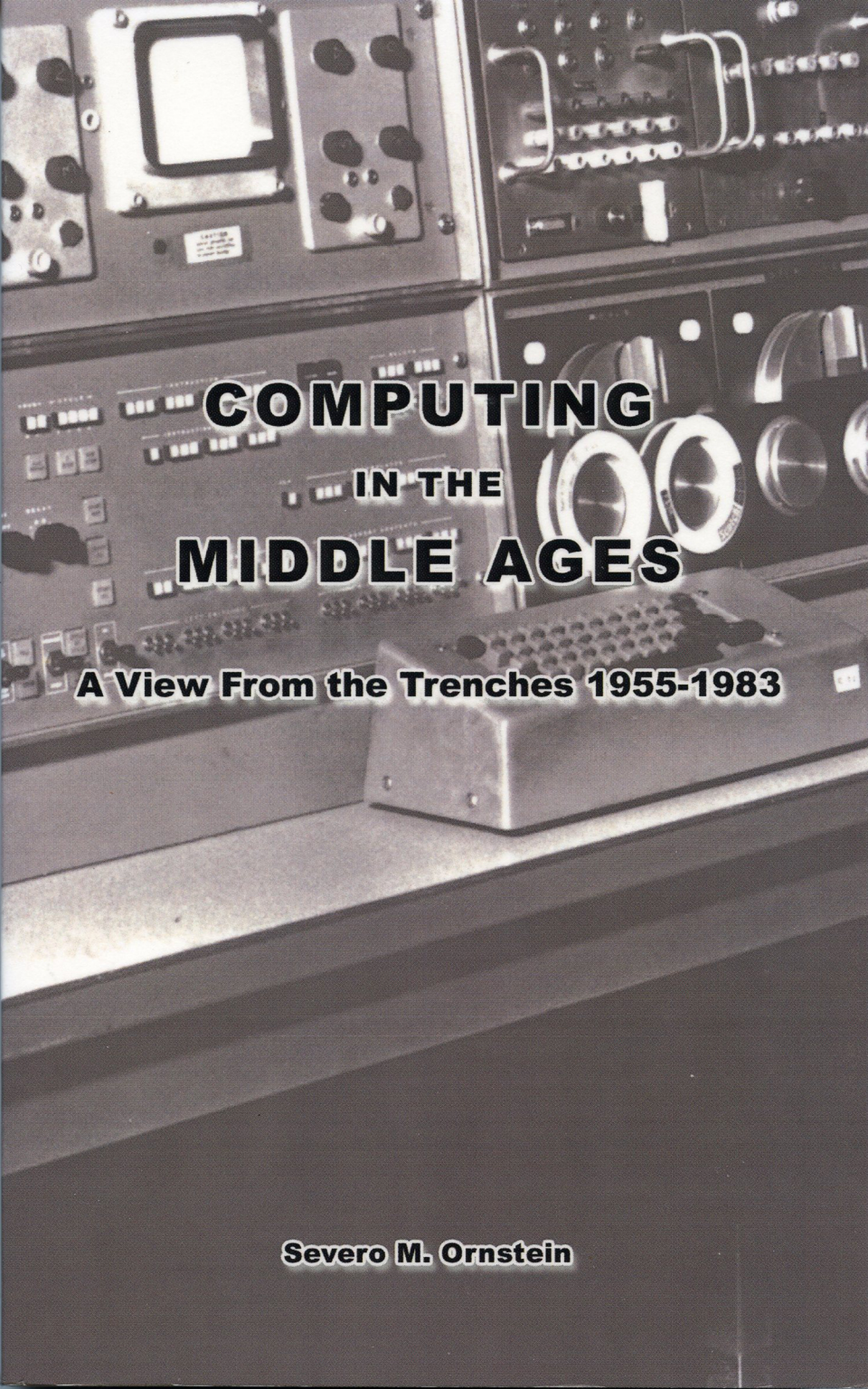


COMPUTING IN THE MIDDLE AGES

A View From the Trenches 1955-1983

Severo M. Ornstein

Computing in the Middle Ages

A View from the Trenches
1955 - 1983

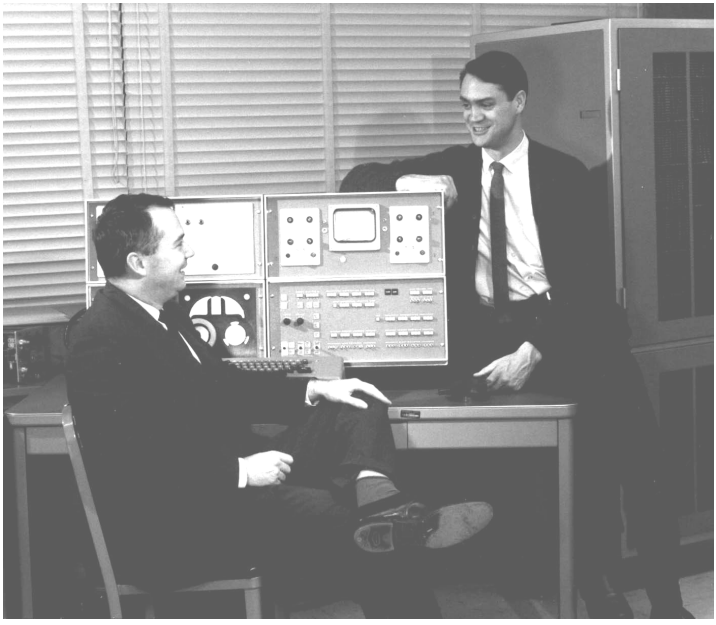
By

Severo M. Ornstein

ISBN: 1-4033-1516-7 (Ebook)
ISBN: 1-4033-1517-5 (Softcover)

This book is printed on acid free paper.

1stBooks - rev. 10/15/02



For Charlie and Wes

Table of Contents

List of Illustrations

Preface

You turn the key—the engine comes to life. You press a button—up goes the window. Another and the antenna emerges and music washes over you. Still more buttons and your seat adjusts itself to you, you are warmed, lights come on, the windshield is scrubbed, the garage door gently rises, and finally you are on your way. Who could have foreseen such a world in the days of hand cranks, dust goggles, and isinglass curtains? And yet somehow we got from there to here. How did we do it?

In a similar vein one might ask how, given its monstrous, room-filling forbears, computers such as the modest one on which I'm presently typing came to be. Surely not all at once, not overnight. There was no road map. What paths led through the myriad by-ways that brought us to this point?

It's difficult to believe that over the course of a few decades anything like today's personal computer could have evolved from the giant machines[1] of the late 1940s and early '50s. Except in Mitchell Waldrop's excellent book (see Bibliography), the story of how this dramatic change came about has not been at all well told. Often one gets the impression that the personal computer revolution began in the mid 1970s. But that is only when it began to touch the consciousness of the general public and to blossom into an industry. By then, the image of the personal computer was already quite well formed and machines not fundamentally

[1] The terms "computer" and "machine" are used interchangeably herein.

different from today's were already in use in research laboratories. The real story began many years earlier with the scientists and engineers of a relatively unknown laboratory at MIT. They were the originators of much that has followed and were to spread productively throughout the evolving computer science community over ensuing decades.

Many recent so-called computer histories, catering heavily to the public lust for a peep at the rich and famous, have explored, ad nauseam, the eccentricities of Bill Gates and Steve Jobs and their brethren—to the point that they have become almost mythic figures. Their stories and their garages have become legendary. But these are Johnny-come-latelies who have achieved notoriety for the most part not for innovative science or even for engineering, but rather thanks to their extraordinary ability to exploit ideas pioneered by others, to turn them into financial empires. But what about the pioneers themselves, the ones who did the scientific and engineering groundwork on which these empires have been built? Where and who are they? Too often they appear only in fading photographs, wearing outmoded suits, usually standing in front of giant machinery that bears no apparent resemblance to the computers of today.

A few years ago we tuned in hopefully to a four-hour PBS special that claimed to provide some of the history behind the computer revolution. But the focus turned out to be once again on exploiters rather than explorers, and the history I'd known and experienced was largely ignored or glossed over. At about the same time, the untimely death of a dear friend and former colleague, Charlie Molnar, emphasized how few of us remain who are in a position to help straighten out the record. Charlie's death affected me deeply and reminded me what extraordinary human beings

I had been privileged to know and work among. And I thought how little recognition is given to such key people, thanks to their personal reserve and their preoccupation with their work rather than the limelight.

I initially set out to write this book because other histories seemed to ignore or even contradict my thirty-odd years of experience in the computer field. Developments that I was sure had marked important milestones were given short shrift or were overlooked altogether, and forces that seemed to me to be crucial in setting directions were sometimes not even mentioned. Debates that had once almost led to bloodshed, seemed to have been utterly forgotten once the passage of time had resolved them or rendered them irrelevant. And finally, for many younger people, history appeared to have commenced only yesterday—whereas I was sure I remembered the day before, and the day before that. Indignantly I set out to put the record straight.

Then I remembered the fire-lookout.

In 1950, one of my college roommates spent the summer in a fire-lookout atop a peak in the Cascade range of Washington. From the lookout a vast mountainous region of the northwestern U.S. spread out before one, and over the course of the summer my friend drew a map of what he saw from his eyrie. Inspecting his map in the fall, and comparing it to official maps of the same area, we were vastly amused to discover how distorted his view of matters had been. The geography near the lookout was spread out and enlarged, whereas the more distant features were all crowded together around the fringes of his map. Like the New Yorker magazine's cover burlesquing a Manhattanite's truncated view of the U.S., my friend's view of the country surrounding his lookout was compressed into insignificance.

Contemplating such myopia, I realized that my version of computer history would likely turn out to be just as distorted as all the others—just differently so. Moreover, as I considered pawing through dusty archives, I began to lose heart. It looked like a lot of work. Did I really care that much? Meanwhile I'd made a serious mistake. I'd mentioned my idea to a number of friends who began egging me on. What had I let myself in for?

The previous year I had been asked by a former colleague to give a lecture to his introductory computer class. Although his class was becoming increasingly computer literate, they had little or no comprehension of where the technology they were toying with had come from. Feeling that some background would not only be salutary but might even interest them, we agreed that I should try to summarize my own experiences of thirty years in the field and discuss the evolution of computer technology from that vantage point. It seemed a decidedly self-centered approach, but it had the sizeable advantage that little or no preparation was required. In the event, the class showed surprising interest in the history, and a number of the students later sent messages urging me to write it all down. I was pleased that I'd piqued their interest, but writing it down seemed absurd. After all, it was just my own narrow, personal view of what had happened.

Nonetheless, the seed had been planted and as time passed I began to understand that one of the things that was bothering me was the absence of the human theme in the history of what might be called the "middle-ages" of computer development—the period between the days when only a handful of "giant brains" existed and the time when computers suddenly began to spread like wildfire throughout the society—roughly the early 1950s to the mid

1980s. De-emphasis of the individual is an important component of the scientific/technological outlook, and indeed objectivity is an essential ingredient of all serious science. But history is not the same thing as science. In addition to the facts of what happened, there are the often colorful human stories of how it all came to pass. Ultimately history is a tale of human endeavor, and as such, it inevitably captures our interest in ways that purely technical descriptions cannot.

Another thing that bothered me was the absence, in most of the histories I'd seen, of a bridge that gave adequate feel for how and when so many of the things we now take for granted had come about. These things were not specified in a book handed down out of the clouds. They arose gradually, as people's understanding grew. As part of the generation (now slowly withering) that did much of the work during the middle-ages, I decided that perhaps after all I should follow the urging of my friend's students and attempt to record matters as I had experienced them. Those years formed an important and exciting era in the history of computing, and although I myself was never a major figure in the field, I had the good fortune to work alongside others who were, on a number of projects that have turned out to be central to the computer revolution.

Despite a swarm of hand-waving vision-painters and post-hoc vision-claimers, much of the progress that has occurred in the computer field has arisen simply from people's curiosity about the next step—the desire to push the frontiers—rather than from clear visions of the future. There are important exceptions, of course, but they are indeed exceptions rather than the rule. Because one thing leads to another, evolution and growth have been nearly exponential, and this has produced the impression that virtually

everything happened very recently. But many of the crucial steps and decisions that enabled yesterday's explosion actually took place many years earlier.

Of course, the invention of the transistor at Bell Laboratories in the 1940s and the later development of integrated circuits were of such profound importance that without them, my story and most of the world's computers would simply not exist. These inventions permitted computers to become smaller, faster, cheaper, more powerful, and more reliable. But arriving at today's personal computer was not only a matter of making things smaller, faster, and cheaper. The entire image of computers, what they were and what they were good for, needed to be transformed. The very name 'computer' itself reflects the device's earliest use, and it took many years before the image of numerical computation as a primary focus began to fade. Many of the developments that took place during the 1960s and early '70s represented efforts to redefine the way users would perceive and interact with computers. In the early '60s I was deep in the trenches, where the battle to define and build the earliest personal computer, the LINC, was taking place. The course pursued by the team with which I was working was far from the mainstream of that period, and our trajectory carried us straight through the middle of many vigorous controversies.

My life, I regret to report, reveals a stunning lack of single-mindedness. During college, dilettantism seemed likely to qualify me at best as a ski bum; the notion that I might one day have a profession, let alone a career, seemed laughable. Nonetheless, despite the odds, life developed into a fascinating and rewarding ride. After the lecture to my

friend's class, one of his students asked him, "Did that guy ever make a bad career move?" Pondering this question has left me somehow depressed. "Career moves" are part of the paraphernalia of today's heavily-packaged lifestyle in which twenty-year-olds are already preparing for retirement. I realized I'd been fortunate to work in an earlier era when one could more easily allow the search for interest and/or fun to guide one's choices. My own acquaintance with computers commenced not with design but with programming, and at the outset I marveled at the ingenuity of the machines which must, I felt, have been devised by wizards. Nonetheless, over time, as mystique metamorphosed into understanding, I gradually shifted into hardware design and in due course not only assisted in the birth of a number of different computers, but even taught courses in machine design at my old alma mater and elsewhere.

I must have been blessed with a good nose for sensing where important work was taking place and finding a way to join the team. I never felt that I was a real innovator, although it's not always easy to distinguish major inventions from the myriad smaller ones that help make the major ones work. In any case, I found that I was able to make contributions to these groups as a high-energy enthusiast, helping to turn ideas into reality. I was delighted to be able to participate and always felt amply rewarded for those contributions that I was able to make.

My professional life was filled with numerous and often colorful characters, many of whom appear in the following pages. Some of these have been my friends over the better part of a lifetime. My goal has been merely to show them as they appeared to me, but not to caricature them, and I hope

that both they, and others who know or knew them, will find them honestly, if sometimes bemusedly, portrayed.

In parallel with my professional life, and sometimes deeply entwined with it, runs the narrative of my personal life, which naturally affected the course of my career. For the most part, however, it is not germane to the tale I wish to tell, so except for brief mention in a few places, I have chosen to eschew that part of the story. Let me just state briefly that I have been married three times and have four children and (at the present writing) four grandchildren. My wife, Laura Gould, with whom I've shared the last twenty-eight years, was, like me, a sort of maverick computer scientist and teacher. And like me, in the almost twenty years since we gave up working in the computer field, she has completely changed her spots, becoming a writer immersed in the history of science (but not computer science). My own interests shifted away from engineering and toward the humanities, in particular to music which I had once briefly considered as a possible career.

What is contained herein, then, is a tale laced with anecdotes and commentary, half way between a personal memoir and a more general history. Rather than attempting to give an objective and comprehensive overview—a God's-eye view—I have tried instead to describe what it was like to be in the belly of the beast, to participate with colleagues in the process of exploration, and to give some idea of where concepts we now take for granted came from. My own biases and opinions are writ large throughout and because my own experience failed to touch many of the important branches of computer development, they will be found missing here. This is thus the antithesis of a comprehensive story. Nonetheless, I hope that from this tale will emerge some idea of the enormous differences between the computer world of

today and that of the 1950s to the 1980s: and that the reader will be able to enjoy, and participate vicariously in, some pieces of the journey that connected the two. Because computers have so thoroughly invaded the lives of so many of us, I have particularly wanted this story to be accessible to everyone, not just computer buffs. I have therefore eschewed technical descriptions (or committed what some will no doubt consider egregious simplifications) and tried to paint images that could have been comprehended by anyone who might have happened to be at our elbows during that remarkably critical epoch of computer evolution.

Acknowledgments

Memory is a fragile vessel to hold even the most significant events of a lifetime, and in my case considerable ullage has occurred. I make no apology for this; it is the natural distillation that time imposes. But as a consequence, I have found that I needed reconstructive help in places, and I've not hesitated to seek it among friends and colleagues. However, I have used such material only where it has lent support to my own sometimes hazy memories, and it is therefore I who must take sole responsibility for the facts, the semi-facts, the non-facts, and the outright opinions recorded here.

To list the many friends who have contributed reminiscences and helped to adjust my memory would not only be tedious for me and for you, the reader, but, given the narrow and personal perspective of this work, could even prove embarrassing to some who will no doubt prefer to remain in the shadows. Nonetheless, I feel compelled to single out two people, Wes Clark and Les Earnest, who read the manuscript and patiently identified my most serious misunderstandings and memory lapses. And my wife Laura Gould provided her usual thorough editing assistance. To the rest I hereby proffer my blanket thanks for your encouragement and assistance: without you, my record of history would have been far less coherent and, in places, less correct.

Introduction

Life is not an orderly progression, self-contained like a musical scale or a quadratic equation. For the autobiographer to force his life and his memories of it into a strictly chronological straight line is to distort its shape and to fake and falsify his memories. If one is to try to record one's life truthfully, one must aim at getting into the record of it something of the disorderly continuity which makes it so absurd, unpredictable, bearable.

Leonard Woolf, *The Journey Not The Arrival Matters*

Spread out across the surface of my desk are several manuscript versions of the score for a piano quintet that my father wrote not long before I was born in 1930. The copies differ slightly in a few places, and my task for this coming fall is to compare these different versions and, by careful selection with the assistance of other musicians, enter into my computer, and thence publish via my laser-printer, what will no doubt become the Urtext version of this composition. I've been doing this sort of thing for the past decade, in the course of which I've transcribed well over two thousand pages of my father's music. In the early part of this century he was one of America's best known pianists and a notorious composer of radical music. He has gained another distinction in recent years; at the age of 109, he is not only still alive and well, but lucid and able to convey fascinating images of life in Paris nearly 100 years ago.

Every day as I commence to work, staring out at me from the surface of my computer's screen are the side-by-side figures of me and my oldest chum from college days. It's an image, pieced together with the magic of computer graphics, from a couple of individual photos we snapped of one another on a recent climbing trip in the Sierra Nevada mountains. Søren Kierkegaard, the Danish philosopher, wrote a book entitled *Purity of Heart is to Will One Thing*. I'm sure he was right and I wish I possessed purity of heart, but alas, it is not the way of most lives, including my own. Instead the life that is revealed behind the smiling faces on my screen is a messy, steel-wool-like affair, made up of a jumble of interlacing threads. There are listings and files for the thousands of pages of music scores on which I've been working; there is the material that comprises what I'm writing here; there is information about a series of chamber-music concerts that we hold in our home each year; there are correspondence and email files containing letters and messages to and from around the world on a host of different subjects; there are files concerning our wills and our accounts and those of our elders whose affairs we must now handle; there are files that describe and tell how to manage our property here in a relatively remote wilderness area; there are recipes, address lists, and so many other things that it's quite hopeless even to summarize them.

In fact, it's frightening to contemplate how dependent I have become on my computer and its link into the network that connects me to friends and information around the world. In honor of that fear I periodically go through a paroxysm, copying everything onto backup disks and toting them to the house of a friend that I hope will not be damaged by the earthquakes, fires, gremlins, etc. that could jeopardize my precious information here at home.

It may strike the reader as odd that, in listing the kinds of material contained in my computer files, there is no mention of material from the many years I spent actually engaged in the computer profession. There are indeed a number of papers in the literature with my name on them, but they were published before computers and word processing became commonplace. Most were written in the old-fashioned way, with pencil and paper, a pink-pearl eraser, scissors, staples, scotch tape, and an infinitely patient secretary who turned my hieroglyphics into draft after draft as the chaos gradually converged toward finished copy. Those papers, now yellowing somewhere in the bottom of a desk drawer, give evidence that at one time I played a part in the evolution of the personal computer, networks, laser printers, and the like. At the time, although we hoped and believed that the things we were designing and building would prove useful, I myself certainly had no idea that I would ever make the kind of routine and personal use of such facilities that I now enjoy.

Having lived through these developments, it all seems to have happened amazingly quickly—and indeed it has. But not quite as quickly as some people seem to think. So let me take you back a few dozen years to where, for me, it all began.

Chapter 1

In which I meet some early computers, write my first programs, learn a little history, and switch professions.

It was the fall of 1954 and as I strolled across the parking lot on my way to work, I noticed a climbing rope in the back of a large gray Hudson—the car that looked like an upside-down bathtub. I jotted down the license number and later that day, having obtained the owner's name, I went in search of him. Like me, Howard Briscoe had been hired as a geophysicist to work in the exploration department of Gulf Oil's Research and Development Company in Pittsburgh, Pennsylvania. Three years before I'd graduated from Harvard as a geologist and then spent a year in graduate school at Berkeley where I'd substituted music for sleep. After that I'd gone to work as a geophysical trainee with the Gulf Oil company, which meant working with seismic surveying crews first in New Mexico and later in Oklahoma, looking, of course, for oil. By the time I met Howie I was married, had a young daughter, and was already bored with the routine work of correlating seismic records. I'd decided that it should be possible for some sort of automatic machine to do most of what I was supposed to be doing and had begun thinking about just what such a machine would have to do, when the climbing rope appeared.

Howie was also a geologist by training, but unlike me he'd had experience with a computer—MIT's new

Whirlwind[2] computer. He'd been a member of the Whirlwind programming staff, had written Whirlwind's symbolic assembler, and later became a member of a group known as the GAG (Geophysical Analysis Group) in which he had written a program to process seismic data. Gulf, as one of the sponsors of the project, had hired Howie to help them move in similar directions. My thoughts about automating our work fit right in and we quickly became friends.

Howie had helped to teach a summer session course about Whirlwind at MIT and as he began to describe the machine to me I found myself fascinated. My only previous brush with a computer had been very indirect, when our high-school math teacher described to the class his visit to the ENIAC at the Moore School of the University of Pennsylvania. He had tried to convey to us the excitement he felt watching the flashing lights of the world's putative first electronic computer, but his enthusiasm fell on the largely deaf ears of a group of fifteen-year-old boys with other things on their minds. But now, stimulated by Howie's enthusiasm and excellent teaching, and with the definite purpose of relieving myself and others of tedious, routine work, I became a well-motivated student. Our regular work didn't demand anything like our full attention, and we spent the rest of the time discussing computers.

I had previously assumed that computers were only for the electronically sophisticated, which I certainly was not. The idea that one needn't fully understand its electronics

[2] Whirlwind, at MIT's Digital Computer Laboratory, was at that time the world's fastest digital computer.

but could treat a computer in purely logical terms, making it do one's bidding through a complex sequence of instructions, was a tremendous and empowering revelation. I felt I might be able to deal with that.

At that time there was serious debate whether the future belonged to analog or digital computing. Looking back with today's understanding and with today's technology, it's hard to believe that anyone could have thought that analog computing was a serious contender. But back then the emphasis was still on numerical computation and the great diversity of tasks for which computers would eventually be used could hardly be foreseen with the technology of those times. The idea that you could make machines that would run so fast that you could afford to break computation of continuous functions up into jillions of tiny individual steps was by no means universally accepted. The speed of operation had an enormous impact on the sorts of jobs for which one could reasonably consider using a machine, and speeds such as we expect today in every home computer were then all but inconceivable.

In due course Howie lent me a book about England's EDSAC computer (an indirect descendant of the ENIAC) and I spent some time studying the mysteries of its "bootstrap code" —the small initial program that runs when you push the Start button and brings into memory the much larger program you actually wish to run. The bootstrap code needed to be extremely compact: the memory that held it was small and prohibitively expensive, and it had to be laboriously reloaded, instruction by instruction, by hand, every time the machine was restarted. Restarting was pretty frequent in those days because the electronics were terribly fragile and machines broke down

with frustrating regularity. Great ingenuity had therefore been expended on constructing a bootstrap program consisting of the fewest possible instructions. The initial program modified itself as it ran by overlaying some of its instructions with others that it brought in. It thus modified its own behavior in an extraordinarily clever and confusing way—like a serpent devouring its own tail, I thought. I'd never encountered anything like this before and was genuinely excited. I was also thrilled to find that, with considerable effort, I was able to unravel it all.

I learned later that because computer memory was such a scarce and expensive commodity in those days, similar effort went into compacting almost all programs. Diabolically clever schemes were worked out for reducing the size of programs. Devising ways to compact really important, frequently-used programs was often a group undertaking with details being worked out at a blackboard, sometimes over a period of days or weeks. Howie told me that on one occasion the latest version of a Whirlwind program that had been heavily worked over was left overnight on the blackboard. When the programmers arrived next morning they were horrified to find that an over-zealous janitor had carefully erased it all. (I was later to experience a similar catastrophe myself.)

Such cleverness often meant that early programs were difficult for anyone but the author(s) to comprehend[3]. As memory began to be less prohibitively expensive, and as

[3] And at the close of the millennium, the entire world trembled in anticipation of potential disaster stemming from the trick of saving memory-space by ignoring the first two digits of the year in specifying dates.

programming became more widespread and the size of programs grew, the need for straightforwardness and clarity came to dominate the need for compaction. Eventually an entire sub-discipline, software engineering, came into being to explore and delineate methods for enforcing straightforward organization and for improving clarity in today's enormously complex programs and systems.

After I'd mastered the EDSAC's bootstrap program, Howie lent me a manual describing Whirlwind and how to program it. Under his tutelage I worked my way through the manual and wrote out the suggested exercises. Of course these programs were never going to be executed since Whirlwind was hundreds of miles away in Cambridge, Massachusetts, and remote computing still lay far in the future. Besides, Whirlwind had far more important things to do. Mine were just paper exercises designed to teach one how to write programs. After I became somewhat proficient, Howie made the startling announcement that most of the instructions I'd been using were actually unnecessary and were provided only to save memory and make programs run faster. He explained that there were, in fact, only a handful of truly basic instructions and that all the others could be emulated (i.e., the machine could be made to perform precisely the same function) by programs made up only of this basic set. I tested the validity of this claim by programming one or two of the more complicated instructions using the basic set, and once again I was stunned: the bloody thing seemed to consist almost entirely of sheer ingenuity, balanced atop the merest pinpoint of material reality. Where would it end? What did it mean? I was now not only fascinated, I was hooked.

The Gulf research lab had purchased a "small" computer (it probably cost upwards of $100,000 which was serious money in those days) that had a magnetic drum memory. That means just what it says: the main memory of the machine consisted of a rapidly rotating drum whose surface was coated with magnetic material (think mag. tape) on which the information (data, instructions, etc.) were written. Thus if you wanted to get at a location that had just passed under the read heads, you had to wait for a full revolution until it came by again—not exactly *random* access. In order to speed up the running of programs, the machine had a complex addressing structure that allowed the programmer to specify where the next sequential instruction was to be found. I wrote a tiny program or two for the machine, but I have to confess that the thing about it I remember best was the large, shallow glass-fronted bin which held a length of randomly curled up wide magnetic tape, used for secondary storage. This bin took the place of reels. Because the tape was allowed to fall into it at random (but always maintaining its width parallel to the bin's shallow front-to-back dimension), there was little inertia (no heavy tape reels) to deal with in quickly accelerating the section of tape under the read/write heads. I didn't understand the motivation at the time, but was impressed with the beauty of the way the tape curled up in its bin.

One night Howard and I went to a lecture at the University of Pittsburgh. It was given by a couple of people from MIT named Belmont Farley and Wesley Clark. The topic was the computer simulation of neuronal activity in the brain. Here were yet more revelations. The programs I'd been writing were the dimmest of student exercises consisting of boring things like adding tables of numbers together. Now I was to see how the simple capabilities of

this incredible machine could be turned to far more compelling tasks. I'd always wanted to know more about how the brain works and here were people using this lovely new instrument to chip away at understanding that very thing.

Their talk stimulated my imagination and I immediately began to ponder whether computers could really lead to an understanding of how the human brain works. Similar fantasies were soon to captivate not only the press, but also a sizeable number of members of the embryonic field that would subsequently become known as computer science,[4] leading to outrageous and unfounded speculation about what computers were "soon" going to be doing for us in all sorts of domains. I'll have more to say about that topic further along. In any case, I think I knew from that night onward that one day I wanted to be involved in the kind of work that Farley and Clark were describing.

Little did I know what lay in store.

Howie and I were both avid rock-climbers. That fall (1954) saw us clambering over cliffs that we discovered on the periphery of Pittsburgh. Soon we bumped into a group known as the Pittsburgh Climbers who introduced us to climbing areas further afield in West Virginia, as well as to local ski areas. When spring skiing season arrived, Howie and I arranged a trip to old haunts in New England, and on our way through Boston we stopped to visit Whirlwind to

[4] Neither the name nor the discipline would make its appearance for several years, and when it did, many argued that to call it Science was misleading since it was really "only" engineering!

let me get a look at the machine for which I'd been writing exercise programs.

It was an impressive and memorable visit. The thing didn't look like a pinpoint after all. In fact it filled several floors of the Barta Building at MIT and great bundles of intestine-like cables traversed holes in the floors and walls between rooms. The main memory (all 4,096 words of it!) had originally consisted of 32 registers of switches that had to be set by hand. These were then replaced by 256 registers of electrostatic storage tubes, on the surface of which bits of information were stored as tiny charges. But these about-to-be-obsolete devices had recently been replaced by an experimental magnetic-core memory in which the bits of information were stored in planes of tiny magnetic doughnuts, polarized this way or that to represent the zeros and ones that the machine used to encode instructions and data. This core memory (initially 1,024 words) had been developed not long before by Jay Forrester, who headed Project Whirlwind. (Bill Papian, a graduate student of Forrester's who had done much of the actual work of building and testing the prototype core memory, would one day become my boss.) A special Memory Test Computer (MTC) had been built expressly to try out the new memory—which proved so successful that it had promptly been moved to Whirlwind where it now resided. A variety of less successful devices had been tried in other machines. The development of core memory was to spur the growth of digital computers by providing the standard main memory technology for the next decade and a half.

The machine's main registers, however, were constructed from flip-flops which were very fast memory devices, much faster than the core memory. These registers, consisting of groups of flip-flops, were where the real action

took place as the computer charged along executing instructions, one after another, from the main (core) memory. Each instruction typically involved a series of steps that took place within and between the registers, so the instruction steps, and hence the registers, needed to work significantly faster than the core memory itself. Someone unplugged a spare flip-flop from Whirlwind's gigantic racks and handed it to me. It was about eighteen inches square as I recall, perhaps four inches thick, bristled with vacuum tubes and other inscrutable electronic components, and weighed maybe eight or ten pounds. It seemed pretty compact and if anyone had suggested then that within thirty years you wouldn't be able to find such a thing without a microscope, there would no doubt have been great whoops of laughter.

But all those tubes failed with great regularity, producing a healthy paranoia on the part of engineers of the time. To cope with reliability problems, a system of *marginal-checking* was devised in which the voltage provided by the power supply could be varied on a unit-by-unit basis in order to reveal incipient failures in elements during testing. Marginal elements could then be replaced before they failed during actual use and gave wrong answers. Engineers who cut their teeth under these circumstances were to retain their sensitivity to reliability in their later work and the early development of projects such as the ARPANET (see Chapter 14), benefited greatly from deep concern about reliability issues.

At the time I wasn't fully aware of the fortuitous timing of my encounter with Howie. I didn't realize how near the headwaters we were, how recent were a number of major developments that would propel the forthcoming rapid evolution of computers. It was little more than a decade

earlier, under the pressure of World War II for more and better computation, that the first attempts to build *any* kind of electronic computer had been undertaken. The result was ENIAC. Earlier machines had used relays, electromechanical devices, to store information, but their mechanical properties had drastically limited their speed and reliability. The ENIAC used vacuum tubes and could run programs many times faster than any previous machine.

In those days computers were still thought of principally as mechanisms for computing functions of variables—things like X = a(b2 + c3), and of course far more complicated kinds of expressions. For some time there had been fixed purpose devices that could perform a very limited set of operations, for example, a desk calculator that could add, subtract, multiply or divide pairs of numbers. In the 19th century, an Englishman named Charles Babbage had taken a tremendous intellectual leap by suggesting that one should be able to construct a *general purpose* machine, one that could be made to do a wide variety of different computations using the same basic machine structure. The functions themselves, the rules for what was to be computed, seemed to be comparatively fixed and, in ENIAC's day, were thought of as more or less a part of the machine's definition. Of course even though it was comparatively stable, it had to be possible to change it for different computations, and so in early computers, switches and plug-boards, into which a nest of cables could be plugged by hand, were used to define the sequence of operations that the machine was to follow for a particular computation. Setting up or changing programs defined in this way was an unbelievably ponderous operation. (My wife, whose father, the mathematician D.H. Lehmer,

worked on the ENIAC, tells me that as a child she and her brother were among the world's first *un-programmers*. It was their task to remove the wires from plug-boards and sort them by length when it was time to reprogram the machine.)

Creating anything as complicated as a computer requires a multiplicity of skills. Rarely do all of the necessary talents exist in a single individual, and in the design of dramatically new and innovative computers it has often been the case that a collaboration has arisen between individuals who conceive the overall machine and those who have the specialized electrical engineering ability and imagination to turn that conception into something real that works. The rôles are not really separable, of course, and each must play in the other's pond to a large degree. The relationship is not unlike that between an architect and a building contractor when a structure of dramatically new design is to be erected. Each participant, of course, must have the capacity to understand the other's discipline, but not necessarily to deploy it with full force and imagination. In the case of the ENIAC, John Mauchly was the conceptualizer and J. Presper Eckert was the engineer/builder. This is, of course, gross oversimplification. Building a new computer invariably involves an entire team of specialists as well as numerous subcontractors, specialized parts manufacturers, and fabricators who provide all of the individual pieces.

In 1944, along came John von Neumann of Princeton's Institute for Advanced Study, who was helping to develop the mathematics for the atomic bomb. That work required enormous amounts of computation, and although the ENIAC wasn't powerful enough, and reprogramming was extremely cumbersome, it nonetheless looked like a

promising point of departure. Von Neumann, working together with Eckert and Mauchly and other senior members of the ENIAC group, had come up with the design of a new machine which they called the EDVAC. EDVAC incorporated two innovations the importance of which would be hard to overstate.

The first was abandonment of any attempt to force the decimal number system into the machine's innards as had been done with ENIAC. Human beings have, by and large, settled on the use of decimal numbers so firmly (probably because of their ten fingers) that most people are surprised to learn that the choice of ten symbols (0 through 9) is, in fact, arbitrary and that larger or smaller sets of symbols, leading to other numbering systems, are possible and even more convenient for certain tasks. In Eniac's time, programmers, being used to decimal numbers, were most comfortable defining their instructions for the computer in decimal terms. The underlying hardware, however, could be made much more reliable if it had to distinguish between only two states, which meant that the machine represented information internally in binary form where only two symbols (zero and one) are used. To resolve this discrepancy, ENIAC's designers had attempted to mask *in the hardware*, the underlying binary nature of the machine in order to present the façade of a decimal machine to the programmers. Doing the conversion necessary to create the façade had vastly complicated ENIAC's hardware, so the decision to push the conversion into the software in EDVAC allowed great simplification of its hardware. Programmers quickly adjusted to working in binary and as the uses of computers have extended far beyond the

solution of numerical problems, the early attempts to make machines appear decimal now seem ludicrous.[5] Of course today *users* (as opposed to programmers) employ decimal numbers all over the place and have no need to be aware of, or to deal with, the underlying binary machine.

The second innovation was the decision to store not only the data but also the program itself in the memory of the machine. Today the usefulness of this seems so transparent that we can't imagine doing it any other way, but it was not always so. In thinking up the idea of putting the program into the memory as opposed to designating it by the temporary setting of switches and wiring of plugboards, the EDVAC designers opened the door to a flood of possibilities, only some of which they could have foreseen at the time. The most obvious benefit was the ability to switch programs rapidly, but of course it allowed far more flexible programming all around. A running program could operate on itself, changing parameters and even instructions on the fly. The ability to swap small pieces of large programs into and out of the main memory onto other, less-expensive media, would lead to virtual memory systems, multiprocessing, and a host of other techniques that today we take for granted. In a sense, it was the natural next step toward greater flexibility, but at the time, given the general unreliability of components, it was both a powerful insight and an act of considerable bravery.

In any case, the idea was of such profound significance that the stored-program computer, which is what everyone has used ever since, is still referred to as a "von Neumann machine." This gives more credit to von Neumann than he

[5] Although IBM later built a number of decimal machines.

deserves. He was the first to publish a description of the proposed new EDVAC machine and he signed it as sole author, but in fact Eckert had been thinking along these lines well before von Neumann showed up on the scene and they had worked together on the EDVAC design. Sadly scientists and engineers, even the best of them, are not immune from the urge to claim more than their share of credit.

Early in 1946, Maurice Wilkes, at the Cambridge University computer lab in England, saw a copy of von Neumann's report; a few months later he arrived in the U.S. in time for a series of lectures at the Moore School describing the machine more fully. Back to England he went, busily designing a similar machine of his own en route. He dubbed his machine the EDSAC in honor of EDVAC which had inspired it, and three years later, in the spring of 1949, it began functioning as the first such machine the world had ever seen. This was the very EDSAC described in the book that Howie had lent me, and that I'd been puzzling over just months before our visit to Whirlwind. The final piece of the puzzle, that opened the door to forthcoming generations of computers, had been the development of core memory at MIT.

In retrospect I find the felicity of my timing unbelievably fortuitous. These prior events, which were to have such widespread consequences, had all taken place within the decade preceding our visit to Whirlwind, some only months before. At the time Howie handed me the book about EDSAC, I understood that there were only a few such machines in the world and that they were all highly experimental. I don't think that I, or anyone else, realized the immensity of what was soon to happen and how rapidly it would all take place.

In trying to explain my new profession to my father (the word "programming" had not yet entered common parlance), I described to him what I knew of this prior history. When I mentioned Presper Eckert, to my surprise he said that a number of years earlier Eckert had been a dinner guest at our home. My family were friends of Herbert Welsh, and the Welsh's son, Frazier, was one of Eckert's closest sidekicks. He brought Eckert to dinner one night under the illusion that he and my dad, both geniuses in their own fields, might find one another interesting. But as Eckert was apparently a prototypical nerd and my father is a black hole so far as science and engineering goes, they apparently had had little to say to one another. As a youngster I had known Frazier during the period when he was helping Eckert with the design of the early UNIVAC machine, but his life was cut short by the crash of a glider he was piloting.

By the time I came along, the idea that one could use computers as something more than mere "computing engines" was spreading fast. It was the era in which computers were referred to as "Giant Brains" and speculation about what they would be able to do ranged wildly. Just as nuclear power was at one time envisioned as a magic solution to *all* of the world's energy problems, so computers were thought of as potentially able to solve all of our most perplexing intellectual problems. Nor was such speculation limited to the media. No one really knew where the limits, if any existed, would eventually be encountered, and even some experts indulged in over-zealous speculation. Such speculation still goes on today of course, but, except for a few daring individuals, no longer on the scale or with the wide-eyed naiveté of those earlier times. Gradually, as the technology began to mature and

applications proliferated, more and more people became involved and the naiveté diminished. That is not to say that we have arrived at the end of the road; far from it. It's just that expectations have become somewhat more realistic and computers no longer seem like magic. When gadgets such as Palm Pilots, hand-held GPS devices, etc., appear on the market, we may be amazed at their cleverness and usefulness, but it no longer seems like a miracle.

Whirlwind was the first machine that was fast enough to do interesting things in *real time*. Most earlier computers simply did their processing as rapidly as they could; it was hoped that they wouldn't take unreasonably long, but there were no hard deadlines. Real-time computing means that something outside the computer imposes fixed, short deadlines on the work to be performed. For example, suppose a computer is processing data from a device that generates the data at fixed intervals. Unless the computer is ready to handle the next piece of data when it arrives, the data will simply be missed—it will "fall on the floor" (to use computer vernacular). It's the same thing as someone working on an assembly line who mustn't fall behind. In some applications, e.g., control of aircraft functions, rates are high and failure to keep up can have devastating consequences. Careful matching of speed is critical in the design of such systems.

A particularly significant form of real-time computing takes place when a human user interacts directly with a computer. To be useful, the machine must perform fast enough, during the various interactions that occur, so that the human is not forced to work at an unreasonably slow pace. For example, when a contemporary user moves a pointing device such as a mouse, it's vital that the computer be able to keep track of the mouse's movement and reflect it

on the screen by correspondingly moving some visual indicator (a cursor). If the computer can't "keep up," the user's hand movement is not accurately reflected by movement of the cursor on the screen and the entire mechanism is ineffectual. Many other elements of interactive use demand that the computer respond in a timely fashion. Before one could consider using computers interactively through a display screen as we do today, they needed to become fast enough to cope with such use. Whirlwind was the first computer with powerful interactive display capabilities, and as such it was the great-granddaddy of all personal computers. It was in fact personal in another sense as well, since it was generally used by one individual at a time[6] and during that user's time, the machine was completely at his disposal. For many later machines, this was felt to be too profligate a way to use such expensive equipment, and different ways of sharing access would soon be explored.

During our visit, someone mentioned that MIT had recently opened a place called Lincoln Laboratory, in suburban Lexington, where qualified staff people were being sought to work with computers. Howie and I had both become somewhat disenchanted by the conservatism of the Gulf Oil company, and not long afterward Howie disappeared from Pittsburgh to go to work at Lincoln Lab. Before leaving, he suggested that the training he'd given me might qualify me for a job there as well. (Programming had not yet become a recognized profession and anyone with an aptitude for such things was a potential candidate.) A little

[6] Although when driving the Cape Cod System (which we'll encounter shortly) it was shared by multiple "users."

while later I somewhat doubtfully submitted an application, and to my surprise and delight, I was called for an interview.

I remember little of the interview itself except for two things. First, in the middle of the interview I was asked directly what my main shortcomings were. (I was so startled by the candor of this question that I think I was quite forthcoming.) Second, I was given a sort of test to take home and complete. It contained a number of simple logic problems and asked you to discuss some of the issues involved in designing an automated traffic light system for a city. There were no explicit programming problems, for the simple reason that most applicants could not be expected to have ever written a program. The test seemed not only easy and fun, but was thought-provoking as well. The next day a number of people interviewed me in what seemed a very offhand manner. Having been stress-tested with serious physics questions for employment at Gulf, I was perplexed to find myself enjoying the process. Was I being interviewed for a serious job? What was going on?

I learned that Lincoln had been set up to provide MIT research services to the Air Force. Foremost among these was the design of a modernized air defense system. Our uncomfortable alliance with the Soviet Union during World War II had been but a brief interlude in the over-arching hostilities between the communist Soviet Union and capitalist America. By this time that semi-religious war, having been dubbed "the cold war," was in full swing, and the consequent military buildup was proceeding apace on both sides. The need for an upgraded air defense system was a manifest part of the process.

There had been various studies and proposals for ways to upgrade the existing system. MIT had been pursuing a

method that would utilize a computer to handle a wide variety of relevant tasks—the tracking of aircraft (inter-continental ballistic missiles had not yet become a significant threat), assignment of weapons, control of interceptors, etc. A small prototype system that utilized Whirlwind, known as the Cape Cod system (some of the system's radars were on Cape Cod, peering out over the Atlantic) had been constructed within the Barta building at MIT where Whirlwind was housed.[7] It was a semi-automatic system which meant that Air Force personnel, wearing telephone headsets and seated in front of large consoles with screens and buttons and light-guns (a forerunner of today's mouse), interacted with the computer to operate the system. The Air Force had seen the prototype system demonstrated and, after the usual hemming and hawing, the powers that be had decided to develop a full blown version of it—which formed the raison d'être of Lincoln Lab. Lincoln was originally organized in 1951, on the MIT campus, and moved to Lexington in 1952.

Numerous stories, some no doubt apocryphal, emerged from the Air Force reviews of the Cape Cod system. One that I particularly like involved a General who, understanding little of what he was seeing, nonetheless felt compelled to manifest interest. He stepped over to a young Air Force operator who happened to be overseeing a radar monitor and asked him to explain his job. "Well Sir," he said, "you see that little red light there? You see this button

[7] Previous Navy support for Whirlwind had been under threat, and the adoption of Whirlwind by the Air Force was quite critical to the continued development of Whirlwind, the legacy of core memory, and IBM's construction of the follow-on XD-1 (see below), etc.

here? Whenever that little red light goes on, I push this button and the red light goes off." Radars sometimes pick up a lot of irrelevant noise and the red light was a data-overflow alarm. The button suppressed the alarm. It's not recorded whether or not the General pursued the matter.

The follow on system to be developed by Lincoln Lab was dubbed SAGE, which stood for Semi-Automatic Ground Environment. (Acronyms, I was to learn, are the life blood of the military.) At the center of the system stood a giant new computer, the XD-1, designed cooperatively by Lincoln and IBM. By the time I arrived on the scene, the basic machine was already in place in a gleaming, windowless, super-secret building behind the main lab which housed a full-scale prototype *Direction Center*—computer, consoles, and all. Alas, no operational program inhabited the glistening new core memory and it was in order to correct this deficiency that the lab was seeking people like Howie and me.

Despite dire predictions by my disgruntled Gulf employers that I was letting myself in for what would soon become a sweat-shop operation with hundreds of programmers forced to scribble away in giant bull-pens, within weeks my family and I were on our way to Boston.[8] After leaving school in Cambridge, I'd been banished by the need to earn a living first to the oil fields of New Mexico and Oklahoma, and then to the wilds of Pittsburgh. Now, in the summer of 1955, we were on our way back to the part of

[8] Gulf was so disgruntled, in fact, by the departure of Howard and me and a couple of others to MIT, that when a short time later a geologist pal of mine came to solicit financial support for MIT, he was none too politely shown the door.

the world I knew and loved, within easy reach of the outdoor joys of all New England. I couldn't believe my good fortune.

And I still had no idea what was in store.

Chapter 2

In which I arrive at Lincoln Lab, am introduced to the wonders of air defense, security, and other matters, and worry about my children's future.

The SAGE system was the first really large-scale programming task undertaken anywhere by a sizeable crew of programmers. (Not surprisingly it exposed many of the communication problems between programmers that are still with us.) Shortly after I arrived at Lincoln, a course was given to teach programming to us novices. In those days computer people came from all sorts of disciplines. Programming as a profession hardly existed; there were virtually no experienced people available, nor any computer science departments to teach them. So people had to be recruited from other walks of life and given on-the-job-training. As we gathered I had an opportunity to inspect my fellow students. We were a motley crew consisting of everything from musicians and historians, through chemists and mathematicians, to a handful who had actually already written a program or two. I was pleased to find myself in the latter category and came gradually to recognize that I actually had a gift for this new craft. With Howie's training behind me, and finding considerable similarities between Whirlwind and XD-1 (the machine with which I was now faced), I breezed along.

After a week or so, it became apparent to everyone that the course was redundant for a few of us; we were culled

and given other assignments. It seemed that not only was an operational air-defense program lacking, but the overall system hadn't yet been fully designed. The OP SPECS (Operational Specifications) which defined the system were just being written, and with no more background in air defense than a woodchuck, I was unceremoniously handed the task of writing the Crosstelling Spec. What in God's name was Crosstelling? The only thing I knew about it was that it came late in the schedule, thank heavens, after everything else was finished.

It developed that the country was divided into sectors, and that the sectors were in turn divided into sub-sectors (which were really the operational units) with a Direction Center at the heart of each. Since airplanes, especially those that didn't belong to the Air Force (or even the U.S.), could hardly be forbidden from crossing between sub-sectors, some coordination was required for handing over the tracking of planes, controlling of interceptors, etc., between the sub-sectors. This function was called Crosstelling, a name inherited from an earlier manual system in which human operators followed the tracks of aircraft on radar screens and coordinated matters by talking to one another on telephones. Now it had somehow fallen to me to define how this coordination should be handled by computers, and then to write it all down in an official OP SPEC with a bright red cover stamped SECRET.

I was horrified. Not only did I feel incapable of handling the task, but what was to become of a country whose Crosstelling was to be specified by an ignoramus like me? My number two daughter was born at about that time, and for the first time I began to fear for my children's future.

I want to pause here to relate a couple of anecdotes that help to explain my later sentiments about security rules. While I was getting an introduction to SAGE, my pal Howie was writing the manual describing how radar-data input to the machine was handled. This was classified information, so as Howie produced each page, it was quickly stamped SECRET. Unfortunately Howie's clearance had temporarily lapsed, and consequently he was not allowed to proof-read the sections he'd just written.

An even more ironic incident took place the day a SECRET stamp arrived on the desk of another member of the lab, presumably to cover the possibility that he might write down a SECRET *thought*. Curious about the stamp, he tried it out on the blank top sheet of the giant pads that sat atop all of our desks (used for doodling and other vital government work). That night he was awakened from a sound sleep by a telephone call from "Security." He had left a "SECRET document" on his desk, requiring that he get out of bed, drive to the lab, and put the document away in his securely padlocked file cabinet.

Despite such ironies, I soon started to get to know my confreres. My immediate superior was a wonderfully relaxed and friendly Japanese guy, Jiro Ishihara. (Japanese? Hadn't we recently been at war with those guys? What was one of *them* doing defining our new air-defense system?) I had a lot to learn, it seemed. And Ish, it turned out, was just the one to teach me. He was wonderfully supportive, and suitably irreverent. I learned an enormous amount in a hurry, much of it by osmosis. One thing I learned was that the Direction Centers were to be connected by high-speed telephone lines that could transport 1,000 bits of information every second—*every second, mind you!*—and in each direction! Why in five minutes, we could ship 300,000

bits! I began to relax. With such incredible capacity, we could probably do the job. (The tiny modem in my home computer today handles 56 times this amount; high-speed communication lines, thousands of times.)

The weeks went by and gradually I learned more. Ish lived not far from where we did, in a similar duplex. He invited me to join him one afternoon and, as we lay on chaises in his sunny back yard discussing work matters, I was struck by the difference between this job and the one I'd held previously in which work was strictly relegated to the office. Here I was coming to learn that the boundaries were comfortably permeable—it was assumed that you had some genuine interest in your work that persisted even beyond the confines of the office and 5 PM. Sometimes we would take off in the middle of the day, go somewhere for lunch, and then head to the Barta building at MIT where some exercise of the Cape Cod system was under way. I much preferred this easy interplay of work and living.

One day there was an open-house demonstration of computer facilities at MIT (not, of course, including the Cape Cod system, which was classified). A new core memory had been installed in MTC, replacing the one that had been moved to Whirlwind, and a program had been written that enabled it to play music (by carefully switching the inputs to an amplifier back and forth at just the right frequencies). This was my first encounter with, and must have been a very early instance of, the use of a computer to play music. Artificial speech would come much later. A somewhat grimy but broadly-smiling individual was overseeing the demonstration. Although I had no way of knowing it then, this fellow was to appear at various times throughout my future and, in the process, would become a

lifelong friend. His name is Tom Stockebrand and we will encounter him again later.

Eventually I managed to write a respectable specification that was duly reviewed by the Air Force personnel attached to Lincoln, whose difficult job it was to oversee the definition of the system. It was a back-and-forth process in which each side educated the other; they educated us about air defense and we educated them about what a computer could and couldn't do. Such interactions, between computer experts and customers with a problem to solve, were to become commonplace in the years ahead, and already the occasional arrogance of computer people, thinking they understood the customer's needs better than the customer, was beginning to manifest itself.

Once the specification was completed, it was time to write the program. The Systems Development Corporation (SDC) had become a partner in the SAGE development effort and it was their responsibility to write the actual code. So I spent time working with one of their programmers, conveying the specifications to him. Thereafter crosstelling gradually drifted away from me as I became involved in other matters. I had become interested in exploring various ways in which the computer could track aircraft in the presence of noisy radar data. (I believe SAGE would have failed utterly in the presence of active "jamming" by an enemy, but fortunately it was never put to the test). I was conducting some experiments on XD-1 when someone spilled a cup of coffee down the throat of the card reader (the only program entry device). Everything came to a halt as the IBM technicians took the reader completely apart and cleaned every piece. After two days of this, glass partitions and Guards and Rules were put in place. No further coffee need apply.

Emergency main-power-off buttons occupied the ends of every one of the numerous long racks of computer electronics. One night a janitor, wielding his broom too vigorously, accidentally bumped one of them, turning off all power to the Direction Center. For days thereafter, technicians could be seen going through the machine replacing the buttons with recessed versions, thereby protecting the nation from errant broom handles. You can think of this as an early forerunner of those "Are you sure you want to…?" messages that appear on your screen when you're about to take some irreversible action.

Chapter 3

In which I make some comparisons and mark some contrasts between "back then" and "now."

In the "middle-ages," what computer industry existed bore little resemblance to that of today. Many of the differences have come about as the result of the enormous proliferation of machines. This has, of course, been a snow-balling process: as price comes down, demand and quantity increase; as quantity increases, price comes down further, etc. Producing things in small quantities doesn't justify specialty shops; it's only when quantities become large that it makes sense. And so the nature of both design and production have changed dramatically. In the 1950s there were no chip manufacturers because there were no chips. Although they bought small electronic components (resistors, capacitors, transistors, etc.), the few relatively large computer manufacturers tended to make many of the parts they needed, such as printed circuit boards, themselves. There were no huge overseas manufacturing facilities of the sort that exist today. Asia was barely emerging into the 20th century having been badly damaged as a result of World War II. There were standing jokes about the Japanese only being able to copy things, not devise them. But these attitudes soon disappeared as Japan rapidly rebuilt following the war and, with it's then relatively cheap labor costs, quickly took over the manufacture not only of television sets but also of the entire cornucopia of emerging

household electronic gadgetry—not to mention automobiles.

The rôles of people were very different then as well. Start-ups and today's entrepreneurial fervor all lay ahead. If you wanted to work in the investigative part of the computer field, exploring new terrain, there were really only two choices: you could join one of the large computer manufacturing firms (most likely IBM), or you could associate yourself with some Institute or University where research would be funded with government money of one sort or another. While the profit motive certainly underlay much of what went on, there was, in my own experience, a much larger element of sheer exploratory excitement than there is now. People were not working on products—they were working in particular areas of computer research, testing which ideas were viable and which were not. Many ideas that seem laughable in retrospect, had to be tested before their flaws became apparent.

Gradually the wheat was winnowed from the chaff and as the design of the basic elements began to settle down, the search for ways to exploit them began to intensify. Of course people had been considering possible applications for computers for a long time, but the mere size of earlier machines precluded the exploration of most of the kinds of applications that exist today. It was the advent of the computer on a chip (or a few chips) in the late 1960s that opened up a world of new possibilities, not only for small personal computers (and ultimately laptops) but for applications in all kinds of devices from automobiles and washing machines to a plethora of hand-held devices. Given the exponential rate at which the computer enterprise has grown, it would be nearly impossible to point to a knee in the curve, let alone a beginning. Nonetheless, looking

back forty years or so, one can say that at that time, nothing comparable to the present industry existed. And as a consequence, the rôles that individuals served, their ambitions and satisfactions, were all very different from those one finds today.

Over the last few years we've watched as a friend and neighbor, a young Silicon Valley Turk, set about forming a new "startup" in the approved manner. For a time he sprouted ideas right and left for a possible new enterprise, until one finally grabbed him—the idea of making an electronic book, something you could carry around with you easily, that would hold a number of books, be more readable than your typical laptop, and embody only features and capabilities related to reading (as opposed to the cornucopia of features in a portable computer.) He then pulled together a group of his buddies, rented some office space, started refining ideas, building rough prototypes, and gathering initial funding. It was today's classic story. Back in "my time" such a person would have been a top-notch member of some research lab, but today, when so much of the terrain has been explored, relatively few such labs still exist. Instead the bright people are now utilizing their brainpower to create new products (and fortunes) from leading-edge technology. In one sense, this is not so different from what we did in building new kinds of computers using the then-new transistor technology. But today, the really new things that are being created are not themselves computers, but rather devices and gadgets with a microprocessor buried inside. Another major difference is that most of our work was either government sponsored research or was directed at specific customers with particular applications. By contrast, a large fraction of

today's products are aimed directly at the consumer market, i.e., the general public.

Back when I was a kid, we could all identify the makes of cars at a glance; you didn't have to look twice to tell whether it was a Ford or a Chevy. Looking at modern automobiles, it's hard to tell one from another. Of course there are now all kinds of different models—convertibles, sedans, vans, station wagons, SUVs—all easily distinguishable. But within each of these categories variety is surprisingly limited compared to earlier times. Why? It's no doubt partly a matter of conservative and highly competitive marketing that tends to produce rather uniform styling. But style itself has been shaped in no small part by engineering considerations that have have come to be generally understood and accepted over the course of many years of experimentation. Back when the constraints were less well understood and therefore less stringent, greater experimentation resulted in more diversity. Gradually the nooks and crannies of the design space were explored, the blind alleys discovered by everyone and foreclosed, and the design space thus narrowed. Of course there are still distinctions based on cost, but increasingly the features that distinguish luxury from economy models are revealed only on close inspection and manifested mostly in special gadgetry.

The same sort of convergence has occurred over the years with a host of other items from airplanes to toasters as the design constraints (including ergonomics) have become better and better understood over time. It's a bit surprising that the design of something as complex as a computer has settled down so rapidly to a similar kind of superficial uniformity. As with cars, there are now a number of different models—laptops, desktops, minicomputers,

mainframes, supercomputers, etc. But in the smaller versions, where most of the proliferation has occurred, you have to look carefully at special features, mostly having to do with speed and memory capacities, before you can distinguish the various hardware offerings from one another. Although they are certainly different in their software and in their methods of operation (and *maddeningly* incompatible with one another), nonetheless, in terms of basic user facilities, they're all much the same, each of them presenting you with a screen, a pointing device, and a keyboard, all of which unambiguously suggest highly interactive use.

This was by no means always the case, and part of my task is to explain how this style of usage arose from very different beginnings. Understanding history is important because it makes us less cocksure about the present, let alone the future, and helps us to understand that change is the only real constant. Despite the uniformity that presently exists, and the hype and euphoria surrounding the personal computer revolution, it may well be that the explosion in numbers today arises less from the perfection of current design than from the fact that they've become good enough to permit them to be produced and utilized in quantity. But we shouldn't forget that significant changes and improvements may well lie ahead, and even some designs that lie outside our current imagination.

The computers that existed around the time I arrived at Lincoln, in the mid-fifties, bore no resemblance whatsoever to the computers most people are familiar with today. Many of them cost millions of dollars and filled large rooms, and even the "smaller" ones cost many tens of thousands of dollars. But there's more to it than just the size and cost and general appearance. Almost everything was

unrecognizably different, not just the machines themselves but the way they were used, the type of people who used them, and what they used them for.

In the early fifties there were practically no computers around to speak of, at least by today's standards.[9] The few that existed were all different from one another; programs written for one would run only on that machine and no other. They also broke down a lot. It was a bit like the early automobiles for which the driver needed to be something of a mechanic. The people who worked with these early machines generally knew them inside out. Gradually there came to be a separation between hardware people, who specialized in understanding the details of the underlying machine, and programmers, who specialized in writing the programs that went into the memory and were executed by the machine. I certainly encountered many programmers who had little or no idea how the machines they were programming actually worked—I was in that class myself for a number of years—and, perhaps more surprisingly, I met a number of hardware designers who never could and never did write a sensible program. But for some years there was no distinction between a *user* and a *programmer;* users wrote their own programs to do whatever job they or their superiors wanted done. The distinction arose only many years later with the proliferation of machines together with some understanding of the common tasks for which

[9] Since writing this, I have been surprised to learn that a 1957 survey indicates that by then there were already roughly 5,000 machines in the United States alone including a tiny number of "small" (i.e., on the order of $80,000 apiece) computers based on magnetic drum memories. By leaping into the field at Lincoln, I came in contact primarily with some of the larger machines, which gave me a biased picture of how things stood.

sizeable numbers of non-computer-savvy people might want to use them.

In the early middle ages, just as there were only a few machines, there were very few people who understood and dealt with them. They constituted a "brotherhood" of cognoscenti. (Indeed not all of the people involved were male although most were. Nonetheless, I hope I will be forgiven for eschewing the word "personhood.") As the number of experts grew, the average level of scientific distinction gradually diminished from the early days of such giants as Alan Turing, John von Neumann, Norbert Weiner, and Vannevar Bush. In the early-to-middle middle ages, there were still few enough knowledgeable people within the technical community that many of them tended to know one another. Despite strong differences of opinion, they were working—in the larger sense—collaboratively, toward common goals. Today's nerds constitute an altogether different breed of cat. They are, for the most part, working competitively in the marketplace. There are infinitely more people involved and, as there can be only a limited number of celebrities at any given time, the ones who today tend to be widely known are the few who have percolated into the economic stratosphere. I don't mean to suggest that there are any fewer extremely bright and creative computer people than there used to be—quite the reverse. In their millions, they have become indistinguishable in the crowd, whereas in the middle ages, the few there were often knew one another and stood out like sore thumbs .

The computers that most people today are familiar with are the modest-sized, modest-priced personal machines. There are, of course, larger, more powerful computers for big jobs, but none covers the kinds of acreage that the

gargantuan old computers occupied. They filled rooms and covered walls that were often plastered with panels containing zillions of flashing lights. They were big because the pieces from which they were constructed were big. Big components meant that cables and connectors were required to join the pieces of the machine together, and designers quickly learned that, as my later mentor Wes Clark put it: "nature abhors a connector." But aside from the unreliability of the cables and connectors that hooked the various pieces of the machine together, the individual pieces were themselves far less reliable than today's electronic components, which often contain more electronics on a single chip than could then be fitted into a large building. Despite the marginal-checking mentioned earlier, the mean time between failures of the overall machine was sometimes hours or even minutes. The flashing lights were there because they provided the only visible clue as to what was happening—or, more frequently, to what *had* happened when the machine ground to a halt. When that occurred, one performed what was appropriately called a postmortem, using the switches and buttons to probe around carefully in the remains, trying to understand what had gone awry.

By the time I showed up at Lincoln Lab in 1955, this situation was already changing rapidly. Commercial computers, still big and expensive but of greater uniformity, had begun to appear. Although there were some competitors (Remington Rand, General Electric, Control Data, etc.), IBM (partly as a result of their involvement in SAGE which gave them the jump on memory technology) soon came to dominate the field. The situation was often referred to as "IBM and the Seven Dwarfs," the dwarfs being the other computer manufacturers.

For a number of years IBM had been manufacturing and selling card-processing equipment that in some very general sense could be thought of as "computers." But other than giant IBM, few places (of which thanks to government money Lincoln was one) could afford to experiment with building machines of their own design. In the case of XD-1, the machine was built by IBM, but due to the unusual application and its requirements for connecting to large numbers of special terminals, radars, etc., many special features existed, some of which had been designed, at least in part, by Lincoln people. XD-1 was a unique beast, the prototype for a later production version of an air-defense computer that would become known as the AN/FSQ-7, copies of which were eventually installed inside Direction Centers around the country.

The big machines formed a gravitational center for groups of programmers who used them. Within a group there was a lot of voluntary cooperation and sharing of programs. Indeed, organizations tied to particular kinds of machines, such as SHARE for IBM and later DECUS for DEC computers, maintained libraries of donated subroutines (chunks of program that performed frequently needed tasks) that were available to all.

Because they were few and expensive, access to the most powerful of the early computers was hotly contested. In fact it is only in relatively recent times that costs have come down to the point where contest for access has become less of an issue. Most people today think of a computer as something you turn on and off as you happen to need it, although of course that's not true of the big machines used for functions such as Air Traffic Control and by businesses, which typically run continuously in the service of large scale operations of one sort or another.

During much of the period I am describing, many of the machines were big and expensive, and methods for extending or sharing access were being explored. Debate about how this should be accomplished forms a major thread of computer history that commences in the mid-fifties when styles of usage were already beginning to diverge.

The style of use for most of the large, expensive machines might best be characterized as "hands-off" computing. The user and the computer were deliberately insulated from one another, the unstated but underlying motivation being to protect the machine from the users and to keep the costly beast busy rather than allowing it to sit there idling while some programmer mulled over his latest program bug. Expert intermediaries ("operators") handled the machine; on a good day a programmer might be allowed near the operating console to watch what was happening as the program ran, but was not allowed to touch the switches.

Much of the work done on a computer in the early middle-ages involved debugging of programs. There were, of course, some "production runs" in which programs that had been debugged were used to process real live data or perform some calculations, but the majority of such runs took place in the wee hours when at least some of the programmers were asleep. A substantial amount of preparatory work took place long before you were ready to approach the computer. You wrote programs with pencil and paper on giant pads of coding sheets. Sophistication had advanced to the point that you no longer had to write programs in the ones and zeros that the machine understood directly. Instead you spelled out each step in something called assembly language. Every program step

corresponded to a single machine instruction, but at least the names of the instructions and the referenced memory locations could be specified mnemonically. These would then be translated by the Assembler program into the machine's native binary language.

Whenever I design anything—from a house to a program to a computer—I find that I need to have a picture of the whole thing laid out in front of me in order to fit it together in my head. (Other people seem to manage seeing just one piece at a time.) The programs I was writing were tiny by today's standards, but nonetheless covered many sheets of paper and I needed a giant wall upon which to paste them up. The walls of my office at Lincoln were totally inadequate, but at home I had some sizeable empty wall space, so once the general design of the program was worked out in major steps, I would often retire to work at home as I wrote out the detailed code. The walls of our dining room were thus often covered with flow diagrams that depicted the broad design, and with coding sheets containing the numerous detailed steps. My style was by no means unique.

Once you finished writing the program, the coding sheets were handed to someone whose job it was to transfer the information onto punched cards or tape.[10] Here we come to the women's rôle in earlier computing, for the key-punchers were often poorly paid women and the work was

[10] Whirlwind, like its descendant TX-2, used punched paper tape rather than punched cards. The now largely forgotten photoelectric paper tape reader sucked paper tape through its jaws at a terrifying rate. In an attempt to minimize punching errors, most tapes were punched twice, and the duplicates "verified." Also forgotten now is the battle between the rectangular holes of IBM cards and the round ones in Univac's cards.

painfully tedious. There were, of course, some outstanding women programmers, but I recall few male key-punchers. Only when you had a deck of cards in hand did you sign up for computer time. At this juncture the two styles of use diverged and as my first serious encounters with computers involved the IBM style of access, I will describe that process here and postpone for the moment discussion of the more civilized approach.

When your allotted time arrived, you bore the deck of punched cards to wherever the computer was located[11]. There using a console bristling with lights and switches the operator ran people's "jobs" in a sequence called "batch processing" mode. In order not to waste a millisecond of precious machine time between jobs, your bundle of cards was concatenated with those of other hopefuls—separated by special "job cards" that indicated to the supervisory program just what was to be done to the ensuing group of cards—and loaded into a card reader, where either little fingers or photocells sensed the holes and fed the information on the cards into the computer's memory. Once there, the assembly program translated your symbolic instructions into binary instructions which the machine could understand, and then punched out a more compact deck of cards in binary format.

At last it was time to try running the program. The switches were used to start the program going, at which

[11] The holes in cards were, of course, nothing more than just that and could be interpreted in any number of ways. Apparently the number theorist D.H. Lehmer, was carrying a large deck of program cards, when he encountered someone who asked him, "What have you got there?" His answer was, "A number."

point the programmer stood back, listened to the rumbling of the air-conditioning, nervously chewed his or her fingernails, and hoped that something useful would emerge from the jackhammer-like line-printer over in the corner. If all went well some useful results might appear, but most runs ended badly with the program immolating itself in one fashion or another. At that point the state of the machine was "dumped" to the printer, the programmer's run was finished, and the ensuing hours or days were spent poring over inscrutable printout, often in base 8 (octal) numbers, trying to decipher what had gone wrong. Once the problem was located and corrected, you applied for another "run." This pattern of activity was repeated over and over again, often for a period of days or weeks, until finally the program had been whipped into shape and began functioning satisfactorily.

A substantial price was paid for this sort of operation. Debugging was a slow and painful process that could extend over weeks or months, depending on the size and complexity of the program. Often programmers would realize within minutes what trivial thing they had done wrong, but would nonetheless have to wait for hours, or more likely until the next day, to try again. The consequence was that programs which might have been debugged in hours often required many weeks.

Over time minor improvements were incorporated into the procedure, mostly to minimize wasted time between jobs. Later systems used spooling programs in which cards were converted to tape on a peripheral computer, the tape was then put on the big machine where the program was assembled and run, and the output went to tape for later listing on a line printer. Despite such improvements in utilizing the computer efficiently, a lot of one's life was

consumed in trudging around carrying (and occasionally dropping!) heavy decks of cards; waiting for one's turn; poring over printout; bantering with the computer operators and the people who punched the cards; sometimes punching an extra hole or two in a card yourself, or even (don't tell IBM!) gluing a little piece of card material back into a hole in a binary card to shortcut reassembly. Overall, because the process was so cumbersome and involved the programmer with a variety of specialists, there was far more social interaction per debugged instruction than takes place today. The evolution of the comparatively asocial nerd, in isolated partnership with his machine, still lay in the future. But the important point is that in that world, programmers were the drones; the machine, the queen.

There were a few individuals who disagreed with this entire set of attitudes and felt that the price of such cumbersome operation was far too high for the supposed benefits. These people were convinced that the size and cost of computers were bound to fall, making such shoe-horning of multiple users unnecessary. And so a very different style of usage existed in a few research-oriented settings such as Whirlwind's, where much closer interaction between user and machine was the norm. In such settings, the individual users spent comparatively long stretches of time with the machine, typically operating it themselves, and identifying and and correcting program bugs at a high rate so that programs converged much more rapidly to correct performance.

The divergence between these two approaches reflected a major philosophic schism. Although Whirlwind's more direct, personal style of usage persisted, it remained comparatively rare throughout much of the middle ages.

Eventually, however, it led to the very first personal computers. But I'm getting ahead of my story.

One might ask what has eliminated the need for the kind of protective insulation that the more typical style of batch-processing usage enforced. The answer lies in a number of things. First of all, if you do something that causes trouble with your own, truly personal, computer, most of the cost will be borne by you, not others. No great expense is involved, probably no one else will be hurt. This contrasts with the scene in which a large, very expensive machine was shared sequentially among many people. If something you did caused trouble or even delay, the cost could be enormous and hurt many other people.

When you're initially writing and testing programs, failures of unexpected sorts are virtually certain to occur. Today, with so many people using computers for so many things, there is no way for all of them to be programmers. Instead, most people who use computers aren't writing and debugging programs at all but instead are using application programs that provide their only interaction with the machine. These programs have (presumably) been carefully crafted by programmers to allow the user to perform certain kinds of common tasks: email, word-processing, spreadsheets, bookkeeping, graphics, network access, and the myriad other jobs for which large numbers of people today use computers. In using the machine for these purposes, you no longer fiddle directly with the innards of the machine by pushing buttons and setting switches. Rather you manipulate programs indirectly through devices (mouse, screen, keyboard) and mediating programs that have been carefully crafted to prevent you from accessing parts of the hardware or software that could cause trouble. Much of the protective insulation that was

formerly provided by people and regimented procedures is now enforced by the operating system and the application programs of the computer itself. In olden times such insulating software simply didn't exist, leaving the machine much more vulnerable to user (programmer) errors.

So, in theory at least, the application programs that users interact with have been carefully debugged before users get their hands on them. Debugging takes place behind the scene, out of sight, in labs dedicated to designing and writing the systems and application programs that people will later use. Furthermore, today's debuggers utilize programming and debugging tools that were undreamed of in the 1950s. These have dramatically reduced frustration and shortened the debugging cycle.

Well then, why aren't things better today? Why do our machines still crash? There are at least three answers. First of all, they *are* better. Todays users, who expect their computers to be as reliable as any other piece of consumer electronics, can't possibly imagine how flaky computers used to be. Second, Parkinson's Law: As our abilities expand, so do our appetites. And as memory has gotten cheaper and more plentiful, our desire to fill it with new, larger, "improved" programs has more than kept pace. And as the refinements and embroidery increase, they bring with them new and more subtle interactions and failure modes. And finally, economics. If all programs were tested as thoroughly as the space shuttle's computer programs, we would have far fewer failures than we now have. Unfortunately such thorough testing is expensive and takes time, two things that are intolerable in a fast-moving, highly competitive marketplace. Experience indicates that today's debugging is often far from thorough. So of course, crashes still happen. Nonetheless, such occurrences are today

considered outrageous rather than routine and (one hopes) can eventually ruin a company if allowed to get out of hand.

But if computer users in those days were all programmers, presumably conversant with the machine, why did it need to be protected from them? First of all, by the late '50s most programmers weren't any longer so knowledgeable about the innards of the machine hardware, and, because of the increased contention for access to these rare and expensive gadgets, it was vital not to waste a precious moment. The IBM regimen meant that all thinking was done *away* from the machine so that it never sat idle while programmers scratched their heads puzzling over something. More users could thus be accommodated per unit time. Beyond that, IBM thinking simply demanded regimentation: preserving order; limiting people to their allotted time; keeping them from breaking switches, spreading grubby fingerprints, spilling coffee; making sure they collected suitable postmortem information, etc. This was sometimes referred to as "80-column thought," referring to the rigidity of the 80 columns of an IBM card.

In 1957 IBM introduced an entirely new kind of computer language called FORTRAN (FORmula TRANslation), which broke the one-to-one correspondence between the steps that the programmer wrote down and the steps that the machine executed.[12] A program known as the FORTRAN *compiler* translated programs written in the FORTRAN language into steps that could then be executed

[12] There were earlier instances of this sort of thing, and of course such breakthroughs rarely come all at once, but certainly FORTRAN was the one that had the most widespread impact.

by the machine. This was an early move in the direction of allowing programmers to write in a language more attuned to the problems they were dealing with and less tied to a particular machine's capabilities. Not only did this make writing programs easier and more natural, but because the program steps were now independent of the particular machine, it held out the promise that a FORTRAN program might be translated to work on any of several *different* machines, so long as each new machine had a compatible FORTRAN compiler. Thus the notion of machine-independence was born. Although it was only a first step in this direction, it quickly found favor with many programmers and some FORTRAN programs are still in use today. Debates about "higher-level languages" have filled the air in the years since, but today most programs are written in languages that run on many different machines.

Machines and computer languages, of course, actually evolved together—like bindweed. Many people today tend to think of "the computer" as performing their job, with only the vaguest notion of what lies inside. Behind the modern screen lie many layers of software, microcode,[13] and hardware, each dependent on all of the underlying layers, and the entire mess dependent on the years of work and understanding that led to this remarkable pyramid. In the early days the pyramid didn't exist and the user/programmer had to deal with the underlying machine more directly in its own terms without the helpful buffering

[13] Machine architecture has evolved in such a way that the underlying hardware performs only some very basic operations. The execution of the kind of instructions that used to be built into the hardware, is now performed by sequences of these basic "microcode" operations.

of the many layers that assist present day users and programmers.[14] These layers arose gradually, starting at the very bottom near the hardware, and eventually working their way up to the elaborate application programs that today's users have come to rely on. In many ways the development of these insulating software layers has been even more challenging than the development of new and improved hardware.

[14] Of course when trouble arises, this multiple layering can be bewildering and can (and does) lead to finger pointing.

Chapter 4

Enter Sputnik and ARPA, I'm nearly arrested, and a briefcase blows away. MITRE arises, I switch jobs again, and encounter various missile problems. A computer is murdered

In October of 1957, Sputnik rose into the sky and we all got up in the early morning to watch it rise—together, as it turned out, with our job security. It was no coincidence that shortly afterwards an organization within the defense department that became known as ARPA (the Advanced Research Projects Agency) was born. Within a few years ARPA, through its Information Processing Techniques Office (IPTO), was to become the dominant governmental institution sponsoring computer research around the country. I'll have more to say about ARPA below when I and numerous colleagues begin working indirectly under ARPA sponsorship.

Computer people have always tended to work at odd hours, often in extended, marathon-like spurts that conform poorly to the usual "business day." This behavior almost certainly had its origins in the days when the cost of machines led to demand for their full utilization, 24 hours a day. It may seem odd that such behavior continues long after such requirements have disappeared, but there are now other reasons. Part of it has to do with today's enormous competitiveness and the rush to get products to market, but I think there may be even more powerful

underlying explanations. Computer programmers deal with highly complex systems. "Getting your head around" such problems often requires near total immersion in order to keep track of everything, so programmers and designers tend to work in bursts that are tuned to the particular piece of work that they are engaged in. When it's finished, they go home and catch up on sleep and the rest of their lives. This is behavior that those not so profoundly engaged in a complex creative enterprise find hard to understand.

And so it was in the "early" days. Lincoln Lab was situated on Hanscom Air Force Base property, the computer was in use 24 hours a day, and I frequently had computer time at night. During this period I lived directly across the air base from the lab and often bicycled to work. Coming home I would take a shortcut that crossed one end of the runways. Late one night, as I was preparing to set off toward home across the runway, an apparition stepped out of the bushes. I noticed out of the corner of my eye that the apparition was wearing a uniform and was just a little shorter than the cannon that he was holding as, pedaling furiously to keep my bike-light alive, I swept past him. A second later I heard a rather dubious "Halt?" and suddenly a vision of the small figure struggling with the cannon and perhaps setting it off in my direction leapt to mind. This caused me to apply the brakes, whereupon, of course, my light promptly went out and everything went black. Slowly I pushed my bike back to where he stood, unsure what to do with me now that he had me. He stepped into a small booth, grabbed a phone from within, and called for reinforcements which arrived amidst sirens and flashing lights looking very spiffy in white hats and gloves. I myself was in shorts (it was a hot summer night), sported a shaggy beard, and appeared quite harmless if perhaps somewhat

disreputable. Fortunately I was able to produce my Lincoln Laboratory badge which, after carefully matching me to my photograph, seemed to satisfy them, although I'm sure they had no idea why I might be out bicycling at that time of night. Anyway, with an admonition not to get tangled up in a drogue chute behind some alighting jet fighter, they were off again in a torrent of sirens and flashing lights. I continued on my way after assuring the young guard that he'd served his country well that night.

I spoke earlier about the division that had taken place by this time between hardware and software specialties. An incident that took place at about this time will serve to illustrate the depth of the divide. Driving to work one morning, I came upon someone I recognized walking along Route 128. It was an odd place to be wandering on foot, and as he appeared distraught I pulled over and waited while he came up to the car. As he stuck his head in the window I asked the obvious question—what the devil was he doing there? "I lost my bag" was all he could say at first, but then gradually the full story emerged. As he had been getting into his car that morning to go to work he'd had his briefcase in one hand and a laundry bag in the other. He put the briefcase on the roof, opened the rear door, tossed in the laundry bag, jumped in the front seat and drove away. Only much later, as he was turning off of the highway toward the lab, did he suddenly remember the briefcase. He slammed on the brakes, got out, and stared at the now empty roof. When I arrived he was starting to retrace his entire route on foot in hopes of finding the missing briefcase.

Weeks before I had encountered him for the first time. He was a hardware designer and by that point I had become a passably skilled programmer. He wanted to learn

to program and I had taken it upon myself to teach him. He was an enthusiastic student, always ready to say "I've got it," so I gave him a small separable piece of the program I was working on to write as an exercise. Periodically he'd proudly bring me his work for review and each time I would point out what he'd done wrong. "I've got it now, for sure," he'd say eagerly each time. But after a month I began to realize that programming would never be his métier. About a week before I met him on the road, we'd had a session in which I'd explained that I really needed to get the job done and that if it didn't work this time, I was going to have to finish it myself. After that he'd worked his abilities to the bone, and this time, absolutely convinced he'd finally got it right, he was on his way to a meeting with me—when the briefcase containing the fruits of his labors blew away.

As he explained his misadventure, I felt a surge of relief. Although one could hardly fail to feel sympathy for a grown man with his head in one's car window on the verge of tears, I nonetheless realized immediately that his misadventure had averted the painful session I'd been anticipating in which I knew I would have to explain that his work needed to be done over. I succeeded in keeping these thoughts to myself and after bequeathing what sympathy I could, continued on my way to work while he set off once again on his futile quest. Later that day he told me that he had encountered some youths on the way and had hired them to walk the entire route (probably a dozen miles) from his home to where he'd stopped, promising a reward if they found the briefcase. They never did, and a few days later, after we'd worked up the bit of program together, he turned to me and announced that it was just as well they'd never found it. As I recall, he folded his tent at

that point and returned straightaway to the hardware fold, never to re-emerge.

At about that time (in 1957) the rumbles began about MITRE. The powers that be at MIT had concluded that building air defense and related military systems was likely to be an on-going business from which MIT should disengage. Besides, there had been growing frustration between MIT and the Air Force which had substantially different goals. It was therefore decided that Lincoln should shed those tasks specifically related to air defense, etc. and remain more of a general research institution. The MITRE corporation was duly formed, and it was decreed that the group in which I'd been working would move en masse to the new organization. But I (and one other chap) didn't *want* to become specialists in designing and building air defense systems, which seemed, at that point, to be MITRE's mission. I was interested in more general research, and aside from my personal interests, I was beginning to have some broader concerns about the rôle of the military in society. (These doubts were to sharpen and deepen over the course of my career and ultimately to shape the direction of my life for several years after I retired.) I remembered how the country had felt before World War II and didn't like the way it was beginning to feel now, maintaining a large, on-going military establishment with steady-state paranoia about the Soviets. Clearly MITRE was going to be even more directly tied to the military than Lincoln, and I simply didn't want to be a part of that. So I decided to stay within the MIT fold. I was importuned by my then boss, Charlie Zraket, who was leaving for MITRE and would eventually become its president. He was a wonderful person to work for and I regretted leaving him, but I resisted all blandishments and as my colleagues began to move to their

new quarters, I shifted my office and allegiance to another group within Lincoln.

In my early days at the lab I'd been unable to avoid noticing a fellow who seemed to have an extraordinary vocal range that he employed to great effect in support of what many would consider rather decided opinions. Not that he was inflexible or unpleasant—just definite. (He's been described as "the only person I knew who spoke in italics.") If a "discussion" arose that included him, people in offices for a considerable distance on either side of where the discussion was taking place would become aware of his views. By the time of the MITRE schism, he was in charge of a small group in another division at Lincoln and it was for him that I then went to work. His name was Frank Heart.

Frank's was a small but closely-knit group of about half a dozen people. Another member of the group was Will Crowther, a rock climber of legendary prowess whom I'd already encountered on weekends at the Shawangunk cliffs near Poughkeepsie, New York. (William Shockley, inventor of the transistor, was another Shawangunk climber and one of my favorite climbs was a route involving an overhang known as Shockley's Ceiling.) Will wrote immensely clever code, seemingly with both hands, and he taught me to play bridge, the de rigueur lunchtime activity in the new group. By this time the lab had acquired an honest-to-God commercial computer, an IBM 704 with which I was to become intimately acquainted over coming months. The first days in my new job were spent familiarizing myself with the programming manual of this new machine. As I warmed up on some minor, now-forgotten projects, I noticed that the name Noam Chomsky often appeared on reams of paper emerging from the printer. Who the devil

was this Chomsky fellow, eating up so much valuable computer time? And the chap's wife was soaking up time too. What could possibly be so important, I wondered[15].

Frank's group worked on a wide variety of projects, especially those with a real-time flavor. Lincoln had spearheaded such use of computers, but these kinds of applications, prevalent today, were relatively new and unknown at the time. New uses for computers were being investigated, but the expansion into new areas was a slow process that often required overcoming strong preconceptions and biases. Many who had problems that actually cried out for a computer solution simply weren't aware of the possibilities. People had accepted that computers could help with bookkeeping problems, but the burden of proof that they might be useful in some new arena often lay with the computer people themselves. This was Frank's specialty and over the years he pioneered many new applications in addition to the one for which he is best known, the ARPANET. Today computers are often oversold as the solution to every problem, but with the exception of the always-receptive military, at that time they were viewed with healthy skepticism by other prospective clients.

The process of exploring new application arenas was one that first required familiarizing oneself with some existing operation. This could be a delicate proposition

[15] Noam Chomsky, I was later to learn, was an MIT professor and one of the foremost linguistic theorists of the age. His work, some of it utilizing computers, profoundly influenced the field, and in more recent years he has achieved further renown as a progressive political and social philosopher. He has an international reputation and has published an almost uncountable number of books.

because not far beneath the surface was the implication that computer people, fresh off the street so to speak, might be able to understand and help solve some problem better than those who had been dealing with it for a long time. Under such circumstances it was all too easy to offend a potential "customer," especially if one's approach was at all hasty or arrogant. It behooved one to adopt an attitude of some humility, because, as often proved to be the case, the problem needing solution was usually more complex than it first appeared.

We frequently found ourselves investigating some totally unfamiliar branch of science or engineering, attempting to understand it well enough to be able to decide whether or not there was some piece of an operation that could be lubricated by the application of computer technology. If approached with some care, we usually found that people were enthusiastic about explaining their work, delighted that some crazy engineers took interest in their often obscure segment of the world.

Will and I began working together on a project that involved a missile-tracking radar. By now ICBMs were becoming a potential threat and the question had been raised: Would a missile reentering the atmosphere leave a wake that could be detected by radar in case the radar failed to notice the missile itself (for whatever good *that* might do)? Although I was beginning to have an aversion to military work, I knew this was an important question, so I put aside my embryonic "anti-war" concerns.

Not long before, just after dusk on a freezing cold night, I'd stood atop a hill outside of Boston where Lincoln's giant Millstone radar tracked missiles and satellites and performed numerous scientific experiments. My friend Howie Briscoe was, by then, working with the Millstone

group and he'd invited me out to see the radar in operation. That night it was to track a communications satellite being launched into orbit from Wallops Island down in Virginia. When the rocket was fired we were able to follow its trace on the radar screen, but almost immediately a shout came from outdoors and we rushed out into the crystalline night. There, some 500 miles away, one could clearly spot the rocket as it rose above the earth's shadow into sunlight, the wake from its engines clearly visible spreading out behind it in a giant orange plume as it ascended. Of course it was an altogether different question whether a ballistic missile, whose engines had long ago shut off, would leave a wake visible to radar as it rushed silently back into the earth's atmosphere. The project we were about to embark on would seek an answer to that question.

An elaborate experiment was being prepared at Wallops Island in Virginia. Multi-stage, solid-fuel rockets were to be boosted up above the earth's atmosphere. There the final stage would be turned around and fired back down into the atmosphere at high speed, simulating the reentry of an incoming missile. The earlier stages were to fall into the Atlantic in an area from which (I trust) ships were excluded, and we were assured that the final stage would completely burn up as it came back down. (Later, as we watched a launch, we were to wonder about this. Even though the missile went out over the ocean away from us, it appeared to be directly overhead and I had the eerie feeling that if anything went wrong, it would fall directly back on us. It seems that as you look up into the sky, everything more than about seventy degrees from the horizontal appears to be pretty much directly overhead if you have no other point of reference.)

A radar was to follow the missile on its upward journey and then track the final stage as it was fired back down into the atmosphere. A computer and memory were attached to the radar in such a way that the path the antenna had taken while tracking the missile rise and return would be memorized, thus enabling the radar to be repositioned along this path to discover if any residual wake could be detected. My job was to construct a plot of the trajectory the radar followed during the experiment. Wiggles appearing in my plots of early tests seemed to suggest a program bug of some sort, but closer investigation revealed that the antenna actually jigged and jogged as it moved about because the servomechanism electronics controlling the movement were improperly adjusted. I'd earned my salary even before the first shot was fired.

You need to know about one final refinement. The radar was situated on the mainland, several miles inland from the island where the missile firings actually took place. Thus before the missile took off, the radar was aimed horizontally along the earth's surface. This resulted in a great deal of noise ("ground clutter") in the received signal. It was impossible to distinguish the missile within this clutter and thus it could not be tracked as it took off. And if you couldn't track it from the outset, how would the radar ever locate it as it soared into the sky? A solution to this problem had been worked out. A set of movable bicycle handlebars (I kid you not) was mounted beside the giant radar with a telescope attached to them. The radar antenna could be slaved to these handlebars so that whenever they were moved, the antenna followed the movement. The idea was that an operator would look through the telescope and watch the missile taking off. He would then follow it up with the telescope by moving the handlebars to keep the

missile in view. And of course the radar antenna, slaved to the handlebars, would follow along. Shortly the missile would rise out of the ground clutter, a radar operator would then be able to identify the missile target on the radar screen, the radar would be locked onto this target and disconnected from the handlebars. Rube Goldberg had nothing on the U.S. Air Force. But, on the other hand, what could go wrong?

The launches took place at night, presumably making it easier for the fiery missile to be seen through the telescope. We had all seen television images of missiles being launched from Cape Canaveral (later renamed Cape Kennedy) and slowly lumbering into the sky, and somehow no one had bothered to explain that those were liquid-fuel missiles. By contrast, solid-fuel missiles, such as those to be used here, took off lickety-split, like a bullet fired from a gun. Thus on the first launch the telescope operator was totally unprepared for what happened. As the missile tore into the sky his head whipped back to watch it while his hands, the handlebars, and the entire guiding apparatus never budged. On the second launch the telescope operator was better prepared, but one of his colleagues, deciding to immortalize the event, snapped a photograph of the operator at the handlebars just as the missile took off. The camera's flash utterly blinded him and by the time he'd recovered his vision, the missile was somewhere out over the Atlantic.

I believe that the experiment was eventually made to work, although I don't recall whether or not a radar-visible wake was ever detected. I'm sure the answer is known to the missile-tracking community. Once everything was working, however, the programming fun was over, we programmers were no longer involved, and the running of

the program became just another part of routine activity. I have dwelt on this story, not only because the anecdotes remain memorable, but also because this project illustrates the importance of direct interaction between the programmers and the projects for which the programs were being written. Such interaction was (and in many cases, still is) an essential part of getting the job done properly. Real-time applications can rarely be circumscribed and fully defined in advance. All too often there are side effects and constraints that need to be discovered and allowed for, and there seems no good alternative to direct experience for getting things right.

Frank, always on the lookout for new applications, became interested in bigger rockets—of the Cape Canaveral variety. Soon several of us were on our way to Florida aboard our first jet airplane, a spanking new Boeing 707. Despite some concern about the unfamiliar spoilers on the leading edge of the wings before takeoff ("Look at those strange things. What do you suppose they are?" "Can you see, is there one on the other wing?" "I guess they'll remove them before we go." "Hey, we're rolling—could they have forgotten?"), miraculously we arrived in Florida intact, and the next day we were shown around a launch control center.

Our introduction to the site included viewing films of prior missile launches. We saw pictures of gleaming white Atlas missiles, sitting on the launch pad shedding ice in the approved manner as the engines roared into life. The missile would shudder a bit, rise a few feet into the air—and then suddenly explode in a tremendous ball of flame. Moments later, another missile would replace it and the same scene would be reenacted. One after another we watched our tax dollars going up in a lengthy series of

spectacular explosions. I was reminded of the films showing the evolution of the airplane through every kind of conceivable contraption, most of which met similarly disastrous ends. Such things are the price of experimental development and illustrate the story I am attempting to tell here. Most computer fiascos made somewhat less dramatic exits from the stage, but the same story of trial and error accompanies most engineering endeavors.

We were interested in understanding how computers were being used in the missile launching business and thought that perhaps we would be able to make some useful suggestions about functions that could be automated. In particular we suspected that a computer might prove useful in the long and complex checkout procedures leading up to the final countdown and launch. We knew that they would be using a computer for the more obvious applications, but suspected that they hadn't thought about some of the less apparent possibilities. Remember, this was the late 1950s and computers were still a Big Deal then, even at Cape Canaveral. There were no computers in washing machines or automobiles in those days, only in Big Operations. It turned out that there was a single IBM 709 (or perhaps it was a 7090) that, as we'd suspected, was primarily used when a missile was in the air, to keep track of where it was going and what it was doing. It provided humans with information, but didn't *do* anything much beyond that; it wasn't actually controlling anything, just monitoring and reporting.

We noticed that a large number of television cameras in the control center were all aimed directly at the launch pad. We asked why there were so many and were told the following story. All of the rockets carried explosive destruct packages so that in case the rocket went astray, it could be

blown up before it got to where it could cause real trouble when it came down. The destruct package was in the first stage (presumably because by the time the later stages fired and might misbehave the thing would be well out over the ocean where it could do comparatively little harm). As the rocket rose into the air, the computer continuously plotted an "intercept" point which was where the rocket would land should the engines quit. Drawn across the chart on which this intercept point was continuously plotted was a heavy black line. The rockets were aimed out over the Atlantic, and so the intercept point normally moved eastward, away from the coast, away from the heavy line. But if the intercept point ever strayed in the wrong direction, in particular, if it ever crossed the heavy line, the range safety officer was to push the button that exploded the destruct package.

One day things went very wrong indeed, in a way no one had anticipated. Those watching outdoors were treated to a spectacular show. When the missile was fired, somehow the second and third stages of the rocket took off together, leaving the first stage sitting simmering on the launch pad. Indoors, the computer plotted the intercept point of the errant later stages and sure enough, it began to move inland as the thing flailed drunkenly about. The range safety officer, with his eyes glued to the intercept point, saw it cross the forbidden line and so he pushed the button igniting the destruct package. Having been stunned to see the later stages take off solo, those watching outside now were further startled to see the innocent looking first stage blow up right on the launch pad, utterly destroying the pad in the process. Meanwhile the range safety officer watched in horror as, instead of stopping as expected, the intercept point continued to wobble inland. Ultimately the

upper stages fell, harmlessly, into the Banana River. We were told that the multiple video monitors were installed shortly thereafter (along, presumably, with destruct packages in *all* of the stages). Technological humility far too often arrives after the fact and computer hubris all too often comes a cropper on the unexpected. Many years later, lessons such as this were to lead me into the center of debates about the advisability of allowing computers too large a hand in deciding to retaliate against a presumed hostile missile attack. The various "Star Wars" proposals, which tend to rely heavily on complex computer decision-making, have seemed, and still seem to me, extremely questionable enterprises, fraught with the dangers of unanticipated circumstances.

Although we never actually did any work at Cape Canaveral, some time later, still on the rocket kick, we were able to provide help at Vandenberg Air Force Base in California. There test missiles were going in the opposite direction, out over the Pacific, and sending back telemetering information to special equipment that wrote the data onto a sequence of magnetic (computer) tapes in one long string with no gaps. No one had thought ahead of time about the problem of *reading* such tapes. Data on tapes normally came in well-demarcated, manageable-sized blocks, each of which fit comfortably into memory. Here, however, was a seemingly unmanageable quantity of information, all in one indigestible pile. Fortunately we managed to devise a scheme in which, as the data was being read into the memory, we simultaneously wrote it back out alternately onto *two* other tapes. While one of these was receiving data, the other could catch its breath and write an end-of-record mark, thus making it possible to break the data into manageable-sized pieces for later

processing. In addition we slyly repositioned the incoming data pointer occasionally so that although the data continued pouring in, it never actually overflowed memory and suffocated the machine. It was a clever counterpoint to the prior oversight, and we were particularly delighted because it used the IBM tape system in a way that no one had anticipated. It also kept the operators hopping, mounting and unmounting tapes at our direction. For once, we were in charge.

At about this time IBM introduced a new pair of computers—the 1620, which was a "scientific" machine, and the 1401, which was the "business" version. The distinction was based on presumed differences in need between these two communities of users. (Even though by then the term "general purpose digital computer" had become standard parlance, the underlying idea hadn't yet fully sunk in.) The 1620 was dubbed the CADET. Then some wit, noting that the machine had no ADD instruction, suggested that CADET stood for "Can't Add Doesn't Even Try," and the name was quietly dropped. Some time later word went around that a programmer, in a fit of what must have been exquisite pique, had *shot* an IBM 1401 computer full of holes. What more need one say? Such an urge is surely familiar to many.

It was now time to move on to the next project which, by contrast with trying to detect a missile attack, was concerned with detecting the presence of enemy submarines. If they didn't get us from above, they would get us from below. It seemed we were exploring submarine detection by sonic means using steered arrays of detectors. Involvement in this project would soon immerse us in studies of underwater sound transmission.

But first I have to tell you about the piano.

Chapter 5

A piano enters the lab and comes up against TX-2. DEC is formed and there is an error on Page 217. Fourier is proven sound and we land on an aircraft carrier.

I am a reasonably good pianist. I've been immersed in music all my life and felt deprived at Lincoln without a piano to practice on. I knew that Oliver Selfridge, an early computer whiz I'd met through Howie Briscoe and who was now at Lincoln, had a piano in his office, and I thought, why not me too? It was an unusual request, but Frank rather liked the oddity of it and I promised not to practice except during lunchtime or after work hours. I located a suitable junker and made arrangements to have it arrive as unobtrusively as possible, but when the phone rang an incredulous guard said "There's a man here claims he's got a *PIANO* for you???" I arrived at the loading dock to find two guards restraining the mover, who was intent on getting a peek inside the super-secret Lincoln Laboratory. We finally got him out and rolled the piano through the halls by some raised eyebrows to my office. Soon my lunchtime practicing became accepted background to the bridge game.

Now another use for my piano began to form in my mind. For many years I had watched my father struggling with music notation, first scribbling down a quick sketch of a piece of music before he forgot it, and then later going back over it, laboriously turning shorthand scribbles into

readable scores. Because it took so long to notate music, much wonderful material had been forgotten and lost. Although it was many years before word processing would become commonplace, I suspected that a computer might be able to help with the problem of notating music, just as text processors now help writers of prose. My initial thought was that the computer should be able, by analyzing the sound, to decide what notes were played, and then print out a score. After all, musicians could do it; why not a computer? Knowing what I now know, I realize how naive this was. I was neither the first nor the last to underestimate the sophistication and complexity of human cognitive processes.

From the time I first arrived at Lincoln, I was subliminally aware of a very special group within the lab, the Advanced Development Group. Many of the people in that group had worked on Whirlwind at MIT's Digital Computer Lab which had later been absorbed into Lincoln. These were the people who actually designed experimental computers and that group was, to my mind, the core of the matter, the pinnacle, the promised land to which anyone with a grain of ambition aspired. Lincoln had a seminar series in which lectures about the structure of the group's new experimental computer, TX-2, were appearing with increasing frequency. These lectures were presented by people who came to be my secret heroes: John Frankovich, Jim Forgie, Ken Olsen, Dick Best, and last but not least, TX-2's chief architect, none other than the self-same Wesley Clark, whom I'd heard speak in Pittsburgh in the fall of 1954.

I indicated earlier that beyond the division between hardware and software, equally important specializations developed *within* each of these disciplines. On the hardware

side there were already distinctions between the architects—those who designed the logic of the machine, viewing it principally from the programming point of view and breaking it into its logical components and logical steps—and the engineers, who were specialists in the world of physical realities and who dealt in electrical and mechanical constraints. The lectures by Best and Olsen were about the circuitry of the machine and I strained to understand the unfamiliar material. The lectures by Clark, Forgie, and Frankovich, on the other hand, were about the machine's logical structure and I understood that part of the story relatively easily. It seemed to be a magnificent edifice and I loved the excitement and feeling of esprit de corps that emanated from the members of the group. I later learned that Clark and Olsen had collaborated on the design of previous machines, including the MTC computer mentioned earlier.

I discovered that TX-2 had an analog to digital converter that enabled it to read analog signals—such as the sound waveforms produced by the piano—and then print them out. I was curious to see what these signals looked like, so I borrowed a tape recorder, played a few passages onto the tape, read it into TX-2 and printed out the waveforms.[16] I was stunned to find that the printout looked like absolute gibberish. I could see clearly where the first note had been struck, but beyond that everything looked like garbage. It was back to the drawing boards. I decided to try something simpler so I played just three notes in succession. When I looked at the new waveforms I realized

[16] By what subterfuge I managed this brief invasion of the sacred precincts I no longer recall. TX-2 was certainly not accessible to hoi polloi.

that my idea was in serious trouble; I couldn't even tell where the second note had been played, the whole thing was still just one big jumble.

Looking at the mess, I decided I'd better learn something about what was going on, so I went to the library and fished out a book on the physics of music. By the time I'd finished reading the chapter about pianos, which described the incredible complexity and variability of the actual acoustic waveforms, I realized that analyzing the sound with a computer in hopes of identifying the notes being played was far more of a task than I'd imagined and for all practical purposes, hopeless. This was my first head-on encounter with the fact that many things we humans do so naturally that we assume they must be easy, are in fact extremely difficult (if not impossible) to program a computer to do. Such tasks seem easy for us only because they are accomplished using sophisticated mechanisms buried so deeply within our central nervous systems that we are blissfully unaware of their operation and thus of how they work. Giving up on getting the computer to analyze the sound waveforms, I began to explore the under side of the keyboard of my ancient upright with the idea of installing a switch beneath each key that could record directly the notes as they were played. That was about 1958 and it was as far as I carried my thinking on the subject at that time. However, I continued mulling over the problem for many years until, thanks to numerous advances in computer technology, as well as my own understanding of the problem, it ultimately bore fruit in 1980 in the form of the first music score handling program, *Mockingbird*. More about that later on.

In late 1957 a subset of the TX-2 designers, under the leadership of Ken Olsen who had been the chief engineer,

left Lincoln to form a company that was going to make and sell logic modules that could be combined under relatively simple rules to construct all manner of digital devices—including computers. The modules were derived from the circuits that made up TX-2. These brash young men were, I believe, among the first to leave Lincoln with entrepreneurial ambitions. Ken was apparently convinced that computers should be mass-produced gadgets and was impatient with the cumbersome progress at Lincoln. They decided to call their company Digital Equipment Corporation—DEC for short. We wondered whether or not they'd make a go of it.

But back to underwater sound transmission. I've indicated that Frank was always on the lookout for new application areas. Today, when computers have become ubiquitous, one can pretty safely assume that workers in practically every arena of scientific endeavor are aware of their existence and most of the more obvious application areas have long since been thoroughly explored and exploited. And even if some potential new arena for use arises, those working in that arena will probably automatically consider utilizing a computer. But in those days it was a very different story and so we often found ourselves acting as missionaries carrying the true word to the benighted. This often meant that we suddenly had to learn a lot about some new and unfamiliar area of work in order to understand how computers might be utilized beneficially. (The arrogance of many computer people who leapt to conclusions in fields they inadequately understood sometimes retarded rather than speeded progress, but of course our group never behaved in such a fashion.)

In any case, we were certainly uninformed about the complex matter of underwater sound transmission, so

Frank went to the library and found a book on the subject which he gave me to "look over." When I opened it that night at home I knew I was in trouble. Like a pervasive mold, every page was covered with a layer of differential and integral equations of the most appalling inscrutability. It had been a long time since I'd dealt with anything like that. I decided there was only one thing to do. I chose one of the more frightening-looking pages and inserted a note to Frank that said "Not a bad book, but the author is really quite careless. Note, for example, the blatant error on page 217." I tucked the note into the book and next morning placed it on Frank's desk before he arrived. After some time he appeared in the doorway of my office with an ashen face. "How on earth did you find it?" he said, looking stunned. Now it was my turn to wonder—what could he possibly mean? "That mistake—how did you find it? Did you really read that entire book last night?" What the devil was he up to? Was he turning the tables and conning me? I decided to come clean—he was my boss after all. "Er, uh…mistake?" I asked sheepishly. "Yes, of course. This one, right here on page 217" he said, pointing to the book. And there, indeed, right smack in the middle of page 217, was a blooper (probably the only one in the entire blasted book!) so obvious that even I was able to find it within a few minutes.

We called it a draw and turned the whole matter over to the group mathematician/physicist who, miraculously, appeared to understand it all. Ships, it seems, have propellers that make a lot of noise as they churn through the water. The noise travels comfortably along for surprising distances. Each ship's noise consists of frequencies characteristic of that particular ship, or kind of ship. As the noise percolates along through the water,

certain frequencies become attenuated—the higher frequencies have a more difficult journey and tend to erode faster. Meanwhile, lurking on our shores are arrays of undersea detectors that act, for sound, somewhat like a radar receiver and can be electronically "steered" (never mind how) to point in whatever direction you wish. The problem we were to investigate was the sorting out of all the mixed signals that might arrive from the many ships driving around out there in the ocean. The question, of course, was whether one could spot an intruder among the normal shipping sounds. It was a bit like a blindfolded listener at a symphony concert trying to spot an errant clarinet in the string section.

We decided that a good way to start to explore the problem was with a simulation. We split the job into three parts. The first was a program that simulated the ships' noises. We could specify how much of each frequency a particular model of ship would generate and we could specify where various model ships were located in our model ocean. The second program utilized all those incomprehensible equations (well, some of them anyway) in that dreadful book. These defined how the ships' noises would look after they had traveled through the ocean getting all jumbled together to where our simulated detector array sat lurking. The third program pretended to be the detector array itself, puzzling out from this jumbled mess where it thought various ships might be. Once we had this set of programs in place and working together, we could vary all sorts of parameters and rules to see what the effects might be, what worked best, and so on.

Attached to the IBM 709 was a large oscilloscope screen about 18 inches in diameter. With sufficient programming effort, one could paint an image on this screen, but as there

was a big buffer memory that held the image and it took some time to move information into this buffer, there was no way to update pictures fast enough to give the kind of dynamism you see on today's screens. However, it was better than nothing and gave us some visual check on the waveforms our programs were producing. The image I remember best was one that we put up to verify that our wave synthesis programs were working properly. We all knew, as Fourier had shown at the beginning of the 19th century, that if you added a series of appropriately related sine waves together in the proper proportion, the combination would gradually approach a square wave as you added in more and more high-frequency components. That much we knew theoretically, from the mathematics. But now we were actually able to see it happen before our very eyes. As we added more and more components, we could watch the resulting waveform change shape, the corners gradually becoming sharper and squarer. Looking at the screen I felt a new level of conviction deep in my gut. I thought how Fourier would have loved to be able to peek over our shoulders.

Sometime during this period, a number of us were overwhelmed by the need to do some fieldwork in order to come to closer grips with the problem with which we were struggling. We were invited to spend a few days aboard an ASW (Anti-Submarine Warfare) aircraft carrier, traipsing around out in the middle of the Atlantic Ocean. We were to fly out and land on the carrier at sea, and eventually ride it into Norfolk, Virginia where it would be staying in port for a while. We were presumably going in order to learn more about submarine detection, and I suppose we did, but I knew a boondoggle when I saw one and this was a textbook example. When the time came, we piled into a small twin-

engine aircraft and headed out to sea. The carrier was several hundred miles off shore and when we arrived and looked down at it, it suddenly came to me that we were doomed. There was obviously NO way to land an airplane on an object that small, and yet as we descended I realized that the fool flying our plane was going to attempt it anyway. We were seated facing backward so that when we landed and the tail hook grabbed a cable bringing us to an abrupt stop, we would be squashed flat against, rather than thrust forward out of, our seats.

As we approached I braced myself for the end, but then at the very last moment, we were suddenly flung into the air again as the engines roared into renewed life. The seaman who was our guide managed to put on a worried look and say "Gosh, that's never happened before." It was some seconds before we realized he might be joking. By this time we were pretty thoroughly unsettled (unmanned might be more honest), and as we circled for another try, we totally forgot what to expect so that when we actually landed and were driven backwards into the seats as the plane's hook grabbed a cable, my only thought was, "Thank God, we've hit something soft." Within seconds, before any of us could regain our composure, the door was flung open and a smiling face was saying "Well, are you guys going to stay in there all day?"

From that point on things went immeasurably better. We had several wonderful days on board during which we learned many things, among them the fact that the guys flying the helicopters which were buzzing around dunking sonar detectors into the ocean listening for submarines, led an extremely dangerous life. The mortality rate was frightening. Giant scars on the main deck gave grim evidence of the hazards involved in the entire operation.

We watched as the twin-engined planes were hurled, one after another, into the air by steam-driven catapults, to then snoop with detectors from higher up.

One afternoon we were told that there were to be night operations. "Good," I thought, "we'll finally see this thing lit up from stem to stern like a Christmas tree." That evening, as we stood in the tower above the deck listening to the returning planes approaching, I waited for the lights to go on. As the sound of the approaching planes grew louder I wondered if something had gone wrong. Suddenly, out of the night, a plane appeared and, in total darkness, dropped onto the deck and screeched to a stop. "A miracle!" I thought. But then it happened again—and again and again. One after another the returning planes materialized out of the darkness and settled securely onto the deck. It turns out, of course, that you don't light up a carrier, which would advertise it as a target for enemy guns or bombs. Instead the pilots were watching narrow light beams that guided them in. Here was technology that really worked. It had to.

More fascinating experiences were in store for us before we finally drove (I'm told that's the proper verb for a navy craft of this size) into port. We thought we had learned our way around this marvel of topological complexity, a mechanical city housing some three thousand sailors. Among other things we had discovered the ship's store where denizens (including VIPs) could purchase items at ridiculously low prices. Those of us who were (then) cigarette smokers had packed into our luggage as many cartons of essentially free cigarettes (I think they were something like 9 cents a pack!) as we could cram in. But as we were approaching land I discovered yet another unfilled corner of my bag, and so leaving the others I rushed to the

store for yet another carton. Getting there took some time and involved a tortuous trip up and down numerous ladders and through endless narrow passageways. Eventually I arrived and made my purchase, whereupon I was overcome with a feeling that there must be a shortcut. Grabbing a nearby ladder I ascended ten feet—to find myself emerging immediately beside my companions!

Although the visit was memorable and fascinating from many viewpoints, I don't think we learned anything that really helped or influenced our study of underwater sound transmission. If we had, it would have detracted from the purity of the boondoggle.

Chapter 6

A moment of skepticism

"Often, especially in those particularly complicated parts of the physical world constituted by living organisms, the knowledge that is accessible is separated from what we really want to understand by very hard questions whose answers we do not know how to obtain. At one time or another in their lives most scientists realize the extent of this separation but they also perceive, probably correctly, that public faith and concrete investment in science depend on connection between knowledge and result that is much simpler. So, they are led to make extravagant claims for the operation of science, both for the objectivity and compelling power of a formulaic "scientific method," and for the direct applicability of elementary knowledge to problems of human welfare. When challenged, they throw up an obfuscating cloud of quite interesting and sometimes even quite useful results of scientific investigation, in the hope that no one will notice that the original problem has not been solved, or that it has, but by a pathway quite unrelated to what they have been doing."

Richard Lewontin—The N.Y. Review of Books, Mar 6, 1997 p. 52

During those years the burgeoning computer field was seeking and finding ever more arenas in which to

flourish. There often seemed no limit to the possibilities as new application areas opened up almost daily. Today's world is full of hucksters selling computers for every conceivable use, often in situations where a far simpler tool would work as well or better. But during these earlier years we appeared to many as Merlins, able to enlarge their capacities beyond their wildest dreams. The opportunity to shape and hand someone an unexpected, powerful, new tool is wonderfully gratifying, and with such experiences filling our lives, a general euphoria pervaded the field. I suspect that a somewhat similar thing may be happening today among scientists as they unravel genetic codes.

Such euphoria, however, can become a problem, so overstimulating the imagination that flights of fancy beget immoderate and unwarranted claims and predictions. Of course no one could be sure where the hard boundaries would eventually begin to establish themselves, and it is difficult to tease apart motivations and assess levels of optimism, but in retrospect it is clear that over the years many rosy images were portrayed of what computers might do for us—images that were far too optimistic and based on insufficient understanding. This is still happening today as the boundaries between fact and fiction, reality and fantasy, become increasingly blurred.

Lay people, always anxious for miracles, have had even less basis for judging matters than the cognoscenti, and the desire to appease their longings led to statements and views by some "experts" that created public expectations which simply could not be fulfilled. The hype that for many years, particularly in the 1980s, surrounded the branch of computer science known as Artificial Intelligence is typical. Some distortion and exaggeration can be laid at the doorstep of the media, but not all. In my opinion, such

indiscretions are folly—they grab headlines, but in the end damage the credibility of practitioners. Prudence would dictate a more moderate course than some have chosen to follow in their eagerness to satisfy an urge to notoriety.

The topic that has probably evoked the greatest number of false hopes over the years, perhaps because it seems the most easily understood by the lay person, is the search for a computer system that could reliably understand continuous speech, independent of its content. Humans apparently accomplish this task effortlessly, so it was initially thought that it must be easy. Forty years ago some were predicting that within a short while we would be dictating freely to machines. It was not necessarily anticipated that the machine would (at least initially) *comprehend* what was being said, but it was expected that it would at least be able to perform the seemingly straightforward secretarial task of turning the sounds of speech into printed text.

Gradually it became clear that even that deceptively simple task was far more complex than had been imagined. Tasks that seem easiest for us humans are the ones we accomplish through mechanisms that have become so built in, so instinctive, that we are largely or completely unconscious of the means by which they work. That makes them seem easy, but ironically, it also means that we don't really understand how we accomplish them. That, in turn, means that we don't know how to tell a computer to do it either. We may, of course, be able to find some way to accomplish the same end, without understanding how humans do it. But that's not what the over-optimists had in mind forty years ago.

It has turned out, as anyone might have predicted who thought about it deeply, that recognition and comprehension cannot easily be separated; for example,

although we tend to think we're speaking in words, the boundaries between the words are not at all apparent in the actual sound waveforms of continuous speech. It's not dissimilar to the trouble I encountered in trying to sort out the notes of my piano playing. Finding these word boundaries seems likely to be tied to the overall process of comprehension with lots of feedback going in both directions. The consequence of these and numerous other unanticipated complications is that, although great strides have been made, today computers are still unable to understand the sort of continuous speech that even a young child has no trouble comprehending.

This week I needed to check on the arrival time of a United Airlines flight and I used their automated system. It's quite clever and allows one to use speech as an alternative to pushing the dial buttons for numbers and choices. A naive user might conclude that the system understands speech quite reliably—until they noticed that the repertoire of responses is extremely limited and involve either numbers or highly prompted individual words. Although wonderfully useful, this word-at-a-time interaction is not the sort of discourse that is meant when we talk about understanding continuous speech. That said, however, it is important to acknowledge that systems now exist that, when carefully trained to recognize a particular individual's voice, do a reasonably good job of recognizing carefully spoken continuous speech. And we will undoubtedly do better and better over time. But it's taken a lot longer than the enthusiasts (many of whom extracted tax dollars for their work) promised.

A related subject, language translation, has suffered a similar history. Given the subtle differences between languages and the sorts of things they express (not to

mention the varying cultural attitudes underlying language differences), the problem of translating from one language to another is a difficult one even for humans fluent in both languages. Many things simply cannot be fully translated as each language inevitably adds its individual flavor to the underlying ideas being expressed. At one end of the spectrum lies poetry, probably the most difficult sort of writing to translate in that it typically capitalizes on the individual flavor of a language. On the other hand, translating straightforward scientific information written in standard, widely-accepted technicalese, is a somewhat less formidable task.

It is unfortunate that there has never been a clear public retraction, acknowledging that earlier predictions were dramatically overblown, and perhaps stating more realistic expectations. To quote Lewontin again, a "much-proclaimed program…wasted away, died, and was buried in a remote corner of the cemetery without a public funeral. The heirs simply took the cash from the estate and invested it in another enterprise." In this case, the new enterprise has too often been the same old lady simply clad in a new gown with differently arranged sequins. Periodically the news media get hold of the story of some new advance in speech understanding or language translation, and the hoopla is run by the innocent public yet once again.

Chapter 7

In which I join the TX-2 group and encounter a different culture and some memorable characters. I simulate another machine, avoid a fire, and start to dip into hardware

The leader of the Advanced Development Group, or TX-2 group as it later came to be known, was Bill Papian who, I knew, had done his graduate work at MIT under Jay Forrester, constructing the very first core memory. The chief designer of the group was Wes Clark. I recognized him as we passed occasionally in the halls, but even though I knew he was a friend of Frank's, it would never have occurred to me to introduce myself; I thought of him as the first real genius I'd come across in the computer field and held him in too great awe to approach him casually.

Then one day the news went round that someone in the TX-2 group had suddenly died. What kind of a vulture, I thought, would take advantage of such a tragedy. And yet, there it was, an opportunity held out by fate. I expected that there would be contention for such a plum position and I certainly didn't look forward to telling Frank that I was considering leaving his group. I had come to understand that Frank was someone to whom work relationships were very personal. He both gave and expected great loyalty and would not be pleased by what he would no doubt view as desertion on my part. On the other hand, he had often spoken admiringly of his friend Wes, and we had discussed

the exciting developments that were taking place in the TX-2 group. I felt that in many ways Frank himself would have liked to move into that milieu, but unlike me, he now headed a group of his own, and was thus no longer such a free agent. So as Frank sat in his chair, listening to my announcement, smoke could be seen pouring from his ears as the conflict within seethed. As he himself ultimately said, what could he say?

I suppose I must have been interviewed for the position; I don't recall. All I know is that a short while later I was moving again, finally invading the precincts of the promised land. I was excited and a bit intimidated. I knew that my life was undergoing a major shift and braced myself for the challenging new world I was entering. I was delighted to find that the members of the group, my mythic heroes, were extremely friendly and helpful people. Even Wes, when you could get to him, was attentive and encouraging.

Before I go on with the story, I need to back up and give a bit of TX-2's background. Prior to TX-2 there had been a TX-1 and, following that (oddly enough) a TX-0. TX-1 was the first of the series. It was to have been a vacuum-tube machine designed as a test-bed for the first large core-memory array. But the design was never approved and the machine was never built. Instead Clark and Olsen proposed another machine, TX-0, for the purpose, to be built using transistor circuits designed by Olsen and other members of the group. That machine was built and continued to be used for many years. At the same time Clark began designing a much larger, more powerful transistorized machine that would incorporate many novel features (not to say bells and whistles). It was this machine, TX-2, whose design I had enviously watched evolve from a distance. Its world was

very different from the ones I'd inhabited up until then and, in many ways, its method of utilization presaged what was to take place throughout the world many years later.

To begin with, you might think that as TX-2 was one of the very first computers built from transistors rather than vacuum tubes, it would therefore have been of relatively modest physical size. But computer people are big spenders; give them an inch and they take the proverbial mile. So of course the same space (well, actually a bit less) had been filled with twice as much smaller circuitry. This made it a very dense and powerful machine. It was also an experimental machine and that meant that often people had their hands and tools inside of it, in the racks which held the modules of which it was built, forever changing and improving it. I'd seen this before, with the IBM machines, but in that setting, repairmen had invaded the machine's innards only to heal something that had gone wrong or to do routine maintenance in the middle of the night. Here, people seemed to be constantly fiddling with, modifying, and improving things.

TX-2's racks were spread out so that some parts of the machine were a considerable distance from the console where the main power switch was located. In order to warn anyone whose hands were in the wiring to stand back whenever power was turned on, a deafening air-horn was sounded several seconds in advance. The power-up sequence actually involved many steps; the air-horn being merely the first and most apparent. Once power was up, further internal set-up was initiated by a button labeled CODABO. Here, at last, was an acronym one could love; it stood for COunt Down And Blast Off. Which brings me to the next, very important matter, the way TX-2 was used.

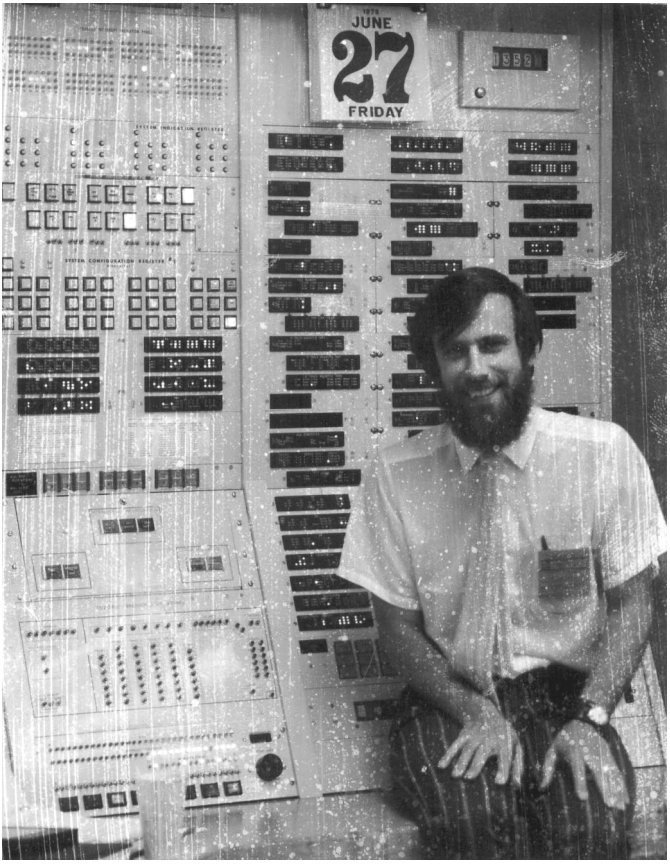
In those days, TX-2 was run totally differently from any machine I'd encountered before. The difference was much like the difference between taking public transportation and driving one's own car. First, it was operated directly by the programmer rather than by an intermediary operator. Second, you didn't just get a momentary shot at the machine and then carry away a printout to be pored over later, somewhere else. TX-2 users simply debugged their programs right at the console, sitting there sometimes for hours at a stretch. (The longer runs were usually at night.) This appeared to be a waste of valuable computer time, but it meant that programs could be debugged in a fraction of the calendar time that it would otherwise have taken.

This way of using a computer evidenced a profoundly different philosophy: It emphasized optimizing the time of the human beings, rather than the time of the machine; it also looked forward to the day when a few seconds of unused computer time would no longer be so costly. Today when I turn off the computer on which I'm writing and go to bed for the night, I waste more computer power than earlier machines could provide in several months, running full bore. Wes Clark, who as its principal architect, was king of TX-2, foresaw this state of affairs and deliberately arranged the use of TX-2 in this then-unorthodox manner. I recall a seminar he gave at about this time in which he looked forward to the day when computer power would become virtually free. I remember the lecture vividly because he said that one should think about a computer as something that would one day perhaps just be painted onto any handy surface. I can still see the gesture he used, painting back and forth on the top of the desk from which he was lecturing. "Surely," I thought, "that's going a bit far."

Another thing about TX-2: It had a screen (about 10 inches square) on which you could both paint and change images in a big hurry. For use with this display, Clark had invented a light pen, a device somewhat like Whirlwind's and XD-1's light guns, which allowed you to select items painted on the screen. Using these features, a young graduate student from MIT named Ivan Sutherland was constructing a system that displayed drawings with which users could interact, stretching, bending, and adjusting them in real-time. This first serious demonstration of interactive graphics would become famous as "Sketchpad" and would eventually move Ivan into the ranks of computer immortals. But the man behind such developments, the one who had had the vision to foresee the need and the possibilities of interactive computing, was the architect of TX-2 itself, Wes Clark. Another young graduate student named Larry Roberts was using the display to explore ways of compressing the information contained in pictures to facilitate their transmission over phone lines. Larry would later become famous as one of a number of so-called "fathers" of the Internet. (The Internet has a surprisingly numerous paternity. More about this later on.)

Back to the horn. Although its sound seemed excessive, it was, in fact, a device of great importance. Earlier I'd encountered a fellow who had nearly been killed at Whirlwind years before when someone turned on the power unexpectedly while he was working with his hands in the machine. There were some high voltages in the racks and the shock he suffered had damaged him for life. The TX-2 horn was designed to prevent such accidents, but it also had more humorous effects. If a particularly smug new programmer appeared, somehow mention of the horn

would be overlooked in his training, and when he went into the computer room alone at night for his first run and turned on the power, we knew he would be suitably chastened.



The author at the Console of TX-2

I mentioned before that somehow certain people stood out from the crowd. One of these was a colorful fellow by

the name of Tom Stockebrand who worked in the TX-2 group. Stocky was the chap who had been demonstrating MTC's ability to play music years before. At about this time I heard the story of his encounter with IBM. It bears repeating not only because it reveals Stocky's delightfully whimsical character, but because it emphasizes the enormous divide between the academic and the commercial worlds.

It seems that Stocky had been hired by Lincoln to work on the tape drives that were part of the XD-1. These were the first of the big vertical units with the dual vacuum columns that allowed a small section of the tape over the read/write heads to be moved quickly without having to jerk the big tape reels themselves into motion so abruptly. Later, for many years, these tape drives became such an icon for a computer that whenever there was a television news story involving a computer, one of these units would be flashed on the screen as though it were "The Computer."

Stocky had been sent down from Lincoln to the IBM plant near Poughkeepsie where these drives were being manufactured. He was to work for a while on the production line and thereby learn about the drives from the bottom up. In those days, IBM was extremely straight-laced. You may think this is still true today, but things have relaxed substantially since those times. Tom Watson was a no-nonsense leader who wanted his employees if not to salute, at least to stand up straight, wear white shirts with coats and ties, sing the company song at the company picnic, etc. Into this starchy company strode Stocky who wore dirty fatigues, was often shoeless, and was his own man, not T.J. Watson's.

Stocky went to work on the production line in Poughkeepsie, and soon eyebrows began to raise. Finally

one of his co-workers, unable to stand it, approached and said, "Tom, how is it that some days you come in here looking like a bum and smelling like Limburger, whereas other days you come in looking quite spiffy?" Stocky, in his usual forthright manner, answered that some days he woke up feeling bad; it would be raining and miserable and he'd just grab whatever clothes came to hand and come to work. On other days the sun would be shining, the birds would be singing, he'd feel like a million dollars, and so he would get all spruced up. The questioner's jaw dropped; clearly such whimsy was beyond comprehension, probably illegal, certainly immoral, and absolutely not in the IBM book. Meanwhile Stocky, warming to his subject, said "You know, some days I get up feeling so good, I put on my Tux." As the man's eyes bulged, Stocky realized what he'd committed himself to.

A few days later he strode in clad in tails and top hat. All morning he worked on the line. Not a word was said. At noon, down the aisle came his boss, his boss' boss, and the plant manager. They invited him into the front office, and (here's the surprise) instead of giving him a lecture, they offered him a job—at double his salary, whatever it might be. So white shirts were OK for the plebs, but at least some at IBM apparently understood how important spunk and originality were. Stocky, of course, turned them down.

Now he was working in the TX-2 group on the design of a man-killer tape unit so enormous that it was hoped one would never again have to change tape reels. Of course one would have to be able to search such a large space at great speed and to permit this, motors of substantial horsepower were involved. In the course of checkout, occasionally a tape would fail to stop, and when the end of the tape came off the take-up reel at Mach 2, the sound that it made was

deafening; it also turned the computer room into a lively place where one would sometimes find oneself wading around in snippets of tape.

Up to this point I had been doing only programming, but I now began to drift gradually into hardware design. My initial assignment in the new group was to write a simulation program on TX-2, this time mimicking something altogether different from ocean sounds—namely another computer, the FX-1. For some time the search had been on for new memory technology that would allow faster computers to be built. Magnetic cores were reliable and had the considerable advantage that they retained their information even when power was switched off. But it took a while (we're talking microseconds) to switch their magnetic field from one direction to the other and that limited how fast you could cycle the memory, which in turn limited how fast you could run the overall machine. A promising new technique, using thin magnetic films, had been developed at Lincoln and a computer was to be built that would incorporate a thin-film memory in order to try it out. The machine was known as FX-1 and would take some time to build. Meanwhile, in order to allow programs to be written, debugged, and ready to go when the machine started working, I was to construct a simulated version of FX-1 within TX-2 on which these programs could be checked out. The simulation would also help to verify the overall logic design of FX-1.

A simulation of this sort does much the same thing an actor does when he takes on the rôle of a character in a play or a movie. The actor, to the best of his ability, takes on the persona of the character he is representing. In this case, I was to write a program for TX-2 that would make it behave like some other machine, namely FX-1. I was only supposed

to simulate the actions of the machine (its instructions) not the underlying circuits. The actor's abilities must encompass all of the abilities of his character; for example if the character is to speak French, then the actor must be able to speak French convincingly as well. (In movies it is sometimes possible to fake certain abilities, such as, for example, facility in playing the piano, but the audience is supposed to believe that the actor is performing the action.) Similarly, the machine on which the simulation was to run (TX-2) had to be able to perform all of the steps that the simulated machine (FX-1) could perform. To implement some FX-1 steps would require several, sometimes many, TX-2 instructions, but that was OK because it wasn't necessary that the simulated FX-1 run especially fast. Finally, in order for an actor to portray his character convincingly, he must understand the character thoroughly in order to understand how to represent him under all circumstances. In order to do this job, I needed to understand in some detail how the logic of FX-1 worked.

I had never looked at a machine so closely before. Two Johns, John Frankovich and John Laynor, were my mentors. They explained to me how FX-1 was meant to work. We sat for several days in front of a blackboard and gradually I pieced together an image of the machine. I then spent some time designing the program and figuring out how to fit my simulated FX-1 inside the structure of TX-2. Then I wrote the program. Fortunately I'd been prepared for my first night's run; I knew about the horn and had heard it. But I was somewhat taken aback when I overheard Frankovich asking someone, "Is that fire-extinguisher still in there by the printer?"

Ah yes, TX-2's printer. It was one of the very first-ever xerographic printers. It was a monstrous machine, cobbled

together from pieces of a couple of very early Xerox copiers. It painted images on a drum by shining a CRT beam through a mask that defined a specific set of characters.[17] It was fully programmable, too fully perhaps. The paper it used came on a large continuous roll six or seven inches wide, and it was up to the user, employing a set of giant shears hanging nearby, to chop it into suitable pages (programs marked the page boundaries so you would know where to cut) as it emerged from the printer. John's inquiry about the fire-extinguisher stemmed from the fact that a program bug could cause the paper to travel too slowly (or even, God forbid, stall) in the printer's heater station with potentially catastrophic consequences. The paper should be warm as it emerged, but not brown and hot—and certainly not flaming. As this was the first TX-2 program I'd written, and as it was going to drive the printer, John had reason to be concerned. He showed me where the fire extinguisher was.

But in the event, things actually went swimmingly. I managed to avoid setting the place on fire, and one night not too much later, the program began working. I'd checked out my simulated console and screen (which used a subset of TX-2's switches and a piece of the TX-2 screen as FX-1's switches and screen) and all of the FX-1 instructions, and everything appeared to work properly. Even though it was eleven o'clock at night, I called Wes at home to announce

[17] After it was built, someone invited Xerox representatives to see it and pointed out that it would make an excellent product. They said "Very interesting!" then departed and were never heard from again. In that same era, Xerox was being lauded in the financial community as one of the best managed companies in the world.

victory. To my astonishment, instead of the "Good show" or whatever I'd expected, he said, "I'll be right in. What kind of coffee do you drink?" He arrived shortly with a giant thermos of fresh coffee and as I watched, wide-eyed, he set about driving my FX-1 through the night—writing and running programs, flipping switches and entering instructions at blinding speed, as though he'd been programming it for years. Before I knew it, patterns were flashing on the screen—*my* screen. I was thrilled. In all the time I'd been building it, writing the simulation program and verifying the individual FX-1 operations, I hadn't stopped to consider actually writing programs for it. So this was the first time it had really been used and, not surprisingly, I was delighted to see it respond. I was also thrilled to have a boss who behaved as this one did. We must have been there until three in the morning, playing with our new toy. It was a memorable night for me and formed the first layer of cement in a friendship that has continued to this day.

My next job was to help connect an IBM tape drive to TX-2. A stodgy old IBM tape unit was anathema in that free-wheeling environment, but it needed to be done; someone important wanted to transfer data between TX-2 and the IBM machine out back—perhaps that guy Chomsky again for all I knew. We fell to with a will. Chi Sun Lin, a delightful colleague, would design and build the hardware interface while I was to do the programming. In a design such as this, the trick is to build a minimal amount of hardware, just enough so that when manipulated by the software, everything that needs to happen can be made to happen—and fast enough. This is not because hardware designers are inherently lazy; it's because that way there's

less to build and to break and because software is far easier than hardware to rearrange in order to get things right.

Lin was very clever at pushing as much of the design as possible into the software, and for the first time I found myself writing a program that reached into and manipulated special hardware at a very low level. It was up to the program to run the tape forward and backward, time out the legislated gaps, and write or read the data to or from the tape at the appropriate rates. My prior experience with reading and writing information on tape was with commands that transferred blocks of data one direction or the other. The program I was now writing performed some of the micro-operations that were involved in such block transfers. I watched Lin operate an oscilloscope, probing the hardware as the program ran and pointing out when, and in what way, the program was doing the right or the wrong things. We finally got everything working so that we were able to write and then read back information, but we also had to be sure that the tapes we were making were in a format compatible with the IBM 709 computer. As a preliminary verification we wrote a record on the tape and then dipped the section we'd written into a volatile solution containing extremely fine iron filings. In solution, the filings were free to move about on the surface of the tape and they hauled themselves over to the areas that had been magnetized when we wrote the tape. Then the liquid evaporated, leaving the filings sitting on the tape. Peering through a magnifying glass we could actually see what we had written—honest to God visible bits—and in the right place as well. Suddenly I felt I had one toe in the hardware world.

Chapter 8

The Big Dealers vs. the Little Dealers. We poise for a leap

Not long after our IBM-compatible tape unit began working, Wes asked me to poke around Lincoln, looking for specialized hardware that people had built for various projects—projects that might have utilized, and benefited from the flexibility of, a small computer, had a cheap enough one been available. Given that I had only the vaguest idea of what he had in mind, it's perhaps not surprising that I found only one or two potential applications. Nonetheless it soon became evident that a new computer was in the works. Not just any old new computer, but an altogether different sort—one that would make a statement about what computers of the future should be like, how they should feel and be viewed and used. The statement was heretical; it crossed swords with prevailing opinion within the computer research community in what amounted to a religious war. Like the cold war, this one never developed into open hostilities, but it nonetheless manifested two profoundly opposed philosophies, and lasted for many years. It's important to note that this debate took place within the comparatively narrow confines of the university and government-funded research community. The wider computer community, represented by industry, persisted in believing that batch-processing was the only sensible way to do business and that both factions within

what might be called the alternative research community were pursuing unrealistic approaches.

Almost everyone within this alternative research community agreed that the batch-processing form of sharing was impossibly cumbersome and that some improved form of interactive use must be found. But beyond that basic agreement, opinion diverged markedly. On one side were the "Big Dealers," those who believed that for the foreseeable future useful machines would continue to be extremely expensive to build and maintain, and that one therefore needed to find ways of sharing them more efficiently. The Big Dealers observed that when a person interacted with a computer, the computer often spent a large fraction of its time waiting for the person to do the next thing—strike the next key, whatever. Aha! they said, that means that the computer should be able to serve a number of people at (approximately) the same time. While one user is scratching his head or raising his finger to strike a key, the computer should be able to serve the needs of others.

The Big Dealers' solution was therefore to divide up the machine's cycles in such a way that many users, sitting at individual terminals remotely connected to it, could use it at essentially the same time. Of course the users weren't actually using the machine truly simultaneously, because these computers could really do only one thing at a time. However, the idea was that it could switch its attention between users so rapidly that each user would have the *illusion* of having the entire machine to himself. This approach came to be called "Time-Sharing."

Before proceeding I need to distinguish carefully between two very different kinds of sharing. Indeed machines often serve multiple purposes "simultaneously."

In fact the machine on which I am presently typing is a prime example—it is shared by numerous programs which perform a variety of different functions. Although only one is operating at a time, several are in the memory and ready to go if I ask for them. But there is a difference between such *cooperative* sharing and *competitive* sharing that occurs in a Time Sharing system. Since I am the only person using this machine, for the most part I am not competing with anyone or anything.[18] However, in a Time Sharing system multiple users are competing for computing resources and under those circumstances, what one person does can influence the access of another. I will reserve the capitalized term Time Sharing, for this specific kind of use between individuals at terminals accessing a central machine and competing for its computing horsepower.

A crucial question was whether a way could be found to avoid making users at the terminals wait for the computer's attention. Of course there was no way to anticipate when each user would require servicing, so time was sliced into very tiny segments and complex mechanisms were devised to dole out access to the needy. It was vital to keep down the overhead (in machine cycles) of managing the sharing, in order to give the users service that was sufficiently fast to maintain the illusion that each had the sole attention of the entire machine. Jobs that required lots of computing could be handled in the cracks, between the more urgent business of servicing users sitting at

[18] Occasionally I ask the machine to perform a task that requires the full attention of the machine in which case it's taken out of my hands while that task is performed. But that is under my control—and I can even change my mind and cancel the operation if I become impatient.

terminals. Such jobs would thus take many times longer to complete in this piecemeal fashion, but who cared? Occasionally a user at a terminal might do something that triggered the need for extensive computing, but most users, it was presumed, needed very little of the computer's attention most of the time.

The possibility of some such a form of shared use had been discussed from the mid-fifties onward. I remembered such discussions almost from the time I first arrived at Lincoln. But John McCarthy, at MIT, was the first to document the concept of general purpose Time Sharing in a memo he wrote on January 1, 1959. That note bounced around MIT and resulted in two demonstration projects being undertaken, one at MIT under Prof. Fernando Corbato and another at BBN under McCarthy and Ed Fredkin on BBN's PDP-1 (the original copy of DEC's first full-scale computer based heavily on the designers' experience with MTC, TX-0, and TX-2). Both systems became operational in the summer of 1962. Not long thereafter almost the entire computer research community commenced a headlong rush down the Time Sharing path, and for many years thereafter Time Sharing, and the search for better ways to implement it, dominated research in computer system architecture within the ARPA community. Sizeable amounts of money, manpower, and ingenuity were expended on this approach over the course of many years.

In these systems, the older punched-card, batch-processing gave way to the use of teletype-style terminals connecting individual users to the central machine. Using these devices, communication between user and computer took place via lines of text typed back and forth. This form of interaction required the user to learn the arcane language required to communicate with the computer, but since at

that stage it was still a limited cognoscenti who were using machines anyway, the impediment of language was accepted by most users. Although looking back, these devices and this sort of interaction seem unbelievably cumbersome, they nonetheless provided a primitive form of "interactive" use. Access to the machine was much more direct than anything that had previously been experienced except by the handful of people who had used the MIT machines, Whirlwind, TX-2, etc. Eventually somewhat faster, sleeker versions of terminals appeared, including the so-called "glass teletype" in which the text appeared on a screen rather than on a roll of paper. But virtually all shared the same fundamental serial text style of communication. And in fact that was not too badly matched to the limited computing power available to the individual users of a Time Shared system.

A very different view was held by a tiny community of "Small Dealers" led by Wes Clark, who felt that real-time, interactive use via a display screen was crucial and that Time-Sharing would never be able to provide such capability. Of course fast displays had existed on dedicated machines (Whirlwind, MTC, TX-0, TX-2, etc.) since early days, and these displays had been an integral part of the computer itself. The terminals for Time Sharing systems lay distant from the computer and were connected to it by low-speed telephone lines that could handle the rates of typewriter-like devices but not the far higher rates required to service display screens. Nor could early Time Sharing systems themselves provide the kind of prompt, high-speed service required by multiple users working with display

screens.[19] In fact, the delays and uncertainties of timing in Time Sharing systems mitigated against real-time use of any sort except at extremely slow rates. There were some attempts to circumvent these limitations, but essentially time-sharing was anathema to real-time computing at anything other than very slow rates.

Instead, Clark argued that a computer shouldn't have to be such an awesome affair, that it should be possible to build a computer that could be used by a single individual, moved from place to place as required, and turned off at night with a clear conscience, not so differently from other pieces of laboratory equipment. Such a machine should be able to provide interactive service via a display screen to an individual user. The vision was thus of a truly personal computer, not merely the illusion of one as promised by Time Sharing. Such a vision seemed so implausible at the time, and was so contrary to received wisdom, that the only way to make any headway in promoting it would be to put together a demonstration prototype, and that was precisely what Clark was quietly preparing to do. In sending me around the lab, he was discretely looking for potential clients for such a machine.

In the days before integrated microcircuits, trying to build a "personal computer" was a daunting enterprise and required faith that ultimately the size and cost of hardware would shrink dramatically. It was clear that for the time being such a computer would be far less powerful than the existing big machines and would be too expensive for an

[19] Today a substantial fraction of the "horsepower" of personal computers is dedicated to making them more accessible to the user through the graphical user interface that everyone now takes for granted.

individual to purchase. The trick would therefore be to build a reasonable approximation, keeping the cost as low as possible while still demonstrating all of the important features (albeit in primitive form) that might someday constitute such a device. The challenge was to show that many problems could be handled by a small computer if it embodied just the right array of features. Over the ensuing few years, a comparatively modest band of workers, led by Clark, would devote themselves to constructing such a computer to be used initially by workers in biomedical research applications whose needs provided much of the initial impetus for the design. I was fortunate to be a member of that troupe.

The war between these two opposing views is one that today is largely forgotten, having been rendered obsolete by the later development of microcircuits that ultimately allowed the vision of the Small Dealers to flourish. The Time Sharing systems of those years, like other dinosaurs, are now a thing of the past, but for many years they dominated the research computing scene. Clark, as a member of a 1961 MIT Long Range Study Committee, differed with virtually the entire rest of that committee, which enthusiastically embraced Time Sharing as the solution to the Institute's computing problems. Time Sharing was the bandwagon and the Small Dealers were decidedly beyond the pale. A small number of us, however, were persuaded by the force of Clark's conviction, and some, in particular Charlie Molnar whom we'll meet shortly, felt that they were able to see a possible way through the maze of technical obstacles.

Although in the long run the Small Dealers' image has come to dominate in that the vast majority of people sitting in front of machines today are utilizing individual personal

computers, nonetheless virtually all machines (from personal computers to large servers of various types) are today "time-shared" in that multiple things are going on at the "same time." During the years of Time Sharing dominance, a large amount of software development took place and a generation of proficient programmers grew up, honed their skills, and developed important understanding using Time Shared machines. These systems allowed programmers to develop many of the machine-utilization tools that everyone depends on and takes for granted today. Two important examples are the *multiprocessing*, which allows you to work "at the same time" with a number of different programs (word-processing, file-management, email, spread-sheets, etc.), and *virtual memory* which permits programs to expand beyond the limits of the (expensive) central memory by sloshing in and out of the much larger and cheaper hard disk memory. In addition, Time Sharing enabled exploration of higher-level languages and other software development tools, as well as development of many of the instincts and insights that characterize personal interaction with a machine.

Most of these things would not have been feasible on the early small machines as they required the power and capacity of the larger machines. Of course even on the larger machines, the speed of the interaction was limited by the fact that they were far slower than today's computers, especially given that their attention was spread among so many users. The issue of speed is far more important than it may seem because interactive usage simply becomes unworkable if the machine's response is too slow. If the natural rhythms of a human being are too heavily compromised by a tool, the net effect is destructive rather

than constructive. Try running at one step every second or watching a movie run at one-tenth speed.

But there are many reasons why personal computers have replaced Time Shared use. The price is right and that means not only that individuals can afford to purchase them, but that within corporations bureaucratic involvement is minimal. A second important reason is territoriality—it's *mine,* and I don't have to compete with anyone else for its use. As compared with Time Sharing systems, it provides uniform response time that doesn't vary depending on what others are doing. And although we're all victims of indifferent software, at least with a personal machine we aren't subject in addition to the whims of system wizards and administrators.

As noted earlier, most advances in computer development have consisted simply in taking the next step forward. Often these steps follow more or less obviously from what has gone before. Occasionally, however, someone takes a more dramatic leap based on insights that, in retrospect, appear almost prescient. We were about to experience such a leap of faith and hope. In this instance it involved relatively few elements that in themselves were dramatically new. Rather, it consisted of bringing together, into a single machine, features that in combination constituted a new kind of entity, the forerunner of what would one day turn into the personal computer that has today become so ubiquitous. In fact the machine that arose is generally recognized as the world's first personal computer.

This was no accident. Clark had been one of the earliest, and continued to be the most ardent, advocate of personal computers throughout the era during which batch processing and Time Sharing dominated almost

everywhere, including practically the entire MIT computing community. What was about to happen involved putting one's money where one's mouth was—nothing less than an effort to manifest an improbable vision in concrete terms—and in the face of strongly opposed mainstream opinion.

Chapter 9

The birth of the LINC. I become a midwife and leave Lincoln

In the summer of 1957 a young man by the name of Charlie Molnar came to work in the TX-2 group. In the fall he commenced graduate studies under Prof. Walter Rosenblith[20] at the Communications Biophysics Lab at MIT and continued using TX-2 for his thesis work. There had already been interaction between some members of the TX-2 group (notably Clark, Farley, and Papian) and people doing neurophysiological research down at the main campus of MIT. Wes' interest in neurophysiology had been evident in the lecture he'd given several years before in Pittsburgh, and one day, overhearing a conversation between him and Jack Rafael (developer of the thin-film memory), I was delighted to discover that, like me, Wes was hoping to find a way to do something with computers in a totally different, non-military arena.

Neurophysiological research up to that time had been bedeviled by the lack of suitable laboratory equipment. The "dry" sciences—especially physics following the atom bomb—did considerably better in acquiring necessary research tools (big atom-smashers, for example). But the "wet" sciences got the dregs. The technological and

[20] Rosenblith later became Provost of MIT and, subsequently, Foreign Secretary of the National Academy of Science.

logistical barriers that separated experimental work from the processing and analysis of data limited the progress of research. There was no way that data could be processed as an experiment proceeded in order to influence the course of the experiment. Instead researchers had to wait for the results, which, in the case of experiments with animals, invariably meant starting over with a different animal with no way to know whether the electrodes were in exactly the same place as before, etc.

In order to facilitate the processing of neurophysiological data, special equipment was designed and built for particular purposes—such as the averaging of response signals, for example, in order to increase the signal-to-noise ratio.[21] But such special equipment was, by definition, limited to a particular kind of data processing. What was needed instead was a much more general purpose machine, one that could be adapted through suitable programming to diverse kinds of data processing. Such a machine should be able to take in signals directly in analog form, convert them to digital form, subject them to whatever kinds of processing the researcher wanted under control of parameters that the researcher might wish to vary, and finally display results—all in real time. Batch-processing and Time Sharing were anathema to such use. The researcher needed sole access to the machine for extended periods. In addition, the machine needed to be

[21] Clark had designed a machine called the Average Response Computer (ARC) for this purpose. Individual responses were so noisy that they could not be seen. But by summing a sequence of responses, the noise, being random, tended to be self canceling, whereas the actual response signals, always occurring at the same time, added together and thus became visible above the noise.

small enough to live comfortably within the confines of a laboratory environment. Ideally such a machine should be viewed as just another piece of laboratory equipment that could be at the user's elbow during the course of an experiment, to be turned on and off as needed. No such machine existed and the size and cost of most computers rendered such thoughts essentially fantasy. Nonetheless, the desire for such processing provided strong motivation to a computer designer such as Clark who had contact with the needs of neurophysiological researchers.

As Wes ruminated about these matters, Charlie's studies were interrupted by family illness, causing his student deferment to be canceled. (We were between wars, but the draft was still in force.) He was called to active duty, but fortunately he was assigned to the Air Force's nearby Hanscom field and, although his thesis work was put on hold, he began discussions with Clark about the possibility of designing a computer specifically around the needs of neurophysiological researchers. Thus commenced a friendship and alliance between the two that would grow over ensuing years into the closest collegial partnership either was to know. Although Charlie had been working with TX-2 for some time, it was only at this point that I began to be peripherally aware of the presence in the lab of a trim young man clad in Air Force uniform.

Today, thanks to the great proliferation of computers, the design of most machines is constrained by the need for compatibility with existing programs and methods of use. But in the early sixties things were still much more in flux and style had by no means settled down. The initial stages of design of a new computer were therefore the hardest to understand because the overall conception, the recognition of a hole waiting to be filled, required in many ways the

keenest insight, the purest invention. When the developers of the first xerographic copying machine put their toe in the water with a market survey, they were told that there was no need for such a thing—people didn't want to do that much copying. It took imagination and conviction to defy this now laughable analysis. Similarly, during the middle-ages, when a wide variety of currents and cross-currents swept back and forth across the conceptual space as a host of now long-forgotten architectural experiments took place, it took especially keen insight to decide that the thing to do was to try to build a personal computer.

As the concept for a new kind of computer begins to firm up, as the niche it is to fill becomes more clearly defined, a process of distillation commences in which ensuing design becomes a matter of slowly refining the ideas in ever greater detail. The first steps define the broad architecture of the machine and delineate how it will be perceived by the user.

When Wes asked me to tour the lab looking for possible applications that would help to justify such a machine's sponsorship, I now believe that my search was a post hoc exercise. He already knew, generally, what he was about. As time passed those of us working with him came to rely on his instinct and judgment—too much perhaps. Many months later I came into his office one day to find him looking worried and staring abstractly into space. When I asked what was the matter, he said, with some agony in his voice "Charlie thinks I know what I'm doing!" I didn't dare tell him that the rest of us were suffering from the same delusion.

That spring (1961) Wes disappeared from the lab for an extended period. When he returned he brought with him notebooks containing a preliminary design for a new small

computer. A number of us gathered around to listen as he laid out the prospective design. We took notes furiously. Then he disappeared once more, leaving us to try to remember, ponder, and critique what he'd done, and to figure out how to program such a beast. This latter task fell to me and a newcomer named Mary Allen Wilkes, a philosophy major from Wellesley who had recently joined the group. Together we tried to sort out what Wes had said and to understand how his new brainchild worked—well enough to be able to write some trial programs and see how it felt.

In the weeks that followed, Wes would reappear periodically to accept our insights, answer questions, and and then quietly announce that the design had changed—here was the new version. Undaunted, we'd go back to our deliberations, readjust our thinking and start over. Fairly quickly the process converged and a solid design began to emerge and acquired a name—the α-Linc. It was Wes' design; we were mere hangers-on, and I use the term advisedly. I wasn't used to a machine design that constantly shifted under my feet, and I hung on for dear life.

Gradually the description of the machine stabilized and started to be reduced to logic diagrams. I'd had no direct experience with logic diagrams, but Wes felt strongly that everyone should learn as much as possible about the entire process; there was to be no escaping into a narrow specialty. He made it easy for us novices by starting with simple pieces of hardware logic and showing us how they were put together to form bigger, more complex devices. A gentler, more encouraging teacher would be hard to find. I began designing small exercise devices (not unlike the process I'd gone through several years before when I was first learning to write programs). Gradually I began

following the design of the machine as it progressed, learning new things every day.

Wes decided to build the machine out of the new logic modules that DEC was now marketing—the very same Digital Equipment Corp. that Ken Olsen, Harlan Anderson, Dick Best, et al. had departed from Lincoln to form not long before. In principle, these modules could be treated as black boxes that performed various logic functions and could be connected together following a few simple rules; but that principle was only approximately true, and there was more to learn here than with programming. A programmer was provided with a well-defined set of instructions whose definitions had hard edges. If an instruction didn't work as advertised, the machine was broken by definition and you called in the hardware experts. With logic design, even with DEC's comparatively well-behaved, well-defined black boxes, the "analog" world of electronics, with all of its complexity, was lurking just below the surface to rise up and bite you in some mysterious way if you unwittingly overstepped some bound.

It was at this stage that Charlie began to emerge from the shadows and forcibly entered my consciousness for the first time. Although as an Air Force officer he was still only peripherally involved officially, he was nonetheless rapidly emerging as the co-star of the project and was starting to make major contributions to the design. In fact, although I came to realize it only later, he had been doing so for some time. As Wes described it later, Charlie had an intimate relationship with every electron in the known universe. He not only had serious electrical engineering credentials, but superb taste and judgment. He was the most thorough and careful engineer I was ever to come across. Thus when

important engineering decisions needed to be made, Wes turned naturally to him.

But Charlie was also just plain fun to work with. He had an irrepressible sense of humor and unmatched skill at weaving it into every aspect of technical work. He seemed to view engineering as an enormous joke that, if properly done, could be played on the the Gods who give grief to engineers. He was also a titillating and natural teacher who employed the Socratic method everywhere. Those of us who were effectively his students, quickly adopted his "Don't Trust Nobody" dictum.

Another colorful member of our "gang" was Tom Stockebrand who, as I've indicated earlier, had prior experience with several different kinds of magnetic tape units. Wes wanted a pair of small "snapshot" tape units to be a standard part of the α-Linc. These would be a significant innovation, not only technologically but logistically. They were the forerunner of later diskettes and floppies in that they provided the user for the first time with small, removable devices for storing one's own programs and data—something far more compact than the cumbersome card decks of commercial machines. Wes had devised a clever method by which pre-numbered blocks of information on the tape could be located, read, and written. I was given the job of figuring out how to pre-mark the tapes with the information necessary to control the tape movement and locate the blocks. Greater faith, I thought, hath no man than he who assigns such a sticky bit to a novice like me. Stocky was to design the physical tape units themselves—tape heads, belts, reels, motors, etc., and associated mechanics and electronics. We began working together closely as I started to understand how the tape control logic would work.

The Original LINC Crew

Wes and Charlie continued to oversee what we were doing. At one juncture, as we were discussing a bit of my design, it became obvious that although it worked properly, I had misunderstood some crucial fact. I defended myself, pointing out that after all, my design did work. Whereupon Wes drew himself up and announced that "In this business it's not sufficient to be right; you've got to be right for the right reasons." It has sometimes seemed to me that this statement defines a nice demarcation between the engineering and the scientific mind-sets.

As we moved into winter (1961-62), the actual machine began to take shape. I was about to discover further differences between constructing a machine and programming one. In programming, the design and coding were pretty much the entire job (aside from uncovering and

correcting mistakes). The job of converting one's code into a running program was but the work of a few minutes with the aid of an assembly program, compiler, or whatever. Not so with hardware. Once the logic design was complete on paper, the job was only just barely begun. The ensuing process of transforming the design into an actual physical machine ready to be debugged, was a major piece of work. For instance, you had to decide where each of the various logical pieces would be located physically within the machine, and if you did a bad job of laying that out, the wires that connected things together could become nightmarishly messy. In order to minimize such "steel wool," things that required lots of interconnections needed to be located as close as possible to one another in an orderly way. (In designing later, far faster machines, the lengths of wires would become critical in determining the speed of the machine because even at the speed of light, the length of time it takes signals to travel along the wires can become significant. These days, the very operability of the machine depends on how well the wiring is laid out. In the comparatively sluggish α-Linc, it was more a matter of neatness and order.)

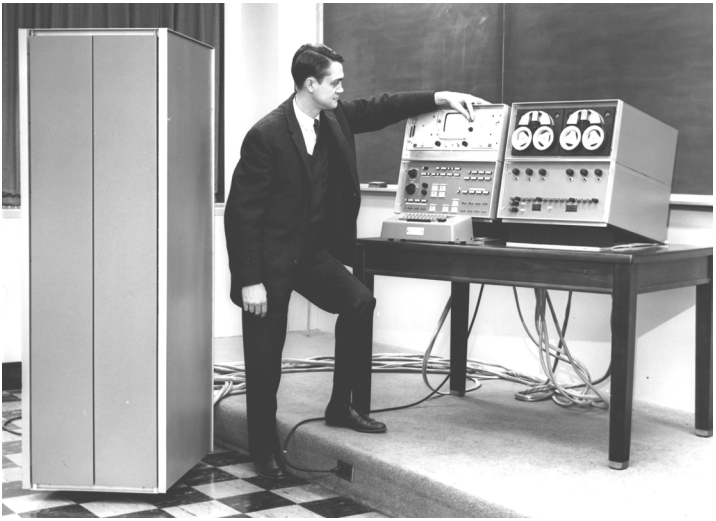
Once all of the modules had been assigned to specific locations within the machine, wiring lists needed to be prepared that would be used by technicians to install the thousands of wires, in this case soldering each one in place in turn. As the years progressed, a wide variety of techniques for wiring machines have been tried and discarded as, one after another, new techniques replaced them. This process has been driven by the ever-increasing compaction of logic elements into more and more dense packages. The density of packing, the level of "integration," has increased extraordinarily rapidly until today the

number of external connections (i.e., those *between* packages) represent only an infinitesimal fraction of the total inter-connectivity. Virtually all of the wiring of earlier machines has been replaced in today's computers by fine traces within the individual integrated circuits themselves. The compression of many circuits onto a single chip has eliminated the need for the numerous wires and cables and connectors that joined together the various parts of earlier machines.

Somehow, as those things will, a schedule had evolved, along with a target date for a demonstration of the working machine to all of Lincoln. Early one foggy morning, having worked throughout the night, Stocky and I were able, for the first time, to write and read blocks of information on the prototype tape units. As the rising sun struggled to pierce the mists, we celebrated our achievement by climbing a nearby radar tower to peer into the fog. Despite the gray weather, we were exultant; we happily ate breakfast while we waited for the others to arrive and share our excitement. For months afterwards I was to look down at my feet and see a splotch of bright orange paint that had transferred itself from the antenna ladder onto one of my shoes as we clambered up. It was a mark I secretly treasured.

Miraculously everything came together on schedule and in March of 1962 the demonstration went off without a hitch. The final machine consisted of four boxes: a control panel with lights and switches for manipulating the machine; a box containing a small (3″ x 5″) display and analog knobs that could be used to vary parameters in a program; a box that contained plugs for connecting the machine to external devices, both analog and digital; and

finally our precious dual tape unit.[22] These sat together on a desk and were connected by cables to a larger box, variously described as refrigerator-sized or coffin-sized, containing the power supplies, logic, and associated electronics that made the whole show go. We pushed this part as far out of sight as we could, thereby suggesting that it would someday disappear altogether. Few of us anticipated that most of the other boxes would also disappear and that the whole shebang would one day fit into something you could tote around as easily as a notebook.



Wes Clark Demonstrating the LINC at Lincoln Lab

[22] This describes what, after a few minor modifications, ultimately became a "Classic LINC." The prototype initially demonstrated at Lincoln Lab differed slightly in detail.

Almost immediately the machine went on the road to Washington, where first it was demonstrated to the National Academy of Sciences and then in labs at the National Institutes of Health (NIH). These demonstrations were extremely successful and over the coming months plans emerged for an enlarged program in biomedical computing under NIH support. Things looked extremely promising early in 1962, but then we hit an unexpected snag. Lincoln management, accustomed to the more generous overhead allowances of its military funders, was unwilling to deal with a new set of sponsors. Following this unfortunate revelation, at our weekly meeting Wes announced to a stunned gathering that he would be leaving Lincoln Laboratory—and that, by the way the α-Linc (which everyone had naturally assumed stood for Lincoln) was henceforth to be the LINC—the Laboratory INstrument Computer.

Aside from their superb technical abilities, the human qualities of Wes and Charlie stand out as utterly unique in my experience—I could not imagine working with a finer group of people. After the meeting I went for a walk with Wes and assured him that whatever home could be found for the LINC would suit me fine; we soon learned that most of the others in the group felt similarly. Stocky, however, opted to join DEC, taking with him the design of the tape system which he and others redesigned and turned into the DEC tapes that would serve as the principal program input/output mechanism on forthcoming DEC machines[23].

[23] The idea of small, portable snapshot devices, originally represented by LINC tapes, would ultimately metamorphose into today's floppies, diskettes, etc.

When I saw Stocky a few years later he was unrecognizable. Heretofore he had been at best a casual dresser. (I retain an image of him hopping around on one foot in the lab, busily massaging the other foot as he argued some engineering question with Charlie. He'd been barefoot and had come a cropper on a raised electric plug fixture.) Now he was wearing a suit and showing surprising interest in business matters. It seemed out of character, but I needn't have worried. Before we leave Stocky for good, I must punctuate his departure with a story from more recent times when I looked him up a few years ago in semi-retirement in Albuquerque.

Like many inventive engineers, Stocky had a small lab in his home and, peering into the clutter, I could see that the suit had been only a temporary diversion; Stocky was clearly still the same lovable, glue-and-safety-pins fellow I'd known in the past. Not long before, the house had been burgled, and in their haste the thieves had wreaked havoc in most of the rooms. Miraculously, however, Stocky's lab, in which precious experiments were under way, hadn't been touched at all. When the police arrived, they started methodically going through the house, room by room, recording without comment the extensive damage. On reaching Stocky's lab, however, they drew back in horror, exclaiming "Boy, they really trashed *this* place."

The rest of us felt bound to one another; we had practically become a family by then. Each of us had poured a large quantity of blood into the LINC development and we weren't inclined to let it go. Moreover, we had developed a conviction that we were on an exciting trail and that our futures lay ahead in biomedical computing. I was tired of the military ambiance of Lincoln, fed up with the absurdity of having to show a badge to guards who

knew me. So although the immediate future was uncertain, I felt no hesitancy about leaving Lincoln, and as 1962 drew to a close we prepared to pull up stakes and move.

Chapter 10

We move to Kendall Square where we accomplish the impossible

By the time we left Lincoln in January of 1963, considerable spade work had already been done in providing a new venue for our work. Earlier in the year a proposal had been made to NIH to establish a multi-institutional Center for Computer Technology and Research in the Biomedical Sciences, with MIT acting as host. Many of the major universities in the New England region had signed on in principle and the proposal had been accepted and funded to the tune of some $5 million for the first year, with the promise of more substantial funding ($27 million) over the ensuing five years as the center grew. In the first year, an office was to be established that would simultaneously work on developing plans for the forthcoming center and proceed with further development and dissemination of the LINC. This office was given the absurd, albeit descriptive, name of The Center Development Office for Computer Technology and Research in the Biomedical Sciences—affectionately known as the CDO. We moved into a building in Kendall Square in Cambridge, snuggled up against MIT proper. Ironically just up the street lay MIT's grand Technology Square complex wherein an ambitious Time Sharing system (Project MAC) was simultaneously under development. Our modest quarters above a health-food store symbolized the

contrasting attention being paid at the time to the two visions of computing.

Our initial task was an ambitious experimental program, funded by a combination of NIH, NASA, and NIMH (National Institute of Mental Health) and known as the LINC Evaluation Program. Approximately twenty copies of the LINC were to be built, installed, and evaluated in a variety of biomedical research laboratory environments around the country.

We knew that since there would be no maintenance organization to repair these machines when trouble arose, each laboratory would have to be able to provide for the care and feeding of its own machine. In order to empower the researchers to take on this formidable and unaccustomed task, we decided to produce the LINCs in kit form and to bring the researchers (each with an accompanying technician as desired) to Cambridge in two waves. There, over the course of *one month* for each wave, they would assemble and debug their own computers under our supervision. Virtually none of these individuals had any prior experience with a computer and in one month we would have to teach them not only how to program the machine but also how it worked, and how to maintain it as well. It seems we knew no fear in those days.

It is interesting to contrast this situation with today's. At that time both hardware and software were far less complex than they are now, even in the most modest contemporary computer. But the hardware, with all the little separate pieces and connections, was inherently more susceptible to failure than that in a modern computer (although the urge to press the limits of speed and capacity has always helped to depress hardware reliability). Not only was the hardware inherently more vulnerable back

then, it also contained a much larger fraction of the total system complexity than it does today. The LINC came with no elaborate operating system or application programs, only a relatively straightforward assembly program. Virtually all of the software that a user would employ would have to be developed and written directly by that user, who would thus be intimately familiar with its purposes and implementation. Today a far larger fraction of the overall system complexity lies in the software provided to users by the computer manufacturer and others. Most users are thus utterly at the mercy of the developers of the system software as well as the vast array of application and utility programs that inhabit a modern computer. Meanwhile the hardware, with all its connections that used to flap vulnerably in the breeze now compressed into a few integrated circuit chips, has become much more reliable.

My experience suggests that hardware designers are, by nature, more careful and more thorough than most software designers and programmers. They have to be, given the relative difficulty and cost of correcting hardware design errors. The cost of a hardware error today can run into many millions of dollars and this produces a healthy sense of paranoia in the minds of hardware designers. Modern programming tools, on the other hand, provide software developers with powerful means for manipulating and easily changing programs. Furthermore, the rapid decline in the cost of memory has encouraged the constant addition of features in programs, often leading to rococo structures that no one fully understands. Software engineering practices have been developed to contain and counter these problems, but unfortunately they are more often honored in the breach than in practice. The consequence of these things is a software industry afloat in revisions and changing

versions that often leave the user futilely shaking a fist at the screen. The end user has been conditioned to accept (if not love) programs that, as a result of endless bells and whistles he'll never use, are cumbersome, flaky, and often poorly suited to his real needs. The acronym KISS (for Keep It Simple, Stupid!) has been thrown out the window by the new generation of programmers feasting on today's memory abundance.

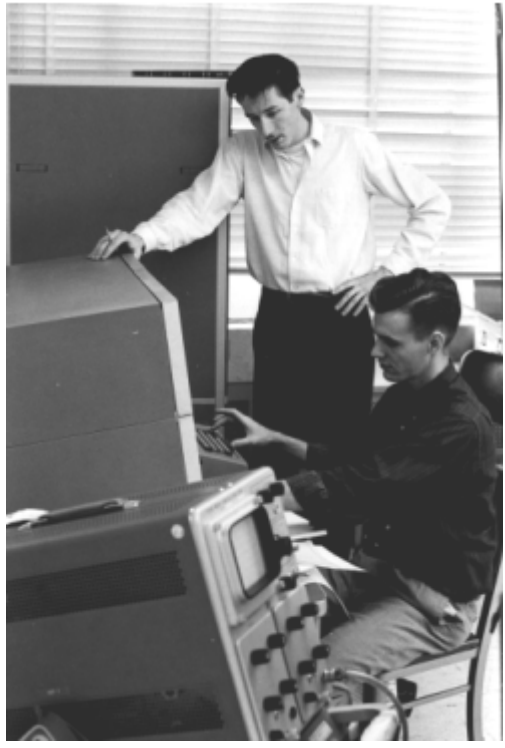
Teaching programming and providing significant hardware understanding to twenty users with virtually no engineering background (*doctors*, for heaven sake!) was an ambitious undertaking, particularly as, before it could commence, the LINC needed major redesign after which we would need somehow to assemble some twenty or so kits of parts. But we were young and invincible, and by that time a number of us felt confident of our own understanding of the machine. None of us, however, had had any experience in even limited quantity production. At about this time one of the crew (Don O'Brien) reported a vivid dream in which Wes was driving a steamroller directly towards the edge of a precipice while Charlie was furiously attempting to attach wings. Bill Simon (another member of the crew) sat in the rear, speed-reading a book on aerodynamics! An altogether apt image it seemed.

By now (the spring of 1963) it was ski season and one day Mary Allen appeared on crutches wearing a serious looking cast on her leg and saying "The doctor says I can go back to work tomorrow!" That week at our regular gathering, Wes announced in no uncertain terms that there was to be no further skiing until the machine was finished. We were a small team and every absence presented a sizeable problem.

Mary Allen

At this juncture, Mishell Stucki joined the team. Mish was a long, lean ex-Harvard student who had previously worked under the vigilant eye of Charlie Molnar at the Communications Biophysics Lab. It soon became evident that Mish carried Charlie's careful work and thought habits to an extreme that even Charlie could envy. Over coming years he would exhibit a level of caution and fastidiousness that could immobilize any project he was working on until he was completely satisfied that he understood every relevant nook and cranny of underlying theory. Mish and I,

who made a sort of Mutt and Jeff team (Mish being nearly twice as tall as I) worked closely together over the next few years and for a while, following my separation from my first wife, we shared an apartment. (One day the apartment was burglarized and we tried to imagine what kind of misshapen thief had made off with a pair of my pants and one of Mish's jackets!)



Mish and Howard[24]

[24] Howard Lewis was one of our superb technicians.

The LINC redesign went smoothly for the most part. Wes redesigned the control panel and I helped design the logic that underlay it. I then redesigned the somewhat expanded tape-handling logic. Charlie reworked the physical tape unit itself and where our work met was in the signals between his tape heads and my tape logic. In lieu of a quality control department we had instituted a betting regimen—the better to keep errors to a minimum. Wherever anyone foresaw a possible problem, a bet (usually consisting of one or more martinis, depending on the complexity and importance of the matter) would be placed to the effect that the designer could not possibly get it right on the first try. The tape head signals for the various channels varied in polarity and underwent individual sets of inversions as they passed through amplifiers and gates on their way from the heads to the logic circuits. The same applied to the signals going in the other direction for writing on the tape. The likelihood of an error somewhere in this chain seemed semi-infinite—a situation that clearly warranted a substantial wager. I bet Charlie (I forget how many martinis) that he couldn't possibly get all of the polarities right on the first try, figuring that getting it right would more than repay the cost of all those martinis. Although Charlie was nearly infallible, this was a difficult challenge and some insurance seemed in order. Charlie took the bet seriously and pondered the matter carefully, but nonetheless he lost and collapsed on the floor in a fit of despondency that lasted for days. Meanwhile, of course, things were corrected and ultimately the tape units worked.

Charlie

While this was going on, Mary Allen was working on an assembly program for the machine that would allow programs to be written using mnemonic instructions and address tags. Since the redesigned machine wasn't yet working, it was arranged that she should use TX-2, which of course required that first she do for the LINC what I'd previously done for FX-1, namely, write a LINC simulator

that she could then use to debug her LINC assembly program. Lifting herself thus by the bootstraps required much traveling back and forth to our old haunts at Lincoln to access TX-2.

One night I arrived late to a darkened lab and discovered Wes underneath a table fiddling with the controls of an oscilloscope and muttering to himself "First you've gotta get its attention." I had just managed to get the logic for the control panel working and after we turned on the lights I demonstrated it for him. It seemed to work properly, and after we had checked out everything he said, "Lets give it the Forrester test" —whereupon he picked it up and, to my horror, proceeded to drop it onto the workbench from a height of about six inches. After that of course nothing worked, and when the shock wore off I realized I'd been given yet one more lesson: in this business, it's not enough to get it working; it's got to go on working, even when it's thumped. Hardware design was proving a tough racket.

We had become a very close-knit group by that time, closer than any other group with which I would ever work. We worked without regard for the clock until we were ready to drop, eating together irregularly in the nearby deli which grew accustomed to serving breakfast to us at three in the afternoon or dinner at seven in the morning. Despite the closeness, we remained rugged individualists. One day five or six of us were gathered around a huge set of tables that had been assembled for working on the big logic drawings. Each of us was working away at his or her own piece of the design when suddenly I realized that all of us were whistling or humming different tunes—one Mozart, one Brahms, one Bach, one Rachmaninoff, and one Bartok.

Then, as such things will, at a critical moment a serious memory flap arose. The authorities (Wes and Charlie) weren't sure why the memory didn't work and reserves were called in. Professor Jerome (Jerry) Cox, another superb electrical engineer who himself was looking forward to receiving one of the machines, arrived from Washington University in St. Louis. Together, he and Charlie eventually figured out what was wrong and worked the necessary repairs. Once the memory was working, I could begin checking out the tape logic. I recall feeling both satisfaction and some loneliness as the others all trooped out about 10 PM leaving me alone with the machine and a fast-approaching deadline.

When I left in the wee hours, I had a rather messy test program stored in the memory. Since without working tape units I had no way of saving it, and since reentering it via the control panel was a slow and tedious process, I left a note requesting that if at all possible the memory contents should be preserved. Next day when I arrived, I found a complete memory module (all 4,096 12-bit words in a neat 6" by 6" by 6" package) sitting on my desk with a note saying "Here's your program." Recognizing that the little cores in core memories retained their magnetism even when not powered, Charlie had simply unplugged the memory module containing my program and plugged in a replacement unit for his work with the machine. In the light of today's technology such a maneuver seems utterly mundane, but at that time the idea of casually replacing a computer's memory startled and bemused everyone. Sure enough, when I plugged things back together that evening (with the aid of the requisite memory-installing rubber mallet), my program was sitting there intact, waiting for me

to proceed. Over the next few nights I managed to debug the logic and get the tape units working.

Our technicians developed a proprietary attitude towards the physical machine as it came together, and with good reason; they knew how much trouble a superfluous bit of solder could cause and quickly learned which of us could be trusted with a soldering iron in our hands. I soon found that by merely picking up a soldering iron I could produce a technician from out of the woodwork, one who would be more than happy to implement the change I'd had in mind.

Once the tape units were working it was time for a celebration. A large number of us piled into Bill Simon's car and headed for a restaurant in downtown Boston. Parking, even back then, was notoriously difficult, and as Bill drove around back streets searching for a space, Wes' voice rose periodically from the depths of the rear seat saying, "Go around front, Bill. They're holding a place for you." We all laughed, of course, at the absurdity of this repeated mantra, until we happened to pass that way when, lo, there indeed was a place, right in front. On another occasion Bill was taking a break throwing darts at a board in the lab. He was making heavy weather of it when Wes stopped briefly in the doorway to watch. Then as Bill advanced to the board to retrieve the darts for the next round, Wes picked up one of them and saying, "No, Bill. Not that way; *this* way." without looking, he tossed the dart backward over his shoulder as he withdrew and disappeared down the hall. Bill and I watched, open-mouthed, as the dart found its way directly to the bulls-eye. As I observed our mentor for clues to success, not only in engineering but more generally, I noted that he occasionally made such improbable predictions. I finally figured out that he was relying on the

fact that outrageous predictions that chanced to be fulfilled would be remembered, while the numerous ones that failed would be quickly forgotten. Thus are soothsayers' reputations formed.

One day not long afterward I was called into the lab to observe a tape unit that was behaving erratically. It had somehow gotten into a mode in which it was rocking gently back and forth as though searching for, but never finding, some block on the tape. Such problems were my domain, and gradually our heads bent lower and lower over the tape unit in an attempt to discover precisely what was going on. Suddenly the air was rent by a violent explosion right under our noses. We reeled back in time to see a dense black cloud, a miniature mushroom cloud, rise majestically out of the unit. After the dust had cleared we discovered the remains of a capacitor which had exploded, calling a halt to both the tape's unusual behavior and our investigations. More lessons in electrical engineering.

The tape units were being put together by a small company in Nashua, New Hampshire. Late one evening Charlie and I set out to visit them and retrieve the prototype unit that had been lent them as a model, along with the drawings. Later, as we got into the car to return home with it, I noticed that Charlie was chuckling. When I asked him what it was about he said that he could see evidence that they had taken the unit entirely apart and then reassembled it, a totally unnecessary procedure as the model was only to provide a general picture of how things went together. Charlie speculated that they had realized that this tape unit was likely to become a commercial success (as, indeed, a later DEC version did), and that in their eagerness to understand it, they had taken it apart and then put it back together again. The reason Charlie was chuckling was, of

course, that without understanding the logic in the main machine that actually controlled the unit, there was no way that exploring the physical unit was going to reveal anything about how it worked since there was almost nothing *there* in these mechanically primitive units.

While we were working on the LINC, people at DEC, who shared our Lincoln/TX-2 roots and consequently some of the same attitudes, had been designing a small machine of their own, the PDP-5. It had a simpler set of instructions than the LINC and lacked the special features that lent the LINC its unique power—the small magnetic tapes, the display, and the analog capabilities. It was a more conventional computer than the LINC, except for its "logical size." It was meant to be a small machine, but only became physically small when repackaged some time later as the PDP-8, which was to prove an extremely popular computer. Small computers were beginning to make headway, but, except for the LINC, none of them had an integrated display that looked forward to a more interactive form of use.

Finally we got the redesigned machine working and found manufacturers to provide or build all of the necessary pieces. By far the most complicated part was the wired frame into which the machine's numerous logic modules were plugged. On its surface lay the nests of wires connecting everything together. Every wire was recorded in an enormous wire-list, copies of which were then provided to the company that manufactured the frames. But we knew that there would be errors, and debugging a machine is difficult enough when everything is properly wired together. Half a dozen wiring errors can render the task virtually impossible. We devised a method for verifying the wiring and somehow everything got finished—in the nick

of time. By early in the summer of 1963 when the first group of "participants" showed up, many of them famous medical researchers, we were exhausted but nearly ready for them.

A LINC Kit

While the finishing touches were being put on their kits, we lectured to them, stalling for time. Some of these middle-aged gents hadn't been worked so hard in decades. A tremendous amount of information needed to be pumped into them (one said he felt like pâté de fois gras), and we drove both them and ourselves ruthlessly day after day, night after night. They were first amused, then frightened, and ultimately exhausted by what was happening to them. But there was no time to spare. We had to get them up to speed and help them get their machines assembled and debugged before the second wave of participants was due

to descend. Somehow we and they managed it, and three days after the first group departed with bloodshot eyes and unopened golf bags, we were confronted by the second wave. This time we were somewhat better prepared and things went more smoothly, if just as intensively. Shortly after they too left, we packed up all of the machines and watched as they were loaded onto a large moving van that was to deliver them to the various laboratories around the country. Heaving a great sigh of relief, we staggered home to sleep for a week.

In retrospect I cannot imagine where we found either the bravado or the strength to bring it off, but somehow it all went like clockwork. Somewhere along the way we had become possessed of missionary zeal. We were cracking open a whole new area of computer application, one that unquestionably had a humane purpose, while at the same time promulgating a dramatically new technique of computer usage in which complete control was vested in an individual "owner." No doubt our vague understanding of these matters carried us, like soldiers in war, well beyond our normal capacities.

Chapter 11

Tragedies overtake us; the prime number drop; we move to St. Louis where we design some new building blocks and encounter the evil synchronizer bug. I assist in brain surgery

We were of course elated at our success, and naturally a celebration seemed in order. One afternoon shortly thereafter we found ourselves purchasing the necessary supplies at the local supermarket when suddenly the news went round the store that President Kennedy had been shot. We rushed back to the lab where we listened to the radio in shock and disbelief. When it became apparent that he was indeed dead, one by one we dispersed to our homes where we retreated into our individual sorrows. We had watched TV with deep concern in our local deli. as Kennedy had defied the arrival of Soviet missiles in Cuba. Aside from missing this man who had so captivated the entire nation, we wondered what would happen now.

But far closer shocks were in store for us as our dream of a major inter-university center suddenly foundered on the rocks of MIT politics and eventually vanished before our very eyes. As plans for the proposed center had unfolded and it became evident that it had the potential to become a major MIT institution, members of the academic faculty naturally began to take serious interest in the development. Gradually a power struggle emerged regarding who would control this new organization. Having conceived and developed the machine that gave

birth to the whole idea, our group was inclined to retain control of developments. But of course we were outsiders (none of us had serious faculty positions) and naturally the faculty expected to exercise its normal prerogatives in such a situation. Ultimately the decision fell to Professor Charles Townes, the renowned physicist who was at that time provost of MIT. In a dramatic gathering in Townes' office one afternoon it became clear that he had little choice but to back his faculty.

It was a terribly discouraging time, particularly for Clark who had been the real pioneer and the driving force behind the development. He said sadly at the time that the decision had set back the introduction of computers to medical research by many years. Probably this estimate was unduly pessimistic as events were to unfold, but certainly our hopes were dashed that day by political forces. Even though NIH had already set aside substantial funds for it, the Center was not to be. Unwilling to hand over the infant we had borne, we had no choice but to seek another home for our activities. Today one might well think in terms of starting up a commercial enterprise, but that was far less feasible back then as the revolution we had in mind would have been incomprehensible to potential funders. Besides, we had academic leanings and were determined to cultivate further the embryonic relationship between computers and biomedical research that we were hoping to foster. That meant looking for a university with good engineering and medical schools in which the top management would look favorably on our enterprise of interbreeding the two disciplines. Experience had taught us how important such top-level support was.

Early that spring (1964), various members of the team flew around the country visiting potential host institutions.

We finally settled on Washington University in St. Louis, where Jerry Cox had already established a beachhead with his Biomedical Computer Laboratory near the Medical School. George Pake, who would later become the director of Xerox PARC and still later director of research for all of Xerox, was at that time Provost of Washington University, and his understanding and support for what we were trying to do was a major factor in our decision to move the group to St. Louis.

Soon afterward, Wes and Charlie began work on a new proposal to NIH, describing the situation at Washington University, what work we would do, etc. The writing was going slowly and Jerry Cox, sitting in St. Louis, began to fidget, wondering what was holding things up. Finally, overcome by curiosity and concern, he climbed onto a plane and came to Boston. When he arrived he found Wes and Charlie locked in a debate about whether 2047 was or wasn't a prime number; Charlie said no, Wes said yes. (A prime number is one, like five or seventeen, that is divisible by 1 and by itself but by no other numbers.) There sat the untended proposal, while Wes tried successive candidate divisors. Jerry promptly ordered the two of them back to work, promising that *he* would figure out whether or not 2047 was prime. So, starting where Wes had left off, he continued looking for possible divisors. Finding none, he announced that 2047 was indeed prime—whereupon Charlie's hand instantly shot out and he said, "Wanna bet?" Had Jerry known Charlie better, he would never have accepted the bet, but as it was, he agreed to treat everyone to dinner if proved wrong. It turned out that in his part of the search, Wes had overlooked the fact that 2047 is divisible by 23. Jerry paid up, but was ever after convinced

that he'd been duped by what he referred to as "the prime number drop."

Later, in a memorable meeting between our NIH supporters and Washington University senior management, Pake confidently told the NIH people that he intended to take all of us on as faculty and staff members, regardless of whether or not NIH decided to continue their support of our activities. Occasionally the convictions of a single individual can help to shape history, and that day Pake fearlessly backed something he had come to believe in deeply. This was the kind of support we had not found previously and we were all extremely grateful.

But the NIH people also wanted to know what we were planning to *do* at Washington University. They understood that we were going to continue supporting the LINCs in their various domiciles around the country and that we hoped to build a strong set of ties between Washington University's engineering and medical schools (whose prior degree of separation was symbolized by the giant park that separated the two schools geographically.) But, they insisted, what were we going to DO? What new research directions did we have in mind pursuing?

We'd had some rather vague discussions about this matter, but nothing terribly firm had come out of them. We felt that it was important to be able to design computers, and related pieces of special purpose digital hardware, far more easily than was then possible. It should be feasible, Wes felt, to come up with a limited set of building blocks that were truly logical elements of significant power, in which the rules for interconnection should be extremely simple. The builder should not be required to have any electrical engineering background whatsoever, so all questions of electrical loading, timing, etc., would somehow

have to be pre-solved. One should furthermore be able to build systems of arbitrary size and complexity; the units would need to be extensible. We called these building blocks "Macromodules," lending them a reality which, at that stage, they didn't posses at all. In truth, we had little idea how to accomplish such an ambitious goal.

In the meeting all eyes were on Wes as the group's guru, and watching him I realized to my dismay that he was in no state to make a strong case to the NIH committee. The collapse of our enterprise at MIT and the consequent split-up of the group had been a devastating blow, and at that moment he was depressed and it showed. To their credit, our supporters decided to continue their backing, despite the tentativeness of our presentation, trusting that the miracle we'd produced would continue to grow and prosper. Ultimately it was settled that we would move to Washington University, and shortly thereafter firm offers arrived in the mail.

The move from Cambridge to St. Louis represented far more serious geographic dislocation than had the move from Lexington to Kendall Square in Cambridge, and some of the group opted to stay in Massachusetts while others chose to accept appointments at other universities. Mary Allen punted and climbed onto an airplane for a year-long trip around the world. Of greatest concern was the fact that Charlie, so essential a member of the group, would have to remain in Massachusetts for another year in order to complete his Ph.D. and his military assignment. As we shared a bottle of scotch one night, Wes expressed grave concern on this point. "Charlie is the key," he said.

The rest of us packed up and in the summer of 1964, one by one we arrived in the "Hub of Missouri" (later referred to by some of us as the "Hub of Misery")—a play

on the Bostonians' famous assertion that Boston is the Hub of the Universe. My second wife, Elizabeth, and I and our cat arrived at midnight, and as we pulled in we spotted a large, rotating sign indicating that the temperature was 100 degrees F. It was clear that we were in for a rough stretch, and I later told Wes that all I could promise for sure was a three-year hitch.

Mish and I, being two of the first arrivals, soon set to work together trying to figure out what Macromodules might be. Unbeknownst to us, we were embarking on one of the more innovative periods of our careers—at least of mine. Wes was still suffering postpartum depression and watched over us from a somewhat greater distance than usual. We found that a truly clean sheet of paper opens the mind wonderfully, and as neither of us was afraid to appear the fool in the other's eyes, we plunged in fearlessly.

We soon realized that working with the constraints and goals mentioned above, the kind of signaling system used inside most computers simply wouldn't work. Most computers operate based on a central clock that ticks regularly[25]. These are known as synchronous systems since all changes in the machine take place exactly at clock ticks. Between ticks, the news about what happened as a result of the previous tick must be able to propagate everywhere throughout the machine so that it can figure out what to do on the ensuing tick. If the clock runs so fast that the next tick comes before the results of the previous one have

[25] When a computer is advertised as a 300 megahertz machine, it means that its clock is ticking three hundred million times every second—a frightening number when you think of it.

arrived everywhere, then the machine can make a wrong decision about what to do next and chaos will result.

Signals take some time to percolate throughout a machine; they travel along wires and pass through various logic elements that tend to slow them down. The bigger the machine, the longer the wires, the more elements, and thus the more the slowdown. Because we wanted to be able to use our modules to construct machines of arbitrarily large size, for any given clock rate, as larger and larger systems were built, sooner or later some signal would arrive too late at some unit and would produce an error. We therefore concluded that our building blocks would have to be self-timing (asynchronous) in that they would somehow have to guarantee that any change would be made known throughout the system before proceeding to the next step. This led us to invent a totally new signaling scheme and to devise circuits that implemented this scheme.

Every day Mish and I would together invent something new and then, going home and sleeping on it, one or the other of us would have a new insight and would arrive at work to announce excitedly that yesterday's idea was all wet and here was a better one. And then we would be off once more at high speed. We recorded our progress with frequent Polaroid photos of our chalkboard with one or the other of us standing smirking beside our latest brainchild. Sometimes our disdain for yesterday's idea or for one another's current suggestion would become surprisingly harsh and in the excitement of invention our anxiety at times verged on hostility. It was merely overstimulation, but others listening to us shouting at one another sometimes must have thought that we were having a dreadful row. They seemed surprised to see us later jawing calmly together over lunch. Occasionally Wes would

appear and we would brief him on our progress. I remember that on one occasion, after listening silently, he turned to us and announced that we were making good progress, that we'd now reinvented ILLIAC III. That sent us scurrying to find out what the hell ILLIAC III might have been. (It was, indeed, a previous asynchronous machine.)

As we continued to work, our ideas gradually started to jell and a group of basic building blocks began to crystallize in our minds. We were able to describe these blocks as logical units and understood how they would operate and could be connected together to form arbitrarily large systems. Our new signaling scheme extended naturally as the system grew. Wes started spending more and more time with us and the excitement intensified. Another bright spot was the arrival, at this juncture, of a student of Charlie's from MIT, one Warren (Mackie) Littlefield. (It was to Mackie's computer class that I spoke not long ago). Charlie had indicated that I was going to like Mackie and, as usual, he was right. Mackie's contagious enthusiasm and high good humor provided an excellent antidote to some of the less appealing aspects of St. Louis. Our friendship was instantaneous and electric, and has endured to this day, despite significant later shifts in direction for both of us. He immediately began offering welcome critique of everything in sight, and with such good humor that we all promptly accepted him into the fold. Besides, he was frequently right.

The problems of reviewing scientific research are substantial. Who, after all, is competent to judge what goes on at or near the leading edge of investigation; which activities appear likely to be fruitful and which should be judged dead-ends? NIH engages in a routine process of peer review in which qualified experts drawn from the broader technical community, often as consultants, gather

to review the work of a particular group of researchers. This process relies on the integrity of the workers involved because it could obviously tend to produce an old-boy network of mutual admiration and support. My impression, however, is that it works extremely well in most cases since it is the nature of scientists to try to knock down ideas and kick holes in theories in the broad search for scientific truth. In any case, our group was occasionally the subject of such site visits. One, in particular, stands out in my memory because it included Alan Perlis as one of the reviewers of our work. Perlis was one of the deans, if not *the* dean, of American computer science. Besides, he was a memorable character, already bald as a billiard ball and with a bottomless sense of humor. I was to re-encounter him again many years later in a very different context and ultimately he was to become a good friend.

By this time some of our support was coming from the Information Processing Techniques Office (IPTO) of ARPA and soon we were visited by the director of that office, our old friend from Lincoln, Ivan Sutherland, together with his deputy, Bob Taylor (who would soon become the director). Ivan was by far the most astute technical critic we had had and we were anxious to see what his reaction would be. He quickly grasped our explanations and promptly sat down with a large sheet of blank paper and set about designing a small computer using our new Macromodular building blocks. We watched with delight as he filled it in and as he was putting on the finishing touches he said, "By Golly, fellas, I think you've done it!" Coming from Ivan, this was a twenty-one gun salute and we were thrilled. I like to think that this may have been Ivan's first serious brush with asynchronous systems, something that would increasingly

occupy his attention as the years rolled by—and Charlie's too, for the rest of his life.

But there was one problem that we seemed unable to solve: how to join a system built out of these units with some other system, built either of similar units or of synchronous (clocked) devices. In the course of working on this problem, we stumbled[26] onto a fundamental conundrum that would occupy both theorists and practical designers for some time to come. It came to be known as "the synchronizer problem" or "the glitch problem." The problem was enunciated in simple terms in the literature in a joint paper by Mackie, Tom Chaney (another member of the lab) and me entitled "Beware the Synchronizer."[27]

A rough description is included in Appendix 1, but what is perhaps more interesting is how difficult it was to convince others that the problem even existed. By the mid-1960s, many logic designers (I among them) had, to varying degrees, lost contact with whatever electrical engineering roots they might have once had. Many had come to believe that the units with which they were working were ideal "black boxes" that always obeyed a relatively simple set of rules, typically specified by the unit's manufacturer. Unfortunately those rules embodied tacit assumptions, which in some perverse cases, could actually be violated. Such situations were exceedingly rare in most computer

[26] Wes reports that it was I who insistently pressed the question "But what if they *do* arrive simultaneously!?"

[27] Mackie and Tom wrote a more complete paper at about the same time: W. M. Littlefield, and T. Chaney, "The Glitch Phenomenon", Technical Memorandum No. 9, Computer Systems Laboratory of Washington University, St. Louis, 1966.

systems and many designers argued that they simply couldn't occur at all. But careful analysis led us to conclude not only that they *could* but that, under the right circumstances, they *must*. To resolve the matter, Tom designed some very sensitive experimental apparatus that enabled him to focus narrowly on the situation that would provoke the anomalous behavior and then observe its occurrence. And sure enough, it showed up just as predicted, right there on Tom's oscilloscope.

During 1965-66, much of our time was devoted to supporting the LINC. DEC had decided to manufacture and sell LINCs and part of our contract with NIH required us to document the machine so that anyone who wanted to could obtain a kit and build one. After all, it had been built with taxpayer money; the knowledge should be available to the public. The documentation effort was substantial. Every feature of the machine was depicted and explained in gory detail in a giant series of documents. Others besides DEC intended to build the machines as well. One company (Spear Electronics) decided to redesign it using the then new emitter-coupled logic (ECL) circuits that would make it faster, and shortly they did precisely that.

DEC's interest in the LINC was fading (during 1964 DEC had made some 50 or 60 "classic" LINCs) as PDP-8 demand increased, so to keep their interest alive, Wes persuaded them to combine the LINC and their PDP-8 into a single machine that would be called a LINC-8. Wes was providing consulting help in merging the two designs and he sub-contracted a piece of the work to me. Much of the logic of the LINC tape commands could be handled by a PDP-8 program, and Wes gave me the job of writing that program. Some hundred and fifty LINC-8s were built but later DEC redesigned it as the PDP-12 which, for a number

of years, was a truly popular machine (over a thousand were sold) that ultimately embodied most of the world's LINCs.

In addition to the documentation effort, we also helped a number of people, who had obtained separate funding outside of the LINC Evaluation Program, to procure and assemble their LINCs. And we continued supporting the various LINC users around the country who were struggling to incorporate the computer into their research. This involved organizing a number of meetings in which the users could share experiences, programs, etc. In addition, many of us traveled a good deal during this period, working with and helping the researchers with their varied tasks in their own labs. The diversity of applications was striking[28] and we had to become conversant with bits and pieces of wholly unfamiliar disciplines in order to be useful. I myself spent time in half a dozen different labs, consulting on the design of experiments and helping to connect equipment and write suitable programs. I recall one in particular that stands out in my memory as by far the most dramatic.

One of the participants in the program had been a neurosurgeon and neurophysiological researcher at Washington University by the name of Sidney Goldring. He intended to use his LINC for mapping response areas on the surface of the human brain. Not only is this important as a general matter of understanding the structure of the brain, but in addition he hoped to be able to determine

[28] A more complete listing can be found in the ACM Conference Proceedings, *History of Personal Workstations*, W. A. Clark, *The LINC Was Early and Small*, Palo Alto, CA, 1986.

individual response areas in the course of brain surgery in order to be able to avoid damaging critical areas. I learned that brain surgery is at the same time both more primitive and more sophisticated than I had imagined.

The LINC was located in the amphitheater directly above and overlooking the operating table. Thus while working with the machine we were able to watch the proceedings in detail. For a lay person it was a frightening experience to see saws and drills at work on the skull of a live human being. Even more astonishing was the fact that these patients were often conscious during the operation. Although we were given to understand that they were not suffering significant pain as a consequence of the proceedings, it was difficult to believe that, and at one point, as an operation was commencing, I heard Howard Lewis, who had accompanied me on this visit, murmur "That's a brave man down there."

This was a preliminary phase of the study, and we were helping to get the equipment and the programs in shape. An array of electrodes were placed on the surface of the exposed brain and various stimulae (light flashes, sound, touch, etc.) were presented to the patient. The signals from the electrodes were fed to the LINC's analog inputs whence they were sampled, recorded, and displayed. By moving the electrode array to different locations on the brain surface it was possible to locate the areas of response to the various sorts of stimulae.

That, at least, is the general picture I retain of what was happening. Not surprisingly these people, who had never encountered a computer before, had the program in something of a tangle, but it was a simple matter to show them how to put it all straight and leave them with programs that did precisely what they wanted and that

they more or less understood. One morning Dr. Goldring arrived looking deeply troubled and announced that he had spent a sleepless night having come to the realization that the research, which he had previously assumed would occupy his attention for the next ten years, would now, in all probability, be completed within a year. He was wondering what he would do for the remainder of the decade!

Fortunately not all of the experiments were quite so dramatic and some even produced amusing anecdotes. In one operant conditioning experiment a rat was being trained (I don't recall why) to wait for a fixed interval between successive pressings of a lever. Properly spaced pushes would produce reward in the form of food. If the lever was pressed too rapidly or too infrequently, no food would result. The rat seemed not to understand the situation and initially pressed the lever quite randomly, only occasionally producing food. Then at one point his tail apparently itched and after a successful pressing, he turned and chewed on his tail for a moment to relieve the itch. Turning back he again pressed the lever and, as this had produced the proper interval, food appeared. The researcher claimed that you could see the rat momentarily stop and ponder the situation—and then quickly turn and bite his tail again and return to press the lever once more He had decided that this action—biting his tail between pressings—was what produced food. I learned that such "superstitious behavior" is not uncommon in training animals. (Humans too, it would appear, are not immune from such confusion between cause and effect.)

In the years that followed, the LINC would have the effect of speeding up research in many areas of biological exploration. The real-time, interactive processing of

experimental data was absolutely revolutionary and changed both the face and the pace of biological research forever. The LINC furthermore provided much of the impetus for further work in computers in medicine that took place under the leadership of Clark, Molnar and Cox at Washington University's computer laboratories long after I had left. This includes such things as advances in cardiac monitoring and the development of the PET scanner.

Today there is hardly any piece of medical equipment that doesn't incorporate at least one computer in its inner workings, and doctors, medical technicians, and even patients now take for granted the ability of instruments not only to measure, but to analyze and present data in graphical and pictorial form. The LINC was the forerunner of such equipment, and as such it remains one of the high points of my own evolution as a computer scientist.

Chapter 12

Charlie to the rescue; we take a LINC to Chile and climb down an elevator shaft. Water juice

About a year after the rest of us had arrived in St. Louis, Charlie finished his work at MIT and shortly thereafter came to join us. He had, of course, visited occasionally in the interim, but now he arrived to take up a full-time position at Washington University and to work directly with our group. Everyone was, of course, immensely pleased, perhaps Wes most of all as he had come to depend heavily on Charlie's insights and abilities, and the two of them had developed a uniquely close working relationship. We had sorely missed Charlie's full participation and now welcomed his critique of all we'd been doing. He immediately zeroed in on the fundamental problems and implications of the Macromodular work we had been doing. In years to come he was to carry his ideas about such self-timed circuits to lengths none of us could then foresee. In his final years he came to California to work with Ivan, as the two of them had become convinced that self-timed circuits provided a possible key to faster computers for the future. As I write this, Charlie has been dead for over three years, but that question is still under investigation by Ivan and his associates.

One day in the spring of 1966, Wes came to me and asked how I would like to join Charlie in taking a LINC to Santiago, Chile, to be used in conjunction with

presentations that were to be made there at a meeting of the International Brain Research Organization (IBRO). I was delighted at the opportunity; IBRO had all the right connotations so far as I was concerned—International and Brain Research. What could be better? I learned that a giant squid that lives off the coast of Chile has an extraordinarily large nerve that lends itself nicely to experiment, and this helped to turn Santiago into a center of neurophysiological research. A number of the presenters at the meeting would be from the U.S. and some of these had incorporated LINCs into their research so integrally that in order to demonstrate their work at the meeting a machine was needed on the premises. It was our job to provide it.

Prior to the LINC, a computer was not something that you just unplugged and carted off to a new location. Or if you did, you were almost certain to be in for a considerable stretch of reincarnating it in its new home. Computers, aside from their size and the fact that they tended to be built into their living quarters, were fragile affairs and moving them around was not generally considered feasible. However, we had already had considerable experience moving LINCs on special moving trucks, and even once in the back of a rented station-wagon. We had found, to our great delight, that most of the time you just cabled the pieces back together (which took perhaps fifteen minutes), plugged it in, turned it on, and by gum, the bloody thing simply worked. This doesn't seem like much of a miracle today, but in those days it was new and stunning.

Taking a LINC to Santiago, nonetheless, was a considerable challenge. Among other things, before we left a machine had to be prepared to operate on the 50-cycle power that would be available in Santiago. Charlie, who understood such matters, added the necessary extra pieces

and jury-rigged them temporarily inside the machine. The requisite export papers were obtained and a few weeks later we were on our way with the LINC on board. At that time, neither Charlie nor I had traveled much outside of the U.S. and the trip turned into an epic adventure for us both.

It started out literally with a bang. Shortly after taking off from Miami, the plane suddenly dropped precipitously. There were shrieks as heads momentarily appeared, prarie-dog like, above the seat backs and then settled back down as we abruptly bottomed out. Charlie turned to me with a concerned look on his face and said, "Are you thinking what I'm thinking?" It turned out I wasn't. I was hoping we weren't going to end up in the drink; Charlie, on the other hand, was wondering what the impact had done to the heavy equipment he had rigged inside the LINC. Had it come loose and slammed into the logic boards?

As it turned out, he needn't have worried. When we got to Santiago next morning, the main cabinet (containing his modifications) was nowhere to be seen. Six items had been sent and six containers were sitting there, but the main cabinet wasn't among them. Amidst torrents of Spanish, it eventually became clear what had happened. Six items had gone into U.S. export control in Miami and six had come out the other end, but one of the items that had entered as two boxes bound together emerged as two separate items. Six items went onto the plane and not surprisingly it was the heaviest one that was left behind. As Jim Morris was to remark many years later, "You're always one off in this business."[29] The next day we put through a call to Wes for

[29] Having a count off by one used to be a common programming error.

help, and then waited while the official machinery slowly ground its teeth on the problem.

The Chileans were an extremely hospitable lot and, unlike we compulsive Americans, not given to worrying. The consequence was that over the ensuing week, as we waited for the missing cabinet to arrive, we attended numerous parties and events (including a memorable formal audience with the local Monsignor who must have wondered what, in the name of the Almighty, a computer might be). Our hosts introduced us to the wonders of Pisco Sours, we sampled the excellent Chilean wines, and stuffed ourselves with cherimoyas in orange juice. Between parties, we leaned out of the window of our hotel room, gazing at the frenetic activity in the street below—people scrambling onto overcrowded busses, the outermost clinging only to other, slightly more inboard, passengers. One day, riding a bus, I noticed Charlie flinch as he peered forward. Turning, I found that we were about to collide with the car in front of us. To our amazement we not only collided with it—we kept right on going. The car had broken down and the bus driver was giving it a lift, pushing it fearlessly through the dense downtown traffic.

Finally the LINC arrived, having been stranded on the Argentine frontier, waiting for a "runway incident" (read earthquake damage) to be repaired. On the way to the airport we stopped at a "restaurant" that turned out (to the chagrin of our host) to be a front for a brothel, and at the airport we watched, horrified, as the LINC came out of the plane's cargo door upside-down with the temporary power supplies once more menacing the logic boards. Eventually we got it to the room where the meeting was to take place, and where the other pieces had been waiting. Finally we could go to work putting the machine back together and,

left alone, we set about it. To our delight, in short order we had a working machine. As we set out to celebrate, however, we discovered that we had no way to lock the door behind us—we'd not been left a key and the cavernous building we were in was utterly deserted and vulnerable to the all-too-frequent theft. Undaunted, Charlie took the door lock apart and discovered that, with one of us on either side, we could reassemble the lock in such a way that it locked the door. Fine, but how would the one on the inside then get out? I think it was Charlie who spotted the elevator shaft. It was tiny and clearly intended for a one-person elevator, which hadn't yet been installed. It was a perfect escape hatch for a rock-climber, however, and after we put the door together, I shinnied down the two floors into the basement. After stumbling around in the dark for some time, with the aid of Charlie's shouts I found my way out and we went off to celebrate. Euphoric and left to our own devices, we threw caution to the winds with disastrous digestive consequences.

A short while later other participants started to arrive from around the world and the meeting took on a more routine aspect. The contrast between the high-pressure, high-tech environment we had left behind in the U.S. and the world we were plunged into in Santiago was truly mind-boggling. To me the Chilean people seemed wonderfully human and I reveled in their warmth and irrepressible humor. (They explained that in my ruptured Spanish I had been asking not for a *glass of water* as I'd thought, but rather for *water juice*. "Ah, you Americans," they said, "you always want the very essence of the thing.") On a more serious level, among many of our confreres I found far more liberal social and political views than I was accustomed to encountering in the U.S., and I felt great

empathy with their attitudes and concerns. When eventually it came time to depart, it was with great ambivalence that I climbed aboard the northbound plane. The LINC had proved to be the star of the show and within a few years more of them would begin to appear in South American laboratories.

Days later in St. Louis, sitting in a traffic jam and looking out over the sea of cars, each with but a single occupant, probably most of them like me fuming at the traffic snarl, I found myself wondering which society was truly more sensible—more humane. In the years that were to follow, I would find myself pondering this sort of question many times.

Chapter 13

Macromodules work; St. Louis is hot; we build the Chasm, and I depart for Boston

After our return from Chile, we resumed work on Macromodules. By this time, with Mish pressing ahead in our absence, the designs had been reduced to circuit diagrams, and not long afterwards sample modules were ready in the lab. These prototypes were substantially larger than they would ultimately be, but here, finally, were physical manifestations of our many months of conception and design. On the day we first pushed the button activating a simple sequence of actions between modules, I broke out a bottle of wine that I had brought back from Chile for the specific purpose of celebrating the occasion. Despite warnings that good Chilean wine "doesn't travel," I had fetched along a bottle of the very best stuff. To our great disappointment we discovered that the warning had been well-founded. Nonetheless, the first Macromodules performed as expected, and the wine misfortune was soon forgotten amidst the jubilation.

It was a time when the Viet Nam war was beginning to show signs of serious unpopularity. Mackie and I both felt that the war was a terrible mistake and began participating in a daily noontime silent vigil on the campus. The gatherings were small at first and people passing by wondered what we were doing, standing there in a circle looking so glum. But we were profoundly distressed by what our government was doing and felt the need to

express our sentiments in some way. So, for a change, we dressed in suits in order to indicate that at least some substantial citizens also disapproved of the war. We stood there wondering what else one could do.

In the summer of 1967 I took a lengthy vacation with my family. Just before we left St. Louis, a heat wave struck producing numerous power outages and brownouts as air conditioning everywhere was turned up full tilt. Without power and air-conditioning it was impossible to work, so we packed the car and fled westward, never stopping until we arrived in the cool of the Rockies. We swept through the west, visiting old climbing haunts in Canada and ending up in San Francisco to visit friends in Berkeley. The three years I'd agreed to stay in St. Louis were drawing to a close and as I began looking for something to do next, the Bay Area seemed appealing. However, it was 1967 and Silicon Valley and all of its job opportunities still lay a bit in the future. I stopped in at one or two commercial computing outfits, but with no letters of introduction. They'd never heard of a LINC or a Macromodule and I wasn't even an official electrical engineer. What use could I possibly be?

Knowing that I was planning to leave St. Louis soon, Wes kindly set about helping me find something fun and interesting to do. I'd had a good time in South America and liked to travel. A job running a computer center in Nigeria came up, but that sounded too full of the wrong kinds of challenge and I demurred. By this time Ivan Sutherland was teaching at Harvard, and Wes sent me off with a good strong recommendation to talk to him. Harvard was looking for someone to teach an introductory computing course that would lay the foundations of both hardware and software. Knowing of my keen interest in music and computers, Ivan put me together with a graduate student

and others with similar interests. He made me a very tempting offer that seemed to be everything I'd wanted, and to this day I'm not sure why I didn't jump at it. Instead I chose to go to work once again with my old friend Frank Heart who, by now, was in charge of a group of people and a number of projects at the consulting firm of Bolt Beranek and Newman in Cambridge, Massachusetts. And once again, although there was no way to foresee it, I was moving in a direction that would soon carry me into the next major computer wave—networking.

At about this time Charlie decided to build a combined LINC plus Macromodule system for modeling neuron behavior (specifically the spike output behavior of a neuron in response to specifiable distributions of inputs). In addition to allowing one to study the behavior of the neuron model, the system would demonstrate the power of combining the flexibility of a general purpose computer with special hardware (the Macromodular part of the system) which speeded up the most frequently repeated parts of the operation. Another researcher, Antharvedi Anné, and I assisted with the design. When it was finished we needed a name for it, but anything truly descriptive was hopelessly lengthy. I proposed that we call it simply CHASM, standing for "CHarlie's, Antharvedi's, and Severo's Machine," and in looking back through the literature from that era, sure enough I find a paper bearing that name boldly in its title (with no explanation of its derivation).

As the time to leave St. Louis drew near, I wondered more and more what I was doing and why I was leaving. We had worked together so closely and so intensively for so long that the idea of separating was painful. And besides, the promise of the Macromodule project, on which I'd

worked so hard, seemed large at that juncture. As history was to unfold, the Macromodular approach would eventually be overtaken by advances in microcircuits and microcomputers. The cost/performance ratio of computers was about to plummet and tiny, virtually throw-away microcomputers would obviate the need for building, even temporarily, most special purpose equipment. Ironically we were hoist on our own petard: Macromodules were simply too expensive to compete with the forthcoming miniaturization. However, the core of the approach—self-timed systems—was to survive and to attract the attention of advanced designers for many years. Today it is alive and well and may yet provide a fundamentally new approach to the design of future machines.

But at the time, none of this future was obvious and it was with terribly mixed emotions that in the fall of 1967 we drove across the bridge through East St. Louis heading once more for New England.

Chapter 14

I join BBN; some comments on ARPA; Encounters with TENEX; some coffee-tables, and a teletype through the wall. The ARPANET is born

Bolt Beranek and Newman (BBN) was a somewhat unusual firm: it had many ties with MIT and liked to think of itself as half-way between a company and a research institute of some sort. It had been founded in the 1940s by three MIT professors in architectural acoustics who had given the company its name, but by this time it had grown considerably and had strayed into computers as well as other fields only remotely related to acoustics.[30] J.C.R. Licklider, who became a grand old man of computer science, had worked there on an early experimental Time Sharing system on a DEC PDP-1 computer. (The PDP-1 was the first full-blown computer that DEC made and BBN's was the very first of the series. Ben Gurley, a former TX-2 engineer, had designed the PDP-1 with some oversight from Wes Clark.) By the time I arrived on the scene, BBN was running a commercial Time Sharing system

[30] One wit quipped that the only reason BBN didn't run a brothel in Cambridge was that they didn't have the appropriate talent. This was unduly harsh, but after years of academic association, BBN did seem comparatively "commercial" to me. *Time-sheets*, for heavens sake!

(TELENET) and had two computer divisions. The one I was joining was headed by Frank Heart, and its mission was to design and build a wide variety of computer systems for clients. The other, headed by Jerry Elkind, was more explicitly research-oriented and, among other things, was developing experimental Time Sharing systems supported by government money.

In computer research, government money had increasingly come to mean ARPA money, because ARPA had become the agency that provided by far the largest share of government funding for computer research.[31] Initially hired by Jack Ruina, then director of ARPA, J.C.R. Licklider had set up an Information Processing Techniques Office (IPTO) within ARPA to oversee and fund research in the burgeoning computer field. The National Science Foundation and the National Institutes of Health continued some support, but far below the level of the funds available to ARPA through the Defense Department's always ample budget.

Of all the money that the Defense Department spent, that which was funneled through the ARPA office (as IPTO was known within the computer community) was unquestionably some of the very most productive. It fostered much of the computer research and development that took place in this country during the middle-ages. Much of that work took place at universities, but a few places with close connections to Universities such as BBN

[31] As mentioned earlier, some of the support for our Macromodule research had come from ARPA.

and the Stanford Research Institute (later SRI[32]) which were centers of excellence outside of a university, also gained ARPA support for computer research.

I was personally always unhappy about this situation. It was part of a more general concern that so much of government-funded research was focused on military matters and funded through the Department of Defense (DOD). I felt that this distorted national research priorities and left far too little for other, non-military concerns. However, although DOD of course hoped that military applications would eventually be found for the products of the research ARPA funded, the money from IPTO was largely dedicated to very general, quite unspecific computer research. When developing tools as broadly applicable as a "general purpose computer," it is impossible to set limits on what one might be used for. In any case, during the mid-1960s it was nearly impossible to do significant computer research anywhere in the U.S. (except perhaps at IBM) without working indirectly for ARPA one way or another.

As first head of IPTO, Licklider had established a standard of excellence that was to persist for many years beyond his own tenure. The people who followed him—Ivan Sutherland, Bob Taylor, Larry Roberts, Bob Kahn—were outstanding individuals who dedicated a few years of their lives to overseeing federally-sponsored research in the field. Of course it was an important and powerful position,

[32] At that time, SRI was a part of Stanford University. It was changed into a separate nonprofit corporation sometime in the 1970s, which was when its official name was shortened to "SRI." That happened during the Vietnam War as a result of student activism that banished all classified research from the University.

but I also came to realize that the job entailed elements of personal and professional sacrifice in that these individuals could have had (and eventually did have) jobs with many more emoluments outside of government, perhaps even as ARPA-funded researchers. I remember riding to the airport one day with Larry Roberts when he was head of the office, and discovering that as a government employee he was constrained to use the cheapest available car rental, which meant that we had to be ferried from some distant rental site to the airport.

It may seem strange that I ended up in what was nominally the less research-oriented division of BBN. This had primarily to do with my long-standing friendship with Frank, but it also had to do with the fact that the other division was focused heavily on research in Time Sharing systems. This Big Deal concept was anathema to the Little Deal "Gospel according to St. Louis," that I arrived bearing. But beyond these factors, it's also incorrect to characterize the construction of dedicated computer systems as non-research. Designing and building the kinds of systems Frank's division was working on involved plenty of research. It's just that in addition to the research, there were commitments to finished systems, hard deadlines, etc. It seems that often one person's research is another person's routine engineering. In any case, the term "research," (like terms such as "executive" and "mansion"), has become so over-used that it has lost much of its original pith.

Although in Frank's division a significant effort was under way in medical computing, it was of a very different sort from the direct, real-time laboratory use in which I'd been involved. I also wanted to explore a totally new application area and so I began working with Wally Feuerzeig who, together with Seymour Papert of MIT, was

researching applications in education. Wally and Seymore had designed the special computer language LOGO, which they felt would be easy and instructive for children to learn. I began looking for ways to design a machine that directly implemented that language.

I had also agreed to teach the computer design and programming course that Harvard had wanted, so it was arranged that I would be hired through BBN to teach a graduate seminar at Harvard. This would enable me to pull the material together for an undergraduate version of the course in ensuing years and would provide me with teaching assistants. Not surprisingly, since this was the first time I'd taught such a course, preparation took a good deal of my time and perhaps for that reason I failed to make progress with the LOGO project and eventually became discouraged about my ability to make a contribution to it. I also came to realize that I didn't share my colleagues' deep conviction that computers could revolutionize education. So my enthusiasm waned and I began to think that perhaps after all I really belonged in the other BBN division which was developing hardware, something I by now understood. So I moved over into Jerry Elkind's division and began to learn more about Time Sharing.

There was plenty to learn and plenty of bright new people to get to know. Danny Bobrow, like me, was teaching a course at Harvard and at that time I met Ray Tomlinson, who would later become famous as the originator of the @ sign for email addresses (totally swamping the far more significant fact that he had extended the intra-site message exchange system into an inter-site system, thereby creating the first primitive email system—not to mention his more fundamental contributions). Ed Fiala, who would turn up again later at Xerox PARC, was,

as always, deeply buried in the most esoteric complexities of the system software. Perhaps the most memorable figure of all was Dan Murphy who, because of his last name, was blamed for virtually everything that went wrong. Our boss, Jerry Elkind, would later hire me to work in the Computer Science Lab at Xerox PARC in California.

The group was preparing to shift from an SDS-940 system to a DEC PDP-10. That was an upgrade to a more powerful machine and the BBN engineers had designed special hardware to make the PDP-10 more amenable to Time Sharing. The new system, consisting of hardware and software felicitously intertwined, was referred to as TENEX and would become a de facto standard at many ARPA-supported sites over ensuing years.[33] One of the most important features of TENEX was a virtual memory system that allowed programmers to believe they had more high-speed memory than actually existed (by surreptitiously swapping information between high-speed memory and disks). And of course the system allowed multiple programs to share the machine and run "at the same time." These features (virtual memory and multiprocessing) are a standard part of virtually all computers today. I received my first introduction to these concepts from Jerry Burchfiel, a senior member of the TENEX group.

A substantial number of research projects at BBN depended on this Time Sharing system and some of them involved real-time processing. As I've indicated, real-time processing doesn't mix well with Time Sharing since such processing demands the attention of the computer in reaction to events beyond the computer's control. The

[33] TENEX was eventually licensed to DEC as TOPS-20

computer must be ready to jump up and salute whenever it is needed. But under Time Sharing, the computer's attention is directed to the needs of the multiple users sitting at their terminals. There is an inherent conflict in trying to serve these two demands simultaneously. Nonetheless, plans were afoot to design a special hybrid processor that would be attached to TENEX, enabling it to provide some level of real-time processing.

I tried to put my attention on this undertaking but ultimately failed, because—as in Don O'Brien's earlier nightmare—I felt that indeed we were trying to attach wings to a steamroller. When I questioned a colleague about the wisdom of the whole approach, to my dismay he simply quoted "Ours not to question why; Ours but to do or die." This was a long way from the sort of response I was accustomed to, and I wondered what sort of outfit I'd fallen in with. But then a consultant, Chuck Seitz, appeared on the scene and we quickly found that we had a good deal of common ground and shared many of the same attitudes. Among other things, he was teaching a course at MIT similar to the one I was teaching at Harvard and we showed one another what we were up to.

A large disk system was to be added to the PDP-10. Today the capacities of disks are measured in megabytes, or, increasingly, in gigabytes. Back then, disks were measured in feet. We're talking about a set of disks roughly four feet in diameter. As the disks were being installed, they filled the halls near the machine room and then gradually disappeared, one by one into its maw. In order to be able to get at information quickly, they turned at alarming rates, and in doing so, they swept a film of air along on their surfaces. In order to allow the bits of information to be closely spaced so that lots of them could be crammed in, the

heads that read and wrote the disks needed to be very close to the disk surface. As the heads moved in close, the film of air offered substantial resistance. (Recall that your automobile rides on the air in your tires and that metal and rocks can burn up upon reentering the atmosphere; air becomes a material to reckon with when sufficiently compressed.) The heads thus had to be forced toward the disks to overcome the air resistance.

Perhaps you're wondering, "But what happened when the disks slowed down and the resistance of the air film decreased?" Excellent question. The designers had considered this matter and in such a situation—when the power failed, for example—a safety feature instantly retracted the heads from the disks. The day arrived when the disks were working and everything had been tested—everything, that is, except this safety retraction mechanism. Alas when the power switch was shut off for testing, there ensued a frightful screeching sound as the heads failed to retract and instead ground their way into the surfaces of the disks. Shortly thereafter the halls were once again filled with disks as replacements began to arrive and the damaged ones were converted to coffee tables or discarded. The safety feature was redesigned and after a few weeks, things were once again ready for action.

The question then arose whether or not to test the redesigned mechanism. After considerable debate it was decided NOT to test it. Power failures were infrequent and should one occur, we would know soon enough if the mechanism worked. In such contests between man and the perversity of nature, man is invariably the loser. Within days a power failure occurred and once again the halls were filled with disks. But engineers are a persistent lot, and in

the fullness of time the disks were tamed and became part of the working TENEX system.

One day, for a reason I can no longer recall, I needed to talk to John Barnaby, one of the TENEX programmers. When I entered his office, I couldn't help noticing a huge, ragged hole in the wall by his desk. I could see right through it into the neighboring office, but somehow I knew that it would be unwise to ask how it had come about. As we talked, that hole stared at us like a pink elephant whose presence everyone has tacitly agreed to ignore. Later I learned that, despite a mild manner, this was a fellow who could become quite distressed if he encountered trouble with his program or the system on which it was running. I can't say for sure which of these problems had arisen, but I did discover that on a night not long before my visit, something had infuriated him to the point that he (a largish fellow) had picked up his roughly fifty-pound model-33 Teletype (the terminal of choice at the time) and hurled it bodily through the wall. Programmers of his caliber were hard to come by, so this minor infraction was overlooked. The hole remained for some time and I thought of it as vivid testimony to the frustrations induced by over-stressed Time Sharing.[34]

Although people raised an eyebrow at such behavior, many understood it full well and secretly sympathized with

[34] Some years later my wife Laura became a victim of a Time-Sharing system known as TENEX, the computing environment with which she was then interacting via a model 33 teletype. She is a skilled knitter and in between responses from TENEX she had ample time to beaver away at an elaborate sweater containing thousands of very fine stitches. When it was finished she dubbed it her "TENEX Memorial Sweater" since most of it had been knitted while waiting between responses from TENEX.

and even admired such a forthright response. I've often thought that if more people responded as Barnaby did (or the guy who shot the 1401 full of holes), expressing their frustration with similarly overt action, perhaps the computer industry would be forced to shape up and give us less defective products. As it is, we may never know how many personal computers continue to be secretly abused by their frustrated users.

I realized I didn't belong in this Time Sharing division—I'd settled among disciples of the wrong religion. I was becoming quite discouraged by this point and felt that perhaps after all I'd made a serious mistake in turning down Ivan's offer at Harvard. I'd found no place at BBN to apply my skills constructively, but when I talked to some of my colleagues at Harvard their lives sounded differently bad as they described their struggles to avoid committee work and obtain funding. I was having a serious talk with Frank about the situation, when he pushed across the desk a document that had just arrived from Washington. He said it was a request for proposals (RFP in the lingo) for building some sort of network of computers and suggested that I take it home and look it over. With very little enthusiasm I put it in my briefcase to read that evening. I didn't understand all of the details, but I got the general drift and decided that what it described was a relatively straightforward, if not simple, engineering job. The next morning I went into Frank's office and, putting the document on his desk, told him I felt that we could certainly build it, but that I couldn't imagine why anyone would want such a thing. The network the document described was to become known as the ARPANET, forerunner of today's Internet.

There is no question that, in retrospect, my initial sentiment ("Who would want such a thing?") seems ludicrous. However, hindsight is far easier than foresight, and the request for proposal that came from ARPA made no mention of e-mail or the World Wide Web.[35] These things, which actually caused networking to "take off," were to come later, more or less as afterthoughts. Instead, at the outset, there was talk of eliminating duplication and fostering "Resource Sharing" —the sharing of programs, results, and access to computers among workers at the various sites, mostly universities, that ARPA was supporting. But these things seemed difficult to believe in, given the diversity of machines, interests, and capabilities at the various sites. Although it seemed clear that with suitable effort we could interconnect the machines so that information could flow between them, the amount of work that would then be required to turn that basic capacity into features useful to individuals at remote sites seemed overwhelming—as, indeed, it proved to be. A long road and many years of difficult, ground-breaking work lay between the interconnection of the first few ARPANET sites and the world we now inhabit in which a computer that can't access and utilize the Internet would be deemed virtually useless.

As originally conceived, the network was to consist of the large host computers at each site "talking" directly to one another. It was Wes Clark who, one day riding in a car with Larry Roberts (and others), urged the innovation

[35] Amazingly, today there exist almost no copies of the original RFP from IPTO. I take this as some indication that I was not alone in failing to grasp the historic significance of what was happening.

wherein the network traffic handling was off-loaded from the main ("host") computers at each site to a small auxiliary computer. This small computer was dubbed an Interface Message Processor or IMP. It is probably not overstating the case to say that this suggestion was critical to the success of the entire network project. In addition to relieving the host of the work of handling network traffic for other sites, it had the advantage that each host had only to deal with its IMP in a standard fashion, rather than having to interact with the different types of computers at all the sites to which it was connected. Eventually, of course, since the IMPs really only passed the bits along blindly, the host programs *did* have to deal with and understand one another. Standard host-to-host protocols, by which the hosts communicated with one another, would be defined and refined over the coming years. Think, for instance, of the simple protocol whereby we all say "Hello" and "Good-bye" when we use the telephone—a device which, like an IMP, simply passes the data (our voices) along without understanding anything we say. If someone violates these conventions, it causes at least momentary confusion.

Anyway, here at last was something I could get my teeth into—a challenging engineering problem. Although I didn't know where it would lead, it didn't seem like a bad idea, so back I came to Frank's division as we began to put together a proposal in answer to ARPA's request. Before we could write a sensible proposal, we felt we needed to understand in detail how what we were proposing would work, and that meant that we needed to do a pretty thorough system design. It soon became evident that some very large corporations and defense contractors were also interested in bidding on this job and we felt that tiny BBN would need to have a really superior proposal if it were to

stand any chance against these giants. We had another problem as well. Several of us knew Larry Roberts from Lincoln days and we knew that he would be concerned about any appearance of favoritism and would therefore be cautious about giving the job to BBN. We therefore had to write a proposal that not only Larry but *anyone* could see was the best of the lot. Fearlessly we plunged in and set about not only figuring out how the system should work but actually proceeding to design it to a level of detail unusual for a proposal.

Our old colleague Will Crowther was still working at Lincoln, but we thought that for a job like this he might be induced to come to work with us. We discussed it with him and even before he had completed the formal transfer, he began acting as an informal consultant in the design. At that time Bob Kahn, who would later become head of the ARPA IPTO office, was working at BBN. He was interested in the potential of the network, had contributed to the error detection scheme contained in the proposal, and wanted to learn some fundamentals of hardware design. So I commenced working with him on the design of the interfaces and other special hardware that would be required on the IMPs. Teaching is an excellent vehicle for coming to grips with and understanding a problem, and we enjoyed working together as the interface designs quickly took shape.

The hardware design was relatively straightforward since it was obvious what was required and the choices were somewhat limited—it was just a matter of being careful and thorough. The more complex design decisions lay in the software, where most of the character and behavior of the IMP, and thus of the network, would be determined. The software team was made up of Dave

Walden, Will Crowther, and Bernie Cosell, with Frank also deeply involved and the rest of us making occasional suggestions and comments. It was just the right level of difficulty and we all enjoyed ourselves enormously, feeling that the design of systems such as this was ideal grist for our mill. We had high regard for one another's abilities and our mutual understanding was such that a great deal of abbreviated language was used in communicating with one another. I was getting to know Dave Walden for the first time and I remember being impressed that such a bright young guy had such good judgment as well. Bernie was already well known to me (and, in fact, to the entire BBN community) as a programming wizard. He was notorious for having inadvertently caused Bob Newman (the "N" of BBN) to imagine briefly that he was typing to a person (Danny Bobrow) at another terminal, when in fact he was actually typing into Bernie's cobbled-together version of Eliza, a simple simulated-psychoanalyst program originally concocted by Joe Weizenbaum at MIT.

As more and more time was spent in preparing the proposal, the "overhead" costs began to pile up. The company's management, feeling that the probability of BBN winning the contract was vanishingly small, was appalled at the amount of money that was being spent in preparing the bid—more money (I believe) than BBN had ever before spent on such a thing. But by the time we finished, we had great confidence in our design. As I recall it did not fully comply with ARPA's RFP; we felt we had found better ways to do a few things. We had already designed all of the special hardware and had actually written the time-critical inner loop of the program, as well as designing the rest of it in some detail.

We, and a small number of other bidders, were individually called to Washington for discussions with Larry. We defended our design decisions with great vigor and Larry quizzed us unmercifully about every detail. As the weeks passed and we remained in the running, our hopes began to rise. Then, to the astonishment of many (probably especially to some of the larger bidders, not to mention the BBN management), we were finally awarded the contract. That evoked a hilarious telegram from Ted Kennedy, congratulating us for winning the contract to build the "Interfaith Message Processor." But aside from the hilarity, we were elated as well as appalled at what lay before us.

Recognizing that we would need to beef up our forces, we shortly hired Will Crowther and began hiring the best students from the course I'd been teaching at Harvard. The first of these was a very bright young fellow by the name of Ben Barker. Shortly after he came on board, Ben joined us for a meeting with AT&T people at their New York headquarters in order to discuss details of their lines and the 50 kilobit modems to which we would be connecting. AT&T was somewhat reluctant about the whole endeavor; they didn't really want novices like us mucking about with their terminal equipment, and moreover felt that the entire enterprise was somewhat silly.[36] But Larry held a heavy governmental sword over their heads so they laid on a rather formal meeting, replete, it turned out, with cigars, candy, and nameplates at each seat. As we entered the

[36] Later, when we complained about interruptions in the lines on the order of a few hundredths of a second, they simply could not comprehend why anyone would possibly care about such a tiny matter.

room, I thought to myself that this must be rather heady stuff for a young student such as Ben, but later, when our plane home was canceled by a snowstorm and we were reduced to riding the train back to Boston, suddenly Ben snapped his fingers and said "I should have thought of it earlier; we could have taken Dad's helicopter." Clearly Ben could take AT&T and their nameplates in stride.

We had selected the Honeywell 516 computer as the basis for the IMP[37] and had presented extensive justification for the choice in our proposal—including the fact that we planned to use a hardened, military version built like a tank. Now it was time to convey to Honeywell all of the additions and modifications to the basic machine that would be necessary to turn it into an IMP. At that point we were not familiar with the particular logic packages that Honeywell used and so the designs that we gave them, though detailed, were in terms of general logic rather than explicit packages. It seemed to us a straightforward task to render the design into Honeywell's own logic modules, but to be sure that there were no mistakes, we spent a number of sessions explaining everything carefully to the designer they had assigned to the project. This was to turn out to be an extremely painful part of the project. The software was under our control; we were doing it ourselves. But for the hardware we were dependent on Honeywell's special

[37] This was based in part on work that Dave Walden and Alex MacKenzie had done earlier evaluating the Honeywell 516 for another project. Honeywell, furthermore, was willing to help BBN bid on the proposal (they were hard-pressed to respond quickly to the several bidders that wanted to use the 516), and quickly agreed to build the special hardware for us. Many computer companies hate doing special hardware, wishing to reserve any available talent for new machines that can be sold in quantity.

systems division and that turned out to be a hotbed of incompetence.

Week after week we struggled to get the Honeywell engineers to understand what was wanted. In the process we became familiar with their logic elements and were thus able to see that one mistake after another was being made. As time passed it became evident that the design was converging too slowly for the schedule and ultimately we were forced to accept a machine that we knew we would have to rework substantially. Ben was a great help in all of this and together we finally managed to get it going. We then gave a set of revisions to Honeywell so that ensuing copies would, we naively presumed, be correct. Alas, we were to learn, it would take many months before all the corrections were finally incorporated. In the meantime we had to make repairs to every machine that arrived. We instituted a plan under which we gave Honeywell test programs and insisted that they run them in their plant in our presence so that we could certify each machine before it was shipped. One day, despite our previous day's refusal to accept a machine, it showed up on a truck at the BBN loading dock. Frank, watching the proceedings through a window, registered disbelief as he saw me refuse delivery, turn the truck around, and send it back. This caused some chaos at the Honeywell plant and, along with some of my more forceful language, apparently garnered sufficient attention that things then began to change.

In the meantime the software crew had been busy and a test version of the program began to operate in the first fully-functioning machine. Since we had only that one machine to work with, for testing purposes the communication interfaces (both to the host and to the modem lines) were shunted so that outgoing channels were

connected directly to incoming channels. We were elated that, despite the troubles with Honeywell, things were going well, but then a strange thing began to happen After many hours or days of continuously successful operation, the machine would suddenly stop in a strange state, never the same way twice. We had what amounts to a system designer's worst nightmare—an extraordinarily rare, seemingly random, intermittent failure. Despite numerous attempts, we were unable to catch it in the act in order to capture some symptoms. Then suddenly Ben or I (or perhaps the two of us together[38]) remembered the synchronizer glitch problem that I'd encountered several years before in a totally different environment. Could this be a manifestation of the glitch problem, and what could we do to verify this suggestion?

There were lots of 516s out in the world serving faithfully in many other settings—that was one of the reasons we had selected it—and no such problems had been reported before. But we also knew that our several high-speed interfaces were driving the machine's input-output section much harder than most other applications. And most other applications didn't require the machine to operate continuously, 24 hours a day, for months on end without a single hiccough. Ben pored over the logic diagrams of the machine and finally, in the part where requests for service by external devices are handled, he found a possible culprit. He designed a clever piece of hardware that aggravated this part of the machine even more intensively than our interfaces did, and lo, it stopped

[38] I had described the problem to my Harvard class as part of the introduction to synchronization.

almost at once. We then excitedly hooked up an oscilloscope so that we could observe what was happening and, dimming the room lights, so that we could see a very faint occasional failure trace amidst all the bright correct ones, we peered at the scope's face. And sure enough, there it was—an occasional failure that Ben's device had made just frequent enough to become visible.

This was one of the best pieces of hardware detective work any of us had ever experienced. We were thrilled and, now having hard evidence in hand, we immediately called Honeywell. Their preliminary reaction was defensive: the troublemakers at BBN were not only complaining about their interfaces and the schedule, but were now questioning the very design of their basic 516! After considerable persuasion we got them to produce the machine's original designer from the back room, the first really competent Honeywell engineer we'd met. He expressed considerable skepticism at first but agreed to come have a look, and after listening carefully to our explanation and peering with us at the evidence on the scope face, he was finally forced to concur. In the meantime, Ben had devised a simple fix to the machine that would cure the problem.[39] We instituted the change and the failure never occurred again. We recommended that Honeywell adopt the change, install it in all future machines, and retrofit existing machines. I believe this happened; we certainly checked all future machines that came to BBN to be sure the fix was installed.

[39] The problem can't be truly *eliminated,* but Ben's fix reduced its probability to the point where its occurrence was measured in hundreds of years rather than hours.

The day approached when we were to deliver the first IMP to UCLA and in short order packers arrived to encase our precious cargo in a sturdy wooden box. Given that the version of the machine Frank had chosen was "hardened" —it had a rugged steel case, and appeared capable of withstanding a direct nuclear hit—this extra packaging seemed a bit redundant, but it provided a place to affix a destination address label. Finally I added numerous arrows saying "This Way Up," and a note that said simply "Do It To It Truett." Frank, in an excess of compulsiveness, had decided that someone from the team should accompany the machine on its journey—I mean *ride with it in the cargo plane!* This proved impossible—the air transport company simply refused—but it was watched carefully onto and off the plane. Truett Thach, a member of the BBN team, was waiting in Los Angeles and shortly more of us joined the festivities when it finally arrived at UCLA.

Given the careful testing the machine had undergone at BBN, including not only the self-test, but also testing with another IMP in the same room through a pair of modems, we were not surprised that the machine worked as soon as it was plugged in and turned on. In fact, since it was shipped with the program installed, I believe it came alive of its own accord when it was plugged in, thanks to a "watchdog timer" we'd provided to reset and restart it automatically should it ever falter. We had met the schedule, something quite unprecedented in the computer field. The UCLA crew had expected that we would spend several days getting the machine working, and in fact had relied on this to allow them time to finish their own preparations of the program and interface for their host machine. Thus they had to scurry some, but shortly the connection was made and the much-touted initial message

was sent through the IMP from one part of the Host to another. Having done a great deal of such testing on our home turf, we were not as impressed with the event as the UCLA people were.

Frank Heart with an early IMP

The initial contract had called for four IMPs, and right on schedule the other IMPs were delivered to their prescribed sites. There was a good deal of reluctance by some of these initial site managers because they viewed the network as an intrusive nuisance that interfered with their routine operations. But once again Larry held a trump card (funding) and they were forced to knuckle under and provide suitable hardware and software interfaces. It would be several years, however, before the fruits of this trouble were to begin paying significant dividends. During those years, the software communities at the ARPA sites would gather their forces and begin to develop standards by which diverse computers could communicate with one another.[40] Given the diversity of the hardware and software systems that would be joined into a single network, this was a complex and daunting undertaking. Steve Crocker, then a graduate-student at UCLA, would become visible on my radar screen as one of the principals in this enterprise and it was some years later that Bob Kahn and Vint Cerf, together with others who have received considerably less credit, would define the discipline that would allow networks to be joined together to form the Internet as we know it today.

By this time my second marriage was beginning to falter and would soon break-up. Throughout this difficult period I continued teaching at Harvard and in the second year we turned my course into an undergraduate one. On opening day nearly 100 students showed up and we were forced to turn quite a few away as we had insufficient lab space and equipment. Fortunately by then I had lined up

[40] The fact that TENEX and the IMPs all came from one place (BBN) facilitated much of the early ARPANET protocols and application work.

some superb teaching fellows (including Ben Barker) and together we went to work. I had a good set of lecture notes from the preceding year and felt much more secure about what I was doing. I began to relax and enjoy the teaching. It was exciting to see understanding dawn and my informal manner shortly led to friendly badinage with the class. We were working together, bringing them on board for an exciting journey. I asked Dave Walden to give a lecture on Time Sharing systems and he did a superb job. At one point he lost the thread of where he was going and simply turned to the class and said, "Come on guys—help me out here." They were a bright bunch and quickly put him back on track. When the time came for the final exam we put together some really intriguing problems and after it was over, as one student was leaving, she turned to me and said "That's the best final exam I ever flunked!"

Other things bound us together as well. It was a time of anti-war protests, and the students had clearly noted the resist button on my jacket and knew that I was on "their side." When I came across some of them gathered in a crowd outside a meeting in which the faculty was considering taking a stand against the war, they hurried me inside. At BBN I felt considerable ambivalence about my work. On the one hand I was strongly against the war but on the other, here I was working hand in glove with a branch of the Department of Defense. I was able to see that the military connection of our work was pretty tenuous, that the implications of what we were doing were extremely general, but nonetheless, under the circumstances I had some level of discomfort drawing my pay indirectly from the Department of Defense. Frank knew this and one day as we departed for a meeting at the Pentagon, I mentioned that I was considering moving my resist pin onto "the

general's" jacket when we all hung up our coats at the commencement of the meeting. Frank wasn't sure whether or not I might actually do such a thing. Neither was I.

Meantime, Ray Tomlinson, a bright colleague from my brief TENEX excursion, noticed that it would be relatively simple to extend the already-extant intra-TENEX inter-person communication feature to permit communication between individuals at TENEX systems at remote sites. Ray inadvertently achieved immortality through his adoption of the "@" sign to indicate a remote network address. More importantly, his "hack" marked the beginning of inter-site email which promptly blossomed into the most widespread use of the network. As email started to become ubiquitous, other mail programs began to show up with improved user interfaces. However, truly "user-friendly" systems would appear only after the switch from teletype-style, terminal-based systems to the sorts of display screens virtually everyone uses today.

As the ARPANET grew and prospered, it shifted imperceptibly from an experimental vehicle into a utility as people began to depend on it more and more. We had, of course, realized from the outset that reliability would be critical and had taken numerous steps to assure that when an individual IMP died, the network as a whole would continue to function as traffic was rerouted around the sick IMP. Right from the outset, features were designed into the IMPS that allowed them to be monitored, reloaded, etc., from what soon became a Network Control Center (NCC) at BBN. Alex McKenzie, a member of the team, was an early and ardent advocate for viewing the network as a utility. Alex, a meticulous fellow with a booming voice as forceful as Frank's but in a lower register, took over management of the Network Control Center and was to find himself

increasingly acting as a buffer between the users of the network and the providers at BBN.

Alex showing the NCC to some Chinese visitors

Because of its special rôle in overseeing the network, there were particular sensitivities at the BBN node. One day it became necessary to install a jumper—a single short piece of wire—between two points on the back-panel of BBN's own IMP. The IMP was busy running in the network at the time and, because of its critical role at the network control center, we decided to ignore the normal discipline, whereby one would turn off a machine before touching the wiring. The place where the jumper needed to be installed was awkwardly close to the base of the machine and as I got down on my hands and knees, with the crew surrounding me shouting suggestions, I peered at the pins to which the

jumper needed to be attached. Finally I reached forward gingerly—and almost immediately touched something I shouldn't have, abruptly shutting off the machine. Chaos ensued as the control center crew erupted into action, kicked us out of the way, and quickly restarted the machine—so quickly, in fact, that it was up and running before we could take advantage of the time to install the jumper while the power was off!

So there we were again, right back where we'd started. One thing was clear—I'd had my chance, and it was now up to someone else. I couldn't believe it when Ben Barker stepped forward and said, "Here, let me." Ben was a very tense fellow and I'd noticed that intense concentration on his part was often accompanied by a shaking hand. Here, I thought, was a catastrophe not only waiting but impatient to happen. Ben crouched down and as I knew it would, his hand commenced shaking. Then, as I watched in disbelief, his hand suddenly stopped shaking as it shot out, shoved the jumper into place, withdrew, and immediately resumed its shaking. To this day I don't know how he managed it.

All of the early sites were Time Sharing systems and users of those systems accessed remote sites on the network through terminals attached to their local "Host" system. The IMPs acted somewhat like the phone company, merely passing information along. The Host computers dealt with the terminals and connected them logically with the desired remote Host. But there were users, not associated with any nearby Host, who wished to access remote sites by connecting directly from their terminals to an IMP and thus to the network. This unanticipated development required an upgrade to the IMP that added a sort of mini-host inside of the IMP through which terminals could speak with remote Hosts. In addition to the mini-Host software, we

needed hardware through which the terminals could be connected to the machine—these terminals were of many different types and speeds. Ben came up with a clever idea for this hardware which accommodated a wide variety of different terminals. While the software team made the necessary additions to the program, we set about designing and building the hardware.

This was an era in which a technique known as "wire-wrap" was the method of choice for building prototype devices. By this time we were using integrated circuits (ICs) of the sort that would now be described as small scale integration. These little bug-like gadgets plugged into sockets on one side of a board. On the other side, the prongs of the sockets extended about half an inch, forming something that looked like a miniature, dense bed of nails. The wires that provided the required interconnections between the ICs were wrapped around these prongs. Special machines (owned and operated by sub-contractors) did the wire-wrapping, but the mechanisms for providing the machines with the information from our drawings regarding precisely *which* sets of points to connect together, was, to say the least, primitive. Today such mechanisms are fully automated, but in those days most wiring lists were made up by hand[41]. Thus one had to contend not only with design errors, but also with clerical errors introduced during the transcription process. This made debugging something of a nightmare and caused harsh thoughts, and sometimes even words, to be exchanged between us and the company that did the wire-wrapping for us.

[41] Some research institutions were then busy developing automated systems but they hadn't yet percolated to places such as BBN.

After the hardware was built, Tony Michel, another member of the growing IMP team, and I worked together in debugging it. As always, there was considerable time pressure and as we worked late night after night, Tony started to fall asleep even as he was running oscilloscopes and moving probes around. He later confessed that I'd pushed him so hard during that period that he had seriously considered quitting. But he didn't, and shortly we had working hardware. The software required major upgrading and the whole thing took a number of months to complete, but eventually we got it all working.

As people began using the Terminal IMPs (called "TIPs"), they were plagued by problems with the telephone connections between their terminals and the TIP. Strictly speaking, these weren't BBN's problems (just as the connection from your computer to your Internet Service Provider isn't part of the Internet), but the users didn't care about that; they viewed their terminals as connected to the remote site and weren't interested in the distinction between the individual line connecting them to the TIP and the further network connections to the remote sites. If any part of it failed to work, BBN got the blame. So we developed a system wherein periodically a computer at BBN dialed each of the terminal ports on every TIP in turn, connecting to and testing each one to be sure that the terminal lines were working properly. These, of course, were just plain old regular phone lines, and sure enough one day one of them seemed to be failing. Unable to figure out what was wrong, the technicians decided to listen to the line as it was dialed and tested. They heard it ring and try to whistle at the TIP (think FAX), but instead of the expected answering signal, to their astonishment, they heard an angry voice shouting "Oh, it's YOU again is it!"

and the receiver was slammed down. Someone had entered a wrong number in a table of TIP phone numbers, and an innocent victim had been receiving the repeated test calls. At that time faxes hadn't yet come into widespread use and the recipient of the calls, never having heard such a thing, assumed it was some prankster repeatedly calling and then whistling into the receiver.

There was considerable interest abroad in networking, particularly in England where much parallel thinking about so-called "packet-switched" networks had taken place. Shortly Donald Davies, who had been a pioneer in thinking about such networks, came by for a visit. Davies might easily have built a network in England before the ARPANET, had he had access to the kind of funding ARPA was able to provide. Interactions with our group began to develop in other parts of Europe as well, and within a year or so we were making trips both to report on our work in various meetings and also to explore with potential customers their need for computer networks. Once again Frank's instinct for seeking and finding new applications was an enormous asset.

I myself think the verdict is by no means in regarding the ultimate benefits and costs to society of computer networking. Obviously it's already had an enormous impact, but things are moving so fast, and the air is presently so full of hype, that it's impossible to say how it will all ultimately settle out. Like so much technology, the final effect will depend on the way society chooses to utilize it. Today the glowing predictions are largely based on optimistic assumptions about how this will work itself out and about potential economic benefits. But there are other impacts as well, and I can envision possibilities that are far more ominous than those presently being touted. I suspect

it would be wise to reserve judgment until the present wave of euphoria settles down and matters clarify themselves as developments unfold over coming years.

Most of us have forgotten the sorts of hype that accompanied the advent of television in its early years. It was hailed then as a great educational vehicle. But against the limited educational programming that actually exists today, one must balance the far more prevalent entertainment programs, which, by and large, have had a dumbing-down effect on society—not to mention the enormous perversion of the political process that has resulted. Perhaps computer networking, because it permits multi-way, rather than just one-to-many communication, will provide a countervailing force, as some are hoping and predicting, but much remains to be seen.

Before I leave the ARPANET (and its offspring, the Internet), I feel obliged to comment on what has happened, in recent years, to its paternity. As its importance has become obvious to everyone (even to me), so-called "fathers" have been cropping up all over the place. It's the same old story of the press identifying and celebrating certain individuals as "the Father of the Internet," whereas in truth the thing came about as a result of the convergence of numerous technical developments and the ideas and energies of a large number of individuals. Although some people were obviously more central and influential than others, trying to point to any one person or a few individuals as responsible for either the end result or the vision is absurd. Nonetheless, regrettably, a number of former colleagues have allowed themselves to be singled out and celebrated as particularly important figures, whereas others, probably even more central, who are by nature more reticent, have received far less attention.

I have come to believe that it was J.C.R. Licklider and his disciple Bob Taylor, who provided the principal impetus for networking in the U.S. Not that they themselves "invented" any part of it. But they foresaw and believed deeply in the tremendously cohesive impact it would have and used their influence to push research and development in the direction of networking. The depth of their conviction and Taylor's extraordinary persistence in pursuing their vision of the computer as primarily a *communication* device, are, to me, absolutely stunning—especially given that neither one was anything like a technical whiz. It may be precisely because they themselves were not directly in the trenches, where the many formidable obstacles were often all that one could see, that they were able to hang on to their vision. But hold to it they did, and my hat is off to them. It's also off to Larry Roberts, the person who not only ran the ARPA office during that period but also had ample technical ability to climb repeatedly into and out of the design trenches during the crucial early years of the network. And likewise to Frank Heart, whose firm hand on the tiller and general paranoia, produced a design that set the kind of high standards for later developments in networking that would enable it to become a new kind of utility for the entire world.

A number of years ago, BBN held a self-congratulatory party to commemorate the twenty-fifth anniversary of the beginnings of the Internet. Forgotten was the horror with which, at the time, the earlier management had viewed the cost of writing the proposal. Instead, now there was much raucous breast-beating. Frank, however, who had spearheaded BBN's initial involvement in networking and had overseen its blossoming, was a model of reserve, saying only, with considerable eloquence, that it was a rare

privilege to have had the opportunity to ride the crest of such an important technological wave. Would that all who have been involved could emulate the reserve such a statement reflects.

Chapter 15

The Aloha Network precedes Ethernet; the first microprocessor appears; we design a true multiprocessor

During the late sixties and early seventies there were annual meetings of a Systems Conference in Honolulu. Each winter people trooped happily to the Hawaiian meetings and some significant papers were presented there. These meetings had been put together by Professor Norm Abramson, a computer scientist at the University of Hawaii, with a view to both bringing his colleagues to his surfing domain and helping to enhance University of Hawaii computing.

The University of Hawaii has a scattered campus with branches on several islands. In order to permit computers on the various islands to communicate with one another, Norm devised what I believe may be the first instance of multi-way electronic communication over a shared medium. Traditionally, when there were only a few devices to be connected together, direct links were provided between each pair of devices. But it was impractical to provide dedicated links between all the pairs of Hawaiian islands; instead radio was the obvious means of communication. Norm's innovation was to share a *single* radio link among *all* the sites as opposed to having separate individual links between each pair.

To understand Norm's proposal, imagine half a dozen people sitting blindfolded in a room so that speech provides the only means of communication among them. Norm's idea was simple enough—whenever anyone has something to say, he first listens. If someone else is speaking, he waits until he hears silence. He then speaks up, says to whom he wishes to speak, and then says his message. A problem obviously arises if two or more people happen to start speaking at once; there will be a conflict and confusion will result. But, being able to hear, a speaker can promptly notice any conflict and desist, wait a bit, listen for silence, and try again. If the delays before retry are varied randomly, then eventually some speaker will get through, whereupon the others will wait for silence before trying again. This, very roughly, was the basis of Norm's scheme for an inter-island computer network which he called the Aloha Network. He demonstrated that it worked, and used it for communication between the various islands.

Later on, as computers proliferated and there was increasing need for groups of machines to intercommunicate locally, others explored further the idea of using a shared medium. At Xerox PARC, Bob Metcalfe and David Boggs used a shared coaxial wire (they dubbed it "Ethernet") to interconnect locally large numbers of computers at very high speed. They introduced numerous refinements into Norm's initial scheme that vastly increased its capability, and eventually the idea of Local Area Networks (LANS) spread through the computer world, creating a major sub-industry. But as far as I know, Norm was the first person to build and demonstrate a working system based on the concept of a shared medium for multi-way electronic communication.

At about this time the first microprocessors made their appearance. As we discussed the implications of this development at BBN, Frank raised the question whether it might not be possible to build a powerful computer by ganging together a number of smaller computers in some way. The IMPs were already limited in the traffic they could handle and it was clear that the way the network was growing much more powerful switching nodes would be required in the not too distant future. But IMPs aside, Frank's question was titillating and had implications across a broad range of other applications that demanded heavy computing power. We started mulling over possibilities.

ILLIAC IV at the University of Illinois was thought to be one of the most powerful computers in the world at that time. It was what is known as an "array processor" and consisted of multiple computers that operated in lock step, simultaneously performing the same operations but on different sets of data. The image that began to emerge in our minds as we discussed possibilities was quite different, however, and consisted of multiple processors, each able to perform any task that came along, and all working cooperatively but independently to do whatever tasks needed doing at the moment. Each machine would work on whatever task needed to be performed next, and in general all the machines were doing different things at any given time. With any machine able to perform any task, the more machines there were, the faster the tasks could be performed and thus the faster the overall job could be handled. This seemed a promising image and as we considered it, we realized that having multiple machines had another advantage as well.

Because reliability had for so long been uppermost in our minds, we began to think of ways that multiple

computers might also be used to check on one another. Over the ensuing months, questions about how to arrange this absorbed much of our attention, and soon high reliability became as important an objective as high speed. The reliability objective raised many fascinating questions. If the ability to excise a troublesome machine existed, how could one avoid the possibility that the trouble-maker itself would do the excising? If the ability to reload a neighbor's program existed when it seemed to be failing, how could one be sure that one's afflicted neighbor wouldn't reload one with its crazy program? Clearly some sort of voting mechanism was called for. Some of the questions that arose bore a striking resemblance to problems that arise in human society and someone craftily suggested that such a machine should be called "The Association of Computing Machines" —a play on the name of the largest and oldest computing society, The Association for Computing Machinery (ACM).

As our ideas began to firm up, we commenced looking around for a suitable computer on which to base our design. There were, in fact, very few choices. As we envisioned it, the computers would need to work very closely together, so closely in fact that they would need to share access to a common main memory. Most computers are designed in such a way that the connection between processor and memory is carefully tuned for high speed and is protected from any external access (other than carefully crafted Input/Output mechanisms) that might interfere with it or slow it down. As we would need to intervene in this processor/memory connection, we needed to have access to it. Our eye was caught by a new machine, the SUE, that was made in unusually modular form by Lockheed Electronics and in which access to this connection was explicitly made externally available. Although we

would have to design a good deal of special hardware to mold the machine to our special purposes, its general structure seemed amenable to the design. We visited Lockheed in Los Angeles to discuss possibilities, but then, before things could get fairly under way, an unexpected interruption occurred which turned my life upside down for a time.

Chapter 16

China!

Nearly a year before, in May of 1971, I had attended a workshop in Aspen, Colorado. The topic was computer kits for education and I found myself among an elite group of computer scientists from around the country. Over drinks one evening I put forward an idea that had occurred to me some time before. How about a group of computer scientists making a visit to China? I love to travel and had lain awake speculating on the most exotic places one could go. China seemed a suitably implausible and inaccessible target in those days. The U.S. had not had diplomatic relations with China for over twenty years, but some loosening of the barriers was in the wind. No one then knew that Nixon was quietly preparing to make the trip himself (it would be announced a couple of months later), but ping-pong diplomacy had commenced. Realizing that formal invitations would be required, I reasoned that a group of distinguished American computer scientists might, just conceivably, tempt the Chinese. So although I was a rather junior member of the group that night, I tentatively raised the question of a possible visit to see what response I might get. To my delight no one laughed and several people expressed real interest.

When I returned to Boston I drew up a formal proposal setting out the parameters of a potential visit: its goals, personnel, timing, estimated costs, possible funders, etc. I circulated the proposal to a group of colleagues for comment and a short while later some of us made an appeal

directly to the board of the computing umbrella society AFIPS (American Federation of Information Processing Societies) for moral and financial support in the unlikely event that the trip might actually take place. I regret to report that we were made to feel small and given to understand that when a computer group ultimately went to China, it would be *they* and not some bunch of professors and mavericks who would go. We quietly withdrew.

A few weeks later I drew up a list of some fifteen of the country's best-known computer scientists. It was an impressive list of names, only a few of whom I knew personally, and as I picked up the phone I wondered what sort of response I'd get. My first call went to Al Perlis. As mentioned earlier, I'd encountered him several years before when he was part of an NIH visiting committee reviewing our work at Washington University. At that time I'd been a junior member sitting quietly in a corner and I was quite sure that he wouldn't remember me. Nonetheless, when he came on the line I briefly reminded him of our "meeting" and promptly launched into a description of my proposed trip to China. As I went on I sensed that his interest was piqued. After a while he interrupted. "WHO did you say you were?" he asked. I quickly reiterated my earlier explanation and returned to China. "Sounds interesting." he said. "Who's going?" I fearlessly read off the names on my list, to none of whom I'd yet spoken. "Sounds like a blue-ribbon group," he said. "Count me in." I put a check mark beside his name and moved on to the next person on my list. This time I was able to announce that Al Perlis was going, and that was a clincher; if it was good enough for Al, it must be OK. As I continued down the list, my task became easier and easier, and the sheet rapidly filled with

check marks. By the time it was finished I felt I'd earned my certificate as a con man.

I had asked everyone to send me their curriculum vitae and over the next few weeks some extremely impressive packages were to arrive on my desk. My next move was to try to contact the Chinese embassy in Ottawa. (The U.S. had no formal relations with China at that time, and thus no embassy.) I managed to find their Ottawa phone number and enlisted the help of Tom Cheatham, one of the prospective participants, at Harvard in making the call. I thought that the Harvard connection would enhance our credibility, but when the phone was answered we suddenly found ourselves face to face with the impenetrable caution and reserve of the inscrutable east. We explained briefly what we wanted, but all that came back in return were barely perceptible grunts of acknowledgement. Clearly we needed help.

At that time there were articles in the paper about increasing contacts with the Chinese, and someone named Daniel Tretiak was often quoted as a China specialist. His name turned out to be in the Boston telephone directory and when I called and explained what I had in mind, he expressed considerable interest and suggested that he might be able to help. We arranged for him to come to dinner and when he arrived I was startled to see someone only about four feet tall striding up the front walk alongside a statuesque black woman who turned out to be his delightful wife. I wondered briefly whether I had somehow found my way into a James Bond movie, but such thoughts were quickly dispelled by their completely natural charm. Here, moreover, was someone who spoke with authority, obviously had connections that could be invaluable, and spoke fluent Chinese. He agreed to accompany us to

Ottawa and introduce us to the staff at the embassy there with whom he was already familiar.

This was a real break. I contacted Cheatham and not long afterward the three of us stood outside the embassy in Ottawa. Stretching as high as he could reach, Tretiak rang the buzzer. When the inscrutable east answered, Tretiak swung into rapid fire Chinese and shortly we were ushered into a mini-China, replete with what I would come to recognize as all the standard trimmings—antimacassars, a large thermos of tea, and profound reticence. I had never before seen human features that revealed so little; the full meaning of the word inscrutable was becoming clear. Proceedings were one-sided as Dan, unreeling his quiet Chinese, explained the purpose of our visit. I, meanwhile, tried to determine whether they thought we were spies or merely insane. After a while, Dan indicated that it was time to hand over the formal request I'd prepared together with the sheaf of résumés. After a few further "pleasantries" we were returned to the street, where Tretiak assured us that it had gone well. I'm not sure how he knew, perhaps because we hadn't been bodily ejected. In any case, who could tell what the officials at the other end in China proper might conclude. These guys were mere functionaries; the total absence of any reaction was, of course, precisely what they'd been trained to exhibit. Back in Boston I reported to the prospective participants and then proceeded to forget about the whole thing. We'd done what we could do; now it was in the lap of the Gods and the powers-that-be in Beijing.

Nearly a year later, as spring came on and we had heard nothing, we concluded that the predictable thing had happened to our request; it had been "filed" somewhere. We'd asked to go in the summer when school would be out.

Many of the participants were professors who now began dispersing to the far corners of the globe. Then one day in April, just when I'd completely forgotten about it, an innocent looking envelope appeared on my desk. When I opened it there it was: an invitation for six computer scientists to come to China, together with their wives, for a three week visit!

Overnight the world changed. The U.S. government suddenly expressed interest. A letter arrived from Nixon's Science Advisor, Ed David, on impressive-looking White House stationery. The National Academy, which for years had been trying to establish contact with the Chinese scientific community, but had been rebuffed because of its close ties with Taiwan, commenced treating us with respect and promptly offered to pay our travel expenses. Gradually the realization came over me that this was to be far more than a mere travel experience—suddenly we were in the middle of international politics at a critical moment in history. The importance of our mission was belied by the modesty of the letter that had arrived. We had apparently been singled out as having just the right character to comprise the first scientific delegation in a generation: We had no direct government ties and were researchers and academics in a field that the Chinese wanted to learn more about. Our invitation was a signal that the door was cautiously opening.

I had the immediate problem of selecting six out of the fifteen people we'd originally proposed. Having suggested the trip in the first place, and having done virtually all of the arranging, there was no question in my mind that *I* was going to go. That left five others to select. I didn't want to take responsibility for making such a difficult choice single-handedly, and arranged for the group to self-select from

within its own ranks by voting. When the dust settled, the six who went were myself, Al Perlis, Herb Simon, Wes Clark, Tom Cheatham, and Anatol Holt. Although the most junior member of the group, I was nonetheless the person who had made the formal contact and whom the Chinese thus recognized as the head of the delegation, a rôle I would slowly grow into. In the meantime Nixon and his cortège took over the spotlight and made history with their trip, which I was sure had helped pave the way for our own venture.

The days leading up to our departure were frantic. As we climbed onto the plane for Hong Kong, some of us were meeting one another for the very first time. But we had many hours to get acquainted during the long flight and several days thereafter in Hong Kong during which we made final arrangements. As we finally crossed the border and were met by our Chinese hosts, the significance of our visit was brought home to us—we were greeted with great gravity and formality and treated like V.I.P.s. Each of us had been provided with a private car, a driver, and a translator. It took some time for us to adjust to such treatment.

Although we hadn't reckoned on it, we had unwittingly brought along our own personal ice-breaker in the form of Al Perlis. Al was suffering from multiple sclerosis and by that time could walk only with difficulty using canes. The Chinese instantly produced a wheel-chair and the combination of Al's disability, his perpetual good spirits and omnipresent sense of humor (which the Chinese immediately sensed), broke through all the barriers of formality and allowed genuine human contact to flourish. Our hosts also recognized the heroism of someone who would make such an arduous trip under the circumstances,

and Al quickly became their favorite. Eventually we were to meet with China's senior and most honored scientist, Kuo Mo Ro, head of the Chinese National Academy. As he shook hands with each of us in turn I watched as a broad smile burst onto his face when he came to Al. Kuo Mo Ro was elderly and recognizing a fellow sufferer, drew him instantly to his side. Sadly neither Al nor his wife Sydelle nor Kuo Mo Ro is any longer with us. In later years I would come to know Al and Sydelle as close friends and to appreciate even more deeply his extraordinary pluck and great humanity and her devotion in caring for him.

Our visit took us initially to Canton where there was very little to see in the way of technology. There, however, we were treated to the first of many banquets at one of the best restaurants I've ever been to. Our host for the evening was a giant of a man who, although clearly friendly, had the aspect of a menacing Chinese tank driver from an American comic strip of the period. I am basically a non-drinker (all this talk of Martinis and Scotch notwithstanding), and had been assured that the Chinese don't drink. I was therefore eying the multiple glasses at our places with some uneasiness when, to my disappointment, this leviathan filled one of his glasses with a villainous looking liquid, stood up, and began to toast me as head of the delegation. He finished up with a phrase I was to come to loathe, "Gan Bei" which clearly meant "bottoms up" (literally "dry glass"). In a flash I saw that my choice lay between utter disgrace and, in all probability, throwing up all over the gentleman as I attempted futilely to duplicate his act. Wesley, bless his heart, recognizing my predicament, promptly stood up and, after a short speech, casually tossed back the entire contents of his glass. International relations had been rescued from the brink, but some damage had

been done to my personal reputation. In the fullness of time, it would be repaired.

Over the next two days we visited a museum and a kindergarten, and were treated to a movie of a heroic, revolutionary ballet that we were later to see live in Beijing. We also witnessed two major operations utilizing nothing but acupuncture for anesthesia. These events allowed us to become accustomed to our new environment, to our hosts, and to the behavior that was expected of us. We were quick studies, although I never completely managed to forego occasional attempts at informal communication. But it never worked. We were being shoved into the rôle of semi-official emissaries, whether it fitted or not. Everyone was extremely friendly and hospitable, but there were clear bounds to which we Americans were unaccustomed.

After Canton we proceeded to Shanghai where we presented lectures and held discussions with members of the Shanghai Computer Institute of the Academy of Science. The computers we were shown were several years behind what one would find scattered around the U.S. at that time. In reading over a summary of our trip that Wes and I wrote shortly after the event, I find the statement: "Generally speaking, Chinese computer technology seems to be pursuing a course not dissimilar to that followed in the early days of development in this country, that is, prior to the great proliferation of machines in the mid-sixties." It seems that regardless of when one looks back, the "great proliferation" appears to have just occurred.

We found that practically all of the pieces of a machine were manufactured, often by hand, in the factory where the machines were produced. There were no separate suppliers, no sub-assemblies; no real support industry had yet evolved. We later saw evidence, in other contexts, of an

absence of the sort of production engineering that would be taken for granted in the U.S.—even things as simple as laying out the production process in a sequential line. Instead, items were moved around hither and yon as they passed through the various stages of production.

After Shanghai came Beijing where we exchanged lectures with scientists at the Computer Institute and Tsing Hua University. The Chinese were particularly curious about large, powerful machines and about the latest advances in large-scale integrated circuit design (something our particular group was ill-prepared to discuss in any depth).

Our visits to Tsing Hua University revealed the magnitude of the changes wrought by the cultural revolution. The University was now administered by a Revolutionary Committee consisting primarily of workers, peasants, and soldiers, with some professors and students also participating. Teachers and professors went to work on farms and in factories as part of their "re-education," and in general academic standards appeared to have relaxed.

Between lectures we visited the cultural and scenic sites that were to become standard stops on China tours over coming years. We were housed in a great caravansary next to the Forbidden City and one night, having gone out on the balcony of my room to cool off, I was dismayed to hear the door into my room click shut behind me, neatly locking me out. The neighboring room had a similar balcony and I could see that the door was open into its room. There seemed little alternative, so once again utilizing my rock climbing experience, I was soon on the adjacent balcony. Stepping into the room I surprised two very middle-eastern-looking gents in their underwear (it was fiercely

hot) who, as I gesticulated, roared with laughter as I proceeded out and into the hall to return to my own room.

A meeting in China

Rarely were we allowed to sleep, and by the penultimate banquet in the Great Hall of the People we were in a state of utter exhaustion. On our way out of China, however, I was finally able to square accounts with the tank driver in Canton. This time, as we sat down to our final banquet, I noted with relief that my glass had been filled with innocuous looking red wine. But aside from that, I was ravenous. I had learned that at these banquets, etiquette required that one place the food on one's companion's plate, never on one's own. He, in turn, did likewise. It was rude to decline an offering. As I chatted with my tank driver, I became aware that he was slowing down. I, on the other hand, was still famished. With unswerving determination I continued to feed him until he

was finally forced to gesture "no more." As we staggered out into the moonlight, he put his great arm around my shoulders, indicating that the score had been evened and I had regained his respect.

When we arrived back in Hong Kong, waiting at the train station was the woman with whom I was destined to share the rest of my life, Laura Gould. She had been teaching computer science to humanities students at U.C. Berkeley and had received Berkeley's Distinguished Teaching award. We had met earlier that year at a follow-on workshop to the one where I had originally proposed the China trip. The two of us spent a short holiday on Japan's northernmost island, Hokaido, recovering from the stress of the visit to China, and then headed back to the U.S. We stopped briefly in San Francisco where we visited a place that was later to become home territory for both of us. George Pake, the former Provost of Washington University, had been hired by Xerox to put together a research center in Palo Alto and a group of very bright computer scientists had been assembled in what was bemusedly referred to then as Pake's PARC—the Xerox Palo Alto Research Center. In addition to George, I knew a fair number of the people who had gathered there and we spent an hour or so sitting around in the bean bags that would become emblematic of PARC, discussing the trip I'd just been on.

Eventually Wes and I wrote a report of our trip for *Science* and Mary Allen wrote a feature cover-article for the Washington University magazine. (see Bibliography)

Chapter 17

Pluribus struts its stuff

The adventure in China, together with the short vacation in Japan, had taken just over a month and now, back once more at BBN, we settled into our new life together. Laura shortly commenced working in the speech-understanding group at BBN, and I picked up the threads of the multiprocessor design. We had assembled a powerful team of designers, including many who had worked on the earlier IMPs. Before we could design couplers to interconnect the machines, however, we needed to understand the SUE in great detail, and here we encountered the first of what would turn into a succession of problems with Lockheed. Over the ensuing months we came to realize that Lockheed had not fully understood the implications of the structure that they had devised, and certainly had no inkling that anyone would ever try to do with it what we proceeded to try to do—although it was within the advertised capabilities. They were unused to customers who had read their specifications carefully and expected their machines to meet those specs. In holding their feet to the fire, I fear I was not always kind or patient. On the other hand, they deserved all that they received, and not a few Lockheed heads rolled in the course of our interactions. We were ultimately forced to help them redesign two critical integrated circuits in order that the machine would work as promised. After endless phone calls and far too many trips back and forth across the

country, we were able to assemble working multiprocessor hardware.

In the meantime the software team had been preparing a version of the IMP program that would be structured to work on this new kind of multiple machine. In addition to the regular IMP job of transporting and routing messages in the network, there were tasks associated with doling out the work among the processors and keeping track of which machines were healthy. The individual machines watched over one another for possible misbehavior and periodically tested one another. If a member machine or other element appeared to be causing trouble or otherwise misbehaving, a vote could be taken to excise the errant piece from the system. The individual machines could also reload and restart one another if trouble occurred. And finally, they watched for, and brought on board, new members as new machines were plugged in and powered up.

As the reliability features of the program began working, the overall machine started to take on the feel of a gyroscope, manifesting something resembling a will of its own. Our goal was to make it impossible (i.e., extraordinarily unlikely) for any single failure (short of a total power failure) to damage the overall machine's functioning for more than a brief period. Turning off or removing one machine had no effect—the overall system continued to perform the network job. The day came when we demonstrated the machine to some visitors and challenged them to see if they could break it by pulling out cards, throwing switches, etc. (One thing at a time of course.) One of the men bent down and, with a knowing smile, threw the main power switch on one of the racks (which simultaneously shut off several machines). As there were other racks containing duplicate pieces of hardware,

the system kept right on running. Peeved by his failure, he reached down again and this time turned the switch on and off repeatedly as fast as he could, then went to the next rack and did the same thing and on to the next, and so on. At this, the system paused, causing him to smile triumphantly. But after a few seconds, to his astonishment (and our delight), it resumed operating.

The Pluribus Team[42]

[42] Left to right, Front row: Tony Michel, Marty Thrope, Bob Bressler, Dave Francis; standing at left: David Katsuki, Dick Garber, Ben Barker (inside rack); standing at right: Mike Kraley, Steve Jeske, Will Crowther; above; Frank Heart, Severo Ornstein

It was now time to name the machine. I had learned years before that the two most important features of any computer were its color and its name. We all agreed on a pleasing sort of pale blue for the color, but the name proved more difficult. Several names were proposed and after a number of straw votes we finally settled on the name *Pluribus*. This was a double pun. Not only did it evoke the "e pluribus unum" (from many, one) of the dollar bill, but it also spoke to the fact that the machine consisted of multiple busses. A "bus" is a sort of machine backbone consisting of a common set of wires and a discipline for using them by means of which the various parts of the machine communicate with one another. In this case, where there were multiple machines, there were multiple backbones, all interconnected by our specially designed bus-couplers.

While we were working on the design of the Pluribus, another multiple machine called C.mmp was being built at Carnegie Mellon University (CMU) in Pittsburgh. Their goal was fundamentally the same as ours—high speed. But while we needed more speed to handle multiple high-speed network lines, the CMU group was interested in the speed in order to facilitate Time Sharing. The overhead of switching attention among multiple users ate up machine cycles, and of course the faster the machine, the more users could be accommodated. Hence speed was doubly important. The problems of sharing a multiprocessor (i.e., multiple computers joined together) among multiple users (with varying demands) was indeed daunting.

It is somewhat unfortunate that the term "multiprocessing" was already being employed by the Time Sharing community to describe what happens when a computer is used to perform multiple jobs at *more or less* the same time—by chopping the jobs up into little pieces and

processing the pieces in rapid succession. Single computers have been doing multiple jobs "at once" for many years; you don't need more than one processor to do that. Now, however, we began using the term "multiprocessor" to mean multiple actual hardware processors working together on a job—any job, including possibly a multiprocessing job. With a multiprocessor you get truly parallel operation, in which multiple, independent tasks are *actually* executed simultaneously. This is very different from the *illusion* of simultaneity that is created when multiple jobs are handled piecemeal in rapid succession by a single processor.

In fact, both the Time Sharing application which the CMU people were trying to fit onto their multiprocessor, and the IMP job, which we were trying to fit onto ours, are multiprocessing jobs—that is, numerous and somewhat unpredictable tasks must be handled without dropping anything on the floor. There was friendly competition between the two groups, and we met jointly from time to time to share experiences and views, and to poke gently at one another.

They felt that the machine we were designing was applicable only to the sort of specialized job that the IMPs performed, where the program was broken into sub-pieces (tasks) when it was being written. With Time Sharing, who could tell what kind of program some user at a terminal might suddenly launch? How could it be broken up into pieces that could be handled simultaneously? So they argued that there was no way to fit Time Sharing, with its inherent unpredictability, onto such a machine.

The IMP, on the other hand, faced real-time deadlines that Time Sharing did not. Yes, it would be nice if Time Sharing systems ran fast, so that users wouldn't be annoyed

by slow response, but if the IMP failed to keep up with the passing of messages, the result was catastrophic failure rather than mere annoyance. Each group thought that the other was a bit wrong-headed, and we were proceeding down our independent paths when a call from the outside world came in from a country I'd visited before and had loved, Chile.

Chapter 18

Chile revisited; changes at BBN; I head for PARC

It seemed that the Chileans wanted to learn about networking. They weren't the only ones. We'd traveled to a number of foreign lands by this time, spreading the word about the usefulness of networks. News of the ARPANET had spread and there were commercial applications for special networks waiting to be built in many places. So Laura and I and Alex McKenzie climbed onto a plane and headed for Chile.

Chile had changed in a major way since I'd been there before. My first visit preceded the days of Salvador Allende, an era when things were relatively peaceful if somewhat disorderly. By this time Allende had come and gone, Pinochet was in power, and there was an abundance of "law and order." The streets were patrolled by armed guards, and a strict curfew was in place. Despite this, we encountered the same wonderful Chilean hospitality I remembered, although the midnight curfew conflicted with the custom of 11 P.M. dinners and people rushed to return home promptly after eating. We were told that violators could be shot on sight and if one didn't make it home in time, one spent the night wherever one happened to be when the curfew hour arrived.

Our lectures were more or less routine descriptions of the network we had built, together with the reasoning behind major decisions. We were speaking to a collection of

higher-ups from the communications and banking industries. We kept asking who our hosts were and got evasive replies, indicating that it was some branch of government we clearly didn't understand. The day finally came when we were to meet our host face to face and it was an evening I won't soon forget. We were taken to the top floor of a downtown hotel and as we stepped from the elevator we left the world of Chile and entered the world of USA circa 1950. The contrast with the street below was staggering. A band was playing, the lights were dimmed, people were swaying in a slow dance. We were ushered to a table where we were greeted by our host—a military man replete with the classic dark glasses, speaking impeccable English with no trace of an accent. It turned out he was a product of the U.S. Air Force Academy in Colorado and over dinner he explained to us how little the people of Chile understood what they needed. When I finally got back to the hotel, I felt I needed a bath.

The extreme discrepancy in wealth was both apparent and appalling. By now we in the U.S. are experiencing some of the same thing, although the process of wealth concentration hasn't yet proceeded as far as it had in Chile at that time, and of course the overall abundance is much greater here than there. Nonetheless, memories of what we witnessed there cause me to be concerned about the extent to which we are redistributing wealth in our own country and to wonder what will become of us if we continue in the directions we've been going. When I first went to Chile in 1966, I returned to a United States in which a homeless person was a rare sight. Today it has become all too commonplace.

But at that time other things were beginning to bother me. BBN had changed in the eight years I'd been there. The

company had grown enormously and changed character. Concern about the bottom line was replacing the earlier focus on interesting research. The ARPANET was turned over from Arpa to the Defense Communications Agency, which was interested in its utility, not in research. At the same time there was increasing emphasis on building systems for the military, some of which involved classified work. By this stage in my life I didn't want to have a thing to do with classified work and I began to think about leaving. To my everlasting regret I made a false start, and ultimately took over a year to make a final decision to depart. Despite this, when we left BBN threw us a truly grand and heartwarming party. I gave away snow tires, snow shovels, windshield ice-chippers, and all such wintry paraphernalia. We were heading for a place where it seemed unlikely they would be needed.

Chapter 19

A place of character(s); I love my Alto; the imprint of Engelbart; Thacker's decision; the importance of tools; an unfortunate erasure; Bob Taylor—missionary; el Dorado

After many years I was finally returning to El Dorado. By this time (early 1976), Silicon Valley was in full swing and a large number of our friends and colleagues (including many from BBN) had already found their way to Xerox PARC, which was rapidly becoming a leading center (in fact *the* leading center) of computer research in the U.S. I had decided to follow this migration. Although it was part of an enormous corporation, PARC had by then acquired a far more academic feel than BBN. The place had a decided "religion," but researchers were encouraged to pursue their own interests and projects. Furthermore, the religion was one that I could happily sign on to. It emphasized personal machines, tied together in a local network based on the Ethernet that had been developed at PARC. There were shared facilities, notably a laser printer and large file storage, but the structure allowed most of a researcher's work to be done on his or her personal machine, and relegated sharing to those things where the economics required it but where contention would be minimal. This seemed an eminently sensible solution that has indeed become widespread as the years have passed.

As at BBN, there were two computer labs at PARC with related interests but somewhat different character. I found

my way into the Computer Science Lab, then headed by Jerry Elkind (my former boss from the BBN TENEX interlude), later to be headed by Bob Taylor. Laura joined the System Science Lab down the hall where Alan Kay held court. As at BBN, we were fortunate to be able to work together in the same place but on different projects. The mirror in the bathroom where we took our morning shower was covered with traces of the diagrams we sketched on its misted surface as we discussed our work with one another.

Bob Taylor

PARC, at least the computer labs, had a very special character that was set in part by the lab leadership and in part by the nature of the virtuoso researchers assembled there. When Taylor became head of the Computer Science Lab, I never knew his door to be closed. He always welcomed visits by any and all members of the lab and was happy to discuss just about any topic. But while he insisted on complete openness between people on anything related to the lab, he also insisted that people's personal lives were strictly their own business. At one level there was great informality and individual freedom with a total absence of the usual structures for keeping employees in line. People came and went as they pleased; it was simply taken for granted that everyone would pull his or her share. (One of the PARC people later went to work for a while at Xerox headquarters in Connecticut and was appalled when suddenly the lights went off at something like ten o'clock at night. It was simply assumed that by then everyone would have gone home. Not so at PARC, where people could be found working away at every hour of the night.) On the other hand, there was a feeling of cohesion and esprit de corps, of everyone pulling together in the same general direction, and diversions were definitely not encouraged, especially when they ran counter to the group ethos and overall direction.

Bean-bag "chairs," which filled the offices and conference rooms, came to symbolize the informality of the place. They represented one of many devices for encouraging casual contact and interchange between researchers. A weekly gathering of the Computer Science Lab, known as Dealer, in which one or more individuals stood up and presented some new research findings or questions, was a free-for-all of ideas. People often

volunteered to be the speaker at these gatherings, knowing that plenty of slings and arrows would be forthcoming—as well as cheers, laughter, and applause when deserved. And those who were too shy to volunteer were eventually volunteered. Everyone who wasn't in the middle of some compelling experiment attended Dealers. They were chaired by Bob Taylor in a ritualistic manner, pipe and Dr. Pepper in hand—directly beneath a sizeable "No Smoking" sign. (Almost everyone else in the lab had given up smoking by that time.)

In addition, speakers would occasionally come from outside of the lab and when they did so, many came with justifiable trepidation. The reputation of PARC for unflinching outspokenness (some said arrogance) had spread to the outside world and intimidated many. We showed little mercy to one another and came to accept a level of forthrightness that could stun an outsider. One day, as he was beginning his presentation and looking down on this unusual audience slouching comfortably in their bean-bags, a somewhat nervous speaker, looking up at the No Smoking sign, asked tentatively, "If I smoke in here, will I be stoned?" Instantly came the reply, "Depends on what you're smoking." Another poor chap was attempting to make an improbable case, and his audience were beginning to fidget. Suddenly, from the back of the room, came a clarion voice. "Bullshit!" it said. The speaker staggered as though shot. The ensuing discussion left him bereft of both argument and topic. To someone with fragile nerves, unused to this level of candor, such a stroke could be unmanning and left some speakers badly-shaken.

Potential new researchers were invited for a visit and circulated around the lab, chatting with various members who expressed interest in their specialty. The prospect

would then make a presentation to the lab of some piece of current work as part of the interview process. For many this was a daunting experience, although of course, some relished this kind of spontaneous give and take. (I'm one of the few people who endured this process *twice*, having declined the resulting offer the first time around.) After the interviewee had departed, Taylor would call together the senior members of the lab to discuss everyone's views. From the many such discussions in which I participated, my favorite comment came from Jim Morris who possessed an unusually dry sense of humor. We had been interviewing a very tall fellow, taller than any member of the lab. Others had been chewing over his presentation, his competence, and likely contributions to the lab. When it came Morris' turn to comment, he said simply, "Seems pretty bright, for such a big guy."

The computer that existed at PARC when I arrived early in 1976 was the Alto. I believe PARC was the first place where every researcher was provided with a personal machine as a standard piece of office equipment. The day I arrived, one was wheeled into my new office. The Alto had been designed by Chuck Thacker and Ed McCreight, based on a description in an earlier memo by Thacker. Butler Lampson, Bob Sproull and others had written the software, including the operating system, the compiler, and some of the early applications. The philosophy underlying the Alto depended heavily on work done earlier by Doug Engelbart, at SRI.

In the late 1960s, Doug, one of the truly great innovators of the modern computer era, was experimenting with new ways for users to interact with a computer via

displays.[43] I visited his laboratory when one of the early ARPANET IMPs was being installed there and had seen his revolutionary system in operation. That system formed an absolutely critical step in the development of today's graphical user interface. Doug foresaw the very sorts of use we all experience today, but the inexpensive chip technology, upon which such use is dependent, was simply unavailable in the 1960s. Instead his system was based on a Time Shared computer (an SDS 940). In the computer room down the hall, each user's work was displayed on a small, individual, vector-type CRT (See Appendix II). Mounted in front of each CRT was a television camera whose video signal was piped to a television monitor at the user's station (think desk, although these users were often gathered in beanbags). Mixed into each user's video image was a cursor, controlled by the user's mouse, that allowed items on the screen to be pointed at. Doug had designed the mouse for precisely this purpose, and although today he is finally being recognized as having "invented the mouse," his contribution is far broader, and encompasses the basic concept of direct user control through an interactive display. His prototype system on the Time Shared machine using primitive television techniques was cumbersome by today's standards, but it pointed the way to modern usage. In 1968 he gave a historic public demonstration of his work that blew the minds of all who were present. Alan Kay, who

[43] Earlier machines, such as the SAGE machines, TX-2, and the LINC, had displays and input devices that were used to interact with programs, but Doug extended and generalized this use so broadly that it led to a qualitative shift in the way users thought of interacting with their programs and data.

was there, describes Doug as "dealing lightning with both hands."

The next steps took place at PARC just a few years afterward. In Engelbart's system, the image produced by the computer consisted of line segments. This limited the kinds of pictures that could be presented. Television was used only in a passive sense, to transport the images from the computer room to the user's monitor. The PARC researchers decided instead to create television images directly from the computer—images in which every point of the television scan was individually specified by a bit in memory. This meant that for the first time arbitrary pictures could be generated at television rate and resolution.

A preliminary system was built that included an expensive external memory for holding the bits of the image. A short while later, in designing the Alto (which was to be a personal computer and thus needed to be comparatively inexpensive), Chuck Thacker decided to dedicate a significant segment of each computer's *main memory* to holding the bit-for-bit image of the display screen. Having the bit-map directly in the main memory, as opposed to an external memory, allowed manipulation by programs at a level of intimacy (and thus speed) never before possible. The cost and availability of integrated circuit memories at that time brought this scheme barely within reason, and the decision by Thacker to dedicate memory in this seemingly profligate fashion was an act of

considerable daring.[44] The payoff, however, was enormous. By facilitating the graphical user interface (as it has come to be called) it opened the door to a fundamental change in the way people interacted with computers. Today, of course, displays are based on many different kinds of underlying technology, but the basic notion of a bit-map, contained in the computer's main memory, which the user references by means of a mouse of some sort, has remained the canonical "user interface" everyone today takes for granted.

In many ways the Alto was also a descendant of the LINC, but of course in much more modern dress—so much so that the quantitative changes resulted in qualitatively different user capabilities. By today's standards the Alto was big, (the size of a hotel-room refrigerator), clunky, expensive, slow, and had limited storage. Nonetheless, functionally it embodied virtually all of the features to which the world has since become accustomed.

Another change manifested in the Alto was the disappearance of the console that, in one form or another, had graced the façade of earlier machines and provided the principal user access. What permitted the lights and switches to disappear was the development of a thing called a PROM. ROM stands for read-only-memory, and the P stands for the fact that, with special equipment, you could initialize (Program) the ROM with whatever you wanted before you put it into the computer. Once in place, however, the contents of the PROM could not be changed.

---

[44] Thacker says, "The use of main memory resulted from my internalization of the truth of Moore's law—semiconductor memory was barely cheap enough to do it at all, but would get much cheaper very quickly."

When programs fail to function properly, they rarely just cease operating: often they wreak havoc inside the memory, altering its contents. A PROM could provide a safe haven for bootstrap programs that, following a crash, could bring in a small debugging program to peer around inside the rest of the machine after the running program had gone aground. It could then report what it saw, on a printer or a screen, or on whatever other handy output device the machine might have. A different piece of bootstrap program could then reload the repaired version of the program for another run at the wall. Thus the need for the switches and lights, that had previously been used to explore and modify the machines innards, simply went away and was replaced by a program in a secure part of the memory—the PROM.

The bit-map display screen and the disappearance of the console had many practical consequences. For example, together they meant that debugging, heretofore a frightfully cumbersome process, was dramatically speeded up. But they also had profound philosophical implications. Note that when access was provided by switches and lights, the term "user" implied someone who was going to tinker directly with the machine. Now a "user" could be someone who interacted only with functioning programs that provided one or more applications such as a text editor or a graphics program having nothing whatsoever to do with the machine's innards. To the user, therefore, the Alto presented much the same façade as do the machines of today.

The final novel element in the PARC scene which tied everything together was the Ethernet. Not only were all the individual researchers' Altos interconnected, but special servers for printing and for large-scale file-storage were also

connected to the Ethernet. The Ethernet owes a substantial debt to Norm Abramson as mentioned earlier. Bob Metcalfe, who together with Dave Boggs developed the Ethernet at PARC (and later went on to found 3Com), had discussed the Aloha Network in his PhD thesis and he exploited the basic scheme (with many important refinements) in producing the original Ethernet at PARC.

All of these elements had come into existence at PARC by the mid-1970s and numerous experimental software applications, forerunners of many of today's most common user programs, were rapidly proliferating. "Windows" (the technique whereby rectangles of visual material could be overlain on the screen) were already being developed and demonstrated about the time I arrived. (Many years later Microsoft acknowledged the importance of this idea by adopting the name for its revised operating system). A text editor called BRAVO, very similar to Microsoft WORD (from which WORD, in fact, derives, having been rewritten at Microsoft by Charles Simonyi, Bravo's implementor at PARC) was in place and was already being upgraded. Most of the refinements that have taken place since are too complex to describe here and, although they underlie the ease with which today's users accomplish many of their tasks, they do not fundamentally change the user's perception of the machine.

In almost every technological field, the development of tools has proven critical. In no field has this been more true than with computers. Here tools have been required to build other tools and gradually a giant pyramid of such tools came into being, with each layer dependent on the one below it. Computers themselves quickly became a principal tool for building further computers. And if the tools for building the machines themselves formed a pyramid, the

programming tools for creating software systems constitute a veritable Mt. Everest. As a consequence of this pyramid of tools, the race, particularly in the software field, has often gone to those who first understood the need and went to work building the underlying tools. The people at PARC understood this situation better than almost any other group and this, more than anything else (other than perhaps the driving force of a few insightful individuals), led to their preeminence in the field. Of course, like anything else, it could be carried too far—and sometimes was.

So much for an introduction to the PARC scene as it existed when I arrived. Because I hadn't come with a particular project of my own in mind, my initiation took the form of working on a project that was already under way. The Computer Science Lab had earlier designed and put together a laser printer. It was a giant affair and consisted of a number of elements that filled an entire room. It was connected, via an Alto (which formed one of the elements), to the Ethernet so that its use could be shared by anyone with a computer connected to the Ethernet. Having successfully built this monster, people had then set about making something smaller and better integrated. In another PARC lab, Gary Starkweather tore apart one of Xerox's standard copiers and installed a laser mechanism in it for writing a scanned image onto the drum. Meanwhile Bob Sproull and Butler Lampson in the Computer Science Lab set about identifying parts of the image conversion job previously handled by hardware that could be turned over to software (and microcode), thus reducing dramatically the quantity of specialized hardware required. This process of distillation and reassignment was pretty well completed by the time I arrived and it had become obvious what

functions would be needed in the much-reduced, specialized hardware for the next generation. I was to design that hardware. This was something of a challenge as it had been some time since I'd actually done any design myself and instead had been supervising a team of designers at BBN. New, more sophisticated chips had come into being, and furthermore the design process at PARC employed new and unfamiliar design tools. So there was a lot to learn in a hurry.

One of the people who had worked on the design was Bob Sproull, whom I'd previously encountered at various places along the way (Harvard, the Stanford Artificial Intelligence Lab, etc.) Bob was leaving shortly to lecture in India and just before he left he spent a day imparting all of the information I would presumably need to proceed. Bob moves *very* fast, and as he sped along in his explanation, I was hanging on by the fingernails. At that point I hadn't the vaguest notion how a laser printer might work. Thankfully Bob made squiggles on the white board as he spoke and I knew that I'd be studying that white board carefully for some days to come.

Next morning I arrived in my office to find it being rearranged as I'd requested. This involved shifting the precious white board a foot or so to the left, and to my horror I saw a workman bellying up to the shifted board as he screwed it down in its new location. In the process, he had removed a sizable portion of its contents onto the front of his overalls. I was to spend the next week scrutinizing that white board with the squinting eye of an archaeologist, attempting to induce it to release its secrets. Eventually, with the help of others, I came to understand what was needed and in due course the hardware was built. Bob returned in time to help with the debugging and to write

the necessary program to make the whole shebang go. Shortly we had a much more compact printing facility called a Dover. It was then duplicated and soon became the workhorse printer, not only for PARC, but for a number of other computer science labs around the country as well. Some of the same logic that drove that printer now resides, in far more compressed form, in the tiny commercial laser printer that sits on my desk here at home.

Many of the programs that came out of PARC arose because the person devising them wanted such a tool. Chuck Thacker, as a hardware designer, envisioned a graphic logic-design system that would permit the automatic production of wire-lists directly from logic drawings, previously a terribly error-prone manual process. The front-end of such a system would be a program that permitted one to construct and manipulate screen-based logic drawings, so Thacker wrote such a program and modestly called it Simple Illustrator (SIL) because it was useful for making all sorts of drawings as well as logic diagrams. The mouse at PARC had three buttons and Thacker used these in conjunction with various keyboard keys to provide the necessary array of graphic functions. Most programmers would have used the keyboard keys mnemonically, but Thacker, realizing that the right hand was occupied driving the mouse, chose to use keys in the lower left side of the keyboard in such a way that the necessary combinations lay conveniently under the left hand. I have noted with pleasure that this piece of cleverness, obvious perhaps, but only once you think of it, has found its way into the conventions for standard functions (such as Cut, Copy, Paste, and Undo) on the Apple Macontish.

Although the Alto was a wonderful computer, the usual thing had happened: Ambition had already outrun current capability and a new, more powerful machine was needed for the software research proceeding in the lab. Thacker and Lampson had all but completed the design of a couple of prospective machines, and a new arm of Xerox, which was trying to find ways to utilize ideas from PARC and bring them to market, was supposed to build a number of these research-prototype machines. But by the early spring of 1977 it had become clear that this was not going to happen and that if the machines were going to exist, our own lab would have to find ways to finish off the designs and build them. The hardware people, including me, balked at this idea. We didn't want to take on the chore of building a machine that had already been designed; we wanted to do more innovative work of our own. At this point, enter the arch persuader, Bob Taylor. But before I continue with the story, I must pause to say a few words about Bob himself.

When I had arrived at PARC, Jerry Elkind had been director of PARC's Computer Science Laboratory. By this time, however, Jerry had gone on to other matters and Taylor had taken over as director. Managing a bunch of prima donna computer scientists, most of whom had several outstanding alternative job offers at any given time and all of whom had specialties he couldn't possibly hope to understand, was a job that only a Texan would undertake—and Bob qualified. He has been described as an administrator with no particular technical ability, but vision and dedication were seldom so felicitously combined. I think of his intuition as something like an itch on the back: You can't scratch it yourself and can only direct someone else to the right general area. But when they finally light on the right spot, you certainly know it. I think that's how Bob

directed research. Although he was not an engineer, he had a truly extraordinary sense of what was important and what was needed. In this he no doubt adopted much of J.C.R. Licklider's philosophy, but he believed in it to the bottom of his soul. No doubt his zeal at times verged on mania (as viewed by some who stood in his way), but I think the importance of this zeal, and the vision that underlay it, has often been underestimated. His technical betters at PARC repeatedly bowed to his judgment, and, following his lead, deployed their skills to the best of their abilities in pursuit of his vision. In many ways the Alto was the culmination of a sizeable piece of that vision.

Bob's managerial style consisted in gathering together the brightest researchers he could find and giving them all the support and encouragement he could muster, with little direct interference. In addition to the regular staff members, he attracted to the lab a variety of associates from universities who were given part-time appointments. One of the more colorful (and brightest) of these was Bill Gosper from MIT. Bob's complacency was tested one day when a young associate of Gosper's showed up in the lab wearing a long, brightly-colored African dress. Nothing unusual about that, except that the associate was male. Bob didn't say anything, but I suspected that his Texas background was a bit stressed. More to the point, I can imagine that he was also thinking, "if some Xerox executive comes by today, I'm going to have a problem." Sometime later Bob, Ed Fredkin (visiting from MIT), and I were deep in discussion in my office when the young fellow happened by the door. Fredkin immediately leapt up to give him an enthusiastic greeting. Clearly he was well known and highly regarded at MIT. In fact a year or two later he showed up in my office, this time wearing a three piece suit

and looking extremely successful. He had started his own company and it wouldn't surprise me if today he is a very wealthy man.

The disagreement within the lab about whether and how to proceed with the building of an altogether new computer was grist for Taylor's mill; you could almost see him rubbing his hands in glee. He arranged for a series of lab-wide meetings to discuss what to do. I felt rather like a condemned prisoner sitting down with a group of executioners to discuss the forthcoming beheading. Over the course of very few meetings the size of the steam-roller became evident: It was made clear that we could be either the lubricant or the sand in the wheels of progress—it was entirely up to us. Laura and I took a long backpack trip at that point, in the course of which the reality of the situation became apparent to me. We actually had no choice. Unless we wanted to leave PARC (which one of the group actually chose to do), we were in for it.

I was asked to head the project to build the larger of the two machines, the Dorado, and acquiesced only under the condition that it be a joint partnership with Ed McCreight, whom I'd identified as one of the brightest, if not *the* brightest, person around. One of us had to know what he was doing, and Ed, in addition to being brilliant, was someone I knew I could work with comfortably—anyone could. Of course Butler, who understood the design better than anyone else, would also be on hand to help bail when the water started to rise around our necks. But there had to be some quid pro quo and I laid down other demands as well: I asked for and got a sizeable crew of people and then specified a schedule that I believed in, which everyone else thought absurdly pessimistic. Butler took me aside and told me that if people had any idea how hard such projects were

going to be, they would never have the courage to undertake them in the first place. I took this under advisement, but decided that there was a downside to over-optimism in which morale went through the floor as schedules started to slip. With this bunch, there was no question of people not working hard, regardless of official schedules. As Taylor himself had wisely observed, the far bigger problem was to keep people from burning themselves to a cinder. His technique had been to hire highly motivated people and thereafter frequently tell them, "You look tired, better go home and get some rest." I stuck with my prediction; others could believe what they liked.

Not surprisingly, it turned out to be both a lot of hard work and a lot of fun. It took us just about the two years I'd predicted to get a prototype working. Half way along, as spirits were flagging a bit, we took the entire group for a several-day retreat at a posh resort in Yosemite Valley during which we reviewed where we stood and what remained to be done. It was a welcome breather. It had been such a struggle that when the machine finally began working, some worried that it might never be possible to build the multiple copies to populate the lab as planned. Once again our faithful technicians were to surprise us and in the fullness of time working machines began to roll off of a mini-production-line (affectionately dubbed "The Garage" by Thacker) under their ministrations.

The Dorado was a much more powerful machine—faster, more capacious memory, disk, etc.—than the Alto and, of course, had a very different bottom-level internal structure. But from the user's point of view it did not appear significantly different from the Alto—just a *lot* faster. However, it was physically much bigger and it quickly became clear that, unlike the Alto, having one of

those in your office with you would be intolerable. It was big and noisy and generated a lot of heat, requiring serious air conditioning. So we (reluctantly) retreated to the old-fashioned idea of a machine room in which rack mounted machines could live and make all the noise and heat they wanted. Nonetheless, they were personal computers, with a cable connecting the screen, keyboard, and mouse in each office with the user's own machine in a rack, roaring away downstairs. From the user's point of view, things had gotten quieter, office space had been freed up, and, most importantly, things happened in a fraction of the time they had previously taken. The Altos had had removable disks about 15" in diameter, weighing perhaps five pounds, and containing about the same amount of data as fit on a diskette circa 1990. These had been replaced in the Dorado by enormous drives (think fork-lift) whose platters held the unbelievable quantity of eighty megabytes per machine—and later the even more unbelievable quantity of three hundred megabytes each. Many times more capacious disk drives now fit comfortably in a tiny laptop. Back then it took two strong men to lift one, and a fork-lift was employed to put the drives in and out of the machines—a mere twenty years ago.

Chapter 20

Music, music, music; bright students; Mockingbird

By the beginning of 1980 I was at last coming somewhat free of the Dorado project and was looking forward to revisiting my old interest in developing a computer-aid to notating music. I began poring over piano scores trying to understand the rules that governed standard music notation. What I really wanted was a music typewriter in which the computer would sense what was played on a (piano-like) keyboard and would produce a score directly from that. I was pretty sure that composers would relish such a thing, but I also knew that it was considered bad form to "compose at the piano." I decided to inquire of several prominent composers—Leonard Bernstein, Aaron Copland, Samuel Barber, and Virgil Thomson as it turned out—just how they went about the process of writing music. Almost all of them said "Well, yes, I use the piano because I happen to be a skilled keyboard player—but no one else does." I knew what to make of that. But of course no matter how much I waved my hands, trying to describe what I had in mind, none of them could possibly have any image of what I might be talking about. I doubt very much that at that point any of them had ever seen a computer screen, much less a mouse, in action.

I decided to consult existing computer-music experts and arranged to give a talk at IRCAM, a computer music research center in Paris. But I came away feeling that most

of my audience were surprisingly naive, not so much about computers but about music. I tested the waters at a few other places where computers and music were being joined, but found little sympathy for, or understanding of, what I was up to. The only person who showed any enthusiasm for what I was proposing was Don Knuth at Stanford. It seemed clear: I would have to proceed on my own and trust that if I succeeded, the results would speak for themselves.

It would, of course, not be difficult for a computer to sense the note-strokes, that is to determine what notes were being played and at what times. That kind of information had long ago been recorded on piano rolls, and of course we could get the same kind of information into a computer. But piano roll "notation" could not be read in any useful way by human beings. Instead, for human consumption, music (particularly the timing) is represented symbolically. Furthermore, in addition to the noteheads, scores include a great deal of clarifying notational information that exposes the internal structure of the music and suggests roughly how it should be played. This symbolic representation, which has evolved over the centuries as music itself has evolved, is essential in enabling humans to read and play, music at reasonable speeds.

So the question was how to transform raw note-strokes into the symbolic form in which music is normally represented in scores. The more scores I studied, the more doubtful I became that it would be possible for a program to infer the symbolic information directly from the raw note-strokes. I had seen far too many people come a cropper working on precisely this sort of artificial intelligence job, and as a pragmatist, intent on demonstrating a working tool, I was determined not to fall into that trap. I concluded that any attempt to do the transformation automatically

would, at best, be imperfect and would therefore require user tools that could redo any part of the job to repair mistakes. So why not start by building those tools? They would enable a person to do the complete job, and later on any parts of it that turned out to be susceptible to automation, could be automated. This sort of amanuensis approach to problems that require human intervention, in which the computer acts as a sort of scribe/assistant, had been promoted almost a generation earlier by J.C.R. Licklider. It seemed to fit this problem well.

Gradually I worked out a scheme in which the user and the computer in a kind of partnership could work over the note-strokes, gradually massaging them into something that resembled a score. I knew I could connect a keyboard to a Dorado in such a way that the program could measure the note-strokes, but beyond that lay a sizeable programming job with which I knew I would need help.

Like several other Silicon Valley companies, Xerox had an arrangement with MIT whereby each year a few selected students would spend the summer working under the direction of a senior researcher on some mutually interesting project—an apprentice arrangement that benefited everyone involved. They came for three successive summers, and in their third year they chose and pursued a project that would constitute the topic for a master's thesis.

In the late 1970s I was one of those who helped to choose the students, and a great pleasure it was. PARC had an excellent reputation so we had the cream of the crop from which to choose. But each year the choice became more difficult as the students seemed to get brighter and brighter. Having eliminated anyone who didn't have straight As, we were still confronted by tough choices. We

gave them exercises to work out while we stood by and observed how they approached problems. Most were imperturbable and exhibited frightening competence. They were also surprisingly personable, belying the typical MIT nerd image. The first year I deliberately chose a fellow by the name of John Maxwell III who seemed so bright and creative that I wondered if perhaps he would turn out to be a bit eccentric. (I needn't have worried—John has today become one of PARC's leading researchers.) In his first year I was pleased to find that John chose to work in Alan Kay's group where unconventionality was not only tolerated but cherished. Toward the end of his second year, about the time that I was mulling over my music notating ideas, I learned that John had expressed interest in working on some sort of music project for his thesis. Within days we began working together in a partnership that was to prove wonderfully productive.

The system we designed was one in which the user was provided with powerful tools for superimposing information about musical structures (measure lines, chording, note durations, note groupings, etc.) onto the raw note-strokes. Once that was done, the program could then utilize this information in making a reasonable guess about further refining the symbolic representation—which could, in turn, be further adjusted and corrected by the user.

Simple synthesizer keyboards were already available, and although MIDI (the now standard musical instrument-to-computer interface) hadn't yet been devised, it was simple enough to connect a keyboard to a Dorado in such a way that all note-strokes could be measured in great detail and the instrument's sounding apparatus operated by a program. By the time that was done, John had some elementary software ready to try out. In addition to

deciding on the underlying data formats (which would determine what kinds of user features would be practical), we had discussed which features to include initially and as John worked away, I stood back and cheered. Every few days more features came alive, and then we would discuss what to do next. Over the course of the summer of 1980 I saw my dream of over two decades gradually become a reality as John built one of the first applications on the Dorado utilizing its new operating system. He dubbed his program "Mockingbird," and it was a triumph. It was enabled not only by John's extraordinary cleverness, but also by the Dorado that was far more powerful (especially in its graphics capabilities) than anything previously available. In the fall, we gave an enthusiastically-received, PARC-wide lecture demonstration. Among other things it showed off the Dorado itself for the first time. A few months later, after finishing a few more features, we made a demonstration videotape.

Today Mockingbird is recognized as a pioneering classic. Although modern personal computers are far more powerful than the Dorado, some of the fundamental ideas embodied in Mockingbird have yet to be adopted in the many commercial music software products that have appeared on the market in recent years. Part of the reason for this is that most of these products are designed to cater to a marketplace in which many of the complex features of classical music are irrelevant. We, on the other hand, were working towards a tool that we felt would be useful to the next Beethoven when he comes along.

John, Severo, and Mockingbird

Chapter 21

The beginning of the end; CPSR and nuclear wars; clerical errors and the end of an era

In truth, my story is drawing to a close. In some ways it was over by the time I'd arrived at PARC. Except for the explosion of the Internet, the world into which I then stepped, though confined within PARC's computer labs, was already surprisingly little different from the computerized world we all inhabit today.[45] Indeed there are some important technical differences that make today's computers more accessible and usable, but the Altos that were then spreading throughout PARC were—with the exception of size, cost, and speed—very little different in principal and mode of use from today's personal computers. I had seen it all begin with the LINC and had followed the development (albeit with some side trips) through to the point where large commercial interests would now grab the ideas and run with them, smoothing the rough edges, making the devices smaller, cheaper, and faster, but adding very little that was fundamentally new. Proliferating applications seemed to be where the future lay.

[45] Granted, we had to share access to printing and large file storage facilities which today sit respectively on my desk and inside my computer.

In 1980 Ronald Reagan was elected president, and within a short time I began to fear for my life. I had been concerned for many years about the threat posed to future generations by the growing stockpile of nuclear weapons, but here was a president who didn't appear to have sufficient comprehension of the implications of these weapons. The Soviet leadership seemed equally unaware and blasé, and it began to appear entirely possible that a serious nuclear blunder might be committed by one side or the other. The threat seemed to be moving inexorably closer in time.

In October of 1981, in response to what seemed like increasingly irresponsible statements by then Secretary of Defense, Alexander Haig, I sent an email message to the PARC community, expressing my concerns and announcing the formation of a new net discussion group. Not surprisingly, there was immediate response. A vigorous discussion ensued that ended the following year in the formation of an organization that came to be known as Computer Professionals For Social Responsibility (CPSR). Over the next few years Laura and I would be preoccupied with building and running that organization and attempting to put it on a stable financial footing. Although we are no longer directly involved, it is still alive and well today, over twenty years later.

When Ronald Reagan made his famous *Star Wars* speech, some computer scientists climbed on the bandwagon where money would be plentiful and where there would be challenging engineering problems. Others, looking at both the technical problems and the threat posed by an increased arms race, decided that the whole enterprise was a dangerous and provocative undertaking. The *Strategic Defense Initiative* was the subject of intense

debate within the computer science community and sadly even caused rifts between colleagues. Public debates, many arranged by CPSR, raised the issue to national prominence and pitted scholars and scientists against powerful forces with vested interests. There were many detailed arguments having to do with the need for such rapid response that one was forced to rely on extremely complex and therefore untrustworthy computer systems for *launch on warning*. Several of us wrote a book *Computers In Battle—Will They Work?* I like to think that our small organization contributed to limiting the extent of the project, although probably the cost and repeated experimental failures have proved far more effective than our meager efforts. Unfortunately, like so many defense programs, once begun, Star Wars seems likely to continue indefinitely. It has great appeal for those legislators who hope for technological solutions to problems, and it is lucrative for the participants. These two factors seem sufficient to guarantee it perpetuity, and indeed, as I write this a new president, George Bush, is pressing forward with development of such a system, ignoring the advice of scientists and concerns about violating the international Anti Ballistic Missile treaty.

In the summer of 1982, as Laura and I were preparing to depart for a climbing trip in the remote mountains of British Columbia, I got a call from the personnel office at PARC. They announced that through a clerical error (blunders everywhere) I had been overlooked in an offer of early retirement, but that in fact I qualified—and did I wish to take it? Good Lord, I said, how would I know? I explained that the car was packed, that we were planning to leave immediately for a month's vacation, and that I couldn't possibly answer such a question on the spot. Since the oversight had been theirs, I was allowed to postpone the

decision, and as we drove away I said to Laura "Guess what?"

As we drove north we discussed the situation. The years in which I had been able to make a significant contribution were nearing their end. Many of the big questions about machine architecture had been explored and the whole computer field seemed to be maturing to the point where commercial exploitation, which had never much interested me, would increasingly dominate the scene. We had been fortunate to ride the crest of a wave of exploration and innovation for nearly thirty years, an unbelievably exciting period. Surely it was a good time to "leave 'em laughing." In fact, we had intended to retire within the next few years in any case. We wanted time to explore the world's mountains before we were too decrepit to do so. We weren't sure whether we could afford retirement at that juncture—I was only 51—but decided that we could probably manage it if we sold our house in the bloated Silicon Valley market. So from a dusty phone booth beside the road in the middle of nowhere in British Columbia, we called a startled California real-estate agent and told her that the key was under the mat, could she please try to sell our house while we were gone. Then we climbed onto a helicopter and proceeded to forget the rest of the world while we enjoyed mountain climbing for the next month.

When we returned home we found that there were interested potential buyers for our house, so I called the personnel office to inquire when I finally had to make a decision about retiring. An embarrassed pause ensued, after which I was told that *another* mistake had been made, that I actually *hadn't* been eligible after all! But, as I explained shortly thereafter to George Pake, in the course of the

month away, my mind had, in fact, decided to retire. After a brief negotiation it was agreed that I would remain for one more year and then take early retirement.

When I told Bob Taylor what had happened, after exploding all over the personnel department, he said simply "What will you *do*?" He, who had such fierce dedication to a mission, could not comprehend that, despite my apparent enthusiasm, I'd been merely dabbling in the computer field and had numerous other interests and concerns[46]. For me the years working with computers and computer design had been a delightful game that brought me in contact with many extremely bright people. I'd played the game with great energy, and even at times with considerable conviction, but there were many other things that I found equally compelling. By then I was deeply involved in the struggle to combat what I saw as various forms of nuclear lunacy, and I'd always wanted to be able to dedicate more time to musical endeavors.

At the same time a more local war was shaping up. For many years there had been friction between Bob Taylor and his boss, George Pake, the director of PARC. Bob's missionary zeal rendered him all but unmanageable. Furthermore, he had a large, gifted, and dedicated following within his laboratory that gave him tremendous leverage with any superior who might have serious disagreements with him. After years of struggling with Bob, with whom he naturally didn't always agree, George moved on and away to become director of research for all of

[46] Taylor himself retired a good many years afterwards and has since been awarded the National Medal of Technology for his many contributions to computer science.

Xerox. A new director, Bill Spencer, was put in place at PARC and, discovering Bob's intransigence, he almost immediately made the fatal mistake of trying to force Bob to "behave." Bob promptly resigned. Spencer may have been somewhat startled by that, but he had no way of anticipating the revolt and mass exodus that would ensue. Some of the rest of us, however, foreseeing this likely eventuality, decided to pay a visit to the president and chairman of the board of Xerox in order to voice our concerns and plead for their intervention—alas to no avail. The die was cast and as we predicted, within a very few months the lab began to empty as, one after another, people joined Taylor in forming a new Systems Research Center for DEC in downtown Palo Alto. Within a year most of the senior members and many of the junior members of the Computer Science Lab, arguably the best computer lab in the world at the time, had disappeared from Xerox.

It was the end of an era and seemed a terrible waste at the time. In retrospect, however, perhaps it was fitting. PARC had made truly extraordinary contributions to the computer field during the 1970s, an act that would have been nearly impossible to follow. Since I was on a retirement track, I stayed on through most of the good-bye parties until the time came for my own departure. Although I'd decided to retire, I nonetheless toyed briefly with the idea of joining my chums in their new digs, but ultimately decided it was time for a new and different life, pursuing other, long-postponed interests.

A bright note was struck by the fact that almost exactly the day I officially retired, the 25th anniversary celebration of the birth of the LINC took place in Washington—the LINC, which had been so much the crucible in which I was truly formed as a computer scientist. Laura and I attended

the celebration where we were delighted by a talk given by Alan Kay and bored stiff by a speech given by Margaret Heckler, the then secretary of Health, Education, and Welfare. which was sponsoring the celebration and belatedly trying to claim its share of the credit for what had happened. Most of all I was thrilled to re-encounter numerous old LINC friends whom I hadn't seen in many years.

Chapter 22

A review of the bidding

There is considerable debate about who "invented" the personal computer, and when. There is no simple answer to such a question and the definition of the term itself has evolved over time. Lots of people contributed ideas over many years as increasingly close approximations were constructed, eventually culminating in the machines we know today. Those who receive most of the publicity are those who made such machines widely available and made the most money. And indeed if by "personal" we mean that most people can own one, then certainly general availability and affordability are important features. As a final exercise, let me trace the story backward in time, pointing out what I see as the principal links in the chain of dependencies on which present personal machines rest.

What ultimately made it possible to sell computers at a price most people could afford were developments in integrated circuits, in particular the development of the microcomputer chip in the late 1960s. Until then the minimum cost of anything that one could call a computer was several tens of thousands of dollars—way beyond the reach of most individuals. Computers were also physically big, needed air conditioning, and often required the ministrations of technicians. The advent of the Intel 8080 chip provided, for the first time, a core computer building block that was affordable. Other, more advanced chips followed in rapid succession.

Almost as soon as these chips appeared, hackers commenced using them to begin putting together things that were certainly personal and certainly computers, but bore no resemblance to what we today call a personal computer. Little machines, most notably the Altair, consumed the attention of hackers, but they were strictly playthings for nerds and most people would have had no interest in such gadgets. Nonetheless by the early 1970s, some of the hackers, Steve Jobs among them, were able to start selling small, relatively inexpensive computers that could sit comfortably on a desk.

But interacting with these early small, really inexpensive machines was still extremely arcane and cumbersome. Before personal computers could spread beyond nerdville, they had to become both more usable by, and more useful to, ordinary people. That required standing on the shoulders of those who had pioneered user accessibility and user applications. By the mid 1970s, PARC had developed and propagated the Alto within its walls. All of the important functional and user features of today's personal computer were manifested in that machine. Xerox, however, failed to take advantage of what its researchers had done. The reasons they so profoundly dropped the ball are explored in the books *Fumbling the Future* and (more recently and more thoroughly) *Dealers of Lightning*. [See Bibliography]. Whatever the reasons, they certainly missed the boat, and one day Steve Jobs found his way inside Xerox PARC where he saw what was missing in Apple's early machines. He shortly hired a key Alto software developer (Larry Tesler) and quickly adopted the PARC concepts into his machines, whereupon Apple was up, up and away, heading fast towards the Macintosh.

But where had the PARC ideas come from? The researchers there had helped to define what such a thing as a personal computer might be. It was a matter of defining the capabilities and method of use that would make a computer broadly useful and appealing. Initially these researchers had to work with the relatively big, expensive, clunky components that were available to them, foreseeing that the parts would eventually become sufficiently small, cheap, and fast that computers could become accessible to everyone. The researchers not only brought together ideas from many sources, they also made enormous contributions themselves. Probably the two most important of these were the decision to embed the screen image in the computer's main memory and the development of local area networking. Capitalizing on these key architectural features, they proceeded to implement a graphical user interface and to develop prototype versions of numerous applications that would surge through the personal computer markets over years to come.

But many of the ideas of interactive use through mouse and screen that came out of PARC and that we all experience today, were refinements of concepts and techniques pioneered by Douglas Engelbart at SRI in the 1960s. Lacking any such personal machine as an ALTO, Engelbart and his associates had developed their ideas using a specially doctored-up Time Sharing system. In 1968 Engelbart gave the historic demonstration of his work that permanently altered the minds of his contemporaries and laid the foundation for much of the ensuing work at PARC.

During the mid-1960s, although some of the smaller machines were shrinking in size, their cost, for most people, remained out of reach and Time Shared use of giant machines dominated the research scene. These machines

allowed a limited amount of interactive use through non-display terminals, but their more important contributions consisted in the exploration of memory management schemes and multiprocessing. We have to go back all the way to 1962 to discover the earliest machine that was designed specifically for individual use—the LINC.

Although he clearly believed that computers would one day become small, cheap and fast, I doubt that Wes Clark, or anyone else at that time, envisioned today's personal computers with any precision. In 1962, a personal computer meant something that an individual researcher (not a homebody) controlled and could casually turn on and off like other pieces of lab equipment. It also had to be within plausible reach of a typical lab manager's budget which, at the time, meant on the order of $25,000. While he was at it, Clark threw into the mix a collection of features that would make machines useful far beyond the medical research community for which the LINC was originally conceived. The display screen and control knobs were primitive instances of today's screen and mouse that clearly suggested interactive use. LINC tapes were the original forerunners of today's diskettes, floppies, etc., upon which personal computers still depend. All in all it was to prove a stunningly prophetic design, but despite its seminal rôle, only a handful of people today have ever heard of the LINC.

And of course the LINC did not come totally out of thin air either. Its predecessors were TX-0, TX-2, and Whirlwind which, despite their enormous size and cost, could arguably be dubbed "personal" computers. Unlike most other machines of the day, that were either doing batch-processing or, later, Time Sharing, TX-2, when I knew it, was used for long periods by individual programmers. This

was at Wes Clark's insistence, and only over his dead body (i.e., after he left Lincoln) was TX-2 outfitted with a Time Sharing system.

To me, these then seem the major steps in the long process that has led to today's personal computer. You will note that although PCs, or derivatives (clones) thereof, now dominate the personal computer field in terms of numbers, in this review I have altogether failed to mention IBM. As happened on so many previous occasions, they were practically the last to "get it." Personal computers were fundamentally anathema to IBM thinking, so it is hardly surprising that IBM climbed onto the personal computer bandwagon late in the game. The initial success of PC's was due not to any significant conceptual contribution, but rather to IBM's giant size which swamped everyone else once they entered the market. But it was only with considerable reluctance, and the assistance of Microsoft, that a semi-reasonable user-interface eventually come to inhabit PCs. Their later profusion arose from the fact that the PC was made an open system for software developers and of course from the numerous clones that have arisen. Finally, to complete the circle, it's been suggested that IBM was rescued from the dustbin of history only by the millions of dollars of government money that poured into their coffers from Air Force contracts associated with SAGE in the 1950s. Sic transit, and all that.

Epilogue

In the summer of 1999 Laura got a call announcing that the following year there was to be a mathematics conference in Berkeley honoring her father, her mother, and her grandfather, all of whom were number theorists associated with the University of California at Berkeley. Many years ago, in searching for ever larger prime numbers, her father, D.H. Lehmer, constructed a number of devices known as "sieves" whose purpose was to mechanize the search by using techniques well known to mathematicians for bypassing factorable numbers. Today such algorithms are either programmed for fast digital computers or built using electronic devices, but prior to ENIAC there were no electronic devices available for such computation. Instead these early sieves were semi-mechanical devices made out of gears, bicycle chains, "electric eyes" and other unlikely, Rube Goldbergian paraphernalia. One machine that used strips of movie film punched with a conductor's punch (for those who remember trains and trolleys), spools from sewing thread, etc., had to be lubricated with baby powder and thus became known as the "Babychine." These ingenious machines bear a superficial resemblance to some of Babbage's early attempts to build a mechanical computer, and indeed these *were* computers, albeit of a rather specialized sort.

Today they reside in the Computer Museum History Center collection at Moffett Field in Mountain View, California. Thinking that it would be nice to have these machines present at the forthcoming conference, and even

operating if they could be resuscitated, we visited the History Center, together with Wes Clark who happened to be in California at the time. At present the History Center is more like a warehouse where things are lined up in "visible storage," awaiting funding for a more gracious setting. As we walked down the aisles, past dusty pieces of ENIAC, old SAGE machines, ancient CDC, IBM, and GE machines, we were struck by the enormous cleverness and the variety of ideas that were tried and later abandoned as the technology marched forward, obsoleting one after another of these conceptions. And yet, here, like the Neanderthals, were the manifest steps that had been necessary for that march to proceed.

The sieves had been set out for our inspection and we considered how they might be safely transported the fifty odd miles to Berkeley for the conference. One in particular, built with a large number of heavy steel gears, was encased in a metal box held together with nuts and bolts that looked as though they might have been useful in constructing the Golden Gate bridge. That one apparently weighs nearly a ton and gave us considerable pause. But others, the Babychine and the bicycle chain machine (which had been previously replicated by a graduate student for a master's thesis), appeared to present no problem.

Along one aisle we came upon a "classic" LINC, sitting patiently beside less familiar neighbors. Perhaps because of its physical size, or perhaps just because of its age, it was not located, as it should have been, together with later personal computers—an Altair, an Atari, an Alto—and we pointed this discrepancy out to our hosts. This brought home yet once again the need for reminding people of the historic significance of the LINC as the earliest computer designed explicitly for individual use and embodying

primitive versions of most of the features that make today's personal machines so useful.

Seeing the Alto reminded me that many years ago I'd been responsible for sending one from Xerox-PARC to be placed on display in the Boston Computer Museum (now absorbed into the Science Museum). As the machine was strictly for display and didn't have to work, I'd unearthed a problematic chassis and told the technicians to stuff it full of boards that they'd been unable to repair. There were plenty of those and they'd filled it up indiscriminately, putting memory boards and CPU boards and whatever they had into random card slots. We'd bundled the whole thing up and shipped it off to Boston.

About two years afterward I got a call from someone who announced that he was trying to get the machine running (!) and did I have any drawings that I could give him? I couldn't help laughing at the image of the poor guy struggling to comprehend a machine thus thrown together from stray broken parts, and I finally managed to persuade him that it was a futile effort. Since then a working Alto has apparently found its way into the Computer History Center as Altos have taken their place alongside earlier computers as memorabilia.

In 1980, while John Maxwell was putting the finishing touches on Mockingbird, Laura and I trekked around the Annapurna range in Nepal—a 250-mile circuit that includes the traverse of a pass nearly 18,000 feet high. Toward the end of the trip, near the village of Ghorapani, lies Poon Hill, so named because, at that time at least, it belonged to a Major Poon of the Nepalese army. On top of the hill the Major had erected a most remarkable structure, a whimsical wooden tower boasting two rickety stairways—one presumably for up, and the other for down. From this tower

the view to the north includes some of the highest and best-known Himalayan peaks, among them, the Annapurnas, Dhaulagiri, and Machupuchare. Turning to the south, you look down toward India over the plains of southern Nepal where lesser mountain ranges disappear into the distance.

Five years later, building our home in the hills above the Pacific, we decided to call the place Poon Hill in honor of the view which faintly resembles that to the south from Poon Hill in Nepal. Later, in printing my father's music, I used the imprimatur *Poon Hill Press*. A couple of years ago one of our friends, wondering where the name Poon Hill might have come from, searched the web and found references to Nepal. Thinking these irrelevant, he then tried Poon Hill *minus* Nepal—whereupon his screen filled with the titles of music I'd published that were listed at the American Music Center's web site.

I have not yet built my own a web site, and the older I get, the less I rush to embrace the latest technological fashions. Next week an old college friend of mine and I will set out to climb a mountain in the heart of the Sierras. It has been suggested that it might be prudent to take along a cell-phone, just in case. But I can't imagine doing such a thing—a large part of the purpose would be lost. Today's generation is accustomed to being "connected" at a level that is foreign to us old-timers. As always, something is gained—and something is lost.

After I retired in 1983, we pulled the plug—disconnected from the network and remained in electronic darkness for nearly a decade, during which time the ARPANET metamorphosed into the Internet. Meanwhile my son David, who years before had dropped out of high-school, was becoming a rising computer jock. Eventually he persuaded me that getting reconnected was in order, but by

then I had little idea how to go about it. He must have had one of the more gratifying days a son can experience when he appeared one afternoon with a modem under his arm. Within a couple of hours he had me on-line and had given me an intensive course in networking. After giving me one final now-let-us-review-what-we-have-learned lecture, he shot off to his next appointment, leaving me feeling that the baton had been securely passed.

Reflecting on the preceding decades, I have come to realize what a remarkable time it was. When we retired, we wondered what it would be like to be no longer working. That was nearly twenty years ago, during which time I've learned that the enemy is within; that all that scurrying about was in the nature of the beast and that retirement could only alter its directions, not diminish its intensity.

Now that I've come to the end, I can see that, after all, I have failed to achieve the principal thing I set out to do—to bring you, the reader, inside of the process so that you could feel what it was actually like. It was an impossible goal, of course. One wishes so deeply to communicate at that level—to share the actual experiences that have moved one throughout life. But in the end all one can do is to describe the externals without really penetrating to the heart of the matter.

It is particularly difficult when one is describing a process that is, by its nature, largely unfamiliar to most people, where often the most dramatic moments are utterly internal and not manifested in any external way whatsoever. Sitting in a chair, chewing on a pencil, mulling over a problem, suddenly there it comes, the insight you

have been waiting for that solves today's problem so elegantly. It's the experience of the beauty and the excitement of such moments as well as the gradual unveiling of where it might all be leading that one wishes to convey and to share—the feeling of exhilaration as one achieves understanding, comes over the next ridge on the way to the summit. But analogies never really capture the flavor or the feeling that comes with a particular experience.

These misgivings notwithstanding, perhaps by describing some of my own experiences and reactions, I have been able to convey some of the differences between the styles and motivations that existed earlier and those that obtain today. And I hope that in the process I have managed to transmit a whiff of the flavor of discovery and revelation during those now bygone years and to provide some new insights into the origins of the surprising little devices that we have come to take so much for granted and that have so dramatically altered the lives of us all.

Appendix I

The Synchronizer Problem

Most computers are "synchronous" devices, that is they operate based on a clock that ticks regularly. Changes take place in the state of the machine only at clock ticks. What happens on a given tick depends on the state of the machine produced by the previous tick. At each tick, information about the changes that are produced must percolate everywhere throughout the machine before the next tick comes along—otherwise outdated or changing information could produce incorrect behavior. The maximum clock rate is thus dependent on the nature of the circuits, the wire lengths, etc., that is, all of the things that determine how fast signals travel around inside the machine.

But what about signals that come into the machine from outside devices (such as disks, mice, printers, modems, etc.) that operate at their own times, quite independent of the machine's clock? Consider, for example, a disk feeding data into a computer's memory. The data comes off the disk at times that depend on the spinning of the disk, unrelated to the timing of the computer's clock. Suppose a new piece of data has just arrived and is ready to be read into the memory. Several different parts of the machine need to participate in taking in the datum—the next instruction must be delayed, memory pointers must be reset, etc. If the disk simply holds up its hand at an arbitrary time with respect to the computer's clock and says "Datum Available," then, if the hand happens to go up at about the same time as the clock ticks, some parts of the machine may

see the signal and start to process the datum, while other parts of the machine may not notice and may blandly proceed with the next instruction. Chaos will result.

This much was well understood and to deal with it a "synchronizer" was normally employed. On each clock tick, the signal that says "Datum Available" would be sampled. If it was found to be on, then a "Datum Ready" signal could be turned on. Because "Datum Available" is sampled *at only one place in the machine*, it would either turn on the "Datum Ready" signal or, if it just missed it on one tick, it would turn it on at the next tick. Because it comes on only at a clock tick and not at some random time, it will have settled down and be stable (either on or off) by the time of the ensuing clock tick. It can thus be used throughout the machine to bring about the processing of the datum.

This standard solution appeared to work and had been used for many years. But it depended on the tacit assumption that the device that stored the "Datum Ready" signal would either be On or Off at a given clock tick (depending on whether the "Datum Available" signal had been noticed or not on the previous clock tick.) It was generally believed that the devices used to store such signals (flip-flops) switched between their two states in a short, specifiable time—far shorter than the interval between successive clock ticks. What we found was that under certain circumstances that switching time could not be depended upon. In particular, if a flip-flop was activated by a marginal signal (such as occurred for example when the "Datum Available" signal happened to turn on *just* as the clock was ticking), then rather than turning over crisply in the usual way, the flip-flop might stall in an in-between state for much longer times than the specifications indicated—in fact, theoretically, for *arbitrarily long times!*

These special circumstances were statistically very rare, happening only when the coincidence of signals was extremely unfortunate. But we were able to demonstrate that once in a great while, the "Datum Ready" signal, supposedly rock-solid by the time of the next clock tick, could itself still be wavering in indecision and could thus produce erroneous behavior.

This was shocking news and meant that most systems had a vulnerability that had hitherto been unrecognized. As the speed of machines (their clock and data rates) had increased, the problem started to show up with increasing frequency. We suspected that many unexplained computer failures of the sort that are never traced but that don't appear to re-occur after the machine is restarted, were cause by synchronizer failures. As recorded above, I encountered precisely this situation when working on the beginnings of the ARPANET.

It is interesting to note that when we first discussed this failure mode, there were a number of serious computer scientists (some from the sacrosanct halls of MIT) who argued that it simply could not happen. Those of us who had *observed* it, not only knew that it could, but understood that under the right circumstances it was inevitable. It was shocking to find so much resistance among supposed scientists.

Mackie describes the scene as follows:

"Tom and I made it our mission in life to convince the world that there was a real problem here, and spent the better part of a year taking on any comer who claimed he could produce a glitch-free circuit that could resolve the arrival order of two asynchronous signals unambiguously (without using a "Trinary Flop-flop," as we called the concoction). We referred to such designs, publicly, as

"move-the-glitch" circuits. In private we disdainfully referred to them as "perpetual motion machines." In every case we were able to determine to where, in the design, the ingenious designer had moved the glitch. In each case, the glitch persisted, and the presenter was brought into alignment with this new "correct" view of the universe. Our fervor approached that of religious zealots. It was our mission to stamp out anti-glitch apostasy wherever we could find it.

"You might recall an ally we had in this effort. I'm sure you met him. He was a truly mad Englishman (I believe he worked for Motorola, but I could be mistaken) whose name now entirely escapes me. Tom and I met him at a conference in Chicago. He had written a paper on how this "arbitration" problem, which is inherent in all digital computers, would eventually doom the design of truly large scale machines, as the error rate caused by these random events would become so high that it would eventually become intolerable. His article saw no solution to the problem. We convinced him that there was indeed light at the end of the tunnel.

"Hence the second paper which dealt with the solution to the problem, the building of what we called, at that time, an "Interlock." It showed how using one or more boxes containing a simple tri-state flip flop ("Zero," "Wait," and "One" states), one could build a system that would arbitrate the arrivals of any number of asynchronous signals with non-ambiguous outcomes.

"I remember an embarrassing incident surrounding the first demonstration of the Trinary Flip flop Interlock. Charlie had invited those of us who had been doing some of this 'moonlighting' (we did this in addition to our daytime work of building macromodules) research to

present it to the entire team. So on this particular day I made a demonstration of my Trinary Flip-Flop Interlock design with no small amount of hubris. But for some reason it wouldn't work properly. I stared at it, but I couldn't figure out what was wrong. Then Mish, who had been watching intently, pointed out that I had wired it up in reverse, and by simply swapping the connections everything would work properly. The connections were reversed and everything did indeed work as advertised. I felt both humbled and pleased at the same time. I've certainly never lost my admiration for Mish who was able to smoke that one out so nimbly. The true industrial-strength Synchronizer that is in use today was, of course, eventually designed and built by Charlie Molnar."

Appendix II

The Bit-Map Display

The basic device used in most computer display systems for many years has been the cathode ray tube (CRT)—the same basic device that underlies your television set[47]. Here, roughly, is how a CRT works. At the rear of the tube sits an electron gun capable of firing a stream of electrons toward the front face of the tube (the face being the part you look at). The back surface of the front face is coated with a material that glows briefly when electrons strike it. By controlling the aiming of the gun while turning the electron stream on and off, one can paint a picture on the front of the tube. Depending on the particular kind of coating used, an illuminated point will glow for a shorter or longer time after being irradiated. The faster an image fades, the easier it is to present rapidly changing images (as in television), but the more frequently the points need to be re-illuminated in order to avoid flicker, or even fading, of the image.

These basic facilities have been used in a wide variety of ways over the years. In one method, the gun is used as a graffiti painter uses a spray can. The beam is turned on and

[47] In recent years various alternative display technologies are used, particularly in situations where size and weight are critical (as in laptops and other portable devices). However many CRT-based displays still occupy desks and counters around the world and no doubt will for some years to come.

off and it is moved about, just as the graffiti painter turns the spray on and off with the finger-tip while continuously redirecting it. This produces a series of illuminated lines against an otherwise unlit background. Of course the picture must be displayed repeatedly or it will shortly fade from view. If a more slowly fading coating is used, then problems will occur in depicting changing images. However, the principal problem with this scheme is that the computations of aiming directions for many kinds of images are extremely complex, and thus most systems that have used this technique have been limited to handling segments of straight lines, for which the computation is relatively straightforward. Many early display systems, including the SAGE consoles, used this sort of "vector" display—sometimes in conjunction with character-forming masks.

In an alternative usage, an image is created by displaying a sequence of dots to be brightened, each of whose X and Y screen coordinates can be held in a table in the computer's memory or provided by specialized instructions. Whirlwind, TX-2, and the LINC all utilized this basic "point" display technique (with various frills). Only the points to be illuminated needed to be addressed; the remainder of the screen remained dark.

Today computer screens present an image the way television does, by repeated scanning of the entire screen while the electron beam is brightened or dimmed as the scanning proceeds. To understand the scanning process, consider how you read a book. Unless you are a speed reader, you start at the upper left-hand corner of the page and scan the first line, left to right. You then reposition your eyes to the left end of the next line down, which you again scan from left to right, etc., on down the page. That, in

essence, is how a television screen works. The gun is aimed at the upper left corner, and then sweeps rightward across the top of the screen. As it sweeps, the beam is brightened or dimmed so as to "paint" the topmost line of the desired image. When the gun gets to the end of the first line, the beam is turned off while the gun is repositioned to the left end of the next line down, whence it again sweeps rightward, painting the image's second line as it goes. This process continues until the entire screen has been painted by a succession of horizontal lines, at which point the beam is turned off, the gun is repositioned to the upper left corner, and the entire process repeats—over and over again. This process is called refreshing the display, because the screen quickly goes blank if the process stops. It all happens at unimaginable speed: The entire screen is painted in this way *several tens of times each second*. At such speed, a human is unable to follow the detail of the process, and instead sees the picture in its entirety. If the picture changes on successive repaintings, we see movement, just as we do in a movie made by rapidly projecting a sequence of still pictures. All such devices rely on the perceptual limitations of our human visual system.

For normal television, the intensity of the beam, as it paints successive lines, is supplied by a television camera simultaneously scanning a live image (or by a previously recorded signal of that sort). In a modern computer screen, the bits representing the image are stored in the computer's memory where it can be readily manipulated by programs and whence it is repeatedly used for refreshing the display as described above. It takes a lot of memory to store an entire screen image—there are many, many pixels (picture elements), each of which occupies a bit in the memory (or several bits for gray scale or color). For many years the cost

of memory was far too great to allow such extravagance. It was only when the cost of memory began to fall that this sort of use became economically feasible.

Bibliography

1. W. A. Clark and C. E. Molnar, *A Description of the LINC*, Ch. 2, *Computers in Biomedical Research*, R.W. Stacy and B. D. Waxman (eds) Academic Press, New York, N.Y., 1965.

2. Mary Allen Clark, *China Diary*, Washington University Magazine, Fall 1972

3. S.M. Ornstein et al, *Computing in China: A Travel Report*, Science, Vol., 182, 1973.

4. Kent C. Redmond and Thomas M. Smith, *Project Whirlwind; The History of a Pioneer Computer*, Digital Press, 1980

5. S.M. Ornstein and J.T. Maxwell III, *Mockingbird: A Composer's Amanuensis*, Byte Magazine 9-1, 1984.

6. W. A. Clark, *The LINC Was Early and Small*, in *A History of Personal Workstations*, Adele Goldberg ed. ACM Press, 1986.

7. S. M. Ornstein, *Computers in Battle: A Human Overview*, Ch 1, *Computers In Battle—Will They Work?*, D. Bellin and G. Chapman (eds).,Harcourt Brace Jovanovich, New York, N.Y., 1987.

8. K. Hafner and M. Lyon, *Where Wizards Stay Up Late*: *The Origins of the Internet*, Simon and Schuster, New York, N.Y., 1996.

9. S. Sellager, *Nerds 2.0.1: A Brief History of the Internet*, TV Books, New York, N.Y., 1998.

10. S. McCartney, *ENIAC: The Triumphs and Tragedies of the World's First Computer*, Walker and Co. New York, N.Y., 1999.

11. M. Hiltzik, *Dealers of Lightning*, Harper Collins, New York, N.Y., 1999

12. J. Naughton, *A Brief History of the Future: The Origins of the Internet*, Overlook Press, NY, 2000.

13. Leo Beranek, *Roots of the Internet, A Personal History,* The Massachusetts Historical Review, Volume 2, 2000

14. M. Mitchell Waldrop, *The Dream Machine,* Viking Press, 2001

INDEX

About the Author

In the late 1950s Severo Ornstein worked at MIT's Lincoln Laboratory, then at the forefront of computer research. In 1961-62 he participated in the design of the LINC, the world's first personal computer. He was later responsible for the hardware design of the IMP, the computer that handled messages for the Arpanet, forerunner of the Internet. During the same period he taught computer design at Harvard, and in 1972 he organized and led the first delegation of computer scientists to China. In 1981, together with his wife, Laura Gould, he founded Computer Professionals for Social Responsibility (CPSR), an organization concerned with the role of computers in society.

Computing in the Middle Ages is designed for the lay reader who wishes to understand some of the background of the computer revolution. It provides an easily understood and amusing account of what took place in computer research between the 1950s and the 1980s. The achievements of those days were later exploited by companies like Apple and Microsoft, which brought personal computers to the consciousness of the general public.

During that era — when both the design of computers and expectations about the ways in which they could be used were undergoing dramatic change — the author was "in the trenches" where seminal experiments were taking place, first at MIT and later at other universities and research centers. His unassuming story — a breezy and irreverent memoir enlivened by amusing anecdotes from his professional and personal experience — gives a human dimension to the otherwise dry and often obscure process of scientific and engineering innovation. Developments are brought to life and explained in terms that can be understood by anyone. Along the way you'll meet a number of memorable characters who, although often overshadowed in the public mind by entrepreneurs, are widely recognized as pioneers in the field of computer research.

1st Books Library

ISBN 1-4033-1517-5
90000
9 781403 315175