Award Lecture

COMPUTING IN ENGINEERING EDUCATION

From There, To Here, To Where?

Part 1: Computing

BRICE CARNAHAN
University of Michigan
Ann Arbor, MI 48109

The ASEE Chemical Engineering Division Lecturer for 1990 is Brice Carnahan of The University of Michigan. The 3M Company provides financial support for this annual lectureship award, and its purpose is to recognize outstanding achievement in an important field of ChE theory or practice.

Brice earned his BS and MS degrees from the Case Institute of Technology (1955, 1956), and his PhD from the University of Michigan in 1965, all in chemical engineering. His doctoral research was on radiation-induced cracking of paraffins. Between 1959 and 1965, he worked closely with Professor Donald L. Katz, first as technical director of the Ford Foundation project *Computers in Engineering Education* and then as associate director of a follow-on NSF project, *Computers in Engineering Design Education.*. He joined the faculty of the University of Michigan in 1965, where his research activities have focused on applied mathematics, modeling, digital computing, and development of software for computer-aided process analysis and dynamic simulation. He is coauthor of two Wiley Texts, *Applied Numerical Methods* and *Digital Computing and Numerical Methods.*

He and his colleague, Professor James Wilkes, are responsible for the required computing course for all freshmen engineering students at the University of Michigan, for which they have produced a steady stream of texts and instructional aids over the years.

Professor Carnahan was a founding member and first interim chairman of CACHE. He has subsequently served as CACHE vice-chairman and chairman, and is currently active as board member and publications chairman. He has held elected AIChE positions as CAST Division Director, Vice-Chairman, and Chairman, and is a member of the Editorial Board of *Computers & Chemical Engineering.*

Since the early 1980s, Professor Carnahan has been intimately involved with the planning, implementation, and management of the Michigan College of Engineering heirarchical, multivendor network, now incorporating over 2000 attached machines of widely varying power.

He has received numerous honors, including the University of Michigan's Distinguished Service Award (1974), the AIChE CAST Division Computers in Chemical Engineering Award (1980), the University of Michigan College of Engineering's Outstanding Teaching Award (1984), and the Detroit Engineering Society's Chemical Engineer of the Year Award (1989).

Notice of the 3M Lectureship award for 1990 came to me as a complete, though a very pleasant, surprise. Many chemical engineering academics have had greater impact on their specialties, including engineering computation. Nevertheless, I very much appreciate this singular recognition.

I would be remiss if I did not here acknowledge the special contributions of two Michigan faculty to my professional life and, indirectly, to this award. The first is Don Katz, one of the greats of 20th Century chemical engineering, who provided me at a young age with opportunity, responsibility, encouragement, and financial support for pursuing my interests in chemical engineering computing. He is sorely missed by all who knew him. The second is my colleague, Jim Wilkes, with whom I have worked and taught on an almost daily basis for the past thirty years. That sounds like a long time, but in fact, the years of our collaboration have passed all too quickly. They have been filled with much work, a sense of accomplishment, and lots of fun. Thanks, Jim. It's been great working with you. Here's to the future...and, yes Jim, I *will* work on that revision of Chapter 6...soon....

WHAT IS COMPUTING?

It is a bit disconcerting to be introduced as an "expert" on almost any topic, since the audience then expects the speaker to make the complicated simple, to provide clever insights into the nature of a phenomenon, or to predict the future accurately. It is especially onerous to be labeled a "computing" expert. The truth is that no individual can get a handle on more than a few small subspaces of what has become an enormous and amorphous computing universe, including, but not limited to:

1. **Design and manufacture of hardware for symbolic (mostly numerical) operations, storage, display, and**

communication (*e.g.* networks)

2. Ancillary electronic equipment (*e.g.*, sensors, a/d converters)
3. Software (*e.g.*, operating systems) for hardware management, communication, and user interaction
4. A wide variety of procedural, object-oriented, and other tools for creating applications
5. Application programs for:
   - Creating and publishing documents
   - Organized storage and retrieval of information
   - Business and financial transaction/record keeping
   - Implementation of numerical and non-numerical algorithms
   - Engineering/scientific analysis, design, control, and simulation
   - Creation of graphical images
   - Visualization of computed results
   - Image analysis and pattern recognition
   - Integrating media (text, graphics, video, sound, TV) for education and entertainment
   - Knowledge-based tools predicated on rules and heuristics
   - Language, semantics, organization of the brain and human thought processes

Everyone, both lay and technically trained, is profoundly affected by "computing," but each of us has a private version of what computing is, based on our own limited experience (much like the elephant and the blind men).

I chose the lecture title primarily because this is a meeting of engineering educators, and few technological developments have had (and will in the future have) so pervasive an impact on engineering education and research as has digital computing. Unlike many important technological developments in the history of engineering, computing has not "matured" after fifty years of steady (often spectacular) advances. In fact, as we enter the last decade of this century, the pace of change is accelerating significantly in all of the areas listed above. The question mark in the title will let me end with some conjectures about current trends and the future.

Computing developments in engineering education have occurred by and large during my professional lifetime, starting in the mid-1950s. I would like to start from the perspective of a newly graduated (in 1955) chemical engineer, trace some of what I perceive as the most important computing developments over the past fifty years or so, and then make some predictions (guesses, really) about what the future may hold *vis-a-vis* computers and computing in engineering education. I chose to put "engineering" rather than "chemical engineering" in the title because computing in chemical engineering isn't all that different from computing in other engineering disciplines.

I would like to . . . trace some of what I perceive as the most important computing developments over the past fifty years or so, and then make some predictions (guesses, really) about the future . . .

In fact, many of the computing tools used most by both students and faculty (*e.g.,* word processors, data-base managers, spreadsheet programs, drawing and plotting packages, electronic mail and conferencing software) are essentially "non-technical"; of course, "technical" computing (involving large-scale programs for symbolic and numerical mathematics, analysis, design, and control) is also important to all of us some of the time, and I don't want to leave it out—I just want to take a broader view of what computing in engineering education is now and what it is likely to be in the future.

THERE—THE EARLY YEARS

Let's start with the "there" part of my title. "There" for me started when I graduated from Case Tech in 1955, within months of the introduction of the IBM 650, the first widely available commercial digital computer. That event passed without my knowledge. I had heard of (and seen, on television) the UNIVAC computer, mostly because of its use in tabulating and predicting the vote in the 1952 presidential election. The only computing device I had seen personally was an enormous unused mechanical analog integrator (covering perhaps two-hundred square feet of floor space) in the ME department at Case that had been used to solve some ODE's during World War II. The twelve-foot long K&E sliderule hanging on the wall of the same room looked a lot more useful to me. It was a prop for teaching new freshmen about fast and accurate calculation (three digits still isn't all that bad!). That giant rule, along with the dreaded drafting exercises (where were you, Claris CAD, when I needed you?), is retained vividly as part of my freshman memory.

I am surprised at how little most students (and faculty) know about the personalities and historical events that led up to the successful IBM 650 venture. Mention "light-bulb" and the response is "Edison"; "airplane" and the response is "Orville and Wilbur Wright"; "telephone" and the response is "Alexander Graham Bell"; "computer" and the response is (almost always) silence or (inaccurately) "IBM." Although many mechanical or electromechanical calculating machines were developed (very early by Pascal, late in the 19th Century by Burroughs and Hollerith, and during the first half of the 20th

Century by IBM and other companies), what most of us would call programmable digital computing developed along an essentially independent path, with ideas generated by a small number of clever, determined, and sometimes irascible, individuals. Table 1 shows a chronology of a few milestone events from the early history of digital computing.

Babbage,[1] who for a time held Newton's chair at Cambridge, is a tremendously interesting personality. His mechanical analytical engine incorporated the most important conceptual elements of the modern serial digital computer "architecture," with the exception of the stored program. Much of what we know about Babbage's analytical engine stems from its promotion by Lady Ada Lovelace (hence the name for the programming language Ada), who was Lord Byron's daughter and a mathematician of some note. Babbage never got his engine to work, despite the expenditure of a great deal of his own money and earlier support from the British Admiralty (the first federal R&D proposal?). This failure was not caused by a flaw in his design, but because of his unusual management style and problems with accurate metal machining. Parts of his machine were built in the 1950s and are on display at the Science Museum in London (see Figure 1).

Nearly a century passed before Atanasoff designed the first all-electronic (vacuum tube) computational circuitry and built a special purpose digital computer at Iowa State University for solving twenty-nine (why *twenty-nine* is not clear) simultaneous linear equations. His work was interrupted by World War II, and his contributions are often slighted by historians. However, a recent thoroughly documented book[2] makes it clear that Atanasoff's contributions were substantial, and that they influenced the subsequent development of the ENIAC by Eckert and Mauchly at the University of Pennsylvania's Moore School.

The ENIAC was the first truly programmable digital computer; all programming was done manually with switches and cables. It was used for computing firing tables for the military, and its existence became public knowledge in 1946, after World War II. Some statistics: the machine was 100 feet long, 8.5 feet high, and several feet wide; it had twenty 10-digit registers in its arithmetic unit (each 2 feet long), and 18,000 vacuum tubes. An integer add required 200 microseconds, making it something like a 0.005 Mips (Million instructions per second) machine. The ENIAC (see Figure 2) was two to three orders of magnitude larger physically, and its typical instruction time was three to six orders of magnitude longer than today's computers!

In a classic 1946 paper,[3] Burks, Goldstine, and von Neumann first introduced the stored-program and other architectural concepts that appear in nearly

TABLE 1
Digital Computing: Early History

Date	*Machine • Description • Developer*
1833-1848	Analytical engine • mechanical general-purpose computer • Babbage at Cambridge and London
1939-1942	ABC linear equation solver • first all-electronic computational hardware • Atanasoff at Iowa State University
1944-1946	ENIAC (Electronic Numerical Integrator and Calculator) • first general-purpose electronic computer • Eckert and Mauchly at the University of Pennsylvania
1946	EDVAC (Electronic Discrete Variable Electronic Computer) • paper • stored program concept • Burks, Goldstine, and von Neumann at Princeton
1947-1952	Mark I, II, III, IV • electromechanical computers with separate data and instruction memories • Aiken at Harvard
1947	Whirlwind • special-purpose radar processor, first machine with core memory • MIT
1949	EDSAC (Electronic Delay Storage Automatic Computer) • first operating stored-program machine • Wilkes at Cambridge University
1950	BINAC • first American stored program computer • Eckert and Mauchly Co. for Northrup Aviation
1951	UNIVAC • first commercial computer (48 built) • Remington-Rand Corp.
1952	IBM 701 • first core-memory machine (19 built) • IBM
1955	IBM 650 • first high-volume computer (hundreds built), drum memory • IBM
1955	IBM 704 • first large scientific machine, first built-in floating point unit • IBM

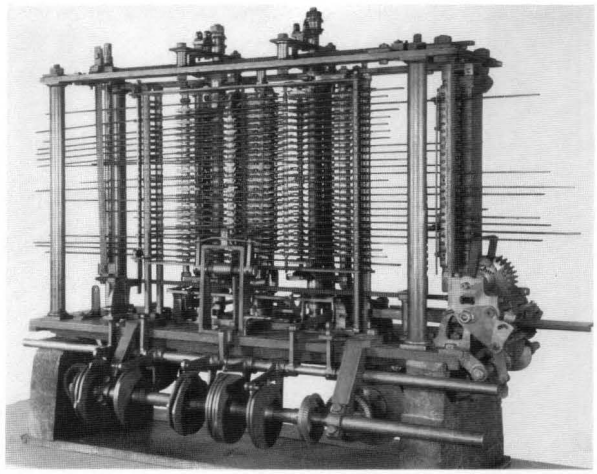

Figure 1. *Part of the mill (arithmetic unit) of Babbage's Analytical Engine, constructed after his death from original drawings. (British Crown Copyright, Science Museum, London)*

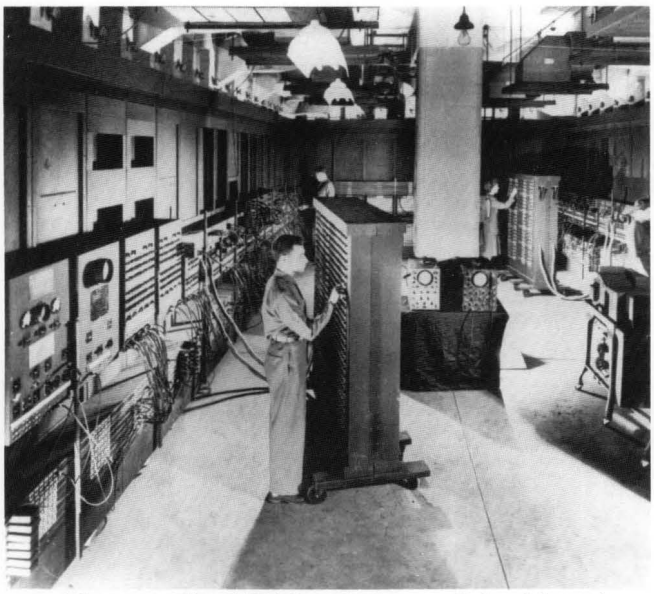

Figure 2. *The ENIAC at the Moore School (1946).*

all of our current (serial) computers; they called their machine the EDVAC. EDSAC, built by Wilkes at Cambridge University, was the first true stored-program machine built on the EDVAC model; it became operational in 1949.

The first American stored-program machine was the BINAC, built for Northrup Aviation by Eckert and Mauchly (who left the Moore School in 1947 to start their own company). It was fully functional by mid-1950 and served as the basis for the first commercial digital computer, the Remington-Rand UNIVAC, released in 1951; forty-eight UNIVAC systems were built, and the cost per machine was $250,000 (about $3 million in today's dollars).

IBM entered the digital computing business shortly after Remington-Rand, introducing its first computer, the IBM 701, in 1952; nineteen were built. The IBM 701 was the first stored-program machine to use truly random access magnetic core memory (previously developed at MIT in 1947 for a special-purpose radar signal processor called the Whirlwind).

At the same time, IBM was developing two other machines. One was a follow-on core-memory machine with the first built-in floating-point unit, the IBM 704; it was not really available in quantity until 1957-58. The second was a less expensive "mass-market" computer, the IBM 650, with a magnetic drum memory. IBM eventually built several hundred of them, mostly for rental. The University of Michigan rented an IBM 650 in early 1956 to replace its mostly unsuccessful research computer with mercury delay line storage called the MIDAC (MIchigan Automatic Digital Computer). The few who actually used MIDAC derisively said the acronym really stood for "Machine Is Down Almost Continuously." As I recall, the rental rate for the 650 was $35 per daytime hour (but only for hours when it was up!).

The presence of the new computer had nothing to do with my decision to go to Michigan for PhD work in the fall of 1956. I chose Michigan because it was one of the few schools with its own nuclear reactor, and I wanted to work with Joe Martin on a chemical/nuclear engineering problem. When I met with Joe for my first counseling session, he told me about the new University computer and that the mathematics department was offering a new course on digital computing, the first at Michigan. Once I was in that course (with about twenty other students) I knew that I wanted to be involved with computers far into the future (even though my research was to be unrelated to it). In fact, I became a teaching assistant in that first computing course the next term it was offered.

For those (most of you) who weren't around at that time, here is a picture of what students did during that first course offering:

- **Each of us learned to operate the computer and then signed up for, at most, one hour at a time to solve our problems (I always ended up with the 2:00-3:00 AM slot!).**
- **The machine had no keyboard or printer—just a card reader and card punch. All communication was through punched cards or directly with keys on the console (the lights displayed information in *bi-quinary* format—you might want to look that one up!).**
- **All programming was in the machine's language; each instruction contained an operation code plus two addresses, one for an operand and another for locating the next instruction in the memory.**
- **The "operating system" consisted of a four-card machine-language loader. Program execution could be initiated, interrupted, or stepped one instruction at a time, directly from the console; the light pattern on the console was the only feedback available to the programmer/operator (the repeated light patterns from infinite loops were always fun to watch).**
- **The machine had a rotating-drum memory with fifty memory cells arranged in each of twenty "cylinders' around the drum surface. Because of the time required for interpreting an instruction, retrieving the operand, and then processing the instruction, placement of both the data and the next instruction was critical for efficient execution. The location of each program instruction and data item on the drum had to be carefully considered, since a drum is not a random-access device.**

How do you think a current student working on a Macintosh would respond to the following directions?

If the instruction address is an *even* number, the *data address* should be three word positions later (on any cylinder) and the next *instruction address* should be four word positions beyond that. Since there are fifty word positions around the cylinder, the correct drum rotation angle for the next

instruction if 50.4 degrees. . . . If the instruction address is *odd*, the data address should be three word positions later and the next instruction address should be five positions beyond that, so the drum rotation angle for the next instruction is 57.6 degrees.

Not to worry—part-way through the course we began to use the GAT assembler, written by Graham, Arden, and Galler of the University of Michigan Computing Center. That helped a bit (symbolic names for operation codes and addresses) but still left the angle determination to the programmer. Then one day, late in the term, the SOAP assembler arrived. . .and life was never the same thereafter. The O in SOAP stood for "optimal," and the SOAP assembler took care of all those nasty angle details. After struggling with the machine's language, SOAP seemed nothing short of a miracle (I was amazed, like the monk in the XEROX ad).

I still have my programs from that course. The first was (you guessed it), "Find the volume of a cylinder, given the radius and height as data." I remember thinking that I could have done the whole thing on a slide rule in a tiny fraction of the time it took me to learn how to run the 650 and get the program working. But later in the course we were each asked to solve a problem of our own. I decided to solve the two-dimensional heat-conduction (Laplace) equation in an L-shaped section of a furnace wall. I can still remember the thrill of getting the program working—and not just working, but working with variable mesh sizes. It was my first exposure to the true power of the computer and of numerical methods.

For me, the computer die was cast!

TRENDS IN COMPUTER PERFORMANCE

In those very early days, it was clear to me that computers would get faster, more reliable, and less expensive—but *not* that they would get incredibly smaller, and orders-of-magnitude faster and cheaper (on a $/instruction or $/memory location basis). Data from the recent (already classic) text on computer architecture by Hennessy and Patterson[4] on the relative performance of several classes of computers over the past twenty-five years or so is shown in Figure 3. The performance index is based on the time to completion of a mix of typical programs.

By and large, prices in current dollars of the various categories of machines have stayed fairly stable. Supercomputers typically cost many millions, mainframes sell for $500,000 to several million, minicomputers from $50,000 to $500,000, and microcomputers from $1,000 (minimal personal computers) to $75,000 (for high-performance workstations). Note that the rate of improvement in the performance index is undiminished over a twenty-five-year span and varies from about 18% per year for supercomputers to about twice that for microcomputers.

Figure 4 shows a different performance index for supercomputers and microprocessors that is particularly relevant to numerical engineering computations, MFLOPS (Millions of Floating-Point Operations Per Second). Although supercomputer processors still perform floating-point operations one to two orders-of-magnitude faster than the fastest current microprocessors, the message here is clear: the latest RISC (Reduced Instruction Set Computer) microprocesors (the middle curve) portend a rapid closure of the floating-point performance gap by relatively inexpensive microprocessors.

Figure 5 shows the rapid price/performance decreases over the past decade for DRAM (Dynamic Random Access Memory) chips used in computer

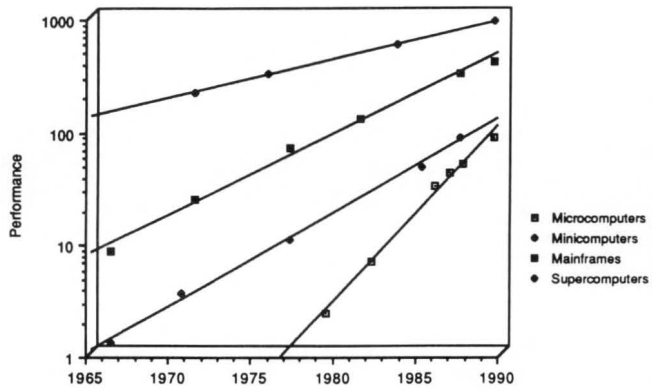

Figure 3. *Relative performance by computer class (data from Hennessy and Patterson[4]).*

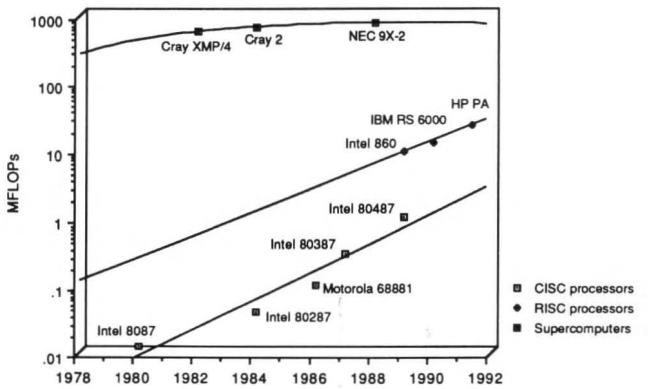

Figure 4. *Floating-point performance of supercomputer and microcomputer processors (most data from Intel).*

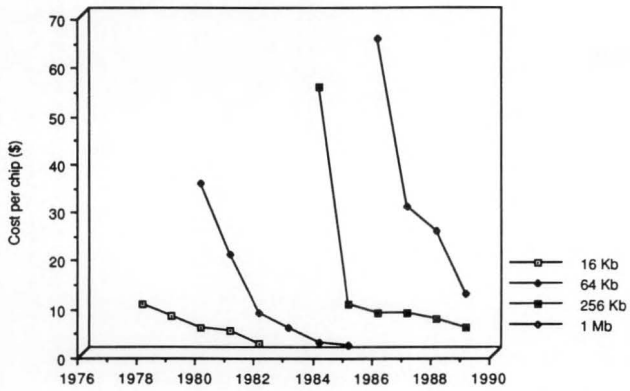

Figure 5. *Costs of several generations of DRAM chips (data from Hennessy and Patterson[4]).*

TABLE 2
Hardware/Software Milestones

Year	*Milestone*
1960	ALGOL • Magnetic disks
1962	Time sharing (Dartmouth) • Virtual memory (ATLAS at Manchester)
1964	Pipelined processors (CDC 6600) • Microcoded processors, 32 bits, byte (IBM 360)
1965	Interactive graphics, Sketchpad (Sutherland)
1966	Multiprogramming • Minicomputer (DEC PDP/8) • Real-time computing
1967	Multiprocessing • Memory cache (IBM 360/85)
1969	Minicomputer (DEC PDP/11) • PASCAL
1970	UNIX
1971	4-bit Microprocessor (LSI-Intel 4004) • IBM 370
1972	Vector processor (CDC STAR)
1974	Personal (mini)computer (XEROX Alto), bitmapped display, mouse • Laser printer • Local Area Network (Ethernet)
1975	Object-oriented programming (Smalltalk) • 8-bit microprocessor (Intel 8008)
1976	16-bit microprocessor (Texas Instrument 9000) • Supercomputer (Cray I) • ARPANET • C
1977	Microcomputers (Apple II, TRS-80, PET)
1978	DEC VAX • Intel 8086 microprocessor
1979	Spreadsheets (VisiCalc) • Hayes Micromodem
1980	RISC processor (Berkeley, Stanford, IBM)
1981	Graphical user interface (XEROX STAR) • IBM PC • DOS • Epson dot matrix printer
1982	Compaq portable • Cray XMP/4
1983	Apple Lisa • Gavilan laptop
1984	Macintosh • HP Laserjet printer
1985	Workstation (Apollo) • Desktop publishing (Postscript)
1986	IBM 3090 • Windows graphical user interface
1987	Sparc RISC processor (SUN workstation)
1988	Cray Y/MP (8 processors, 6 ns clock) • Convex, Alliant minisupercomputors • Stellar, Ardent, Silicon Graphics, graphics workstations • visualization • massively parallel processing (Connection machine) • OS/2
1989	Open Software Foundation (Standard UNIX)
1990	Superscalar RISC processor (IBM RS6000)
1991	ACE-MIPS RISC processor consortium • HP PA RISC processor • Apple-IBM agreement • Pen-based, notebook, handheld microcomputers

main stores. Here the prices are in current (inflated) dollars. Note that for each chip category there is a similar pattern of a steep (nearly ten-fold) fall in prices as the chip goes into production and that the price cycles are almost identical despite the successive quadrupling of capacity.

Some long-range trends in computing equipment development are:[4]

- **Performance growth ranges from 18% per year for supercomputer processors to 35% per year for microprocessors.**
- **Dynamic RAM chip element density increases about 60% per year. 4-Mbit chips are now in mass production and IBM has announced plans to begin producing 16-Mbit chips. Hitachi has already fabricated a 64-Mbit chip in its laboratories.**
- **Chip transistor count increases about 25% per year, doubling every three years.**
- **Hard disk bit density increases about 25% per year, doubling every three years.**
- **Hard disk access time improves slowly (only 3 to 4% per year).**

PREDICTING THE FUTURE

Who, in the late 1950s, would have guessed that national computer meetings that brought together a few hundred participants then would, only thirty years later, sometimes attract in excess of 100,000 attendees—and be held only in one or two dreadful places like Las Vegas and Anaheim for lack of room elsewhere? Who then could have guessed the scope of the computing business now?

Well, some did. I remember a talk by Thomas Watson, Jr., in 1959, at the dedication ceremony for the University's new IBM 704. He predicted that by 1990, the computing business would be as big as the automobile business. That didn't quite happen, as sales by the major computer companies are still substantially smaller than for the major auto manufacturers. Of course, had the car companies delivered performance improvements comparable to those for the products of the computing industry, we would all be driving $1 Ferraris across the continent in a few seconds, and car-company sales might not look so big (one disadvantage—the car would be very, very small!). If revenues from information-related businesses such as communication are added to those for the computing manufacturers, Watson's prediction has probably already come true. In any event, it is certain to come true before the turn of the century. Oh, that I had had some investment cash in 1959!

What about other early predictions? In 1945, Vannevar Bush, inventor of the electronic analog

computer at MIT and Director of the Office of Scientific Research and Development during World War II, postulated a future device that is clearly similar to the personal computer we (almost) all know and love. In an article entitled "As We May Think,"[5] he wrote:

> **The MEMEX will be for individual use, about the size of a desk, with display and keyboard that would allow quick reference to private records, journal articles, newspapers, and perform calculations.**

Unfortunately, in 1967, in an article entitled "MEMEX Revisited," he wrote:

> **Will we soon have a personal machine for our own use? Unfortunately not!**

How wrong he was, with the first microprocessor only a few years away. Of course, Vannevar Bush had apparently been wrong before. As a consultant, he is reputed to have advised IBM in the early 1950s that one-hundred IBM 650s would saturate the market, since they could do all the computing that the world needed done! (Could he have been right?)

After hearing many predictions over the years, I don't think that even the brightest are good at predicting the future of computing much beyond the next generation of hardware and software. This is not to be critical. Who among us in 1956 (slide rule hanging from belt) would have predicted that in 1990 I could buy a pocket calculator for $50 (in greatly inflated currency) that uses a procedure-oriented language, can retain several programs indefinitely, computes to at least eight-digit accuracy, and operates for months on end on a battery smaller than a dime?

THREE DECADES OF STEADY PROGRESS

Table 2 shows a chronology of major hardware/software developments during the past three decades, as I see them. I have verified most of the dates, but a few are from my own recollection and may be off by a year or two.

Having gone from "there" to "here" in the general categories of hardware and software, Table 3 shows several areas of chemical engineering where these technologies have had the biggest impact. Here I have not tried to arrange the list in strict chronological order.

Bob Seader (University of Utah) was the recipient of the 1990 Katz lectureship in our department. One of his two lectures was entitled "A Brief History of Computing in Chemical Engineering." His superb lecture covered the subject so well that I couldn't possibly improve on it here. A printed copy of Bob's lecture was sent to every chemical engineering department chairman last fall, and I highly recommend that you locate and read it. If you cannot find a copy, contact me and I will send one to you.

TABLE 3
Computing in Chemical Engineering

Topic
• Process unit modeling
• Data analysis/reduction
• Physical property estimation
• Steady-state simulation
• Costing
• Reservoir simulation
• Optimization
• Scaleup without pilot plants
• Dynamic simulation
• Process control
• Control system design
• Process synthesis
• Batch-process simulators/schedulers
• Knowledge-based (AI/expert system) synthesis and design
• Graphics and visualization
• Molecular and property modeling (polymers, composites)
• Microelectronic processing/sensors
• Integrated process/control/information management systems
• Biochemical system modeling/simulation/design/control
• Intensive use of numerical analysis tools:
- linear and nonlinear algebraic/transcendental equations
- ordinary differential equations, stiff systems
- partial differential equations (finite difference/element methods)
• Education/training
• Office, plant, education networks

Editor's Note: The second half of this award lecture will be published in the next issue (Winter 1992) of CEE.

REFERENCES

1. Morrison, Phillip and Emily, *Charles Babbage and His Calculating Engines,* Dover, New York (1961)
2. Burks, Alice R. and Arthur W., *The First Electronic Computer: The Atanasoff Story,* University of Michigan Press, Ann Arbor, MI (1988)
3. Burks, A.W., H.H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," report of the Institute for Advanced Study, Princeton (1946). Reprinted in *Datamation,* **8, 9, 10** (1962)
4. Hennessy, John L., and David A. Patterson, *Computer Architecture: A Quantitative Approach,* Morgan Kauffman, San Mateo, CA (1990)
5. Bush, Vannevar, *Endless Horizons,* Public Affairs Press (1946) ❐