Published by The Internet-First University Press

The Early Years of Academic Computing:

A Memoir by Douglas S. Gale

This is from a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period when Cornell developed its own decentralized computing environments and networking became a national strategic goal.

Initial Release: April 2016

Published by The Internet-First University Press
Copy editor and proofreader: Dianne Ferriss
The entire incremental book is openly available at http://hdl.handle.net/1813/36810

Books and Articles Collection – http://ecommons.library.cornell.edu/handle/1813/63
The Internet-First University Press – http://ecommons.library.cornell.edu/handle/1813/62

C. Memoir by Douglas S. Gale

Preface and Introduction

Bill Arms and Ken King have chronicled much of the early history of academic computing. Accordingly, I have tried to focus my contribution to *The Early Years of Academic Computing* on topics less covered by King and Arms. My contribution is organized into four chronological periods, roughly spanning the last 50 years of the 20th century, and each has an organizational theme: career transitions, the microcomputer revolution, the networking revolution, and the changing role of academic computing and networking.

The theme of the first period -- career transitions -- will consider where the early practitioners of academic computing came from. Computer Science was first recognized as an academic discipline in 1962, and the early years of academic computing were led by people who had received their academic training in other disciplines. Many, if not most, came from the community of computer users and early adopters. The first period was characterized by the emergence of computing, both academic and administrative, as something of importance in higher education. Computing centers were created and awareness of their importance began moving up the administrative hierarchy.

The second period will focus on the emergence of decentralized computing driven by the advent of small, inexpensive, and powerful microcomputers. This paradigm shift involved more than hardware and software. It was a paradigm shift in how the technology was managed, organized, and supported. Higher education played a key, and largely unrecognized, roll in this transition.

The third period will focus on the contributions of higher education to the evolution and widespread use of computer networks. Much as the "printing" revolution was based on the technology of movable type and mass-produced paper, future generations will probably not differentiate between the technologies of computing and networking.

The fourth period will focus on how the relative roles of academic computing, administrative computing, and networking changed in the final decade of the century in response to the microcomputer, distributed computing, and the Internet. In particular, this period marks the relative decrease in the role of centralized academic computing and an increase in the importance of administrative computing.

The third and fourth periods were also characterized by the growing recognition that computing and networking were strategic to the role and mission of higher education.

Contents

Chapter 1

Career Transitions

Mille viae ducunt homines per saecula Romam (A thousand roads lead men forever to Rome)

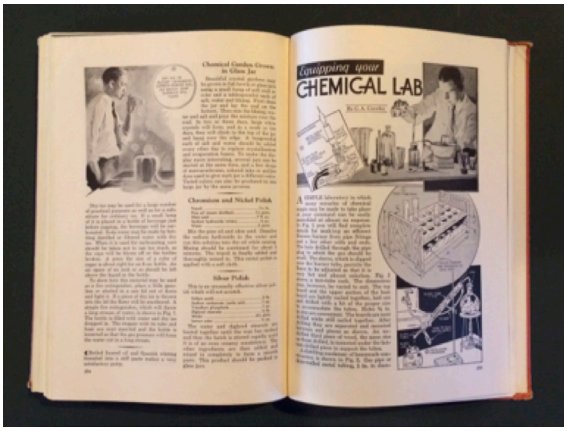
Although computer science was identified as an academic discipline in 1962, at Purdue University, in the early years the ranks of academic computing practitioners were largely filled by individuals who received their formal training in other disciplines. What disciplines? What was their background? Why were they drawn to computing? How did the transition occur?

This section will follow the author's transition from a student, to a scientist and user of computer technology, and to an information technologist. My experience in conducting oral history interviews for the Internet Legacy Institute (www.internetlegacyinstitute.org) indicates that my experience, from user to practitioner, was typical of the career transitions of many of the early academic computing leaders. Hopefully, it will provide some perspective on why and how so many educators and researchers were drawn to this new discipline.

Computing for a Kid in the Fifties

For kids with technical interests, growing up in the 1950s was much different than for one growing up in the 21st century. Computers were outside our range of experiences. If we had heard of them at all, they were regarded as something used by businesses to sort punch cards or by research laboratories to calculate new artillery tables. But they had little relevance to what we did on a day-to-day basis. Our heroes, Buzz Cory (Space Patrol: http://users.bestweb.net/~foosie/spacepat.htm), and Tom Corbett (http://en.wikipedia.org/wiki/Tom_Corbett,_Space_Cadet), as well as older heroes such as the Lone Ranger and Superman, didn't need or use computers to explore the universe and bring evildoers to justice. Computers weren't referenced in either our textbooks or our comic books.

Mechanical calculators, which relied on ingenious combinations of gears to perform simple arithmetic calculations, were developed as early as the 17th century (http://en.wikipedia.org/wiki/Mechanical_calculator). But the complexity of their construction and resulting cost limited their use. If a family had a computer at all, it was likely a cheap slide rule or simple mechanical adder/subtracter.



The Boy Mechanic *Photograph by Doug Gale*

So what did a technically oriented kid do growing up in the '50s? We experimented with things, such as disassembling fireworks to gather the gunpowder in order to make skyrockets and small cannons. My friends and I built a rocket ship, which was basically a collection of old junk and anything that had a switch, and reenacted our heroes' adventures. As we got older, the switches started to become functional and many of us became ham radio operators. My call letters were K0HKY. We made things. I spent many hours as a kid pouring over a copy to the "Boy Mechanic" that my father gave me for Christmas in 1953.

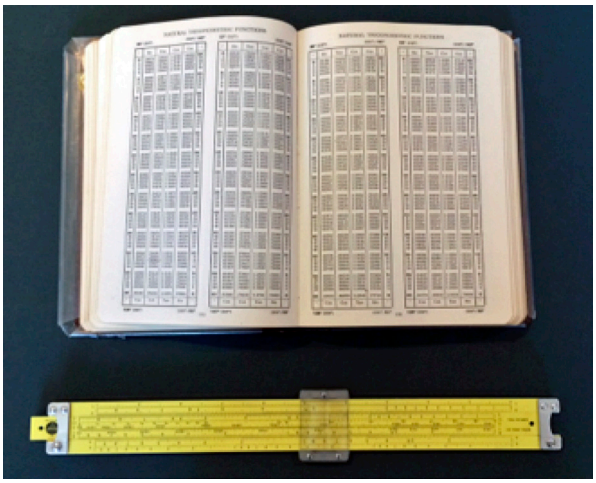
It covered everything from building a boat or a working telephone to equipping a home chemistry lab, complete with dangerous and unstable items.

This hands-on, hardware-oriented approach to technology influenced many of our later careers in computing.

Computing for a Student in the Early Sixties

As a physic's major at the University of Kansas in the early '60s I, like all my classmates, did computations either by hand or with a slide rule. In my case, it was a cheap bamboo slide rule that my uncle won in a poker game. It wasn't until I went to graduate school that I could afford a Pickett, all metal slide rule.

The two indispensible tools of a science or engineering student were a slide rule and the CRC Standard Mathematical Tables, which contained tables of logarithmic and trigonometric functions, as well as other useful data.



Slide Rule and CRC Tables
Photograph by Doug Gale

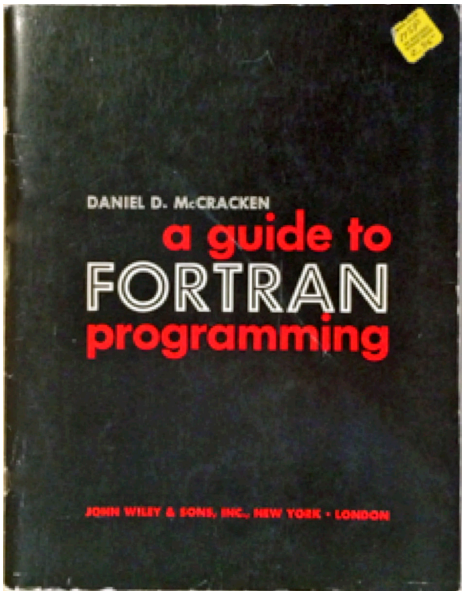
I was introduced to computers in 1963 when my roommate, an Electrical Engineering major, was taking a computer-programming course using the university's General Electric mainframe. It was a perfect match. He was looking for real problems to program, and I was trying to find a way to dress up a mundane compound pendulum laboratory assignment. Using numerical techniques to eliminate the small-angle approximation was just the ticket. I did the experiment and the mathematics, and he wrote the computer code. One of the key observations I made was that my roommate spent a great deal of time writing computer programs; so much so that he added an extra semester to his undergraduate career.

Although most students using computers came from the physical sciences, mathematics, or engineering, there were a small number from the liberal arts as well. One older friend was completing his dissertation in English — a concordance of the works of George Bernard Shaw. He had a rented keypunch on this front porch because it wouldn't fit through the front door. This was ground breaking stuff in the 1960s.

In the fall of 1964, I began work on my master's degree in physics at the University of Minnesota. Towards the end of my first year, my research professor had me begin doing elastic-scattering calculations in preparation for my thesis research. The process was laborious and required a series of calculations on a mechanical calculator.

The Marchant Calculating Machine Company was founded in 1911 and acquired by the Smith Corona typewriter company in 1958. Within a few years, electronic calculators eliminated the market for such machines, and a few years later personal computers eliminated the typewriter business as well.

SCM Marchant Calculator
Photograph from Wikipedia



A Guide to Fortran Programming
Photograph by Doug Gale

The results of those calculations were then used to look up scattering parameters on tables not too unlike the old artillery range tables or the trigonometric tables in the CRC Handbook. I spent a great deal of time over the first summer completing the calculations that would be needed for my research. When I returned to school in the fall, my advisor said, "Now that you understand how elastic scattering works, let me show you an easier way to do it," and instructed me to buy a copy of Daniel McCracken's "A Guide to Fortran Programming" and to write a computer program to do the calculations that I had previously been doing by hand. That was how you learned to program a computer in those days. I still have my copy of McCracken's brief guide on my bookshelf.

McCracken's *Guide to Fortran Programming*: For two decades McCracken's guides to programming in the Fortran (FORumla TRANslation) programming language were the standard Fortran textbooks and were translated into fourteen languages.

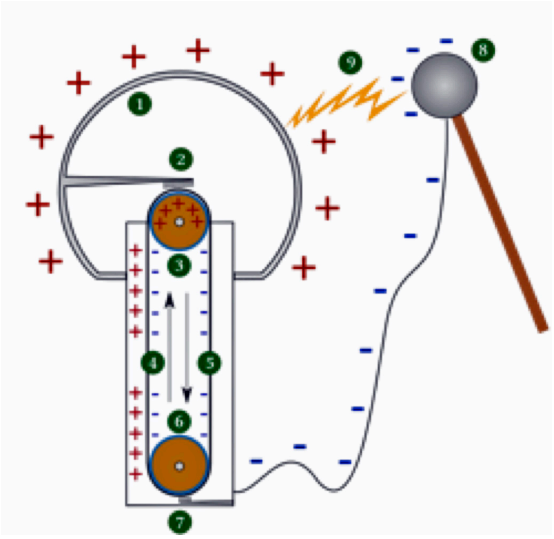
In the fall of 1965, I decided to take a three-quarter course in numerical analysis in the mathematics department to give some formal structure to what I was doing on an ad hoc basis. One assignment in particular made a lasting impression. We were asked to calculate the value of a 4x4 determinant both by hand and by using the University's CDC-6600 mainframe. Control Data Corporation's 6600 was arguably the world's first supercomputer and was designed to solve large scientific problems. Its 60-bit word length in single precision and 120-bits in double precision set the performance standard for many years.

Calculating the determinant exactly by hand was tedious and took the better part of a day, as the 16 elements were all fractions and the lowest common denominator was extremely large. The results of numerical calculations on the 6600, however, ranged from plus infinity to minus infinity! We students were baffled until our instructor demonstrated that the matrix was so ill conditioned that our numerical techniques were useless. I've never forgotten the point he made so dramatically, particularly as I observed the increased reliance on canned numerical analysis programs.

The process of running a computer program consisted of writing the necessary code, usually in Fortran. You would then enter the code onto punched cards on a cardpunch machine. In addition to the code itself, you had to include header and footer cards that contained additional information. These "JCL or Job Control Language" instructions would tell the computer what you wanted done with the program. You would then turn in the punched cards to the computer center; sometime later your deck would reappear, along with the output, on the shelves adjacent to the input window.

It was about this time that I learned the importance of marking the sequence of the punched cards in the deck upon proving the unstated rule, "An unsequenced deck will be dropped and shuffled." Although the last eight columns of the 80 columns of data on a card were reserved for a sequence number, I found it faster to decorate the top of the deck with geometrical drawings done in various colors with felt tipped markers. It also helped in finding my deck on a shelf filled with dozens of other decks.

My thesis work involved nuclear scattering done on a university's Van de Graaff accelerator. My interest in Van de Graaff accelerators began when I was in high school and made one for the Kansas City Science Fair. I naively thought that I could get higher voltages by simply making a bigger machine. I solved the problem of finding a large metal sphere that I could afford by covering a large (3-foot diameter) weather balloon with paper mache painted with conductive carbon paint. The crowned drive pulleys were turned on a homemade lathe, and the belt was driven by a motor liberated from my mother's vacuum cleaner. The whole thing was more than 7-feet tall with the accelerating column made from a large cardboard tube.



Van de Graaff Generator
Illustration from Wikipedia

A metal sphere, atop an insulating column, can be raised to a high voltage by carrying a static charge to the "terminal" on a moving belt. The generator may be used as a particle accelerator by adding a second evacuated insulating column to allow charges to be accelerated from the terminal to the base.

It didn't work very well. The surface of the sphere didn't have a high enough conductance, the accelerating column didn't have a high enough resistance, and there wasn't any mechanism to equalize the potential drop down the column. This was another early lesson that first-order approximations are often inadequate.

The construction of the accelerator at Minnesota begun in 1937 had followed, albeit with far better implementation, the idea that bigger would be better. The accelerator was enclosed in a steel tank filled with compressed air to allow for higher terminal voltages and was three stories tall. The danger was that a spark from the high-voltage terminal would cause a fire inside the tank and raise the air pressure above what the tank could withstand. The external tank remains a fixture on the Minnesota campus. I was the next-to-last student to do their research on the machine, as it was phased out of service shortly after I graduated for a newer accelerator. I found the "hands on" aspect of my research particularly appealing.

University of Minnesota Van de Graaff
Photograph by Doug Gale

The author standing in front of the compressed air shell of the original Minnesota Van de Graaff in 2014, forty-five years after it was decommissioned. It was just too big to move and remains a monument on campus. Each Christmas when it was operational the graduate students would adorn the top of the tank with a brightly lit Christmas tree.

My research made extensive use of a form of computing no longer in fashion --analog computation. Particles were identified by using an electrical analog multiplier to generate a signal proportional to $(E-\partial E)^{0.6}\partial E$, where ∂E was the energy lost by the particle in traversing a gas-filled proportional counter, and E was the energy as it was stopped in a solid-state detector. The magnitude of that signal provided a unique signature for identifying different particles. That information was used in analog circuitry to filter out unwanted interactions. The whole process is now done by recording all "events" and doing a post-analysis on a digital computer.

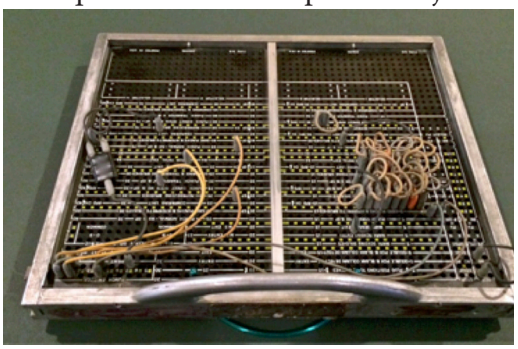
The Late 1960s: Computing Spreads to Smaller Institutions

After completing my master's degree in 1966 I took a break from my education, partially to give my spouse a chance to work on her master's degree and partially to earn some money, and accepted a position as a physics instructor at St. Cloud State College, which was located about an hour from Minneapolis.

While at St. Cloud, I decided that I needed some formal training in computer programming, so when I saw that the University of Minnesota was offering a graduate course in computer programming that was to be offered for three hours one night a week, I immediately signed up. The first class was a bit of a surprise. I was the youngest person in the class. Everyone else was in their 30s or 40s and working for CDC. And the course was CDC machine language programming, not a high-level language such as Fortran. Each week we were given an assignment that was due at class time the next week. Since I was located an hour away and did not have local access to a CDC supercomputer, I would spend the week writing and very carefully reviewing my program. On class day, I would drive to Minneapolis early enough to submit my stack of punch cards in time to get the output before class. The discipline of being meticulously careful not to make mistakes and getting it right the first time were invaluable lessons. Although I didn't know it at the time, the experience of working in machine language would have a profound impact on my later career.

In the mid-60s, using computers in general education courses was relatively uncommon outside a few elite institutions. Programming was considered the specialized province of engineers, scientists, and accountants. I decided, however, to include it in my general education physics classes and prepared a two-page hand out, "Super Simple Fortran." I would cover the material in a single lab period and give each student a personalized assignment due the next day. After much moaning and groaning, the students would always successfully complete the assignment. What I overlooked, unfortunately, was the impact a horde of panicked students would have on the college's limited public keypunch machines.

While I was teaching, my spouse, who had a bachelor's degree in physical education, was taking courses towards her master's degree. Since she had done some keypunching for me the previous year when I was working on my thesis, she decided on a lark to take a programming course. The only one offered was a one-quarter course that covered machine language, assembly language, and Fortran. The instructor was Jack Steingraber, who went on to publish a number of books on computer programming. That course changed her life, as the next quarter she took a part-time job as a computer operator in the college's computer center.



The college only had two computers, an IBM 409 and an IBM 1620. Both were used almost entirely for administrative computing, although the 1620 saw limited academic use. There was no centralized academic computing support organization. The 409, which was introduced in 1959, was an accounting machine that read punched cards, made simple decisions, and printed the results.

IBM 409 Programming Board *Photograph from Wikipedia*

The 409 was "programmed" by moving jumper wires around on a removable panel. To speed changing programs, there were multiple panels, all pre-configured for a particular task.

In contrast to the 409, the university's IBM-1620, which was introduced in 1959 and marketed as an inexpensive "scientific computer," was programmed via a console typewriter or punched cards.

The 1620 was so widely used by scientists and mathematicians that the textbook, *Numerical Methods and Computers* by Shan Kuo, included an appendix of operating instructions for the IBM-1620 that included flipping electrical switches on the console front panel.

IBM 1620 Computer

In 1967 my spouse and I attended a conference in Chicago on the then avant-garde topic of "Computer Assisted Instruction" or CAI. I returned to campus determined to make more use of computers in physics instruction. In one course, I asked the students to write a program to simulate putting a rocket into earth orbit. (*Rocket Trajectory Simulation*, D.S. Gale, Amer. Journal of Physics, 38, 1475, 1970.) The objective was to find, by trial and error, successful launch parameters. While the students learned a great deal, they were unsuccessful in finding a stable orbit. Once again I was reminded of the limitations of numerical approximations. I had one student contact me 30 years later to tell me how that assignment changed his future career to information technology.

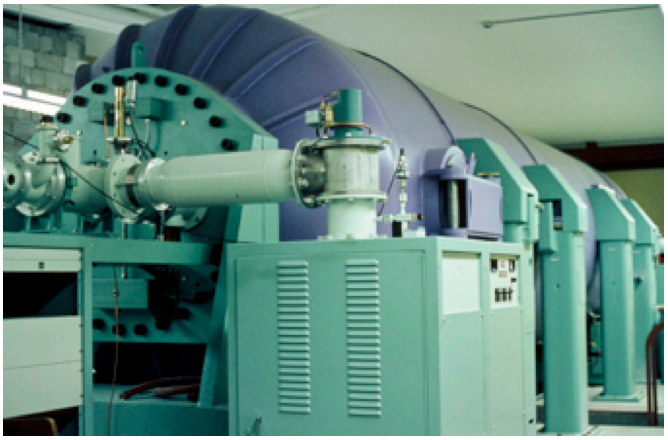
My spouse decided to do her master's thesis in education using CAI. Specifically, she wrote a 1620 assembly language program for golf instruction in physical education. The objectives were for players to learn which club to use in different situations and also to learn rules and proper etiquette. The course was then used for a small class of 10 students, who would individually sit at the console responding to queries from the computer. That, of course, required that the university dedicate the entire 1620 to that one application and behind the curtain there was an operator desperately feeding punched cards into the card reader trying to stay ahead of the student. The course was successful and popular with the students.

"Professional educators," however, did not in general embrace computing in the '60s. When my spouse submitted a paper based on her thesis to one of the education journals, it was rejected, because "it is common knowledge that computers are of little importance in education." She was so disgusted that she dropped out of the education community and joined the computing community. This "late adopter" pattern within the K-12 education community would be repeated in the early '80s when microcomputers were introduced.

The Early 1970s: Computing Becomes Ubiquitous for Research at Research Universities

In 1969 I decided to return to graduate school and complete a PhD in Physics. I selected Kansas State University (KSU) because they were installing a new tandem Van de Graaff accelerator, and I would be on the ground floor of building a new machine. One of the attractive aspects of the program was that graduate students ran the accelerator. We were the operators as well as the researchers. If something broke while we were using the machine, it was up to us to fix it.

Data acquisition was done largely with analog electronics much like what I used at Minnesota. The output, however, was punched tape, which we quickly converted to punched cards for further analysis on a digital computer.



KSU Tandem Van de Graaff
Photograph by Doug Gale

A small group of graduate students purchased army surplus cots that we used when we had multiple days of beam time. We lived on pizza and coffee. The tandem was painted purple, one of the school colors.

The late 1960s and early 1970s were a period of political and social unrest in the United States, much of it led by students. In December of 1968 a fire, believed to be set by an arsonist protesting the Vietnam War, had gutted Nichols Hall only a few hundred yards from the KSU physics building and computer center. In April of 1970 the computing facility at the University of Kansas, only 81 miles from Kansas State University, was bombed. (http://kuhistory.com/articles/this-is-no-joke/) The explosion and subsequent fire injured three students and caused extensive damage. Several months later, a science building at the University of Wisconsin was bombed and resulted in the death of a physics researcher. (http://en.wikipedia.org/wiki/Sterling_Hall_bombing)

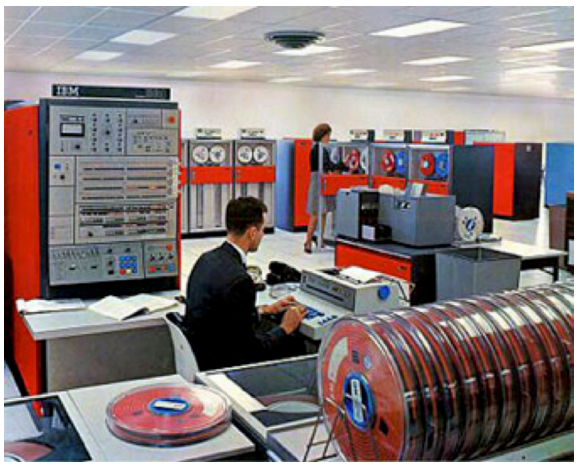
Since Kansas State's accelerator was located beneath the computer center, the graduate students working on the accelerator were concerned for their safety. The faculty felt that the radiation warning sign and locked door leading down to the accelerator were sufficient protection. We graduate students were less sanguine and prepared an evacuation route via a metal ladder that led to a hatch above the accelerator vault. We also had a stash of iron pipes for personal defense, if needed.

The theoretical portion of my dissertation required running two very different computer programs. The first was to determine the optical model parameters that best described the scattering of the colliding particles to first approximation. While the calculation was straightforward, it had to be done many times because the process basically consisted of guessing two variables, calculating the results, and then comparing the results with the experimental data. The difficulty was that there were multiple variable pairs that provided an acceptable fit. The challenge was to find the best pair. The process was much like finding the deepest valley in hilly terrain on foot. Running the program during the day on the university's IBM 360-50 was not an option because of the associated computing charges. However, the computer center made time available at night when the machine was idle. For over six months, I always had several programs waiting to be run overnight. It helped that my office was across the hall from the computing center, and my wife was manager of user services.

The IBM 360-50 was widely used at colleges and universities during the late '60s and '70s. The IBM System 360 "family" of computers used a consistent architecture for computers with different price points, and was very successful as it allowed customers to purchase a smaller system and migrate to a larger one as needed. The architect of the 360 family, Gene Amdahl, later founded the Amdahl Corporation, which directly competed with IBM. The development of the System 360 was a "bet the company" decision, as the development costs were twice IBM's annual revenue at the time.

The project manager, Fred Brooks, became famous for his book, *The Mythical Man-Month: Essays on Software Engineering*, that became required reading for new technology managers. The book is most remembered for Brook's Law, which states "adding manpower to a late software project makes it later."

The 360-50 could perform up to 48 kiloflops (floating point operations per second). This contrasts to Apple's 2015 MacPro, which peaked at 6 teraflops—more than a hundred-million times faster.



IBM 360-50
IBM Archives

The second program required in my dissertation research was much more computationally intensive. It involved solving coupled partial differential equations. The program could not be run on the university's IBM mainframe, even if they had been willing to give me the required time. The 360 had a 32-bit word length and researchers had found from hard experience that 48 bits were the absolute minimum to solve the equations numerically. (One researcher spent the better part of two years unsuccessfully trying to get the program to run on the System 360 architecture.) Since the program was running successfully on a CDC 3600 computer at Argonne National Laboratory, the decision was made to send me there to complete the calculations. When I left I was given $30,000 in computing funds—which I burned through quickly—and was given another $20,000. That too went quickly, and I requested the University send more money. They did, $10,000, but said that if that didn't get results to not come back. Fortunately, I was able to reach closure. The whole experience illustrates how researchers at the time were challenged to do the necessary calculations on the available computers.

Shortly before I graduated, the nuclear laboratory got a mini-computer, a Digital Equipment Corporation or "DEC" PDP-11, to replace its obsolete paper-tape system and much of the analog electronics. I made a deliberate decision not to get involved with the new computer and graduated on schedule.

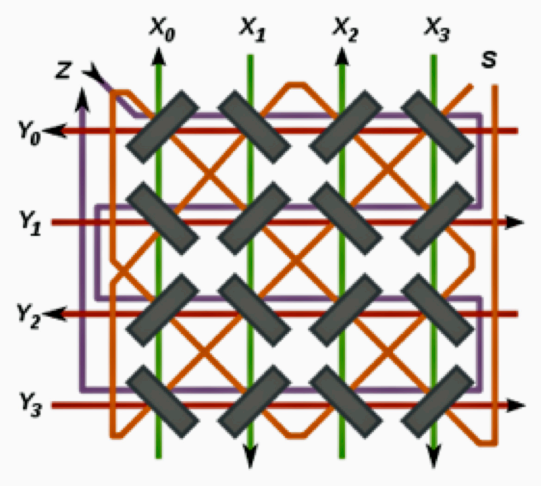
The Mid 1970s: Computing Becomes Ubiquitous at Smaller Institutions

In 1972 I accepted a position as Assistant Professor of Physics at East Texas State University (now Texas A&M-Commerce). While there I quickly became the "user from hell," consuming around 30% of the cycles on the University's IBM 360-50 mainframe, as I continued to analyze data I had compiled at Kansas State. While the university had a computer center, its primary focus was supporting administrative computing. Academic users were pretty much on their own. The center had in place a structure for supporting research computing without real dollar charges for times when the computer was not running administrative jobs. Other than myself, there was little research computing being done on the University mainframe.

The output from my calculations on the university's 360-50 were sent to a CDC computer somewhere in Maryland via satellite link, where more numerically intensive, coupled-channel calculations were done. I don't recall what the funding mechanism was to pay for those cycles. As a researcher, I didn't care where the computer was located or how it was paid for; all I was interested in was the results. This cavalier attitude later helped me understand researchers' attitudes when I became a computer center manager.

I spent the summers of 1973 and 1974 at Argonne National Laboratory as part of a visiting research program. They had a tandem Van de Graaff and my research was based on bombarding various exotic isotopes of uranium with lithium ions. Some of the uranium isotopes were quite rare in nature, and required "cooking" more common isotopes in a nuclear reactor for as long as two years.

The Argonne accelerator had a home-built computer system for a much more sophisticated form of data collection than I had used earlier. Rather than using analog circuitry to filter out signals from interactions that were not being studied, all signals for all reactions were recorded on the computer and the unwanted reactions filtered out in post-analysis on the computer. This "event recording" strategy meant that the stored data could be reanalyzed for other studies.



The primary memory was built from a surplus magnetic-core unit that was physically bigger than I was. Magnetic-core memory was the predominant form of memory at the time and consisted of a matrix of very small toroids (doughnuts), called cores, which could be magnetized either clockwise or counterclockwise by an electric current through wires in the center of the cores.

Core Memory Wiring Scheme
Illustration from Wikipedia

This diagram shows a simple scheme for magnetizing and subsequently reading the magnetic state of individual cores. Even though magnetic cores are no longer in common use, we still use the term "core memory."

After the event data was analyzed for the particular reaction being studied, the information was transferred to magnetic tape for further analysis.

The summer stays at Argonne were augmented by trips to Argonne during the school year. Typically, I would travel with a suitcase of magnetic tapes and the clothes on my back. It was during one of my stays at Argonne that I purchased my first pocket calculator for $100 (a bargain price at the time) at Marshall Field's Department Store. It could add, subtract, multiply, divide, and calculate square roots.

After the two summer appointments at Argonne, I continued to get beam time on their accelerator. Utilizing that time, however, was exhausting. Along with one or two students, I would drive straight through from Texas to Chicago, and then spend the next 24 to 48 hours at the accelerator. That would be followed by spending another 24 to 48 hours analyzing the event data and reducing it to a format that could be transferred to magnetic tape. Unfortunately, that could only be done on the home-brew computer at the lab. Thus, after three to five sleepless days, we would drive straight through to Texas and try to catch up on missed classes.

Upon return to campus, I would transfer data from magnetic tape to punched cards for easier use in running analysis programs on the local mainframe. As I recall, at one point I had over a million cards neatly stored in filing cabinets. Unfortunately, East Texas is humid and I had a serious problem with cards absorbing moisture and jamming in the card reader. At one point I had tables set up outside in the sun with thousands of cards drying out.

The grueling routine ultimately resulted in changing my research focus to something that could be done locally, and ultimately led to a career switch. If the Internet had existed, allowing data to be easily transferred and the experiments to be run remotely—and local campuses had access to high-performance computing resources remotely—I would probably have finished out my career as a nuclear physicist.

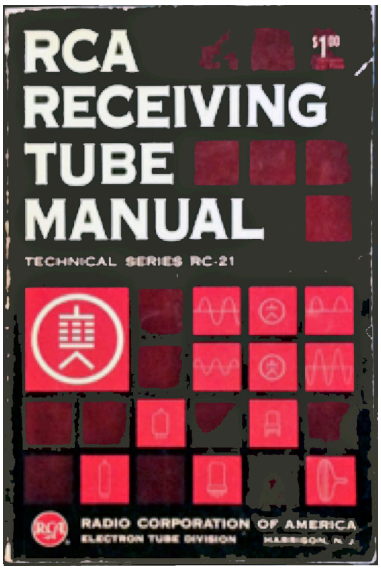
In the 1970s there were six U.S. mainframe computer companies: IBM, which had most of the market, and five others called the BUNCH, which was the nickname given to Burroughs, Univac, NCR (National Cash Register), CDC (Control Data Corporation), and Honeywell. In the 1960s, RCA and General Electric also made computers, and the group was characterized as "IBM and the Seven Dwarfs." Increasingly through this period, IBM dominated the market.

The early 1970s also saw the emergence of the minicomputer as a computing alternative for small companies or research groups. An early definition of a minicomputer was a machine whose cost was in the five-digit range. They had their own architectures and operating systems, and were commonly used for device-control and instrumentation. Digital Equipment Corporation, or DEC, was particularly prominent in higher education and research. The term "VAXinated" became a common joke in the research community.

The physics department at ETSU took advantage of my somewhat eclectic background and assigned me to develop an observationally based, introductory astronomy course, and a musical acoustics course targeting music majors. The department's budget was tied to credit-hour enrollment and such courses were essential to maintaining a viable and rigorous physics program. Both courses were very popular, even though I had a reputation as a tough grader.

I was also assigned to teach the undergraduate "electronics for scientists" course that was routinely taken by undergraduate physics majors. The focus of the course was instrumentation of experimental science using transistors and integrated circuits. When I began teaching the course, there was a shortage of textbooks on the subject, so I taught largely from my own notes.

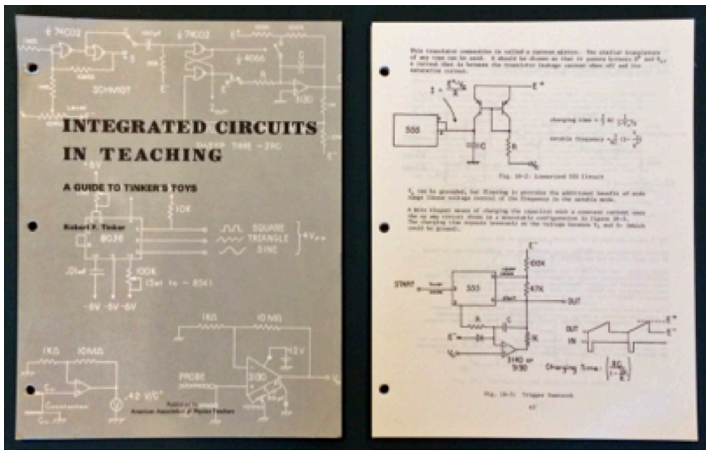
When I took the corresponding course as an undergraduate a decade earlier, the applied portion focused almost entirely on vacuum tube electronics, even though the transition from vacuum tubes to transistors was well under way. By the time I taught the course a decade later, vacuum tubes had almost entirely been replaced by transistors and integrated circuits.



RCA Vacuum Tube Manual
Photograph by Doug Gale

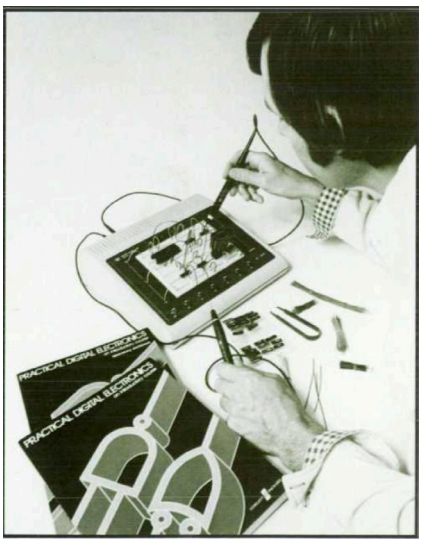
During this period I attended a "Chautauqua" Workshop developed by Robert (Bob) Tinker, of the Technical Education Research Center and the American Association for the Advancement of Science, and funded by the National Science Foundation. The workshop targeted scientists and engineers, and focused on using integrated circuits and digital logic for device control and data recording. Tinker and I shared a return airplane flight and spent the time talking about hardware modifications needed to convert an analog color TV to a digital color monitor -- something that was a rarity at the time.

Tinker's Chautauqua workshop was taught from mimeographed loose-leaf handouts and reflected the lack of commercial textbooks.



Tinker's Toys
Photograph by Doug Gale

By 1974 I had begun introducing simple digital electronics into the undergraduate "electronics for scientists" course, and shortly afterwards began teaching a one-semester "digital electronics" course organized around the newly released Hewlett-Packard Model 5035T Logic Lab. The course was popular with computer science students as well as science students.



HP had the challenge (and opportunity) of teaching an army of analog-trained electrical engineers about digital electronics. They developed a training course complete with lab hardware and instruction manuals. Students would construct various digital circuits on a plug-in board. The lab included an assortment of digital measurement and probe tools. The objective, of course, was to sell HP test gear.

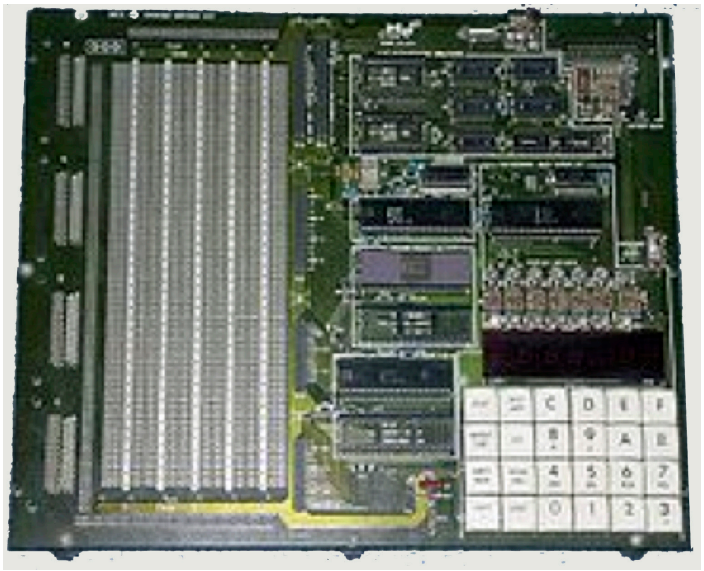
HP Model 5035T Logic Lab
Hewlett-Packard Archives

The decade also marked the emergence of the microprocessor, in which an entire central processing unit (CPU) was implemented on a single chip. That, combined with the increasingly less expensive 7400 series logic chips led to the development of the microcomputer, a term used in the 1950s by the science fiction writer Isaac Asimov. The marketplace was crowded with companies developing new microprocessors and support chips. The Zilog Z-80, MOS 6502 Motorola 6800, and Intel 8080 chips in particular led to the development of a wide range of consumer computers, including the Altair 8800, Radio Shack TRS-80, and Apple I.

Hewlett-Packard wasn't the only company to market training materials for their digital technology. Other chip vendors introduced educational "training kits."

Intel sold a number of heavily subsidized "kits" to encourage users to adopt their new products. This picture shows their bubble memory prototype kit, which showcased their 1-megabit bubble memory module. The kit included a printed circuit board and a bag of parts.

Bubble Memory Prototype Kit
Photograph by Doug Gale



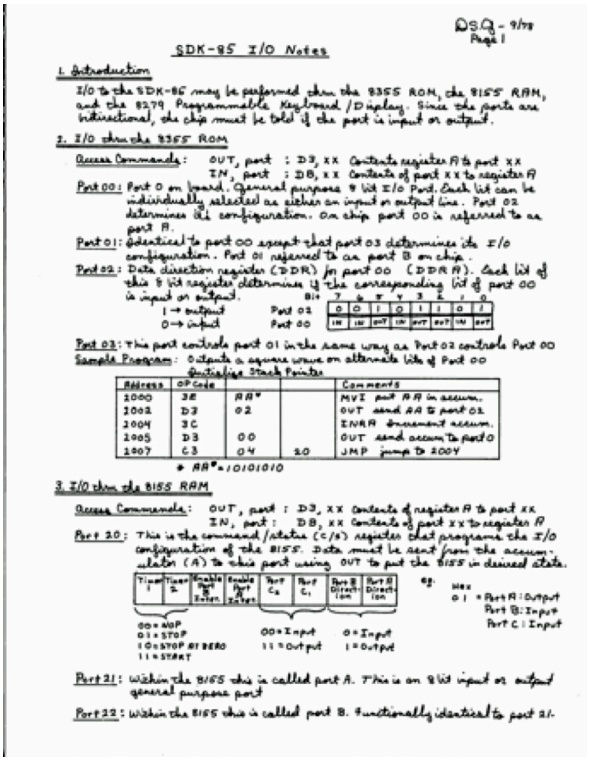
Intel provided a System Development Kit (SDK) when they launched a new microprocessor. It was a natural follow up to the digital logic course to begin teaching a microcomputer-based course on the SDK-85 in 1976.

The SDK-85 was a single board microcomputer system using the Intel 8085 chip, which was based on the popular 8080. It came as a kit and the completed circuit board included a 6-digit LED display, a simple IO bus, a 24-key keyboard for direct insertion, examination, and execution of a user's program, and a large space for wire wrap components.

SDK-85 1Single Board Computer
Photograph from Wikipedia

The course was organized as a "hands on" graduate

workshop that met for 3 hours once a week in the evening, to allow local professional engineers to attend. The students would assemble the single board computer the first evening and were given assignments of increasing complexity as the course progressed. The computer could be programmed directly from the keypad or through a serial port. As the students progressed, the programs became more sophisticated and included attaching various input and output devices.



DSG - 9/78
Page 1

SDK-85 I/O Notes

1. Introduction

I/O to the SDK-85 may be performed when the 8355 ROM, the 8155 RAM, and the 8279 Programmable Keyboard/Display. Since the ports are bidirectional, the chip must be told if the port is input or output.

2. I/O thru the 8355 ROM

Access Commands: OUT, port : D3, XX Contents register A to port XX
IN, port : DB, XX Contents of port XX to register A

Port 00: Port 0 on board. General purpose 8 bit I/O Port. Each bit can be individually selected as either an input or output line. Port 02 determines bit configuration. On chip port 00 is referred to as port A.

Port 01: Identical to port 00 except that port 03 determines the I/O configuration. Port 01 referred to as port B on chip.

Port 02: Data direction register (DDR) for port 00 (DDRA). Each bit of this 8 bit register determines if the corresponding bit of port 00 is input or output.

1 → output
0 → input

Bit 7 6 5 4 3 2 1 0
Port 02 0 0 1 0 1 1 0 1
Port 00 IN IN OUT IN OUT OUT IN OUT

Port 03: This port controls port 01 in the same way as Port 02 controls Port 00

Sample Program: Outputs a square wave on alternate bits of Port 00 (initialize Stack Pointer)

Address	OP Code			Comments
2000	3E	AA*		MVI put AA in accum.
2002	D3	02		OUT send AA to port 02
2004	3C			INR A increment accum.
2005	D3	00		OUT send accum to port 0
2007	C3	04	20	JMP jump to 2004

* AA = 10101010

3. I/O thru the 8155 RAM

Access Commands: OUT, port : D3, XX Contents of register A to port XX
IN, port : DB, XX Contents of port XX to register A

Port 20: This is the command/status (C/S) register that programs the I/O configuration of the 8155. Data must be sent from the accumulator (A) to this port using OUT to put the 8155 in desired state.

Timer I | Timer II | Enable Int. Timer | Enable Port A Ints. | Port C_2 | Port C_1 | Port B Direction | Port A Direction

00 = NOP
01 = STOP
10 = STOP AT ZERO
11 = START

00 = Input
11 = Output

0 = Input
1 = Output

eg: Hex 01 = Port A: Output, Port B: Input, Port C: Input

Port 21: Within the 8155 this is called port A. This is an 8 bit input or output general purpose port

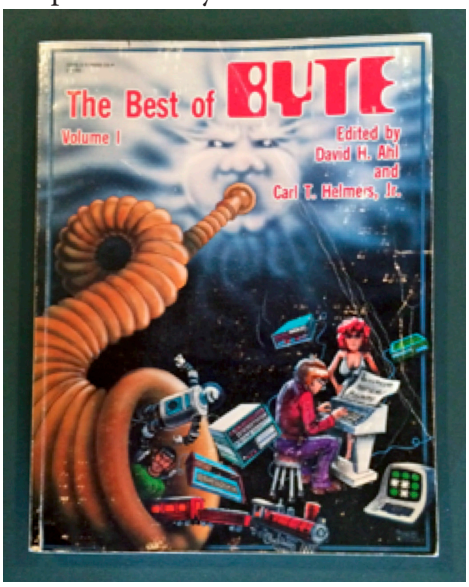
Port 22: Within the 8155 this is called port B. Functionally identical to port 21.

Teaching Materials
Photograph by Doug Gale

When I first offered the course, there weren't any suitable textbooks so teaching was done from manufacturers' technical literature and handwritten lecture notes, such as the ones shown here. Typical labs included the construction of a digital thermometer and the control of a model radar tower made from Erector Set parts borrowed from my son. More sophisticated labs employed analog-to-digital and digital-to-analog conversion circuits. By the end of the decade textbooks began to become available, but having written a Laboratory Manual while at St. Cloud, I had no desire to undertake converting my notes to a textbook.

The course proved to be popular not only with physics and engineering students, but with computer science students as well. So much so that the course credit could be taken either in physics or computer science, and led to a joint-degree program offered by the two departments with courses in microelectronics and microcomputers. A student could major either in physics or computer science and minor in the other discipline. (*Integrating Microcomputers and Microelectronics into the Physics Curriculum*, D.S. Gale, Amer. Journal of Physics 48, 498, 1980.) One thing led to another, and I began teaching a graduate Telecommunications course in the Computer Science Department —something that would prove useful later in my career.

"Amateurs" and "hobbyists" played a vital role in microcomputing during the mid- and late 1970s. Some were young entrepreneurs like Steve Jobs and Bill Gates, who went on to found major corporations. Others were working engineers and scientists who wanted to exploit the opportunities presented by the new technology. This period could be viewed as a precursor to the early 21st-century "maker movement," which enjoyed engineering–oriented pursuits such as electronics, robotics, and 3-D printing, in addition to more traditional arts and crafts. Many of today's computer science luminaries gained their experience at places like Bell Labs and the Rand Corporation rather than at academic institutions.



The microcomputing culture had its own "do-it-yourself " magazines and paperback books.

Byte Magazine
Photograph by Doug Gale

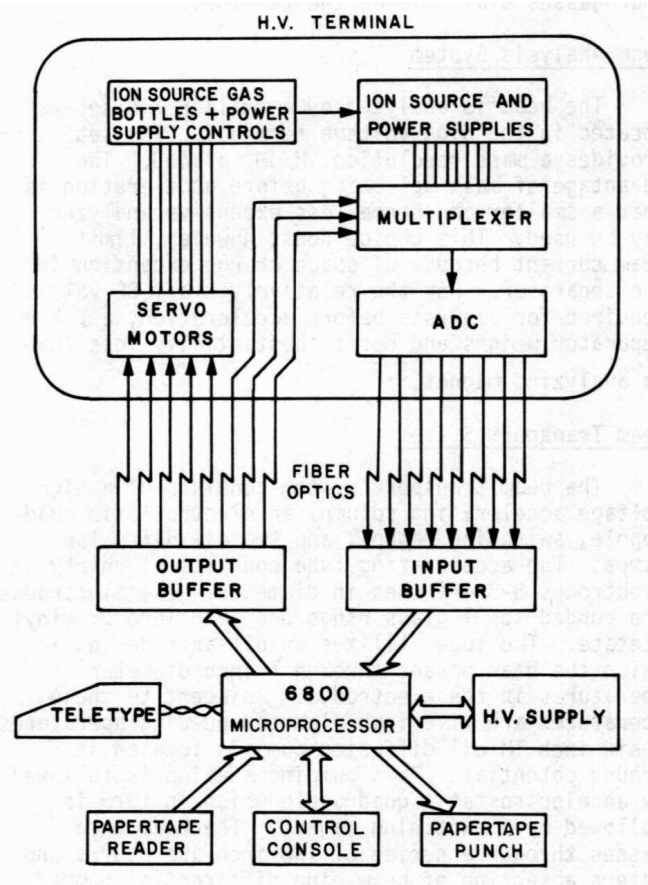
Byte magazine was a monthly publication dedicated to microcomputers. It was widely read by both computer hobbyists and professionals, and its articles covered both hardware (complete with wiring schematics) and

software (usually machine or assembly language). *Interface Age* was another popular magazine that targeted the "home computerist."

Cookbooks
Photograph by Doug Gale

This period also marked the publication of a wide range of "cookbooks" that stressed practical electrical circuits to supplement the large amount of similar "how to" manuals being provided by the chip vendors. Again, these "cookbooks" were an eerie precursor to the cut-and-paste approach favored by the 2010's maker culture.

Although having fun playing with digital electronics, I had not abandoned physics research. In 1975 I began the design and construction of a 300-kilovolt heavy ion accelerator. The scientific objective was to study atomic reactions on the surface of material by bombarding them with heavy ions; the personal objective was to move my research closer to home. The design of the accelerator, particularly the ion optics of the accelerating column, involved extensive numerical modeling and was the focus of one of my graduate student's research.



The accelerator had several unique features at the time. The first was the use of then state-of-the-art, ultra-high vacuum techniques that totally avoided the use of any organic materials. The second was the use of fiber optics to transmit signals to and from the high-voltage terminal. The fiber optic connectors that became common in the 1990s were not available. Fiber optic strands were hand polished and then epoxied onto LED transmitters and detectors. The third was the extensive use of digital electronics and microcomputers for telemetry and voltage control.

300 kV Heavy Ion Accelerator Telemetry
Scientific & Industrial Applications of Small Accelerators, 4th Conference, 1976

When I ran out of grant money to purchase a needed piece of data acquisition electronics for the 300 kV heavy ion accelerator that was halfway to completion, one of my students, who was also a teaching assistant in the microcomputer course, suggested that we build a computer system and I/O electronics that would replicate the functions of the dedicated electronics we could not afford. In an act that can only be viewed as hubris, I agreed.

We used the astronomy dark room to make printed circuit boards. We added stepper motors to the milling machine in the physics shop to drill holes in the boards. We cannibalized military-surplus disk drives for memory. We knew enough to be dangerous -- and very creative. At the time, I jokingly said I didn't know what an operating system was until I wrote one.

In the summer of 1979 I was offered the position of Director of Decentralized Academic Computing Services at Cornell University. Because of teaching commitments, I was not able to move to Ithaca until December of that year. Although I had illusions at the time of continuing my physics research, in fact it marked the end of my career as a research scientist and the beginning of my career as an information technologist and research administrator.

Chapter 2:

The Early '80s and the Microcomputer Revolution

By the 1980s computing was well established in higher education. Leading institutions were spending approximately 5% of their budget, exclusive of ancillary operations such as dormitories and athletics, on computing. Expenditures were more or less evenly split between academic and administrative applications. Administrative computing was done almost entirely at a central computing facility; most academic computing was done there as well. Faculty and students were still learning about the technology, but applying it in new and novel ways. However, change was on the horizon.

The Cornell Years: Distributed Academic Computing Services (DACS)

The early 1980s marked the beginning of a fundamental shift in computing. The introduction of the minicomputer a few years earlier made it possible for a small department to own its own computer, separate from the centrally managed corporate mainframe. Similarly, the introduction of microcomputers made it possible for individuals to own a personal computer.

The fundamental paradigm shift was not one of hardware but of management control: from centralized to decentralized. The term "microcomputer" fell out of favor in the mid-80s and was replaced with "personal computer or PC" reflecting that the fundamental shift was not based on computing power but on management control.

Cornell was one of the first universities to recognize that a paradigm shift was underway and decided in early 1979, under the leadership of Douglas VanHouweling, Director of Academic Computing, and J. Robert Cooke, Chair of the Faculty Computing Committee, to recommend the creation of a separate division within Cornell's computer services unit to support decentralized computing. Provost Kennedy issued a policy directive in July of that year creating DACS, for Distributed Academic Computing Services, within the Academic Computing unit. The unit was charged with providing consultation and support for those units owning or planning acquisition of computers.

The first director of the DACS was Alison Brown, who quickly decided she would rather not be burdened with administrative responsibilities, and recruited me in the fall of 1979 to be her replacement and boss. My wife, kids, and I arrived in Ithaca two days before Christmas in 1979.

When Ken King became Vice Provost for Computing at Cornell in late 1980, DACS was moved from the academic computing unit to directly reporting to the Vice Provost, and the name was changed to Decentralized Computing Services, or DCS, because administrative users were following the same trends towards decentralization. King cites Cornell's recognition of this paradigm shift as one of the reasons he accepted the position at Cornell. At the time, computing centers typically had four branches: academic computing, administrative computing, systems, and operations, and had a centralized management culture.

A Rude Awakening

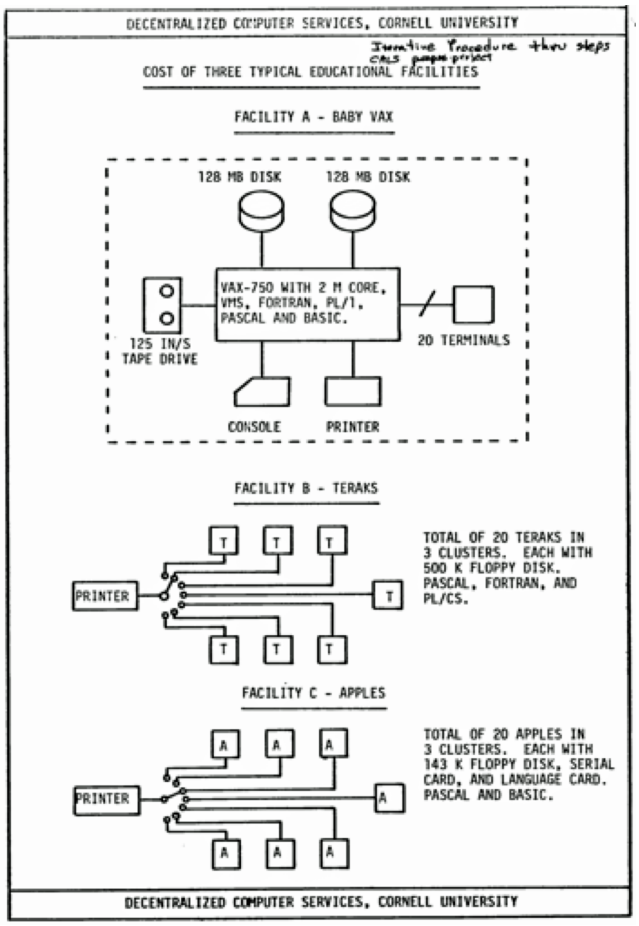
Shortly after arriving at Cornell I was asked to participate in the selection of a new minicomputer for the Business School. They were considering two architectures, both from Digital Equipment Corporation, a DECSYSTEM-20 and a VAX. I quickly realized that, while I knew a lot about microcomputers, I was a real neophyte when it came to minicomputers. The decision boiled down to choosing between an established architecture

that had a large body of mature software, and a newer architecture with relatively little application software —the kind of decision that required experience I didn't have.

My response was to attend a two-day workshop on minicomputers and microcomputers in Washington, DC, to spend a lot of time talking to facility managers, operations staff, and minicomputer owners, and then to develop a one- to two-day workshop syllabus of my own: *Guidelines for the Selection and Operation of Minicomputers and Microcomputers.* I subsequently gave presentations and workshops of various lengths targeting different audiences at a variety of venues, both local to Cornell and nationally, for meetings such as the National Educational Computing Conference and the American Association of Physics Teachers, using a subset of the complete syllabus. The syllabus had seven sections:

1. Introduction: A Brief History of Small Computers and an Explanation of Some Common Elements in "Computer Talk"
2. The Relative Advantages and Disadvantages of Micro/Mini/Midi/ and Maxicomputers
3. How to Determine What Kind of Computer is Best for You
4. A Brief Survey of Hardware and Software
5. How to Select and Acquire a Small Computer System
6. Warranty, Maintenance, and Operating Costs
7. Future Trends

It is important to remember that most of the attendees at these presentations had little experience with computing, and for those that did, it was usually restricted to using a mainframe or a home computer. Few understood the tradeoffs between mainframes, minicomputers, and microcomputers, and virtually none had an understanding of the total cost of ownership or maintenance and support costs and issues.



The first two sections covered basic definitions and the relative advantages and disadvantages of Micro/Mini/Midi/ and Maxi computers. Topics included cost, hardware and software capability, and the number of users that could be supported.

The objective of the next three sections was to assist the attendee in selecting the appropriate kind of computer. Included was a brief survey of then-current hardware and software, costs for hardware and software, as well as how to prepare and evaluate an RFP or RFQ. Benchmarks for various CPU processors and operating systems were presented in detail.

The fifth section was an attempt to introduce to attendees a sense of how much it really cost to operate a computer and/or a computer facility. This section included comparisons of the costs of running typical student facilities, such as a baby VAX (VAX-750), a cluster of 20 Teraks, and a cluster of 20 Apple IIs. I spent a considerable amount of time in this section, because I had quickly discovered that many faculty attendees thought they could avoid "excessive" mainframe charges by purchasing a bunch of microcomputers from the local computer store, without giving thought to things like software, maintenance, power, cool-

ing, site preparation, and printers. And they rarely included funding for staff support. I'm not sure how successful I was in convincing attendees that there was no such thing as a free lunch.

Benchmarks

But users wanted more than theoretical background; they wanted concrete advice, such as "buy product A." In the early 1980s the microcomputer market was in a state of flux. There was a wide array of CPU chips from multiple vendors. There were few, if any, operating system or software standards. In DACS we spent a great deal of time benchmarking various systems in an attempt to provide our users with the best advice possible. This is a benchmark summary document I prepared in the fall of 1980; the subjective opinions were mine. For obvious reasons, it was always presented "in-person."

A Comparison of Some Popular Microcomputers. D.S. Gale, 10/80. The numerical "ratings" are the "seat-of-the-pants" opinion of the author.

Ratings Scale: 1=Terrible 3=Average 5=Excellent 2=Poor 4=Good

Computer	CPU	Word Size	RAM	Base List Price	Typ. List Price	Princ. Prog. Lang.	# inst. to Spr '80	Primary Uses	Operating System	Evolutionary Future	"Power"	Expandability	Local Software Support	Natl. Software Support	Local Hardware Support	Natl. Hardware Support	Reliability
Apple II and Apple II Plus	6502	8 bit	16K-48K	1.2-1.5K	$2K 32K minidisk CRT	BASIC PASCAL Assemb.	55K	Ed., bus, home/hobby	DOS	2 (CPU)	3	3	3	3	3	3 I/O periph.	3
Commodore PET	6502	8bit	8-32K	0.8-1.3K	$1K 8K cassette CRT	BASIC Assemb.	107K	Ed., home/hobby	in ROM internal	2 (CPU)	2	2	1	2	1	2	2
Cromemco Z-2	Z-80A	8-bit	4-64K	1.3-2.8K	$4K 16K minidisk CRT	BASIC FORTRAN COBOL Assemb.	NA	Small Bus, Control, Prof.	CDOS (CP/M compat.)	3 (CPU)	3	4	3	3	3	3 I/O periph.	4
Heath H9	6080A	8-bit	8-56K	0.5-1.6K	$1.6K 16K minidisk CRT	BASIC Assemb.	12K	home/hobby, ed, bus	cassette OS	2 (CPU)	2	2	1	2	1	2	2
Hewlett-Packard HP-85	HP nMOS	8 bit	16-32K	3.2K	$3.2K 16K RAM 2.8K tape Printer CRT	BASIC	NA	Sci/Eng. Prof.	HP-85 OS internal	4 (package)	3	4	2	3	2	5 I/O periph Service	5
TRS-80 Model I	Z-80	8 bit	4-48K	0.5-1.5K	$1.5K 16K minidisk CRT	BASIC FORTRAN Assemb.	150K	home/hobby, ed, bus	TRS-DOS	2 (good package)	3	3	2	3	1	5 Service	2
TRS-80 Model 2	Z-80A	8 bit								3 (CPU)	3	4	2	3	1	5 Service	3
Texas Inst. TI 99/4	9900	16 bit	16K	1.1K	$2.1K 16K minidisk	BASIC Assemb	NA	home/hobby, ed, bus	internal in ROM	2 (package)	3	1	1	2	1	3 Service	2
Terak 8510	LSI-11	16 bit		$6K	$6K 56K RAM ½MB disk CRT	PASCAL FORTRAN RT-11 APL, BASIC	NA	Sci/Eng. Ed., Prof.	PASCAL, RT-11	4 (CPU, software)	4	5	4	4	3	5 I/O periph Service	4

The list includes neither the Apple MacIntosh nor the IBM-PC, as they had yet to be introduced.

Personal Computers (PCs)

Complete microcomputer systems began being mass marketed in 1977 with the advent of the Apple II, Radio Shack's TRS-80, and the Commodore PET. These "home computers" usually ran the BASIC computer language and used a cassette tape recorder for external memory. Although the Apple II was popular in the educational community, most universities and almost all businesses turned up their respective noses at these machines. They weren't "serious computers." Two events changed this mindset. The first was the release of VisiCalc, the first spreadsheet computer program for microcomputers, in 1979. Overnight, it changed the Apple II from a hobbyist "toy" to a serious business tool. I recall wanting an accounting system to parallel the university's official accounting system. I was seeking a way to project anticipated events into a long-range

financial planning model. I was told the system would take 6 months to develop on the university's mainframe database. Instead, I checked out an Apple II with a copy of VisiCalc. That evening I learned VisiCalc and wrote the planning model.

The second event was IBM's introduction of the IBM Personal Computer, or PC, in August of 1991. The introduction gave microcomputers instantaneous credibility. In fact, the term "microcomputer" dropped from users vocabulary almost overnight and was replaced by the term "personal computer." Big Blue had spoken.

One of the most enjoyable aspects of the job was the opportunity to play with all the new stuff.

Introduced in 1981, the Osborne I was one of the first "portable" computers. The front hinged down to reveal a 5-inch screen, keyboard and dual, single-sided floppy disk drives. I can still recall walking across campus with this 23-pound brute.

Osborne I
Photograph from Wikipedia

Nibbles

Initially DACS was quite small: myself, Alison Brown (a systems' guru), Karen Friedman (a software guru), and Joe Nazar (a hardware technician). The challenge was dealing with an overwhelming number of faculty and staff interested in using microcomputers, but without any experience in personal computing. At one point, in desperation, we resorted to waiting until noon to open the office because of the numbers of faculty and staff waiting to talk to someone in person. To leverage our limited resources, we created a one-page folksy newsletter called "nibbles," which covered the issues that people were concerned about at that time.

dacs

nibbles

Decentralized Academic Computer Support ... Orin Hall ... Cornell University ... Ithaca, New York 14853 ... (607)256-4981

introducing dacs...

DACS is an abbreviation of Decentralized Academic Computer Support. We are a group within Computer Services who provides support for faculty and staff either operating or planning acquisition of decentralized computing facilities. For most people this means their own minicomputer or microcomputer. We provide help to users in hardware and software selection, analyzing, site preparation needs, and services such as software and hardware design, development, and maintenance. Some services are free, some are not.

Come to G-46 and meet the staff. We are:

Doug Gale: Assistant Director, Computer Services. Expertise is primarily small computer hardware and interfacing.

Alison Brown: Senior Software Specialist. In charge of providing software support for small computer systems.

Karen Friedman: Software Support Specialist. Software support for Terak system and small computer graphics.

Joe Nazar: Hardware Support Specialist. Hardware support for Teraks and other small computer systems.

We have identified the following goals as things we would like to do. Our plan is to go as far as we can, with whatever resources we have, to meet these goals. Your comments and suggestions - regarding both priorities and content - are welcome.

1. Provide Cornell faculty and staff assistance in the selection, acquisition, and operation of hardware and software systems.
2. Provide Cornell faculty and staff assistance in site planning and preparation.
3. Provide full hardware support for a limited number of small computer systems. This support would include: maintenance, data acquisition and device control peripherals, and input/output peripherals.
4. Provide full software support for a limited number of small computer systems. This support will include: software for data acquisition and manipulation, software for device control, and communications software between supported computers.
5. Provide demonstration facilities for all supported systems.
6. Develop and support facilities that provide a one-semester experience in personal computing to a significant number of Cornell undergraduates.
7. Provide training to Cornell faculty and staff so that they can design and implement suitable small computer systems on their own.
8. Provide specialized resources to Cornell faculty and staff for the development of small computer systems. This facility would include specialized computer hardware, test equipment, and design facilities that are expensive, necessary, but only sporadically used by an individual single user.
9. Provide custom hardware design and interfacing services for Cornell faculty and staff.
10. Provide custom software design and implementation services for Cornell faculty and staff.

...and nibbles

nibbles is a one-page newsletter whose purpose is to provide a vehicle of communication between small computer users. It will contain an informal potpourri of notes related to small computers and their use. The catch is that you have to fill out the attached form to stay on our mailing list. It's our way of knowing what you're interested in and also of minimizing our production costs.

Circle October 15

The Mini/Micro Seminar Series will resume this fall on a monthly basis. The first one is Wednesday, October 15 at 4 p.m. in G-14 Orin Hall. Ahmed Mahjoub from Philips Laboratories, one of the authors of the new Modula compiler for the UCSD Pascal operating system, will talk about the Modula language and the Philips Laboratories' implementation. Modula is similar to Pascal but offers some advantages for realtime processing. The next two seminars are Wednesday, November 12, and Wednesday, December 10. Speakers will be announced. Coffee and doughnuts will be served at all three seminars.

A Publication of CORNELL COMPUTER SERVICES. September 1980 Volume: 1 Number: 1

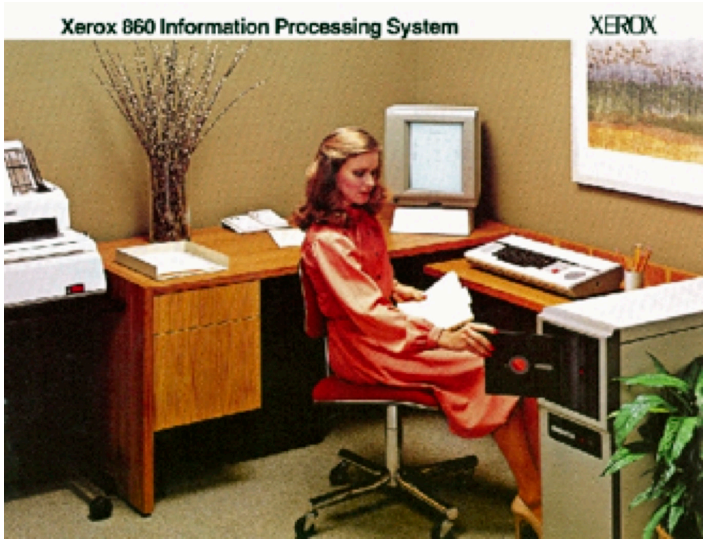
The Great Debate

Shortly after I arrived at Cornell I was asked to participate in a traveling debate sponsored by the business-oriented Data Processing Management Association (DPMA), which changed its name to the Association of Information Technology Professionals in the 1990s. The debate was between decentralized computing, using minicomputers and microcomputers, and centralized computing using mainframes. I would argue for a decentralized computing environment, while a member of the hosting DPMA chapter would champion centralized computing. The discussions were "lively" and underscored that the fundamental shift was centralized versus decentralized and not mainframe versus minicomputer or microcomputer.

Word Processing

Word-processing in the late 1970s and early 1980s was done on dedicated and expensive stand-alone machines, such as the Wang 1200 and Xerox 860, because the hardware and software capabilities of available microcomputers, such as the Apple II, were very limited. My wife, who was part of the administrative computing unit at Cornell, provided Xerox 860 technical support to the College of Arts and Sciences, before moving on to implementing a student information system.

The Xerox 860 was a high-end word processing system that cost around $14,000 and was popular in the early 1980s. It featured "WSIWYG" or "What You See Is What You Get" software and could display 70 lines of 102 characters on the monitor.



Xerox 860 Word Processor
Photograph from Wikipedia

By the mid-1980s there were dozens of WSIWYG word processing packages, such as "WordStar" that ran on personal computers like the IBM-PC, and the market for stand-alone word processors collapsed.

The Apple Lisa

The Apple Lisa is little remembered but played an important role in the evolution of microcomputers. Steve Jobs had visited Xerox's Palo Alto Research Center in 1979 and was very impressed with the mouse-driven graphical user interface (GUI) that they had developed. When he returned to Apple, they began development of a Motorola 68000-based computer to transform PARC's research into a marketable product. The result, the Lisa, was introduced in January of 1983. The unique feature of Lisa was the integrated package of software applications: LisaWrite, LisaCalc, LisaDraw, LisaGraph, LisaProject, LisaList, and LisaTerminal—and an integrated package that was not available elsewhere. A Lisa was on my desktop at Cornell.

The Apple Lisa
Photograph from Wikipedia

Priced at $10,000, the Lisa was considered by the market place as "too expensive." Ironically, an equivalently equipped IBM-PC XT (introduced in March of 1983) was also about $10,000, and the software applications were not integrated. Relatively few Lisa computers were sold, although at Cornell they were adopted by the Cornell School of Hotel Administration as part of a pioneering system for hotel administration.

Microcomputers in Instruction: The Terak Story

Cornell was one of the first universities to embrace microcomputers in undergraduate instruction. The Computer Science faculty were strong advocates of structured programming and foreswore languages such as Basic,

that were used in most microcomputers at the time. Tim Teitelbaum, in the Computer Science Department, had developed a PL-1 synthesizer that ran on the university's IBM 360-168 mainframe and was used in a course required of all engineering and most science freshman. The synthesizer created a "padded cell" environment that simply did not allow the student to write unstructured code.

See: An Oral History Conversation: The Paradigm Shift from Centralized to Decentralized Computing at Cornell. King, Kenneth M.; Cooke, J. Robert; Teitelbaum, Tim; Gale, Doug. http://hdl.handle.net/1813/41195

In the late 1970s, to alleviate strain on the university's mainframe facilities, Tim developed a version that could run on the Terak 8510, a microcomputer. The Terak was considerably more powerful than contemporary microcomputers based on Intel, Zilog, and Motorola CPUs. One downside was that they cost around $5500, as I recall. Another was the 40+ pound weight of the base-processing unit alone.

Cornell made a major investment in Terak computer rooms to service the introductory programming course that Tim taught and was required of all engineering students. DACS was charged with running the facilities. The problem was that there were never enough Teraks. The days before an assignment was due were chaos. The undergraduates hired to run the facilities and ration use were faced with students concerned about passing a required course. We could measure how crowded the facilities were by counting the number of fights that broke out as students vied for machines. My efforts to get the various Deans to commit funds to expand the facilities were generally unsuccessful. It was always someone else's students who were using the facilities.

Terak 8510/a
Photograph from Wikipedia

This was a new variant of the old problem of how do you pay for computing. The new twist was that we didn't have an accounting mechanism that would allow us to use the somewhat-effective mechanisms that we developed in the '60s and '70s. Other than paper sign-in sheets maintained by the computer room assistants, we didn't know who was using a Terak. And once the student was on a machine, we didn't have a clue as to what he or she was doing. We did try to manually limit each student to an hour of machine time. Before committing additional funding, the Deans wanted evidence that it was their students who were using the machines and how much they were using them.

The Teraks were built with a DEC LSI-11 processor and could run a basic version of UNIX, as well as UCSD Pascal. It had a graphic's processor and could display both text and graphics in monochrome.

The Apple Logon Machine Fiasco

Our attempts to use the older paradigm of "charging by the drink," by adding an accounting function to microcomputers that duplicated time-sharing's logon procedure, was a complete flop. Next to each Terak in our labs we placed a small box with three lights: red, yellow, and green. Each box was hardwired to its associated Terak's interrupt line, and to a central Apple II microcomputer that was similarly connected to each Terak in the room. We developed software for the Apple, using an obscure structured language developed in Europe, to provide authentication, data recording, and most importantly rationing functions. When a student wanted to use a Terak, they would logon to the Apple II logon machine by entering their name and password. Since the memory capacity of the Apple II was limited, the password was not verified against a database. Rather the

student's password was an encrypted version of the student's name. The encryption algorithm was our secret. When a machine was available, the student's name was flashed on a monitor, the associated Terak was enabled, a 60-minute timer was started, and the box beside the Terak showed green. When a student had 5 minutes left the light turned yellow, and when his or her time was up, the light turned red and the machine disabled. Brutal, but effective.

It actually worked—as long as the load on the system was light to moderate. For reasons we could never discover, the Apple software we had developed would crash when the load was high—typically the evening before an assignment was due. We succeeded in making the problem worse.

The solution to the problem was the gradual migration away from a "pay by the drink" funding model, the adoption of less expensive Apple MacIntoshes for the introductory computer science course in 1984, and the widespread ownership of personal computers.

Maintenance

In the 1980s computer maintenance was a major expense. The yearly maintenance cost for a mainframe typically was in the six digits. Maintenance was also a major expense for microcomputers, particularly in the early 1980s. Because of the mild summers in Ithaca, many of the buildings at Cornell were not air-conditioned, and temperature was simply regulated by opening or closing windows. Unfortunately, a Terak and a person each put out around 150 watts of heat, so a 30-student lab generated approximately 9,000 watts of heat. One lab in the engineering building was particularly troublesome. It only had a few windows and on a calm day it would become uncomfortably warm, and the Teraks would start failing as their integrated circuit chips overheated. Tom Everhardt, the Dean of Engineering and all around fine fellow, and I went 'round and 'round on the need to air-condition the room. I went so far as installing thermistors on the surface of the CPU chips to monitor and document the fact that chips were failing because of the heat. In 1980, a room filled with microcomputers and students was something of a novelty.

At $10,000 a pop, the Teraks were not a consumer item that you could buy at the local computer store. The company was small and did not have a network of national service centers. To repair one, you had to box it up and send it back to the factory in Arizona. (Remember, the base unit weighed more than 40 pounds.) That was expensive and had a turnaround time measured in weeks. We quickly decided that it would be cheaper to send one of the DACS technicians to Arizona to work at the factory for several weeks and learn how to repair them, and then maintain a local inventory of parts. Upon the technician's return from Arizona, we borrowed a VHS video recorder and on a Saturday afternoon, fueled with takeout sandwiches and a six pack of beer, put together a cheesy "How to Fix a Terak" video. Although it was originally intended for our own internal use, as other universities learned of its existence it became widely requested.

Email, The Killer App

Although email was being widely used by the small ARPAnet (Advanced Research Projects Agency Network) community, and within a few companies such as IBM, it was just emerging in the academic community in the early 1980s. There were two technical challenges. First, how do you exchange information between two computers, and second, how do you present that information to an end user on a computer terminal.

Cornell's primary computing system in the early 1980s was an IBM 360/168 running the "Virtual Machine" or VM operating system, which allowed the machine to run multiple guest operating systems, such as CMS (Conversational Monitor System) or MVS (Multiple Virtual Storage), each within its own virtual machine. One such system was RSCS (Remote Spooling Communications Subsystem) that was designed to exchange files between remote systems and users.

Using RSCS as the interconnecting mechanism, Ira Fuchs at the City University of New York and Greydon Freeman at Yale University created BITNET, which originally stood for "Because Its There," in 1981. (http://en.wikipedia.org/wiki/BITNET) BITNET was a point-to-point "store and forward" network. Files were temporarily stored on one computer and then transmitted to another computer at a later time over a leased telephone line. A handful of other universities, including Cornell and the University of Pennsylvania, quickly joined the new network.

The business model was simplicity itself. If a university or college wanted to join BITNET, they paid for a 9.6 kbps (thousand bits per second) leased telephone line from their institution to a BITNET institution, and agreed to let two or more other institutions connect to them.

The second challenge was to present the information to a person on a terminal in a useful way. Typically, each campus wrote its own email editor, and at Cornell it was Steve Worona who wrote the Cornell editor. It was only sometime later that email editors became standardized.

At the time, there were no vendor-independent, communications protocols for exchanging files between computers. SMTP or Simple Mail Transfer Protocol for an IP-based system wasn't developed until 1982. BITNET was, initially at least, limited to IBM mainframes running RSCS. Even with that restriction, BITNET grew rapidly, and at its peak connected almost 500 organizations throughout the world. Throughout most of the 1980s, it was the lingua franca of the higher-education community.

BITNET, and higher education's role in its growth, was more important than simply being a part of email's evolution. It introduced the utility of email to hundreds of thousands of students, who spread out to other universities and industry, and created marketing demand for what was to become the Internet. With the passage of time, BITNET came to mean "Because Its Time."

The new technology was not without its growing pains. One day some system programmers from the University of Pennsylvania sent Bob Cowles, who was an ace VM system programmer at Cornell, what seemed to be a routine email. The next day when Bob logged onto Cornell's mainframe, everything seemed to be normal until the screen froze and a small PacMan icon begin traveling back and forth across his terminal screen eating the characters displayed. (PacMan was a popular computer game in the early 1980s.) Bob's attempts to enter commands into the system were ignored. After his screen was completely devoured, the bold letters "LOOF," which spelled "fool" backwards, began flashing in his face.

The innocuous email he had received the day before had contained unprintable system commands between the characters of the message. Since those commands were not recognized as characters, they did not display on his terminal screen. They did, however, allow the Penn programmers to gain control of Bob's VM session. They then created a virtual session that initially mirrored a real session, but later switched to the PacMan routine. The joke identified an early crack in our cyber infrastructure.

The Apple Macintosh and the Apple Consortium

In 1983 Stacy Bressler, Apple's regional sales representative, and Dan'l Lewin, Director of Apple's Macintosh Division, brought a prototype Apple Macintosh to Cornell. The machine resembled the computer that would be introduced to the world in January of 1984, but with a lot of parts and wires hanging outside the box. Lewin didn't want to let it out of his sight and carried it with him in a large box. He even went so far as to purchase an extra seat on the airplane for Apple's new entry into the consumer marketplace.

DCS had arranged for Bressler and Lewin to demonstrate the machine to a select group of faculty under a non-disclosure agreement. Dan'l was horrified when he entered the secluded classroom and found almost

two-dozen faculty waiting to see the rumored computer, and stood at the door checking names against the non-disclosure list. Apple took product security very seriously.

The demonstration was successful and the faculty were impressed, particularly with the graphical user interface and mouse. Apple wanted Cornell to commit to being an early adopter of the new computer, in large numbers, in exchange for a significant price break. For the faculty, however, the lack of a programming language more structured than BASIC was a deal breaker. Steve Jobs felt that BASIC was good enough, but finally relented and added Pascal as a programming language when Apple received similar pushback from other universities.

The Macintosh was introduced to the world in January of 1984. It had 128 K of RAM, a 512 x 342 pixel monochrome display, and a single 400 KB 3.5-inch disk drive in a single self-contained plastic cube. The keyboard and mouse were separate. It utilized a Motorola 68000 CPU chip, which was arguably slightly more advanced than the Intel 8086 (actually the slightly variant 8088) CPU used in the IBM-PC. The "Mac" was built in California in one of the industry's first automated factories. They were very proud of their factory and featured it in our visits to corporate headquarters.

The Macintosh "Bundle" made available to students in the spring of 1984 included a Macintosh, as well as writing and drawing software for the then-unheard of price of $1,000. The recommended retail prices were $2,495 for the Mac, $495 for the printer, and $195 for MacWrite/MacPaint.

The Student MacIntosh Bundle
Photograph by Doug Gale

As part of the early adopter arrangement, Cornell became a member of the Apple University Consortium. Originally envisioned as being a half-dozen or so universities, it quickly grew to two dozen at the time of the Macintosh rollout in January of 1984. William Arms' contribution to this incremental book captures some of the excitement at the student rollout.

Apple Consortium, January 1984

Stanford University	Boston College	University of Rochester
Harvard University	Princeton University	University of Pennsylvania
Yale University	Brown University	Northwestern University
Carnegie Mellon University	University of Chicago	University of Notre Dame
University of Michigan	University of Texas	Rice University
Cornell University	University of Washington	City University of New York
Dartmouth College	Reed College	Drexel University
Brigham Young University	University of Utah	Columbia University

The Consortium had regular meetings that were very upbeat, and reflected a mood that we were part of a fundamental change in the way computers were interacting with people. The famous 1984 Superbowl ad captured that mood perfectly. (https://www.youtube.com/watch?v=VtvjbmoDx-I)

I remember two consortium meetings in particular. At one, in Pittsburgh as I recall, I happened to sit at the same dinner table as John Scully and his teenage son, and could not help observing how Scully was grooming his son on how to be a business leader. Coming from a family where I was the first to go to college, I found that quite instructive.

The second was in San Francisco. The reception after the day's events was held in a converted warehouse in which multiple floors, each with multiple rock bands and light shows, opened to a central atrium; the effect was psychedelic. As a long-time fan of classical music, I confess that the atmosphere pulsed with energy.

While the utility of the Mac's graphical user interface seems obvious in retrospect, it was not universally accepted at the time of its introduction. I recall having a lengthy and somewhat acrimonious debate with a colleague at Cornell who felt the interface was a passing fad. We decided to settle the argument by each of us purchasing Apple stock; myself at the going price and he short selling. (I must also disclose that he totally disagrees with my recollection of this event.) Fortunately for him, the price of the stock didn't change much in the months following the introduction. Unfortunately for me, I sold mine months later before it took off.

Institutional Leaders

Who were the institutional leaders during this period? Obviously, the Apple Consortium members were considered leaders by at least one vendor. My personal list, however, would include: Carnegie Mellon University for their work with Project Andrew; Stanford and the University of California-Berkeley for their work in computer networking; Dartmouth for their pioneering work teaching programming and developing BASIC; the University of Michigan as an early leader in creating a state network; CUNY and Yale for their work in developing BITNET; MIT for their work with Project Athena; and of course Cornell.

Microcomputers and Elementary and Secondary Education

Introducing computing into elementary and secondary schools was a slower process, in part because they lacked centralized, information technology units organized to train and support faculty and student end users. In 1982 I was asked, on a consulting basis, to evaluate a draft proposal by the Ithaca Public Schools to their board requesting funding for the acquisition of a large number of Apple II computers. The school system was very unhappy with my report, which faulted the proposal for requesting too much money for hardware and too little for teacher training and support. My report was not included in the presentation that went to their board.

The Downside of Distributed Computing

One unintended consequence of the widespread deployment of microcomputers was to destroy one of the most useful features of timesharing on a mainframe—the ability to exchange information between individual users. We had created islands of isolated computers and users! The only way to move information between microcomputers—assuming they were running the same operating system -- was the physical movement of diskettes, jokingly referred to as "sneakernet."

Sometime in 1980, Fred Hiltz in the Cornell Veterinary School began working on a protocol to allow computers, including microcomputers, to communicate with each other over an asynchronous serial connection. The protocol had the catchy name of "FITS" for FIle Transfer System. The problem was widely recognized nationally, and in February of 1981 EDUNET, an activity of EDUCOM, set up a Task Force to address and consider solutions to the problem. Initially the task force was asked to look at two draft protocols, FITS and a similar protocol being developed at Wisconsin. In the meantime, Alison Brown in DCS had begun working with Fred on actual implementations of the FITS protocol, which by this time was quite well-defined.

I remember traveling to North Carolina to discuss their work with Lou Parker, Jr., the Director of the North Carolina Educational Computing Services, who was also on the task force, and discussing a possible collaboration between Cornell and a similar effort in North Carolina. One morning shortly after returning to Cornell I entered Alison's office and asked her how work was coming on the FITS implementations that she had been working on for months. She answered, "I'm not working on that anymore; I'm doing a KERMIT implementation."

She had learned of a similar file transfer effort at Columbia University, which had been named after Kermit the frog from the Muppet television program; feeling that they were further along in their work than she was, Fred decided to join the Columbia effort. (http://en.wikipedia.org/wiki/Kermit_(protocol)) That egoless spirit of working towards a solution characterized the academic computing community in higher education. For many years Kermit was the de facto standard for computer-to-computer file transfer.

The Emergence of Networking as Strategic

Even with the immediate success of Kermit, it was obvious that the underlying problem of allowing computers and peripherals to talk to each other—irrespective of different hardware and software—remained. The network had become strategic.

At the time, networking at Cornell was largely the responsibility of the Systems and Operations group. Because they were doing an exemplary job, DCS decided to focus on the software needed to get devices on the network. In late 1982 or early 1983 DCS hosted a spaghetti dinner at Alison Brown and Ken Wilson's house. Our pitch to Ken King was that we needed a small VAX running Berkeley Unix both to better support the increasing number of small system users moving to Unix, and to begin exploring software networking protocols, particularly TCP/IP. King's quick agreement makes me suspect that it was the Bison Bridle Principle ("you can lead a bison anywhere you want to as long as it's where he wants to go") and not our spaghetti that led to a quick decision.

Advanced Scientific Computing

The first half of the decade of the '80s also marked a growing national concern that the United States was losing its competitive advantage in high-performance, scientific computing. The 1982 "Report of the Panel on Large Scale Computing in Science and Engineering," also known as the "Lax Report" (http://www.pnl.gov/scales/docs/lax_report1982.pdf), concluded that without new resources "the primacy of U.S. science, engineering, and technology could be threatened relative to that of other countries…" Cornell's Ken Wilson was a member of that panel.

Under the leadership of Wilson and Alex Grimison, Cornell had already begun addressing some of the needs of large-scale computational users by attaching a Floating Point System (FPS) Model 164 array processor to the university's IBM mainframe. Wilson envisioned a field house filled with thousands of array processors. The software challenges associated with mating the two dissimilar architectures and operating systems were formidable, and Wilson became a legend within the Cornell Computing Center's systems group for his ability to overcome the challenges. This effort ultimately led to the Cornell Theory Center, which will be discussed in the next section.

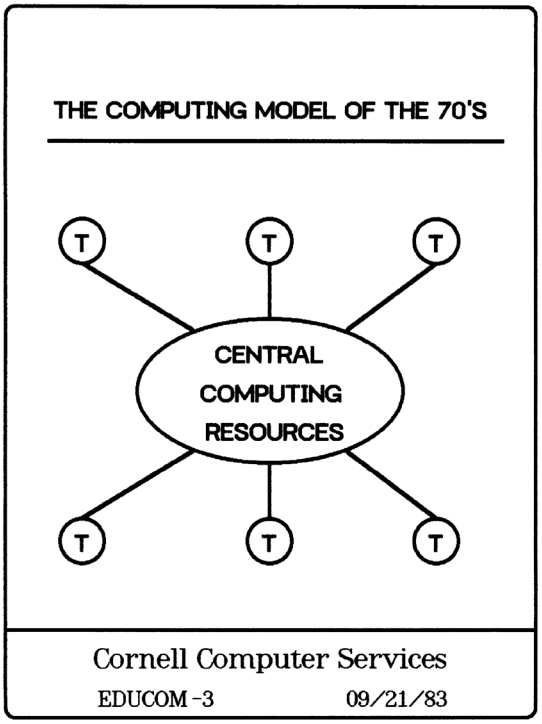
Cornell's Strategy for the Microcomputing Revolution

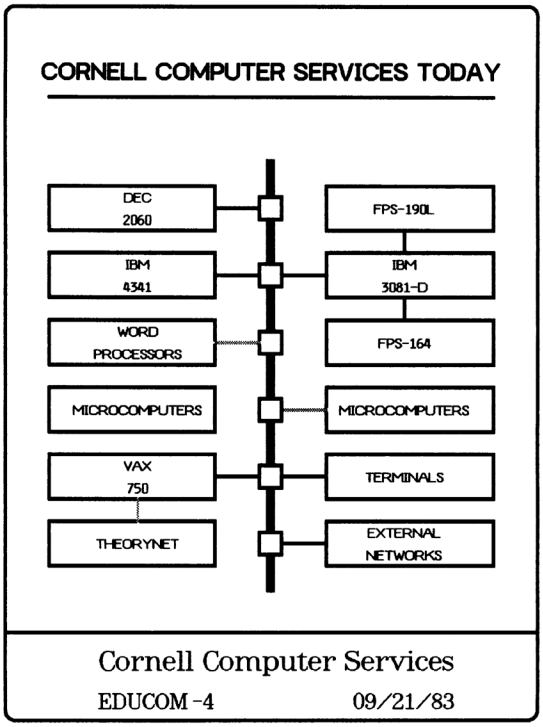
In May of 1983 I was invited to join Edward Friedman, Stevens Institute of Technology, and James Penrod, Pepperdine University, in participating in a plenary session on Personal Computing Implementation at the September EDUCOM meeting. That presentation encapsulates the changes that had occurred at Cornell and a few lead institutions and would shortly occur throughout higher education. (One slightly discouraging aspect

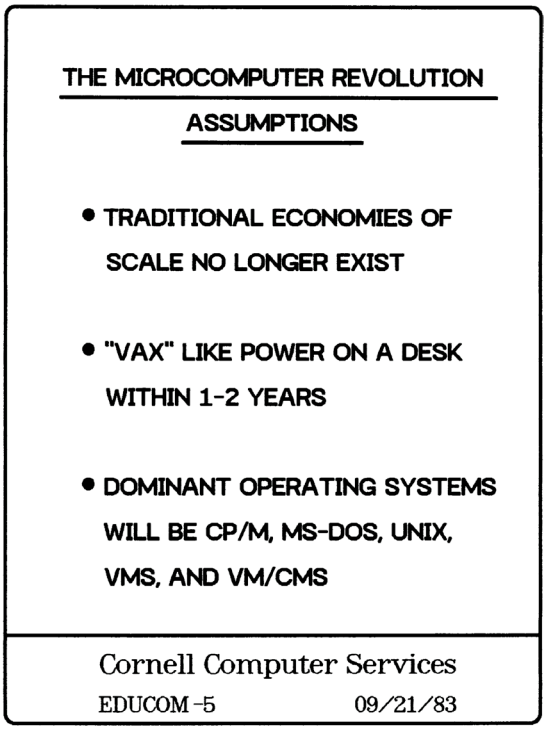
of the presentation was that I got more questions afterwards about how I had prepared my foils (on a Lisa), than about the content of my presentation.) The presentation is illustrative in what we got right and what we got wrong.

Foil "EDUCOM-3" showed the computing model of the 1970s, which was based on sharing centralized, computing resources. Terminals were connected to mainframes with proprietary networks. SNA for IBM, DECnet for Digital Equipment, and XNS for Xerox were typical. There was generally little interoperability between them.

By 1983, as shown on "EDUCOM-4", Cornell had begun making a transition to a single backbone supporting multiple systems. Much of the work developing the interface code for these devices was done at universities in groups such as DCS.



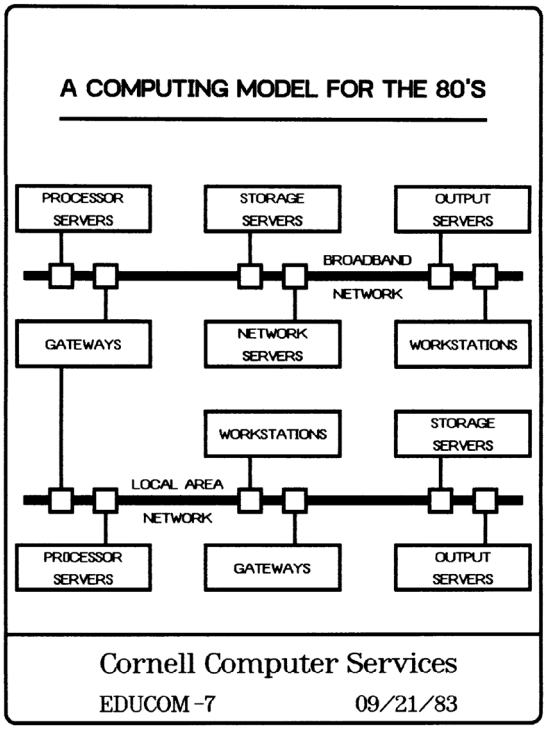


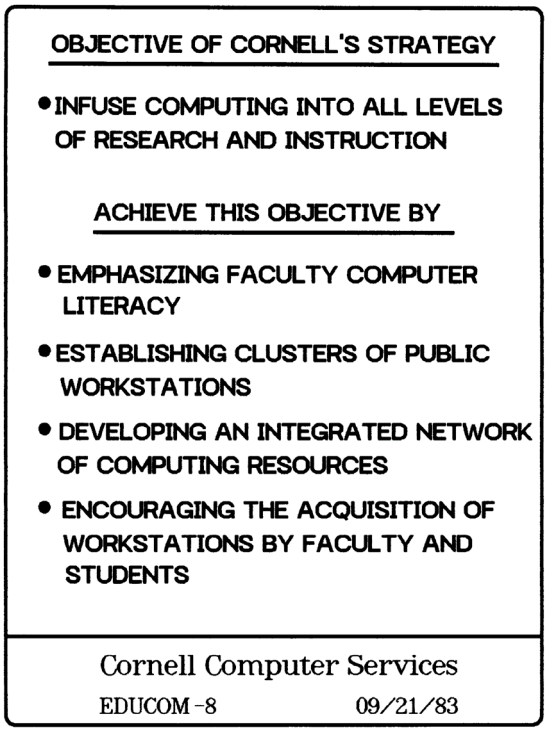


EDUCOM-5, The Microcomputer Revolution Assumptions, was basically an extrapolation of Moore's Law. (The number of transistors per square inch on integrated circuits will double every year.) One consequence of this was that traditional economies of scale no longer existed. Specifically, the cheapest computing was no longer necessarily the sharing of a big machine. Because of Moore's Law and the economies of mass production, the cheapest computing might be multiple smaller machines. Efficiency and keeping a computer running 100% of the time were no longer as important as they once were.

The implications for Cornell, EDUCOM-6, were that microcomputers would be the dominant source of cycles for most users within 2 years and, surprisingly, the total cost of computing to the institution would increase because of a growing user population and more complex systems. Cornell did not regard microcomputers as a silver bullet to decrease institutional computing costs. And, of course, our support mechanisms would have to change radically.

The fifth foil, EDUCOM-7, shows we also clearly understood, earlier than most, that a campus broadband network was going to be the core of a computing model for the 1980s. Our objective and the strategy to achieve those objectives (EDUCOM-8) were fairly common among institutional computing leaders.



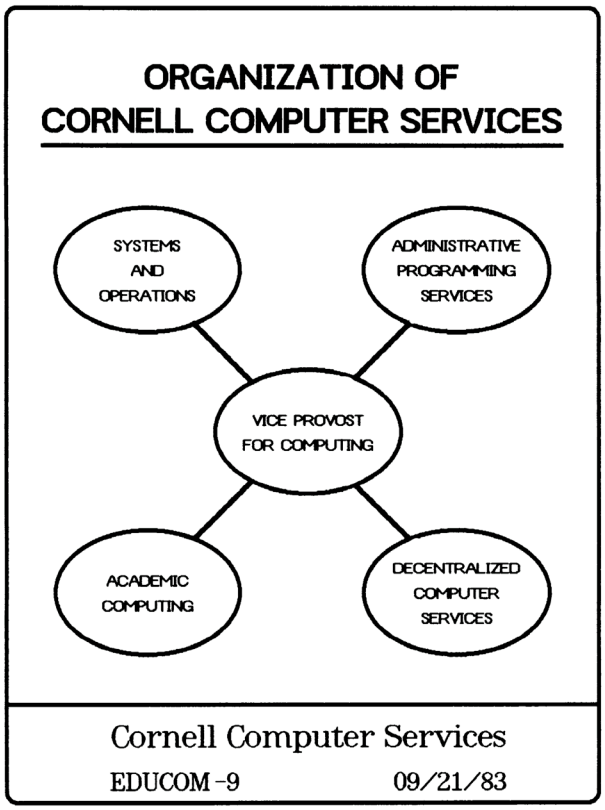


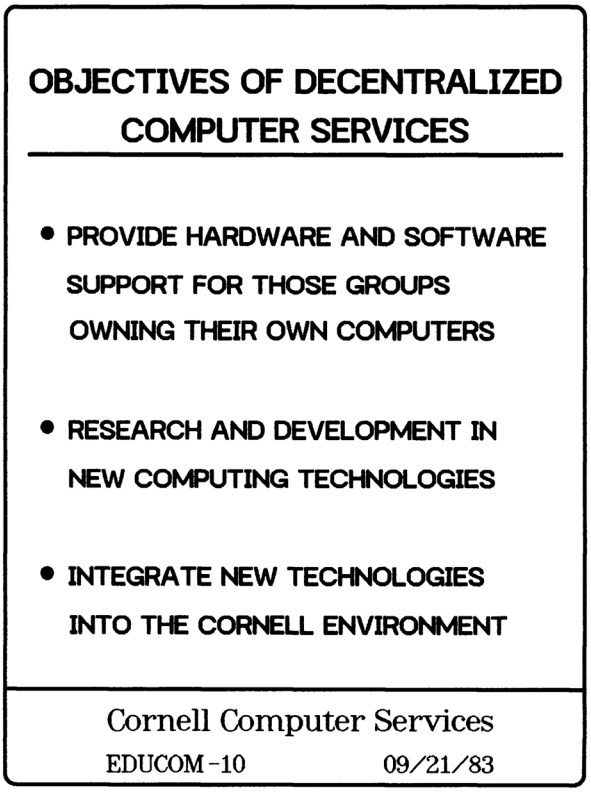
Cornell's Strategy Redux: What We Got Right, What We Got Wrong, and What We Simply Didn't Understand

The technical stuff we got right. Microcomputers did quickly become the dominant source of computer cycles. And a network linking computing resources, from microcomputers to supercomputers, and from printers to scanners, both on campus and nationally, would become the integrating factor.

What we didn't get right was the most efficient organizational structure to support the new realities. Nor did we understand the real role and mission of DCS. We had taken the role of DCS to be "consultation and support for those units owning or planning acquisition of computers." Based on that assumption we created a four-branch organization (EDUCOM-9) consisting of Systems and Operations, Administrative Computing, Academic Computing, and Decentralized Computer Services. What we didn't realize was that the objectives

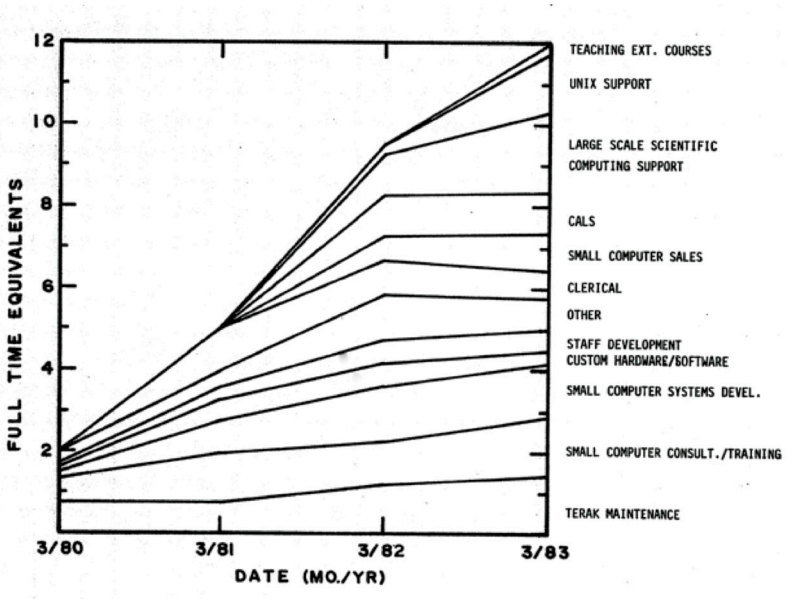
of DCS, shown in EDUCOM-10, were largely crosscutting and applicable to the other three organizational branches as well.





In 1980 I was told by Ken King, I believe, that "DACS has all of the gold and none of the bricks; it should have so much fun that everyone in the organization should want to work there." In short, DACS and later DCS, was a change agent whose purpose was to change the centralized mindset of the organization.

The rotating door within academic computing reflected this interpretation. In May of 1981 Alex Grimison stepped down from the position of Interim Director of Academic Computing to join DCS and to take charge of a scientific computing group. I assumed his role until we were able to recruit a new Director of Academic Computing, Gordon Gallaway. In short, we were having fun in DCS and were functioning more as an internal "skunk works" than an operational unit, such as system and operations. Our mistake was to envision DCS as a permanent organizational structure rather than a temporary change agent.



By the spring of 1983, DCS had grown to 12 FTEs. As shown below, the new functions within DCS were large-scale scientific computing, Unix support (including networking interfaces), and managing small computer sales through external vendors. While the first two would go on to become part of Cornell's "Theory Center" supercomputing initiative, computer sales and small computer support gradually devolved back into more traditional support structures. The centralized culture of the Cornell Computer Center had been changed. A similar organizational

phenomenon occurred in the latter half of the 1990s, when internal "networking skunk works" sprang into existence to drive the networking revolution.

Institutional Collaboration

The early 1980s also marked the growing importance of national organizations facilitating the exchange of information between university and college computing center leaders. It was also marked by a great deal of sharing of experience and knowledge.

In the summer of 1980 I attended a regional meeting, the New England Computing Conference. It was at that meeting that an old hand from either Vermont or New Hampshire passed on to us newbie's the three rules for a computing center director: 1) today is as good a day to die as any other; 2) a horrible end is preferable to horrors without end; and 3) you never control anything. At the time I was amused, but didn't fully appreciate the truth in his words.

The two largest higher-education computing organizations at the time were CAUSE and EDUCOM, but the meetings seldom exceeded several hundred attendees and everyone knew everyone else. A history of both organizations can be found at http://www.educause.edu/about/mission-and-organization/roots-educause. While neither organization was wholly academic nor administrative, in the early years CAUSE seemed to have an administrative computing and "hands-on" feel, while EDUCOM had more of an academic and management/policy feel. In any case I became a regular EDUCOM attendee, although later in the decade I attended both and was on the CAUSE Board when the two organizations merged in 1998. In general there was little overlap in attendees between the two organizations.

The most enjoyable and arguably most useful of the national meetings was the Annual Seminars on Academic Computing held in Snowmass, Colorado. When I first attended in 1982 there were around a hundred attendees. We met in a single small resort hotel, the Mountain Chalet. Attendees frequently brought their families, who would enjoy the Colorado mountains while we attended sessions. Toward the latter part of the decade I would sometimes arrive a few days early and backpack from the Maroon Bells over Buckskin Pass and into Snowmass.

A fourth meeting that I routinely attended was the ACM (Association of Computing Machinery) SIGUCCS (Special Interest Group in University and College Computing Centers) meeting that was held each spring in St. Louis. The meeting was small with a very practical "hands on" feel.

It is hard to describe the excitement and comradery of those early meetings, as compared to the same meetings one or two decades later. In the 1980s I would typically know half or more of the attendees. Computing was just emerging as a discipline, and there was a close bond of togetherness.

Chapter 3: The Late 80s and the Networking Revolution *(not written)*

Chapter 4: The 1990s: The Changing Role of Academic Computing and Networking *(not written)*

Epilogue[1]

by Kenneth M. King

Douglas Shannon Gale died on October 26, 2015, at the age of 73, with his memoir half finished. He intended to write a chapter covering the networking and distributed-computing revolutions that occurred during the late 1980s, and a chapter covering the impact of networking and personal computing on academic computing at universities from the 1990s up to the present. Doug was in the middle of the events that shaped the transformation of computing at universities during and after the 1980s, and he contributed to the evolution of computing and networking in a number of important ways. It is my hope that those who worked with Doug, as I did, will contribute to an effort to extend his memoir by describing his activities and contributions during the years that he would have covered had he been able to complete his memoir. Doug's memoir is part of an incremental book started with the hope that people involved in the early years of academic computing at universities would add their own memoirs, so that that important history will not be lost. Contributions to an effort to complete Doug's memoir will become part of an incremental Epilogue appended to his Memoir.

I met Doug in 1980 at Cornell when he was Director of Decentralized Computing Support. At Cornell in the early '80s, as at most universities, the primary source of computing cycles supporting research and instruction were centrally operated and controlled mainframes. Doug led the effort that resulted in faculty and students largely controlling their own computing devices. This transformation led to an enormous increase in faculty and student productivity and an exponential growth in new computing applications. It also dramatically changed the role of people supporting academic computing at universities. His activities at Cornell are covered in Chapter 2 of his memoir and in a video interview of Doug done a few weeks before he died. This video can be seen at: https://ecommons.cornell.edu/handle/1813/41195

When the formation of NSFNET as a three-tier network was announced in the fall of 1985, Doug was Director of Computing at the University of Nebraska, Lincoln. NSFNET consisted of a backbone network that connected the National Supercomputing Centers; Regional Networks that connected universities to the backbone; and local area networks that connected faculty at universities to a Regional Network. Doug immediately began setting up a Regional Network called MIDnet. MIDnet connected universities in Iowa, Nebraska, Missouri, Oklahoma, and Kansas to NSFNET. In September 1986, MIDnet became the first Regional Network to become fully operational. Also in 1986, EDUCOM formed a networking and telecommunications taskforce (NTTF) to advance the connection to NSFNET of every college and university in the country, and later to the world. Doug was very active in this organization and in creating an organization of Regional Networks called FARNET. As NSFNET evolved into the Internet, Doug made major contributions to improving university connectivity at a number of institutions and also served a stint as a program officer at NSF, which was funding network expansion. Overall, Doug was a major player in advancing networking for more than three decades.

Throughout his career Doug meticulously preserved records of meetings and papers devoted to computing and networking. He also had a collection of computers dating back to the earliest days of microcomputers. In 2010, Doug founded the Internet Legacy Institute to preserve and archive information and original source materials about the creation and evolution of the Internet. This organization continues and has made major contributions to the preservation of networking history. Doug travelled extensively with a camcorder to interview people involved with networking during its formative period. This historical record is in the process of being donated to organizations dedicated to preserving historical records. His records will be of enormous value to people seeking to understand events that changed the world.

---

1 Released on April 27, 2016