Computer Connections

People, Places, and Events in the Evolution of the Personal Computer Industry

Dear

Kristi

This is the numbered limited distribution manuscript that I have produced for my new book entitled *Computer Connections*. As a manuscript, it contains numerous errors although I've tried to reduce them to a minimum.

I have provided you with this for your perusal, and would appreciate any comments if you run across errors.

This is the "Christmas, 1993 Edition" that will go to print in final form early next year, apparently under the auspices of Osborne-McGraw Hill.

Please do not make copies or pass this edition along to anyone else other than your immediate family and associates. Thanks, and

Merry Christmas,

Gary

Gary Kildall

Computer Connections

People, Places, and Events in the Evolution of the Personal Computer Industry

Gary Kildall

Prometheus Light and Sound
3959 Westlake Drive
Austin, Texas, 78746

To My Friends and Business Associates

Please Respect That
This Manuscript is
Confidential and Proprietary
And Do Not Make Any Copies

Any Corrections or Comments are Greatly Appreciated

CHAPTER 1 *One Person's Need For a Personal Computer* 9

Kildall's Nautical School 9

CHAPTER 2 *Seattle and the University of Washington* 13

Computer Life at the "U-Dub" 13
A Blunder 14
Transition to Computers 15
Getting into Compilers 16
Linguistics and Compilers 17
The Computer Chronicles 19
Booting a Computer 22
Bootstrapping the Industry 22
Systems Programming for that B5500 24
The Age of Timesharing 24
The Computer Center Corporation 26
The New U-Dub Computer 27
Going to War 28
Going from the UW to "War" 29

CHAPTER 3 *The NPS Helps Foster Microcomputing* 31

Life at the Naval Postgraduate School 31
How to Program the IBM System/370 33
How to be a Professor at NPS 33
Back to School at the U-Dub 35
A Story About Light Bulbs 36
Departing from "Cush" 37

CHAPTER 4 *Silicon Valley Micros* 39

New Technologies 39
A Small Ad 40
An Introduction to Intel Corporation 41
The Age of Intel's 4004 Microprocessor 42
Using a Teletype Model 33 42
Grace Hopper's 4004 45
Bugs in Your Programs 46
The Age of the Intel 8008 Microprocessor 46
Noyce and Computer Games 47
The NPS Microcomputer Lab 48
A High Level Language for Microprocessors 49
Toward Self Hosted PL/M 50
The Age of the Intel 8080 51

CHAPTER 5 *The Advent of CP/M* 53

The Quest for a Fast and Cheap Storage Device 53
Origins of CP/M 55
The Astrology Machine 56
Jim Warren and DDJ 58

CHAPTER 6 *The Early Hardware and Software Suppliers* 59

Omron, Altair and IMSAI Corporation 59
The Early Days of Microsoft 61
The West Coast Computer Faire, SSG, and the Electric Pencil 63
The Softcard 64
Battle of the BASICs 65
The Culture at IMSAI 67
ED Goes on the Big Screen 68
Beginnings of Micropro 68

CHAPTER 7 *Digital Research 71*

The Early Days of Digital Research 71
The Day Intel Went Flying 72
CP/M and Software Pricing 72
Launching the New DRI 74
PL/I and New CP/M Versions 75
Why the Heck PL/I? 76

CHAPTER 1 *One Person's Need For a Personal Computer*

Kildall's Nautical School

My grandfather Harold was a seafaring man and ran the Seattle to Singapore route as Chief Mate of a 1920's steamship.

Lumber shipment from Everett, a port north of Seattle, to Singapore was a lively trade. After the longshoremen finished loading Grandpa Harold's ship, he noticed a mongrel dog pacing the pier. That mongrel decided quite boldly to board grandpa's ship. He named the dog "Hector."

Hector became grandpa's sidekick on the voyage to China, even lying beside him at the midnight watch while Harold scanned the horizon and worked navigational positions of his ship in crossing the Pacific Ocean.

Grandpa's freighter arrived at Singapore harbor and Hector became agitated. On impulse, Hector jumped overboard near a Sampan rowing by. Well, his real owner was riding on that Sampan and scooped that mongrel from the water. Two weeks earlier, it seems Hector's master had sailed from Everett on his own ship's schedule.

Hector was nearby that day of sailing at Everett, but certainly not aboard his master's ship, and that sailing could not be delayed. Hector's little dog-brain recognized the situation, and scented grandpa's ship that was going to the same place as his lost master.

The story appeared in *Reader's Digest* in 1957. I always liked this story, and it was true, and grandpa told us many others like it. This is a book about computers. Why tell a dog story? I promise that it will be relevant, but not for awhile.

Harold returned from sea duty in 1927, maybe partially because my grandmother, Gwendolyn, would go to sleep with tears nearly every night he was away. Grandpa started a navigation school in Seattle, first located at the University of Washington, then later moved to downtown Seattle. It was called Kildall's Nautical School and, for fun, it became known as "Kildall's College of Nautical Knowledge." He taught merchant seamen to become officers onboard commercial ships. His students became Third Mates, Second Mates, Chief Officers, Masters, and Pilots. Grandpa Harold taught in that school to within a week of the day he died at 92 and educated over 30,000 students, many of which could not multiply zero times zero when they started.

I must admit that although I am one of grandpa's admirers, he was a bit disorganized. The Coast Guard did all the exams on his students, so he'd quiz his students afterward on their test questions. After a few years, his nautical school had bits and scraps of paper nearly everywhere. These scraps recorded questions and answers, kind of like *Cliffs Notes*, but randomly unsorted throughout the school's office as if arranged by a small explosive charge. Still, the "college" was known as the best in the country.

My dad, Joseph, followed in grandpa's wake and worked his way to a Master Ocean license, the top rank in merchant shipping. World War II came about, and Joe started with the "college" as an Instructor. The school was valued by the government because Harold and Joe trained the skippers of the Victory Ships that dodged those Wolfpack U-boats to bring supplies to the war effort in Europe.

Dad took all those scraps of Coast Guard Q&A and put them into a voluminous card catalog. He had every question that the Coast Guard examiners could conceive, right there in that office. Not that the students were just prep'd on the questions, because there is no doubt that they were trained in every aspect of Marine Navigation and Engineering. The Coast Guard did not have a chance against the two. Harold and Joe's students knew their business quite well, indeed.

Being in the lineage, I was expected to continue the tradition of the Kildall School, which I did for a time in the early 60's. Grandpa, then slightly retired, lectured on Magnetism and Compass Correction and, on occasion, circled the cigarette smoke-filled room to check his students' progress as they slaved at finding the answer to yet another of the school's "exercises."

Grandpa, dad, and I were a good trio. To my personal benefit, I learned the mathematics of Navigation and Trigonometry, and spent extra hours canvassing the dusty, rare old books of the school's small library that covered such oddities as "How to Rebabbit the Bearings of a Double Acting Steam Engine."

To navigate you take "sights." A sight is a reference point to a celestial body, such as a star. You look through a sextant, find the proper angle, and then look at your chronometer. Next, look in the Nautical Almanac for that exact time and date. Use standard, but tedious, calculation forms to plot the result. That tells you where you are on the face of the earth.

Joe often described a "machine" that he wanted to build. "You punch-in the navigation data and turn the crank, and out comes the position of your ship." He envisioned properly-shaped cams and gears to do this, because he was a mechanical person by nature. It wasn't until the microcomputer was invented that the "crank" was truly feasible. It's a connection we'll pick up in 1972 at Intel Corporation in Santa Clara. Strangely, dad's dream formed a link all the way to today's personal computing and operating systems.

In the early 1960's, I worked at the Kildall "college," and became heir-apparent to the Kildall School throne. Instead, I decided to get an advanced education. This didn't set well with my father, because it caused a discontinuity in the lineage. Not only that, my high school grades were so bad that there seemed no possibility of entry to college.

However, I wrote a petition to the Regents and cited my work as an Instructor at the Nautical School. Because of the Nautical School's recognition, I gained entry to the University of Washington in 1963 by that petition, but by an entirely too close of a margin.

The Kildall Nautical School taught me processes that high school hadn't. Such as the ability to do mathematics of a sort and, most important, the mental tools to dissect and solve complicated problems, and to work from the beginning to the end in an organized fashion.

However, the practice of Navigation, in the traditional sense, was fast becoming obsolete. My father tried to nix my leaving the Kildall Nautical School, but the challenge of becoming a full-fledged teacher was too much to ignore.

I signed for the University of Washington. My intent was to become a high school teacher. I didn't ever teach high school. Now that I have passed through three teenagers and their studies, I believe that Divine Providence did me a personal favor. But, I worked very, very hard and got top grades. I think that's because most of my fellow students were party animals, and I, being a "greaser" in high school, had that out of my system by college. During those first years at the UW, I studied the basics of Mathematics. Meanwhile, I worked at Harold and Joe's Nautical School. It was hard, rewarding, and fun.

My father was a precise man. His handwriting was unscripted block perfect. His workmanship could not be excelled whether it were in the schoolroom or over a lathe in his workshop. He took on a project to build an 83 foot yacht when I was only sixteen, and it was sort of finished when he died thirty years later. I think the project was too big, but that was the way he lived.

CHAPTER 2

Seattle and the University of Washington

Computer Life at the "U-Dub"

Taking courses at any university is a problem for someone who didn't study that hard in high school. Ok, so I had some good background in mathematics from the Navigation School, such as fundamentals of trigonometry, but, hey, go take on calculus.

All those guys from the seventeenth century, like Keplar and Newton came up with some complicated stuff. They did it to figure where the planets would be. That made it easier to navigate. And, navigation let people do business, and it let them make money. There were two other subjects that I enjoyed, and these came into play in later years. My father's precision in drawing brought an interest in art. Pencil drawing became a part of my life and allowed me to promote and exercise the graphics tools developed by Digital Research in later years. I studied art at that time, and I enjoy it today. Another was history. Mathematics, art, and history. These are three topics that kept my interest. We all have some. And, they are the mead of one's existence.

In 1963, at the "U-Dub," as it was called, I found myself in the evolutionary transition between mechanical calculation and digital computers. My first class in Numerical Analysis in 1964 was from a professor in his eighties. This nice old gentleman was retiring, and not too soon in my opinion, as he seemed quite feeble and at somewhat of a loss of memory. He taught the trade of computing with Marchant calculators. A Marchant calculator is a big, black machine, about three times the size of a typewriter. It has a giant carriage on top with about ten holes with little mechanical numbers that pop up in the slots. There are buttons all over the front with digits, decimal points, and keyed symbols that multiply and divide to control this mechanical nightmare.

They used these Marchant calculators for Numerical Analysis, starting in about the 1940's. A business would hire rows and rows of Marchant calculator operators to compute so-called "Finite Differences." This led to tables of figures that let you, for example, toss a mortar precisely into your enemy's backyard.

You sat there and pushed those keys while the mechnicals made a host of noises by cranking and grinding who-knows-what, until the rattling stopped. You got a number. You put the number on a form, and eventually, with enough numbers and forms, you got a result. Sometimes the resulting number was correct.

These operators also made tables to figure where the Sun, Moon, stars and planets would be at a particular time, much like the tables that solved the navigation problems that we talked about eariler. In a sense, dad's navigation "crank" was somewhat inconveniently embodied in the Marchant with a "form" alongside.

A Blunder

By the way, here's a little story about Numerical Analysis. Numerical Analysis is based upon computing formulas by stepwise refinement to within as close a tolerance as possible.

What that means in English is that the more you push all those damn Marchant keys, the closer that mortar will get to your neighbor's rear door when you decide to launch it. So, if the real answer is to aim that mortar at a 41.34567 angle, and the Marchant operator computes it as 41.34566, you have an "error" of 0.00001. Then, you decide if that is acceptable (which it is not for bankers, but might be for a Sergeant, or Boeing Corporation).

But, for myself, I always liked the term "blunder" used in Numerical Analysis. A blunder, contrary to an error, means that you didn't push the proper keys on that Marchant or, possibly pushed them in wrong order, or you didn't write down numbers properly as they appeared. In short, you screwed-up. I like the word "blunder."

So, I make this proposal (with tongue placed properly at cheek level), that one should refrain from stating that "I have an 'error' in my program," for that would mean that one is within .01% of a working program that has no error. This makes no sense, because all engineering managers know that this .01% doubles the project length.

Instead, we engineers must from forever onward refer to a program that operates incorrectly as one that has a "blunder." Under interrogation as to the progress of your project you state quite objectively and in clear tone,

"Sir, my Device Driver does not yet operate, it has a Blunder."

Blunders are not errors. They are simply mistakes you made in writing your program. C'mon, your boss knows everyone has "blunders." Hey, he probably spilled a glass of wine on the hostess at that party last night for the company president.

Blunders can go on for years, and your manager will be completely confused, but knowingly accepting. A blunder. Try it tomorrow. Trust me.

Transition to Computers

Coincidentally, my study-student-friend, Dale Leatherman, stopped me outside our class a few days after that old professor retired. He opened his notebook and showed me an unpunched computer card. At the top was printed "FORTRAN STATEMENT." He told me that this computer stuff was going to be a "big thing." That was 1965.

Dale and I took courses together in computer studies, starting with assembly language programming for the IBM 7094. For you computer youngsters, here's the way it worked.

You walk into the keypunch room with your two-foot long cardboard box of FORTRAN STATEMENT cards. Then wait in line to get to a keypunch machine. Finally, someone tires of typing, and you get a seat.

Each card in that stupid cardboard box is like a single line on your computer screen today. I don't even want to think about it, because you young guys have it all too easy. But, my Computer Psychiatrist told me that "eet izz important to get zees feelingks about der dumbkopf keypunches out in der open." All us oldtimers remember that the keypunch machine is like a typewriter, but it shuffles cards through its mechanism and stamps rectangular holes in a column below each character. You leaf through your box, alter each necessary card, and then relinquish your keypunch to the next anxious student. Then, take your precious box of cards to the shelf outside the computer room and wait and wait and wait.

The computer room hides the IBM 7094 with SAC-like security doors. Computer operators, trained in "Grumbly 101," sit behind that door, attending to their work and throwing down coffee without even their own notice.

I know this because I spent about 1,312,467 hours looking through that little peephole window to see if the operator would arise from his or her throne to go to the restroom and possibly retrieve my box of punched cards on the way back into SAC.

If you stayed around very late at night, you might get a "turnaround" of an hour or so to get the printout. The program usually didn't work, so it was back to the keypunch. To know what it was like, go back to the heading for this section, "Transition to Computers," and re-read it until you fall soundly asleep.

Getting into Compilers

Let me start with a short note about why I write about compilers. First, they are my primary interest in computing. Compilers, when perfected, can be elegant to the point that you want to paste a printed listing on your wall, like artwork. Ok, so you have to be into writing compilers to get my meaning, but when your compiler works, you are very proud and want to show it off.

But, for most engineers, compilers are simply mind tools. Our minds are too limited to write complicated programs that do things like GEM, Windows, or D-Base without them. Compilers make programs that computers understand from programs that people understand. They are sort of like natural language translators who sit in a business conference and make English into Japanese.

The UW bought a new computer. It was a Burroughs B5500. This computer, an offshoot of the English Electric KDF-9, was unique. It was designed to run ALGOL (ALGOL stands for Algorithmic Language, an improvement over FORTRAN, and the precursor of today's Pascal language).

The B5500, although today severly outdated, is a "stack machine." That means that data is held in temporary memory that is built and discarded as its program runs (that's the way the "C" programming language works today). The B5500 and ALGOL were the basis for compiler design at Stanford University and the model for Bill McKeeman's work on XPL. And, XPL led to Intel's "defacto standard processor."

During my graduate studies, I used Bill McKeeman's XPL. It was called a compiler-compiler. XPL is a program that makes compilers. What you do is plug-in rules of syntax, much like grammar rules of English but constrained to the rigidity of computer languages.

Linguistics and Compilers

For me, linguistics was only word phrase games, like the word "wind" that can mean a stiff breeze, or what you do to your watch to make it keep time. (No, don't answer with "I change the battery." I'm talking about the old windup kind, Jeez.)

Noam Chomsky was a well-known Linguist who helped define the difference between "syntax" and "semantics" in natural language through his work on "Transformational Grammar." He studied natural language by allowing for sentences like this.

"He broke the window with his little sister"

(and, I always liked this next one)

"He fed her dog biscuits"

(or, how about this one that took me awhile to figure out),

"Time flies like an arrow"

Just like the word "wind," these sentences have multiple meaning, but the form, or syntax, is correct in English.

The connection is simple. Chomsky's theories and symbolism for natural language helped in processing newly defined computer language forms. Take a look at those phrases. The syntax is correct, but the meaning is ambiguous.

McKeeman's XPL processed computer statements, not English, but it's sort of the same idea. That is, XPL looked for proper form (syntax) in the computer language, not the meaning (semantics). In order to make XPL work, you added more ALGOL programming to cause the compiler to "understand" each statement that you parsed.

Unlike natural language, there can be only one meaning to a computer sentence. That's why you cannot just "tell" a computer what to do. It needs to process the exact meaning of each thing you say, in context. And, even in context, the meaning may not always be clear. Here's an example.

"Hey, you know old George. He's down near the crossroads. Yeah, well he done got a new Intel 486 with a Connors Drive and 16 Megabytes of RAM. An' old George just booted it up this mornin', don't ya know. But 'pears old George warn't too happy with it, 'cause he coundunt get that durn Microsoft Windows to work."

"Worse yet, George tol' me it near broke his left foot and scuffed his new pair of Lucheses that he'd been savin' up fur near a month or so, don't you know fersure. But, he's feelin' better now 'cause that 'durn 486 is stuck setting thur in that oak tree up yonder, an' it seems awork a mite better now, don't you know."

Get my point?

Ok, you might retort "hey, man, I've got a computer that understands English at home." Fine, its vocabulary of, say, about 1000 words must be trained by the operator speaking those words over and over. It's like having a baby that's started to talk. And, what it eventually understands is a severe subset of English, with vocabulary limited to very special functions and people, such as mom and dad.

Unfortunately English, and other natural languages evolved somewhat inconsistently and constantly change. That's why we have computer languages that adhere to very strict rules of syntax and semantics.

The Computer Chronicles

"Ok, Gary, why are you bringing up the *Computer Chronicles* in the middle of a discussion about languages and compilers," you might say. First, I wish to brag about my seven-year stint with this *FINE* Public Broadcasting Show (viewed by over a million people, syndicated throughout the USA on *every* PBS channel, I might add, and even shown in China, and ... Ok, I'll stop).

Wait..., wait..., don't stop reading. I mean, after all, I was a *STAR*! But, for reasons quite unfathomable to me, I was not asked to place my computer-worn fingers in that cement down on Hollywood Boulevard at Gruman's. Hmmm... could it be my acting? Someone wrote in once and said, "Gary is very good as co-host of *Computer Chronicles*," but it looked like my mother's handwriting.

Ok, so I'll make it relevant. I agree that the *Chronicles* is not in the timeline of our discussion, but a particular show one day demo'd how natural language doesn't match computer languages.

Stewart Cheifet hosted and organized the show, and I co-hosted which means "show-up Gary, on time, hey?" Stewart selected the computer companies to be taped in three segments for each show. He came up with the latest in modems, printers, PC's, games, and whatever else imaginable during those seven years from 1982 to 1989. However, there was that one "demo" that stood out, done by "Jim" from company "X" (the real names are withheld for a degree of sympathy). I want to tell you more about *Chronicles* itself before I bring out Jim.

Stewart, the crew, and I did twenty-six shows per year, taped in the studio at the College of San Mateo. The deal was that I drove from Monterey, California, to the college on a Saturday morning, arriving at about 9:00AM. We taped two shows on those Saturdays, so I needed to be there for thirteen Saturdays each year.

Now, work this backward. Hmmm... It takes two hours to drive from Monterey to San Mateo, so that means I've got to leave home at 7:00AM, and, since I must wear a suit, that translates to 6:00AM. This is *not* a good time on Saturday for a DRI Standard Issue Party Animal.

No big deal, we taped the "opens" in the morning, and I could make as many mistakes as necessary because our director, Peter, would open the intercom and gently ask "did you like the take?" Considering my oft condition, the answer was usually "no," and Stewart and I would do another pass at it.

I can say, respectfully, that Stewart nearly always made it through his thoughts on those opens, for we had no teleprompters. He did flub a few times in the seven years. I enjoyed it immensely when he did. I give him a 96% on those "opens."

After that, we rolled in the marketing managers and promoters to set up hardware, usually two or three at a time, and do Q&A and demos.

Stewart is a connoisseur of fine software and hardware products. And, being trained in Law, he could talk endlessly, without great effort during taping. But, I believe he made a singular error by selecting Jim for the show that day.

Jim came on the set with his "HAL 9000." I was, of course, curious. This appeared to be the name of the talking computer in Arthur C. Clark's movie, *2001*, but I gave no judgement. Jim's black box sat there before me on the set, with our director, Peter in the editing booth rolling tape. "Mr. Jim," I asked, "just what does your HAL 9000 do?" I awaited a response that I knew was coming, but sincerely did not want to hear.

"Yes, Mr. Kildall, my computer talks and thinks, just like the HAL 9000 in the movie *2001*. It uses Artificial Intelligence."

"Just like the HAL 9000?" Yes, "Just like the HAL 9000." Well, I was of course taken aback by such a wondrous feat of the time, as talking and thinking machines were quite out of my realm. Jim continued with, "My HAL 9000 talks to you, makes judgements, and comes to conclusions, all by itself."

If you've viewed the *Chronicles* show, you realize that the segments are short. So, we got right into it. "Mr. Jim, can we get a demonstration of this immensely intelligent machine that you brought here today?"

Jim wanted Stewart, a person with extensive broadcasting experience, and quite a clear speaking voice, to "talk" to HAL. Ok, so the cameras are still rolling, right? Stewart surrounded himself with a headset and microphone while Jim "booted" HAL.

"Stewart, say hello to HAL." Stewart, in his finest speaking voice obeyed and said "Hello." In fact, he did this about eight or ten times and got random responses in computer-trashed English, like:

"Igmm Very Sorgy, Plse Say it Agan," or "I can not ungderstand wat youg are sayging," or "Plse say that one morge tigme" or "Igmm Very Sorgy, Plse Say it Agan."

Well, I can say with little reservation that the demo was not going well for Jim or HAL. So, changing the subject, I touched carefully on the topic of Artificial Intelligence with Jim after that debacle to fill the segment, because I surely did not want to repeat it.

The crew was going to lunch, so we broke down the set, and I was left alone with Jim. He wanted to do the segment over, because he felt "it just didn't go right." I can't imagine how he could have come to this conclusion, do you?

Tape's not rolling, and the set's hot lights are off. I patted Jim on the back, and assured him that it would show well, but he should get HAL's feelings on it to make sure.

I have not talked with HAL since, but that demo was the biggest waste of video tape, computer and TV cable time that I had ever seen. But, for the good, Jim and his friend, HAL, showed that natural languages and computers are not a good fit, and that was worthwhile.

I'll never forgive Stewart for setting up that segment; although I will allow a bunch of other PBS programming transgressions over the years of the *Chronicles* that are private to Stewart, the crew, and myself. The events will not be publicly disclosed unless under severe bribery, for which I show a weakness at times. I had fun with those guys at KCSM.

Oh, by the way, if you ever happen to watch re-runs of the *Chronicles*, take a look at the background. It makes us look like we're located in a building at San Francisco overlooking the Bay. Most "guests" were quite surprised to find us in the basement of the College of San Mateo studio on a set with a fake window and picture of the Bay. I'm not complaining. It was quite a nice set, with the exception that the Transamerica Building is shown half-finished in that picture background and remained so for the seven years that I did that show.

Booting a Computer

Ok, so let's get back to the PC industry.

I don't know about you, but the word "booting" a computer seemed a little strange when I first used that B5500 at the UW. To "boot" you turned on the power of that room-sized computer and stuck-in a single punched card into the card reader. The card had a little itty-bitty program that read another program from disk, and that program read another, and so-forth, and that started the whole computer.

So, the origin of the phrase "to boot your computer" comes from the word "bootstrap," which means to "pull yourself up by your own bootstraps." Today, this is the small built-in program in a computer system that lets you load your operating system.

But, "bootstrapping" has also been used to describe the process of writing a compiler for a language using the language itself. It's just like writing ALGOL in ALGOL, or "C" in "C." And, it's hardly different from describing English in English.

Where do you start bootstrapping? Consider this. "C" is written using "C," and there was a programming language before "C" called "B," and "B" was written using "A." What do you think "A" might have been? Hmmm... assembly language perhaps?

Bootstrapping the Industry

Compilers for computer languages, like FORTRAN and COBOL, were made to help take away the pain of assembly language programming. They make computer statements in their strict rules of syntax into the instruction set of the computer.

Computers in the 60's didn't have much main memory, and "assembly language programmers" could do a better job at fitting memory than automated compilers. So, high-level languages supported by those compilers were not considered efficient even by the late 60's.

There is a direct trace from the two Burroughs B5500s at the UW and Stanford University through to compiler development in XPL for the Intel chip set, and on to CP/M and DOS.

High-level languages like ALGOL on the B5500 changed the trend of using assembly language for "efficiency." For example, you really couldn't write a B5500 assembly language program unless you were a real technonerd. In 1966, the UW hired me to maintain the ALGOL compiler, and that old B5500 became my learning machine. I saw a ton of sunrises over that Computer Center.

It was a great personal computer. I'd just plant a sign that said "B5500 Down For Maintenance" at about midnight and open shop again at 6:00 AM for student programs. Ok, so that's not fair, but that was a long time ago, in a galaxy far-far away, so don't blame me, 'cause I was just a kid, Ok?

During this "maintenance," I'd take the whole B5500 system down, "reboot" and play in the B5500 ALGOL sandbox. In my own defense, I did, on occasion, actually fix some errors (sorry, "blunders") in the ALGOL compiler. However, I seriously doubt that I ever deserved all of those paychecks.

I will personally attest that this ALGOL compiler was, for the day, an elegant piece of code. ALGOL was written in ALGOL, and this got everyone's attention. thirty years ago that was a neat hat-trick. How do you write a compiler in its own language? You have to have an ALGOL compiler to write the ALGOL compiler, right? Chicken and egg.

Solution? Write a small compiler using another computer, such as that KDF-9. This little program compiles code for the B5500 small ALGOL subset. Then, you start writing the ALGOL compiler using that subset, gradually improving the compiler step-by-step. Chicken and egg again.

It's called "incremental improvement," and that's how all complicated programs are written. You just gradually improve to reach your goal in small steps. That's how humans solve problems. Hey, it's not such a big deal. After all, that's how Mom Nature solves problems as well.

Systems Programming for that B5500

Locally, it was only Dick Hamlet and myself there to take care of the B5500. Dick is a very astute computer person. He is a tall, sandy-haired, bushy-faced guy with freckles and piercing eyes. He did not tolerate incompetence whether it be intellectual or otherwise, but rewarded you with a smile when you sparked his interest.

He understood the B5500 Operating System like no one I ever knew, and he gave a lecture to the new Computer Science Group at the UW where he described the thickness of a listing as a measure of the complexity of an operating system. I never forgot that line, because it was so often true. I always used that measure to determine the output of my own engineers in later years.

I learned much from the architecture of the B5500 computer. In particular, I learned about data structures for organizing disk drive information.

In later years, I extrapolated from these B5500 data structures to create the first personal computer operating system, which I named CP/M.

The Age of Timesharing

There was a new technology in the works. It was called Remote Access. At the UW, we used a data communications setup attached to the B5500. Only the privileged Oracles could hook up. There were eight ports tied to the B5500.

Dick Hamlet had a direct connection to the B5500 from his office, which I coveted dearly. There was another in the computer room. Some officials had the others, and I was never allowed entry there. But, they never used the lines, so it didn't matter anyway.

A modem was connected to a third port. I got the university to buy a portable acoustic coupled modem for me. This let me make a phone call to the B5500. "Big deal," you say, "I do this every day using my 2400 baud built-in modem, connected to Compuserve, Prodigy, Dow-Jones, and places that I don't even know I'm connected to...."

Cut me some slack. Hey, that was almost thirty years ago. This acoustic coupler, or "modem" as we call it today, was grey, and about the size of a shoe box with two holes in the top to hold the handset, surrounded by foam padding that came ajar immediately after shipment from the factory, which, by the way had usually gone out of business directly after the sale.

Hook your teletype to the acoustic coupler. Then make a phone call to the B5500. Stick the handset into the coupler. Ok, then you "log-in" using your secret password that everyone seemed to know anyway. If you're lucky, you see the sign-on:

```
          University of Washington B5500 Remote Access, Version 1.0

*
```

The asterisk was the "prompt" to tell you to type more stuff. But, quite frankly, that 110 baud acoustic coupler failed more often than not, and the message appeared like this

```
          Uni#44sityof Wash,:gton B5522 #@%%8<< version?[5 z
```

Oh, great response Mr. B5500, and this is only at 110 baud with a stupid phone stuck into the coupler with foam cushions falling out everywhere.

So, you stuff that phone and foam back into that lame coupler, a little harder this time, and try again, most often viewing a similar response and being not slightly entertained by the random letters and symbols that the coupler, the B5500, and AT&T had collectively selected. Dial again, and hope for the best.

I'm sorry, dear reader, that I got a little worked up over that. Phew..., Ok.., ok,.. I'm really Ok now. I have occasional episodes of "Early Remote Access Shell-Shock." My Computer Psychiatrist tells me that the effects will decrease with time once I come out from under my computer desk when I try to use Compuserve.

The UW Computer Center bought one of those workhorse Teletype Model 33's. It's a heavy, mechanical device that, by observation, shouldn't operate any more than a Boeing 747 should be able to fly.

I built a grey wooden box for that teletype and went around showing how remote access to a computer system over telephone lines works. Or, I should say, didn't work. I went to Seattle Pacific College in 1966 with my modem and teletype encased in its package on rollers.

The demonstration was typical. I couldn't log-in. The computer was down, or for some other reason didn't respond. Following that, I never gave a live demo again. But, Dick Hamlet believed in remote access to computers through timesharing and formed a company to provide that sort of service.

The Computer Center Corporation

I mention the Computer Center Corporation, or C-cubed, as it was called, because it predicated an interesting series of events involving Bill Gates, Paul Allen, and myself.

Dick Hamlet, Monique Rona, Carl Young, director of the UW Computer Center, and my undergraduate professor David Dekker started C-cubed. It was one of the first timesharing computer companies in Seattle and used Digital Equipment Corporation's DEC PDP-10s. C-cubed, located near the UW, was in business for only a short time and folded.

C-cubed depicted the mentality of the late sixties. No one wanted to manage punched cards, nor did they wish to wait at the operator's door in favor of a sampling of program output. Instead, timesharing became the watchword. Remote access to the B5500 sparked Dick, Monique, and David's interest in creating the C-cubed business.

There were two kids from Lakeview High School who broke into Dick's DEC PDP-10s and stole passwords. This way, they could hook up to Dick's computers from a remote teletype and use it for free. The two kids were Gates and Allen.

Don't vilify Gates and Allen for this activity. Breaking in was not that hard. It was like stealing candy from the corner computer store. And, it was theft of the very basic sort.

Dick was one of the best systems programmers I had ever run across, and it didn't take him long to discover the theft of time on his machines. He found the culprits and cleverly allowed Gates and Allen access to C-cubed, so that he could recode and test the operating system to help prevent illegal access.

Dick explained how Gates and Allen did it. It's fairly simple. They "logged-in" to his DEC PDP-10 system from a remote location using a valid account and password. A sample limited account was given to Lakeview students by C-cubed, mostly through Monique's interest in the school. Well, the DEC PDP-10 used virtual memory that swapped RAM to disk. This left data in memory pages that were given to other programs.

Dick's machine wasn't heavily used, so most of memory had operating system data structures like passwords, still lying around in memory. Gates and Allen just dumped "pages" of reassigned memory to their computer terminal and watched for things that looked like passwords. Once they found a few, they simply used them at no cost to themselves. The reason that I consider this theft is that I, even being a very good friend of both Dick and Monique, could not afford to pay C-cubed time-sharing charges, although, I sincerely wished to do so.

Dick found the Gates and Allen scam when a customer complained about a bill for services when, in fact, no services were rendered. He fixed it by zeroing memory pages when they were swapped out and released. That took extra computer time but stopped the banditos.

I was attending the UW then. I noticed a couple of kids programming at C-cubed one day. They were Gates and Allen. I ran into them then, and again several years later.

There was not enough commercial interest in remote access to computers around Seattle in those days. Besides, Boeing Airplane Company was going broke, and, as a result, so was Seattle. C-cubed was hit squarely by this and folded. In those late 1960's, you could drive south on I-5 and, stuck almost in front of you, was a huge billboard that read

"Last One Out, Please Turn Off The Lights"

The New U-Dub Computer

That old IBM 7094 computer was getting a bit tired in 1965. So, the university tasked the Computer Science team of professors, headed by Dr. Hellmut Golde, to find a new one.

I, now being Hellmut's Ph.D. student and general lackey, was, in turn, tasked with creating a set of "benchmark" programs. The Computer Center normally pumped-out FORTRAN programs for the students, so I made several boxes of sample programs and put them on magnetic tape. That was our benchmark. Our UW team went to IBM at Yorktown Heights and then visited CDC and General Electric for throughput timing using our test.

The CDC 6400 severly won that benchmark and ran about two to three times the capacity of the IBM System/360. Hellmut and the team made the decision, and the UW contracted for the CDC computer.

Three top IBM salesmen immediately approached the President of the university, Dr. Charles Odegaard, and told him of the error of his ways, and that he would be humiliated by the decision he had made, and that the decision should be reversed.

Odegaard listened patiently to the IBM salesmen and, when they were completely done, immediately sent them packing, with contracts flying loosely about in the room. I respect Odegaard for not succumbing to the IBM "end run" that was so prevalent in those years.

But, I got a call while at the UW "Hub" from my then wife Dorothy. She said I had received a draft notice. Uncle Sam wanted my bod for the Viet Nam War.

Going to War

The Viet Nam War was unfortunate for me personally, and for the world in general. Being selected to serve in the Army wasn't particularly appealing because I had been offered the opportunity to enter the first Master's Degree Computer Science class in 1967. There were to be twenty of us in this first class of CS there, and a letter from the Draft Board wasn't really in the grand scheme of things.

I went to the draft center, like many of you did, and was probed by the government doctors in the chilly line of recruits. One doctor asked if I knew that I had a heart murmur. Relieved at the possibility of exemption, I said, "No," but it turned out his question was rhetorical. The doc asked everyone that one. I was sworn in to serve in the Army. Damn, all of a sudden visions of rice paddies flew through my head.

Of course, I tried a quick pass at the Draft Board to convince them to allow me to get to a master's degree. They laughed and told me there was not a chance, "you're going to war, young man."

Well, my dad had friends who were captains in the Navy. I know you're not supposed to use connections, but, quite frankly, I didn't want to get shot at.

Dad connected me to one of his buddies, and I joined the Navy with his recommendation. That satisfied the Draft Board. Not only that, I got a reprieve to finish my master's degree while I worked toward my commission as an officer.

It took two summer sessions at Officer Candidate School at Newport, Rhode Island in 1967 and 1968 to complete my training to become an ensign.

When I got there, I didn't like the training one bit. They cut off all my hair on the first day. It took all summer to grow back. They worked us. Guys dropped out. I told myself to never look back on OCS and think it was Ok. It wasn't. It was a bummer. I was destined to become an officer on a destroyer tossing shells into the forests of Viet Nam.

Oh, by the way, this is a neat story about OCS. They had a course in navigation that we took daily. Well, of course, no one knew that I had been teaching it for years. So, when the weekly quiz came, I finished it momentarily after going over the problems twice. I'd turn it over to the instructor, and he'd visibly take notice that they were always all correct answers. Good old Harold and Joe at that Nautical School had something to do with those test scores. I graduated with the Technology Award, and nobody knew why.

The Navy allowed me to continue my education to the master's degree in 1969. During that year before graduation, I taught seaman apprentices how to win their data processing stripes. We had weekly Navy Reserve meetings at an ex-WW-II building on the south of Lake Union in Seattle.

These were horribly boring events. I headed the data processing squad, and we skipped-out right after colors and drove to the Naval Station at Sand Point, just off Lake Washington. There they had the oldest IBM collators, sorters, and machines programmed by inserting plug-in wires into "patch boards." We all got through Reserves, and my guys all got their stripes. Most important, we didn't have to march all evening between "colors" like the other guys.

Going from the UW to "War"

Graduation day came in December of 1969. I was to receive my M.S. with honors, for which I was very proud considering my high school grades. However, the prospect of a destroyer tour as a lowly Lieutenant JG seemed a dismal prospect.

But, the President of the university, Dr. Charles Odegaard, had been impressed by my work in their new Computer Science Department. Recall, I had been the technical advisor in selecting their new CDC 6400 computer. Dr. Odegaard was on the Board of the Naval Postgraduate School at Monterey, California. Then, Captain Williams of the Reserve Officer Training Corp at the University of Washington, who I had never met, called me for his counsel.

The Captain asked me to take a seat opposite his desk. You vet's know that it is very important that you take the seat that the Captain tells you to, and I obeyed, assuming a position of seated attention in that chair. Captain Williams, in his khaki regulars, looked me squarely in the eye. "Mr. Kildall, I understand that you are graduating in Computer Science." I was not aware that President Odegaard had made a private recommendation to the Captain.

"Mr. Kildall, you have a choice to make." Well, the Captain proposed that I could either be an officer on a destroyer off the coast of Viet Nam or take the post as an Instructor in Mathematics and Computer Science at the Naval Postgraduate School at Monterey, California. This particular question made me understand the length of a microsecond. "Well, sir, I would like dearly to serve my country in battle, but I think I shall take the second option, if you please."

Captain Williams warned that if I taught at the Naval Postgraduate School, I would probably not reach the level of Admiral. I took a pensive stance for a moment and then told him that I would accept that risk.

CHAPTER 3

The NPS Helps Foster Microcomputing

Life at the Naval Postgraduate School

The Naval Postgraduate School, generally referred to as the NPS, is located at Monterey, California. It is a very good school in all respects and placed at a magnificent oceanside locale. Although I have not had the opportunity to make a visit to Viet Nam, I am quite sure that Monterey has a nicer coastline.

They teach only postgraduate courses at NPS. The students are mostly U.S. Navy Military, with a smattering of foreigners. The students are dedicated. They have to be. Some of the foreign students were said to be put before a firing squad on return to their homeland when they failed. I don't know if this was true, but I was a bit more lenient on them because of that.

Strangely, the NPS contributed significantly to the rise of microcomputing in the early 1970's.

Breaking from the UW's B5500 computer mainframe was tough for me. Recall that I had the B5500 to myself for hours at a time. The B5500 had been my rather large personal computer. Now, I had to use that IBM System/370 at the NPS, just like any other programmer.

Not only that, it didn't have any good high level language. So, here I am, stuck with a stupid System/370, no high level language, and no direct access to the computer. This was a technonerd nightmare.

But then, Old Captain Williams' words rang in my ears, and I took particular solace in the fact that I was not floating over the swirling seas of the Gulf of Tonkin with no computer at hand whatsoever.

Ok, so first you make friends with the computer operator at NPS's Computer Center. Then, make friends with the systems programmers. I met a nice lady, an astute programmer, Kathy Strutynski, who was later instrumental in programming upgrades to CP/M, and with relationships between Microsoft and DRI many years later.

The NPS accorded me a few specific tasks. First, I was to teach computer courses. Second, I was to be on the "Crypto Staff," and third I was to take inventory of liquor at the Officers' Mess.

I enjoyed the teaching with a passion, but the crypto duty over that three years resulted in only one message that transmitted a notice to an Admiral "that a free turkey was due to arrive on Thanksgiving." It took about an hour for me to decode that one.

I never could figure the inventory. But, there was a little lady named Mabel Johnston, who, at about her 75 years of age, could figure it quite nicely. She patiently showed me journals of numbers in column after column because I, being the Naval Officer In Charge, was required to verify her numbers to the penny.

Honestly, I just watched her and believed her additions, subtractions, allocations to inventory, taxes, and on and on and on. That was a risk, but after all, you must believe in a grandmother, right?

Mrs. Johnston knew the ropes. Years later, "Visicalc" on the Apple took some of the toil. Do you think that Bricklin's Visicalc made a substantial contribution to the success of Apple's PCs? I think so, because if I could have had Dan's spreadsheet at the time, I'm sure I would have impressed old Mrs. Johnston. But, somehow, I don't think she would have taken notice of any sort. She would've just moved to the next page in her ledger.

How to Program the IBM System/370

Here I am at the Naval Postgraduate School in 1970. Ok, better than Viet Nam, but not better than the U-Dub's B5500. The NPS had just installed a System/370, one of the first delivered by IBM.

The IBM System/370 was an enigma for me. No languages. No access. I started with Nicklaus Wirth's PL/360, a sort of high-level assembly language. It wasn't any good. Wirth went on to define the Pascal programming language which was a considerable improvement. Faced with no reasonable high-level language software-tools, I went to PL/I, IBM's runaway attempt at displacing FORTRAN and ALGOL.

How to be a Professor at NPS

As for my role at the NPS, some considered me a hard teacher. I don't think I was. A bunch of students said I was. So judge for yourself.

I often taught the Data Structures course. The class was called CS3111. To combat boredom, one semester on a first day, I entered the classroom and told the thirty-odd students that "we will now have a pop-quiz." Everyone must place their books and notes on the floor and they, being good Naval Officers, did so.

The problem that I proposed was to write a program that would "perform symbolic differentiation and simplification of polynomials to any order."

Well, I can tell you right now that was a huge order. And I, being professor-in-charge, may have barely gotten into a solution within that fifty minutes. Cruel, but I wanted to get a line on how my students thought about the *problem* of problem solving. I didn't really expect them to get to a solution of any sort whatsoever.

Those students worked away at that problem for fifteen minutes, at least. Then I said, "stop what you are doing now. I want you to think about how you are solving this problem, and what mind tools you are using at this moment."

Of course, the whole thing was a ruse, and the students were completely relieved, because not one could come close to a solution on that first day. I asked how their thought processes were working, and several of those students came up with good and different ideas on problem solving.

Ok, so I had this pop-quiz, right? The course proceeded through the semester, and I didn't give any other tests or quizzes. We worked entirely on problem solving, using Knuth's algorithmic notation and McCarthy's LISP List Processing symbolic programming language. At the end of the semester, I gave a final exam. There was only one question.

I don't think was being a mean professor. Do you?

Why did this group of students after two more semesters with me as their professor, post a finely emblazoned plaque at the NPS's Trident bar room that stated all too clearly:

"Class of 71, Kildall CS3111, CS3112, CS3113, Three Time Losers"

Ok, the guys did get their ultimate revenge. According to military protocol, I was required to usher the graduating students to their seats in King Hall. Although I was allowed civilian attire during classroom sessions, my ushering duty required my officer's dress uniform.

I had to let my "Three Time Losers" leave ten minutes early from the classroom, just before their graduation, to allow me time to change from my civvies to my officer's uniform. These were the same guys that I had taught all those three semesters. They knew that Kildall had to usher, and they enjoyed every minute. They knew, that by rank, I must cater to them.

I detected a few snickers here and there in the process. I have kept all their names in a special folder and intend sometime in the not-to-far-off future to reevaluate their grade scores as a basis for retracting a master's degree, or two. After all, I have kept all of their old tests, and I may have made an omission of some sort. Right?

It was a difficult study for these officer students, but, in the end, everyone graduated with their degrees, so I guess it couldn't have been all that bad for us.

But, in early 1972, my three-year obligation to the Navy was over. And, just as I had refused Captain William's option to go to Viet Nam, I declined both a "re-up" and the option of Naval Reserve meetings in a moment that may have been shorter than that "microsecond" that got me to NPS in the first place.

Back to School at the U-Dub

I kept my association with the NPS after my stint in the Navy. The Computer Science group at the NPS asked that I return as a professor, but I felt the need to complete my education.

Dr. Hellmut Golde at the University of Washington Computer Science Department had allowed that I continue work toward the next degree by "remote" from NPS while in the Navy, so I was prepared to finish my Ph.D. in fairly short order.

My research interest centered in compiler code generation. Recall, that this means that a compiler reads computer language statements and makes it into another program that the computer understands. Take, say a program with 1000 statements and make a computer readable form of it. The computer readable form might take 400 bytes of RAM. A good "optimizer" might reduce this by 25% to 300 bytes. This saves storage and generally makes the program run faster.

The name of this analysis is "Global Flow Optimization." You figure out how data values will vary or stay stable at each decision point in a program. Then you use that information to make small programs instead of big ones. That's called compiler code optimization.

Well, there was quite a bit of theory out there. Some was by John Cocke at IBM, and Harry Huskey at UC Santa Cruz, for example, and some was by Jefferey Ullman who was well-published in the Mathematics of Computer Science. But, I thought I had a process for flow optimization that was a general theory, covering nearly all of the current techniques.

A quick program in McCarthy's LISP convinced me of the correctness of my process, and I then spent several months with McKeeman's XPL system making a production compiler. The process worked, but I couldn't create the mathematics to back it. I just knew that it worked, but that's not enough. This all happened in 1971.

But, then I developed an algebraic model of my process. This so-called "algebra" is not like the one we study in high school, but instead is a mathematical modelling theory that lets you describe sets of objects (like program data values), and formulate ways to combine these objects (like through program branching and joining).

Well, my formulation was good, but I couldn't prove my central theorem, and that made my approach worthless as far as academicians were concerned. That was a basic problem, because, after all, I was trying to get the Ph.D., right?

I got out of the Navy in early 1972, without a proof of my theorem, and returned to the University of Washington, but my brain was about broken by then. None of my professors could help, and some were beginning to believe that my central theorem wasn't in fact true at all. This was not encouraging.

A Story About Light Bulbs

Ok, so I'm at the UW and I couldn't discover my proof. I just sat and sat and sat in my UW grad student office, resting my head in both hands, until my eyes shut by themselves late that evening. Nothing.

Then, in an instant the proof came to me. I wasn't even paying attention to it. I awoke in an instant and wrote the entire proof of my central theorem, not finishing until sunrise. I guess that's why they put light bulbs over cartoon characters. The discovery of this proof was one of the grandest experiences of my life, except, of course, for the time I visited Niagara Falls One Day.

It was a general theory, and it was quite elegant, if you will allow me to say so. Jefferey Ullman was a professor at Princeton University and picked up the thesis almost immediately after it was published. He liked it and elaborated on the theory in his own books.

Jeff is an astute Computer Scientist. He invited me to lecture at Princeton, which I did, and then to lecture again when he moved to Stanford. He is now Chairman of the Computer Science Department there. Jeff is an easy-going sort, and a great contributor in promoting the mathematics of the science of computing.

The NPS kept a civilian position open at the school, and I returned there with the rank of Associate Professor. Eventually, they gave me tenure. Tenure, as most of you are aware, means that you have a job "forever" or unless you die first. That is quite a nice deal in education, especially since the charts showed that NPS was second only to Stanford in salary in the entire nation. Not only that, education is one of the greatest scams concocted by mankind. Here's how it works.

You arrange to start your class at 11:00AM (9:30 wake-up, eh?). NEVER schedule more than one class of a different topic on the same day, for this makes you prepare two different lectures in your allotted ten minutes before each class, and if this rule is violated, it can be stressful and confusing in the classroom.

And, You allow your classes to be scheduled for at most three days of the week, although two is preferable. On all other possible occasions, use lowly, rarely paid "lab assistants" to off-load your heavy pressure.

You get holidays on every conceivable occasion, like Spring vacation, President's day, all summer, Christmas for a month, and then throw in Ground Hog's day for dessert.

Oh, yeah, I forgot that you must "Publish or Perish, right?" Well, that's easy enough. Get the university to buy you a Macintosh (say "sir, the portable, if you please?"). Then, on holiday in Hawaii, get a good ole' charge on them there batteries and make up an Industry Opinion Newsletter of your own! Publish every month, make a few bucks, and call it something like, "A Wholistic View to the Future of Computers in the Next Decade." See, this works. It's kind of like Nostradamas, 'cause you don't really have to be specific; you make your money over the first few years, 'cause you've got a DR. on the front of your name, so people sort of believe you.

That Newsletter also prompts invitations to speak that sometimes pay up to $1000 per evening, plus a free meal (don't ever speak at a symposium where they don't pay for everything, including the airfare - you can try for First Class, but I never got it).

I apologize to you readers who are teachers by profession, but c'mon, it's kind of a cushy job at times, right?

Departing from "Cush"

Leaving this sort of job situation to move into industry was one of the hardest decisions of my life. So, I just didn't make it. I'm good at that.

Best of all, the NPS faithfully encouraged one day of consulting work in the Silicon Valley area each week to bring the latest info back into the NPS classroom. I shall remain eternally grateful to the NPS powers-that-be for that particular edict.

CHAPTER 4

Silicon Valley Micros

New Technologies

Ed McCreight came from Seattle. He joined the budding group of researchers at Xerox PARC, the Palo Alto Research Center. Ed knew about my thesis work and invited me for a visit in 1972 on one of my "consulting" days. Alan Kaye was there, and they showed me their work. One was Alan's SmallTalk that used high-resolution graphics and windows.

I was nonplussed by their new computer design, called the ALTO, but their white background terminal with black character fonts of a wide variety drew my attention. I didn't know it at the time, but this was to foster the entire desktop publishing industry.

PARC's new technologies were severely interesting, but having no background in graphics, and having an intense interest in compilers, I thought of their efforts as futuristic oddities.

A Small Ad

Back at the NPS, I had a co-teacher named Gerry Barber. Gerry was tall, thin, pale, and balding at 25. He spoke with a lisp that you accommodated in time and rode his bicycle to school. He wore horn-rimmed glasses and pens lined up in a little case that he carried you know where. He swooned in class one day, and I had to revive him and take him to his home.

Gerry read every journal. He gleaned a small advertisement from *Electronic Engineering Times*. It read "Intel Corporation offers a computer for $25." Well, considering that the NPS IBM System/370 cost about $3M, I thought that was a pretty good deal, and Gerry agreed. This was mid-1972. And, by the way, I'd never heard of this little chip company, called Intel, at the time.

No matter. This is Econ' 101. Ok, the chip was only a 4-bit computer, called the 4004. And, it really wasn't $25, unless you bought about 10,000 of them. The cheapest "development system" was about $1000, and that didn't include the Model 33 Teletype that nearly even completely worn-out, ran about $700.

Still, that was a lot less than the IBM System/370 with a list price of three mill', and certainly not within the budget of an up-and-coming Associate Professor with a wage of $20,000 per year.

Intel sent me the specs for the $25 i4004 chip. This was a very primitive computer by anyone's standard, but it foretold the possibility of one's own personal computer that need not be shared by anyone else. It may be hard to believe, but this little processor started the whole damn industry.

The 4004 was fallout from a project Intel did for Busicom, a Japanese calculator manufacturer. Instead of building a specialized chip for Busicom, they designed a single-chip microprocessor controlled by a small read-only-memory (ROM).

Think of the mentality of the time. You used a timesharing system that varied in its response depending upon the time of day (those of you who use Compuserve or Prodigy know what I'm talking about). These little 4004 computers promised constant response time; although, the problems they solved were about as big as the chip.

There in 1972, My dad's navigation "crank" had arrived in the Intel 4004, but there appeared some *major* programming work to get the crank to actually work.

Intel built a little foot-square blue box with a PC board on the top. The PC board included one of those little 4004's. You could write programs with the "development system," and I had to have one. It was called the SIM4-01.

Oh, by the way, I enjoyed working with Gerry Barber. He was an honest, religious, intelligent man. He was astute.

An Introduction to Intel Corporation

I can say without reservation that the instruction set of that $25 Intel 4004 chip was the worst ever. Its only redeeming value was that it was dirt cheap in quantity, and that drew designers to use it in tons of dedicated applications like portable inventory data collection and specialized electronic monitors.

It was impossible for me to come up with $1700 for the SIM4-01 development system and teletype, so I wrote a simulator on the IBM System/370 that faked the operation of the 4004. Then, I went to work on dad's navigation crank. Well, this turned out to be quite a problem, because navigation requires computing "trig" functions, like sines and cosines, and that little processor had none of these, for sure.

I spent a few months writing trig functions for that little processor using my simulation on the IBM System/370. I didn't actually own a 4004, I just used my simulator on the System/370 to check and debug all of my trig functions.

All my 4004 programs came together, so I called Intel to see if I could scam a development system. I ran into Bob Garrow at a time when he was a fairly well-promoted engineer at Intel. I tried to sell him on the notion of trading my 4004 simulator for one of his $1000 SIM4-01 development systems.

Being a hardware nut, Bob didn't want the simulator. But, he did want the trig routines, because they were the beginnings of a "library" for the 4004, of sorts. Hey, I didn't care, I just wanted a SIM4-01.

Bob always had a smile on his face, and I must have looked hungry for food and hardware. He craftily took me to his favorite Italian lunch spot in Santa Clara, fed me cannelloni, and worked to get a few "extras" in trade for that SIM4-01. In reality, I didn't mind, because I kept thinking about that $3M IBM computer that was my alternative.

Incidentally, five years later, Bob left Intel to found Convergent Systems, a nicely successful computer company in the Santa Clara Valley at the time.

The Age of Intel's 4004 Microprocessor

The SIM4-01 used the 4004 chip but included power supplies and support circuits. It had 1024 bytes (not kilobytes) of RAM for data storage, and up to four 256 byte (not kilobyte) ROMs for programs. We're talking primitive here.

Actually, these were not strictly ROMs. They were called EPROMs, an acronym for electronically erasable and programmable read only memories. They have a quartz window right over the chip. Shine an ultra violet light there for enough time, and the data goes clear. By the way, you have to scam a bunch of 1702 EPROMs because they were a hundred bucks each.

I met Tom Pittman on a journey to Intel. To me, Tom is the first personal computer developer and user. (Later, Tom wrote Tiny-C, for which he is best known.) Clever Tom wrote a "monitor" for the 4004, and a small assembler that actually ran on the SIM4-01 itself.

This, in effect, was the first self-hosted single chip microprocessor development system. Tom's "monitor" took 257 bytes, one more than the 1702 EPROM could hold. He condensed that to 256 bytes without changing the monitor. Ask him how he did it if you run into old Tom, or if you ask me, maybe I'll tell you.

Using a Teletype Model 33

I am going to have fun with this one, because only the diehards have ever experienced the infamous Model 33. It has a big case that holds the printer mechanism and a roll of, normally, yellow paper. On the left side is a paper tape device, half of which is a "reader" that, of course, reads holes punched in that tape. The tape encoding is Ascii, just as we use today.

The second part of the paper tape device is the punch. The punch, of course, made the holes. So, to edit, you make a program and punch a tape. If you make an error, you "backspace" and put "nulls" (all laterally punched holes). Then, start typing again.

Ok, you're saying the next part appears a bit tedious, but if you are a true computer person, you must go through it. After all, we "pioneers" all had to do this stuff two decades ago so that you can enjoy your sweet little lap top while cruising placidly over Colorado at 37,000 feet in UAL First Class. Hey, c'mon, just a little bit of torture for all our troubles in those days.

Here's how programming a 4004 worked in 1972. You buy a UV light, if you can find one, to erase the 256 bytes of memory in a 1702 EPROM for storing your program. Then plug in Tom's monitor and three EPROM assembler.

Stuff some paper tape into your teletype and type in each program line. Use that mechanical paper tape backspace to null out some holes in the tape if you make a mistake. (How'd you like that teletype today in your lap at 37,000 feet? Ok, so keep reading.)

Now you've got a program of sorts on that paper tape. Run the paper tape back through the teletype at ten characters per second, one more time. To understand 10 cps, remember that a very good typist works at 100 words per minute. That's 600 characters per minute, or just about the speed of a teletype.

You get some "syntax errors" from the assembler. So, run that tape back through the reader and stop to "edit" while the paper tape punching is making a new tape, and hope the hell you don't make a mistake again.

Eventually your program is syntax error free, which, by the way, doesn't mean it works right, right? In any case, out pops another paper tape in Intel "hex" format of the machine code.

(I know that you are totally bored with this discussion about paper tape, but I also know that if you have gotten this far, you have enough curiosity to continue. Think of it as paybacks from those of us in the last generation.)

Take out Tom's four EPROMs that hold the monitor and assembler and put in the "hex programmer" that loads up 1702's. Putting these EPROMs in and out would often bend a pin, but by this time, I was so adroit that accidents were normally limited to poking my finger with one of those prongs a time or two every day.

Next, take your EPROMs that have been cooking under that UV light and stick them one at a time into the EPROM programming socket. Run that hex file paper tape that you just got into that stupid teletype again and load each EPROM. This takes 30 minutes to erase the EPROM, and five minutes to load the 256 bytes into that 1702. Meanwhile, reassure yourself that this is a lot cheaper than the IBM System/370 downstairs.

For reference, an average JFK to SFO flight takes about six hours. That's the time it takes to program twelve EPROMs of 256 bytes each, or a total of 4,096 bytes of memory. That laptop you ride around with these days loads program memory of that size in about 1/5 of a second. Hmmmm... 6 hours compared to 1/5 of a second. Not bad. Ok, if you scammed enough of those 1702's you *could* erase four or five of them under that UV at the same time.

Are you bored with this discussion? Good. Hey, you can get through this in five minutes. It took us a year, Ok? Paybacks.

Let's go back to the 4004. So, now that you've written your 4004 assembly language program, you're ready to test. Stick in your newly programmed EPROMs, push the RESET button on the SIM4-01 and GO. Usually nothing happens. That's because there is a bug in your program. But, there's no way to tell where that bug is. The SIM4-01 just sits there and doesn't do anything.

There were a couple of LEDs on the SIM4-01. To find your error, you reprogram and use those LEDs to see how far your program gets before "crashing." That usually took a day or two, or the equivalent of a few flights from coast-to-coast.

I wrote a simple BCD (binary coded decimal) calculator for the National Computer Conference in 1972 using this little SIM4- 01. It took a month, and by the time it was done, I had sat and stared at the teletype so long that I had to put it on a little pedestal, so I could stand and debug, for obvious medical reasons which I will not go into at this time.

So, next time you have an itsy-bitsy bug in your program running under Borland C++ with the Turbo Debugger, don't come to me for sympathy, because I'll just give you a SIM4-01 and stick you on an airplane with an old sixty pound teletype that I happen to have in my garage.

I built a briefcase computer using the 4004 in 1972. I'd take this around for demos, and people really liked it. It may have been the first personal computer. Once, I took this to a lecture at the University of Washington. A guy named Tom Rolander was in the audience, and he became enthralled with this little computer. Tom was to play an important role at DRI.

Grace Hopper's 4004

Here's a story about a lovely lady, Captain Grace Hopper. Actually, she became an Admiral after I met her.

Grace Hopper was self-proclaimed to be the first programmer, and I believe her. She was quite elderly when I met her, but wore the crisp dress uniform of the Navy each time I saw her.

She helped invent COBOL, that business language used throughout the world today. She was a brightly clever, petite gray-haired lady with a wit and brain sharper at her age than most who were half her's. She was like having a charming grandmother who could talk your socks off about computers. Quite a combo, indeed. All her stories were true, captivating, and always drew a crowd.

Captain Grace gave lectures at universities throughout the country. At the NPS, she handed-out strips of thin wire about a foot long that became known as "The Grace Hopper Nanosecond." That piece of wire she gave to those students showed how long it took for electricity, or the speed of light to travel in a millionth of a second. It was a good illustration for everyone to see the limits of linear computing.

Grace became a friend. I was helping to "man" the Intel booth at the National Computer Conference in the Fall of 1972, and Grace came by proudly in her military dress. She gazed sneakily at me with those eyes that shown in that brilliance of hers, though her outward appearance was of frail demeanor. With stealth, Grace reached into her purse and brought out an Intel 4004 chip, then whispered to me "this is the future."

Alas, Captain Hopper was forcibly retired some months later. After all, she was well past mandatory retirement. The Navy was her life, and I'm quite sure she was devastated. There is a good ending though, because popular naval opinion brought her back as an Admiral.

For analogy, Grace had become like the Blue Angels Flying Team that does fly-bys and airshows for the Navy, making a bunch of noise and burning lots of kerosene.

The Navy supports the Blue Angels to recruit Navy flyers. The Navy supported Grace to recruit the interest of college students in bringing their talents to the technology of computing for the Navy. She did a fine job of that. By the way, she was waaaay less expensive than the "Angels" to maintain.

Bugs in Your Programs

It's fun to tell you the story about how Grace and her buddies invented the word "bug" as it applies to an error in a computer program. She worked with John Von Neumann in the days of World War II. John worked on solving numerical analysis problems, like the kind that let you toss mortars around.

They had a repeating error in one of their programs while operating their "relay" computer. Remember, relay computers were built with mechanical clappers driven by magnetic solenoids. That was in the 1940's, before the vacuum tube machines, and certainly before the transistor and chip computers.

Upon investigation, they found that some of nature's real live bugs were in there, preventing the relays from closing properly. So, next time you have a "bug" in your program, try "Raid." Oh, and by the way, that is why CP/M's debugger was called the "Dynamic Debugging Tool," or DDT for short.

The Age of the Intel 8008 Microprocessor

I began to work for Intel as a one-day-per-week consultant in 1973, following my somewhat discouraging experiences with the i4004. I was now the proud owner of a used Model 33 Teletype that I got by trading my 4004 teletype diagnostic program to a repair company in Cupertino.

Intel had developed a new microprocessor chip. It was called the i8008. The 8008 was more advanced, but that wasn't hard to achieve, considering the design of the 4004. It was an 8-bit computer with a "Von Neumann" architecture. That means that the program and data are all in the same memory, like current PCs today.

The software group at Intel in 1973 consisted of Bill Byerly, Ken Burgett, and myself as a consultant. I was under contract to develop a simulator for the Intel 8008, much like the one I had done for the 4004, under direction of microprocessor manager Hank Smith.

The microprocessor group at Intel occupied the space of a not too large kitchen at the time. Bob Noyce, founder of Intel walked through occasionally with his usual smile, and gave a few pats, but they were token gestures. His interest was in making 1101 RAM chips in the other section of Intel's only factory on Bowers Avenue in Santa Clara.

Noyce and Computer Games

Bob Noyce was a clever, somewhat silent, gray-haired guy who mostly wore a white lab coat to cover his casual demeanor. At first, I must admit that I liked him because he was a pilot. But, there is one fun experience that I had with Bob, and this is not intended in any way to put him down, as I admired him greatly for all his contributions.

One engineer at Intel in 1972 had developed a graphics chip set that controlled a little triangular space ship. It mimicked the game of "Space Wars" that had traditionally operated on DEC minicomputers.

Space Wars worked like this. You control the velocity and direction of this graphics "space ship" using keyboard control. The ship operated under those Keplar's Laws of Motion around a "sun" that would pull it in if your spaceship came too close. I thought that it was a good idea for a commercial consumer game.

Later, I used my 4004 briefcase computer to build a simple game of "Nim" that used the 4004's LEDs. Stan Mazor, one of Bob's gurus, suggested we approach Noyce with the idea of making game computers. This was in 1973, before such things were popular. Noyce peered at the LEDs blinking away on my 4004. He looked at Stan and me and said, bluntly, that the future is in digital watches, not in computer games.

I suspect Bob's interest in watches came from Intel's then recent acquisition of "Microma," the first digital watch company. By the way, just like I scammed the SIM4-01, I worked a deal with Bob for a Microma digital watch that I was very proud of, in trade, of course, for more software.

But, unfortunately for Intel lots of other companies, mostly Japanese, also figured out how to make digital watches and flooded the industry. Intel dropped the Microma business. I don't know if gaming would have been successful through Intel, but I have thought about Bob Noyce's decision that day.

He had the beginnings of a Nolan Bushnell-style industry that was sacrificed for a digital watch business that didn't take hold. I'm certainly not saying that Nim was the right game, but the Space Wars chip set was on the right track. It's fun to speculate.

Bob was a friend. He, like all of us, made some decisions that are right, and some that could have made the future unfold in a different manner. But, most of all, Bob treated his people with dignity. He had the foresight to build an industry in microprocessors. I was very sad when he died recently. I sent a note to his wife, Ann. I liked them both very much.

The NPS Microcomputer Lab

Microcomputers became the new "thing to study" at the Naval Postgraduate School in 1973, and the faculty backed me to get the funds to build a laboratory for them. We set up some SIM4-01 stations, then went to the Intel Intellec 8 development systems.

It turned out to be quite a nice lab that jump-started a bunch of guys like Glenn Ewing, who helped engineer the IMSAI computer, and Gordon Eubanks, who founded Semantec, along with a bunch of other projects with the Navy at the Naval Undersea Center at San Diego, and the test center at Keyport, Washington.

We taught the NPS students how to use micros. One project was a CMOS 8080 wristwatch computer that computed "partial pressures of nitrogen" according to standard Navy diving tables.

This means that a Navy diver could descend to a certain depth while the watch continuously monitored time at depth. Then, upon ascent, the diver looked at the LEDs on the watch to tell how long to remain at a certain level to avoid the "bends." Nice package, indeed.

It was a low power, obviously waterproof device, and is still used today. We used CP/M to make that software and others like it for the Navy. That little Scuba computer was designed in 1975 and went into production for the Navy only a few years later.

A High Level Language for Microprocessors

Let's go back to 1973. I must admit that I spent more than the Postgraduate School allowance of a "day a week" on my consulting, but I will not admit how many. I lived in my Volkswagen van at night in the Intel Bowers Avenue parking lot after my eyeballs gave way.

One day, I knocked at the door of Hank Smith's office at Intel. He invited me in. Hank was the manager of our minuscule microprocessor software group. I told him that I could make a compiler for the 8008 so that his customers didn't need to use low-level assembly language. Hank didn't know what that meant, but I showed him how a customer could write:

$$X = Y + Z$$

and that would make several lines of assembly language. He immediately got on the phone and called a customer he was courting. The customer liked it, and Hank, with a smile, said to "go for it." I like corporate decisions like that.

The language I developed was named "PL/M," and it used compiler technology of the time, based upon McKeeman's XPL. PL/M stands for a "Programming Language for Microcomputers," and is still used by Intel customers today, though largely supplanted by the "C" programming language. PL/M allowed a microprocessor programmer to remove himself from the tedium of assembly language programming.

As spokesperson for high-level languages on those little Intel machines, I had to fight vigorously to show the advantages of the approach in public forum. It wasn't too difficult, because I often demonstrated that I could produce an effective program at ten times the hourly rate of an assembly language programmer using PL/M, using about the same amount of program memory space. PL/M became a success.

Meanwhile, Intel had developed the small computer system in 1974 called the Intellec 8 that I mentioned. It must have been the first commercial personal computer, although no one thought of it as that.

Toward Self Hosted PL/M

By "self hosted," I mean that program development and testing take place on the computer that sits directly in front of you, and the program you write can run on that same computer.

Unlike Tom Pittman's self hosted little development system for the 4004, PL/M used a minicomputer like C-cubed's DEC PDP-10. You write the program using the closest DEC PDP-10, run to the paper tape punch and get your "hex" file. Then, you program your EPROMs or load RAM using the paper tape. Messy, and not the right answer.

The Intellec 8 came with an EPROM "programmer" socket on the front, and standard with 4,096 bytes (not kilobytes) of RAM. You could make it 16K by adding three more memory boards at $1000 each. You used a good old teletype to make programs for the 8008 in much the same way as with Tom Pittman's 4004 monitor and assembler. And, Ken Burgett had developed a nice 8008 debugger called MON8.

I horse traded more software and got my own Intellec 8. Then, I got a loan for $1700 to buy a character video display and a huge printer. I was close to techno-heaven. I now had a real personal computer that could actually be used to make real programs.

One of those programs for the Intellec 8 could be a compiler for PL/M that would operate directly on the 8008, rather than be tied to the expensive DEC PDP-10. This would make PL/M "self- hosted" on the Intellec 8.

But, there was a fundamental design problem. The 8008 had a "stack" size of eight return addresses. This means that only seven subroutines can be in effect before an undetectable overflow. And, this made it unusable for PL/M and lots of other micro-computer applications. But Intel, with Mr. Shima's help, came up with a solution.

The Age of the Intel 8080

Shima, as he was known to me, was a reserved Japanese chip architect. He had designed the Intel i4004, which for some reason, I did not hold against him any longer.

In 1973, Intel brought Shima back to their Bowers Avenue facility to upgrade the 8008, especially to take care of the short stack problem. Shima worked at his lab on his design. One day in late 1973, Shima hurried to the office where Bill, Ken, and I were programming. He said that the new i8080 was "running."

Of course, I followed in trail to his lab and found a poor assaulted microchip tethered by thin probes attached to some electronic monitors that must have told Shima something. There was a binocular microscope peering down into the middle of the mess.

Shima told me to take a look through the microscope. I saw nothing but a bunch of needle tips touching some apparently interesting spots on that chip. He told me that half his 8080 chip was running. Not wanting to break his stride, and not knowing what the heck was happening down there, I said, "Wow, Shima, that's really neat!"

Many people attribute Apple's entry with the IIe as the basis for the personal computer. But, I am of the opinion that Shima's little 8080 chip really started it all right then and there.

Ok, now Shima's got a pretty good microcomputer, called the 8080. Intel upgraded their Intellec 8 to a version they called the Intellec 8/Mod 80. It was truly a personal computer, although designed for development, with two exceptions. There was no large memory storage device such as the hard drives we have today. Second, the whole thing was waaay too expensive.

By the way, "backup" storage has two meanings in computing, so I should probably sort them out. Backup storage, in the beginning meant the big hard disk that stored program, data, and left-over student program output that you didn't ever receive, and, incidentally, may be yet awaiting. But, today a "backup" is a copy of the hard disk to something like a little tape drive.

To avoid confusion, I'll use the alternative term of "mass storage" to mean floppy disks and hard disks.

The Intel Intellec 8/80 showed an avenue to self hosted high level language development. I wanted severely to get away from the DEC PDP-10 cross-development and do PL/M compilation and debugging on the Intellec 8/80. I didn't care about the expense, because I had already scammed my own Intellec, so I was only concerned about the lack of mass storage. And, hey, I had my own 9600 baud CRT (although I was still making payments in that area).

I had to solve the mass storage problem before PL/M could be operated independent from the DEC PDP-10 miniframe.

CHAPTER 5

The Advent of CP/M

The Quest for a Fast and Cheap Storage Device

So, the term "mass storage" means things like diskettes and hard drives today, but we just didn't have anything like that in 1973 that was affordable. The best was a high-speed paper tape reader and punch combination.

Some had proposed using cassette tapes rather than paper tape. I tried once to build a controller for a cassette tape in 1974, but failed miserably, thank God. For, if I had been successful, there would have been no CP/M, and certainly no MS-DOS.

I needed mass storage to support PL/M development. Well, Memorex was just down the street from Intel. They had come up with the new "floppy disk" to replace those IBM punched cards we talked about. The floppy disk worked with IBM-style keypunches but made a disk instead of punching cards. (Incidentally, I don't know to this day why a floppy disk is a "disk," but a CD-ROM is a "disc.")

Getting back to the point, think about this. On the one hand, you have a teletype paper tape reader that processes at ten characters per second. Now, the floppy disk holds 250,000 characters and moves data at 10,000 characters per second. That's about the same as 2000 feet of paper tape, operating 1,000 times faster than that old teletype.

Quite a reasonable difference. Not only that, the diskette would, unlike a paper tape, instantly access any portion of the data without rewind or fast forward. That made Memorex's new floppy disk quite inviting for mass storage.

Those payments on my new CRT didn't give any margin for a floppy disk drive at $1500, plus controller at $3000. But, Alan Shugart had started a small operation up the street, called Shugart Associates. Alan, also, was making "IBM Compatible" floppy disk drives. I went to Alan to scam one.

I misused my Intel credentials to get access. Finis Connor worked there as a marketing manager, and I got Finis to give me one of their old test drives. That poor drive was being used to evaluate MTBF (mean time between failure) and had a few hours on it. Like about 10,000. Finis gave me some new felt read/write head pads to replace the worn ones and sent me on my way.

Great. I've got this old, worn, 8 inch floppy drive. And, I don't have a "controller." Even if I did, it wouldn't work in my Intellec 8/80.

See, you need some fairly complex electronics in that controller to make the diskette drive find certain locations and transfer data back and forth to the Intellec 8. I can tell you without reservation that I sat and stared at that damn diskette drive for hours on end and played by turning the wheels by hand, trying to figure a way to make it fly.

I even tried to build a diskette controller for that freebee drive from Finis. Bob Garrow from Intel roughed out a design. I bought the parts. But I, being mainly hardware inept when it comes to debugging the stuff, couldn't get my controller to work.

The absence of a controller for that floppy drive was the only thing between me and a self-hosted compiler for PL/M on the Intellec 8/80. And, it drove me nuts that I couldn't get that little drive working.

The disk drive sat in my office at the NPS for a year. I'd just look at it every once in awhile. That didn't seem to make it work any better.

Origins of CP/M

I became tired of spinning that little diskette drive by hand, and I couldn't wait for a controller to magically pop up, so I used my software simulator for the 8080 commissioned by Hank.

So, I built an operting system program using the simulator. I called it CP/M, or a Control Program for Microcomputers, mimicking the name PL/M. For me, CP/M's sole purpose was to support the PL/M language. Nothing else.

Using the simulator, I developed all of the CP/M operating system in PL/M using a minicomputer.

Just like the early days of the i4004, I used no hardware. But, by using the simulation, I knew that CP/M worked. I just couldn't figure out how to make that damn disk drive work. I even made a program editor, the infamous ED that proved true under that simulator.

This was 1974. Out of frustration, I called my good friend from the University of Washington, John Torode. John was a Ph.D. graduate of the Electrical Engineering Department there, and I figured if John couldn't build that controller, no one could.

He did build it. He designed a neat little microcontroller and, after a few months of hardware and software testing, that microcontroller started to work. We loaded my CP/M program from paper tape to the diskette, and "booted" CP/M from the diskette, and up came the prompt

```
*
```

This may have been one of the most exciting days of my life, except, of course, when I visited Niagara Falls One Day.

We loaded ED onto the diskette, and I typed

```
*ED
```

The ED program editor came up quite well and alive. Ok, so far, so good. I made up a file with ED and put it to disk. The DIR command showed us the file was there, and the TYPE command showed us that what we typed was what we entered using the ED program.

That was in the late afternoon and, feeling lucky, John and I retired to have a Chinese dinner in Pacific Grove. We walked back to the house, and we told one another that this was going to be a "big thing." (I think this is the third time so far that I heard this expressed, remember the other two?) We drank a jug of not very good red wine and took a well-deserved break.

The Astrology Machine

I only mention the astrology machine incident because it shows how CP/M utility programs were developed for the basic OS. I'm lying. I'm writing about it, because it was a fun project, Ok?

Ben Cooper from San Francisco contacted me through reference by Intel in 1974. Ben wanted to make a "game" machine. Put in a quarter, dial your birthdate, and out comes your horoscope. I'm not into horoscopes, but what the heck, Ben was paying by the hour and I'm not that proud.

Ben is a mousey guy, but an entrepreneur and very easy going. We had good fun working and talking through ideas. He always had his nervous laugh for anyone to accept. He knew quite well about hardware design and enough about software to proceed with building his astrology machine.

Ben liked CP/M, even though it was not a commercial product. And, by that time, I had made a BASIC compiler for CP/M that sort of worked. I despise the BASIC language with a passion, so it is hard for me to admit that I made one. But, I can claim to the good that I never sold it.

Also, by 1974, I had built a real two-diskette drive computer using John Torode's controller that we had replicated several times over. It was not a bad looking prototype, and we had it hooked to my Intellec 8/80 with my CRT that I had nearly paid off. And, good old John Torode, busy with his hardware, was starting a nice business of his own in Oakland, called Digital Microsystems.

I kept the handbuilt "system" in a small converted tool shed at the back of my home in Pacific Grove. Ben would come there from San Francisco and work for days on his Astrology Machine. I wrote programs for him, such as drivers for his little printer, and he used my BASIC to program the astrology algorithms.

Ben worked continuously without break. One sleepy evening he used the CP/M command "del *.*" to get a listing of all his files, which would normally be done with "dir *.*"

Unnecessary to say, but that "del" command didn't do exactly what he wanted. It deleted all his files, including programs that he'd worked on for days with no backups. This is why, today, you have the choice to "era" files, rather than "del" them. And, this is why you get a prompt "Are You Sure (Y/N)?" You can thank Ben for that. Ben, a bit discouraged, crashed at my house for the remainder of the evening and drove back home the next day for a short holiday from the topic of astrology.

He made a few astrology machine prototypes, and put one at San Francisco's Fisherman's Wharf. Then, he'd see how people liked it. I guess this would be called a low-budget test marketing scheme.

We watched one night. A young couple, hand-in-hand, walked up and stuck in a quarter. They didn't really bother with setting the birthdate knobs, they just took what happened to be set-in there. The young couple, of course, got someone else's horoscope. They joined loving hands again and happily walked off, evaluating someone's future written on hardcopy as their own. Because of it, they are probably married with seven children to this day.

Ben Cooper got Remanco, a gaming company in Saratoga, to build 200 astrology machines. I'm not sure how he accomplished this, but Ben was resourceful. I believe they have about 199 left, so if you're interested in one, I'm sure Remanco would appreciate a call.

Ben's astrology machine allowed time and interest in developing CP/M utilies. I taught at NPS by day, and worked on "tools" for CP/M by night. The tools, or "utilities," were programs such as assemblers and debuggers, and programs that managed and manipulated files of programs and data on diskette. These were all necessary to effectively develop applications for the 8080 processor. They were needed to support Ben's game, but useful for any other 8080 developer as well.

The astrology machine was, shall we say, not wildly successful, but later Ben formed a company called Micromation in San Francisco that was. It wasn't until he, and a bunch of other grass-rooters got hit by IBM in the early eighties that his successful business went upside-down. After IBM took his market away with the PC, Ben sold what was left of his company, bought a sailboat, and took an exit stage-left from life. This was typical of hundreds of small companies that felt the brunt of IBM's entry into PCs.

Although he may not know it, his tenacity with that old astrology machine helped me sharpen the CP/M tools. If CP/M wasn't working right, he'd tell me straight away to fix it.

Jim Warren and DDJ

I worked as a consultant to Signetics through the summer of 1975. Signetics, in competition with Intel, wanted a high level language like PL/M. I designed and implemented one and called it PLuS for their 2650. This took all of the summer, and both PLuS and the Signetics 2650 microprocessor bombed.

During my consulting, I met Jim Warren who also consulted there. Jim was instrumental in the early PC cult magazine called *Dr. Dobb's Journal of Computer Calesthentics and Orthodontia.* I don't know what that means, but the *Journal* appealed to the grass roots of personal computing.

Jim knew of CP/M and, one day at Signetics, suggested that I sell it to the masses. I placed an advertisment in DDJ for CP/M at $25.

Jim was to have me on his TV program that he called *Computer Chronicles*, which I mentioned before, based at the Public Television Station at the College of San Mateo. Jim left the *Chronicles* that first year to Stewart Cheifet to host. Stewart then engaged me as co-host, and it turned into a full-time spot for that seven year period. Although an unpaid position, the show gave me weekly connections and exposure to technologies like the "HAL 9000." I have never forgiven Warren for that assignment. Ok..., I'll again admit, the show *was* kind of fun.

CHAPTER 6

The Early Hardware and Software Suppliers

Omron, Altair and IMSAI Corporation

In 1975, John Torode and I got a connection to Omron, a CRT terminal manufacturer that specialized in word processing for newspaper editing. John convinced Omron to license CP/M and his controller design for their terminal. We got $25,000 for this, and we split it. This was the biggest pot either of us had won.

Omron built the first real CP/M-based computer hardware product, and, although a bit bulky, it was quite a nice personal machine for the day. We both snagged a couple for our own development. John adapted his controller to Omron's hardware, and I rewrote a diskette driver interface for them.

Meanwhile, during my consulting at Intel that year, we all heard rumors of a new small computer from a company called Altair, headed up by Ed Roberts. They offered a computer for $500. We all stood around at the Intel lab thinking this one over, because we knew Intel was selling their 8080 chip for $360. Hmmm... Someone did a deal with someone else to get this to happen. After all, the Intellec 8/80 was selling for ten times that price.

The Altair was a hobby computer that came as a kit from Robert's company in New Mexico. It had a decent enclosure, lights on the front panel, an 8080 processor, but only 256 bytes of memory.

Dave August, a member of our local computer hobby club (called Computer Phreaks United, or CPU), bought one and built it. You programmed that Altair with toggle switches on the front panel. Ok, so that's a bit primitive, but better than the 4004.

Computer clubs were forming elsewhere, with the Palo Alto group being the most prominent. These clubs had members who were often daytime computer engineers from the Silicon Valley area working for firms like AMD and National. These guys were bright enough to build their own computers with no outside help, and they did. Each one was different and unique in its hardware interface.

Then, a new company formed up in San Rafael on the East Bay of Silicon Valley. It was called IMSAI (even though not at all Japanese). IMSAI took Ed's Altair 8080 direction to produce yet another low cost computer. This time headed for the commercial market rather than the hobbyists.

IMSAI products were marketed by Ed Faber, who I had dealt with at Omron, and I'm sure this helped to gain CP/M's entry into IMSAI's organization.

Not only that, but my student, Glenn Ewing was about to leave the Navy and had consulting deals going with IMSAI. IMSAI had promised delivery of a diskette operating system for their customers but hadn't written a line of code for it. Glenn, being my student at NPS, knew about CP/M and proposed that IMSAI use CP/M rather than make their own disk operating system.

Glenn came to my tool shed computer room in 1975, so we could "adapt" CP/M to the IMSAI hardware. What this means is that I would rewrite the parts of CP/M that manage things like diskette controllers and CRTs.

Well, come on, I'd already done this so many times that the tips of my fingers were wearing thin, so I designed a general interface, which I called the BIOS (BASIC I/O System) that a good programmer could change on the spot for their hardware. This little BIOS arrangement was the secret to the success of CP/M.

With the BIOS in place, a programmer could make CP/M work with their specialized hardware. With all those hobbyists out there, believe me, there was no shortage of specialized hardware. Glenn and I built a BIOS that afternoon and stuck CP/M on an IMSAI. He demo'd it to Ed Faber and the IMSAI engineers, and they loved it.

I licensed CP/M to IMSAI for $25,000. I was "rich," and IMSAI had their OS that they'd been promising.

Then, I heard that two guys had made a small BASIC interpreter for the Altair. The two versions were called "4K and 8K BASIC," indicating the size of memory that they used. These two guys were Paul Allen and Bill Gates, who I had run into at C-cubed years before. I was to run into them often again.

And, oh, by the way, Ed Faber from the early days of Omron and IMSAI went on to help form and market today's Computerland.

The Early Days of Microsoft

Understand that the basis for Gates' and Allen's success was in the computer language called BASIC, developed at Dartmouth University in the 1960's. "Woz" wrote a BASIC for his Apple II, and, later, Gates and Allen in competition sold their "Softcard" add-in board for the Apple, with their own BASIC running on my CP/M. Here's how it all started.

Paul Allen and Bill Gates took their BASIC interpreter to the Altair market out of a company they named Microsoft, based in Albuquerque, New Mexico. The location being set, to some degree, by Ed Roberts' factory nearby. Allen and Gates licensed their BASIC language interpreter to Roberts for his Altair.

Bill came to me in Pacific Grove one day in 1977 to discuss his current quandary about relocation of his Microsoft company.

We invited him to stay that night at our home. Dorothy fixed a nice roast chicken dinner, and we had light conversation. Bill told us about the tickets he had received driving at high speed in his Porsche 911 through the main street of Albuquerque. We found a common topic since I, to my misfortune, also enjoy speed of the automotive type that sometimes, but not often, results in an encounter with The Law.

At this time, I was quite friendly with Bill, and we always talked of the "synergism" of our two businesses. But, quite frankly, I was always apprehensive of his business moves, as I found his manner too abrasive and deterministic, although he mostly carried a smile through a discussion of any sort.

Gates is more an opportunist than a technical type and severely opinionated even when the opinion he holds is absurd.

For example, we faced one another in debate over graphics standards at a National Computer Conference in 1979. Bill took the adamant position that NAPLPS would be the wave of the future. Of course, it wasn't. It's only one example, but don't think for a minute that Gates made it "big time" because of his technical savvy.

Gates and I spent time that day in Monterey driving the Central California coastline while we talked of business possibilities. He wanted to relocate but had not decided where.

We talked of merging our companies in the Pacific Grove area, and we both took that into consideration. Our conversations were friendly, but, for some reason, I have always felt uneasy around Bill. I always kept one hand on my wallet, and the other on my program listings. It was no different that day.

The overriding factor for Bill's decision to relocate was that his family resided in Seattle, with his father being a prominent attorney, and his mother being on the Board of Regents at the University of Washington. His parents lived in a fine brick home at Wyndermere, a posh district just off Lake Washington.

Incidentally, a few months later, Dorothy and I were invited to dinner at Bill's par-ent's home in Seattle, and I can honestly say that their hospitality was without mea-sure, and we enjoyed their company immensely. They were cordial, and it was apparent that they had been "well-to-do" for many years before.

Bill and Paul made the decision to take their little Microsoft company to Redmond, just a short drive across the "Floating Bridge" from the university campus and Wyndermere, rather than join ranks with DRI at Pacific Grove.

Part of the decision to not join with DRI came from the recognition and acceptance that DRI had in the industry. That recognition limited Bill's growth from within a merger.

The combination of Kildall and Gates could have been a killer-deal in those days. I had the operating systems for the decade to come, and he had the opportunistic approach to garner business. But, our attitudes differed entirely, and that could also have been a disaster. I think we both realized this and simply let the "deal" die.

The West Coast Computer Faire, SSG, and the Electric Pencil

I've talked about Jim Warren. He was a writer and an ex-hippie turned techno, but still with the quirks of his "hippie days." Jim was involved with that *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia*, where, I mentioned, CP/M was first advertised.

DDJ promoted hobby computing. And, these were the guys who made the foundation for what we, today, think of as the personal computer industry. Jim also formed up the West Coast Computer Faire in 1976, consisting of a collection of small companies that were promoting small computer products.

I was very impressed by Michael Shrayer's "Electric Pencil" program that he presented there. It was the first small computer word processor. Although not elegant because it only used a TV screen with a forty character line in computer fonts, it did show that one need not use "White-Out."

But, the best was Structured Systems Group's display area. Alan Cooper and Keith Parsons made accounting systems using the IMSAI computer and CP/M. I recall telling Dorothy that SSG's was the first really complete application effort for inexpensive computers, and those applications like SSG's would be "a very big thing in the future." (Have we heard this line before?)

Our new company, Digital Research, was represented there. We even shared a booth with SSG. We sold CP/M and the rather technical, but not too inaccurate, documentation that let the hobbyists and budding micro-based commercial computer users access their floppy disks. CP/M was a success.

The next year the West Coast Computer Faire blossomed. Jim Warren took Le Charge de' Affair by roller skating from aisle to aisle, as the show's floor monitor.

In this second year, there were more applications offered to the public. Seymour Rubenstein from defunct IMSAI had coupled up with Rob Barnaby, their best programmer, to sell the word processor called WordMaster. WordMaster was short-lived and reissued as "WordStar." They called their company "MicroPro."

Seymour and Rob had both applied for positions at Digital Research, but I was adamant about avoiding applications that compete with computer makers and Independent Software Vendors, a phrase coined at DRI to describe small companies that make PC programs for specialized markets. These were our customers.

I felt a need to support my manufacturers with an operating system standard and provide a structure they could depend upon. Although this seems like a business position that doesn't max the profit, realize that many of the manufacturers with whom I dealt were personal friends of mine. Quite frankly, I could not, for example, sell an accounting package against Structured Systems Group, and then face Keith and Alan at the next computer conference.

Gates' philosophy was quite different. He had no problem competing with his customers. And, this is where we differed. Monetarily, Bill had the proper approach to get a stack of bucks, and lose a bunch of friends in the meantime. I guess that's ok.

The Softcard

By the year of the 1978 Computer Faire, Gates and Allen had moved their affairs to a small area located within a bank building in Redmond.

Allen had directed the design of the "Microsoft Softcard" (which I must acknowledge as a trademark, and while I'm at it, White-Out is probably a trademark too, oh, and Borland C++ is one, I'm sure, and, say, what have I left out?, Oh, yeah, CP/M is a trademark, and ...)

The Softcard was an add-in board to the Apple IIe. It was an Intel 8080 processor, with interface circuits, to disable that horrible MOS Technology 6502 processor that Steve Wozniak designed into the Apple. The Softcard was to run CP/M and Microsoft's BASIC.

I was contacted by Bill to determine licensing costs. We sat together at this Faire on the vacated second audience floor, overlooking the activity of the packed floor below and talked through the deal.

He got a good one. I wanted a royalty, but Bill wanted a buy-out, and was stuck on that point. I sold him 10,000 copies at $2.50 each; the deal was consummated, with my attorney Gerry Davis there taking notes, as lawyers always do.

The Microsoft Softcard was a success, because it gave Apple II users access to a large software base from the CP/M application suppliers, but mostly, for Bill, it gave access to his BASIC language. Bill signed agreements to protect the CP/M design under this licence, as shown in the Appendixes. And, the Softcard was the key to Microsoft's success.

This is the issue. Bill's Microsoft had a BASIC interpreter that would operate with the Apple computer using his Softcard and my CP/M. To Bill's dismay, the Apple platform for application software was controlled by Apple Computers, not Microsoft. That's a wild card for Bill. He "needed" to control the platform for his BASIC, and IBM was to provide him with that.

Battle of the BASICs

Gordon Eubanks is the Chairman of the Board at Semantec today. They have been successful at supplying PC software, such as Q&A and Norton Utilities. I need to write about Gordon, because he was a prime mover and often, but not always, my friend. Most important, Gordon fought a personal battle against Gates and Allens' BASIC language.

I met Gordon when he was a student at NPS. He entered the Computer Science program there to get a master's degree. Gordon was a bright student, but very caustic, vociferous, and downright nasty when he felt his professors were incompetent.

Gordon latched onto me when I was teaching at the NPS, and he was a student in 1974. He asked to do a thesis under my direction, using microprocessors, and I accepted. I proposed two topics. One could be a "word processor," and the other might be a compiler for the BASIC language, which I somewhat discouraged.

But, Gordon took the latter. I gave him my version of BASIC to start with. It was written in PL/M, so he could use the NPS computers to do development. Using this program, he made BASIC-E into a fully working language system for the 8080.

This BASIC-E compiler was Gordon's master's thesis and when he graduated, he went back into the Submarine Service as Engineering Officer onboard a nuclear submarine stationed in the city of Vallejo near IMSAI and IMSAI's new employee, Glenn Ewing, who Gordon knew quite well.

IMSAI wanted a way to program commercial products, such as accounting packages for their new IMSAI 8080 computer. Gordon's thesis project was the right direction, but unfortunately for Gordon, BASIC-E was in the Public Domain. Not only that, I wrote much of it before he had worked on it at all. So, he couldn't sell it to IMSAI.

There was one other problem with Gates' Basic. BASIC imitates a tiny version of FORTRAN. But, FORTRAN was built for tossing mortars, not computing the National Debt.

For example, normally in BASIC or FORTRAN, you might want to print the value of 1/3. It comes out .333333333333333 (ad infinitum) depending upon how the computer works. Not only that, internal computations start rounding off, sometimes in the wrong way, and you start to gain or lose pennies here and there. Accountants don't like that.

So, Gordon, while in the Navy, contracted with IMSAI down the street from Vallejo in San Raphael to build a new version of BASIC-E, where all those computations were in decimal (BCD) to make money counting easier to program and to ensure rounding-off didn't happen. Gordon called his compiler "Commercial BASIC," or C-BASIC, and it provided a software tool to make commercial, rather than just hobbyist application programs.

C-BASIC was Gordon's revamp of BASIC-E that he did for his master's thesis, but enough of a revamp that he could sell it. This C-BASIC became the foundation for his company called "Compiler Systems" formed in the Los Angeles basin, run by his frail, but quite clever mother, Mary Eubanks, and controlled by Gordon on his submarine whenever it surfaced.

Gordon became integrally involved in Digital Research in the years that followed. SSG with Alan and Keith in control formed a close alliance with Gordon. SSG developed their accounting software using Gordon's C-BASIC language and, thereby, helped debug the C-BASIC language itself.

Of course, Gates was not happy with Gordon's C-BASIC. This was a threat to him and his plans. I can tell you quite squarely that old Gordon, being a good naval officer hit a direct broadside on Gates with that C-BASIC, because it was very popular with the new ranks of program developers. As a result, Gordon and Bill did not share a love of any sort for one another.

Gordon's C-BASIC was different from Bill's BASIC. C-BASIC was made for commercial products, like SSG's accounting systems, and a host of other products that are alive and well today. Bill's BASIC was for the more casual user. Here's where the two engaged. Bill wanted the entire pie but couldn't get it because nobody wanted his BASIC for commercial applications, like accounting or inventory. This locked up Bill and Gordon in the BASIC language "tools" marketplace by 1980.

When Gordon left the Navy, he went back to full-time computer work, with his mom holding the fort at his company, Compiler Systems. Shortly thereafter, Gordon joined forces against Gates and company through acquisition by DRI.

This all happened during and after the events at the first "commercial" small computer success, IMSAI, so I'll go back to those days for context.

The Culture at IMSAI

I must admit that IMSAI people were a mixed group in 1978. Seymour Rubenstein, before his departure to form MicroPro, sat in his marketing and sales office with his EST cube, like most of the others there.

Rob Barnaby, on the other hand, was a complete technonerd and lived only to write programs. Rob was a rather tall, lanky person with perpetually unkempt brown hair. He spoke in short sentences that were always direct and to the point. If a question did not relate to computers, Rob did not generally answer.

Rob typed continuously at his IMSAI computer with fingers too large to fit a modern keyboard. But, Rob was always a friendly sort, and I personally give him credit for starting the word processing industry.

While at IMSAI, Rob wrote a program called NED. It meant "New ED." Recall that the ED single line editor started working the same day CP/M did. Hey, in case you have doubts, forget them. I'm not taking credit for the design of NED, Wordstar, or any of those products. Especially not the progenitor, ED, even though under close scrutiny any Court of Law might balk.

Although old ED may have been the progenitor, he may well have only luckily out lasted his lifetime as program "editor."

The only reason ED survived is that it came "free" with CP/M. In ED's defense, however, I can honestly state that there is only one worse line editor that came to the fore later, in the 1980's. It was called Bill's EDLIN supplied by a company from up there in the North Woods.

ED Goes on the Big Screen

Rob Barnaby from IMSAI took my ED design and made it "visual" as well. By visual, I mean that he made ED into a full screen editor that allowed scrolling, paging, and cursor actions, that simplified ED's operation.

Rob wrote that visual NED program while working for IMSAI, but IMSAI went into bankruptcy in 1979, leaving a bunch of software around that the principals apparently didn't value quite highly.

One was NED. Seymour seized the opportunity and grabbed it before the bankers found it. Then he grabbed up Rob, and Rob finished NED into the WordMaster that I told you about.

Although Rob was clever, he wrote "spaghetti code." This is assembly language programming that may be efficient but could not be understood by another programmer, because it winds in and out of itself. I know. I tried to fix a bug in NED once. I stared at that program for three days and gave up.

Those of you who remember ED may take solace in the events of ED's demise. DRI had an official gathering, organized by Alan Beebe and the Engineering Department in 1984. The attire was black. ED's death was celebrated by all present. I don't think that was nice, do you? I mean...

Beginnings of Micropro

After IMSAI's downfall, Seymour and Rob formed MicroPro up in Marin County's San Ramon, near the George Lucas Ranch, Sear's Point Raceway, and the eternally boring Renaissance Faire.

Seymour and Rob had shown Wordmaster at that West Coast Computer Faire, but then Rob made WordMaster into WordStar, and WordStar was the first successful word processor. Then, Rob was dismissed by the board.

That may have been a fundamental error for MicroPro because, knowing Rob's programming style, I seriously doubt that anyone there could figure out what he had done. They had to redo the entire WordStar program in the C language, and it took a year or two to get back on-line.

Ok, so Rob wasn't the world's most generalized coder. He was trying to perform a task with the minimum amount of memory. Those were the constraints of the day. Tight, fast code sells products.

I saw Rob Barnaby five years ago. He was living in a small flat in San Francisco. He gained little personal wealth. But, without Rob's work, MicroPro would simply not exist, and Seymour would be out selling plastic EST cubes.

IMSAI died a quick death in 1979. The bankers sent out notices, and we all came to bid on IMSAI's remains. Computers, terminals, test equipment, chairs and tables.

I found a cabinet of particular interest. It contained all of the CP/M source code that seemed to have a gathering of interested people roaming about it. Our attorney, Gerry Davis, got a court injunction to prevent its sale. That's one thing that Gerry did right

CHAPTER 7

Digital Research

The Early Days of Digital Research

Digital Research started operation in 1976. It was named "Intergalactic Digital Research," because a consultant up in Sausalito held claim to the name "Digital Research." The "Intergalactic" was stuck there by Demi Moore, a creative assistant to our corporate counsel, Gerry Davis. We got the real name back when the consultant went out of business two years after.

The first CP/M Version 1.3 was a cheap piece of software, upped at $70. That was dirt cheap when compared to mainframe software priced in the thousands. Most computer software buffs were astonished. But, CP/M set standards in small computer pricing. And, customers liked it. Well, most liked it. On occasion, we'd get a call in customer service like this. "Any piece of software selling for $70 can't be any good...bzzzzzzz..."

God, it would be great if customers still had that attitude. Nowadays, $70 is a high-price. For 70 bucks, the average consumer expects integrated graphics, fax, voice mail, e-mail, data connections to all services, with a two-year no-cost warrantee, and then they make you toss in CD-ROM support with the $1 bonus coupon offer.

The Day Intel Went Flying

There is an old industry rumor. "Gary Kildall came to Intel and offered CP/M for their microcomputer operating system, and Intel rejected it." CP/M became wildly successful in the years that followed, and Intel received the brunt of the "Ha, Ha's."

Like most computer folklore, that's not really how it happened. By 1975, Intel's microcomputer software group had changed control. Hank Smith went to Venrock Investments to become a very successful venture capitalist, and, in fact, was instrumental in funding the two Steves at up-and-coming Apple Computers in Cupertino. Hank Smith, manager of that original Intel microprocessor division later employed me as a consultant to evaluate their investment at Apple Computers.

Bill Davidow took over Hank's position as leader of the Intel microcomputer group. Bill was a good manager but wanted all development to be internal to Intel. I was a wild card for him, and although we had no personal conflict, Bill stopped my consulting service contracts. This was, coincidentally, at the time of CP/M's inception.

Instead, Davidow tasked Ken Burgett, one of those two initial Intel programmers, to build a proprietary operating system. Ken built a good one called ISIS that later became Intel's platform for their own PL/M compiler. I believe it was a good decision by Bill, and a godsend for me and CP/M.

I went high-tailing it back to Pacific Grove and didn't spend much time with Intel after that. Maybe a consulting day or two, until the days of Digital Video Interactive (DVI) in 1990.

Ok, so, I've got this OS that I called CP/M that can be made to fit almost any microcomputer design. More important, I've got the "tool set" for someone to make their own programs without using a mainframe. And, it was priced right at $70.

CP/M and Software Pricing

Let's look at the mentality of the day in 1976. On one of those consulting days in 1976, I sat in a meeting at Intel with marketeer Jim Lally as he discussed pricing.

Says Jim, "Ok, so we sell the Intellec 8/80 for, say, $8,000, and then we sell them a fast paper tape reader for another $2,000. That makes $10,000 off the top. Ok, then we wait until we've sold a bunch of these paper tape systems and then introduce the floppy disk drive at $2,500 and ISIS software at another $1,000."

I was dumbfounded. This was a direct attempt to block advancements in our society's technology for the profit of Intel. It was a good lesson for me. I protested. I wasn't listened to, or even considered. But, Jim Lally is now a very successful venture capitalist. I think this is a statement about how certain people manage to get other's assets.

Accumulation of wealth is made by profits of those in possession of goods, and that makes the goods flow more smoothly. Certainly, Intel found that out, and probably mostly from Lally in those days. I never really liked Jim's approach to marketing, but this pricing issue is what made CP/M a success over ISIS and other proprietary designs.

I enlisted the help of my wife Dorothy. I asked that she take care of mailings in response to the DDJ ad for CP/M. The developers had a choice. They could buy Intel's hardware and software development system priced by Lally at $13,500, by the time they tossed their "old" paper tape reader, or they could build their own and buy CP/M at $70. Hmmm... what would you do if you were a budding scientist?

CP/M was an instant success. The operation in the tool shed moved to Pacific Grove in a small building with an isolated cupola overlooking the normally placid waves of Monterey Bay. Dorothy now went by "Dorothy McEwen," to avoid an association with a "Mom and Pop" operation, and did a fine job of mailing CP/M systems, billing, and invoicing. She contributed in a diligent and directed manner.

But, the success of CP/M was "bang for the buck." It was small, it was fast, it was efficient in operation, it would run on all Intel computers and Zilog Z80s, and it was cheap. No other software product had been priced that way before. Ok, CP/M's price came up to $100 per copy with version 1.4, but no one seemed to care.

Oh, by the way, we did version numbers like this: the first digit was a "major" revision, like 1 in the 1.3, and the "minor revision" was an "update" like the 3 in 1.3. This worked great.

You charge the manufacturers and customers a "minor" fee to get the minor revision, and then issue a "major" revision, like CP/M 2.0 and charge a major fee. That became the way microcomputer software was labelled, and for that purpose only. See, I did learn something from Jim Lally.

Launching the New DRI

In 1978, Digital Research reached $100,000 per month in sales at about 57% pre-tax profit, a milestone that allowed our company to move to new quarters in a converted Victorian home at Pacific Grove.

This Victorian spawned a new, growing phase of Digital Research. It was fun. It was exciting. It made new relationships between people. It made a lot of marriages, and they made a bunch of babies. I kind of like that part.

The Victorian home filled with DRI people had spunk. It was alive. It was active. People moving paper and products. People defining the industry through their work.

On my 39th birthday, all of the twenty or so DRI employees got together and held a party for me. That wasn't such an event, because we had parties every Friday afternoon at that Victorian. They gave me roller skates. You know, the kind that look like tennis shoes mounted on a Formula One car.

The party got a little rowdy, which was ok, but I was tasked with getting more champagne. Ok, so I have on my new roller skates, and I head out to the liquor store down the hill about three blocks away. I kept stumbling over those little acorn nuts that drop constantly from trees in Pacific Grove. But, finally, by the grace of God, I made it to that seemingly distant liquor store.

I found temporary haven by hugging a telephone pole near the liquor store for refuge. As I held tight, a guy in a Corvette came by and said, honestly, "hey, man, you're a *GREAT* skater!" It may have been the best compliment of my life. And, Lord knows how I got that champagne back, but I'm sure it wasn't entirely upon those skates.

Coincidentally, DRI had several "39th" birthday parties for me in the years that followed, and I can tell you, candidly, that DRI knew the meaning of "Party Hardy."

By 1980, DRI had sold millions of copies of CP/M to manufacturers and end-users. We held symposia for thousands of clients at Monterey to train them how to use and adapt CP/M as well as the new MP/M multitasking system and CP/NET. Digital Research was on a roll.

PL/I and New CP/M Versions

Of course, CP/M went through many "versions" like the major and minor ones that I already talked about. In 1978, there was a lot of pressure to make a new one. The pressure came from manufacturers that were using the smaller 5-1/4" disks, like the ones that were introduced with Woz's Apple II. And, our OEMs were pushing for a new standard BIOS for the small disks.

Meanwhile, I had been working intensely on my PL/I project, one that took nearly two years to complete. I've talked about PL/M, the computer language that I designed for Intel's microprocessors in 1973, and now I'm talking about PL/I.

I did not design PL/I. Although the names are similar, these two languages are quite different. My PL/M was a "systems language" for writing tight microprocessor applications, like operating systems and utility programs, such as program editors.

PL/I is a language that lets programmers write scientific and data processing applications. It's an entirely different breed of programming language tool.

CP/M version 1.4 was a success in the marketplace, but it really needed a good high-level language. I gave up on PL/M because, even though it was a high-level language, in the sense of today's "C," it didn't give application programmers the proper operations to make commercially acceptable programs, like office accounting and billing.

We're talking here about new CP/M versions, right? Not, about programming languages. But, I felt that PL/I was necessary to advance the PC platform of applications. And, besides, Tom Rolander was busy on MP/M and CP/Net to support future processors, like Intel's new 16-bit i8086.

Why the Heck PL/I?

CP/M was a stable software item in those heydays of 1978, and it certainly held the title of the Operating System Standard. Ok, CP/M needed an upgrade from version 1.4, which held for a few years, but I felt that greater need to supply a programming language to promote applications under CP/M.

There were not many choices, and still aren't. The standard languages were FORTAN, COBOL, Pascal, PL/I, and C. Ugggh. None really worked for all microcomputer applications that well.

I got a call in mid-1978 from Dr. Robert Freiburghouse at Data General Corporation near Boston. He was trying to pare down IBM's PL/I, a dinosaur every bit as well done as Disney could have produced.

Please understand that I had to live with that stupid, committee-designed, IBM PL/I language on the IBM System/370 for three years by force of conscription. Next to BASIC, it was the worst language ever designed, though running a very close second.

Bob wanted me to attend an ANSI standards committee for a new PL/I they called "Subset G," or PL/I-G. Bob had known about my design of the PL/M compiler for the Intel microcomputers and figured, mistakenly in my estimation, that I might have something to contribute to the new PL/I-G standard.

Freiburghouse's Standards Committee met at the La Playa Hotel at Carmel, California, in mid 1978. I can't say that I was downright nasty at that meeting by telling them that "PL/I, by the way, appears as a somewhat overloaded One-Man Band that should, by its character and value, be immediately placed in prominent display at the Smithsonian nearly next to the Reptiles."

Ok, so that was a little nasty. I was particularly unimpressed by the whole affair, but, what the heck, the meeting was only down the street from DRI, and Freiburghouse threw in a free lunch.

But, in listening, I found that Freiburghouse had worked-up a spec' for a pared-down version of PL/I that looked and acted more like Algol and Pascal, with all the business features of COBOL in a clean, complete, and quite usable programming language that might apply to microcomputer applications.

For one thing, PL/I-G adhered more to the rules of the new genre of programming languages, but it kept all the good points, like automatic access to BCD numbers (the kind that accountants like) and good string processing primitives.

Ok, so after complaining about PL/I for a committee session or two, I finally decided I liked Bob's Subset G. I really hate it when I have to admit that I am wrong. Well, at least, Bob got some jollies.

I engaged the PL/I project in that year of 1978 to make "Subset-G" to support our customers under CP/M. I estimated that the project would take Bill Byerly and myself nine months.

Being a truthful person by nature, I must admit that I still hang my head in shame, for that PL/I-G didn't hit the streets for about two years.

"Hey, Ok, so I went a little over schedule. I mean, that can happen to anybody, right? And... And... Hey... my computer was down all week, and I can't get that code from Bill that I need before I can work on the next piece for Bill that he keeps asking me for."

"I mean, it's not my problem, it's Bill's. Or, if it's not Bill's it must be that salesman at Intel. I don't remember his name, but he keeps promising that memory board, and I can't load the linker 'til I get it, do you know what I mean? Hey, I even drove up there, but my car broke down in Gilroy and that's gonna cost, say, two grand, and don't you think I can get that Christmas bonus early, 'cause its almost Halloween, you know."

"Yeah, that's it, me and my wife are coming to the DRI party as Russian Cosmonauts, I'm Yuri Gagarin and...".

This is how I learned Engineering Management. I've not only heard all the excuses, I can say, quite proudly, that I actually made up most of those excuses myself.

To my defense in tardiness of my PL/I delivery, I make the analogy between building compilers and having babies. To my great fortune, I have not had the opportunity to experience the latter. Here's the analogy.

You can only design and write a few compilers in your lifetime. Otherwise, there are too many of your own creations around to take care of, and all you hear are sounds of crying and screaming when things aren't going right. Besides, although the design concept may be great at the beginning, they take too long to hatch, and meanwhile, you get headaches and can't sleep at night. But, of course, when they are finished, they are always perfect. I've had my last compiler.

But, I would say that this PL/I-80 compiler was quite an elegant design, as it incorporated all of my compiler code optimizing techniques. It made PL/I Subset-G language statements into quite perfect assembly language code for the 8080 to ingest. Hey, cut me some slack and let me brag a bit.

But, the whole project, although it came out properly, was a huge diversion. CP/M needed a revision during that two-year PL/I project. So, the pressures from OEMs and end-users stopped my PL/I efforts for a time to make CP/M version 2.2, and that allowed all those OEMs to make specialized BIOSs to fit other diskette drives that were, unfortunately, mostly incompatible. PL/I-G came out from Digital Research in 1980 and was, by excess, the best compiler built for the Intel chip set.

My interest was in getting a great application tool for the CP/M base through PL/I-G. It was an excellent language for programming applications, and assembly language programmers could not do better than the machine code it produced. Many companies took it up, but there was a larger force in action.