Point

Before we begin our detailed study of the architecture of computer systems, let us briefly review some of the fundamental principles, characteristics, and requirements that guide computer system design and operation. The fundamentals described here apply to computers in general, regardless of size or purpose, from the smallest embedded device to the largest mainframe computer.

In a simple scenario, you use your tablet, laptop, or desktop personal computer to word process a document. You probably use a pointing device such as a mouse or stylus or finger to move around the document and to control the features of the word processor software application, and you use a keyboard or touch screen to enter and modify the document text data. The word processor application program, together with your document, appears on a screen. Ultimately, you might print the document on a printer. You store the document on a solid-state drive (SSD), disk, flash drive, or some other storage device.

The fundamentals of a typical computer system are readily exposed in this simple example. Your pointing device movements and clicks and your text data entry represent input to the system. The computer processes the input and provides output to the screen, and, perhaps, to a printer. The computer system also provides a storage medium of some sort, usually an SSD or a hard disk, to store the text for future access. In simplest terms, your computer receives input from you, processes it, and outputs results to the screen. Your input takes the form of commands and data. The commands and programs tell the computer how to process the data.

Now consider a second, slightly more complex example. Your task in this example is to access a Web page on the Internet. Again, your input to the computer is via keyboard and pointer control device. When you type the Web page URL, however, your computer sends a message to another computer that contains Web server software. That computer, in turn, sends a Web page file that is interpreted by the browser on your computer and presented on your screen. You are probably already aware that Hypertext Transfer Protocol (HTTP) is used as a standard for Web message exchanges.

The elements of this example differ only slightly from the first example. Your command inputs tell a Web browser software application on your computer what processing is to take place; in this case, your desire to access a Web page. The output from your computer is a message to a Web server on the remote computer requesting the data that represents the Web page. Your computer receives the data as input from the network; the Web browser processes the data and presents the Web page output on the screen. Figure 1.2 illustrates the layout for this example.[1]

The major differences between this and the first example are the source of the input data and the fact that the second example requires network connectivity between the two computers. Instead of the keyboard, the input data to be processed by your Web browser comes from a communication channel. (Note that the exact nature of the channel is not important for this discussion.) In both cases, your computer receives data input to process, and control input, usually HTML or XML, that determines how the data is to be processed, performs the processing, and provides output.

These two examples contain all of the key elements found in any IT system, large or small.

- An IT system consists of one or more computer systems; multiple computer systems are connected together using some type of network interconnectivity. As a matter of necessity, network interfaces must conform to standard agreements, known as **protocols**, for messages to be understood by both computers during a message exchange between a pair of computers. The network itself can take on a variety of forms, provided that the interface requirements are met, and are determined by such characteristics as performance, convenience, and cost.

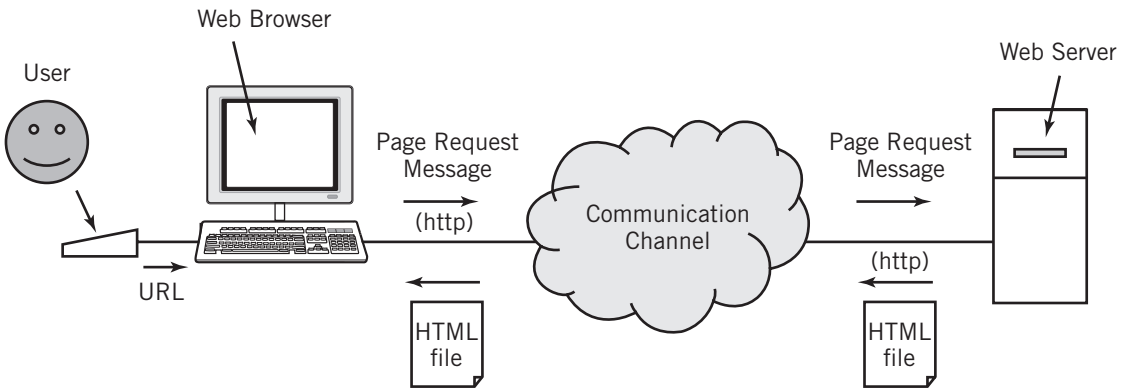


FIGURE 1.2 Typical Web Browser Application Use

[1]The cloud in the image refers to the network connection that makes communication possible between the two computers, rather than the computing services delivered over the Internet known as cloud computing.

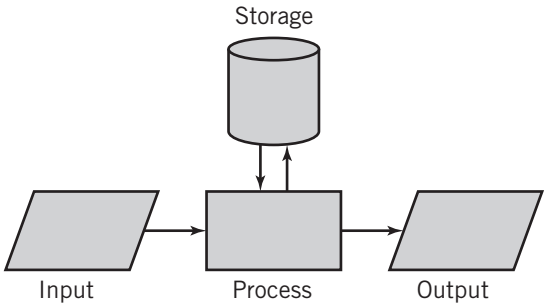


FIGURE 1.3 A Computer Process

- The work performed by an individual computer system within the IT system can be characterized by input, processing, and output. This characterization is often represented by the **input–process–output (IPO) model** shown in Figure 1.3. Storage is also represented within this model. Alternatively, storage can be interpreted as output to be saved for use as future input. Storage is also used to hold the software programs that determine the processing operations to be performed. The ability to store programs and data on a temporary, short-term, or long-term basis is fundamental to the system. In Chapter 2, Section 2.2, and Chapter 15, we will show that all IT systems can ultimately be characterized by the same basic IPO model at all levels, from a single computer to a complex aggregation of computers, although the complexity of large systems may obscure the model and make it more difficult to determine the actual inputs, outputs, and processing operations. The IPO model provides an important basic tool for system analysis and design practices.
- The components of an individual computer system consist of processing hardware, input devices, output devices, storage devices, application software, and operating system software. The task of the operating system software is to provide overall control of the individual system, including management of input, output, and file storage functions. The medium of exchange, both with users and between computers within a larger system, is data. (Note that the messages between computers in the second example are a form of data.) Figure 1.4 is a simple illustration of computer systems embedded in a larger IT system.

Figure 1.5 summarizes the basic operations that are performed during computer processing. These operations, in turn, can be reduced to the primitive operations that are also familiar to you from your understanding of programming languages. The primitive processing operations common to high-level programming languages are shown in Figure 1.6.

1.2 Components of the Computer System

As noted in the previous section, there are three components required for the implementation of a computerized input–process–output model:

1. The computer hardware, which provides the physical mechanisms to input and output data, to manipulate and process data, and to electronically control the various input, output, and storage components.
2. The software, both application and system, which provides instructions that tell the hardware exactly what tasks are to be performed and in what order.

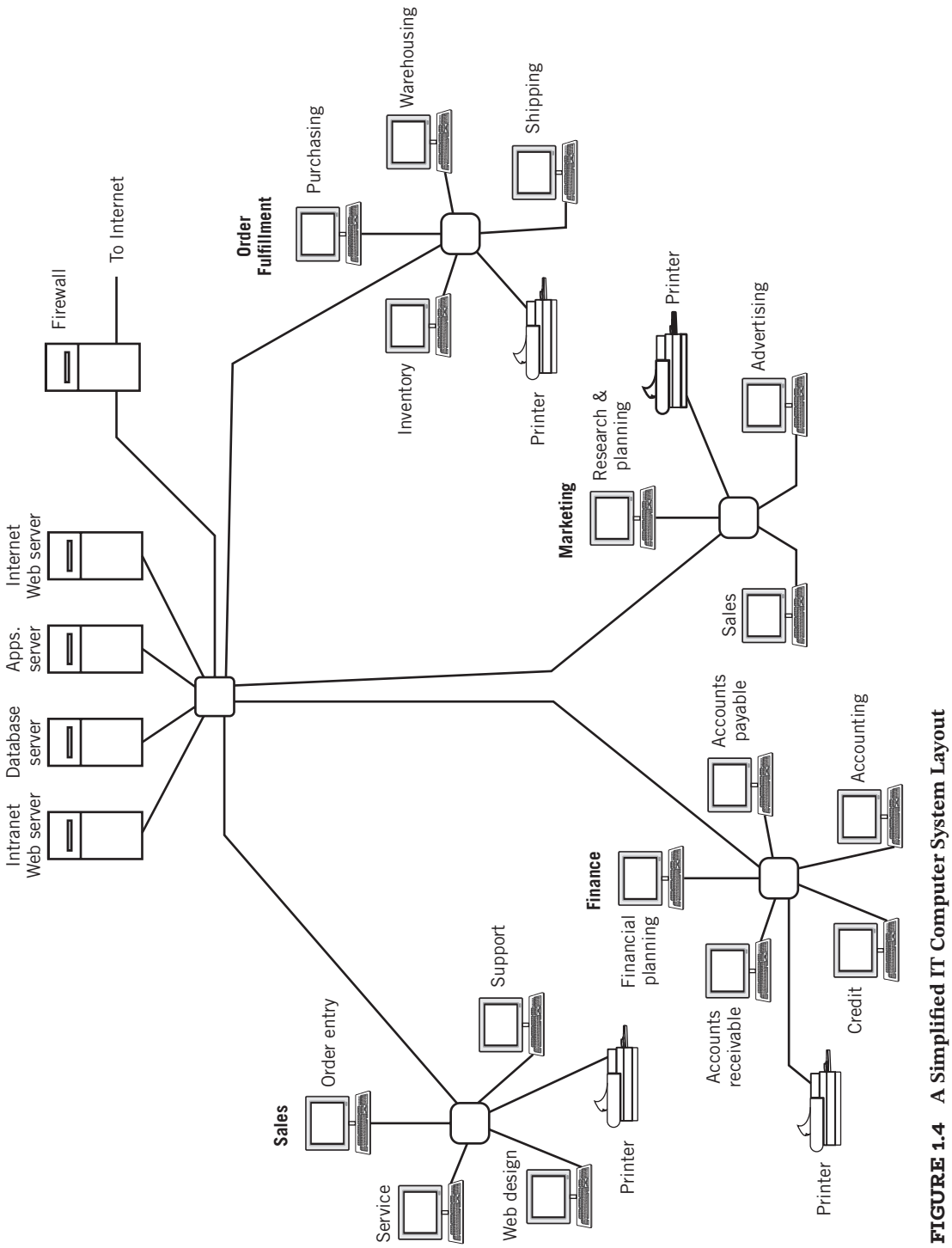


FIGURE 1.4 A Simplified IT Computer System Layout

- Input/output
- Basic arithmetic and logical calculations
- Data transformation or translation (e.g., program compilation, foreign language translation, file updating)
- Data sorting
- Searching for data matches
- Data storage and retrieval
- Data movement (e.g., movement of text or file data to make room for insertion of additional data)

FIGURE 1.5 Basic Computer Operations

- Input/output (including file storage and retrieval)
- Arithmetic and logical assignment statements
- True/false decision branching (IF-THEN-ELSE or IF-GOTO)
- Loops and/or unconditional branching (WHILE-DO, REPEAT-UNTIL, FOR, GOTO)

FIGURE 1.6 Basic High-Level Language Constructs

3. The data that is being manipulated and processed. This data may be numeric, it may be alphanumeric, it may be graphic, or it may take some other form, but in all cases, it must be representable in a form that the computer can manipulate.

In modern systems, input entry, output display, and storage of the data and software used for processing often take place at a location different from the computer where the actual processing occurs. In many installations, actual processing is distributed among computer systems, with particular results passed to the individual systems that require them. Therefore, we must also consider a fourth component:

4. The communication component, which consists of hardware and software that transport programs and data between interconnected computer systems.

The hardware and system software components make up the architecture of the computer system. The communication component connects individual computer systems together to form an organizational system architecture. The data component and also the application software, while fundamental to the operation of the computer system, are supplied to the computer system by the user or vendor, rather than being a part of the architecture of the computer system itself. (It is useful to note, however, that application software and data structure *are* often considered as part of the overall system architecture when one considers the architecture from the perspective of the organization. We explore this issue in Chapter 15, Section 15.3. Note, however, that the focus of this book is primarily on *computer* system architecture, rather than on *organizational* system architecture.)

The Hardware Component

The most visible part of the computer system is obviously the hardware that makes up the system. Consider the computer system upon which you write and execute your programs. You use a keyboard or touch screen and a pointing device to provide **input** of your program text and data, as well as for commands to the computer. A display screen is commonly used to observe **output**. A printer is frequently available as an alternative output to the screen. These are all physical components.

Calculations and other operations in your program are performed by one or more **central processing units (CPUs)**, or "*cores*" inside the computer. **Memory** is provided to hold your programs and data while processing is taking place. Other I/O devices, such as an SSD, disk, and SD plug-in cards, are used to provide long-term storage of your program and data files. Data and programs are transferred between the various I/O devices and memory for the CPUs to use.

The CPUs, memory, and all the input, output, and storage devices form the **hardware** part of a computer system. The hardware forms the tangible part of the system. It is physical—you can touch it, which is what the word "tangible" means. A typical hardware block diagram for a computer is seen in Figure 1.7. In addition to the I/O devices shown in this diagram, Figure 1.8 lists some other I/O devices that are frequently seen as part of computer systems. The diagram in Figure 1.7 actually applies equally well to large mainframe computers, small personal computers, cell phones, and tablets, and even devices with computers embedded in them, such as automobiles, smart doorbells, and home voice recognition devices. Large and small computers differ primarily in the number of cores, the amount of memory, speed, capacity, and the selection of I/O devices provided. The basic hardware components and design are very similar.

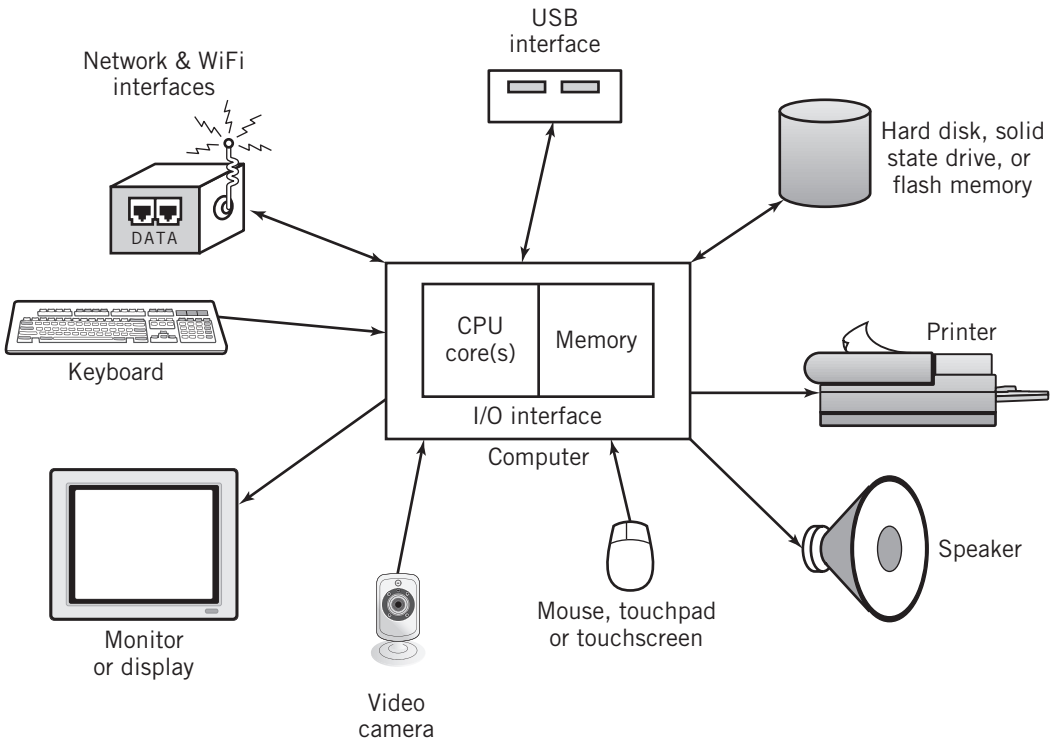


FIGURE 1.7
A Typical Personal Computer System

- Page and document scanners
- RFID and NearFieldCommunication readers
- TV and radio tuners
- GPS receivers
- Cellular and Bluetooth communication technology
- SD, SmartCard, etc. card readers
- Fingerprint readers
- Graphics tablets
- Other mobile devices: accelerometers, gyroscopes, etc.

FIGURE 1.8 Other Common Input/Output Devices

Conceptually, a CPU itself is often viewed as a composition of three primary subunits:

1. The **arithmetic/logic unit (ALU)** where arithmetic and Boolean logical calculations are performed.
2. The **control unit (CU)**, which controls the processing of instructions and the movement of internal CPU data from one part of the CPU to another.
3. The **interface unit**, which moves program instructions and data between the CPU and other hardware components.

(In modern CPUs, the actual implementation is usually modified somewhat to achieve higher performance, although the basic concepts are carefully preserved. More about that later, in Chapter 8.)

The interface unit interconnects the CPU with memory and also with the various I/O modules. It can also be used to connect multiple CPU cores together. In many computer systems, a bus interconnects the CPU, memory, and all of the I/O components. A **bus** is simply a bundle of wires that carry signals and power between different components. In other systems, the I/O modules are connected to the CPU through one or more separate processors known as **channels**.

The main memory, often known as primary storage, working storage, or **RAM** (for **r**andom **a**ccess **m**emory), holds programs and data for access by the CPU. **Primary storage** is made up of a large number of cells, each numbered and individually addressable. Each cell holds a single binary number representing part of a data value or part of an instruction. The smallest addressable size of the cell in most current computers is 8 bits, known as a **byte** of memory. Eight bits of memory can only hold 256 different patterns, so neighboring cells in memory are nearly always combined to form groupings with a larger number of bits. In many systems, for example, 4 bytes of memory combine to form a 32-bit **word**. Modern computers address memory at least 4 bytes (a "32-bit" computer) or 8 bytes (a "64-bit" computer) at a time to take advantage of larger instruction and data groupings.

The amount of primary storage determines the maximum number of instructions and data words that can be loaded into memory from a peripheral device at one time. For example, a computer with 8 gigabytes (GB), actually 8,589,934,592 bytes,[2] of memory would not be able to execute a program that requires 9.7 GB for its instructions and data unless some means is provided within the computer to load the program in sections as each section of the program is needed.

[2] 1 kilobyte actually equals 1024 bytes. Thus, 1 MB = 1024 × 1024 = 1,048,576 bytes × 8192 = 8,589,934,592 bytes.

The amount of primary storage provided in a typical computer has increased rapidly as computer technology improves. Whereas 64 *kilo*bytes (KB) of memory was considered a large amount in 1980, even the least expensive personal computers today usually have 8 *giga*bytes (GB) of memory or more. Large computers may provide many terabytes of primary storage. There are programs on the market that require hundreds of megabytes (MB) or gigabytes (GB) of memory to execute. The inexpensive availability of increased amounts of memory has allowed the design of very sophisticated programs that would not have been possible just a few years ago.

The same is true for secondary storage. Even small personal computers provide long-term storage using hard disks or solid-state storage devices with storage measured in hundreds or thousands of gigabytes. The storage of images and video, in particular, requires tremendous amounts of storage capacity. It is not uncommon to see systems providing tens or hundreds of trillions of bytes (specified as *terabytes*) of long-term storage.

The instructions that form a particular program are stored within the primary storage, then brought into the CPU and executed. Conceptually, instructions are brought in and executed one at a time, although modern systems overlap the execution of instructions to some extent. Instructions must be in primary storage in order to be executed. The control unit interprets each instruction and determines the appropriate course of action.

Each instruction is designed to perform a simple task. Instructions exist to perform basic arithmetic, to move data from one place in the computer to another, to perform I/O, and to accomplish many other tasks. The computer's power comes from the ability to execute these simple instructions at extremely high speeds, measured in billions or trillions of instructions executed per second. As you are already aware, it is necessary to translate high-level language programs into the language of the machine for execution of the program to take place. It may require tens, hundreds, or even thousands of individual machine instructions to form the machine language equivalent of a single high-level language statement. Program instructions are normally executed sequentially, unless an instruction itself tells the computer to change the order of processing. The instruction set used with a particular CPU is part of the design of the CPU and cannot normally be executed on a different type of CPU unless the different CPU was designed to be instruction set compatible. However, as you shall see, most instruction sets perform similar types of operations. As a result, it is possible to write programs that will **emulate** the instruction set from one computer on a computer with a different instruction set, although a program written for the original machine may execute slower on the machine with the emulator.

The data that is manipulated by these instructions is also stored in memory while being processed. The idea that the program instructions and data are both stored in memory while being processed is known as the **stored program concept**. This important concept is attributed primarily to John von Neumann, a famous computer scientist. It forms the basis for the computer architecture that is standard to nearly every existing computer.

The Software Component

In addition to the hardware requirement, your computer system also requires **software**. Software consists of the programs that tell the computer what to do. To do useful work, your system must execute instructions from some program.

There are two major categories of software: system software and application software. System software helps you to manage your files, to load and execute programs, and to accept your commands. The system software programs that manage the computer are collectively known as an **operating system**, and differ from the application programs, such as Microsoft Word or Facebook or Zoom or the programs that you write, that you normally

run to get your work done. Windows, Linux, macOS, iOS, and Android are the best known current examples of an operating system. Others include UNIX and IBM z/OS.

The operating system is an essential part of the computer system. Like the hardware, it is made up of many components. A simplified representation of an operating system is shown in Figure 1.9. The most obvious element is the user interface that allows you to execute programs, enter commands, and manipulate files. The user interface accepts input from a keyboard and, in most modern systems, a mouse, touch screen, or other pointing device. The user interface also does output presentation on the display. On some systems, the output display might be simple text, but more likely the display includes a graphical user interface with a windowing system, and various gadgets for manipulating the windows.

The operating system's **application programming interface (API)** acts as an interface for application programs and utilities to access the internal services provided by the operating system. These include file services, I/O services, data communication services, user interface services, program execution services, and more.[3]

Many of the internal services are provided by the **kernel** module, which contains the most important operating system processing functions. The remaining services are provided by other modules that are controlled by the kernel. The kernel manages memory by locating and allocating space to programs that need it, schedules time for each application to execute, provides communication between programs that are being executed, manages and arranges services and resources that are provided by other modules, and provides security.

The file management system allocates and manages secondary storage space and translates file requests from their name-based form into specific I/O requests. The actual storage

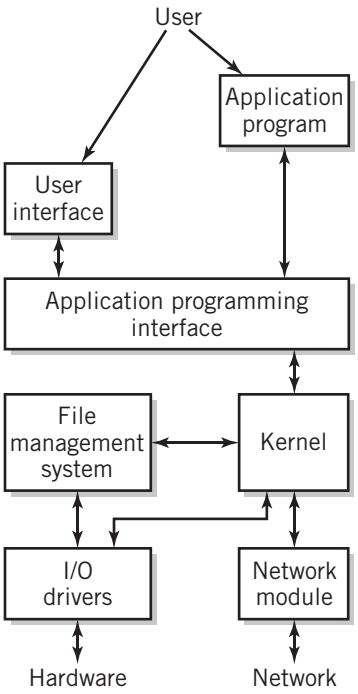


FIGURE 1.9 Simplified Operating System Block Diagram

[3]The same term (API) is also used to describe the services provided by one application to another. For example, Amazon and Google are among many companies whose application software provides API tools to allow users to extend the functionality of the original software.

and retrieval of the files is performed by the I/O drivers that comprise the I/O component of the operating system. Each I/O driver controls one or more hardware devices of similar type.

The network module controls interactions between the computer system and the network(s) to which it is attached.

Traditionally, the operating system software has nearly always been stored on a hard disk, but on some smaller modern systems, especially lightweight laptops and embedded systems such as cell phones, tablets, and e-books, a solid-state drive or SD card is normally used instead. On a few systems the operating system is actually provided as a network or cloud-based service when the system is turned on. In either case, the bootstrap or IPL (Initial Program Load) program in the operating system is stored within the computer using a type of memory known as **ROM**, or **read-only memory**. The bootstrap program provides the tools to test the system and to load the remainder of the operating system from the disk or network. Although the physical medium where the software is stored can be touched, the software itself is considered intangible.

Together, the hardware and system software provide a working computer system environment. Application software, communication support, and user data complete the picture.

The Communication Component

Very few modern computers or computer-based devices operate independently. Instead, they are usually designed to interact with other computers. To work together, computers must have means to communicate with each other through a communication connection of some sort. The computers may be located physically close to each other, or they may be separated, even by thousands of miles. The communication component requires both hardware and software to achieve this goal. Additional hardware components physically connect computers together into multiprocessing systems, or clusters, or networks, or, via telephone, satellite, microwave, Internet, or other network technology, to computers at other remote locations. A **communication channel** provides the connection between computers. The channel may be a wire cable, a fiber-optic cable, a telephone line, a wireless technology, or some combination of these. Examples of wireless technologies include infrared light, cellular phone, satellite, Wi-Fi, and Bluetooth. Special I/O hardware, consisting of a device such as a cellular **modem** or **network interface card (NIC)** within the computer, serves as an interface between the computer and the communication channel. There may be additional hardware within the channel itself.

The communication component also requires additional software within the operating system of each computer to make it possible for each computer to understand what the other computers that they are connected with are saying. This software establishes the connections, controls the flow of data, and directs the data to the proper applications for use.

The Computer System

To review, our general description of the computer is valid for all general-purpose computer systems, and also for most devices with computers embedded in them, regardless of brand name, usage, or size. In more general terms, every computer system consists of at least one CPU, where all the processing takes place; memory to hold the programs and data while they are being processed; and some form of I/O, usually one or more keyboards, touch screens, pointer devices, and flat-screen display devices. Computer systems also include one or more forms of long-term storage, usually hard disks or solid-state storage, but also

possibly network ("cloud") storage, optical disks, and USB or SD plug-in memory. Most modern computer systems provide more than one CPU (or "core") within the computer system. A single CPU can process only one instruction at a time; the use of multiple CPUs can increase processing speed by allowing instructions that do not affect each other to be executed in parallel.

The validity of our general description is true regardless of how complex or simple the computer system may seem.

As a specific example, the large z15 T01 IBM mainframe computer shown in Figure 1.10 can provide complex Web services to thousands of users at a time. IBM z15 mainframes can have as many as 190 CPUs working together, with a minimum of 512 GB up to 40 terabytes (TB) of primary storage. They are capable of executing instructions at rates of up to hundreds of billions of instructions per second! The powerful z/OS operating system can keep track of hundreds or thousands of simultaneous users and divides the time among them to satisfy their differing requirements. In addition to the CPU, there are many large I/O devices—including tape drives and high-speed printers—and disks and solid-state devices that store essentially unlimited amounts of data. The biggest version of this computer alone weighs over 6000 pounds/2800 kilograms!

In contrast, the Apple iPad tablet shown in Figure 1.11 is designed for personal use. Everything is self-contained in one hand-held package. This system only has 3 GB of primary RAM storage and operates at a small fraction of the speed of the z15. Long-term storage is limited to 128 GB of solid-state memory. The entire system, complete with display screen, built-in webcams, multiple network connections, including an optional cellular connection, and battery, weighs about 1.1 pounds (0.49 kilograms, if you prefer).

Although these two systems seem very different, the difference is actually one of magnitude and application, not of concept. The large system operates much faster, can support much more memory, and handles more input and output much faster. It has operating system software that allows many users to share this larger resource. Nonetheless, the fundamental system architecture is remarkably similar in both cases. Even the actual processing performed by the CPU is similar.

In fact, today's CPU operates in the same fundamental way as its CPU counterpart of sixty years ago, even though the construction is very different. Since computers all operate so similarly, regardless of size or type, it is not difficult today to transfer data between these different systems, allowing each system to do part of the processing for higher overall efficiency. This concept is known as **distributed computing**. The fact that different types of computers can work together, share files, and communicate successfully is known as **open computing**. Communication technology fulfills the requirements that make open and distributed computing possible and convenient.

FIGURE 1.10 IBM System z15 T01 Mainframe Computer

FIGURE 1.11
Apple iPad Tablet Computer

Computers are sometimes divided into categories: mainframe computers, midsized servers, workstations, personal desktop and laptop computers, and mobile computing devices, but these categories are less significant than they once were. The capability of today's smartphone far exceeds the capabilities of a mainframe computer of years ago. Indeed, it is not uncommon to see workstations that are being used as though they were midsized servers, or even small mainframes. Rather than attempting to categorize a particular computer, it is usually more productive to describe its capabilities in comparison to other systems being discussed or considered.

1.3 The Concept of Virtualization

The word "**virtual**" appears frequently throughout the computer literature in many different contexts. To name a few applications of the word that appear in this text, there are *virtual* computers, a Java *Virtual* Machine (JVM), *virtual* memory, and *virtual* networks. Sometimes, a synonymous word, **logical**, is used instead: in networking we have *logical* connections. *Virtual* storage consists of a relationship between *logical* memory and physical memory.

It is not important at this point that you understand any of the specific concepts mentioned above. (In fact, we realize that you probably don't.) Since the words *virtual* and *logical* represent a number of important concepts in IT, however, we introduce them here.

In optics, a virtual image is the reflection that you see when you stand in front of a regular mirror. (See, for example, the cartoon at the beginning of Chapter 19.) You know that the image isn't real. For one thing, it's behind the wall that the mirror is mounted on. For another, you can't touch it. Another example that might be familiar to you is *virtual reality*, the technology that allows people to see and interact with things that aren't there. In early, time-shared computing, a large central computer commonly supplied computing services to users at terminals located remotely from the computer. In a sense, it seemed as though the user had access to a computer that was all her own. Starting in the early 1970s, IBM offered the VM (*virtual machine*) operating system to support this concept. (The centralized time-sharing approach is similar, in some ways, to today's cloud computing—one of the goals of this text is to convince you that many of today's "new and exciting" technologies are simply reworkings of ideas that have been around for a long time!)

The *American Heritage Dictionary* offers two applicable definitions of *virtual* that together describe the usage of the word in modern computing:

- existing or resulting in essence or effect though not in actual fact, form, or name;
- created, simulated, or carried on by means of a computer or computer network.

In essence, *virtual* and *logical* are used to refer to something that appears as though it is something different. Thus, the *Java Virtual Machine* uses software to simulate a real computer that works well with the Java programming language, even though the actual computer executes a different set of instructions than the JVM. A *logical connection* in networking offers the appearance of a direct communication link for passing data between two computers, even though the actual connection might involve a complex series of interconnections involving many computers and other devices and a variety of software to make it all look simple. The *virtualization* of a computer allows a single computer to appear as a multiplicity of computers, each with its own operating system and hardware resources. A single mainframe set up as a cloud service might provide hundreds or thousands of virtual computers to users all over the world.

1.4 Protocols and Standards

Standards and protocols are of great importance in computer systems. **Standards** are agreements among interested parties, often manufacturers, to assure that various system components will work together interchangeably. Standards make it possible to build a computer with components from different suppliers, for example, knowing that a graphics card will plug properly into a connector on a motherboard and that the image representations will be consistent between the connector, the CPU, memory, and the display monitor.

Standards apply to every aspect of computing: hardware, software, data, and communications; the voltage of a power supply; the physical spacing of pins on a connector; the format of a file; and the pulses generated by a mouse. Computer language standards, such as Java and SQL, allow programs written on one type of computer to execute properly and consistently on another, and also make it possible for programmers to work together to create and maintain programs.

Similarly, data format and data presentation standards, such as the PNG and JPEG image format standards, the Unicode text format standard, and the HTML and XML Web presentation standards, allow different systems to manipulate and display data in a consistent manner.

Standards can arise in many different ways. Many standards occur naturally: a proprietary data format, belonging to a single vendor, becomes a *de facto* standard due to the popularity of the product. The PDF print description language is an example of such a standard. The format was designed by Adobe Corporation to provide a way of communicating high-quality printed output between computers and printers. Other standards are created because of a perceived need in an area where no standard exists.

Often a committee will form to investigate the requirements and create the standard. The MPEG-2 and MPEG-4 standards, which establish the means for the transmission and processing of digital video images, occurred in this way. The committee that designed the standard, made up primarily of motion picture engineers and video researchers, continues to develop the standard as improved techniques evolve. The JPEG photographic standard and MP3 and MP4 sound standards are other examples of standards that were developed formally. Similarly, each version of HTTP has been formalized after many years of discussion by parties interested in Web communication. A nonstandard protocol or data format is limited in use to its supporters and may or may not become a standard, depending on its general acceptance. For example, DVD videos encoded in the proprietary DVD-ROM format will play on some DVD players, but not on others.

Protocols define the specific agreed-upon sets of ground rules that make it possible for a communication to take place. Except for special applications, most computers perform their operations such that each hardware or software computer unit will understand what other

computer units that they are connected with are saying. Protocols exist for communications between computers, for the communications between various I/O devices and a computer, and for communications between many software programs. A protocol specification defines such communication features as data representation, signaling characteristics, message format, meanings of messages, identification and authentication, and error detection. Protocols in a client–server system assure that requests are understood and fulfilled and that responses are interpreted correctly.

Since the use of a proprietary protocol would be limited to those with permission to use it, protocols are almost always eventually standardized. Although not always the case, protocols that are not standardized tend to die out from lack of use. In fact, international standards are often created to ensure that the protocols are universally compatible. As an example, HTTP, Hypertext Transfer Protocol, guides communication between Web servers and Web browsers on the Internet. The movement of data through the Internet is controlled by a **suite** of protocols called TCP/IP (Transmission Control Protocol/Internet Protocol). Storage devices communicate with a computer using a protocol called SATA. There are thousands of such protocols.

New protocols and other standards are proposed and created and standardized as the need arises. XML, RSS, and SIP are all examples of protocols developed recently to meet new and changing demands. Satellite telecasting, near-universal telephone communication, wireless communications, and the Internet all demonstrate powerful and useful technologies made possible by protocols and standards. Indeed, the Internet is a measure of the success to which protocols that govern intercommunication between computer hardware and software have been standardized throughout the world. Discussions of various protocols and standards will occur regularly throughout this book.

1.5 Overview of This Book

The focus of this book is upon the architecture and organization of computers, computer systems, and computer-based IT systems, including everything from the smallest mobile device to the largest mainframe. Technically, there is a slight difference in definition between the terms “computer architecture” and “computer organization”. We will usually not attempt to differentiate these terms and will use them interchangeably.

In this book we will be concerned with all four components of computer systems: hardware, software, data, and interconnectivity, and with the interactions between each component, with other systems, and with users. We will also look initially at the larger picture: the organization of computer systems as components, themselves, to form enterprise IT systems. Chapter 2 of this first part is concerned with the system as a whole. The remainder of this book is divided into four additional parts, consisting of discussions of number systems and the representation of data in the computer, the hardware that makes up the computer, the networks that interconnect computers, and the system software that the computer uses.

Our first step will be to examine the concept of systems in general. We will look at the characteristics and qualities that define a system. We will then use that basic understanding to look at the characteristics of computer-based IT systems and show how the various elements and requirements of computer systems fit into the system concept. Part 1 introduces fundamental IT architecture concepts.

In Part 2, we will look at the different forms the input data may take, and we will consider the translation processes required to convert data into forms that the computer hardware and software can process. You will see how the various data types that are familiar to you from programming languages are stored and manipulated inside the computer. You’ll learn the many different ways in which math calculations can be performed, and the advantages

and disadvantages of each. You will see the difference between a number and the alphanumeric representation of a number, and understand why that difference can be critical in whether a program works or not. You will be able to relate the size of a word processing text to the storage capability of the computer's SSD or disk. You'll understand how computers process and handle multimedia data, graphics, photographs, audio, and video.

In Part 3, we will take a detailed look at the various components of the hardware and how they fit together. Using an extremely simple model, you will learn how the CPU works, how different I/O devices work, and even how text and graphics manage to appear, seemingly by magic, on the display screen. You will learn what makes some computers faster and more powerful than others, and what that means. You will learn about different ways of connecting I/O devices to the computer and see why you get a fast response from some devices, a slow response from others. You'll learn about USB ports.

Most importantly, you will have the opportunity to see what a simple, program-obedient machine the computer really is. You will learn about the limitations of a computer. We all tend to think of the computer as a resource of infinite capacity, speed, and perhaps even intelligence, but of course that's not true. We will consider how these limitations affect your work as a user, and as a means of specifying a system that will meet your needs and requirements.

Part 4 will provide a careful introduction to the foundational principles of communication and networking. We will consider the basic communication technologies, networking hardware, software, channels and channel media, protocols, and methodologies that are required to support communication between computer systems in an IT system environment. At the end of Part 4, in Chapter 15, we put the pieces together and look at the architecture of networked computer systems.

In the final part, we will consider the software that is used to control the computer's basic processing capabilities. Although computer software falls into two categories, operating system software and application software, we will focus exclusively on the system software. We will be concerned with control and efficient use of the computer hardware, fair and effective allocation of computer resources to different programs, security, storage management and file system structure, system administration, security, user interfaces, virtual machines, and more.

There are also four supplementary chapters covering topics that are somewhat outside the scope of the text, but important and interesting nonetheless. The first supplementary chapter introduces the fundamental logic that makes up a computer. The second supplementary chapter provides case studies that describe the hardware and system software of important real-world computer systems. These examples include the x86 family of PC hardware, the Microsoft Windows family of operating systems, Linux operating systems, and IBM mainframe hardware and software. The remaining two supplementary chapters, on CPU instruction addressing modes and on programming tools, have been maintained and updated from previous editions. The supplementary chapters can be found on the book's student companion resources website, at www.wiley.com/college/.

Additional related topics of current interest may also be found on the book's website.

1.6 A Brief Architectural History of the Computer

Although a study of the history of computing is generally outside the scope of this book, a brief introduction is useful in showing the wide-ranging and quirky path by which IT has arrived to its present position. It is of particular interest to note that nearly all of the revolutionary concepts that define computer systems today were developed between thirty- and eighty years ago; today's advances are more evolutionary and incremental in nature. Today's

smartphone processes instructions that are remarkably similar to those of mainframe computers in the 1960s. Some current cell and network technologies are based on inventions from World War II. This suggests that an understanding of the basic concepts that we are presenting in this book should serve you, the reader, well in your ability to understand the importance and significance of future developments as they occur.

Early Work

It is not possible, nor particularly useful, to identify the date of the "invention" of the computer. Indeed, it has always been the aspiration of humankind to create devices that would simplify people's work. Thus, it is not surprising that people were envisioning mechanical devices to simplify the jobs of routine data processing and calculation even in ancient times. In fact, there is recent evidence of the existence of an ancient computing device used for astronomical calculations. Instead, this discussion covers just a few of the major developments related to computer architecture.

In this context, one could consider the abacus, already in use as early as 500 BC by the ancient Greeks and Romans, to be an early predecessor of the computer. Certainly, the abacus was capable of performing calculations and storing data. Actually, if one were to build a binary numbered abacus, its calculations would very closely resemble those of the computer.

The abacus remained in common use until the 1500s and, in fact, is still considered an effective calculating tool in some locations today. In the late 1500s, though, European inventors again began to put their minds to the problem of automatic calculation. Blaise Pascal, a noted French mathematician of the 1600s, invented a calculating machine in 1642 at the age of nineteen, although he was never able to construct the machine. In 1801, Joseph Marie Jacquard invented a loom that used punched cards to control the patterns woven into cloth. The program provided by the punched cards controlled rods that raised and lowered different threads in the correct sequence to print a particular pattern. This is the first documented application of the use of some form of storage to hold a program for the use of a semiautomated, programmable machine.

Charles Babbage, an English mathematician who lived in the early 1800s, spent much of his own personal fortune attempting to build a mechanical calculating machine that he called an "analytical engine". The analytical engine resembles the modern computer in many conceptual ways. A photo of an early version of the analytical engine is shown in Figure 1.12. Babbage's machine envisioned the use of Jacquard's punched cards for input data and for the program, provided memory for internal storage, performed calculations as specified by the program using a central processing unit known as a "mill", and printed output. Augusta Ada Byron, Countess of Lovelace and the daughter of the poet Lord Byron, worked closely with Babbage and developed many of the fundamental ideas of programming and program design, including the concepts of branches and loops.

A block diagram of the Babbage analytical engine is shown in Figure 1.13. The mill was capable of selecting one of four arithmetic operations, and of testing the sign of a number with a different program branch specified for each result. The sequence of operation was specified by instructions on the operation cards. The operation cards could be advanced or reversed as a means of implementing a sort of "goto" instruction. The second set of cards, known as variable cards, were to be used to specify particular memory locations for the data involved in the calculations.

Babbage envisioned a memory of one thousand 50-digit decimal numbers. Each digit was to be stored using a ten-toothed gear known as a counter wheel. Although the analytical engine was never completed, it should be apparent to you that it contains all the essential elements of today's computers. At approximately the same time, another English

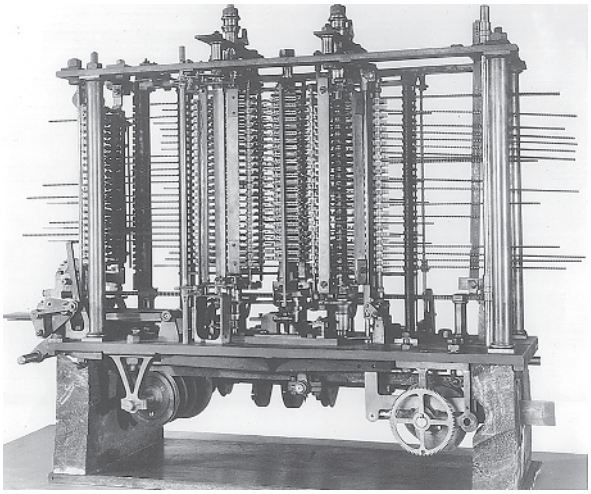


FIGURE 1.12
Babbage's Analytical Engine

mathematician, George Boole, developed the binary theory of logic that bears his name, Boolean logic. He also recognized the relationship between binary arithmetic and Boolean logic that makes possible the circuitry that implements the modern electronic computer.

Computer Hardware

In the late 1930s and early 1940s, several different groups of researchers independently developed versions of the modern electronic computer. The Mark I, built in 1937 by Howard H. Aiken and associates at Harvard University with help and funding from IBM, used thousands of mechanical relays; relays are binary switches controlled by electrical currents to perform Boolean logic. Although binary relays were used for computation, the fundamental design was decimal. Storage consisted of seventy-two 23-digit decimal numbers, stored on counter wheels. An additional counter wheel digit held the sign, using the digit 0 for plus and 9 for minus. The design appears to be based directly on Babbage's original concepts and use of mechanical calculator parts from IBM accounting machines. A similar electromechanical computer was designed and built by Conrad Zuse in Germany at about the same time.

The first totally electronic digital computer was apparently devised by John V. Atanasoff, a physicist at Iowa State College, in 1937. The machine was built in 1939 by Atanasoff and a graduate student, Clifford Berry, using electronic vacuum tubes as the switching components. The machine was known as ABC, for Atanasoff–Berry Computer. It is claimed that Atanasoff worked out the original details as he drove restlessly late one winter night from his house in Iowa to a bar in neighboring Illinois. The machine was not intended as a

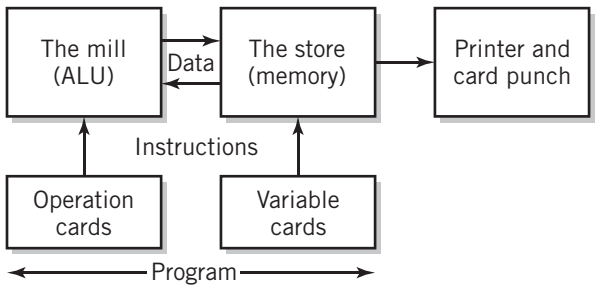


FIGURE 1.13
Block Diagram of Babbage's Analytical Engine
Source: Computer Architecture and Organization, 2e., J. Hayes, copyright 1988 by McGraw-Hill Companies p. 14.

general-purpose computer, but was built to solve physics equations that Atanasoff was working on at the time. There is some doubt as to whether the machine ever worked completely.

ABC was a binary-based machine, just like today's computers. It consisted of an ALU with thirty units that could do addition and subtraction, a rotating drum memory that held thirty binary numbers of 50 digits each, and punched card input. Each punched card held five 15-digit decimal numbers. These numbers were converted to binary as they entered the machine. Despite its limitations, ABC was an important pathmark that led to later significant advances in computer design. It is only recently that Atanasoff has begun to receive recognition for his achievement.

Much of the effort that culminated in a successful general-purpose computer architecture resulted from a wartime need for the solution to difficult mathematical formulas related to ballistic missile trajectories and other World War II research. The ENIAC (for Electronic Numerical Integrator and Computer, believe it or not) is generally considered to be the first all-electronic digital computer. It was designed and built between 1943 and 1946 by John W. Mauchly and J. Presper Eckert at the University of Pennsylvania, using the concepts that Mauchly had seen in Atanasoff's machine, although this was not publicly known at the time.

ENIAC had very limited storage capability, with only twenty locations each capable of holding a 10-digit decimal number. An additional one hundred numbers could be stored in read-only memory. Calculations were performed using decimal arithmetic. Ten electronic vacuum tube binary switches were used for each digit, with only one switch in the "ON" position to represent the value of the digit. Input and output used punched cards. The system could also provide printed output.

Programs could not be stored internally, but were hard wired with external "patch panels" and toggle switches. It took many hours to change programs, and, of course, debugging was a nightmare. Nonetheless, ENIAC was an important machine, some say the most important machine, especially since it led directly to the development of the UNIVAC I, the first commercially available computer, in 1951.

ENIAC contained eighteen thousand vacuum tubes, occupied a floor space of more than fifteen thousand square feet, and weighed more than thirty tons. A photograph of ENIAC, taken from *The New York Times* of February 15, 1946, is shown in Figure 1.14. Even in its day, ENIAC was recognized as an important achievement. ENIAC operated successfully until 1955, when it was dismantled, but not destroyed. Parts of the computer can be seen at

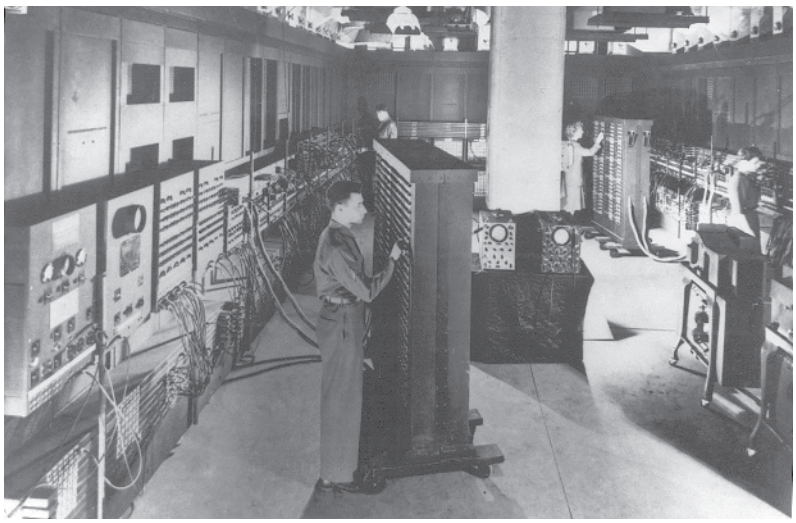


Photo used with permission of Unisys Corporation

FIGURE 1.14
The ENIAC

the Smithsonian Institute, at the U.S. Military Academy at West Point, at the Moore School of the University of Pennsylvania, and at the University of Michigan.

In 1945, John von Neumann, a consultant on the ENIAC project, proposed a computer that included a number of significant improvements over the ENIAC design. The most important of these were

1. A memory that would hold both programs and data, the so-called stored program concept. This solved the difficult problem of rewiring the control panels for changing programs on the ENIAC.
2. Binary processing of data. This simplified the design of the computer and allowed the use of binary memory for both instructions and data. It also recognized the natural relationship between the ON/OFF nature of switches and calculation in the binary number system, using Boolean logic.

The CPU was to include ALU, memory, and CU components. The control unit read instructions from memory and executed them. A method of handling I/O through the control unit was also established. The instruction set contained instructions representing all the essential features of a modern computer. In other words, von Neumann's machine contained every major feature considered essential to modern computer architecture. Modern computer architecture is still referred to as **von Neumann architecture**.

Due to political intrigue and controversy, two different versions of von Neumann's architecture were designed and built, EDVAC at the University of Pennsylvania and IAS at the Princeton University Institute for Advanced Studies (hence the unusual name). Both machines were completed in 1951–1952. The success of EDVAC and IAS led to the development of many offspring, mostly with odd names, and to several commercial computers, including the first IBM computers.

At this point, von Neumann's architecture was firmly established. It remains the prevalent standard to this day and provides the foundation for the remainder of the material in this book. Although there have been significant advances in technology, and improvements in design that have resulted, today's designs still reflect the work done prior to 1951 on ABC, ENIAC, EDVAC, and IAS.

All of these early electronic computers relied on the electronic vacuum tube for their operation. Vacuum tubes were bulky, made of glass, fragile, short-lived, and required large amounts of power to operate. Vacuum tubes require an internal electric heater to function, and the heaters tend to fail quickly, resulting in what was known as a "burned out" tube. Furthermore, the heat generated by the large number of tubes used in a computer required a massive forced-air or water-cooling system. A report reprinted by computer historian James Cortada [CORT87] states that the average error-free operating time for ENIAC was only 5.6 hours. Such bulky, maintenance-requiring systems could not have attained the prevalence that computers have in our society. The technological breakthrough that made possible today's small, sophisticated computers was the invention of the transistor and, subsequently, the integration of transistors and other electronic components with the development of the integrated circuit.

The invention of the integrated circuit led to smaller, faster, more powerful computers as well as a new, compact, inexpensive form of memory, RAM. Although many of these computers played an important role in the evolution of today's computers, two specific developments stand out from the rest: (1) development of the first widely accepted personal computer, by IBM in 1981–1982, and (2) design of the Intel 8008 microprocessor, predecessor to the x86 CPU family, in 1972. The impact of these two developments is felt to this day. Even smartphones and other mobile devices reflect these developments.

Companies have developed better ways of moving data between different parts of the computer, better ways of handling memory, better ways of managing display graphics, and

methods for increasing the speed of instruction execution. As we noted before, there is a lot more processing power in today's smallest mobile device than there was in the largest mainframe computer in the 1970s. Nonetheless, the basic architecture of today's machines is remarkably similar to that developed in the 1940s.

Operating Systems

Given how easy it is to communicate with computers today, it is hard to picture a time when the user had to do everything by hand, one step at a time. We take it for granted that we can type commands at a keyboard or move a mouse and launch programs, copy files, send text to a printer, and perform myriad other computer tasks. We power up and bootstrap our systems by pressing a switch.

It was not always this way. Early computers had no operating systems. The user, who was also the programmer, entered a program by setting it, one word at a time, with switches on the front panel, one switch per bit, or by plugging wires into a patch panel that resembled a cribbage board. Not a pleasant operation! Needless to say, early computers were single-user systems. Much of the computer's time was tied up with this primitive form of program and data entry. In fact, as late as the mid-1970s, there were still vendors producing computer systems with no operating system and computer hardware that was still bootstrapped by entering the bootstrap program, one instruction at a time into switches on the front panel of the computer.

The history of system software, particularly operating systems, is much less well defined than that of hardware. According to Cortada [CORT87],

> *Without more sophisticated operating systems, scientists would not have been able to take full advantage of the power of the transistor and later of the [microprocessor] chip in building the computers known today. Yet their contribution to the overall evolution of digital computers has been overlooked by historians of data processing.*

Part of the reason, undoubtedly, is that software evolved gradually, rather than as a series of important individually identifiable steps. The first operating systems and high-level programming languages appeared in the early 1950s, particularly associated with IBM and MIT, but with only a few exceptions, these efforts have not been associated with individual people or projects.

The need for operating system software came from the increasing computer power that resulted from the rapid development of new computers in the 1950s. Although the hardware architecture has not changed substantially since that time, improved technology has resulted in a continuum of ever-increasing computer capability that continues to this day. It has been necessary to continually modify and improve operating system architecture to take advantage of that power and make it available to the user. Computing has changed from single-user batch processing (where only a single user, with a single program, could access the machine at one time), to multiple-user batch job submission (where each user's "job" was submitted to the computer by an operator for sequential runs), to multiuser batch job execution (where the computer executed several jobs simultaneously, thereby keeping the CPU busy while I/O took place on another user's job), to multiuser online computing (where each user had direct access to the computer), to single-user interactive personal computing, to today's powerful interactive networked systems, with multitasking, easy-to-use touch screens and graphical interfaces, the ability to move data between applications, and near-instant access to other computers all over the world.

Each of these developments, plus various hardware developments—minicomputers, PCs, tablets and smartphones, new I/O devices, multimedia—have required additional operating system sophistication; in each case, designers have responded to the need.

The early computers were used primarily by scientists and engineers to solve technical problems. The next generation of computers, in the late 1950s, provided a punched card reader for input and a printer for output. Soon after, magnetic tape systems became available. The first "high-level" languages, primarily assembly language, then FORTRAN, made it possible to write programs in a language other than binary, and offline card punch machines allowed programmers to prepare their programs for entry without tying up the machine. ALGOL, COBOL, and Lisp followed shortly after. New technology improved the reliability of the computers. All these advances combined to make the computer system practical for business commercial use, especially for large businesses.

Still, these computers were single-user batch systems. Initially, users **submitted** the cards that they had prepared to the computer for execution. Later, separate, offline systems were developed that allowed the cards to be grouped together onto a magnetic tape for processing together. Programs were then submitted to the computer room in the form of **jobs**. A job consisted of one or more program card **decks**, together with the required **data decks** for each program. An output tape could also be used to support printing offline. As an example, Figure 1.15 shows a job that compiles and executes a FORTRAN program.

I/O routines were needed to operate the card readers, tape drives, and printers. The earliest operating systems consisted of just these I/O routines, but gradually operating systems evolved to perform other services. Computer time was very expensive, hundreds of dollars per minute, and in growing demand. To increase availability, control of the computer was placed in the hands of an operator, who fed the punched cards, mounted tapes, and generally tried to keep the system busy and efficient. The operating system provided a monitor that fed jobs to the system and supported the operator by notifying him or her of necessary actions, such as loading a new tape, setting switches on the panel, removing printout, and so on. As system demand increased, the monitor expanded to include accounting and simple, priority-based scheduling of jobs.

It is generally accepted that the first operating system was built by General Motors Research Laboratories in 1953–1954 for their IBM 701 computer. Other early systems included the FORTRAN Monitor System (FMS), IBSYS, and Share Operating System (SOS).[4]

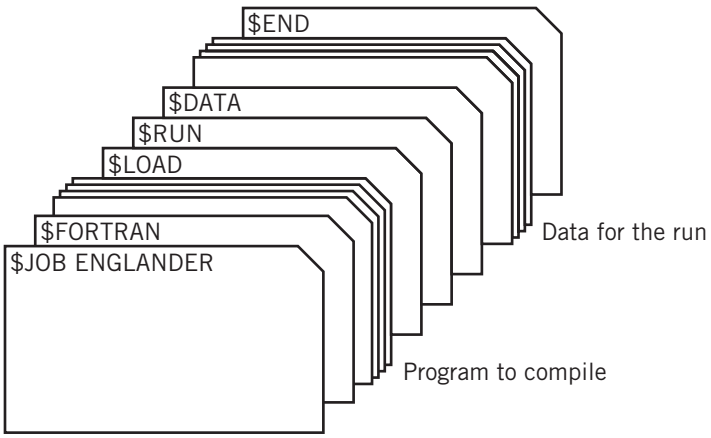


FIGURE 1.15 Job Card Deck Used to Compile and Execute a FORTRAN Program

[4]Share was a consortium of system programmers who used IBM systems and who met to discuss problems and develop solutions. SOS was produced by a team of consortium members.

Many important breakthroughs in operating system design occurred in the early 1960s. These breakthroughs laid the groundwork for the operating system as we know it today.

- In 1963, Burroughs released its Master Control Program (MCP). MCP contained many of the features of modern systems, including high-level language facilities and support for multiprocessing (with two identical CPUs). Most importantly, MCP supported virtual storage, as well as powerful multitasking capabilities.
- IBM introduced OS/360 as the operating system for its new System/360 in 1964. OS/360 provided a powerful language to expedite batch processing, JCL, or Job Control Language, and a simple form of multiprogramming that made it possible to load several jobs into memory, so that other jobs could use the CPU when one job was busy with I/O. By this time, disks were also becoming available, and the system was capable of reading cards onto disk while the CPU executed its jobs; thus, when a job completed, the operating system could load another job from disk into memory, ready to run. This improved the OS scheduling capability. JCL is still used for batch processing! The enormous success of the IBM OS/360 and its successors firmly established the basis of an operating system as a fundamental part of the computer.
- In 1962, a group at MIT known as Project MAC introduced the concept of time-sharing with an experimental operating system called CTSS. Project MAC was one of the seminal centers for the development of computer science. Shortly thereafter, MIT, Bell Labs, and GE formed a partnership to develop a major time-sharing system. The system was called MULTICS (Multiplexed Information and Computing Service), and although MULTICS never fully realized its dream of becoming a major computer utility, many of the most important multitasking concepts and algorithms were developed by the MULTICS team. It was supplied for many years as the operating system for Honeywell computer systems.
- When Bell Labs withdrew from the MULTICS project, Ken Thompson, a MULTICS researcher, turned to the development of a small personal operating system, which he called Unics, later UNIX, to contrast it from MULTICS. He was later joined by Dennis Ritchie. The original UNIX development was performed on a Digital PDP-7 minicomputer and later moved to a PDP-11 minicomputer, then to the Digital VAX computer. These were popular computer systems supplied by the Digital Equipment Corporation between 1964 and 1992. Originally, the system was written in assembly language, but Ritchie developed a new high-level language, which he called C, and the operating system was largely rewritten in C.

UNIX introduced many important OS concepts that are standard today, including the hierarchical file system, the shell concept, redirection, piping, and the use of simple commands that can be combined to perform powerful operations. Thompson and Ritchie included facilities for document production and formatting, including such novelties as a spell checker and a grammar checker. They created many inventive algorithms to improve operating system performance, developed techniques for interprocess communication, and even provided tools for networked and distributed processing. Many facets of operating systems that are taken for granted today were originated in UNIX development.

UNIX earned a reputation for power and flexibility. Because it was written in C, it was also easy to **port** it, that is, convert it for use, to other computers. As a result of these factors, UNIX became an important operating system for universities and was ultimately adopted, in many versions, by the commercial marketplace as well. UNIX and its direct derivatives,

FreeBSD, Linux, and Android, continue to be of great importance, particularly due to UNIX's flexibility in the area of networks and distributed systems.

- Another important innovation, some would say the most important development in making the computer accessible to nontechnical users, was the development of the concept of **graphical user interfaces**. Most historians would credit the invention of the windows and mouse interface to Doug Englebart. This work was done, amazingly enough, in the 1960s, at Stanford Research Institute. A practical windowing system was built in the 1970s by Alan Kay and others at Xerox PARC (Palo Alto Research Center), as part of a visionary computer concept known as the Dynabook project. Conceptually, the Dynabook is the direct forerunner of today's smartphones, tablets, and e-books. The original intention of Dynabook was to develop a book-sized personal computer with a high-resolution color display and wireless communication that would provide computer capabilities (particularly secretarial), games, e-mail, and a reference library. Although the technology of the time was not sufficient to bring the Dynabook as an entirety to fruition, the engineers at Xerox in the late 1970s built a personal computer workstation with a graphical user interface known as Star. It is believed that a visit to Xerox PARC by Steve Jobs, the founder of Apple, in 1979, inspired the development of the Apple Lisa and, subsequently, the Apple Macintosh.

The next important breakthrough in computer use occurred in 1982, with the introduction of the IBM personal computer. The IBM PC was designed as a stand-alone, single-user computer for the mass market. The IBM PC was supplied with a reasonably easy-to-use operating system, PC-DOS, which was developed and also later marketed by Microsoft as MS-DOS. PC-DOS was actually derived from an earlier personal computer operating system, CP/M (Control Program for Microcomputers), but is important because of the tremendous success of the IBM PC and its derivatives. Gradually, PC-DOS and MS-DOS became the prevalent operating system of the era. With later versions, Microsoft made many improvements, including hierarchical directory file storage, file redirection, better memory management, and an improved and expanded command set. Many of these improvements were derived from UNIX innovations. With the addition of Englebart and Kay's user interface innovations, MS-DOS gradually evolved into Windows, most recently, Windows 10.

Even with all these earlier innovations, there continue to be tremendous advances in operating system software. Today's systems, such as Windows 10, Linux, Android, and macOS and iOS, combine much more power on one hand with improved user friendliness and ease of use on the other. There are several reasons for this:

- There has been a great increase in computer speed and power. More powerful integrated circuits have allowed the design of faster computers using multiple CPU cores, faster clocks and larger internal data paths, together with techniques for speeding up instruction execution. Even small personal computers can support gigabytes of memory and many gigabytes or terabytes of longer-term storage. A modern PC may contain as much as two thousand times the memory or more and execute instructions a million times as fast as the 1965 IBM OS/360 mainframe computer. Thus, more capability can be built into the operating system without sacrificing performance.

- There have been fundamental improvements in computer hardware design. Many modern computers are designed as an integrated unit, hardware and operating system software together. Most computer hardware contains special features intended to support a powerful operating system. Such features as cache memory, vector processing, and virtual storage memory management hardware are intended primarily for use by the operating

system. These features used to be available only on large mainframes. A protected mode of hardware instructions, accessible only to the operating system, provides security and protection to the operating system and allows the operating system to protect the system's resources and users. Separate, auxiliary graphics processing units relieve the CPU workload to provide sophisticated display capabilities.

- There have been fundamental improvements in operating system software design. Operating system programs have grown in size and complexity. Increased memory capacity has made a larger operating system feasible. Increased speed has made it practical. Gradually, innovative operating system techniques from large computers have drifted down to the level of the smallest computing device. In addition, program design itself has helped the process. New languages, well designed for system programming, and better programming methods such as object-oriented programming have also contributed to the process.
- There has been a shift in focus to creating operating systems that better serve the end user. This has resulted in much current research on human–computer interfaces, and on the ways in which humans work and use the computer. New work paradigms, based on object-oriented programming and communication technologies, and new interfaces continue to extend the role of the operating system. There is a new willingness to include features that were not a part of earlier operating systems and to modularize the operating system in different ways to improve the delivery of services to the user and to the user's application programs.
- Networking has provided the opportunity for innovative research and development in distributed computing, including client–server technology, shared processing, and cloud computing. There is a continuing progression of new operating system techniques, developed in response to the changing requirements of modern distributed systems.
- The rapid growth of the Internet, and of e-mail use, the Web, and multimedia in particular, has created opportunities and the need for better methods of accessing, retrieving, and sharing information between different systems. The results have impacted network design, user interface design, distributed processing technology, and open system standardization with corresponding effects in operating system design.

Although today's operating systems are highly complex and sophisticated, with many capabilities made possible by modern technology, particularly fast processors, large amounts of memory, and improved graphical I/O design, it is interesting to note that the major operating system features that we take for granted today are all evolutions based on innovations of more than thirty years ago.

Communication, Networks, and the Internet

With the development of large, multiterminal computer systems in the 1960s and 1970s, it was natural that users would want to use the computer to communicate with each other and to work collaboratively. Data was centrally stored in storage that was available to all, so it was easily shared among users on the same system. It soon occurred to software developers that it would be desirable to allow direct discussion among the users, both in real time and in the form of messages that could be stored on the system and made available to users when they logged in. Since data was centrally stored, the addition of message storage was a minor enhancement. "Talk" facilities that allowed users to communicate in real time were added later. These were similar to today's text messaging, although some had split-screen capability

that allowed two users to send messages simultaneously. By 1965, some of these systems supported e-mail, and in 1971, Ray Tomlinson created the standard *username@hostname* format that is still in use today. As modems became available for users to log into their office systems from home and computers became more affordable, software innovators developed bulletin board systems, newsgroups, and discussion boards, where users could dial in and leave and retrieve messages. Gradually, it became possible to support modems on multiple lines, and affordable real-time "chat rooms" became possible.

During the same period, various developments occurred that made it possible to connect different computers together into simple networks. Some were based on direct links between modems on each computer. Others were based on early protocols, notably X.25, a packet-switching protocol using phone lines. By 1980, these various innovations had evolved into a number of international networks, as well as three companies, CompuServe, AOL, and Prodigy, who offered e-mail, Usenet news, chat rooms, and other services to personal computer users. (Ultimately, these developments have led to services such as Picasa, Facebook, twitter, Gmail, and cloud-based services such as Microsoft Office 365).

All of this activity was, of course, a precursor to the Internet. Much of the modern history of networking and communication can be traced back to two specific developments: (1) a research project, ARPANET, whose goal was to connect computers at various universities and research centers, funded starting in 1969 by the U.S. Defense Department and later by the National Science Foundation and other groups, and (2) the development of the Ethernet by Robert Metcalfe, David Boggs, and others, which started at Xerox PARC in 1973. The ARPANET project was responsible for the design of TCP/IP, which was first tested in 1974, and issued as an international standard in 1981. To give you perspective on the longevity of the basic computer concepts discussed in this book, we call to your attention the fact that, with one exception, this date, 1974, represents the *newest major* architectural concept presented in the next eighteen chapters! (Wondering about the exception? Keep reading.)

Because ARPANET and its successors, CSNet and NSFNet, were funded by the U.S. government, its use was initially limited to noncommercial activities. Gradually, other networks, some of them commercial, joined the network in order to exchange e-mails and other data, while the administrators of NSFNet chose to "look the other way". Ultimately, the government turned over its Internet resources to private interests in 1995; at that point the Internet became commercial and expanded rapidly into the form that we know today.

Although it is only peripherally related to the architectural issues addressed in this book, we would be remiss if we did not complete this discussion with a mention of Sir Tim Berners-Lee, of CERN, the European organization for nuclear research, who in 1989–1991 developed with Robert Cailliau the important concepts that became the World Wide Web and Marc Andreessen of the University of Illinois, who, in 1993, developed Mosaic, the first graphical Web browser.

SUMMARY AND REVIEW

This chapter has presented a brief review of the basics of computing. We began by recalling the input–process–output model for computing. Next, we demonstrated the connection between that model and the components of the computer system. We noted that implementation of the model requires four components: hardware, software, communication, and data. The architecture of the computer system is made up of the hardware and system software. In addition, a communication component exists to enable interconnecting systems. We discussed the general architecture of a computer and noted that the same description applies to CPUs both modern and ancient, both large and small. We introduced the

important concepts of virtualization, standards and protocols, noting that these ideas will appear throughout the book. The chapter concluded with a brief history of the computer from an architectural perspective.

FOR FURTHER READING

There are many good general introductory computer texts available for review if you feel you need one. New books appear so rapidly that we are reluctant to recommend any particular one. For alternative coverage of material in this book, you may find recent editions of various books by Stallings [e.g., STAL20] or Tanenbaum [e.g., TAN15] to be useful. Various chapters offer additional suggestions that are specifically applicable to the material in those chapters. The Web is also a rich source of knowledge. Two websites that we have found particularly useful are wikipedia.org and howstuffworks.org. In addition to a wide range of material, these websites also offer numerous references to facilitate further study. Other useful websites include arstechnica.com and realworldtech.com.

The book by Rochester and Gantz [ROCH83] is a fun way to explore the history of computing. Historical facts are blended with other facts, anecdotes, humor, and miscellany about computers. Although the book is (sadly) out of print, it is available in many libraries. You can learn, in this book, about von Neumann's party habits, about movies that became video games, about computer scams and rip-offs, and lots of other interesting stuff. Perhaps the most thorough discussion of computer history is found in the three-volume dictionary by Cortada [CORT87]. Although Cortada is not really designed for casual reading, it provides ready access and solid information on particular topics of interest. Much of the historical discussion in this chapter was obtained from the Cortada volumes.

If you live or vacation in a city with a computer museum, you can enjoy another approach to computer history. Computer museums even allow you to play with some of the older computers. Well-known U.S. museums can be found in Mountain View, CA, Washington, D.C., and within the Science Museum in Boston. The Wikipedia entry for computer museums offers pointers to lists of computer museums scattered through the world.

KEY CONCEPTS AND TERMS

application programming interface (API)
arithmetic/logic unit (ALU)
bus
byte
central processing unit (CPU)
channel (I/O)
communication channel
control unit (CU)
data deck
deck (program)
distributed computing
embedded computer
emulate
graphical user interface
hardware
input
input–process–output (IPO) model
interface unit
job
kernel
logical
memory
modem
network interface card (NIC)
open computing
operating system
output
port (from one computer to another)
primary storage
protocol
random access memory (RAM)
read-only memory (ROM)
software
standards
stored program concept
submit (a job)
suite (protocol)
virtual
von Neumann architecture
word

READING REVIEW QUESTIONS

1.1 Any computer system, large or small, can be represented by the four elements of an IPO model. Draw an IPO model; clearly label each of the four elements in your drawing.

1.2 One way to view an information technology system is to consider an IT system as consisting of four major components or building blocks. This book takes this approach by dividing the remainder of the book into parts, with a part devoted to each major type of component. What are the four components of an IT system that you will study in this book?

1.3 Explain the differences between primary storage and secondary storage. What is each type used for?

1.4 The book divides the software component of a computer system into two major categories. Identify each category and give an example of each that you are already familiar with. Briefly explain the role of each category.

1.5 The book compares a large mainframe computer to a smartphone or tablet, and states that the difference between them is one of magnitude, not of concept. Explain the meaning of that statement.

1.6 Virtualization is a concept that has taken on major importance in the early twenty-first century. Explain what is meant by virtualization.

1.7 What is a protocol? What is a standard? Do all protocols have to be standards? Explain. Are all standards protocols? Explain.

EXERCISES

1.1 Look at the computer ads on the business pages of a large daily newspaper and make a list of all the terms used that you don't understand. Save this list and check it from time to time during the semester. Cross out the items that you now understand and look up the items that have been covered but which you still don't understand.

1.2 For the computer that you normally use, identify which pieces constitute the hardware and which pieces constitute the system software. Now think about the file system of your computer. What part of the file system is hardware, what part software, and what part data?

1.3 Suppose you would like to buy a computer for your own needs. What are the major considerations and factors that would be important in your decision? What technical factors would influence your decision? Now try to lay out a specification for your machine. Consider and justify the features and options that you would like your machine to have.

1.4 Write a small program in your favorite high-level language. Compile your program. What is the ratio of high-level language statements to machine language statements? As a rough estimate, assume that each machine language statement requires approximately 4 bytes of file storage. Add various statements one at a time to your program and note the change in size of the corresponding machine language program.

1.5 Locate a current reference that lists the important protocols that are members of the TCP/IP protocol suite. Explain how each protocol contributes to the operation and use of the Internet.

1.6 Protocols and standards are an important feature of networks. Why is this so?

1.7 Although there is substantial overlap between protocols and standards there are protocols that are not standards and standards that are not protocols. With the help of a dictionary, identify the differences between the definition of protocol and the definition of standard; then, identify a specific example of a standard that is not a protocol; identify a specific example of a protocol that is not a standard.