An Introduction to Computer Technology

Richard F. Walters, PhD
Davis, California

A few short years ago physicians were vaguely aware that computers were "somewhere" in the hospital, dealing with billing and other administrative details, but doing nothing pertinent to patient care. Today, nearly every physician has attended at least one professional meeting in which the use of computers *by physicians* was a major topic.

How did this all happen so fast? What brought the mysterious and somewhat frightening world of computers into the daily practice of medicine, causing health professionals to feel behind the times if they do not know the difference between a bit and a byte? To answer these questions it is necessary to go back into the short, but amazing, history of computers. The original computers have evolved into many species, some of them already extinct, and new genetic strains are now on the horizon. The ways one can communicate with computers must be examined, and then, perhaps, it will be possible to decide how (not whether; that decision has been made) physicians will use computers.

The purpose of this article is to trace the history of computer development and to make some projections into the future. Applications or the true medical impact will not be covered, but in describing computers as they exist today, the conclusion is obvious: computers will play a major role in health care delivery. The value of that contribution depends on the direct involvement of physicians in the process.

Families of Computers—Past, Present, and Future

It is unfortunate that the English language uses the word *computer* to describe the machines under discussion. The French call them *ordinateurs,* and somehow the logical ordering of information seems more appropriate a description than the strict definition of calculation, for today's computer is far more than a calculating machine.

Man has used aids for counting since he first started to look at the world around him. Fingers, sticks, knotted ropes—the history of counting aids is rich and varied.[1] The abacus was the first counting aid so well conceived that it has lasted for centuries, even (in the hands of trained users) outperforming many electronic gadgets of the early post-World War II days.

Viewed from this perspective, today's computers represent the use of electronics as just another aid to man's counting problems. Electric circuitry, currently the most used aid to information processing, may itself be replaced or dramatically altered by other technologies now in their embryonic state, for there are already two other disciplines (optics and bioelectronics) that offer real potential for additional dramatic improvements in speed, miniaturization, and complexity. Nevertheless, the world has been so profoundly affected by the rapid evolution of both electronics and electronic computers that the current potential of this technology remains a constantly moving target. Because of these advances the health profession today must learn to make use of the computer.

The class Computer evolved from early analog devices such as the slide rule and early mechanical devices, such as the abacus, Pascal's mechanical calculating machine, and the Jacquard loom. Although designs that would extend these tools to perform more complex functions have been around for over a century, it was not until electronic circuitry was available that the ideas became reality. The earliest electricity-based computers were created in the 1940s, using several different approaches, Of the various alternatives, the vacuum tube computer soon was recognized as the most promising, and with the invention of the transistor to replace the vacuum tube, reliability considerations made the computer a viable commercial product in the 1950s. Analog computers,

Dr. Walters is Professor and Acting Chair of the Division of Computer Science, College of Engineering, University of California, Davis, Davis, California. He has developed a public domain version of MicroMUMPS for IBM-PC and CP/M systems.

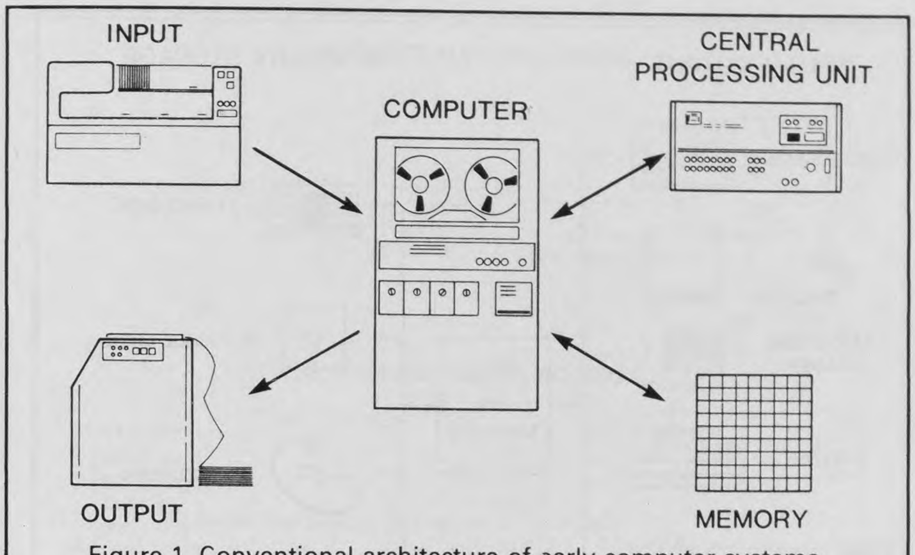

Figure 1. Conventional architecture of early computer systems

which used voltage levels instead of binary states to solve complex differential equations, were used in scientific computing until the early 1970s primarily because of their speed in solving differential equations. Unfortunately, because these computers, like the slide rule, used analog techniques, they could not produce the numeric precision required for present-day computation, and thus became obsolete.

Digital computers evolved at rates that defy human imagination. The earliest digital computers added at the rate of three additions per second back in 1945. By 1952, with the introduction of the transistor, computers could do 3,000 additions per second. With the discovery of integrated circuits (LSI—large-scale integrated circuits, and even VLSI—very large scale integrated circuits), the speed of computation rose another thousandfold, so that by the mid-1960s computers capable of doing 3 million additions per second were produced.

Since then, additional increases in speed have occurred, perhaps reducing addition time another hundredfold, but physical limitations (such as the speed of light) place insurmountable barriers to the achievement of maintaining the rate of improvement shown in the first two decades of computing history. Instead, new approaches to computing are being researched; these changes may well result in the continued dramatic rise in throughput capability. To understand these techniques, it is necessary first to understand the architecture of computers, then trace the evolution of different components of that architecture in the same way that hardware was considered.

Computer Architecture

The Jacquard loom, which revolutionized weaving around 1810, used punched (wooden) cards to control the pattern of threads in a fabric. These cards were fed into the loom in a predetermined sequence, producing repeatable and reliable designs. The first computers were fed with similar instructions, from the outside, producing consistent, repeatable results. However, in the 1940s, Von Neumann, drawing on ideas first proposed over a hundred years earlier by Charles Babbage (now considered the "father" of modern computing), suggested that instructions could be stored inside the computer. To do so required that the computer be constructed of several components (Figure 1): a central processing unit that would do the actual calculations, a memory unit that would store both instructions and data, and devices to communicate with the computer, reading in data and instructions and producing readable output.

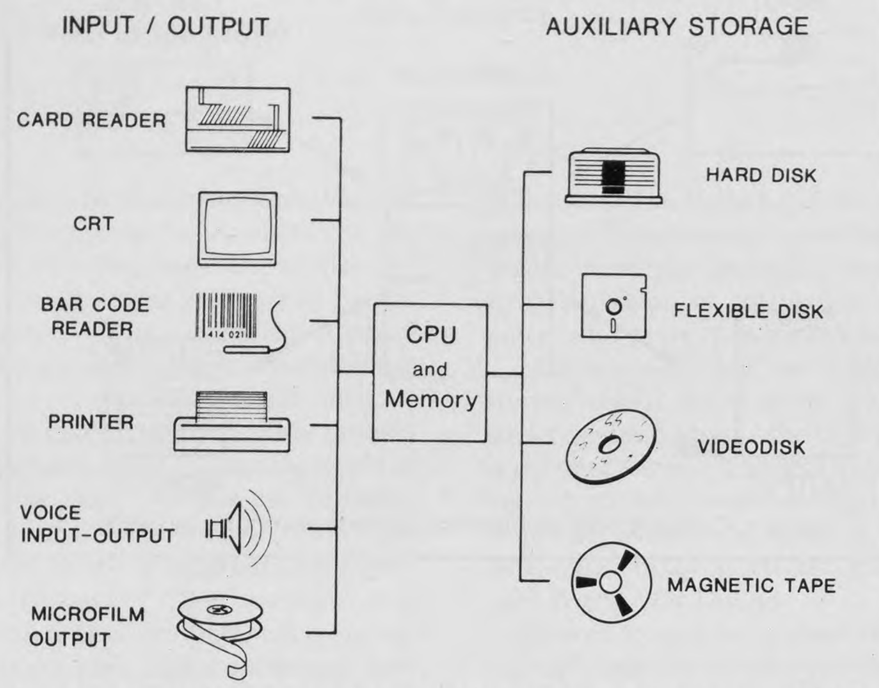

Figure 2. Extended architecture showing options for input/output and auxiliary storage

The four principal components of computers today are, therefore, input, output (I/O), central processing unit (CPU), and memory. This architecture has been slightly modified (Figure 2) to include multiple types of I/O, and to allow for auxiliary storage devices (magnetic tapes, disks, etc) that can be used for the long-term storage of information and to augment memory of the original architecture.

The notion that instructions, as well as data, can be stored in a computer led to the equally profound concept that a sequence of instructions need not necessarily always follow exactly the same steps. In other words, one could have a computer instruction to "test" the value of a number and another instruction to branch to a different instruction if the test met some condition. From this deceptively simple extension to the instruction set of the early computers arose the science of software, the effective use of these and other instructions to solve problems of the real world. It became possible to create programs with infinite variation in the tasks they would perform depending on the information they processed during execution. The so-called expert systems in diagnostic medicine depend ultimately on the test-and-branch instructions available in all computers today.

Software

In the early days, computers were given instruction in the form of codes that were interpreted by the computer and performed one at a time. The "programming" of the computer required a knowledge of the binary code for each instruction as well as the physical location for each data element (Figure 3A). This approach, while simpler for the computer, was tedious for the programmer and led, inevitably, to errors that were difficult to detect and correct.

To simplify the coding process, a set of mnemonic codes written as alphabetic letters (sometimes with numbers) was developed. Each code corresponded to one machine instruction. A special "program" was written to help the pro-

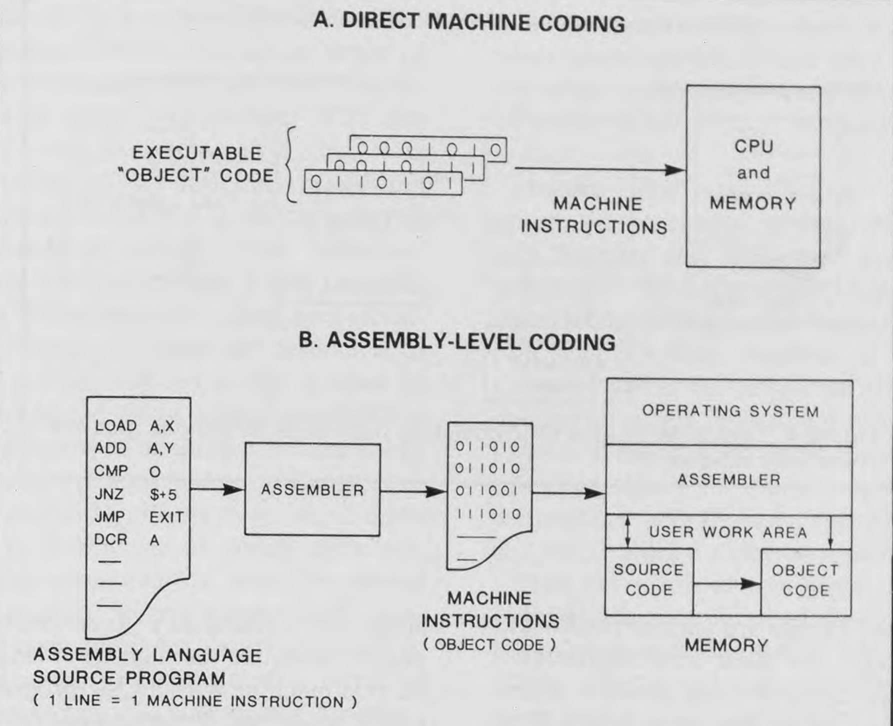

Figure 3. Role of assemblers and operating systems in translating source code to machine instructions

grammer by translating these codes into their machine instruction equivalents, "assembling" them into a block of instructions that would perform the appropriate functions (Figure 3B). These translating programs were called *assemblers,* and there was a different assembler for each computer, which meant that a programmer had to rewrite each application program for each new machine. Furthermore, because each instruction represented only one machine instruction, assembly language programs were lengthy, requiring detailed understanding of each computer's instruction set to achieve effective code.

The addition of an assembler to the computer meant that control of the instructions within a computer had to be handled by yet another program—an operating system. No longer could the programmer flip switches on the front of the computer and branch to a specific set of instructions. Instead, he had to request the operating system to call the assembler into memory, read his new program, and process it, looking for errors and creating the final instructions that could then be invoked by the programmer (again through the operating system) to perform the desired task.

With the appearance of the operating system, computers could do many more things than were previously possible. Since the internal speed of computers vastly exceeds anything in the outside world, means had to be found to use that excess speed when external communications were involved. Operating systems were therefore developed to permit the computer to appear to perform more than one task at a time. One obvious way to speed things up was to assign long printing tasks to a background status that could continue while another program was executing in the "foreground." Idle pieces of time resulting from foreground I/O were snatched by the operating system and devoted to the task of sending out a few characters to the printer, enough in most cases to keep the printer going at maximum capacity.

As programs grew more complex and computers increased in speed, a need arose for a higher

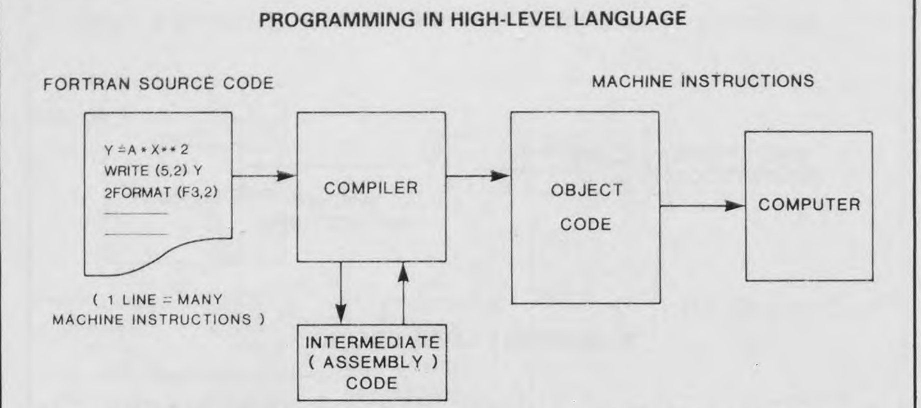

Figure 4. Typical technique for translating high-level languages into machine instructions

level of abstraction. People are used to thinking in terms of total tasks, not their small subcomponents; of sentences, not individual phrases. A programmer would like to be able to say, in effect, "divide the value found in location A by B, and store the results in location A." This type of statement translates into a great many assembly-level instructions. A must be fetched, B must be fetched, the division must be performed (itself requiring many instructions in most assembly languages), and the result stored where the original value of A resided. How much nicer to be able to say, for example, SET A = A/B, and not have to worry about the exact instructions used by a given machine to perform the task.

To help programmers rise to this level of abstraction, a new class of high-level languages was created, with COBOL (common business-oriented language) and then FORTRAN (formula translator) being two of the best known in the early days (they are still around) (Figure 4). A high-level language needs another crutch from the operating system: a special program that will either compile the program, creating first assembly instructions and then executable code, or interpret the program, executing the instructions as they are encountered without storing any executable code. With the advent of high-level languages, the task of programming became one that professionals in other disciplines (yes, even physicians) could learn. No longer was it necessary to learn each machine's instruction set. Once a compiler or interpreter existed on a given machine, it could be used to translate the high-level language into code for that machine without the programmer knowing how it was done. Programs not only became more readable, they also became portable. In fact, a major effort at standardization of high-level languages is a part of the computer industry today, with such languages as COBOL, FORTRAN, PL/I, MUMPS, and Pascal, approved by the American National Standards Institute (ANSI), representing important early results of those efforts.

As programming languages became more user friendly, they opened the doors to more users. As applications became more complex, they also led to the potential for simultaneous use of a computer by more than one user. Once again, the operating system had to become more complex to solve problems of sharing the computer resources between many users and making sure that two users did not both try to change the same piece of information at the same time. Time-share systems were thus created, usually with elaborate security protections to prevent (or at least reduce) their misuse. These systems usually allowed multiple tasks to reside in the computer's memory simultaneously and controlled the use of the CPU in such a way as to maximize throughput. Some operating systems grew so complex that they occupied a large part of the computer's resources, so that as machines and their operating systems grew in size, cost, and complexity, the net performance in terms of user's tasks did not grow at the same rate.

Evolution of Smaller Computers

It is time now to turn to another side of the history of computers—their evolution in terms of size, cost, and performance. For the first 25 years of computer history (from 1945 to about 1970), the general tendency was to assume that "bigger is better." This statement was definitely true in the early years, when economies of scale led to vastly greater performance per dollar. Since individual components were very expensive, it was unthinkable for a small department, let alone an individual, to consider having a computer dedicated to individual needs. This fact led to the control of computing resources by large central administrative units—the director of hospital administration, the central office of a federal bureau, the computer center of a university. Decisions as to which computers were to be purchased or which tasks performed and to the assignment of priorities among those tasks were also centrally controlled. Furthermore, since the computers were expensive, their cost usually had to be justified in terms of performing some necessary, cost-saving function, such as billing and accounting.

In the mid-1960s, however, a new trend became evident. Computers were already becoming less expensive through technological advances in hardware manufacturing and the discovery of new, cheaper components to replace older, more expensive, and less reliable ones. As a result, a new breed of computer evolved—the minicomputer. This type of system (the first ones were developed to help biological researchers in their laboratories) cost less, could be interfaced to laboratory equipment, and could be programmed by scientists in other fields. They led to the foundation of the Digital Equipment Corporation, the largest manufacturer of minicomputers to this day. It became possible, because of these developments, for departments to have their own computers, and the trend toward decentralization started. An analogy can be drawn between this decentralization and the one that took place in human use of power. As mankind developed more and more need for power, humans and beasts of burden were no longer able to supply the work needed, and man turned to water, creating water-driven mills to process lumber, textiles, and other raw materials. Mill towns were formed where water power was available, and the economies of whole nations changed as a result. With the advent of electrical power, however, the situation changed. Power now became available where it was needed, and power-dependent activities were spread out to suit the needs of the people using them, not the source of power itself (which remained for some time water).

In the same way, computers went through a phase of increasing centralization because that was the only way these new tools could be used effectively. With the advent of minicomputers and then, in the 1970s, microcomputers, decentralization was not only possible, it was economical. Costs of computer hardware continued to drop, and performance (speed and reliability) continued to rise. According to *Time* magazine, if automobiles had shown the same trends in cost and performance, "a Rolls-Royce would now cost $2.75 and run 3 million miles on a gallon of gas."[2]

The net result of this revolution in computing capability is that personal computers are affordable in every office practice, whether solo or large group. The functions such computers can perform are seemingly unbounded. The main question that arises, therefore, has to do with who will play a role in deciding how they will be used.

Problems and Opportunities Associated With Computer Evolution

Health professionals who want to make use of computers in the next few years should realize that the evolution of computers has not stopped; it is continuing at a rate close to that described above. New developments will affect their utility in the health care field. These new developments can be considered as problems or as opportunities, depending on the way in which computer-based health care delivery projects are designed. Although it is difficult to predict the future, some trends do seem likely, and these trends will have a major effect on health care computing.

Hardware Advances

The computer industry is moving rapidly toward new generations of hardware. The advances that are taking place affect particularly

CPU design, auxiliary storage devices, and input devices. Each of these elements is critical in the design of clinical information systems. The availability of a new microprocessor chip does not mean the old ones are obsolete; it does mean, however, that flexibility is a key to planning.

The variety of input devices already available exceeds the imagination of most system designers. Voice, touch screen, bar code, ergonomic keyboard—these are techniques one hears of but seldom sees in a commercial system despite their obvious benefits for certain applications. Health care systems should be particularly aware of new opportunities in input, currently the bottleneck to computer use in this field. As new technologies evolve, the foresighted planner should consider carefully their potential in his sphere of activity.

New storage devices (laser-encoded data of highly durable type on devices the size of a credit card, smaller and more densely packed disks, the use of holographic storage) will undoubtedly influence system configurations of office systems in the relatively short-term future.

The rapid appearance of new central processors means that greater attention must be devoted to software compatibility than ever before. Since existing computers are still useful, they must be interfaced with newer or larger machines; the programs developed on smaller systems should be upward compatible, so that they can run equally well on the next generation of hardware, or so that the tasks can be subdivided, and the older hardware can retain some functions while the new equipment takes on newer but integrated functions.

Understanding Clinical Information

Computer scientists have made significant advances in their understanding of knowledge and information. Medical information has played a role in these advances, since many of the most advanced information science projects today use medical information as a vehicle for research.

In a general sense, however, clinical participation in the use of computers as a practical day-to-day tool has been limited. The result has been that there are very few examples today of clinically effective computer-based information systems. The problem-oriented medical record, for example, is a concept that is undoubtedly useful in a paper record, but it is nearly essential in a computer-based system, where cross-referencing problems with medications, tests, and progress notes give the computer tools whereby to improve the utility of the record system by orders of magnitude not only for individual patient care but for community health related epidemiological investigations. There are very few computer-based clinical systems, large or small, that incorporate elements of the problem-oriented medical record. This state of affairs appears to underscore the general lack of participation in the design of these systems by physicians, who alone can grasp fully the meaning of medical information.

Viewed in this perspective, the state of computer usage in handling clinical information today can be considered primitive at best, despite the sophistication of low-cost tools available. Mistakes are waiting to be made, insight is waiting to be elucidated, and the true utility of the computer remains at present unknown.

Comment

The technology now available offers as-yet unrealized potential for improving health care delivery. There is a real urgency for clinicians to inform themselves about these tools. This process should include developing an understanding of the hardware and software that perform the basic tasks, but it must inevitably lead to a review of the clinician's understanding of medical information itself. Here, the computer scientist is at a disadvantage, lacking an insight into the complexities of the medical data base. The physician, and only the physician, can contribute the vital link between the medical knowledge and its effective computerization. The remainder of this issue is intended to help the physician achieve that potential.

References

1. Meninger K: Number Words and Number Symbols. Cambridge, Mass, MIT Press, 1969
2. Machine of the year—1982. Time, January 3, 1983, p 8