

---

# Introduction to Computers in Medicine

*Willis J. Tompkins*

The field of computers in medicine is quite broad. We can only cover a small part of it in this book. We choose to emphasize the importance of real-time signal processing in medical instrumentation. This chapter discusses the nature of medical data, the general characteristics of a medical instrument, and the field of medicine itself. We then go on to review the history of the microprocessor-based system because of the importance of the microprocessor in the design of modern medical instruments. We then give some examples of medical instruments in which the microprocessor has played a key role and in some cases has even empowered us to develop new instruments that were not possible before. The chapter ends with a discussion of software design and the role of the personal computer in development of medical instruments.

## 1.1 CHARACTERISTICS OF MEDICAL DATA

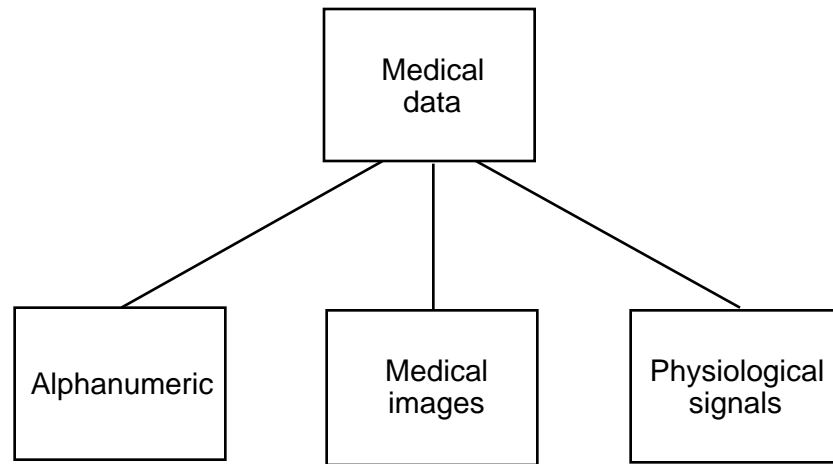
Figure 1.1 shows the three basic types of data that must be acquired, manipulated, and archived in the hospital. Alphanumeric data include the patient's name and address, identification number, results of lab tests, and physicians' notes. Images include Xrays and scans from computer tomography, magnetic resonance imaging, and ultrasound. Examples of physiological signals are the electrocardiogram (ECG), the electroencephalogram (EEG), and blood pressure tracings.

Quite different systems are necessary to manipulate each of these three types of data. Alphanumeric data are generally managed and organized into a database using a general-purpose mainframe computer.

Image data are traditionally archived on film. However, we are evolving toward picture archiving and communication systems (PACS) that will store images in digitized form on optical disks and distribute them on demand over a high-speed local area network (LAN) to very high resolution graphics display monitors located throughout a hospital.

On the other hand, physiological signals like those that are monitored during surgery in the operating room require real-time processing. The clinician must

know immediately if the instrument finds abnormal readings as it analyzes the continuous data.



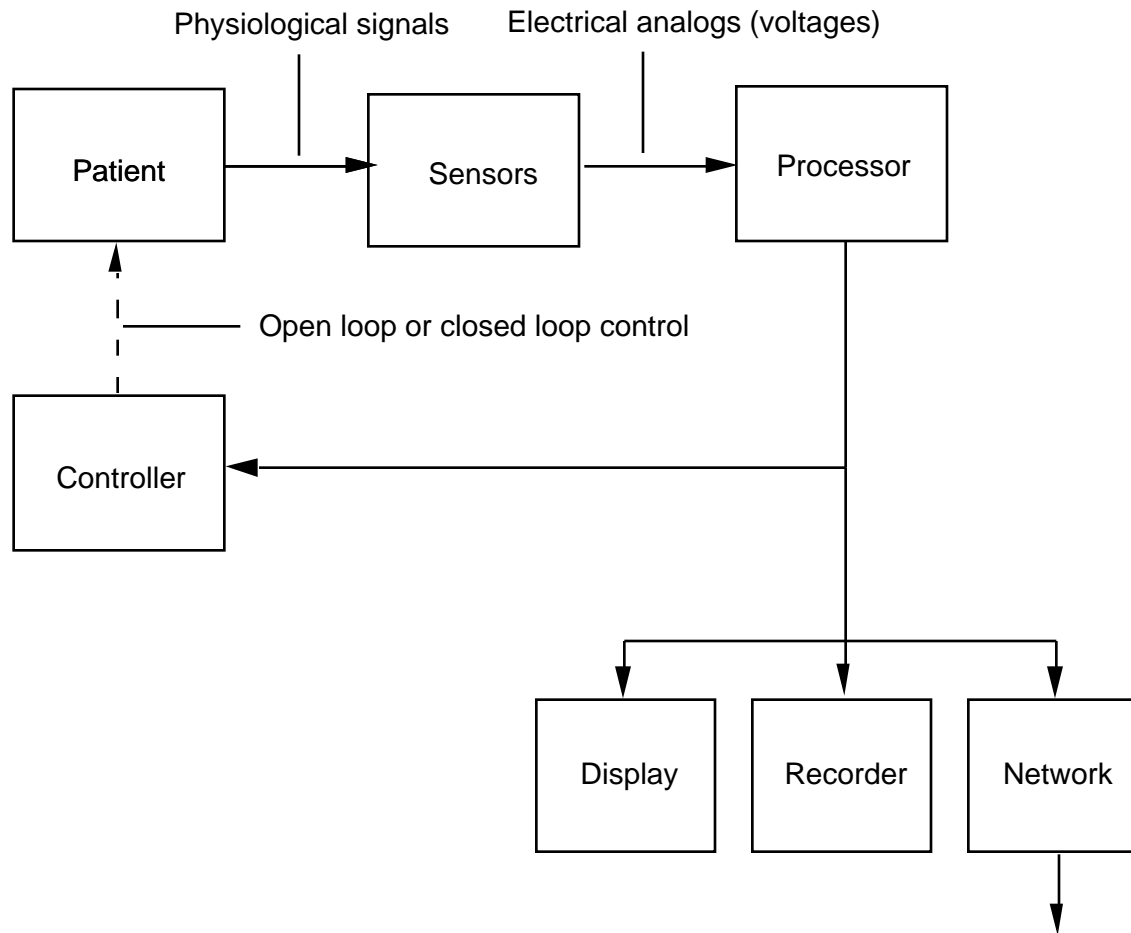
**Figure 1.1** Types of medical data.

It is this final type of data on which we concentrate in this book. One of the most monitored signals is the ECG, so we use it as the example signal to process in many examples.

## 1.2 WHAT IS A MEDICAL INSTRUMENT?

There are many different types of medical instruments. The ones on which we concentrate in this book are those that monitor and analyze physiological signals from a patient. Figure 1.2 shows a block diagram that characterizes such instruments. Sensors measure the patient's physiological signals and produce electrical signals (generally time-varying voltages) that are analogs of the actual signals.

A set of electrodes may be used to sense a potential difference on the body surface such as an ECG or EEG. Sensors of different types are available to transduce into voltages such variables as body core temperature and arterial blood pressure. The electrical signals produced by the sensors interface to a processor which is responsible for processing and analysis of the signals. The processor block typically includes a microprocessor for performing the necessary tasks. Many instruments have the ability to display, record, or distribute through a network either the raw signal captured by the processor or the results of its analysis. In some instruments, the processor performs a control function. Based on the results of signal analysis, the processor might instruct a controller to do direct therapeutic intervention on a patient (closed loop control) or it may signal a person that there is a problem that requires possible human intervention (open loop control).



**Figure 1.2** Basic elements of a medical instrumentation system.

Let us consider two types of medical instrumentation and see how they fit this block diagram. The first is an intensive care unit (ICU) system, a large set of instrumentation that monitors a number of patients simultaneously. The second is a cardiac pacemaker so small that it must fit inside the patient.

In the case of the ICU, there are normally several sensors connected to each patient receiving intensive care, and the processor (actually usually more than one processor) monitors and analyzes all of them. If the processor discovers an abnormality, it alerts the medical staff, usually with audible alarms. A display permits the staff to see raw data such as the ECG signals for each patient and also data obtained from the analysis such as numerical readouts of heart rate and blood pressure. The network connects the bedside portion of the instrumentation to a central console in the ICU. Another network might connect the ICU system to other databases remotely located in the hospital. An example of a closed loop device that is sometimes used is an infusion pump. Sensors monitor fluid loss as the amount of urine collected from the patient, then the processor instructs the pump to infuse the proper amount of fluid into the patient to maintain fluid balance, thereby acting as a therapeutic device.

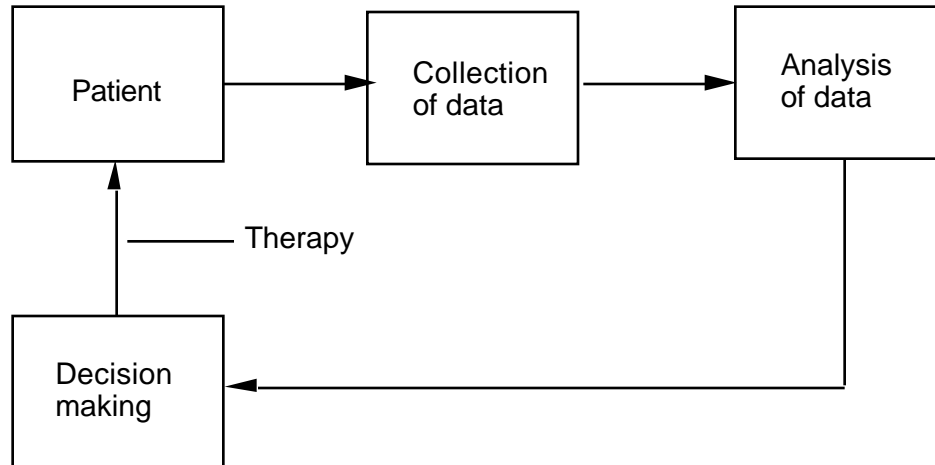
Now consider Figure 1.2 for the case of the implanted cardiac pacemaker. The sensors are electrodes mounted on a catheter that is placed inside the heart. The processor is usually a specialized integrated circuit designed specifically for this ultra-low-power application rather than a general-purpose microprocessor. The processor monitors the from the heart and analyzes it to determine if the heart is beating by itself. If it sees that the heart goes too long without its own stimulus signal, it fires an electrical stimulator (the controller in this case) to inject a large enough current through the same electrodes as those used for monitoring. This stimulus causes the heart to beat. Thus this device operates as a closed loop therapy delivery system. The early pacemakers operated in an open loop fashion, simply driving the heart at some fixed rate regardless of whether or not it was able to beat in a normal physiological pattern most of the time. These devices were far less satisfactory than their modern *intelligent* cousins. Normally a microprocessor-based device outside the body placed over a pacemaker can communicate with it through telemetry and then display and record its operating parameters. Such a device can also set new operating parameters such as amplitude of current stimulus. There are even versions of such devices that can communicate with a central clinic over the telephone network.

Thus, we see that the block diagram of a medical instrumentation system serves to characterize many medical care devices or systems.

### 1.3 ITERATIVE DEFINITION OF MEDICINE

Figure 1.3 is a block diagram that illustrates the operation of the medical care system. Data collection is the starting point in health care. The clinician asks the patient questions about medical history, records the ECG, and does blood tests and other tests in order to define the patient's problem. Of course medical instruments help in some aspects of this data collection process and even do some preprocessing of the data. Ultimately, the clinician analyzes the data collected and decides what is the basis of the patient's problem. This decision or diagnosis leads the clinician to prescribe a therapy. Once the therapy is administered to the patient, the process continues around the closed loop in the figure with more data collection and analysis until the patient's problem is gone.

The function of the medical instrument of Figure 1.2 thus appears to be a model of the medical care system itself.



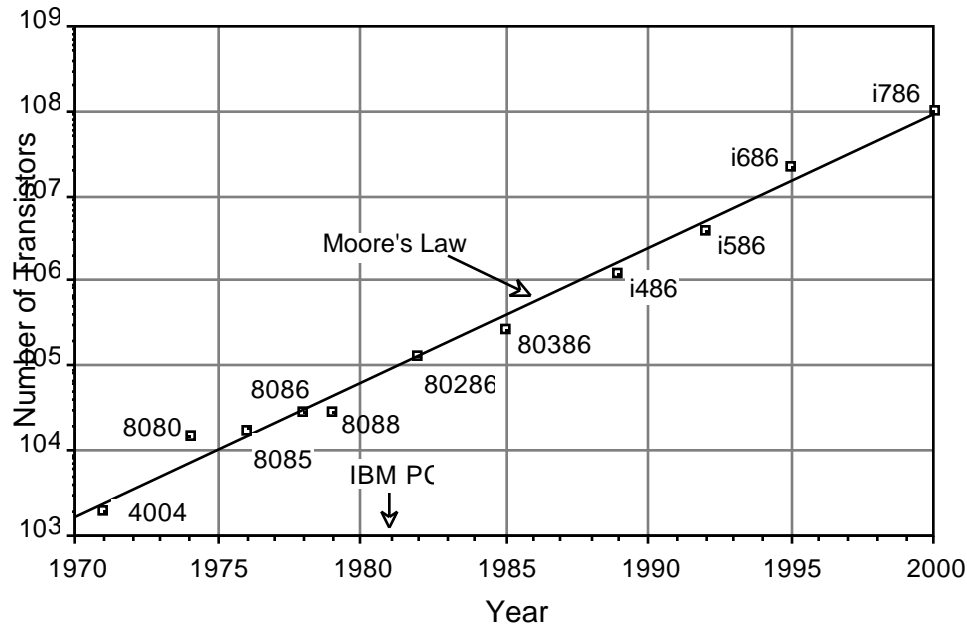
**Figure 1.3** Basic elements of a medical care system.

## 1.4 EVOLUTION OF MICROPROCESSOR-BASED SYSTEMS

In the last decade, the microcomputer has made a significant impact on the design of biomedical instrumentation. The natural evolution of the microcomputer-based instrument is toward more *intelligent* devices. More and more computing power and memory are being squeezed into smaller and smaller spaces. The commercialization of laptop PCs with significant computing power has accelerated the technology of the battery-powered, patient-worn portable instrument. Such an instrument can be truly a *personal* computer looking for problems specific to a given patient during the patient's daily routines. The ubiquitous PC itself evolved from minicomputers that were developed for the biomedical instrumentation laboratory, and the PC has become a powerful tool in biomedical computing applications. As we look to the future, we see the possibility of developing instruments to address problems that could not be previously approached because of considerations of size, cost, or power consumption.

The evolution of the microcomputer-based medical instrument has followed the evolution of the microprocessor itself (Tompkins and Webster, 1981). Figure 1.4 shows a plot of the number of transistors in Intel microprocessors as a function of time. The microprocessor is now more than 20 years old. It has evolved from modest beginnings as an integrated circuit with 2,000 transistors (Intel 4004) in 1971 to the powerful central processing units of today having more than 1,000,000 transistors (e.g., Intel i486 and Motorola 68040). One of the founders of Intel named Moore observed that the number of functional transistors that can be put on a single piece of silicon doubles about every two years. The solid line in the figure represents this observation, which is now known as Moore's Law. The figure shows that Intel's introduction of microprocessors, to date, has followed Moore's Law exceptionally well. The company has predicted that they will be able to continue producing microprocessors with this exponential growth in the number of transistors per microprocessor until at least the end of this century. Thus, in less than a decade, the microprocessor promises to become superpowerful as a parallel pro-

cessing device with 100 million transistors on one piece of silicon. It most likely will be more powerful than any of today's supercomputers, will certainly be part of a desktop computer, and possibly will be powerable by batteries so that it can be used in portable devices.



**Figure 1.4** The evolution of the microprocessor. The number of transistors in a microprocessor has increased exponentially throughout the history of the device. The trend is expected to continue into the future.

The evolution of the microprocessor from its early beginnings in 1971 as a primitive central processing unit to the powerful component of today has made a significant impact on the design of biomedical instrumentation. More computing power and memory are being squeezed into fewer integrated circuits to provide increasingly more powerful instruments. The PC itself has become a powerful tool in biomedical computing applications. In the future, we will be able to develop new medical instruments to address problems that were previously not solvable. This possibility exists because microprocessor-based systems continuously increase in computing power and memory while decreasing in size, cost, and power consumption.

### 1.4.1 Evolution of the personal computer

Figure 1.5 shows the history of the development of the computer from the first mechanical computers such as those built by Charles Babbage in the 1800s to the modern personal computers, the IBM PC and the Apple Macintosh. The only computers prior to the twentieth century were mechanical, based on gears and mechanical linkages.

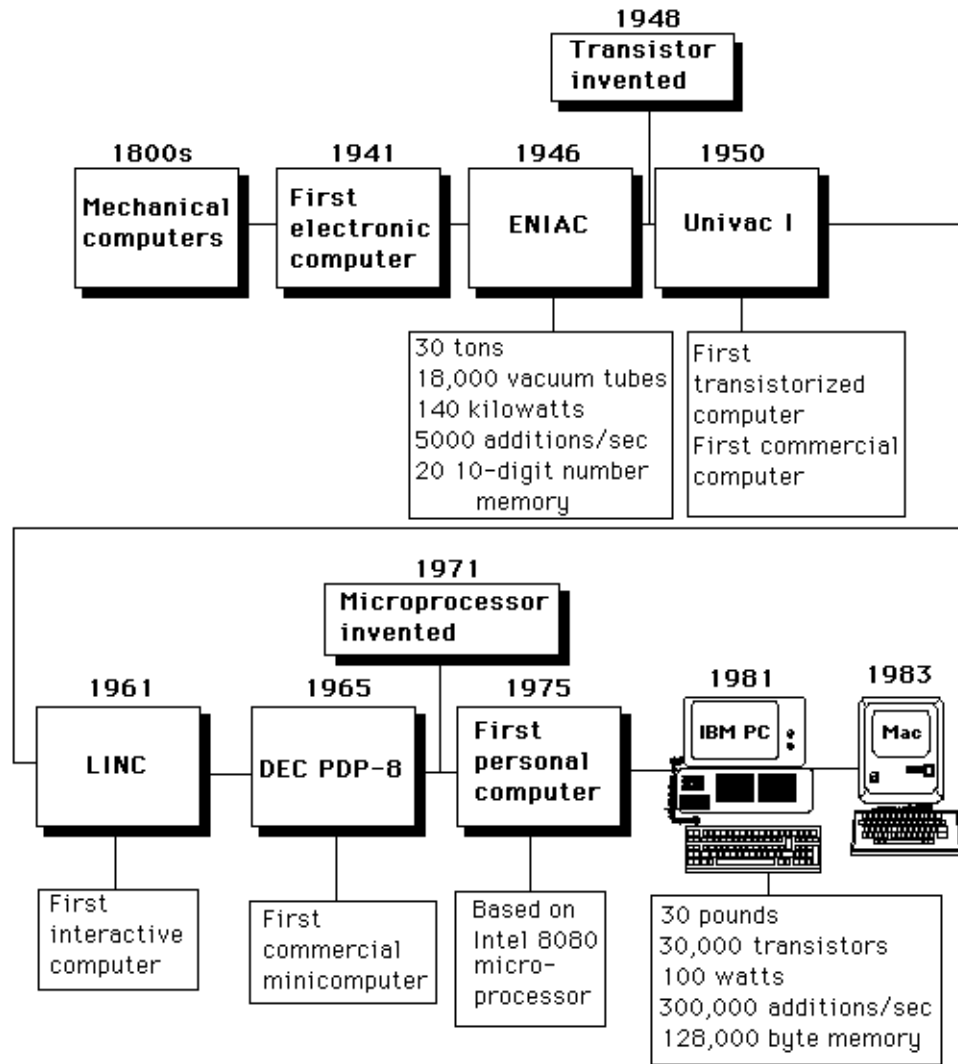
In 1941 a researcher named Atanasoff demonstrated the first example of an electronic digital computer. This device was primitive even compared to today's four-function pocket calculator. The first serious digital computer called ENIAC (**E**lectronic **N**umerical **I**ntegrator **A**nd **C**alculator) was developed in 1946 at the Moore School of Electrical Engineering of the University of Pennsylvania. Still simple compared to the modern PC, this device occupied most of the basement of the Moore School and required a substantial air conditioning system to cool the thousands of vacuum tubes in its electronic brain.

The invention of the transistor led to the Univac I, the first commercial computer. Several other companies including IBM subsequently put transistorized computers into the marketplace. In 1961, researchers working at Massachusetts Institute of Technology and Lincoln Labs used the technology of the time to build a novel minicomputer quite unlike the commercial machines. This discrete-component, transistorized minicomputer with magnetic core memory called the LINC (**L**aboratory **I**Nstrument **C**omputer) was the most significant historical development in the evolution of the PC.

The basic design goal was to transform a general-purpose computer into a laboratory instrument for biomedical computing applications. Such a computer, as its designers envisioned, would have tremendous versatility because its function as an instrument could be completely revised simply by changing the program stored in its memory. Thus this computer would perform not only in the classical computer sense as an equation solving device, but also by reprogramming (software), it would be able to mimic many other laboratory instruments.

The LINC was the most successful minicomputer used for biomedical applications. In addition, its design included features that we have come to expect in modern PCs. In particular, it was the world's first interactive computer. Instead of using punched cards like the other computers of the day, the LINC had a keyboard and a display so that the user could sit down and program it directly. This was the first digital computer that had an interactive graphics display and that incorporated knobs that were functionally equivalent to the modern joystick. It also had built-in signal conversion and instrument interfacing hardware, with a compact, reliable digital tape recorder, and with sound generation capability. You could capture an ECG directly from a patient and show the waveform on the graphics display.

The LINC would have been the first personal computer if it had been smaller (it was about the size of a large refrigerator) and less expensive (it cost about \$50,000 in kit form). It was the first game computer. Programmers wrote software for a two-player game called Spacewar. Each player controlled the velocity and direction of a spaceship by turning two knobs. Raising a switch fired a missile at the opposing ship. There were many other games such as pong and music that included an organ part from Bach as well as popular tunes.



**Figure 1.5** The evolution of the personal computer.

The LINC was followed by the world's first commercial minicomputer, which was also made of discrete components, the Digital Equipment Corporation PDP-8. Subsequently Digital made a commercial version of the LINC by combining the LINC architecture with the PDP-8 to make a LINC-8. Digital later introduced a more modern version of the LINC-8 called the PDP-12. These computers were phased out of Digital's product line some time in the early 1970s. I have a special fondness for the LINC machines since a LINC-8 was the first computer that I programmed that was interactive, could display graphics, and did not require the use of awkward media like punched cards or punched paper tape to program it. One of the programs that I wrote on the LINC-8 in the late 1960s computed and displayed the vectorcardiogram loops of patients (see Chapter 2). Such a program is easy to implement today on the modern PC using a high-level computing language such as Pascal or C.

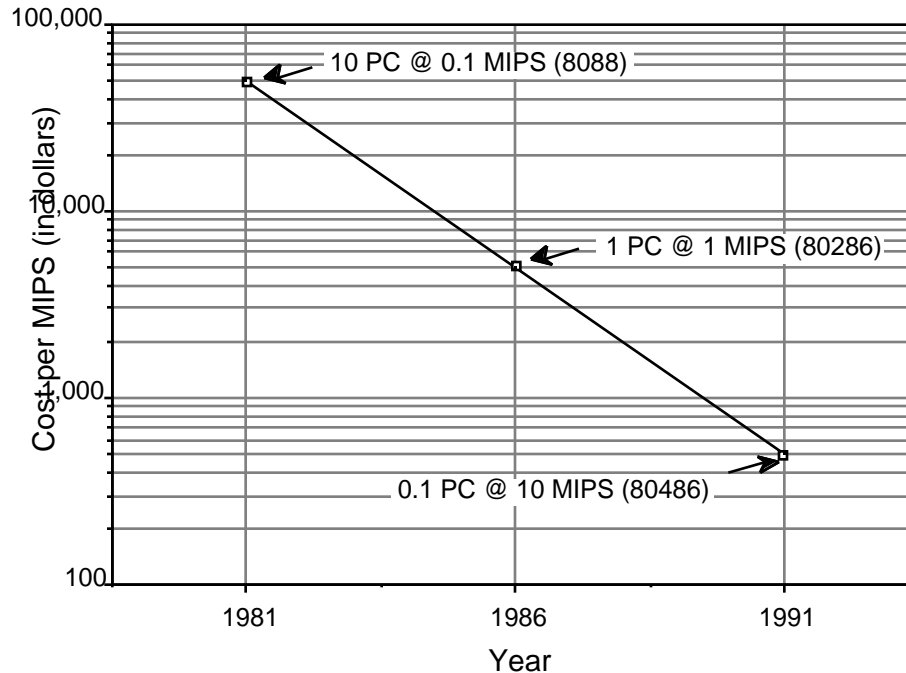
Although invented in 1971, the first microprocessors were poor central processing units and were relatively expensive. It was not until the mid-1970s when useful

8-bit microprocessors such as the Intel 8080 were readily available. The first advertised microcomputer for the home appeared on the cover of *Popular Electronics Magazine* in January 1975. Called the Altair 8800, it was based on the Intel 8080 microprocessor and could be purchased as a kit. The end of the decade was full of experimentation and new product development leading to the introduction of PCs like the Apple II and microcomputers from many other companies.

### 1.4.2 The ubiquitous PC

A significant historical landmark was the introduction of the IBM PC in 1981. On the strength of its name alone, IBM standardized the personal desktop computer. Prior to the IBM PC, the most popular computers used the 8-bit Zilog Z80 microprocessor (an enhancement of the Intel 8080) with an operating system called CP/M (Control Program for Microprocessors). There was no standard way to format a floppy disk, so it was difficult to transfer data from one company's PC to another. IBM singlehandedly standardized the world almost overnight on the 16-bit Intel 8088 microprocessor, Microsoft (Disk Operating System), and a uniform floppy disk format that could be used to carry data from machine to machine. They also stimulated worldwide production of IBM PC compatibles by many international companies. This provided a standard computing platform for which software developers could write programs. Since so many similar computers were built, inexpensive, powerful application programs became plentiful. This contrasted to the minicomputer marketplace where there are relatively few similar computers in the field, so a typical program is very expensive and the evolution of the software is relatively slow.

Figure 1.6 shows how the evolution of the microprocessor has improved the performance of desktop computers. The figure is based on the idea that a complete PC at any given time can be purchased for about \$5,000. For this cost, the number of MIPS (million instructions per second) increases with every new model because of the increasing power of the microprocessor. The first IBM PC introduced in 1981 used the Intel 8088 microprocessor, which provided 0.1 MIPS. Thus, it required 10 PCs at \$5,000 each or \$50,000 to provide one MIPS of computational power. With the introduction of the IBM PC/AT based on the Intel 80286 microprocessor, a single \$5,000 desktop computer provided one MIPS. The most recent IBM PS computers using the Intel i486 microprocessor deliver 10 MIPS for that same \$5,000. A basic observation is that the cost per MIPS of computing power is decreasing logarithmically in desktop computers.



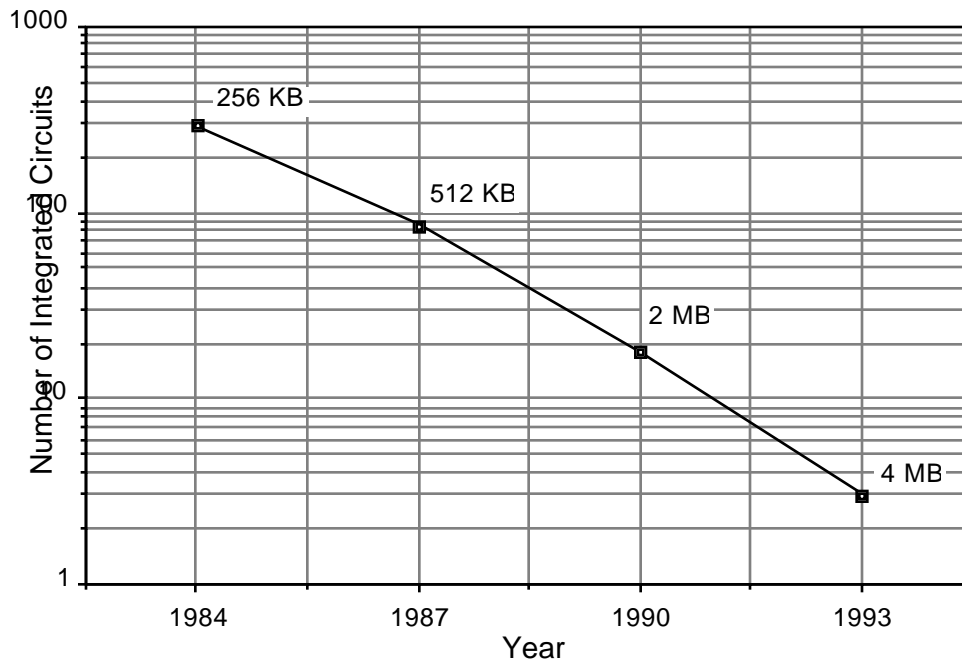
**Figure 1.6** The inverse relationship between computing power and cost for desktop computers. The cost per MIPS (million instructions per second) has decreased logarithmically since the introduction of the IBM PC.

Another important PC landmark was the introduction of the Apple Macintosh in 1983. This computer popularized a simple, intuitive user-to-machine interface. Since that time, there have been a number of attempts to implement similar types of graphical user interface (GUI) for the IBM PC platform, and only recently have practical solutions come close to reality.

More than a decade has elapsed since the introduction of the IBM PC, and most of the changes in the industry have been evolutionary. Desktop PCs have continued to evolve and improve with the evolution of the technology, particularly the microprocessor itself. We now have laptop and even palmtop PC compatibles that are portable and battery powered. We can carry around a significant amount of computing power wherever we go.

Figure 1.7 shows how the number of electronic components in a fully functional PC has decreased logarithmically with time and will continue to decrease in the future. In 1983, about 300 integrated circuits were required in each PC, of which about half were the microprocessor and support logic and the other half made up the 256-kbyte memory. Today half of the parts are still dedicated to each function, but a complete PC can be built with about 18 ICs. In the past six years, the chip count in a PC has gone from about 300 integrated circuits in a PC with a 256-kbyte memory to 18 ICs in a modern 2-Mbyte PC. By the mid-1990s, it is likely that a PC with 4-Mbyte memory will be built from three electronic components, a single IC for all the central processing, a read-only memory (ROM) for the basic input/output software (BIOS), and a 4-Mbyte dynamic random access memory (DRAM) chip for the user's program and data storage. Thus the continuing trend is

toward more powerful PCs with more memory in smaller space for lower cost and lower power consumption.



**Figure 1.7** The number of components in a PC continues to decrease while the computing performance increases.

In the 1970s, the principal microprocessors used in desktop computers as well as other systems including medical instruments were 8-bit microprocessors. The 1980s were dominated by the 16-bit microprocessors. The 1990s were launched with the 32-bit processor, but the technology's exponential growth will likely lead to useful new architectures on single ICs, such as parallel processors and artificial neural networks.

Figure 1.8 compares the modern PC with the human brain. The figure provides information that gives us insight into the relative strengths and weaknesses of the computer compared to the brain. From the ratios provided, we can clearly see that the personal computer is one or more orders of magnitude heavier, larger, and more power consuming than the human brain. The PC has several orders of magnitude fewer functional computing elements and memory cells in its "brain" than does the human brain.

---



---

**COMPARISON OF PERFORMANCE OF IBM PC AND HUMAN BRAIN**


---



---

System	Weight (lbs)	Size (ft <sup>3</sup> )	Power (watts)	CPU elements	Memory (bits)	Conduction rate (impulses/s)	Benchmark (additions/s)
IBM PC	30	5	200	10 <sup>6</sup> transistors (or equiv.)	10 <sup>7</sup>	10 <sup>5</sup>	10 <sup>6</sup>
Brain	3	0.05	10	10 <sup>10</sup> neurons	10 <sup>20</sup>	10 <sup>2</sup>	1
Ratio (PC/brain)	10	100	20	10 <sup>-4</sup>	10 <sup>-13</sup>	10 <sup>3</sup>	10 <sup>6</sup>

**Figure 1.8** The PC and the human brain each have their own characteristic strengths and weaknesses.

Then what good is it? The answer lies in the last two columns of the figure. The speed at which impulses are conducted in the computer so far exceeds the speed of movement of information within the brain that the PC has a very large computation speed advantage over the brain. This is illustrated by a benchmark which asks a human to add one plus one plus one and so on, reporting the sum after each addition. The PC can do this computational task about one million times faster than the human. If the PC is applied to tasks that exploit this advantage, it can significantly outperform the human.

Figure 1.9(a) is an array of numbers, the basic format in which all data must be placed before it can be manipulated by a computer. These numbers are equally spaced amplitude values for the electrocardiogram (ECG) of Figure 1.9(b). They were obtained by sampling the ECG at a rate of 200 samples per second with an analog-to-digital converter. This numerical representation of the ECG is called a digital signal. The human eye-brain system, after years of experience in learning the characteristics of these signals, is particularly good at analyzing the analog waveform itself and deducing whether such a signal is normal or abnormal.

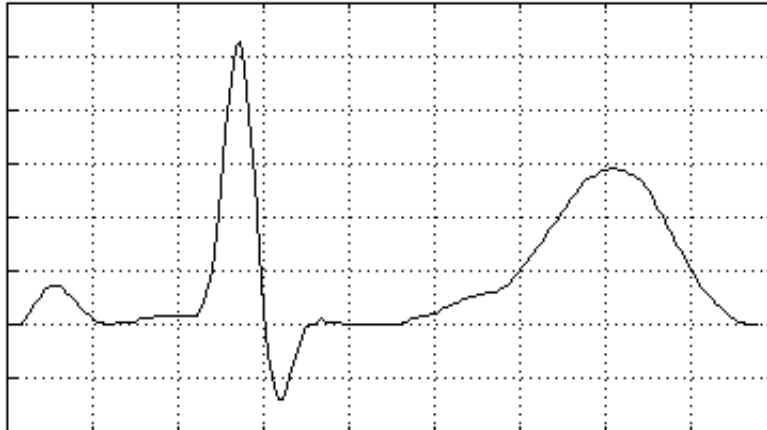
On the other hand, the computer must analyze the array of numbers using software algorithms in order to make deductions about the signal. These algorithms typically include digital signal processing, which is the emphasis of this book, together with decision logic in order to analyze biomedical signals as well as medical images. It is the enormous speed of the computer at manipulating numbers that makes many such algorithms possible.

```

0 0 0 2 5 8 10 13 14 14 14 12 11 9 7 5 4 2 1 1 0 0 1 1 1 1 2 2 2 3 3 3 3 3 3
3 3 6 11 20 33 51 72 91 103 105 96 77 53 27 5 -11 -23 -28 -28 -23 -17 -10 -5 -1
0 1 2 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 2 2 3 3 4 4 5 6 7 8 8 9 10 10 11 11 12 12
12 13 14 16 18 20 22 24 27 29 31 34 37 39 42 44 47 49 52 54 55 56 57 57 58 58
57 57 56 56 54 52 50 47 43 40 36 33 29 26 23 20 17 14 12 10 8 7 5 3 2 1 1 0 0 0

```

(a)



(b)

**Figure 1.9** Two views of an electrocardiogram. (a) The computer view is an array of numbers that represent amplitude values as a function of time. (b) The human view is a time-varying waveform.

## 1.5 THE MICROCOMPUTER-BASED MEDICAL INSTRUMENT

The progress in desktop and portable computing in the past decade has provided the means with the PC or customized microcomputer-based instrumentation to develop solutions to biomedical problems that could not be approached before. One of our personal interests has been the design of portable instruments that are light, compact, and battery powered (Tompkins, 1981). A typical instrument of this type is truly a personal computer since it is programmed to monitor signals from transducers or electrodes mounted on the person who is carrying it around.

### 1.5.1 Portable microcomputer-based instruments

One example of a portable device is the portable arrhythmia monitor which monitors a patient's electrocardiogram from chest electrodes and analyzes it in real time to determine if there are any heart rhythm abnormalities. We designed a prototype of such a device more than a decade ago (Tompkins, 1978). Because of the technology available at that time, this device was primitive compared with modern commercially available portable arrhythmia monitors. The evolution of the technology also permits us to think of even more extensions that we can make. Instead of just assigning a heart monitoring device to follow a patient after discharge from the hospital, we can now think of designing a device that would help diagnose the

heart abnormality when the patient arrives in the emergency room. With a careful design, the same device might go with the patient to monitor the cardiac problem during surgery in the operating room, continuously learning the unique characteristics of that patient's heart rhythms. The device could follow the patient throughout the hospital stay, alerting the hospital staff to possible problems in the intensive care unit, in the regular hospital room, and even in the hallways as the patient walks to the cafeteria. The device could then accompany the patient home, providing continuous monitoring that is not now practical to do, during the critical times following open heart surgery (Tompkins, 1988). Chapter 13 discusses the concept of a portable arrhythmia monitor in greater detail.

There are many other examples of portable biomedical instruments in the marketplace and in the research lab. One other microcomputer-based device that we contributed to developing is a calculator-size product called the CALTRAC that uses a miniature accelerometer to monitor the motion of the body. It then converts this activity measurement to the equivalent number of calories and displays the cumulative result on an LCD display (Doumas et al., 1982). There is now an implanted pacemaker that uses an accelerometer to measure the level of a patient's activity in order to adjust the pacing rate.

We have also developed a portable device that monitors several pressure channels from transducers on a catheter placed in the esophagus. It analyzes the signals for pressure changes characteristic of swallowing, then records these signals in its semiconductor memory for later transfer to an IBM PC where the data are further analyzed (Pfister et al., 1989).

Another portable device that we designed monitors pressure sensors placed in the shoes to determine the dynamic changes in pressure distribution under the foot for patients such as diabetics who have insensate feet (Mehta et al., 1989).

## 1.5.2 PC-based medical instruments

The economy of mass production has led to the use of the desktop PC as the central computer for many types of biomedical applications. Many companies use PCs for such applications as sampling and analyzing physiological signals, maintaining equipment databases in the clinical engineering department of hospitals, and simulation and modeling of physiological systems.

You can configure the PC to have user-friendly, interactive characteristics much like the LINC. This is an important aspect of computing in the biomedical laboratory. The difference is that the PC is a much more powerful computer in a smaller, less expensive box. Compared to the LINC of two decades ago, the PC has more than 100 times the computing power and 100 times the memory capacity in one-tenth the space for one-tenth the cost. However, the LINC gave us tremendous insight into what the PC should be like long before it was possible to build a personal computer.

We use the PC as a general-purpose laboratory tool to facilitate research on many biomedical computing problems. We can program it to execute an infinite variety of programs and adapt it for many applications by using custom hardware interfaces. For example, the PC is useful in rehabilitation engineering. We have designed a system for a blind person that converts visible images to tactile (touch)

images. The PC captures an image from a television camera and stores it in its memory. A program presents the image piece by piece to the blind person's fingertip by activating an array of tactors (i.e., devices that stimulate the sense of touch) that are pressed against his/her fingertip. In this way, we use the PC to study the ability of a blind person to "see" images with the sense of touch (Kaczmarek et al., 1985; Frisken-Gibson et al., 1987).

One of the applications that we developed based on an Apple Macintosh II computer is electrical impedance tomography—EIT (Yorkey et al., 1987; Woo et al., 1989; Hua et al., 1991). Instead of the destructive radiation used for the familiar computerized tomography techniques, we inject harmless high-frequency currents into the body through electrodes and measure the resistances to the flow of electricity at numerous electrode sites. This idea is based on the fact that body organs differ in the amount of resistance that they offer to electricity. This technology attempts to image the internal organs of the human body by measuring impedance through electrodes placed on the body surface.

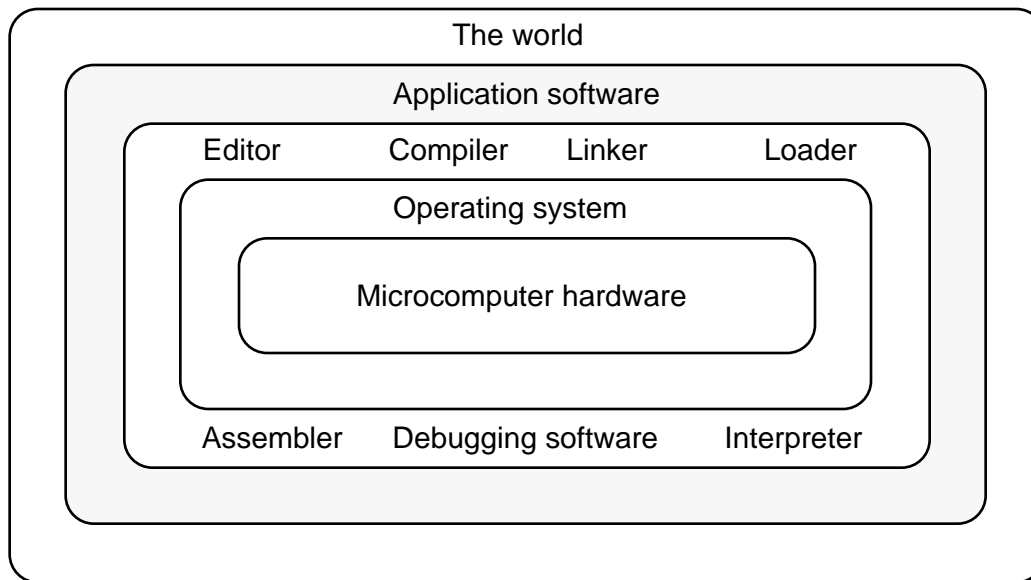
The computer controls a custom-built 32-channel current generator that injects patterns of high-frequency (50-kHz) currents into the body. The computer then samples the body surface voltage distribution resulting from these currents through an analog-to-digital converter. Using a finite element resistivity model of the thorax and the boundary measurements, the computer then iteratively calculates the resistivity profile that best satisfies the measured data. Using the standard graphics display capability of the computer, an image is then generated of the transverse body section resistivity. Since the lungs are high resistance compared to the heart and other body tissues, the resistivity image provides a depiction of the organ system in the body. In this project the Macintosh does all the instrumentation tasks including control of the injected currents, measurement of the resistivities, solving the computing-intensive algorithms, and presenting the graphical display of the final image.

There are many possible uses of PCs in medical instrumentation (Tompkins, 1986). We have used the IBM PC to develop signal processing and artificial neural network (ANN) algorithms for analysis of the electrocardiogram (Pan and Tompkins, 1985; Hamilton and Tompkins, 1986; Xue et al., 1992). These studies have also included development of techniques for data compression to reduce the amount of storage space required to save ECGs (Hamilton and Tompkins, 1991a, 1991b).

## 1.6 SOFTWARE DESIGN OF DIGITAL FILTERS

In addition to choosing a personal computer hardware system for laboratory use, we must make additional software choices. The types of choices are frequently closely related and limited by the set of options available for a specific hardware system. Figure 1.10 shows that there are three levels of software between the hardware and the real-world environment: the operating system, the support software, and the application software (the shaded layer). It is the application software that makes the computer behave as a medical instrument. Choices of

software at all levels significantly influence the kinds of applications that a system can address.



**Figure 1.10** Three levels of software separate a hardware microcomputer system from the real-world environment. They are the operating system, the support software, and the application software.

Two major software selections to be made are (1) choice of the disk operating system (DOS) to support the development task, and (2) choice of the language to implement the application. Although many different combinations of operating system and language are able to address the same types of applications, these choices frequently are critical since certain selections are clearly better than others for some types of applications. Of course these two choices are influenced significantly by the initial hardware selection, by personal biases, and by the user's level of expertise.

### 1.6.1 Disk operating systems

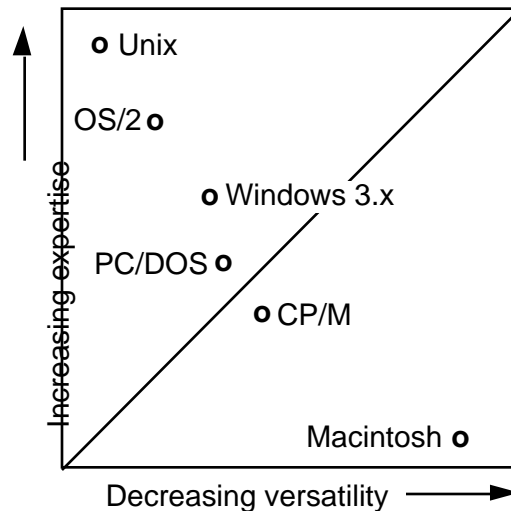
Our applications frequently involve software implementation of real-time signal processing algorithms, so we orient the discussions around this area. Real-time means different things to different people in computing. For our applications, consider real-time computing to be what is required of video arcade game machines. The microcomputer that serves as the central processing unit of the game machine must do all its computing and produce its results in a time frame that appears to the user to be instantaneous. The game would be far less fun if, each time you fired a missile, the processor required a minute or two to determine the missile's trajectory and establish whether or not it had collided with an enemy spacecraft.

A typical example of the need for real-time processing in biomedical computing is in the analysis of electrocardiograms in the intensive care unit of the hospital. In the typical television medical drama, the ailing patient is connected to a monitor that beeps every time the heart beats. If the monitor's microcomputer required a minute or two to do the complex pattern recognition required to recognize each valid heartbeat and then beeped a minute or so after the actual event, the device would be useless. The challenge in real-time computing is to develop programs to implement procedures (algorithms) that appear to occur instantaneously (actually a given task may take several milliseconds).

One DOS criterion to consider in the real-time environment is the compromise between flexibility and usability. Figure 1.11 is a plot illustrating this compromise for several general-purpose microcomputer DOSs that are potentially useful in developing solutions to many types of problems including real-time applications. As the axes are labeled, the most user-friendly, flexible DOS possible would plot at the origin. Any DOS with an optimal compromise between usability and flexibility would plot on the 45-degree line.

A DOS like Unix has a position on the left side of the graph because it is very flexible, thereby permitting the user to do any task characteristic of an operating system. That is, it provides the capability to maximally manipulate a hardware/software system with excellent control of input/output and other facilities. It also provides for multiple simultaneous users to do multiple simultaneous tasks (i.e., it is a multiuser, multitasking operating system). Because of this great flexibility, Unix requires considerable expertise to use all of its capabilities. Therefore it plots high on the graph.

On the other hand, the Macintosh is a hardware/software DOS designed for ease of use and for graphics-oriented applications. Developers of the Macintosh implemented the best user-to-machine interface that they could conceive of by sacrificing a great deal of the direct user control of the hardware system. The concept was to produce a personal computer that would be optimal for running application programs, not a computer to be used for writing new application programs. In fact Apple intended that the Lisa would be the development system for creating new Macintosh programs.



**Figure 1.11** Disk operating systems—the compromise between DOS versatility and user expertise in real-time applications.

Without training, an individual can sit down and quickly learn to use a Macintosh because its operation is, by design, intuitive. The Macintosh DOS plots low and to the right because it is very user-friendly but cannot necessarily be used by the owner to solve any generalized problem. In fact, the Macintosh was the first personal computer in history that was sold without any language as part of the initial package. When the Macintosh was first introduced, no language was available for it, not even the ubiquitous BASIC that came free with almost every other computer at the time, including the original IBM PC.

The 8-bit operating system, CP/M (Control Program/Microprocessors), was a popular operating systems because it fell near the compromise line and represented a reasonable mixture of ease of use and flexibility. Also it could be implemented with limited memory and disk storage capability. CP/M was the most popular operating system on personal computers based on 8-bit microprocessors such as the Zilog Z80. PC DOS (or the generic MS DOS) was modeled after CP/M to fall near the compromise line. It became the most-used operating system on 16-bit personal computers, such as the IBM PC and its clones, that are based on the Intel 8086/8088 microprocessor or other 80x86 family members.

On the other hand, Unix is not popular on PCs because it requires a great deal of memory and hard disk storage to achieve its versatility. The latest Unix look-alike operating systems for the IBM PC and the Macintosh typically require significant memory and many megabytes of hard disk storage. This is compared to CP/M on an 8-bit system that normally was implemented in less than 20 kbytes of storage space and PC/DOS on an IBM PC that requires less than 100 kbytes.

At this writing, Unix is the workstation operating system of choice. For many applications, it may end up to be the DOS of choice. Indeed, Unix or a close clone of it may ultimately provide the most accepted answer to the problem of linking PCs together through a local area network (LAN). By its very design, Unix provides multitasking, a feature necessary for LAN implementation. Incidentally, the

fact that Unix is written in the C language gives it extraordinary transportability, facilitating its implementation on computers ranging from PCs to supercomputers.

For real-time digital filtering applications, Unix is not desirable because of its overhead compared to PC/DOS. In order to simultaneously serve multiple tasks, it must use up some computational speed. For the typical real-time problem, there is no significant speed to spare for this overhead. Real-time digital signal processing requires a single-user operating system. You must be able to extract the maximal performance from the computer and be able to manipulate its lowest level resources such as the hardware interrupt structure.

The trends for the future will be toward Macintosh-type operating systems such as Windows and OS/2. These user-friendly systems sacrifice a good part of the generalized computing power to the human-to-machine interface. Each DOS will be optimized for its intended application area and will be useful primarily for that area. Fully implemented versions of OS/2 will most likely require such large portions of the computing resources that they will have similar liabilities to those of Unix in the real-time digital signal processing environment.

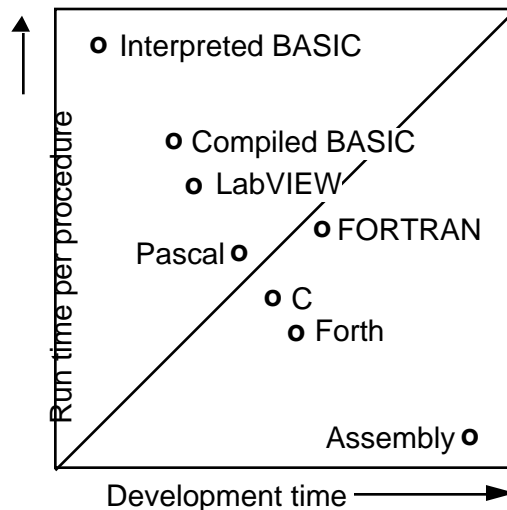
Unfortunately there are no popular operating systems available that are specifically designed for such real-time applications as digital signal processing. A typical DOS is designed to serve the largest possible user base; that is, to be as general purpose as possible. In an IBM PC, the DOS is mated to firmware in the ROM BIOS (Basic Input/Output System) to provide a general, orderly way to access the system hardware. Use of high-level language calls to the BIOS to do a task such as display of graphics reduces software development time because assembly language is not required to deal directly with the specific integrated circuits that control the graphics. A program developed with high-level BIOS calls can also be easily transported to other computers with similar resources. However, the BIOS firmware is general purpose and has some inefficiencies. For example, to improve the speed of graphics refresh of the screen, you can use assembly language to bypass the BIOS and write directly to the display memory. However this speed comes at the cost of added development time and loss of transportability.

Of course, computers like the NEXT computer are attempting to address some of these issues. For example, the NEXT has a special shell for Unix designed to make it more user-friendly. It also includes a built-in digital signal processing (DSP) to facilitate implementation of signal processing applications.

### 1.6.2 Languages

Figure 1.12 shows a plot of the time required to write an application program as a function of run-time speed. This again is plotted for the case of real-time applications such as digital signal processing. The best language for this application area would plot at the origin since this point represents a program with the greatest run-time speed and the shortest development time. The diagonal line maps the best compromise language in terms of the run-time speed compared to the software design time necessary to implement an application. Of course there are other considerations for choosing a language, such as development cost and size of memory space available in an instrument. Also most of the languages plotted will not, by

themselves, solve the majority of real-time problems, especially of the signal processing type.



**Figure 1.12** Languages—the compromise between development time and run-time speed in real-time applications.

The time to complete a procedure at run time is generally quite long for interpreted languages (that is, they are computationally slow). These interpreted languages like interpreted BASIC (**B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode) are generally not useful for real-time instrumentation applications, and they plot a great distance from the diagonal line.

The assembly language of a microprocessor is the language that can extract the greatest run-time performance because it provides for direct manipulation of the architecture of the processor. However it is also the most difficult language for writing programs, so it plots far from the optimal language line. Frequently, we must resort to this language in high-performance applications.

Other high-level languages such as FORTRAN and Pascal plot near the diagonal indicating that they are good compromises in terms of the trade-off between program development time and run time per procedure but do not usually produce code with enough run-time speed for real-time signal processing applications. Many applications are currently implemented by combining one of these languages with assembly language routines. FORTRAN was developed for solving equations (i.e., **FOR**mula **TRAN**slation) and Pascal was designed for teaching students structured programming techniques.

After several years of experience with a new microprocessor, software development companies are able to produce enhanced products. For example, modern versions of Pascal compilers developed for PCs have a much higher performance-to-price ratio than any Pascal compiler produced more than a decade ago.

Forth is useful for real-time applications, but it is a nontraditional, stack-oriented language so different from other programming languages that it takes some time for a person to become a skilled programmer. Documentation of programs is also

difficult due to the flexibility of the language. Thus, a program developed in Forth typically is a one-person program. However, there are several small versions of the Forth compiler built into the same chip with a microprocessor. These implementations promote its use particularly for controller applications.

LabVIEW (National Instruments) is a visual computing language available only for the Macintosh that is optimized for laboratory applications. Programming is accomplished by interconnecting functional blocks (i.e., icons) that represent processes such as Fourier spectrum analysis or instrument simulators (i.e., virtual instruments). Thus, unlike traditional programming achieved by typing command statements, LabVIEW programming is purely graphical, a block diagram language. Although it is a relatively fast compiled language, LabVIEW is not optimized for real-time applications; its strengths lie particularly in the ability to acquire and process data in the laboratory environment.

The C language, which is used to develop modern versions of the Unix operating system, provides a significant improvement over assembly language for implementing most applications (Kernighan and Ritchie, 1978). It is the current language of choice for real-time programming. It is an excellent compromise between a low-level assembly language and a high-level language. C is standardized and structured. There are now several versions of commercial C++ compilers available for producing object-oriented software.

C programs are based on functions that can be evolved independently of one another and put together to implement an application. These functions are to software what black boxes are to hardware. If their I/O properties are carefully specified in advance, functions can be developed by many different software designers working on different aspects of the same project. These functions can then be linked together to implement the software design of a system.

Most important of all, C programs are transportable. By design, a program developed in C on one type of processor can be relatively easily transported to another. Embedded machine-specific functions such as those written in assembly language can be separated out and rewritten in the native code of a new architecture to which the program has been transported.

## **1.7 A LOOK TO THE FUTURE**

As the microprocessor and its parent semiconductor technologies continue to evolve, the resulting devices will stimulate the development of many new types of medical instruments. We cannot even conceive of some of the possible applications now, because we cannot easily accept and start designing for the significant advances that will be made in computing in the next decade. With the 100-million-transistor microprocessor will come personal supercomputing. Only futurists can contemplate ways that we individually will be able to exploit such computing power. Even the nature of the microprocessor as we now know it might change more toward the architecture of the artificial neural network, which would lead to a whole new set of pattern recognition applications that may be more readily solvable than with today's microprocessors.

The choices of a laboratory computer, an operating system, and a language for a task must be done carefully. The IBM-compatible PC has emerged as a clear computer choice because of its widespread acceptance in the marketplace. The fact that so many PCs have been sold has produced many choices of hardware add-ons developed by numerous companies and also a wide diversity of software application programs and compilers. By default, IBM produced not only a hardware standard but also the clear-cut choice of the PC DOS operating system for the first decade of the use of this hardware. Although there are other choices now, DOS is still alive. Many will choose to continue using DOS for some time to come, adding to it a graphical user interface (GUI) such as that provided by Windows (Microsoft).

This leaves only the choice of a suitable language for your application area. My choice for biomedical instrumentation applications is C. In my view this is a clearly superior language for real-time computing, for instrumentation software design, and for other biomedical computing applications.

The hardware/software flexibility of the PC is permitting us to do research in areas that were previously too difficult, too expensive, or simply impossible. We have come a long way in biomedical computing since those innovators put together that first PC-like LINC almost three decades ago. Expect the PC and its descendants to stimulate truly amazing accomplishments in biomedical research in the next decade.

## 1.8 REFERENCES

- Doumas, T. A., Tompkins, W. J., and Webster, J. G. 1982. An automatic calorie measuring device. *IEEE Frontiers of Eng. in Health Care*, **4**: 149–51.
- Frisken-Gibson, S., Bach-y-Rita, P., Tompkins, W. J., and Webster, J. G. 1987. A 64-solenoid, 4-level fingertip search display for the blind. *IEEE Trans. Biomed. Eng.*, **BME-34**(12): 963–65.
- Hamilton, P. S., and Tompkins, W. J. 1986. Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database. *IEEE Trans. Biomed. Eng.*, **BME-33**(12): 1157–65.
- Hua, P., Woo, E. J., Webster, J. G., and Tompkins, W. J. 1991. Iterative reconstruction methods using regularization and optimal current patterns in electrical impedance tomography. *IEEE Trans. Medical Imaging*, **10**(4): 621–28.
- Kaczmarek, K., Bach-y-Rita, P., Tompkins, W. J., and Webster, J. G. 1985. A tactile vision substitution system for the blind: computer-controlled partial image sequencing. *IEEE Trans. Biomed. Eng.*, **BME-32**(8):602–08.
- Kernighan, B. W., and Ritchie, D. M. 1978. *The C programming language*. Englewood Cliffs, NJ: Prentice Hall.
- Mehta, D., Tompkins, W. J., Webster, J. G., and Wertsch, J. J. 1989. Analysis of foot pressure waveforms. *Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1487–88.
- Pan, J. and Tompkins, W. J. 1985. A real-time QRS detection algorithm. *IEEE Trans. Biomed. Eng.*, **BME-32**(3): 230–36.
- Pfister, C., Harrison, M. A., Hamilton, J. W., Tompkins, W. J., and Webster, J. G. 1989. Development of a 3-channel, 24-h ambulatory esophageal pressure monitor. *IEEE Trans. Biomed. Eng.*, **BME-36**(4): 487–90.
- Tompkins, W. J. 1978. A portable microcomputer-based system for biomedical applications. *Biomed. Sci. Instrum.*, **14**: 61–66.

- Tompkins, W. J. 1981. Portable microcomputer-based instrumentation. In H. S. Eden and M. Eden (eds.) *Microcomputers in Patient Care*. Park Ridge, NJ: Noyes Medical Publications, pp. 174–81.
- Tompkins, W. J. 1985. Digital filter design using interactive graphics on the Macintosh. *Proc. of IEEE EMBS Annual Conf.*, pp. 722–26.
- Tompkins, W. J. 1986. Biomedical computing using personal computers. *IEEE Engineering in Medicine and Biology Magazine*, 5(3): 61–64.
- Tompkins, W. J. 1988. Ambulatory monitoring. In J. G. Webster (ed.) *Encyclopedia of Medical Devices and Instrumentation*. New York: John Wiley, 1:20–28.
- Tompkins, W. J. and Webster, J. G. (eds.) 1981. *Design of Microcomputer-based Medical Instrumentation*. Englewood Cliffs, NJ: Prentice Hall.
- Woo, E. J., Hua, P., Tompkins, W. J., and Webster, J. G. 1989. 32-electrode electrical impedance tomograph – software design and static images. *Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 455–56.
- Xue, Q. Z., Hu, Y. H. and Tompkins, W. J. 1992. Neural-network-based adaptive matched filtering for QRS detection. *IEEE Trans. Biomed. Eng.*, **BME-39**(4): 317–29.
- Yorkey, T., Webster, J. G., and Tompkins, W. J. 1987. Comparing reconstruction algorithms for electrical impedance tomography. *IEEE Trans. Biomed. Eng.*, **BME-34**(11):843–52.

## 1.9 STUDY QUESTIONS

- 1.1 Compare operating systems for support in developing real-time programs. Explain the relative advantages and disadvantages of each for this type of application.
- 1.2 Explain the differences between interpreted, compiled, and integrated-environment compiled languages. Give examples of each type.
- 1.3 List two advantages of the C language for real-time instrumentation applications. Explain why they are important.