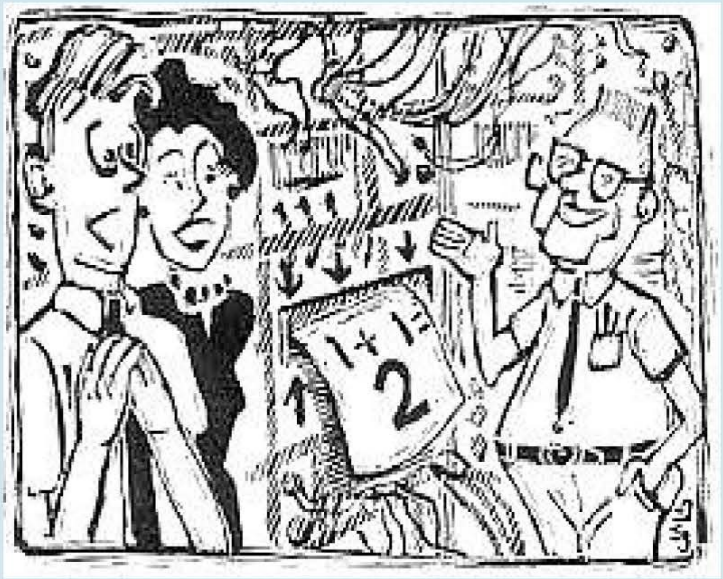
B.C.

Before The Chip



Before the microchip there was the transistor. Before the transistor there was the vacuum tube. And all of this was before the internet was a glint in Bill Gates' eyes.

Al Strano

Many thanks to Julia Morgan-Scott for the illustration (submitted to Mimosa Magazine, 2001).

Table of Contents

Before the Beginning

One night around 2 a.m. in April of 1966 I was silently sticking letters into square holes at the US postal airmail facility in Philadelphia, when I was abruptly disturbed by my shift supervisor Carol Foxx.

"I want you to get out of here," he announced.

This came as a surprise to me since I thought he liked me and that I was doing a good job. It turns out that I was right and that was the reason he said what he did. "Foxy," as we called him, was the commander of the local branch of the Catholic War Veterans. I was unaware that this group had been lobbying congress to pass a new GI Bill. Anyone who had served in the US Armed Forces before 1955 was eligible for a set of benefits when he left the service. Among these benefits were educational assistance, cheaper mortgages and other financial allowances. Yet I had served in the United States Marines from 1959-1963, thus I had not been included. Now it seemed this had all changed because of groups like Foxy's. Lyndon Johnson had just signed a bill that covered "peacetime" members of the military. Because I had no knowledge of this activity I was unprepared for Foxy's statement. He had obviously given it more thought than I had. "You're smart and a good worker. You can do a lot better than this place," he said as we both surveyed the rows of dusty sorting bins and cigarette scarred benches that made up the facility housed in the bowels of Philadelphia International Airport. It was not the most beautiful place to work. My future at the post office was secure, but it contained no great prospects.

"What can I do?" I challenged.

"Go to school!" he quickly retorted.

"Where?" was my less than eloquent answer.

"Bill goes to computer school in New Jersey," he quickly added pointing toward one of my fellow workers at the facility.

And so two days later after telling my wife I would be home late, I accompanied Bill across the Walt Whitman Bridge to Cherry Hill, New Jersey, the international headquarters of Radio Corporation of America and home of the RCA Technical Institute. The institute was only a few years old and had been

started by RCA to train technicians for the rapidly growing computer industry. Bill introduced me to the dean, who took me on a tour to explain the school to me and its requirements. I was delighted to learn that the only educational necessity was a high school diploma, which I had received in 1959. I was enrolled in the next class and spent the following 48 weeks enduring the most rigorous schedule of my life. I started to work at 11 p.m., left the airport at 07:30, arrived in Cherry Hill with just enough time to buy a cup of coffee before class, then attended classes until 1:00 p.m., drove home, ate lunch, did my homework, went to bed, got up at ten p.m. and started the whole cycle over again.

It wasn't easy, but slightly less than a year later, with a certificate of completion in my hand, I was ready to start my career in computers, a career that eventually brought me financial independence, world travel, a new wife and many fond, interesting memories. It is my experiences I wish to share with you.

But before the career started, there was the matter of finding a job. The school was very helpful in this respect. They boasted a 100% placement record, which they did not wish to see blemished. So much so that the last month of school was primarily spent in job interviews. Because the demand for technicians was so great and RCA's reputation was so good, recruiters from all the major computer companies came offering employment. The recruiters would spend an hour or so presenting their company and its benefits to us, then give us their own version of an aptitude test and finally announce which of us they wished to remain for further interviews. Because of my class standing and ability to take tests I was chosen for this honor almost everyday, an honor with mixed blessings. The interviews took place after lunch, which meant that it cut into my precious beauty rest; with my looks I needed all I could get. But the whole year had been spent for this opportunity and my future was at stake. (Some of my classmates were also reluctant to stay for the interviews because they did not wish to miss post time at Garden State Racetrack, which was across the road. Placing a bet on the horses was more important to them than a job.) Our class of twenty was made up of an odd assortment: four of us were military veterans using the GI Bill to pay for school, but many of the younger guys were there because the Vietnam War was heating up and they were seeking a draft deferment. So, while I was using the GI Bill, they were trying to avoid being eligible for it.

Eventually, I received offers from IBM, GE, CDC, Philco, Boeing, DEC and Univac. In 1967 IBM was by far the most prestigious computer firm controlling 85% of the computer market. I and two other students, all of us veterans, were invited to Poughkeepsie, NY, home of IBM, to be interviewed for employment in the manufacturing facility. From the moment we arrived everyone we met, uniformly dressed in their dark suits, white shirts, somber ties and clean-shaven faces, attempted to impress us with the size of the company, our good fortune at being interviewed, and the delight of living in Poughkeepsie, one of the richest company towns in the USA.

If we accepted jobs from IBM we would live in Poughkeepsie surrounded by IBMers and all our activities would be under the eye of IBM. Our obligation to the company would not stop at five o'clock at the gates of the facility. We would belong to IBM and our careers could be affected by our outside activities. Not that I have ever done or have planned to do anything socially unacceptable, I just didn't like the idea of "Big Brother," and the strict dress code rubbed me the wrong way. Today things have changed; however, back then Thomas Watson was the head of IBM and his iron fist and rather puritan ideas on dress and conduct pervaded the company, as well as much of the industry. Stories of people being fired for long hair, flashy clothes or drinking in public were legendary. Having seen it first hand I was not anxious to pull up stakes and start my career in Poughkeepsie. When I conveyed my feelings to the personnel manager, he quickly replied, "If not here, how about East Fishkill?" He obviously missed the point. Besides, a place called Fishkill certainly didn't sway me. Years later during the 1992 presidential campaign, Ross Perot was attacked as being anti-Semitic because his company enforced a "no beard" policy with the result that a Jewish man had been fired. I'm sure many old IBM employees knew why the policy was there and were surprised Perot didn't explain that if you competed against IBM you needed to be "more IBM than IBM." I feel this might have helped his chances rather than the obviously false denials.

But back to 1967. I returned home convinced that I would join ranks with the Davids to fight the Goliath of IBM. I narrowed my choices to GE and Univac. GE would have paid me more money, but would only offer me a job in Washington, DC. It was tempting, but starting a new career away from home introduced an extra level of risk. So, I decided to check out Univac before making a decision.

Hoping to make a good impression, I arrived at Univac's Philadelphia office a few minutes after ten for an eleven o'clock appointment. When I confidently told the receptionist I was there to meet Mr. Reilly for a job interview, she hastily gave me a temporary pass and told me, "Hurry to the third floor conference room. The meeting is due to start at ten." This totally caught me by surprise. I was sure the meeting was for eleven and it was to be a one-on-one interview with my perspective new boss. None-the-less, I raced up the stairs arriving at the conference room just as the doors were closing. My entrance drew some less than friendly stares, causing me to quickly grab a chair. There were over twenty young men in the room, most of who were casually dressed, while I had my brand new interview suit on.

Sure enough, Mr. Reilly was in charge of the meeting. I was stunned when he started to describe "living conditions in Vietnam" and how safe the area where we would be working was. I was outraged. I made up my mind to leave and head for GE; Washington, DC. was a lot better than Vietnam. I swore the recruiter told me the job was in Philly. This was obviously some trick to lure me in and then ship me off to the Far East.

As I started to leave, one of the men in the rear of the auditorium followed me out into the hallway.

"What's your problem," he asked, obviously displeased that I had arrived late and was leaving early.

"I was told I would be offered a job in Philadelphia," I growled, leaving no doubt that I was not pleased with the current state of affairs.

"What's your name?" He checked his list, then looked at another sheet of paper before saying to me with a smile, "I'm really sorry. The receptionist sent you to the wrong room. Gene will be with you as soon as he is finished here." He got me a cup of coffee and took me to the proper office.

My interview with Gene Reilly went so well that two weeks later I reported to work, where I discovered that people at Univac disliked IBM with a real passion. The attitude being that IBM had stolen the lion share of the computer business from Univac, which should have been number one since it had been founded by the same people who had built the first electronic computer "ENIAC" in the late 1940's at the University of Pennsylvania,

where it still resided just a few blocks from the downtown office.

I have often felt my career in computers was controlled by a series of mostly fortunate events. As you will later learn, my fate might have been a lot different if I had gone to work for GE.

CAMIO

Anyone who has gotten anywhere near a computer knows that the whole industry is made up of acronyms, from IBM (International Business Machines) to HAL (the computer in 2001, which was slyly named using the preceding letters of the alphabet for I, B, M) and my favorite NBI (nothing but initials). The first acronym I was taught at RCA was CAMIO, which was a crutch to help remember the components of a 1960's computer system.

"C" was "control," the part that executed instructions and made the whole thing work. Computer instructions are a set of commands used to manipulate data: shift it, store it, fetch it, etc. In the 1990's we all know of the Pentium Processor. In between it was known by another acronym: CPU, Central Processing Unit. In the 60's and 70's the CPU was about as big as 5 or 6 very large refrigerators and required large amounts of electricity and air-conditioning.

"A" was "arithmetic." Early computers could not perform addition or any other mathematical function without an additional box to assist the main control unit (CPU). Without this extra accessory programmers were required to write programs using the basic instructions included in the standard machine. (For example, before the % button was added to standard calculators the user had to provide the appropriate multiplication, storage, addition or subtraction functions. This early manual programming was even more primitive.) As the price of hardware dropped in the 70's arithmetic units became incorporated into the CPU and reference to these devices disappeared.

"M" was "memory," the portion of the machine where data was temporarily stored in order that the control unit could manipulate it. Like an "Etch-a-Sketch" or a slate, it could be used and re-used as necessary.

"I" stood for "input," which describes various devices that could be used to provide data to memory so that it can be manipulated or cause the control unit to perform some function.

"O" was for "output," once again incorporating various devices, but this time their role was to receive the data from the memory and display it in various formats.

Example: **INPUT** provides "names" and "hours worked" to **MEMORY.**

MEMORY holds information.

CONTROL manipulates information.

ARITHMETIC performs calculations.

OUTPUT displays payroll checks from **MEMORY.**

If you did not know that before, you have just begun to become a computer expert. I can still remember the pride I felt when I identified them correctly on our first quiz at RCA.
Now it's YOUR turn!

--> Can you list the five pieces of a 1960's computer?

Number Please

A common expression among computer professionals is "it's all ones and zeroes" or "it's only ones and zeroes," rather silly sounding, but very accurate sayings. Because, believe it or not, a computer from the smallest laptop to the biggest Cray super-computer basically understands only two states -- "on" or "off." These can be translated as "one" or "zero," called binary digits or "bits."

The earliest machines used relays, (electro-mechanical switches) a closed relay equaled a "one" and an open relay a "zero." Then came along electronic tubes. A conducting tube equaled "one," a non-conducting tube a "zero." Transistors and chips have merely emulated the earlier machines in a smaller, faster way.

Because of this, one of the first things a new computer student was faced with was the dreaded task of learning binary arithmetic. Instructors joyfully laid down long strings of ones and zeroes, 100110111010, like that and asked students to calculate the decimal equivalent. The strings would get quite long and the system to accomplish the translation was known by the highly technical term of "dibble dabble." Yes, highly trained and valuable engineers spent hours dibbling (or was it dabbling?) in order to come up with a human recognizable number. And of course, binary math (addition, subtraction, multiplication and division) was enough to keep one awake at night.

Finding it impossible to converse with another human in only ones and zeroes, something had to be done. One of the first innovations was "octal" representation (base 8 as opposed to base 10 = decimal or base 2 = binary). Three binary digits were used to represent one octal digit. It helped a lot.

```
binary  octal
------  -----
001   =  1
010   =  2
011   =  3
100   =  4
101   =  5
110   =  6
111   =  7
and
1000  = 10
```

Whoops! What happened to 8 & 9? Answer: they don't exist. 10 (octal) = 8 (decimal) and 11 (octal) = 9 (decimal), but you never say it that way. It's like counting on your fingers, but not thumbs. To make the representation easier, octal numbers are followed by a subscripted 8, so 443_8 (which is not "four hundred forty-three," rather "four four three octal") or 100 100 011_2 equals 291 decimal.

Nothing to it, right? Well engineers found octal to be limiting, so they expanded one more step to represent four binary digits by one symbol and called it hexadecimal (or base sixteen!). From 0001_2 to 0111_2 is just like octal, but $1000_2 = 8_{16}$ and $1001 = 9_{16}$. That should make you feel better... until you discover that

$1010_2 = A_{16}$,
$1011_2 = B_{16}$,
$1100_2 = C_{16}$,
$1101_2 = D_{16}$,
$1110_2 = E_{16}$,
and $1111_2 = F_{16}$.

Are you still there? Have you retrieved the book from the trash can??
Good for you! Now you know why computer people make a lot of money. They can't balance their checkbooks, but they know that 10_{16} is sixteen and 10_{16} is eight and 10_{16} is two.

Eventually, calculators were manufactured to perform these tiresome tasks. But in the 60's, before the chip, calculators were very scarce and expensive. I saw one in the 70's that was pretty efficient, because it incorporated several number-based systems; it became the standard tool. However, since it was designed for the very same task of working in the binary number system, performing decimal calculations was painfully slow resulting from constant conversion between the two systems. It was almost worth having a second calculator just for decimal arithmetic, but as I said, these tools were expensive and difficult to justify (just imagine the "boss" saying, "You have a million dollar computer in there! What on earth do you need a calculator for?").

This book is meant to be fun; therefore, I will not torture you anymore. If you do wish to learn more about systems of numbers there are tons of more serious books on this subject that will guide you along the way.

One last important thing to remember is that zero is a number. The purpose of this chapter is to make you familiar with the idea of bits, i.e., ones and zeroes, which are the whole foundations of computers.

When RAM was CORE

No conversation between two computer aficionados is complete without some reference to RAM, the acronym for Random Access Memory (the "M" in CAMIO). The more the better. Millions and millions of "bytes" (the term for units of 8 bits). The more you have the faster you can do things and the more things you can do. Every time you wish to use a new device or program, you need to buy more RAM. So much so that most users can upgrade their own memory by themselves in a matter of minutes.

But real memory was CORE, surprisingly not an acronym. The first computer memory was made of small ferric donuts called cores. Ferrite is a metallic compound that has the ability to take and release a magnetic charge quickly. Each bit of memory was a unique core that you could hold between your fingers and peek through the hole.

The cores were arranged in such a way that each had three different wires running through its center. One was the X wire, which determined the bit location within a 32-bit "word" (unit of 4 bytes). The second wire was the Y wire, which determined the word address within a 16,000-word block. The final wire was the sense wire, which enabled the control unit to detect whether the particular bit was charged or discharged; "charged" equaled a one and "discharged" equaled a zero.

Did you follow that? I would suggest you read that paragraph again. If not, remember where it is.

I owe you some explications. First I referred to memory in "words," which is how we all described memory in the early days. Most memory was divided into 32-bit words, at least the machines from IBM were and so were the Univac machines I worked on. Some manufacturers used different numbers of bits in their "word," from 28 to 36 as far as I remember, but there have been others. Since most of the machines were IBM 360's, I'll stick to 32-bit words. The use of the term "byte" to describe memory did not come into general usage until the late 60's; even then it was used to describe mass memory units (discussed in another chapter). But to simplify things, everyone now uses "byte" to describe all units of storage. Another interesting term was "half word," or 16 bits, 2 bytes. The half word is definitely passé, and the real pro knows about the "nibble," 4 bits. What else is left?

We also referred to memory in thousands of words or kilo-words. Up until 1975, the biggest machine I had worked on had 128K words, or 512K bytes or half a megabyte. What the Greeks had to do with it I'm not sure, but we've always talked that way. The comparison in size between then and now is nothing less than astounding. In the first machines, 16K words of memory took up a cabinet the size of a walk-in closet. In fact, you had to walk inside in order to perform repairs.

My wife's favorite story is that her mother was interviewed for a job by Dr. John Mauchley, creator of ENIAC, inside of a Univac I. It was a very hot day, she was pregnant and the doctor thought she would be more comfortable inside the air-conditioned Central Processing Unit. (She got the job.)

Even in the late 70's memory took up huge amounts of space. The SDS Sigma 7 I maintained in Philadelphia during the riots of 1972 had one memory cabinet that was only half full and I decided that if the rioters attacked the computer center I would hide inside the cabinet.

Once on a trip to Univac's manufacturing center, I saw the women who hand-wired the core memories. This was in the 60's, so when the supervisor said, "We hire only women because men don't have the temperament for the job," it did not provoke a sex discrimination charge. As I watched these women deftly wind the wires through the cores I noticed one woman at the end of the line who had a large illuminated magnifying glass. "What's she doing?" I asked. "She fixes the mistakes," was the answer. This process, in addition to the expense of electricity needed to drive these components and the coolant required to counteract the heat generated, motivated manufacturers to look for different ways to create memory. Some of these were film, wire, rods, and some real far out things like gases and magnetic bubbles, even silicon. Whoops, that one worked!

My most vivid memory is rod and wire memory. Univac was working with a way to magnetize the sense wire, thus making the core unnecessary. It was working pretty well with small amounts of memory. We had been using the memory for about a year when faults started to occur at a very fast rate. So many that we were running out of spare parts. Since Philadelphia is real close to Univac's engineering center in Blue Bell, PA I was one of the field engineers involved in finding a solution, something that occurred on most new products and an experience I enjoyed. This time the problem was in fabrication. The memory

boards were sheets of laminated fiber glued together with the wires sandwiched in between. It seems the pressure and heat used to bind the sandwich had been insufficient, resulting in the glue not drying in time. So it continued to spread eventually touching one of the wires causing a fault. The emergency solution was to remove the board, which was about 2 feet square and 2 inches thick, take a large tweezers, then gently pull on the exposed ends of the wires. You could actually feel the wire break loose from the glue when the "bad" wire was pulled. Although it was possible to locate the exact wire to pull, human logic dictated that while you were at it you might as well pull them all. There were 4,000 bits on each board, so that process took awhile. One day I was engaged in showing some of my colleagues this procedure when my boss came into the workroom. He had heard, "Al's fixing a memory board," so he rushed over to see the miracle himself. Fixing the boards in this manner saved his department thousands of dollars for each board, so he was very interested. After a few minutes of silence, no one wishing to disturb a genius at work, he sighed, "is that all? I could do that !" When I offered him the tweezers he declined and left the room.

Unfortunately, this fix was only temporary owing to the glue continuing to spread. So in a few weeks the board would be faulty again and no one, including me, was interested in being a "miracle worker" that often.

I remember a college professor telling me in 1974 that Bell Labs was working on silicon memory and that one day we would witness millions of bits all contained on one small board. The only thing he got wrong was that now there are more bits on even smaller boards.

A discussion about computer memory would not be complete without mention of the parity bit. No, this has nothing to do with the NFL trying to get all teams to finish the season at 8 & 8. It is the method used to ensure the validity of computer data in memory. Each 32-bit word actually had 33 bits, the extra bit is controlled by the memory control unit. If during a "write" or "store" operation into memory the word contains an even number of "1" bits, the parity bit is set to "1" to make the total number of "1" bits an odd number. Wondrously, this is called "odd parity." So, if the number of data bits is odd, the bit is not set. You could use an even parity system, but then you would have the possibility of a word with 33 zero bits. This wouldn't bother the machine, but we humans feel better

knowing at least one bit is on.

If during a computer memory "read" or "fetch" operation a word is detected containing an even number of set bits, the results are rather dramatic. A "parity error" has occurred. Depending on what part of the memory was affected, the following may have happened. If it was in the program data, the faulty program was aborted. If it were in the main operating system memory, the whole computer would stop. In some cases the computer would attempt the operation again or even as many as ten times. This happened in a matter of thousands of a second, so it still seemed instantaneous to the human brain. Sometimes the glitch was momentary and the system proceeded, but often it was a hard fault that then killed the system. It was now a job for your trusty field engineer, who never arrived soon enough and never fixed the problem quick enough. How did I know? My customers told me so. The repair time in the old days might take hours, not the actual changing of the component, but the task of locating it. A 16K block of memory was not only made up of cores, but hundreds of component boards, each of which could be the culprit. I'll cover more on trouble-shooting in a later chapter. Just realize that today, all you normally do is change the whole memory. It's a little expensive, but fast.

Speaking of fast, that's what we called RAM in 1966, "fast memory." It was fast because ferrite cores had to reverse their state in order to be read. The process was called "destructive read out," not that anything was injured or broken but the data was actually inverted when the "read" operation took place. So, in order to maintain the integrity of the data, it had to be rewritten or reinverted, requiring a slow two-step process. Electronic, or fast memory, was non-destructive. That alone made the memory faster. All machines had some fast memory to be used for program execution. In the 60's it was very expensive, so only a minimal amount, normally 16K, was installed.

In later applications, when the price became a little more reasonable, large chunks were utilized so that bigger pieces of program could be loaded and thus speed up execution. This was called "cache memory," because of the limited size miserly program techniques were required to make best use of the facility. Sloppy programming could actually slow the computer system down. More on that in another chapter.

Well, that was pretty heavy-duty stuff!

--> Do you remember how many bits in a word?

--byte? --halfword? --nibble?

Massive Mass Storage

Since about 1975 the computer industry has had an obsession with making things smaller and smaller. That is why you can have so much power in your PC. But it wasn't always that way. For one reason it wasn't easy to do, but the other was no one thought it was important. Power was important. Speed was important. Accuracy was important. And new innovations and features were important. But size was last on the list.

One manufacturer in 1975-76 went so far as to put a small state of the art processor in an almost empty cabinet, telling the bewildered support staff that they didn't want the customers to feel cheated. As it turns out, this particular product failed because, among other things, it occupied too much floor space. Of such genius careers are made.

But in the 60's "big" was "beautiful," especially in mass storage devices, also known as disks and drums. I had the pleasure of seeing and working on some of the best (a term that might be debated) hard disks, as they are called these days. These units were used to store data that the computer could access quickly.

The first I remember was at Univac. I had only been there a week when my boss sent me to a large supermarket account to help out on a problem. It seems that there was a fault in a rotating memory unit. What it looked like on the inside I had no idea, but I was told it contained mercury. It looked like a round iron wash tub, was a dull steel color, rounded on the bottom and flat on top, with at least six large cable connectors on the top, each of which was as big around as the average beer can and contained up to 32 wires within. The reason my expertise was required was that it weighed over 100 pounds. The old unit had to be lifted out of the top of a seven-foot cabinet and of course the new unit lifted back in. It took five of us using two ladders, grunting and sweating, to make the switch. Of course we were all dressed in business attire, white or blue shirts and ties. Univac wasn't as strict as IBM (we could wear sport coats), but all companies required some level of business attire.

Once the unit was secured with 3/4" bolts and the cable connectors plugged in their sockets we turned the unit on. "Let's go to lunch," announced the on-site field engineer. "Aren't we going to test it?" I queried. "Won't know until it

warms up, which takes about an hour and a half," he informed me. I guess it worked because I wasn't called back to display my muscles again.

But that was not the biggest device I saw at Univac. That would be the Fastran. It was so big they had to test concrete floors before moving it across loading docks. One could see into this thing. The front, about ten feet wide and five feet high, had a smoked window. The drum looked like a barrel mounted sideways and appeared to be grooved and turned at 870 RPM. The read-write head was massive, about 1 foot square, with large hydraulic hoses attached to its rear. When it moved one could see it groping its way across the drum searching for it's next location, taking tenths of seconds. (Today we measure access time, or head movement, in millionths of a second!). For this reason, programmers were encouraged to request data that required the heads to move as infrequently as possible. However, once the heads were in place, you had access to 64 thousand bytes (64 KB) of data. The total capacity of this "Bad Boy" was 60 million bytes (60 MB).

Years later when I was working for SDS, I came across another monster, the Data Products disk, which was laid out in 8 round steel platters one above the other about six inches apart. Each platter had a diameter of four feet, two inches thick! It also had a window in the front so one could watch the disks spin around while the heads sought back and forth across the platters.

I guess now is a good time to explain something about disks. A disk consists of one or more platters that contain information in concentric tracks. Think of a phonograph record, but with concentric grooves, not spiraling ones. Once the disk is spinning, an armature launches an arm (similar to a phonograph arm) with the heads attached to the ends. These heads are similar to those used on CD players, VCRs, cassette recorders.... If there are a stack of platters, then a set of arms are launched in parallel. When the arms reach a pre-determined position, the heads are unlatched. They remain connected to the arm, but float freely, riding on a cushion of air provided by the spinning disk. The read-write heads actually fly above the surface of the disk (to follow the analogy, the phonograph needle does not touch the platter). The distance between head and platter is very small, close enough that the head can either read images magnetically off the platter or likewise write images. Like other things that fly,

heads crash! And when they do, something is usually damaged, most often the head, but in bad cases both the head and disk are damaged.

One of the above crashes was the cause for my introduction to the Data Products disk. A head had crashed scratching the surface of the disk. Both the head and platter needed to be replaced. The head replacement was simple. The three-foot arm holding the head was withdrawn so that a new head could be attached to the end. The disk platter was a different matter. We had to unbolt all of the platters above the damaged one and remove them until we reached the "bad" one, then remove it, replace it with a new one, and then replace all the others, one at a time making sure we kept them in the correct order, right side up and at the proper position. There were four bolt holes in each platter, so it was possible to put the platters back in four different positions, only one of which was correct. I had worked as a truck mechanic in the Marines; this reminded me of changing tires on a five-ton truck. Once we had bolted the platters back into place, we lowered the hydraulic cover that had been raised to gain access to the disk and started the device. The engineer I was working with had a funny smirk on his face when he pushed the button. The torque of 8 spinning platters was so great that the drive had to shift gears to get up to full speed. When it did, one could feel the room tremble as the cabinet tried to move across the room like a gigantic unbalanced washing machine on "spin." Our bodies shifted position as we felt like the floor was moving under our feet. Finally it reached full speed and settled into a steady rhythm humming like a jet airplane engine.

Another adventure concerning this same disk was disk cleaning, which had to be done once a month to prevent data loss and crashes. The procedure involved replacing the normal arms with one special arm containing a scrub brush on the end of it. We soaked the scrub brush with cleaning solvent, attached the arm, ran a special cleaning program to run the arm back and forth across the surface of the spinning disk like washing dishes. In order to dry the platter, we wrapped a special lint free cloth around the head and ran the cleaning program again. Of course, we had to do this for each surface. The used cloths were great for car washing.

The last steel disk I worked on was called the high speed RAD (random access disk), which was designed and built by Scientific Data Systems. SDS eventually became part of Xerox

and then Honeywell, but that's another story. I dearly loved the RAD because I managed to gain a reputation as an expert on fixing it. The RAD consisted of two steel disks positioned one behind the other like a set of wheels. One could only see one side of the front wheel through a Plexiglas window located in the front of a removable metal container two feet cubed weighing two hundred fifty pounds. It was housed in a large gray refrigerator sized cabinet, which also contained the interface electronics and "safe." The safe controlled the starting, stopping and spinning of the RAD, while keeping it safe from contamination by pumping air through it. The safe contained high voltage control cards and regulators. It almost never failed, except during startup. Most big computers run all the time, so the RADs might run continuously for months. But every so often, normally for maintenance reasons or major power failures, one would have to stop and start the RADs. The main site I supported had 10 RADs, a very large number for one site. You can imagine how worried I was when we had to shutdown all ten at one time. Invariably, one of the safes would fail. Sometimes they would just fail to attain proper speed, but then on rare occasions they would blow up! Not actually explode, rather the high voltage control components would pop! causing a loud noise, a little smoke, and an hour or two of work for me. The safe was mounted four feet above the bottom of the cabinet and was two feet high, three feet across, and one foot wide. It weighed about fifty pounds and was a lot of fun to manhandle around. On one of these fun occasions I was in the process of lifting a safe into position when the assistant director of the site came into the computer room to see why the computer was not running. As he watched me grunting to get the unit in place, he asked, "Is it a hardware or software problem?"

I looked at him in disbelief and realized he was serious. "I think this feels like hardware," I said sarcastically as I hefted the unit into place.

He then said, "I hope you're right" and left in a very officious manner. I couldn't help laughing, not only at his unthinking blunder, but at his completely missing the point.

My original fame as a RAD fixer was totally undeserved. But since I often felt that some of my accomplishments were overlooked, I guessed it was fair enough. My first experience occurred in the first week after I had returned from SDS training school in California. I had been assigned to assist an older engineer who maintained the biggest and most important

site in our area. The customer was very important and always looked at new engineers with suspicion.

Late one afternoon, the main RAD stopped functioning. It was spinning, but no data could be transferred between it and memory. The whole system stopped. I was alone in the room provided for the support staff because everyone else had gone home. The director of the center came bustling in looking for the regular engineer. When he discovered that he was gone, he immediately called my boss to complain. I was on the phone trying to call also, but the director's call was put through first. When I got through at last, my boss was in a panic.

"What's wrong?" he demanded.

"I haven't had time to look yet," was the only answer I had.

"Try to look like you know what you're doing and I'll get someone over there," he offered. That did a lot for my confidence. To top that, when I got near the RAD, the situation got worse. The director stood directly in my path with his arms folded across his chest and a stern unrelenting glare on his face.

"Are you trained on this equipment?" he barked.

"Yes," a simple yet true reply.

"Have you ever fixed one?"

"In school." He wasn't thrilled, but let me move closer to the RAD.
"I suppose you want to take the system down," he whined.

"Let me look at the RAD first," I answered rather tersely; he was beginning to annoy me. I opened the front door of the RAD and stared at the spinning metal wheel. It looked fine, just as I expected it to. I then walked around to the rear of the RAD, mainly to get away from the group of customers that was growing by the minute as more and more people discovered they could not use the computer. It didn't take a genius to calculate the amount of money being lost as the minutes ticked by. I opened the backdoor, looked around expecting to find nothing amiss. Then my heart grew light and I must have broken out in my biggest smile because right before my very eyes was the data cable dangling in mid-air. It must have been loose for weeks

and finally the vibration from the spinning RAD had caused it to fall out of its socket. I gently grasped it and had barely inserted it back in place when I heard the operator yell, "It's working!"

I emerged from behind the RAD to a heroes welcome. "What did you do?" asked the now beaming director.

"Minor adjustment." It wasn't really a lie. I didn't want the regular engineer to get into trouble, since the cable should never fall out. After calling my boss to call off the cavalry, I walked through the computer center with a new feeling of belonging.

Two weeks later, at the customer's request, I was appointed "site engineer," a sort of promotion with no money attached, but a lot of responsibility. This was the site where I was to meet my wife, so that bit of good fortune continues to have a beneficial impact on my life.

Up till now I have been describing the massive physical size of these devices, neglecting to mention their storage capacity. Each RAD could contain 5.6 megabytes. That is not a type-o, there is a decimal point between the 5 and the 6. The ten cabinets, which stood in two rows, each fifteen feet long and consumed enough electricity to supply a small village, had as much capacity as you can now hold in the palm of your hand.

This quickly changed when we installed our first set of removable disk drives. Each cabinet, which was eight foot tall and three feet wide, contained two drives. Each drive could accommodate a removable unit consisting of a stack of eleven platters (twenty surfaces, the outer surfaces not being used). The disks themselves came in what looked like and was called a cake tin. It was a round plastic cover about one foot high, a diameter of two and a half feet wide, with a rotating handle on the top and a locking plate on the bottom, just like the tin your grandma keeps her angel food cake in. Each disk "pack" had the capacity of 25 megabytes, so one cabinet had almost the capacity of the ten existing cabinets, and since the packs were removable and replaceable, a site had unlimited random-access storage. These units were not as fast as RADs because they had movable heads where the RADs had fixed heads providing faster access to each track. The drives themselves slid out from the cabinet on a drawer and looked like a black wash tub on the inside. While holding the disk pack in the air, one would

remove the bottom from the cake tin, place the pack into the tub, then twist it to the right until it was secure, which would automatically release the lid of the cake tin. Once the lid was removed, one could slide the drawer back into the cabinet and push the button to start the drive spinning. More than one person has tried to do this without removing the cake tin, fortunately the engineers foresaw this by incorporating a safety switch to prevent the drive from spinning. But, of course, at least once, that I know of, the switch failed and the only thing that saved the drive was the operator's quick reaction to the vibration and rattling sounds of the plastic cover rattling against the side of the drive. If the heads had attempted to load, we would have had a real mess on our hands.

Speaking of quick reactions brings to mind the time I took my three-year-old son into the computer center. I was on vacation and was vain enough to believe the place wouldn't run without me. So, one evening, I took a ride with my son. I asked if he would like to "see where Daddy works?" and, of course, he said, "yeah." Therefore my excuse was already made for me. When we arrived my replacement was the only one in the computer center. I introduced my son to the other engineer, then started chatting to find out how things were going in my absence. As any parent knows, kids can vanish in the wink of an eye. One moment my son was watching the flashing lights on the computer and the next he was gone. As soon as I noticed he wasn't next to me I called his name and raced toward the rear of the computer center. There were many things there that could hurt him. The disk pack drives being the newest devices were all the way in the last row of equipment. That is where I found my son watching the "ready" lights on the disks go out. There was only a certain set of switches in the whole computer center a three-year-old could reach, and he had managed to find these switches in no time and push the off button. I shouted to the other engineer, "Put the machine in idle," something one could do then, but not now. Then hurriedly I pushed the start buttons on each drive, which brought the disk units back up to operating speed in 30 seconds. It wouldn't have been a disaster, but it could have caused an error to be logged in the computer's records, causing me to have to explain what happened. My son thought it was all good fun, of course, but as I was the real culprit, I didn't scold him for it. Rather, I never took him into a working computer center again.

I was also fortunate that one of the disks did not exhibit one of its weaknesses at that moment. These particular drives had a

few idiosyncrasies of their own, some of which were caused by the hydraulic actuator that was used to control head movement. The actuator had a bad habit of seeping, not leaking mind you, but seeping; there was a big difference. If the actuator was leaking, an engineer would have to replace it, but seeping meant normally we had only to clean up the pool of hydraulic fluid that had accumulated in the bottom of the drawer and refill the fluid reservoir of the actuator. Some of these seeps got pretty bad, but we were told that since they were new expensive drives, they were not leaks. A side effect of the fluid loss caused us much grief. As long as the disk kept spinning everything was fine. But because the fluid tended to flow down below the disk, eventually finding the drive belt, which ran from a hefty electric motor to the base of the disk spindle, when the seeping victim stopped spinning there was a good chance the belt would slip off upon reactivation. Nothing short of cleaning both the spindle and motor pulley completely before replacing the belt with a new one would solve the situation.

When the problem first surfaced, we tried all types of solutions, but eventually decided the above procedure was the only cure. To gain access to the belt we had to remove a plastic cover from the bottom of the drive. We soon learned it was faster to leave the cover off than remove and replace it each time the belt came off. In time one could find most of the covers in a cabinet somewhere. One of our customers discovered this and demanded that we replace all the covers. So, we did. However, one of them had gone missing. Naturally, we stole one from another site, a process that could have continued from site to site. Maybe that's how the one at this site had vanished.

There are many more stories concerning mass storage, but I will save some for later.

--> True or False?

-- Disk heads fly.
-- Heads are all movable.
-- Elephants fly.

Holes in Your Data?

Your Data Were Holes!

So far I've talked about data from bits to megabytes, but have never mentioned where it all came from and how it got into the computer system in the first place. The answers are: it came from everywhere and was input to the computer in many ways, which, of course, is still not an acceptable answer.

When the first commercial computers were introduced, there was basically no computerized data, so the people who implemented them needed to find a source they could use. Since the applications were for business, they found a ready-made source in the tabulating room. There were machines in that department that already existed for punching, reading, sorting and printing accounting data. This equipment, much of it made by IBM, used a system of cards. The cards were made of very thin card stock and came in a variety of colors.

To say that the punched card came before the computer is an understatement. The first use of them is credited to a Frenchman named Jacquard, who used them to control textile looms in the early 1800's. The use of punched cards were adopted by the English inventor Charles Babbage in the 1830's to feed instructions to his analytical engine, the uncompleted forerunner of the computer of the mid-twentieth century. The code on the TAB cards was called Hollerith, named for the man who developed it, and various machines, for the US Census Bureau to manipulate the 1890's census information. His machines were so popular, even in Europe, that he founded the Tabulating Machine Company. Through subsequent mergers this company grew into IBM.

The holes themselves were rectangles 1/8th inch high and 1/16th inch wide. The card (7 3/8" by 3 1/4") was divided into 80 columns by 12 rows. Each column could contain one punch to represent numbers, two punches to represent letters, or three or more punches to represent symbols. That, of course, is what IBM used, but not Univac. Univac used the same physical size card, but divided it into 90 columns of six rows with round holes. The card looked like 45 columns of twelve rows; however, the top six rows were columns 1-45 and the bottom ones 46-90. Why would they do this? Univac said the round hole was bigger and better. I think it was the fact that Univac never did anything the same way as IBM. Univac held onto the belief that

they would eventually prove their point and then over take IBM as the number one company. Sadly for them, and their loyal customers, it never happened. The customers later had to face the task of converting their information to 80 column cards when Univac finally abandoned the 90-column card.

Now, if you have been paying attention and have a little mathematical skill, you may wonder how twelve rows of punched holes become eight bits of data in a computer. Easy, you obviously convert them by means of software. You spend a lot of time (computerwise) just changing each Hollerith code into a byte of data and then when you wish to punch computer data out onto cards the computer does the same conversion in reverse. Clearly not the most efficient method, but it was more expedient, especially for IBM, to use an existing technology providing customers an easy path from TAB equipment to computers.

So, how did this process work? A great deal of the data originally came from a device called a keypunch, an electromechanical device, which in the sixties was normally operated by a female with some level of typing skill. Most of the keypunch machines wore IBM name tags, were gray in color, desk height and width, containing a keyboard on the desk top and provided the only place to sit in many computer rooms. Each computer room had at least one. To the operator's upper right was the input hopper, where a few hundred rectangular cards were placed in a stack. The operator would type in the data, just as she would type a letter, using tabs and margins, except each card contained only 80 characters. When she hit the carriage return key, a card was fed through the machine and holes were punched into it. (Earlier machines did not have a buffer to contain the data before it was punched, so a type-o meant a mistake and a mistake meant ejecting that card and starting again on a new one. Not very expensive, but not efficient either.)

The resultant punched card was only a piece of cardboard with holes in it. The cards then had to be read through a card "reader" that was attached to a computer. Card readers were fairly big noisy devices that ranged from three-foot square boxes to six by five-foot card processors. These, however, were not tall devices like others I have described elsewhere. The card readers had to be low enough for an average person to load and unload cards into the hoppers. The speeds of these devices ranged from 100 to 1000 cards per minute, which may sound fast,

but if you figure each card might contain less than 80 bytes of data, it would take between two to twelve minutes to read one megabyte of data. Since we were only dealing with kilobytes in the 1960's, it wasn't so bad. What was bad was a card jam. If you are a speed freak, you can calculate how fast a card would be traveling at these speeds. Certainly not warp-9, but for a piece of paper, pretty fast. So a jam was not unheard of. The amazing thing was that jams did not occur more frequently. The path the cards had to follow was never perfectly straight. Often, in an attempt to make the readers more compact, the card path was down right convoluted. Try to imagine a paper back book cover with 200 small holes punched into it traveling in one direction and then being batted on its top edge so that it would make an instantaneous ninety degree turn. Then one inch behind it another piece of cardboard making the same journey with another and another for hours each day. The reader worked 99.9% of the time, but that .1% was brutal. Cards would crumple, tear, crease, and, of course the well remembered, fold, spindle and mutilate themselves.

Now what? First, the operator would clear the jam, hoping to do it without need of mechanical assistance and without damaging the machine (both of which happened fairly often). Then he needed to replace the damaged cards. This was done by finding the master deck, borrowing the necessary cards, taking them to a keypunch to make duplicates (keypunches had a copying facility), putting the new cards through the reader, continuing the original job and, naturally, returning the master cards to their deck. The masters never, I repeat never, went into the reader. Why? because it was not uncommon for the reader to jam again as soon as the operator started it up again. And, if the operator lost the master cards, he might lose his job as well. What if there was no master deck? Then he was in deep yogurt. I've seen operators or programmers trying to iron a crinkled card and pray that it would pass through the reader just once so that they could punch a new deck. Did it work? Do elephants fly? (Do you remember your answer to this previous question?)

If you can imagine how a series of card jams can affect an operator, you can imagine how I felt when called upon to fix the card reader at this point! I was not looked upon as a savior, rather the guy who's fault it was that the machine was misbehaving in the first place. Once I remember being called to a site for just such a problem. By pure chance I was only two blocks away when I got the call. I arrived almost as the manager set the phone down. He was not impressed, just angrily

told me that I was holding up the payroll and the steelworkers in the plant were expecting to be paid within an hour. When I attempted to soothe him by saying "you haven't had any problems in a month," he pointed to several huge, well-muscled individuals outside the door and declared, "Tell them that." One look at the steelworkers' wives spurred me on.

When, you ask, am I going to tell you about how cards were read? How about now. The cards normally moved in a horizontal direction with column one in front. That may sound obvious, but there were other methods; I want to keep this simple for your sake as well as mine. The leading edge was detected by a very small micro-switch, or in later years a photocell, like the ones used in auto-opening doors, but much smaller. The read head was turned on for the period of time it took one column to pass over the head. The card was passing between the head and twelve wire brushes, each brush had a wire connected to it that in turn was connected to a data buffer (a small section of memory that would collect the information). If a particular row had a hole punched into it, the buffer would record a one, otherwise it would record a zero. All twelve rows were sensed at the same time. The head was turned on and off 80 times as the card passed through the read area, differentiating possible columns of zeroes from the blank space between the columns. When the micro-switch detected the end of the card (trailing edge), the reader signaled that the card had been successfully read and the data in the buffer could be transferred to computer memory. It was necessary to wait for the whole card to be read because the card could jam and be damaged, in which case the data in the buffer was disregarded so that the replacement card could be read in it's entirety.

The use of switches and wires was very unstable and prone to errors. The contact between switches and brushes with the cards was very impractical. A deck of cards that was used frequently would develop tracks on it from the brushes and little nicks where the switches contacted the leading edge. The tolerance between the two and the card path was critical -- too tight would cause jams and too loose would miss the holes. Also, the brushes would become frayed and misread. The switches had tiny metal arms that would develop a fault called "oil canning." The switch would buckle rather than lever, once this happened. The only solution was a new switch, which was easy to install, if you had one.

On one occasion I was working with a rookie engineer and

discovered such a problem. The trouble was we didn't have a spare and it was going to take at least an hour to have one brought to the site. I fooled around with the switch and found that if I turned it around, then adjust it in the wrong direction, I could get it to work. Since the customer was naturally in dire straits, I did just that until the new part arrived. The young engineer was very impressed, so much so, that a few weeks later when I was faced with a different problem without parts he suggested, "why not put it in backwards as you did before?" I'm afraid that was a one-off solution; not many problems can be resolved that way.

When the photocell was used to replace the switches and wires, card readers became much more reliable, but still not perfect. Now back to the data flow...

Once the card was read into the computer, it was finally digital data and considered captured. The process of collecting data was therefore called "data capture." (Sounds exciting, eh? No, not unless you're a band of Romulans chasing the Starship Enterprise.) How long the data remained captured was a matter of the equipment available on the particular computer. If the machine had mass storage devices and/or magnetic tape drives (another chapter) the data was captured permanently.

But some early systems had no such luxuries. They were card-only systems, so the data was digital only for the time the program ran. This also meant that the software was on cards and every time one wanted to run a particular routine, he would have to load the entire thing from cards, often several thousand. With reader speed measured in cards per minute, this would take several minutes just to load.

The most well known software of this era was RPG, Report Program Generator (not Rocket Propelled Grenade). Although after I describe a normal series of events, you will see why some programmer/operators would have liked the latter. RPG was used for all types of reports including payroll. So, if you are over 40, there is a good chance that RPG created one of more of your paychecks. The process went like this. The operator loaded the RPG deck into the reader in order to read in the software. Depending on the speed of the reader, this took several minutes. Eventually all the RPG cards were read being followed by cards containing sets of data such as personnel record on the first card, weekly/monthly hours worked on the second card, etc. The reader now stopped and the computer churned for a few

seconds before printing: a check, a copy for finance and a new punched card (or set of cards) updating your personnel record. The card reader would kick-in for another set of data and the next check with auxiliary output would be created. For a finale, the computer would probably create another report for the boss, thus the name Report Program Generator.

That was the way it was supposed to work. However, if one card had a mistake, or was in the wrong place, the whole process could stop and have to be started over. This was not the worst thing that could happen, that would be dropping a deck, or a box of cards. Since the order of the cards was critical, the order had to be restored exactly. Operators loaded and unloaded cards all day long, so the occasional slip was inevitable (anyone for 2,000 card pickup?).

If you were very lucky, the site you were on had an interpretive keypunch, which meant that running an already punched deck through the machine would cause it to print what it read across the top of the card in English. This type of keypunch was expensive, so not all sites had one of these. But the sight of a bewildered operator trying to sort through a jumble of non-sequenced cards was both pathetic and unfortunately common. If all the cards in the deck had been punched by a computer, there was a chance that columns 72-80 contained a sequence number, so reshuffling the deck was easier, provided one could read the holes. Some old-timers were incredible being able to read a punched card as fast as I could read a book. But of course, there was also another expensive machine that could sort the cards back into the preferred order, but this machine might not be near the computer room.

There was a real knack to handling cards; one could tell a real pro from an amateur at a glance. One of the first indications was how he would "fan" the cards. This entailed grabbing a large handful in one hand and vigorously rippling the cards across your other hand. This was a very necessary exercise because new or old cards had a habit of sticking together and, if they did, the result would surely be a card jam. A good operator who fanned all of his card decks before feeding them into the reader could be an engineer's best friend, and conversely, one who didn't wasn't. Many a controversy would arise when a field engineer would blame card jams on the operators. The service people always had their own card decks for testing the machine and it was uncanny how often the test cards would fly through the machine while the customer's cards

would jam after just a few cycles. A typical scenario would be:

- The customer puts in a service call.

- The engineer arrives, then runs his "test" deck and pronounces that nothing is wrong.

- The customer tries his card decks and the cards jam.

- The engineer says, "It must be your cards, my test deck runs okay."

- "I don't care! Running your test deck doesn't pay the bills!!"

This was where a little PR was required, which not all Field Engineers were good at. It took me a while to learn. After a few years Xerox woke up to this problem and we all went to "charm school," where we learned, "The customer may not always be right, but he is the customer."

I eventually learned that if one treated the customer with respect, and that they knew he was doing his best to solve their problem, they would do almost anything to help him, even lie. A potential new customer came to one of my sites to see a piece of equipment in operation. When he and the salesman arrived, the machine was out of order. I was feverishly trying to repair the fault while the salesman bugged me about it. The manager of the site came in and invited the prospect into his office for a cup of coffee. The last thing I heard him say was, "Bad timing -- first time it's been down in six months," which was a boldface lie. Half an hour later I had it fixed and went to find the prospect and salesman. They were gone! My heart sank and I knew I would hear about it, even though it was through no fault of mine. The manager saw me and smiled, "They decided that they didn't need a demonstration after I told them how well it works." After I thanked him profusely, he added, "Don't worry, you owe me one now, but keep that salesman outta here." The salesman got the order and I kept him away from that particular site.

A site I wouldn't take anybody to was the railroad. They had only one piece of our equipment in a huge computer room full of IBM gear. The piece of equipment they did have was probably the most terrifying thing I had ever worked on -- the Univac 1001 card processor. It was a stand-alone ultra high-speed card

reader, standing seven feet wide, four feet deep with extended input trays almost five feet high. Notice the word "trays." It had two, one on the left side and one on the right, with a capacity of 3,700 cards each. They both had the capability of reading 1,000 cards per minute and they could run simultaneously. It also had seven output stackers, which could hold 1,500 cards a piece, each read station had three and the seventh, located in the middle was common to them both. In stand-alone mode, the reader could sort and collate under control of a "programmable plug board." What is that you ask? Patience I say, it's in a later chapter.

Try to imagine what this thing did. The cards were loaded by the box into the input trays, which were on an angle from the reader. The right one is angling outside of the reader cabinet and the left toward the center of the machine. The cards would slide bottom edge first into the read station and then travel front edge first through the read head at 1,000 per minute toward the back of the machine where they were stood on end and then slid via belts toward the center of the machine top-edge first. If the card was intended for one of the first three stackers, a paddle would deflect it into the proper stack. But, if it was intended for the center stacker, it just went there because of a ramp that was common to both sides. The control circuitry of the reader prevented two cards from colliding in the center pocket, supposedly. Each pocket or stacker had a large weight mounted on rollers that stopped the cards, yet slowly rolled backwards as more cards filled the stacker. Usually, the machine would run out of cards before a stacker became full, allowing the operator to empty the stackers. It was impossible to empty them when the unit was running. But one of the features of the 1001 was continuous operation. The operator could continue to load cards into the reader. A photosensitive cell at the front of the machine would stop the process when any of the stackers filled forcing the operator to empty it before continuing. It was this function that caused what now seems a series of ludicrous changes to the machine. Basically, the machine worked fine unless the center pocket received a lot of cards. Because cards entered it from both directions, the stacking was uneven and jams would occur when it was nearly full. The first correction for the problem was to change the metal weight, which rested on top of the stack to help keep a steady downward pressure on the cards, for a new one with a small flag protruding from it's rear, the object being to stop the process sooner. It didn't fix the problem, so about once a month we would receive a new longer flag until the

paper weight looked like it had a surfboard attached to it. The problem persisted, so the next fix was to unlevel the machine. All computer equipment comes with leveling feet and customarily it is important to keep them level, especially card equipment. However, we were instructed first to lower both front feet and then raise both rear feet. I faithfully performed all of these changes until the alterations started to cause other problems. The reason for changing the angle was to cause the weight at the top of the stack to roll easier in order to eliminate the jams. I found that it caused the weight to roll too fast causing jams because the cards vibrated into a loose stack. The funny part was that the machine I supported had never shown the problem in the first place.

I called the engineering center to report my situation. After a long conversation I found that only one site was experiencing the problem--the fuss was all about them. So, I releveled the machine and ignored any further changes to the machine unless I encountered the stated problem.

One incident that remains clearly in mind had nothing to do with the equipment I maintained but was typical of the paranoia caused by computers back then. I received a frantic call from my boss asking if the 1001 at the railroad was involved in printing their paychecks. This puzzled me since all it did was read cards and not print. But the cause for his concern soon became clear. The morning newspaper's headline read "Banks refuse railroad paychecks." Upon further reading the article stated that on some of the checks the dollar amount was not inside the box where it belonged and some hotshot bank manager had instructed his tellers not to accept them. So, first the workers were furious because they couldn't get their money, secondly the managers of the railroad were looking for someone to blame. Then, the people at IBM tried as well to shift the responsibility anywhere they could, which, of course, my boss wanted to make sure wasn't to us. In result, I was dispatched to the railroad to assure that the problem wasn't ours.

In the end it was one of the railroads operators who was blamed for failing to properly align the preprinted forms onto the printer. I knew our reader wasn't the culprit and was finally told it wasn't even used in the payroll process. The two hours of overtime I was paid was gratefully accepted.

If you think card readers sound kind of clunky, you should have seen card punches. They were the computer version of the

vegematic, really great at slicing and dicing; once again, a lot of fun to work on. I was very familiar with two of these babies. The first was called a serial cardpunch, because it punched the holes from column 1 to column 80 two columns at a time. The other was the parallel cardpunch; it punched the entire card all at once. Sounds impressive, eh?

I spent most of my time on the serial punch. Because it was smaller and cheaper than the parallel punch, there were more of them. The serial punch was fairly easy to work on because the punch mechanism was removable; thus one could take it to a workroom. The early versions needed this advantage because they required frequent maintenance. The reason for this, once again, was little switches -- lots of them.

A requirement of any computer device is accuracy and dependable data. Therefore safety/security devices were built into every device. I have mentioned parity checks to keep memory accurate. Before I'm finished, you'll hear about checksums and redundancy checks. But, for now, let's worry about holes. How does one guarantee that the proper holes were punched in a card? In the case of the serial punch, with switches, there were 24 of them. The punch head was designed to punch upwards. The card passed through a thin gap between the bottom and the top of the punch. The bottom contained two columns of sharp steel dies, each the size of the hole in a computer card. As the card passed through the head, it stopped every two columns. At that precise time, a cam in the bottom rotated to its highest position. For each die there was a small rod that could be inserted between the cam and the die. If the rod was inserted, a hole was punched. If it wasn't, a hole wasn't punched. But, how did one know? Each die had a notch in it, with the arm of a small switch inserted into it. As the die went up, the switch was activated causing the control device to recognize that the hole was punched. Of course, one still didn't know for sure. Now, of course, this all depended upon these little switches, which traveled up and down thousands of times daily. It was generally thought that the overtime pay generated by these little guys would buy a field engineer a new car every few years. The punches worked great, but the switches failed all the time, having to be adjusted or repaired continually.

I will never forget Halloween 1967. My first wife and I were having a party. I had just donned my Man from Outer Space costume when the phone rang. It was the dispatcher asking if I could take a service call on a cardpunch. I wasn't on-duty, but

the customer had requested me personally. My wife glared at me and said, "No!" So, that's what I said. A few minutes later the phone rang again. It was the customer; he knew my number because I had given it to him, a habit I soon broke. He begged me to come in because it was imperative, naturally. He was a very nice fellow and, as chance would have it, I had never refused a call before. The site was close to my house, so I promised my wife it would take only an hour. Off I went in my costume to fix the punch. My strange attire drew a stare from the security guard, who seemed disappointed when he was told to send me right to the computer room. For once, my prediction was correct, returning home within the hour with two magnums of champagne and a large box of pretzels as a gift from the grateful customer. Coincidentally, I was at the same site fixing the same punch a few weeks later when I received the call that my first son had been born, which was also good for a couple bottles of bubbly.

Fortunately for the customers, not my bank account, the switches in the punches were soon replaced by photocells reducing the frequency of problems. So I drove the same car for a few more years....

Punched cards were not the only medium using holes to represent data. My second wife's favorite device for that was paper tape, the cheapest computer component you can imagine. Paper tape units were originally used in process control operations, which go back to Jacquard in 1801. However, the statement concerning my wife is very facetious, she shudders every time I mention them. The tape itself was about one inch wide with sprocket holes running down the middle of the tape, slightly off-center. Depending upon the codes used by the reader, there was room for either six or seven round holes within that inch, plus sprocket hole, that were sensed by either wires or photocells, once again. The speed, if you could call it that, was very slow. My wife claims she could type faster than the paper tape could read/punch. A few large readers existed with take-up spools and a series of arms that were necessary to regulate the slack as the tape moved over the read/punch head. But most of the devices were very small and were packaged with keyboard devices called ASRs, Asynchronous Send Receive units. ASRs and the related KSRs were the first terminal/operator consoles, a further example of devices not designed for computers, in actuality they were used by the Teletype Corporation for two-way communications. These noisy mechanical monsters enabled dialog between human beings and computers. KSRs (Keyboard Send

Receive) did not have paper tape units, therefore they were even cheaper. I will discuss them in more detail in another chapter. Back to paper tape. The advantage of paper tape was cost. The medium was paper, so it was inexpensive. The equipment was widely and cheaply available. This inevitably leads to a few disadvantages, most obviously paper tears. But the real problem was that of correction. There was no way to correct a mispunched character, that is why my wife shudders. As a young intern, she was given the task of typing, or keying, a large amount of data onto paper tape, a total nightmare. She sat in front of an ASR pounding the keys from a hand written sheet. Things were okay until she hit a wrong key. At that point, the tape she was creating was worthless. It might be ten or fifteen feet long, but it couldn't be used. So, she had to perform the following act of reading the tape into the computer, stopping at the point of the error, erasing the type-o and then punch a new tape. She would then resume typing, trying to avoid the trail of paper tape wrapping itself around her swivel chair for the next inevitable accident. Eventually she created a full tape that could then be used as input for another device at a different location.

My own memories of paper tape are not as frightening, but still not pleasant. XDS, which is what became of SDS, won a contract to install computers to control one state's power and light company. Several computers were installed throughout the state, each controlling a local grid. These consisted of a CPU, my old friend a RAD and ASR -- nothing else! The reason for this bare-bones configuration was that the computers just talked to one another and the grid control circuitry. Not much human interface here, until something went wrong. Then the only way to talk to the system was through the paper tape unit. Each site had a box full of paper tapes to be used for diagnostics and system restoration programs. Diagnostics are programs expressly created to identify or cause failures. (More on them later.) One of our engineers was sent to one of these installations to fix a problem. He was several miles from nowhere as the hour grew late. He had not made much progress in isolating the cause of the trouble and knew that he wouldn't be relieved for many hours. Around 7 p.m. he called the answering service to inform them that the diagnostic tape had torn and he was unable to continue his trouble-shooting, thus he was going home. The next morning, a fresh serviceman arrived, who quickly isolated the problem and fixed it. A strong suspicion existed that the first man would never have found the solution. And how the tape was torn was a mystery. If the Power Company had

provided the field engineer with the new mylar diagnostic tapes, the doubts would have not existed because mylar tapes do not tear.

The demise of paper tape and punched cards had the following major repercussions throughout the computer industry.

- No more free confetti for office parties and parades.

- Loss of a ready source of pocketsize notepaper. (Every engineer and programmer I have ever known had a few computer cards in his shirt pocket full of notes or instructions.)

- No materials for creative handicraft projects or gift-wrapping ribbons personally inscribed in the holes of the tape.

- And the nice strong boxes that were great for storing all kinds of things, even punched cards.

After this long chapter, it must be time for a quiz.

--> How many columns in a 90 column card?

--> How many rows in an 80 column card?

--> What shape are the holes in a 90 column card???

From BOT to EOT

Had enough paper technology? So did the early computer people, even though card only systems would survive for many years a better way to transport data was needed. And was supplied by means of magnetic tape.

Everyone who has seen an early sci-fi movie will remember seeing a china cabinet sized device with two large reels oscillating back and forth. The films gave the illusion that this was the computer and it was making massive calculations that would save or destroy the earth. In reality it was at best a high-speed input/output device and was more than likely performing a sort operation, which was a popular application back then. More on that later.

The first tapes were actually made of metal and the reels were very heavy. I would imagine that the operators had to be pretty strong. An 800-ft reel weighed about ten pounds. Metal tapes were soon replaced by oxide coated plastic just like the tapes in your VCR. It was lighter cheaper and enabled much more tape on a reel. It soon became the standard, about the only place you can find metal tapes now is in my mother-in-law's attic.

Whatever the media was, tape was a giant step forward. Data could be recorded at from 125 to 1600 bytes per inch and be written or read at from 37 to 150 inches per second. In later years these numbers would increase as data storage became denser thus faster to retrieve.

The tapes were one half inch wide and came in lengths from 50 to 2400 feet and weighed about a pound. There were two types: seven-track and nine-track. By track I mean how the data was arranged across the tape width. Each bit was controlled by it's own read/write head thus track. The reason for the odd number was parity, each byte had its own parity bit. Octal systems used seven track (6 bits per byte plus one) and Hex systems nine track (8 bits per byte plus one). So just as Hex became the standard so did nine-track tape. The bytes were written in blocks usually from eighty bytes, the same amount held by a punched card, to four thousand bytes per block. To make efficient use of tape drive speed, the bigger the block, the faster the tape would travel. So a program that needed big amounts of information for each computation might place one group of data per block, whereas another that required only small amounts might bundle many of these into a block. In any

case, an apparently short piece of tape could contain a lot of data (for the sixties).

The write/read operation was accomplished by moving the tape across a set of write/read heads as the tape moved the selected heads would receive a pulse that would create a magnetic image on the tape. During a read operation the head would sense the image and transfer the information to the tape unit's memory. The read head was positioned directly behind the write head so that the data was always read immediately after it was written providing an automatic validity check. If the data that was read did not match what was supposed to be written the tape unit would automatically backup the tape and try writing it again until it got it right, or until it exceeded a pre-set limit of retries, normally ten. If this happened the job would be aborted which could be a very distressing event.

Some systems had the ability to set a "Bad Tape Mark" which caused the unit to space ahead a certain distance before resuming the write operation. In some cases this was undesirable and it was always a matter of the job's importance or length as to how errors were handled.

Most errors were caused by bad or damaged tape. There were many ways for this to happen. The tape was actually dragged across the heads so after a while the oxide could be worn off and the drive would have difficulty recording on it and possibly even a worse time reading it. The oxide that was scraped off would collect on the heads making them dirty and more abrasive. So the dedication of the operator to cleaning the heads was crucial to the performance of a tape system.
Mechanical failure could also contribute to the cause of problems. Mechanical faults could stretch or crinkle tapes making them unusable, and the every day handling also contributed to the general deterioration of a tape.

To get a better understanding for the problems of magnetic tape handling I will describe how a typical unit worked. Think of a cassette. There is a reel of tape on one side and a take-up reel on the other with a place where the tape is exposed to the recording/playing heads. On computer tape drives the reel of tape is loaded onto one hub with the take-up reel fixed on the other. This means the tape has to be rewound in order to remove the tape. Somewhere between the two reels is the housing for the read/write heads and typically below each reel is a long vertical chamber with vacuum pressure. Now the read/write

operation was simple in its self but getting the tape to the heads and maintaining a constant tape speed was a different matter. The outside diameter of the two opposing reels changed constantly during operation so the speed of the hubs had to be regulated to compensate for the fluctuating circumferences. This was done by means of the two vacuum chambers holding the input to and output from the heads. The vacuum provided a constant pressure without damaging the tape as pulleys or rollers might. Each chamber would hold enough tape to form a "U" shaped loop, which would shrink or enlarge as the hubs rotated.

The drive I will describe had two vertical chambers about three feet high and six inches wide. I have avoided describing the actual tape path because each drive was different. And the operators would tell you they were all terrible. Loading a tape was sometimes a traumatic experience. I will use an example from my early days at SDS. The drive was about six feet tall and three feet wide. It had a row of button/indicators across the top of the cabinet. These would show what state the drive was in at a particular moment.

The first button was power, pushing it would turn the drive on or off which was a lot more complicated than just illuminating a lamp. Blowers, compressors and all sorts of power supplies had to be started. This process could take as long as a minute or two, so drives were normally left on all day.

The tape hubs were behind a glass window, which could be lowered by pushing the "LOAD" button. Now the fun part begins, the operator takes a reel of tape and removes either the ring from around the outside edge of the reel or takes the reel from a can like the ones movies come in. Some reels had a small piece of foam rubber holding the tape to keep it from unraveling and others had a small strip of plastic to perform the same function. Once the tape was loose the operator would jiggle a small amount free, unlatch the mounting hub on the tape drive by releasing a latch in its middle and mount the reel on the hub. He then pressed the latch back in place thus securing the reel in place. The hub contracted when released and expanded when latched, so that it now turned freely. The operators would take the end of the tape between their fingers and thread it through the path. First through the rollers of the input chamber one on each side, then through the set on each side of the read/write heads and then through the rollers on the output vacuum column. Now the tricky bit, the output hub

containing the permanently mounted reel had several open slots in it. The tape had to be lead onto the reel and then held in place with one finger while some of the slack was wound around the hub. If your finger slipped the tape would slide back from where it came and you would have to start over. This job called for long thin fingers. Once this was completed the "load" button could be pressed, causing the window to go up, the input hub to feed tape into the input chamber until the tape reached a sensor, which caused tape to be fed through the heads into the output chamber until it reached another sensor which then caused the output hub to turn, causing tape to move through the entire system until finally the BOT sensor detected the BOT marker and the whole thing came to a screeching halt. THE TAPE WAS LOADED!

Now you demand to know what is a BOT marker or for that matter what is a BOT. It's a "beginning of tape" and while we are at it EOT is "end of tape." Between the two they define the usable portion of the tape. Physically they are identical pieces of aluminum foil about an inch long and a quarter inch wide. The BOT runs along the outboard edge of the tape. It is placed a few yards in from the actual beginning of the tape to allow for a sufficient leader. The EOT is positioned on the inboard edge a few yards from the actual end of the tape to give it a trailer. There are sensors located near the read/write heads that detect either marker.

So far all we've covered is loading the tape, which was a very labor-intensive job. There were auto-load tape drives, which were designed to minimize the operator's task. These required the tapes to be housed in special rings that locked around the reels. All the operator need do was mount the reel on the input hub and push the load button. Vacuum and movable arms took care of the rest, you hoped. It worked most of the time and as long as it did, every thing was fine. Problems would occur when the leading end of the tape became crinkled or creased. This would happen when the tapes were handled manually. What could happen was rather comical. The tape would spin but the lead edge of the tape would miss the track and then try over again, after several attempts it would give up with what sounded like a deep sigh as the vacuum motor shut down. The solution to the problem was simple, cut off the end of the tape. It was very common to do this periodically and was no problem unless the distance from leading edge to BOT became too short. This would make the tape useless so care was taken to always have plenty of tape before the BOT.

Tape was a pretty reliable form of data storage and was normally secure. One of the important functions served by tape was "BACKUP" and I mean the capital letters. Large computer sites regard backups as a form of religion to be performed on schedule no matter what. Banks and insurance companies normally have separate locations where they keep duplicate sets of their backups in case of disaster.

My wife often tells the story of how she was very embarrassed the first time she was responsible to perform a back up when she was a part time operator. She had been told to type COPY on the command console in order to copy the contents of the disks on to tape and then type COMPARE to verify that the contents of the tape matched the disks. The copy part worked fine but when she typed "compare" it copied the tapes over again. She tried it three times with the same result. In desperation she called the operator who had instructed her on what to do. He verbally told her the same thing again with the same result. A call to the head programmer brought him reluctantly into the site; he then typed CMPR and watched as the tape was successfully verified. My future wife was speechless.

"That's not how I spell compare," she gasped.

"It's the way this program does," he answered.

It turns out that the program read only the first two letters: CO was copy and CM was compare. Since both words started with CO, some genius decided to omit the O in compare. It never dawned on this person to use a different word instead. But the computer industry is still full of such brilliant things (like two digit dates...).

So you think you know about tape drives. We haven't even looked at how the images on the tape look. And believe it or not you actually can. There's some stuff called "visamag", which is actually small iron filings. You sprinkle this stuff on to a piece of recorded magtape and the filings arrange themselves in the pattern of the magnetic image. You can see the tracks the gap between the blocks and if you're real good (and have a magnifying glass) the bits. Normally you would take a piece of scotch tape, press it on to the mag tape, remove it carefully taking the image with you and then press it onto a piece of paper or card stock. You now have a picture of what the data on your tape looks like. The next question is "WHY?" So you can measure it. Since you never knew where a tape might be read it

was important that all tapes were compatible. One of the most important areas was the "inter-record gap." This distance was established by the amount of tape transported during the time it took to start and stop the tape. This happened in between each block of data, and was called "start-stop time." Several adjustments and the cleanliness of the drive could effect the gap. And for some reason, I have never figured out, it was something that gave me no end of grief. Yes, after all my bragging about how good I was at fixing things, I must confess that tape drives and start-stop time were my Waterloo. Embarrassingly, I recall one particular event. A new site had four tape drives on our company's system and four on the IBM system. The customer obviously expected the tapes between both systems to be compatible, and as is always the case when you are working in the IBM environment, the burden of proof is on you. My task was to assure that our drives were perfect. I started out with a tape provided by the customer. I was getting faults on all four drives, which should have told me something, but my paranoia got the best of me. I adjusted, cleaned and adjusted again. The faults would vary but there was always something wrong at one point -- they wouldn't work at all. Finally, over my protests, the office sent me help; a very humiliating experience. When Jim arrived, he took one look at the test tape and said, "where did you get that from?" "From him," pointing toward the customer. Jim produced a tape from his bag. It worked fine. He then asked the customer for a new tape, which also worked. The customer then admitted that the tape was very old but he didn't want to risk a new one on strange tape drives. Jim was a hero without even lifting a screwdriver and I was further convinced that I hated tape drives to pieces. The funny thing is that, in later years, I became an expert on tapes, not fixing them but figuring out how to read the data from different manufacturers and getting it to print on a laser printer. But that's another story or maybe another book.

After all we've been through you would think I'd covered all the gnarly bits. But oh, how wrong you are. I have saved two of the worst for now. Because magnetic tapes are exposed to many outside contaminants and contain very important information it is necessary to take exceptional means to protect the integrity of the data. I've already mentioned that each byte has it's own parity bit but it didn't stop there. Each file had a parity byte, which was a sum composed of all the bytes in the file. When the tape was written this byte, which was called the longitudinal redundancy check byte (LRC), was calculated and

appended to the end of the file. While the tape was being read, another LRC was calculated. The second was then compared to the LRC on the tape. If the two were equal, everything was fine. If not, the tape drive would rewind to read the file over again until the LRCs matched, or, eventually, the unit would return a hard error.

Now that was pretty simple. Wasn't it? The real tricky bit was the CRC, cyclic redundancy check. This little guy, used to check data recording or retrieval accuracy for each block on the tape, was actually self-correcting. It could figure out what bit was wrong (if it was a single bit failure) and fix it. "How did it work?" you ask? Darned if I know. It was some mathematical hocus-pocus, which was not always used because it slowed things down and the last thing anybody wanted was to slow things down. But it always sounded great. We will encounter these terms again when I talk about data communications, so remember them. I won't dare try to explain them twice.

Of course there were good parts to fixing tape drives or at least my difficulty in doing so. They were the source of many of the problems my wife (then girl friend) needed fixed. So I was never unhappy to respond to a service call if I knew she was going to be there when I arrived. But I promise I wasn't distracted that much. It was just one of those things. I could fix RADS and Disks but tapes drove me nuts.

Without looking can you remember what LRC stands for?

-->Can you spell the words?

Hammers, Drums, Chains and Bars

Instruments from a medieval torture chamber? How about a musical motorcycle gang? No these were all terms used to describe types of line printers. Which in many cases produced the ultimate product of the whole computer process. A machine could add, subtract, multiply, calculate and compute. But if a person couldn't see the results what good was it?

Line printers were the most visible example of the computer's speed. 200 card a minute card readers and 100 card a minute card punches were noisy but not really fast. But 400 line a minute printers looked awesome (except nobody said "awesome" in the 60's).

The average line printer was not very attractive. It was normally less than five feet tall and four to six feet square, the shorter ones made great work surfaces, accumulating large amounts of stuff on them, which would have to be moved whenever you performed maintenance on them. Somewhere on either the front or the back was a door that allowed the operator to load boxes of continuous form fanfold paper. The sheets of paper were 14 inches wide by 12 inches high. To aid readability, every couple lines was either shaded with pale green or not, which is why everyone called it green bar. The page had pin feed holes running down both edges and was perforated at each page break. The paper was fed through sets of tractors containing round pointed teeth that meshed with the holes in the paper. There were normally two sets of tractors one below the print area and one above. The paper was fed up through the printer and was stacked either behind or in front. Mainly this was a simple process of each sheet folding itself on top of the preceding one, the only problem being that half the pages were face down making it difficult to read your output. The biggest sin someone could commit was to touch the output stack of paper while the printer was running. The results were often disastrous: jammed paper, shuffled pages, or missing reports. A well-run computer center did not allow anyone but the operator to touch the printer, but in some cases a user could gain access and remove his stuff. This was fine as long as all he removed was his own. But, too often they would accidentally remove someone else's and that would lead to all kinds of problems, like we didn't have enough already.

What did this stuff look like anyway? The normal page could consist of 66 horizontal lines of 132 characters each. The

fonts (character sets) used were spaced at 10 characters to the inch and 6 lines per inch. Normally spoken as "CPI LPI". This may sound fairly large for normal print, but since the quality of early printers was pretty poor, the attempt to make up for it was just make the smudges bigger. I will try to describe the print process so you may have a better understanding of why the quality was so poor. The first printers were drum printers, which means that the characters for that font were engraved on a steel drum, which was about fifteen inches wide with a one to two foot diameter. The drum was constantly spinning. The diameter of the drum depended on the size of the character set required. Most printers printed upper case characters only. This resulted in a smaller drum, which allowed the printer to print faster. Why? Because of the following. The drum had 132 columns engraved on it and each column had the complete character set engraved on it. In order to print a line the data for the line was loaded into a 132 character memory called a buffer (these were called buffered line printers) each character referred directly to a physical column on the drum and was electronically attached to a hammer which was positioned opposite the drum. When the character in the buffer was the same as the character on the drum the hammer was actuated. The hammer struck an inked ribbon, which was positioned between it and the paper and then the drum causing a character to be printed. When all 132 columns had been printed the paper would be advanced to the next line and the process repeated. Remember we were printing 400 lines a minute so it took 52,800 matches a minute to print a full page.

So why did less printable characters make a printer run faster? Answer: it took less time for the desired character to spin into place, thus the line could be printed sooner. Some printers were speeded up by having two sets of characters engraved so the same physical size drum would print twice as fast. The real truth of the matter was that few printers ever printed at their rated speed. This was due to the randomness of the data. To obtain maximum speed the data printed would have to match the characters on the drum exactly. To demonstrate the printer speed, test patterns were designed to accomplish this, very impressive but not very practical since what the customer needed was actual data not test patterns.

If that explanation wasn't clear, I have bad news for you. I omitted a few things, like the characters were not engraved in a straight line across the drum, if they were it would have been impossible to print a straight line. The time it took to

fire a hammer on the left edge was different than that on the right edge, this was called latency. To compensate for this the characters were skewed across the drum in a delicately increasing diagonal.

So now you know about paper and characters, what about the ink? It came from a huge ribbon, which was impregnated with the darkest, vilest ink known to man. If you don't believe me just ask anyone who ever changed one. This was the most hated task in the computer room. The ribbons came in two types: spools, which were a few inches high and ran from right to left or vice versa, and, even worse, the rolls which were the width of the printed line, and ran up and down. The wonderful thing about both types was that they were multi-pass ribbons meant to be used for a considerable length of time, days in fact. Normally they would be used until the image created started to fade. This could be a matter of interpretation. The person receiving the output would expect a dark clear image, so he would want the ribbon changed frequently. The operator wished to go home without ink-stained hands, clothes or ears, so he would let it go longer, especially if it was toward the end of their shift. Many a battle was started as to when this particular task needed to be performed. The ribbon was also changed for special jobs, like checks that used magnetic ink to print the special characters at the bottom called "micr." (Magnetic Ink Character Recognition) And then there were dreaded mechanical failures like ribbon skew. The ribbon would start out like a new carpet roll even on both sides, but if some roller or guide got out of alignment, it would start to roll just as when you try to take up an old carpet and no matter how hard you try, it gets uneven and bunches up on one side creating a catastrophic mess. Or the reversing switch would fail and the ribbon would remain at one end and the hammers would beat a hole in it. Because this was a fault, the engineer would have to fix the problem and then change the ribbon. Which is why I hated the job.

So what was so bad? It went something like this. The old ribbon would have to be removed, even though the edges which were normally not printed on still contained plenty of ink and, of course, that's the part the engineer would have to handle. The best way to remove the ribbon was to have it at one end of its pass. This allowed for easier dismounting of at least one end and the placing of the ribbon into a box. The ribbons could be sent back for re-inking so they needed to go into a box. Naturally, like anything else, no one except the people at a factory can put things into a box (the boxes must shrink after

the item is removed). Once this has been done, a new ribbon is secured into place.

Now if you were really good you accomplished this without touching any surface containing ink. A feat reserved for super heroes, mere mortal beings got some on their fingers and of course an itchy nose or ear or shirtsleeve would rub against some part of the machine when you weren't looking. However, a pair of plastic gloves came with each ribbon and they were perfect for allowing the ribbon to slip from your hands at the worst possible moment. Well, so now you know where ink comes from and where it goes.

I've mentioned hammers several times so you may wonder what they looked like and how they worked. They were not claw or ball peen, but square or rectangular matching the size of the characters to be printed. About a tenth of an inch wide and a sixth high. They were mounted across the print head with the ribbon and paper in between them and the drum. The hammer would strike the ribbon causing it to drive the paper up against the drum printing the character. No wonder the images weren't so good. Each hammer was controlled by an actuator/solenoid, which was controlled by the printer circuitry. Some hammers were part of a firing mechanism while others were independent and could be replaced on their own. The hammers very rarely failed but the actuators, which generated a fair amount of heat, did. And when you replaced an actuator it would have to be adjusted so that it fired in synch with the other 132 columns. One of the first printers I had seen was really hot. It contained 132 vacuum tubes, which you could have heated your house with. The tubes also glowed a bright blue and it was neat to turn out the lights and open the back of the printer that would light the room an eerie blue.

They were also known to explode and cause small fires. The hammers were the cause of the drum printers biggest weakness -- "wavy lines". Each actuator had a slightly different firing speed that could be adjusted. Adjusting 132 to match was no easy chore and, once it was right, it didn't last very long. If no one took care of this one would eventually get an almost unreadable line with the higher characters from one line almost touching the lower characters from the preceding one. Chain printers, which had their characters engraved on a chain or bar that traveled in a horizontal direction, didn't have this problem. Instead the character spacing was effected, causing the columns to be printed too close together or too far apart.

Bar and chain printers were advertised to have a major advantage over drum printers in that you could switch font sets by inserting a different bar or chain. This sounded good, but was rarely practiced. Computer operations managers frowned upon any activity that caused their equipment to stop for any reason. So even if some one desired such an innovative idea, like two different fonts or upper/lower case letters, they were soon discouraged, because any job that required operator intervention, habitually was the last to run.

Salesmen, of course, still used this advantage in their sales pitches. I worked on bar printers at Univac and can remember getting frustrated trying to line the bar up with the narrow groove it had to fit in so that I could slide the bar into place. Lots of cold cups of coffee reminded me of how long this would take. I was supporting a Univac exhibition at The Spring Joint Computer Conference in Atlantic City one year when a salesman asked me to demonstrate the ease of bar changing to a customer. Being still new in the business and my first experience working with sales people, that first request caught me off guard (I would learn to expect the unexpected from them). But I had no choice but to comply. I opened the printer cover, released the pin that held the bar in place and slid the bar out of the printer, allowing everyone to examine the bar and inspect the character set. I hesitated a few moments hoping they would forget about putting the bar back in so I could do it without an audience. No such luck, the customer wanted to witness the ease of operation. Reluctantly, I bent over and stared at what appeared to be the eye of a needle, which I was about to thread with a two by four. When the bar slid effortlessly into place I thought I'd missed the slot entirely but to my great relief and surprise the bar was in place and all I needed to do was clip it in position. Doing this I turned around to receive my congratulations only to find the receding backs of the salesman and client. I must have changed those bars a thousand times for cleaning purposes and never got one to go back in on the first try again.

The vast majority of the pages printed on line printers were program listings and reports, most of which were never read. The ecologically insensitive programmers of the 60's did not worry about the rain forests or solid waste. It was very common to generate several hundred pages of output, check one calculation, throw the paper away, change one line of code and run the whole mess again. Don't be too harsh on those old timers, they didn't have video screens or displays to look at;

it was the only way they had to check their work.

An area of constant problems on line printers was the output stacker. The pages would flow out of the printer and settle on to a tray or cage hopefully returning to the accordion like state in which they had emerged from the box. It normally did, if the pages started out correctly. The pages had two different types of fold or perforation, top edge or bottom edge. The first sheet on to the stacker needed to lay with the top edge up. So it was a fifty-fifty chance it wasn't and if it wasn't there was a good chance the page would not stack correctly and each succeeding page would have difficulty lying flat, eventually causing a problem. Normally the problem would be the operator's, so they would take care to make sure the stack started out right.

Another source of problems was the page eject. Since many pages of printout did not fill the page it was expedient to speed up the process by issuing a page eject command that caused the printer to rapidly advance the paper and position the top of the next page in front of the print head. If a series of short pages were received in a row it could cause the pages to stack incorrectly or even fall onto the floor. Both of these examples are for printers with no type of stacking mechanism, which was normal. Some printers came with stacking devices and there were machines available from outside vendors that performed post-printing operations such as stacking, trimming and binding. But by and large the stacking was left to gravity and a little luck.

Lack of luck contributed to one of my most embarrassing experiences. I met my wife when she was a part time operator at a computer center where I was the on-site engineer. It was an unusual romance since in the beginning we never saw each other. She worked on the weekends and I worked during the week, only working on weekends in an emergency. Every Monday when I returned to work there would be a little note sticking to some piece of equipment mentioning some problem or another. They were always minor or else I would have been called in. At first the notes were kind of formal and the only response I gave was fixing the problem. This went on for several months, without much happening, until one particular problem required some two-way communication. So I left her a note (I had always been aware that she was a young girl). I addressed it to the attractive female operator. On Monday her response was longer and less formal. Things progressed from there and finally we

met face to face and started seeing each other outside of the computer room. One weekend I showed up on my own wanting to take her to lunch. She was initially reluctant because the printer was running a long label job, which would run for hours and she had to watch the stacking. "It will be alright," I coaxed. "I can fix some cardboard sides to ensure it stays straight." My arguments finally convinced her and we went off for a sandwich. We were gone only an hour but when we returned there was paper from one end of the computer room to the other. The printer was still happily spewing more sheets onto the pile. The stack had stayed straight from left to right but when it had gotten too high it toppled forward and began crawling across the floor like some kind of paper creature. The purpose of the job was to create continuous sheets that could be fed into a cutting machine to create mailing labels. And the instructions had been to keep all the output in one piece. Stopping the printer we attempted to gently restack the pages without tearing them. This proved to be very difficult. At one point in time we thought we had it, then discovered the pages where one off and had to start over. My stock as a super engineer was badly damaged and my visits on the weekends terminated. Since we recently celebrated our 25th anniversary, I guess I recovered.

Another mechanical part of printers I failed to mention is the VFU (Vertical Format Unit). This device allows the user/programmer the ability to skip to a specific vertical position on the page or to the next page, a very useful and efficient thing to do. Skipping to a subtotal line or a new field for invoices or insurance forms is performed quickly.

The bad news is we're going to use paper tape and wire brushes to accomplish it. The paper tape is a narrow strip that is exactly as long as your page and for every line on the page there is a sprocket hole in the center of the strip. The strip is made into a loop and inserted on a small wheel which has sets of wire brushes arranged in parallel across the loop. Each wire brush represents a channel, normally seven in all. If a hole is punched in the loop at a particular channel at a particular line and then a skip command is issued referring to that channel, the printer will advance the page to the proper line and print at that point. Did you read that over again?

How did it do this? The wire brushes are prevented from resting on a metal drum by the paper loop when the hole is punched the wire can contact the drum allowing current to flow thus

applying the brakes on the paper tractors but only at that line. One can punch several holes in the tape at different lines and thus format his page for different jobs. Channel one is usually reserved for top of page and is set to line three or four, since the printing at the very top of the page could be difficult to see. The big problem with the wires was they would fray and either miss making contact or make contact at the wrong time. The paper would also wear out and making the loop was one of those jobs only a few were brave enough to try. Eventually the wires were replaced by photocells and the loops became mylar. Computer centers had special cabinets in which to keep the loops on tension held rollers. So all that was needed was to clean the photocells and align the paper properly.

So far this discussion has covered printers with 132 print positions and 66 lines per page. Those are not the only options available. There are printers with both shorter and longer print lines and of course the page length is really unlimited. But 132 X 66 was the standard dimension.

I eventually spent a lot of my career in Xerox's Printing Systems Division and was fortunate to participate in the growth of that sector of the business first hand. But the sight of a 500 line per minute drum printer will always be one of my lasting memories.

-->How's yours?

-->What does VFU stand for?

-->What color were the bars on normal paper?

Talking Back

In a previous chapter I told of how my future wife had typed commands on the operator's console to control a tape backup operation. Did I mention what she typed on? Well she typed on an ASR. Remember that from the holes chapter? As I thought about it I was amazed to realize that as late as the early 1980's new computers were being manufactured with KSR and ASR devices as input terminals. The reason behind this apparent backward practice was security! A VDU (visual display unit) did not create a permanent hard copy of what was typed on the console. Computer center managers were paranoid about what actually was typed, so that in the case of some failure or error they could find someone to blame. Was it the operator? The computer? The software? The janitor? They had to blame somebody or something. Without hard copy they had to believe what people told them, which, of course, was dangerous. So reams and reams of paper were stored just so one could go back and find out who was to blame. Of course most of the things that went wrong were innocent mistakes or malfunctions. But there were also outright cases of larceny or sabotage. I was involved in a case where a person was stealing software and was caught because he left the printout behind. He was copying restricted files that required name and employee number to be typed in. Sherlock Holmes wasn't needed to figure out that one. In another case a guy almost got away with some fraud by inserting an extra piece of paper between the print head and the normal output sheets. It so happened that a very sharp systems engineer checked the log that just happened to be also kept on disk and the perpetrator was caught.

I was never trained on the proper maintenance of ASR/KSR's and only met one guy who claimed to understand them. He had a unique method of maintenance: "oil it the first time and clean off the dirty oil the second time you service it." He professed. "How do you know which time is which?" I asked and as far as I remember I never received an answer. One advantage they had over newer devices was noise. If you wanted to be alerted that something was printing, no sweat. One could hear a KSR clacking away across the room, so never a message was missed. One might have trouble reading this message since the print quality was terrible, but you knew you had something.

Early in my days at SDS before I gained a reasonable reputation as a troubleshooter, I was often stuck in a computer center while the "experts" from California ran remote diagnostics on

the system. We communicated via the operator's console, which was a KSR. I could see everything they typed on their end and they could send me instructions. Because of the three-hour time difference I was often working well past midnight. So when there was nothing for me to do I got drowsy. But, thanks to the KSR, the guys on the other end never knew, because if the system had a fault or they wanted me to do something, the KSR acted as my alarm clock. No missed messages on my watch!

When newer systems came out without hard copy terminals (mainly nice quiet screens) computer center managers had hard copy devices attached to them. You weren't going to deprive them of their security blanket.

As a service engineer I had another way of communicating with the computer. By means of the operating panel lights and switches. Yes believe it or not those flashing lights you saw in the movies really meant something. If you were trained you could read what they said. You couldn't read them when the machine was running at full speed, but you could take the computer out of run mode and determine what it was doing, what part of memory it was using and if necessary tell it what to do. Of course, you needed to be able to read hexadecimal notation, understand the internal language of the machine and what you wanted it to do. Because SDS sent me to school for six months I had the knowledge to do this and was often called upon to do it in order to isolate a problem; it was a valuable tool.

There is nearly always another use for everything; I found one for my trouble shooting skills. I was in the process of wooing my future wife, who was in the process of getting her degree in computer sciences. She was fascinated by computers and impressed with my knowledge of them. One of the things I knew how to do was turn the audible alarm on the computer on and off. I wrote a very short machine language program, used the panel switches to load it into the computer memory, what we called finger programming and then punched it out on the cardpunch. She could then send it through the card reader whenever she wished to use it. The program allowed her to play "music" on the computer by altering the position of four particular switches. She was really impressed with me and I was really proud of myself. And I bet you can't even make your mouse squeak!

--> What does KSR stand for?

Near and Far

Review time: CPU, memory, arithmetic unit, RADs, disks, card readers, card punches, paper tape, line printers, tape drives and KSR/ASR. An assortment of all or most of these made up a 1960-70 computer room with each contributing its own music to the symphony. In the 1990's we call it noise pollution, but in the good old days it was music to our ears. Matter of fact, the worst sound you could hear was silence, that meant the computer was down! And unless it was in the wee hours when the only creature stirring was your trusty field engineer performing p.m. (preventive maintenance), you had a problem.

Only a few (not necessarily the brave) were permitted to enjoy this rhapsody. Computer rooms were secure facilities -- only authorized persons were admitted. It came as a shock to me the first time I was denied entrance to a computer room. I had always been in the position of being forced to be there. Being denied entry would have saved me a lot of grief. Yet, it would have cost me a lot of money, so I won't complain now.

But mere mortals were restricted to the outside of the computer room. So, how did they get their computer jobs run? In the 60's many computer rooms had a wall made of numbered cubbyholes. A user (which in fact is a class of human being, in some cases subhuman) would place his job (card deck, tape, instructions) into his assigned cubbyhole. At some point in time (always later than the user desired) an operator would remove the job, run it, and place the results back into the cubbyhole. This wall probably received more abuse than the one in Berlin. Users would accuse operators of avoiding their jobs, losing their output, or aborting it simply because it had overrun the allotted time. Operators developed selective hearing and were known to accept bribes or favor certain people. I have difficulty remembering pretty young women complaining; maybe they were just nice people.

However, I do remember the opposite. I was sent to help out at a university in western Pennsylvania. Upon arrival the fist thing I noticed was how attractive the female operator was. Several hours later I was surprised to see who I thought was the same young lady still working. "Doesn't she ever go home? I asked. "Oh, that's a different one," my colleague said. "All the operators are co-eds and the director seems to pick only the cute ones." None of the male users ever complained about waiting for their results, some made it a habit! A few years

later, the on-site engineer, who I was sent to help, married one of the operators. So, I'm not the only one to find love in the computer center.

Of course, something needed to be done to improve productivity and get people back to work. Thus, was born the computer terminal, named because it was on the end of the line (terminus), either a telephone line or what was called a hard-wired line. The difference being whether or not you needed a modem. !!CAUTION!! I am about to enter a technical part of the discussion. If you are sleepy or bored, which really hurts my feelings, get a cup of coffee or take a nap.

O.K. Now that you are fully alert, what is a modem? Of course, since you all have PCs and are on the Internet, you know that it is a modulator-demodulator. What it does is to convert digital data from the computer to analogue data for the telephone. I used to teach a several hour class on this stuff, but have no fear, I will keep this shorter and sweeter. If you wish to know more, invite me to dinner and I'll tell all. Anyway, a hard-wired line did not need a modem because it was all digital (and you thought SPRINT was first). However, the line was allowed to be only 50 feet long. So, now that someone had a terminal, what good did it do? One could submit jobs "remotely", checking on the results from his desk (anyone more than 50 feet outside the computer room, needed a special telephone number and modem). As years rolled by and disk space became more plentiful, the card images for jobs were stored on and retrieved from these secondary storage devices and did not have to risk the gauntlet of the card reader. The results were likewise capable of being stored on disk if that was desired. But these weren't requirements for remote operation, only niceties. In the early days, not everyone had a terminal on his desk. The operator still had control over which jobs ran when from the operator's console. It was just a matter of keystrokes, not physical labor, so the job stood a better chance. This was called "batch processing." The operator grouped a batch of jobs by some criteria: length of time, importance, production, development, resources used, etc. The groups were called classes and the operator could start all or part of a particular class. This was the mode for a long time until time-sharing was invented, but more on that later.

Use of the terminal and data communications totally changed the way we operated. Users never even saw the computer. They did not even need to be in the same city or state. Also, computers

could talk to each other and share resources. The latter has given rise to many sci-fi films and stories about the computers taking over the world -- still a scary idea.

Terminals came in all shapes and sizes. Our old friend the ASR/KSR, video screens, card readers and punches, even tape drives and line printers. My favorite and the first device I was ever trained on was the RBT, Remote Batch Terminal. Once again fate took a hand. I had just started at Univac. I was awaiting a spot in a computer class (9200). My boss received a telephone call from Ilion, New York, where Univac had its training center, asking for candidates to attend a two-week class on a new communications device. The class had not been scheduled in advance and they needed a few students to fill it up. My boss tried to get some of the older, more experienced guys to go. However, since it was already Thursday and the class started Monday morning, no one wished to go. Reluctantly, my boss asked if I would go; I jumped at the chance. Hanging around the office was a good way to get in trouble, the prospects were running errands or lifting rotating memories.

My two weeks in Ilion were exciting and rewarding. Being a "new guy," I was anxious to learn and very happy to stay late. The veterans in my class all seemed to be preoccupied with their own importance, seeming to always know more than the instructors, a common disease in the computer business (I know, I experienced a mild case later in my career).

The RBT was considered an insignificant device. My peers felt it was a waste of time. They could not have been more incorrect. The RBT was composed of a column punch, card reader, line printer and communications control unit. Best of all, it contained a manually controlled diagnostic system.

In 1967, data communications was in its infancy with diagnostic tools being rudimentary at best. The system in the RBT was simple and great at isolating problems, IF you learned how to use it and then did so. Since the big shots in class didn't want to be bothered, I hogged all the lab time, becoming very familiar with what was to become "my baby."

My work at school paid off almost immediately. A call came into the office for a problem on a 9200 computer. Both of the trained technicians and my boss were unavailable. The secretary, of all people, asked if anyone knew anything about the punch on a 9200. It happens to be the same as the one on

the RBT. So, I said, "I do." She was a little skeptical about sending a new guy, but figured it was better than sending no one. I borrowed some tools, not having received my own set yet and proceeded to the site. I could not operate the computer, but got the operator to run it for me, found the problem (switch adjustment) and fixed what was a very simple problem (sound familiar?). When I called the office to report my success, my boss was amazed and delighted. He had no idea what I had been trained on, thinking the RBT was just a console or something like that.

The next week a new 9200 was delivered and I was sent to help the senior engineer install it. I spent the week with him learning a lot, which was very fortunate because at the end of work on Friday he handed me his tool box and said, "You had better keep this, I don't work here anymore." He and the other 9200-trained engineer had both resigned that morning and were instructed not to come to work on Monday. At that time, many computer companies did not allow employees to work after they resigned due to security reasons. There had been some cases of sabotage reported, therefore the companies paid the two weeks notice rather than take a chance. So, magically I was the senior 9200 man in the office. Two guys were sent for training in the next class, but not me. I was too valuable to go. As fate would have it, I never went to school on the 9200, a fact that eventually led to my leaving Univac.

--> Why is a terminal called a terminal?

Watch Your Language

As impressive as finger programming might sound, it wasn't real productive for actual applications. Instead of using an abacus working with ones and zeroes, a calculator with commands like "plus," "equal," "percent," "memory recall," etc. were required. There had to be a way for humans to express their desires to the computer. Kicking and cursing did no good except to help relieve your frustrations, ask the crew of The Starship Enterprise.

Thus were born computer languages. The earlier ones were called things like Symbol, Metasymbol and Assembler, because they were very close to the machine language of the computer and were a way of representing the code in a more friendly manner. For instance, the computer command to store a word in core memory was a hex 35 followed by the source location and finally the destination location, all in 32 bits of binary data. Using Symbol the programmer would use the mnemonic "STW" to indicate the command, a number to show the source and a label to show the destination, like this: STW 2,DEMO. (Now do you believe that this is easier than toggling 32 switches with the binary code 00110101001000000000001000010000?) The destination, DEMO, would be calculated by the assembler so the programmer did not have to spend time managing memory. This was a step forward but required the programmer to know all of the machine commands and what they did. A task more difficult than most programmers desired.

The next step was to use mathematical expression that was independent from the computer in order to generate scientific programs. The most well known of these languages was FORTRAN (formula translation). Using FORTRAN, the programmer could code

```
Total = sub1 + sub2
```

would generate the following symbol code

```
LW, R1ST1
STW,R2ST2
AW, R1ST2 .
```

Some expressions created three lines of code, some examples would generate many lines. The user didn't need to know any of this machine code, and his program would run on any machine that supported FORTRAN. At least he hoped so. FORTRAN was the

mainstay of the scientific world for many years and is probably alive and well in some places even today.

But it wasn't really suited for the business community like banks and insurance companies. The solution for this came, from of all places, The United States Navy and a lady called Grace Hooper. It was named COBOL (COmputer Business Oriented Language) and was even more English oriented than FORTRAN. The following expression would generate the same code as our previous example.

```
LET TOTAL_AMOUNT EQUAL SUBTOTAL_1 PLUS SUBTOTAL_2;
```

Once again the user was free from machine code and had a transportable program. Now you couldn't write just anything you wanted and expect the compiler to understand it. Only certain words were recognized and they had to be in the correct sequence in order to get the desired results.

Symbol, Metasymbol and the like were assemblers because all they did were assemble a program from the code provided and were unique to each machine. FORTRAN and COBOL were compilers that took "high level" language "source code" and compiled an "object" program, which then had to be run through a machine dependent assembler. Easier for the programmer but more steps involved.

The source code for COBOL and FORTRAN was supposed to be compatible with most computers allowing one to compile the same code on two different computers and get the same results. Don't bet the farm on it! Many an hour was spent trying to transport code from one system to another. A lot of consultant money was made also.

Another very popular computer language is called "BASIC." BASIC is an interpreter, the code again looks like English

```
IF X=Y GOTO LABEL
```

and like COBOL and FORTRAN it generates multiple lines of code for each statement. But unlike assemblers and compilers it does not generate permanent code that can be used over again and again. BASIC does not create any thing permanent and must be run each time just like the first time. Inefficient for frequent use, but one big advantage is it is easy to update, because the source code is the only code.

One of the many dilemmas data centers face is lost source code. Joe Blow wrote the program five years ago. He now works and lives in another state. His program (in object form for speed) is still used every day. But nobody knows where the source is. So if a change is desired, there is no way to do it short of recreating it, or calling Joe, who just might have a copy at home. Sound far-fetched? Not exactly. Early programmers were paranoid and over protective. Joe might have taken the source on purpose and will be willing to sell it to his former employer. "It belongs to the employer," you say. Maybe so, but can they prove it? If that sounds bad, how about the guy who programmed the system to crash the first time his name wasn't on the payroll. Or the guy who's name will always be on the payroll? In the early days there was little if any security and the general rule was only the programmer could read his own code and then only for a short time before he or she forgot what he set out to do. However, code was supposed to be "commented," which meant for each line of code, using a space provided for him, the programmer would document the code with appropriate comments that produced no code but supposedly described what was happening. Great! As long as comments were used and actually did what was described. Like: "Store the totals now," or "Go find the interest rate."

The computer did not read the comment, it read the code. So it might do something totally different like "Add 10% to my pay check" or "Raise all my grades to A's." (Students did work in computer centers.) I'm sure that more than one old programmer is still collecting off an old program nobody can find the source for.

One type of programming where you didn't have to worry about source code was plug board, earlier I mentioned it and promised to get back to it; can't avoid that any longer. Avoiding it was probably a reaction to the way most of us always approached it, with great caution. plug boards were an early way of creating and storing often used but simple programs. The board itself was about 2 feet square, the front was made up of many small round sockets (plugs). The back contained pins that matched up with a receptacle on the front of the computer, actually the control section. The programming was accomplished by plugging wires into the sockets starting with the first position and continuing to select sockets to control the actions of the machine. Each socket represented a different instruction and the order the wires were plugged in created a program. This

often required hundreds of wires. The board would eventually resemble a tangle of colored spaghetti.

In the case of the 1001 Card System I covered earlier, the plug board was used to sort cards. The wires selecting which fields to examine and what to do depending on what was read.

Most plug boards had covers to prevent wires from being pulled loose, a vision that still scares me. Imagine figuring out where a wire belonged in that maze. The boards were mounted on the front of the computer and held in place by means of a locking lever. Some sites had drawers full of them and switched them back and forth all day, while other sites might use the same one all the time. Wide usage ended early on being replaced by tape or disk, but government installations used them for many years.

In an odd way they were the first "firmware" a term which can normally start a heated discussion among computer professionals as to exactly what it means. The first definition being something that isn't soft or hard. So from that vague notion the differences arise. My feeling is that pc chips i.e. The Pentium Processor are firmware, programming that is etched or written permanently into circuitry and not meant to be changed. Originally we called such things "ROMS" Read Only Memories. They were created in a factory by using a machine that "burned" the information into the silicon of which they were made and then were plugged onto circuit cards. When you needed to upgrade or modify them you had to get a new rom from the factory. This gave birth to the "PROM" programmable read only memory. Which could be re-burned at the factory rather than discarded like a rom. Due to the rapid drop in semiconductor costs proms did not stay around very long it was a lot cheaper to use new roms or chips.

There was also a form of firmware that added the ability to selectively perform extra operations not originally included in a processor, by means of adding or subtracting small jumper wires. This was called "Microprogramming". So I hope you can see why I say the plug board was the first firmware.

As time went by, more and more languages evolved, each to suit a particular situation and each creating another set of problems. I've already mentioned RPG and then there was PL/1, Programming Language One, a derivation of PL/1 was "PLUTO" Programming Language University of Toronto (no dog jokes

allowed). Other names were SNOBOL, PASCAL, APL, ALGOL and last but certainly not least "c."

But no matter what the name or intended use they still had to generate machine code and because each might generate different code to do the same task, results could vary from computer to computer. All manufacturers claimed their version was correct and any difference was the other guy's fault. Finger pointing was an early computer game with no joystick necessary.

One situation that often occurred was an error or abnormality that was turned into a feature. Even though the results weren't what was first desired. For one reason or another someone made use of it. Which gave birth to the saying. "If you can't fix it, feature it." But lo and behold the manufacturer would fix it and the person who had learned to live with it would get upset and request to have it unfixed. As a field analyst I was often caught in the middle of this dilemma and was forced to submit a SIDR, Software Improvement Development Request, the manufacturer never wanting to admit to a fault. So fixing a problem was known as an improvement. The customer would complain to his local analyst, me, and the only option open was to fill out a SIDR, a form which could be very simple or extremely difficult to fill out, usually the inverse of the problem. If it didn't work at all the explanation was simple, but if the quirk only happened under certain conditions, the description could require a lot of documentation which was always required and was necessary if you expected a resolution.

The interface between the customer and the home office was the local analyst. This often proved to be a difficult situation. The customer was normally under the impression that his problem would be fixed quickly. In order to facilitate this the SIDR was assigned a severity level from one to five, five being the most severe type of problem. The customer always felt his problem was a five, but it would probably become a two or three. The first time someone gets involved in this activity, he is very optimistic in getting a result. But after a few attempts he becomes more pessimistic. Especially when he discovers the volume of problems being reported and the small amount of resources being applied to solve them. It takes a brave soul to keep on trying. One of the most common solutions received was "Fixed in next release." This looked good on the list of closed SIDRs but did nothing to help the current situation. Unfortunately, many times it wasn't fixed in the next release. So what happened when the customer complained? He

was told to resubmit the problem.

Some users also abused the reporting system by submitting a high percentage of SIDRs. They would actually go looking for problems, creating weird situations which were not practical, even going so far as saying, "It doesn't matter whether I use it or not, it's supposed to work." One such guy really got caught in his own trap. He took a job working for the vendor in their software development department. The first job they gave him was to work on his own SIDRs.

That all may have sounded a little negative but there were plenty of times the serious problems were fixed and others where there was a "work around," a different way to do things, but achieve the desired results.

Today most users are not programmers and don't have to learn a language. Just the codes or function keys necessary to complete their task. But they should realize that every time they push a button, or click an icon, lines of code are being used that were generated in some back room by someone using a language possibly "c."

--> What did FORTRAN stand for?

Hard vs. Soft

If you've been paying attention you might have realized that in the last chapter I became an analyst. What happened to engineer? Well, just as the industry itself moved from a dependence on how to build a machine to an importance on how to make it work better, I made a similar change. It was not a planned or sudden thing, but a definite metamorphosis.

I never had a desire nor was I ever a programmer. I liked fixing things and think I was pretty good at it. However, one of my favorite activities as an engineer was working with the software guys to isolate a problem. They required my skills on the machine and I was awed by their knowledge of the software. After one particular hairy episode, the analyst I was working with asked, "Why don't you get into software? You'd be good at it."

"No, I like fixing things."

"Software needs fixing, too, and you could use your hardware skills to great advantage."

At that moment I discounted what he said, as far as I knew, software was written stuff you couldn't put your hands on. My lead engineer at the time even said, "There's no challenge to software. That's why they call it soft!"

However one thing the analyst had said stuck in my mind, "It pays more."

So at my next performance review, when asked what I wanted to do in the future, instead of saying, "I don't know" or "what I'm doing now," I said, "Get into software." That statement must have impressed my boss because a few months later he called me into his office to announce, "There is a position open for a printing systems analyst, I don't know what that is exactly, but it's yours if you want it. You'll still report to me and you'll still work on the hardware." I said "yes" and it would take five years before I really did that job, leaving Xerox and returning, but it was the opening that would change my career and probably my life.

I then attended software classes, learning some terms, was assigned to be a computer analyst and thanks to my foxy boss continued to work on the hardware. He was getting two guys for

the price of one. The best example was when he sent me out to evaluate a problem to determine whether it was software or hardware. When I called to inform him it was definitely hardware he simply said, "Good, fix it." Which I did.

It was not that easy a transition. On one occasion I was sent to the programming class "Metasymbol." All of the other students were software types. I soon discovered there was a secret software language, which I didn't understand. There wasn't a secret handshake but a code none the less. The class was scheduled for two weeks and by the third day I thought about quitting. I was last in the class and none of my programs would work, I was even having trouble understanding what was wanted. All of the assignments were in a shorthand I didn't comprehend. Luckily my classmates were very nice and they all tried to help me. Without their help I would have failed; something I had never done. Still, at the end of the first week the instructor called me aside and said, "Your working hard but you keep getting further behind. Next week is much more difficult. I don't think you'll make it." Nothing could have made me more determined, I promised to study over the weekend and if I couldn't keep up the second week I would go home.

I was really worried, but had an ace in the hole. The girl friend I've mentioned before was now my wife. She was also a gifted analyst/programmer. She did not write my programs but did tutor and advise me. And after long, sometimes frustrating hours, I broke the code! My problem was syntax. I knew what I wanted to do and how to do it, but getting the assembler to understand, what I wanted was my problem.

On Monday, I ran all of my programs successfully, but still had to prove I was able to keep up. The second week was very difficult for everyone in the class. Except me! We were covering very complicated machine instructions, those that were not normally used by the average programmer, but used by engineers all the time. I now understood the interface between man and assembler and it was a simple thing to convert my machine skills to Metasymbol commands. I was soon returning the kindness of the previous week and helping my classmates. One of them eventually made the following comment. "Who is this guy? Last week he couldn't get anything right and this week he's at the head of the class. Is he a ringer?" It was said in good fun, I think.

I was never as good a computer analyst as an engineer. But I

eventually became a great printing systems analyst. But as I've said before that's another story.

This chapters question:

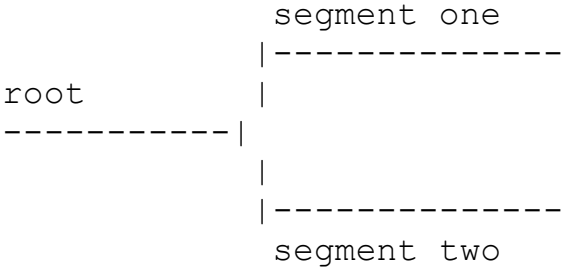
--> Did I make more money?

Laying It On

One of the great mysteries to 70's software writers was how to run a one megabyte program in 128K bytes of memory, which was normally the amount of memory set aside for program execution. This limit related to the maximum amount of disk memory that could be accessed in one command and was called a "page." The limit was very real and strictly enforced by both the hardware and operating system software. One byte over the limit and your program wouldn't even load, probably wouldn't assemble or compile.

The solution was "overlays." An overlaid program consisted of a "root" that was always in memory and an almost limitless number of overlays. Managing all of this was a tricky business. The overlays were on disk and could be brought into memory by means of a command known as a "call." The overlays were known as "segments." They could be of varying sizes from 1KB up to 64KB. The structure of your program would be designed on a piece of paper and called a "tree." The root was shown first; its length indicating its size in bytes. The length of the paper was equal to 128KB. And all the segments used at one time had to fit on the page. So if you needed more memory you designed another overlay by drawing a line parallel to the existing lines. Lines parallel to each other may not be in memory at the same time.

Let's try a simple example. Your program requires 192KB. Your root is 64KB so you design two overlays of 64KB each. When your program starts, the root is loaded into memory and starts to execute instructions. When it needs some of the code in one of the overlays, that segment is called into memory. The two "memory resident" pieces of code can then execute as one program until the code in the non-resident segment is needed. The other segment is then called and replaces the segment previously called in. Now the root and the second segment can execute together, if the first segment is required it is called to replace the second segment and so forth and so on.



A couple of rules: variable data is not allowed in segments, overlaying would wipe it out; segments not in memory at the same time can not reference each other. The root is never overlaid, because of this sometimes the root is very small.

That was pretty simple and easily increased your memory capacity by 50%. To get more you just needed more overlays. The hitch being each overlay complicated your management dilemma. Keeping track of what was in memory all the time was a difficult task and became the source of many software problems. Messages like "Referenced word not in memory" or "Program too large" or "Not enough memory" would bring your program to a screeching halt. And as is the unwritten law, you never got enough information to figure out what went wrong.

Another bad situation was "thrashing." Each segment call required a disk access. Disk access is a very slow process compared to memory access. Like comparing a marathon to a sprint. You can stretch your memory size (this eventually became known as Virtual Memory), but you had to pay a price and that was in speed. If your overlay design called the same overlays too often your program spent a lot of time waiting for disk access, sometimes just to execute one instruction, and then would call the same segment a few instructions later. This was thrashing and would soon result in your program being aborted due to using too much time. Users were allotted only so much CPU time and when it was exceeded they were kicked off the system. They also paid by the second so it could get real expensive real quick.

A good example of this occurred when a certain customer upgraded his system from a memory only system to a disk operating system (DOS). The users had been made to overlay all their programs to take advantage of the disk and became very upset when their programs ran slower. A quick look at the disk I/O indicator showed much input/output traffic. By checking what was being accessed it happened that one particular segment was being called over and over. The remedy was to use a command that "locked" that segment in core, noticeably improving performance. But an overlaid system will never run as fast as a memory only system.

Frequently requested segments that performed generic functions, like "find the square root" or "calculate the tenth polynomial" were cataloged and called "subroutines." They were often supplied by the manufacturer, in files called "run time

libraries." A user could add his own subroutines to this library to save programming efforts. Why reinvent the wheel?

Now if that sounds complicated, wait until you get to the next chapter.

--> What part of the overlay structure was always in core?

Rolls and Slices

That brings us to the next logical topic: operating systems and time- sharing. Operating systems are just huge overlaid programs that control the actions of the computer and all of its components including the user interfaces. Initially each computer ran one job or task at a time. An operator, or the user himself, loaded cards or tape and started the job and it ran until it was finished, one way or another. The first IBM operating system was called OS or OS1, you can guess what that stood for. Then came TOS (T stood for tape), which allowed information to be read from or written to a tape drive, followed by DOS (Disk Operating System), not the one that runs on your pc today. This DOS was used on the large IBM 360 and 370 mainframes. Next was VS (Virtual System), a great source of jokes about what was really virtual about it (like it virtually never worked). We also had VM (Virtual Machine), you really had one but it could pretend to be something else. And, the king of the 1980's, MVS (Multiple Virtual System). By then things had gotten overly complicated. The customers were sold a set of computers, but had to be able to talk to them as if they were one.

Don't get scared, I will not attempt to go in to more detail on each one. Just remember, at the same time, all the other manufacturers were creating systems of their own, some with very similar names. I worked on things called RBM (Real-time Batch Monitor), CPR (Control Program Real-time) and CP-5 (Control Program Five).

They all did similar things, often depending on the environment in which they were used. It was possible to use two different operating systems on the exact same hardware at the same time.

But best of all was allowing more people access to the computer without having to wait their turn. An example of this was to partition part of the computer's resources to run sequential "batch" jobs, but allow the other parts to be used by many people remotely in what appeared to be the same time. Thus was born time-sharing.

Many users could gain access by means of a terminal and communications port. The early terminals were considered "dumb" terminals, meaning they did no calculations or data storage on their own, just like a manual typewriter with communication wires. The connection was called "echo-plex," each key stroke

was first sent to the computer and then echoed back to the terminal before it appeared on your paper (later, video screen). This was a good way to verify that what you typed had actually gotten to the computer. Under good conditions this was transparent to the typist. However, if the system was running slowly, which early systems always did, you would find yourself staring at the terminal waiting for your echo. You could type ahead, if you (not me) had the confidence in your typing skills, the system would eventually catch up and spit your typing back at you, but only for so long.

What happened during that time? Was the machine on a coffee break? No your "time slice" had probably run out and you had been "rolled out" of memory. When your slice came back you were "rolled in." The amount of time allocated to the "foreground," the time-shared portion of the computer's resources, was measured in slices. Depending on the importance of your job, the size and time allocated would vary. But everybody got a slice. The concept was to make it appear that each user had the machine all to himself. However, as soon as everyone tried to access the computer at the same time, this illusion quickly disappeared and performance slowed dramatically. Parameters could be varied to change the situation, like making the slices smaller or giving everyone more or less time. But the pie was only so big and the final solution was for some users to "log off" until performance improved. This could be done forcibly, by the computer operator, but usually was done out of frustration by the users.

I've been testing you. What did roll out/in mean? Roll out meant having the contents of everything you were doing written to disk, and roll in meant having it read back into memory. It was also called being "swapped" in or out, or being "paged" in or out. The location you were swapped to was a temporary location on a high-speed mass storage device. The RADs I spoke of earlier would be used for this, not disk packs. Remember foreground refers to the area of memory reserved for online or time- sharing jobs. There is a background area that is used for batch jobs, the ones that ran from card or tape.

Needless to say, there was a lot going on and a lot could go wrong. The worst thing was a system's crash. Nothing physical was damaged but the operating system would find itself in a situation it could not handle and, in an attempt to prevent serious loss of data, would shut it self off, hopefully without further loss of integrity. At this point, you could lose the

last few things you had done and might need to do them over, which is why you always backed up your data. Everyone hates crashes. In the early days, they were far too regular. But as time slowly went on, they became a rare but still serious occurrence. They will never be rare enough, but as long as there is a push to develop new software, there will be problems. You wouldn't want to put system analysts out of work would you?

They may be the only ones that know.

--> What do VM, VS and OS stand for?

Jockeys and Hangers

Operators, by any other name. The lowest paid but often most critical part of a computer room. Jockeys were called so because they jockeyed the disk packs in a large computer center, which might have as many as 30 or 40 drives. The number of individual packs could be in the hundreds. Periodically, a message would appear on the operator's console requesting a particular pack be mounted on a particular drive. Each pack was physically labeled, usually with a gummed paper tag bearing a code which showed who owned it and so forth. There was also an internal label that was recorded on the disk itself. When the operating system requested a "disk mount," the operator would go to a specially designed rack and locate the appropriate disk, mount it on the drive and return the empty cover to it's proper place. Then he would inform the system that the task had been completed. This was often a hectic job with requests coming in as fast as the operator could service them. Of course, if there was already a pack in place, which was mostly the case, the operator had to dismount it before he could mount the new one. He could do this since the operating system kept track of which disk was where.

Now what could possibly go wrong? The operator could mix the packs up by using the wrong cover to remove a disk. This was usually discovered quickly but still caused a glitch in the operation and unneeded confusion. He could also put the pack on the wrong drive and once again the system would catch this. But worst of all the disk could be damaged by mishandling, slamming a pack into a drive was a no-no. Yet the pressure to keep up often caused this, usually a platter or two would be bent. This was bad because the retractable heads would then try to access this scrunched disk pack. But what was terrible was, if not discovered quickly, the damaged disk could be moved to another drive and damage the heads on it. This scenario could get very ugly with bad heads creating bad disks and bad disks creating bad drives. A data center manager's worst nightmare. Now you know why I said operators were critical to a data center.

In the tape area they were called hangers because one hangs a tape rather than mounting it. Tapes were also easily damaged, but not as often and the fault was not spread from drive to drive. The operators big contribution here was cleaning the drives and failure to do this would soon corrupt lots of data so, once again, they could have a big effect on a manager's sleep. Good operators were equally important to service

engineers. If they did their jobs well, the number of service calls was greatly reduced and thus the C.E.'s job was made a lot easier. So a smart engineer cultivated the operators and reaped the rewards. The not-so-smart ones used them as scapegoats and would never figure out why things never went their way.

I learned early on to work with the operators as much as possible. Not only did it encourage them to do their cleaning chores properly, they could often help to isolate a problem by being good observers. Taking a minute to include them in your trouble shooting went a long way toward improving that relationship.

One time in particular comes to mind. I had been called in on a problem where the machine had been down for over 12 hours. The engineers had made little progress. At about the same time the operators were changing shifts. One that I knew pretty well came over and asked. "Is this still down? I thought with that big puff of smoke, it would be easy to find." "What puff of smoke? Where?" He pointed across the room from where the engineers had been concentrating their efforts. No one had mentioned a puff of smoke, even though we joked about machines burning up or going down in flames. Smoke is a rare occurrence; the voltage levels on newer machines are extremely low. In the very early days, tubes and resistors did actually burn. But this was a transistor machine. I checked the log and there was no mention of smoke. I asked the operator to show me where he had seen the smoke exactly. He stood next to me as I began pulling cards from the machine. On about the tenth one he said, "That's it!" I turned the card toward me and sure enough there was a big black spot where a small capacitor had burned up. I replaced the card and all was well. When I called it in, everyone in the office was pleased, but they refused to give the operator credit. Feeling he had held out on the original crew. When I bought him a coke and mentioned he might have told the other engineers, he replied, "They never talk to me except to blame me when things break, so I don't talk to them."

Of course there were plenty of operators I could have done without. They never cleaned their equipment and then lied about it, or if they broke something they would try to hide it till later, usually making things worse. Many of the better ones were only working as operators on their way up the ladder; the ones that remained were normally the bad ones. I, of course, married the best one. Her mother isn't sure that was a step in

the right direction.

--> What did jockeys do?

--> How about hangers?

Finding Fault

Through all of this dialogue I've spoken of fixing things, which is what I loved to do and was lucky enough to do it for both hardware and software. But what did it take to fix things? Many times it was a slow analytical process, a step by step approach that eventually culminated in the solution to the problem. Sometimes it was dumb luck, other times experience and, last of all, brute strength.

First the classic. The computer reports a fault and displays an error code. You refer to a book that explains what the code means and you understand it. You then devise an approach that will cause the computer to repeat the problem. This can be done via the use of prepared diagnostics tools, or by creating your own diagnostic loop program. My skills at fingering code into the switches really had a useful purpose. Once you establish a repeatable failure you use your knowledge, the system schematics and an oscilloscope to trace the flow of logic within the system. An oscilloscope is an electronic diagnostic tool. You've probably seen one in a sci-fi movie. It has a green screen that displays sine waves (arcs showing the rise and fall of voltages). Its primary job is to slow down the signals generated by electronic devices so that you can see and measure them. It has two or three probes with clips on the ends that allow you to attach them to pins, or test points, to monitor what's happening. You can freeze things that are happening in millionths of a second so as to compare two signals to check their relationship with each other. They are used for testing all kinds of electronics especially computers. It takes a skilled person to use one and they are invaluable for finding some problems and making adjustments, just by moving a probe from pin to pin. (The wires that connect all of the printed circuit cards to each other are run from pin to pin and wrapped around the pin to make a connection. This is done by huge machines that are computer controlled. So, computers really do propagate themselves. The pins are on the backside of the computer cabinet.) By following the schematics, you determine what should be happening and look for what isn't. When you find it, you can then fix it. In the earliest days, you then took a soldering iron and replaced the offending resistor, capacitor or transistor. When, later, we advanced to printed circuits and chips you would replace a chip and eventually you only needed to replace the printed circuit card. This was the classic fix and any engineer was proud when he had successfully completed such a task. How long could this take?

If you were really good, really lucky and the moon was in the right phase, two hours or less. Otherwise, forever.

How frequently do you think the above took place? Not very often. In my 15 years of problem solving, less than ten times a year! That does not mean I didn't fix problems. What it means is that the right circumstances occurred very infrequently. The problems came, but to be able to create a repeatable scenario was difficult and sometimes impossible. At such a time your options were varied, as were the results. One guy I worked with would open the computer door and tap on the ends of the printed circuit cards, which were on the front side of the cabinet, until a failure occurred. He would then replace that card and announce the problem was fixed and make a rapid exit. His success rate was minimal.

Another trick was to alter the voltage levels. Many machines had a switch to do this. Its purpose was to insure the machine would operate correctly at extreme ends of its tolerances. Sometimes when you did this the problem would go away, this was not a solution, but more than one machine was sure to be running with the switch set one way or the other. It was always tempting to leave the switch set especially at 4 o'clock on a Friday. I plead guilty to that offense. In that situation the "fix" was probably only temporary and the problem would come back, but hopefully it would be repeatable when it did. And happen at 10 a.m. on a Tuesday.

The most popular method of trouble shooting, when you couldn't create a repeatable situation, was module swapping. The printed circuit cards (modules) were often grouped by function. Arithmetic, control, I/O, etc. If you could isolate the failure to a particular function you had a smaller number of modules to deal with. If spares were available you could replace the existing cards and see if the problem went away. Many purists frowned upon this practice but as more and more new engineers were pressed into service, as the industry grew, it became more prevalent. The dangers were numerous. If you weren't careful you could introduce a new problem. One of the aggravating things that could happen was the problem would go away when you swapped a card, but not return when you reintroduced the original, which was the proper step to verify the results. What now? The customer would want the new card reinstalled, but your boss would probably disagree, since he paid for the parts out of his budget, which was one of the ways his performance was measured. The parts man didn't want to send a card back that

wasn't bad either and would not put a questionable part back in stock. If possible we would leave the original in place but be prepared to replace it if the failure returned.

How could this happen? Many of the problems were attributed to seating, which was the act of inserting the card's contact pins into the card cage. Each card had from 10 to 50 pin connections. It was possible the card had not been properly inserted originally, had come loose due to vibration or the contacts had become corroded or dirty. Any of these conditions could be fixed by simply removing the module and reinstalling it. Not all cards were easy to remove, being designed to fit snugly in place. Each different machine had different size and shape boards (cards, modules) and a unique module puller. So we engineers had a whole array of pullers, some provided by the company and some home made. They were normally metal or plastic tools which had pins that would fit into holes that existed on the edge of all modules. Some could be used as levers, which eased the cards out, while others, required some amount of brute force. Reinserting modules could be deadly on your fingers since you had to press on the rear edge to insure a tight fit. There were also cards that had levers that assisted the seating process.

The real danger of module swapping arose when you didn't have a spare. The modules that made up a large system were not all unique. The designers made use of as few unique cards as possible, in fact, the same one could be in many different locations performing totally unrelated functions. This was accomplished via the backboard wiring which connected all the components together. It was very possible that all the components on each card were not being used. So a swap could hide a bad component in the system just waiting to jump up and getcha.

Each engineer had his own level of tolerance when it came to swapping. Some did it too often and others refused to do it at all. I was somewhere in the middle and could be influenced by circumstances. One of my favorite stories came about when we were working on a disk controller problem, which had been down for over a day. When this amount of downtime occurs, engineers start working in shifts. Usually the shifts are too long without a lot of rest in between. I was at the end of a 12-hour shift and was about to be relieved by an old timer, who worried more about technique than results. A third engineer, who was junior to both of us, was also on site. Just prior to the old

guys arrival I had an inspiration as to where the problem was. We had been looking in the wrong place. Just as we were about to replace a module that I had selected through a logical process of deduction, my relief arrived. Out of courtesy, I explained what we were about to do. He said, "No your not" and stood between the machine and me. I was shocked. I tried to reason with him. But he became more stubborn. "You haven't proved it with a probe, if you can't show me the problem on a scope, we're not changing any modules until we do. That's the trouble with you young guys; always in a big hurry and never interested in proving what's exactly wrong." I was glad no customer heard that comment. I was tired and wanted to go home, but I wasn't going to leave without trying my idea, believing his real reason for stopping me was so he could take credit for fixing the problem. There was a lot of professional jealousy; our performance appraisals could be effected by how many fixes we got credit for. Since this guy was much older than the rest of us, at least 40, he felt threatened.

I called the junior guy aside and whispered my plan to him. "Find a way to get that clown out of here. I'll make like I'm leaving, but once the coast is clear, I'll sneak back in." Sure enough the junior guy asked him to look at something in a manual that just happened to be in the next room. I slipped back in made the change and was just checking out the results when they returned. "All fixed," I announced. The old guy exploded his face a fiery red and his cheeks all puffed out, "I'll report both of you for this," he screamed. "That should be interesting, explaining how you tried to stop me fixing a problem." I never heard anymore about it, but had made an enemy and would cross swords with him more than once.

Now it's time to fess up. I wasn't always perfect. The facts being that I was on the carpet one morning for refusing to swap modules. I had been called in on a problem which I felt could be fixed quickly. Quickly turned into two then three hours. The lead engineer had suggested mass module swapping at the two-hour mark. Not a logical swap, but just start changing modules until the problem went away. This approach was drastic and normally only used as a last resort. I still felt I was close to the solution. Finally at the four-hour mark a guy showed up with a crate of spare cards and I was ordered to swap them. Fifteen minutes later, the machine was up and running. It was only my good reputation that got me off the hook. "Next time, swap the modules," was my boss's admonishment.

I once got in trouble for a quick fix. Customers become very protective of their equipment and whom they want to work on it. They all had preferred service men. I had a following and Jim, who I've mentioned earlier, had his. There was some justification for this. Jim was good on tape drives, I wasn't. I was good on disks and RADs, Jim wasn't quite as good. But mostly it was just a personal thing.
One day I was in the process of calling in a completed job, when the lead engineer I was talking to got a call from a site, which I could actually see out the window. "Get over there right away, they won't be able to complain about response time on this one." Customers always complain about how soon you arrive after they call. The guaranteed time is normally two hours but when a machine is down, it's always too long.

Five minutes later, I briskly strode through the door and headed toward the computer, which had a printer problem. Because I was pretty sure I knew what was wrong, I carried only a small tool case, not my twenty pounder. The site manager saw me and rushed to head me off. "Where's Jim?" "On another call." "Do you expect to fix it with that?" He was pointing at my case. That's where I made my first mistake. "Sometimes I fix them with these." Pointing at the small screwdrivers in my pocket protector. Things then got worse. Finally, after he allowed me near the machine, I saw that the operator had twisted a control knob too far causing it to go off its track. Now it wouldn't do anything, even when twisted the other way. I took my finger and slid it back in place and returned the knob to its proper place and said, "No tools required. How could I have to apologize? I had arrived in record time, fixed the problem in seconds and been insulted. Yet they had called my boss and complained that I had a bad attitude. I avoided that place like the plague and they never complained about not seeing me.

It wasn't long before I had another run in with the "old guy." He had been promoted to regional headquarters as a support specialist, a transfer that made everyone happy. My personal site was having an intermittent disk problem and had agreed to allow me to work on it over the weekend (nice of them, hey?). However, they had done it through my boss, who had notified the regional manager to get the overtime approved. The regional manager then decided his new expert should monitor my work from headquarters, using a new remote diagnostic tool. This was easy for him since regional experts didn't get paid overtime.

The day started off badly with the expert giving orders and not allowing me to conduct the trouble shooting. The diagnostic tool was unfamiliar to both of us and we were spending more time getting it to work than looking for the problem. Finally we got it running and disk unit five got some errors. The configuration had eight disks drives but used only six. I was told to change the address of disk five to one of the other units, along with a snide remark that my disk maintenance left something to be desired. This ruffled my feathers quite a bit, but I did as directed. With in minutes the disk drive now addressed as five got more errors. The abuse from the remote end was now getting out of control. My integrity as an engineer was being questioned. The old guy was going to get even with me for the time I swapped cards without his permission. He was so concerned about showing me up that he wasn't paying attention to the problem, rather was going to report me for not taking care of my system. After changing disks addresses once more and seeing the problem move to the new disk five, I was sure it wasn't the disk drives but a problem in the controller associated with that address. Drive one through four and six all worked fine. After consulting the logic schematics, I discovered that there was only one place that dealt with the addresses of the disks. I tried to convey this information to the guy at HQ, but he ignored me and told me to clean the heads on the disks so he could run his diagnostics. I, of course, refused. I exchange the card I had identified and watched the system run faultlessly. I had one trick up my sleeve: in order for him to run the system remotely I had to relinquish control, and I didn't. He was going berserk on the other end of the phone threatening to report me for disobeying a direct order. I had to laugh; I hadn't been in the Marines for 10 years. Finally after returning the bad module to it's original spot and having the problem repeat itself, he agreed it was fixed but still promised to report me. He actually did, and my boss had a hard time keeping a straight face when he told me. Seems the old guy had gotten the regional manager out of bed to report me. But the one thing I could count on was that the machine was fixed and that's what counted. Plus, my computer room girl friend had been there and was impressed by my skills.

The brute strength fixes were the mechanical ones. Little deduction was required. Broken belts or gears, burned out motors and worn out fans were easy to spot. Belts and motors were the worst. Just like working on a car, if a belt broke, it was always the inside one that required disassembling half the machine. Of course, you don't wear a suit and tie when working

on a car. If you happened to ruin a shirt or tie, tough luck! We didn't get coveralls to wear.

Software trouble shooting was similar to hardware in that it took different approaches to find the cause of an error. When a fault occurred the first thing you did was check to see if a patch existed for this particular fault. A patch was normally a small set of instructions that were added to the existing code to fix a known problem. Operating systems were full of patches, you never saw a system without one. Even the new releases came with patches, some mandatory, while others were optional. Some patches were patches to other patches. Does a quilt come to mind? Manufacturers periodically sent patch tapes to the field, sometimes directly to the customer and others to the local analyst. With software, unlike hardware, the customer was actively engaged in the maintenance of the system, many doing the work themselves and keeping the analyst out of the loop. The best way was a partnership so that the analyst had a feel for the system and was up-to-date on the status of the system. The worst was when the customer only called when he had a problem, usually one he'd been trying to fix himself. So when the analyst arrived, the situation was critical and the analyst had a learning curve to get over just to understand the situation. A frequent source of trouble was failure to install patches or skipping some, and then installing them in the wrong order, which was almost always fatal.

Most of my sites cooperated and I maintained a good relationship with the software staff. My past as an engineer sometimes worked in my favor and against me in other cases. Those who believed my knowledge helped me understand the whole system, appreciated it. While those who felt analyst should never get their hands dirty, didn't. Some of the worst abuse came from my own colleagues who looked down on engineers as a lower life form because they didn't have degrees and were insulted when I was raised to their level. One even told me to my face that I would never make it. History proved otherwise. The ultimate software fix is to discover a failure, deduce the cause and create the patch yourself. This happens even less than the classic hardware fix. More commonly a change in approach or parameters is the solution. There is no such thing as identical systems. The hardware configurations might be the same but there are scads of internal time limits, buffer sizes, slice limits, etc, etc. Each one has a slight effect on something and each site modifies them to suit itself. In many cases, these are the cause of a problem. A site manager

discovers that a certain application runs faster if he raises a certain parameter. So he raises it some more until some thing goes wrong. But it never dawns on him to return all values to their original levels. And, of course, he never tells anybody. Access to system parameters is closely guarded. Only two or three people were allowed to alter them, having the password necessary. The local analyst may or may not be one of them. Normally, he does but only uses it in cases of great difficulty. It can be dangerous. One day I was at a university working with the site systems programmer and trying to improve system response time. We were monitoring a particular counter. It never seemed to change and we felt if it was lower the system might free up some memory and improve the response time. On our own we decided to lower the number while the system was running. No sooner had we entered the new number and the system crashed. We sat in front of the terminal staring at each other looking foolish. The site manager stuck his head in the door and knew from the looks on our faces that we were responsible. Since he was the one who had encouraged us to tune the system, he was forgiving and just said, "Be more careful next time." The system recovered with no ill effects except our pride.

On another occasion a regional support guy and I were studying a similar problem at a different site. Everything was very slow. He decided to change a number that controlled the number of buffers (subsets of memory) available to a particular task. And, in minutes, the system was running noticeably faster. Hoping to gain some knowledge, I asked, "What did that do?" "Not sure. Sometimes it helps, other times it crashes the system." It's as my wife says, "Sometimes it's just magic."

I'll end this chapter with a story that made me a legend. We had been working on a problem for many hours and had made little progress. My boss ordered me to go home and get some sleep. I went home and after about two hours I dreamed about the problem and what the solution was. I got up, called the site and insisted they do what I told them. When it fixed the problem the guy on the other end of the phone couldn't believe it, but he told everyone how I fixed problems in my sleep.

--> Well after all that can you remember what SIDR stood for?

Moving Experiences

Of course, before you can fix a computer, it must be installed. This is normally a long process of running connecting cables and bolting boxes together. Any troubleshooting takes place after all this is completed. Often there was very little of this to do since the machines were pre-installed at the factory. Otherwise, there could be hours or days spent doing it on the customer's site.

The real challenge was moving an existing system and reinstalling it. The major difference being time. A new system installation is scheduled to take longer than it should, in addition, it's new so nobody has been using it. Moving an existing system is a different kettle of fish. It has to be done on a weekend with promises that it will be up and running Monday morning. The promises are made by the salesmen but kept by the engineers.

My most adventurous move was in 1971 over the Thanksgiving weekend. It should come as no surprise that my girlfriend was involved. The company she worked for had obtained use of the machine they were using by means of a lawsuit, too complicated to go into here. None the less, the settlement ran out at midnight the day before Thanksgiving. No discussions had taken place between the vendor and the customer since the settlement had been reached. The customer had either forgotten when the agreement expired or felt if he kept his mouth shut he could keep on using the machine for free.

This was not the case. The customers head salesman and biggest user had entered into negotiations with the vendor and had agreed to lease most of the system if it was installed in a new site. As site engineer I was brought into the conspiracy, which included my future spouse, who would be the operator that night.

I identified which components would be shipped north and then devised a de-installation plan and secretly started unbolting the units a little at a time each night. By the day before Thanksgiving the units were just about free standing. Precisely at midnight, my accomplice pushed the off button on the computer and I let a team of engineers into the room. We swiftly completed the disassembly, we considered four hours to be swift, carefully labeling each cable and segregating parts into boxes for the new site and boxes that would be returned to

the factory.

At six o'clock a.m., the movers arrived and started the delicate task of preparing the equipment for shipment. The engineering crew took a breakfast break to allow the movers to build ramps and bridges to ease the heavy units across the many thresholds between the computer room and the main doorway six flights below. Normally this would have been days before but keeping the customer in the dark was important to our scheme.

By 10 a.m. the equipment for the new site, except for the boxes containing the cables, were loaded in the first van and the equipment for the factory was being loaded. As had been arranged, one of the field managers showed up to relieve me and supervise the rest of the loading. I had a Thanksgiving dinner date at my future mother-in-law's and wanted to take a nap before hand.

As luck would have it, at 10:30 a.m., the customer showed up with the police. Someone had called him and reported that his computer was being stolen. A few frantic phone calls prevented my relief from being arrested, but the now former customer demanded the return of his keys, which had been given to me while I was servicing his site. He ordered the movers off the premises, but could not prevent the loaded van from leaving, since he had no claim on the equipment. But he did manage to stop the removal of the carefully labeled cables and the tops and doors to some of the cabinets.

I was oblivious to all of this, happily stuffing myself with turkey and pumpkin pie. On Friday, I drove to North Jersey to assist in the reinstallation. Some of the guys from there had come to Philly to help me, so I was returning the favor. That's when I learned of the proceedings the day before. After all my careful planning we had no cables. All the equipment was in the computer room, but no cables. The call went out all over the country for cables right in the middle of a four-day weekend. We needed over a hundred cables of varying length plus the connectors and hardware to put them together. We commenced work and as each UPS truck arrived we got a little further. What should have been an easy task turned into a nightmare. The length of each cable is precisely measured to insure optimum performance. Yet we couldn't afford to hold up phases of the installation just because a cable was a few inches too long, or, in some cases, a few feet. (Of course we could have used the mythical cable stretcher. It was one of the jokes that is

used on the rookies sending them to the parts room for a cable stretcher.) None of us were rookies and we weren't in a joking mood. In one case, a 15-foot cable was used in place of a four footer because it was the only one available. That system would be plagued with strange problems for a long time.

By Sunday night we were getting close. The software guys were waiting to generate the new system software. (I was still an engineer.) The computer room was getting hot and we kept turning the thermostat down but it still got hotter. We were very concerned about the machine overheating until someone turned the thermostat to 90 degrees and the air conditioner kicked on. Obviously the electricians were also working under pressure and had wired it backwards. None the less, we made it and turned the system over on time.

Over the weekend, the president of our company had died and Monday was declared a holiday. Due to this, I had worked three holidays a Saturday and a Sunday. The only one happier with my paycheck was the taxman. I appreciated the money almost as much as the satisfaction of pulling off a real coup.

As fate would have it, a little over a year later, the new customer moved the system back to Philly and my claim to fame this time was taking my sleeping bag in to work, so I wouldn't have to commute, the 50 miles from our new home. The boss had refused to allow for motel rooms since it was a local job. The customer was impressed and wrote a letter to my boss, who wasn't as impressed.

What We Really Did

Just as firemen and the Maytag repairman spend most of their time waiting for something to happen, computer engineers spent much of their time waiting for problem calls. However, we weren't allowed to just sit around waiting. Our tasks between emergency calls were very mundane. The most common of these was P.M., Preventative Maintenance. This consisted of an assortment of cleaning and testing procedures. Each piece of gear had a meticulously prepared schedule of things to be done weekly, monthly and quarterly. The more moving parts, the more work involved, card equipment, printers and tape drives heading the list.

Because most systems were leased and legally belonged to the vendor, customers were required to make the equipment available on a weekly basis, so the necessary tasks could be carried out. Many customers resisted, wanting to get the most out of their equipment. A compromise was normally reached, which required the engineers to perform these functions in the wee hours of the night.

One of the sites mentioned most often so far was where I performed every night from four to eight in the morning. I was not forced to do this, but had requested it. If you can remember the mass storage chapter, you'll recall that I started my RAD career with a lucky fix by simply plugging a loose cable back in. The customer was so impressed that he asked I be assigned as site engineer. Of course they expected me to continue performing miracles. Knowing that it would be more a case of hard work than divine intervention I asked for more time to maintain a system that had been neglected by the previous engineer. When the request was made, he said, "Sure, you can have four hours starting at four a.m." So, I had consigned myself to the midnight shift, which was exactly what I wanted since it paid a 15 % differential. At that point in my life I was looking to make every penny possible.

The most important tool used for P.M. was a vacuum cleaner -- not very glamorous, hey? As mentioned earlier, machines used huge amounts of air conditioning, but even this wouldn't work without proper airflow. Each component cage had several rows of modules. At the top and bottom of each cage was a set of small plastic fans (muffin fans). The bottom set had a small filter between it and the outside air. The filters worked well, so well in fact, that if you didn't vacuum them once a week they

would clog and cause the system to overheat, leading to more serious problems.

More than one engineer got in hot water for failing to perform this menial, but necessary, job. Besides vacuuming, there were rollers to be oiled, gears to grease, tapes to adjust and diagnostics to run. Every unit had its own set of specially prepared programs that were designed to validate its reliability. But just like the starship Enterprise in the 25th century, they rarely showed a fault. The easiest way to anger a customer who had reported a problem was to say, "The diagnostics ran O.K." You were required to run them on all the equipment at least once a month. Some of these took just a few minutes and others up to half an hour. It was less work than cleaning and oiling, so some C.E.'s were satisfied to do nothing else.

So now you know the awful truth. It wasn't all heroic trouble shooting and impressive fixes. The worst part was when an engineer would take a system that was running just fine, perform P.M. and the system wouldn't run when he finished. One customer claimed that 50% of his problems occurred during or immediately after P.M. It did happen and probably too frequently.

A particularly bad experience was when I was promoted to lead engineer and given responsibility for several sites, one of which I had no experience with. The first night of P.M., I accompanied the resident engineer to learn more about the system, even though it was the oldest one we had. After we finished the service, the system would not run and I was of little help. The end result was three days of downtime before it was fixed. The following week I foolishly showed up for the P.M. shift and when we finished the system wouldn't run again. This time it took two days to get it back up. The following week I suggested we skip P.M., but the customer demanded that we carry out our contractual responsibility. I declined to attend the next session and avoided that site until I was promoted to systems analyst.

But the worst thing was EO's, engineering orders. Computers, contrary to popular belief, are not perfect and never have been. Even after years in use, we still find faults with their design. So design engineers create fixes to change the logic and correct the problems. Some of these would require removing and replacing the existing wiring, and could take several

hours. To change the circuits on a printed circuit card would require cutting the original etched lines and soldering a wire in its place. This wasn't too bad because you would use a spare card to make the change and if something went wrong you could always return the original card. But backplane wiring was a real nightmare. Try to picture the back of your china closet, with 10 rows of six inch high slots, each row containing 30 slots and each slot having 50 pins sticking out from it. Got that? Now take a giant heap of wire that is as thick as angel hair spaghetti, and as confusing, and press it on to the 15,000 pins. That is what the back of a 70's computer looked like. A piece of paper is given to you with instructions to locate one particular pin and remove one wire using something called a wire wrap tool, which looks like a pencil with a small hook on the end. Then you must locate the other end of the wire and remove it. A sudden pain starts in your stomach when you have removed two ends and realize they are not the same piece of wire, but lets not talk of the macabre. Let's believe we got the right ends and can easily slide the wire out. Next step is to run a new wire using a wire wrap gun, which looks like a phaser from the early Star Trek shows. You place a piece of wire into the barrel, slide the barrel over the correct pin and pull the trigger. The gun tightly wraps the wire around the pin making a secure connection, you hope. Finding the next location you repeat the process and have added a new circuit to the machine. The average EO had about 10 of these in it and you got to do it sometime between midnight and dawn. How scared would you be if you took a functioning machine, spent several hours rewiring it, and then it wouldn't work. I was terrified, and it happened more than once. Now what? It will take longer to remove the "fix?" than it took to install it. If you had made a mistake, it certainly won't have fixed it. I still get queasy just thinking about those times. I hated EOs to pieces.

And then, of course, there was paper work, form after form, all to be filled out by hand. You don't think it was computerized, do you? We were probably the last people to use our own equipment. The hardest thing being to account for the time when you really weren't doing any thing. You couldn't put that down, so there was a constant search for a new category that meant you weren't doing any thing but didn't say so in as many words.

--> Now what did EO stand for?

Not Quite

All of the devices I've mentioned so far, even the slower and seemingly clunky ones, were actually active parts of computer systems at one point or other. But deep in the catacombs of every computer manufacturer are hidden the failures, the devices that were supposed to solve some major problem or open a new era of computing. Some never worked, others were too costly and some just arrived too late. Because the computer components I will describe are an embarrassment to their creators, I will not name the companies or brand names. I have no wish to insult or embarrass anyone or get sued either.

The biggest failure both in size and money to my way of thinking (and I'm sure there are many I am unaware of) was a full computer system that swallowed millions of dollars in R&D funds. It had been sold to several customers, but never left the factory floor. What became a nightmare to the developers was caused by one weak point: the cooling system. It was water-cooled. Yes, H2O! Water in the middle of all that electronics. The need for this was brought about because of the density of the circuitry. In an attempt to speed up the processing the processors were tightly packed together. Greatly reducing the number of connecting wires and resulting in a much shorter electronic path. It's still hard to believe even with the speed of computers today that speed can be gained by moving components closer to each other, but if you multiply the huge number of components by even the tiniest amount you soon have a considerable distance and thus a measurable savings in processing time. Initially, normal air conditioning and forced air were used. But due to the closeness of everything there was insufficient flow and component failure became excessive. So some bright spark came up with water-cooling. The person was obviously familiar with machine guns. When air cooled machine guns burned out their barrels from rapid firing, the military put water jackets around the barrels to keep them cool. If it worked for machine guns why not computers?

So pipes were run over under and around the circuit cages, attached to an external water chiller and on came the water. Well, this immediately brought forth a whole bunch of plumber jokes and cartoons, like fish in the mainframe, soon decorated many bulletin boards. Funny to some, but not to the engineers. And as things go, solving one problem always leads to another. The high humidity generated by the cool pipes led to rapid build up of condensation and thus to corrosion on the

connectors causing more failures. This lead to the introduction of a dehumidifier. Onward marched the re-engineering. Algae built up in the pipes, which restricted water flow, becoming another obstacle. So now algae cleaner, like the stuff you put in your aquarium, was added. This stuff ate the algae all right and then started on the joint sealer. No one had planned for the introduction of chemicals into the system. It doesn't take much imagination to picture what leaking pipes did to the circuitry let alone call for the addition of a mop to the engineer's toolbox.

All of this home improvement type engineering delayed the development so long that the speed gains hoped for had been surpassed by the development of faster chips. Thus, the whole idea of tightly packed circuits was obsolete before it was ever born.

If I gave you the name of the company, you would know where you could pickup a bargain on some pipe, coolers and dehumidifiers, which are probably still in the back of some warehouse.

The preceding example was not a very visible failure, you couldn't see the components overheating or the connectors corroding. On the other hand the most visible failure I witnessed was a thin film storage system. In the search to develop economical mass storage one company tried using a photographic style process. Data was imaged and then developed on long strips of stiff negatives and stored in a specially designed cabinet. The strips were six inches wide and three feet long. Each could contain about one megabyte of data. This type of storage was only applicable for archival information, which did not require frequent updating. The film was a permanent record and could not be change without producing a new negative, a slow and expensive process. But banks and insurance companies maintain huge amounts of customer data, which is fairly stable, and, thus, provide a suitable market place.

The cabinet contained almost no moving parts, could hold hundreds of strips and was much cheaper than RADS or Disks. The data was therefore considered less volatile. In theory anyway.

The cabinet had no electronic connection to the computer but was positioned in front of the film reader that was attached to the computer. The writing or etching as it was called took place somewhere else, either an off site photo lab or a central

facility like a national or regional office. That was the plan. This plan would provide for a copy to be made and held in a safe place like a vault or fireproof warehouse. Security always being of great interest to financial organizations.

The overall concept appealed to several companies. They could store large amounts of data cheaply and safely with easy access. The retrieval of one strip was slower than disk access but once you had a strip in the reader you had access to lots of data, just by sliding the strip up and down past the optical read head. The design engineers having made the decision to limit the number of moving parts to improve reliability. This was fine for the reader but put all the movement on the strip. The whole operation depended on the rigidity of the film. To retrieve a strip an arm was guided over the cabinet and selected a strip based on the address received from the computer that indicated the physical location of the strip. The cabinet placement had to be exact and remain in the same place. Once the arm located the right strip it would latch onto it, extract it from the cabinet and swing it over to the reader. Since the strip was three feet long the cabinet was over three feet high thus the arm had to extend like seven feet above the floor when it removed the strip it then had to rotate to position the strip above the mouth of the reader. There was no way to prevent the strip from swaying as it was moved from one place to another. To compensate for this the reader had large lips like a big fish to ease the film into position. The read process could then commence. The return trip, however, was a little more precarious. Because the strip would heat up as it was moved rapidly up and down in the reader, it became less rigid and since the slots on the cabinet did not have large lips the strips would miss the slot and be bent in half as the arm was lowered. This situation occurred more on the ends of the cabinet as the angle of the arm got further from the center. The cabinet contained no electronics and had no way of sensing where the film was. The arm couldn't tell if it missed the slot, so it kept lowering the strip until it assumed it was safely in the cabinet it would then release the strip, and go in search of the next one. The results varied: some strips snapped in two, others were launched across the room and some just fell over winding up on the floor or on top of the cabinet where they blocked the slots of other strips, eventually totally frustrating the arm's attempts to retrieve and return other strips. Normally this situation was avoided because someone would see the problem and abort the operation. But on those occasions when no one was around, the results were

disastrous to some and funny to others, depending on whether or not you were part of the development team. Initially, they tried adjustments and refinements to avoid missing the slot, but short of redesigning the cabinet, making it very expensive in the process, there was nothing that could save the project. The decision was finally made to scrap the whole idea. After the death sentence, the two proto-types were still on the floor and people would amuse themselves seeing how far they could launch a strip, possibly the first true interactive computer game "launch strip and duck."

The saving grace about both of these examples is that they never made it to the outside world. There are others I can think of that did and shouldn't have. The result being a lot of downtime to the customer and overtime to the customer engineers. One disk system I know of was so bad the company put a spare unit at each site so the customer could switch in the spare while the c.e. worked on one of the others. When this proved too expensive, the units were withdrawn and replaced with another vendor's equipment.

But when you consider the amount of recalls by the automotive industry, computers haven't done so badly.

The End Was Near

In July 1975, I thought everything was perfect. My wife and I had just bought a home and were awaiting settlement. I was off to learn a new operating system, continuing to expand my knowledge base. By the end of the first day of classes my world was in a shambles. No, I didn't flunk out. Something even worse happened. As I entered the hotel, where I and the rest of the class were staying, the branch sales manager, who worked in the same office as I, aggressively motioned me over to the phone booth in which he was having an animated discussion with whomever was on the other end. He cupped the receiver with his hand and breathlessly announced, "Xerox has gotten out of the computer business." There was no doubting his sincerity. This was no joke. My wonderful career at the company I loved was over. Questions leaped to my brain. Was I unemployed? Would I still get paid? Should I stay in class? Would there still be a class?

That evening was one of remorse and moaning. As word spread, more and more Xeroids gathered in the bar. The higher the position held the deeper the gloom. Sales managers were already looking for work; design engineers were cleaning out their desks. The two instructors teaching my class were on the phone to California seeking advice. Then someone told me something very interesting, and for the time being, relieving. "The only safe ones are the Printing Systems people." Hey that was me, I was still officially a Printing Systems Analyst. I did very little work in that area but that was still my title. I was safe. I had a job.

The next morning was still chaotic but we slowly regrouped. The class continued and upon its completion I returned to Philly. We finalized the deal on our new home and life went on.

In September, believing things had settled down, I took a vacation. On my return I was met at the office door by a colleague from the printing sales department. "You've been transferred to sales, you and I will be working together." He happily announced, as he showed me a twix (Teletype message) stating that all PS analysts were transferred to sales immediately. Before I had a chance to digest this, my boss saw me and called me into her office. "You've been transferred to the computer division," she said with a smile. Whoa! What's going on here? I'm being transferred left and right and nobody's asking me if that's what I want. My protests went

unheeded and I received a call from the new computer division regional manager congratulating me on my transfer and unpaid promotion.

It took a couple years before I got to the bottom of this mystery. A computer analyst in Washington D.C. was married to a printing systems sales manager. She and I actually changed jobs. Of course, she was the only one that knew what was happening.

A short time later it was announced that the entire computer division was to be transferred to Honeywell -- like it or not. My personnel file was certainly on the move.

Many of the engineers I worked with were not happy about the situation. The word quickly spread that Honeywell was a cheap company with poor benefits. And when they announced we would receive a three- percent raise to make up for the benefit short fall, we knew it wasn't out of generosity.

Being one of the senior guys, 34, the engineers sought me out for advice. We decided to have a secret meeting at a nearby restaurant. Spirits ran high and they wanted me to approach management with their protests. I felt equally strong about it having been sold down the river the way I was. I was willing to take on the role of ringleader, but not sacrificial lamb. I agreed to approach management; however, only if everyone gave me their unsigned offer letter from Honeywell which had to be signed by all employees before the deal between Xerox and Honeywell could be finalized. If one didn't sign the offer he had to quit, no benefits no unemployment. Only two other guys gave me their letters so our protest died stillborn.

The ironic thing was that when I reported to Honeywell I found two of my former RCA classmates there. They had accepted jobs at G.E. at the time I had joined Univac. And when G.E. got out of the computer business they wound up at Honeywell. So it was inevitable I would wind up there. I however was at a much higher pay level than both of them as were most of the Xerox guys.

Asked for further advice by my Xerox compatriots, I said, "Give it a year." At the end of one year I was looking for a job.

I was offered a job by one of my customers but turned it down because I felt it would be too confining and also expected to

get an offer from Hewlett Packard. All the interviews had gone well and the manager promised me an offer letter as soon as he cleared up some other personnel matters. The week the offer was supposed to be made, his wife became seriously ill and the offer was delayed. Eventually after the wife's illness lingered, the manager was replaced by a new guy with different ideas. He wanted new college graduates, who could be hired for a lot less than people with 10 years experience.

A while later I contacted Xerox, having been urged on by some of my old friends still there. But the then sales manager told me she wasn't interested in going against company policy since Xerox had agreed not to hire people away from Honeywell with out their permission. This was obviously illegal and there had been some law suits that were normally settled in the employee's favor. But it was only worth doing if the manager was really on your side.

Opening a business on my own was another alternative. After exploring a plastic surfaced ice skating rink and the then revolutionary idea of a computer store (this was 1978), the idea of being my own boss was not that appealing. So I muddled along.

Honeywell wanted to keep as many Xerox people as possible, mainly to keep the existing customer base happy. Honeywell had promised to develop a new operating system based on their hardware that would allow for easy migration from Xerox hardware. Changes to the existing operating system soon dried up, making our jobs less and less interesting. The number of old Xeroids continued to dwindle and the management style continued to be more oppressive.

In February 1979 I was contacted by Xerox again. A female analyst whom I had worked with before was leaving to start a family. The new sales manager had asked her to find a replacement before she left, so she called me. At the meeting with the new sales manager an agreement was reach that the job he had to offer was not what I wanted. He wanted a person to make sales pitches and believed there would be very little technical support required. I could not see myself doing sales calls, getting away from hands-on was not interesting. We parted on good terms both thinking that was the end of it.

Wrong! Chronologically the following things took place, some of them without my knowledge. In June, the first of the new Xerox

printers were delivered. In July, the technical problems were swamping the sales oriented support staff. In August, I received calls from two different places in Xerox, sales and support. The new printers had been promoted as plug and play, which meant they would be very easy to install and maintain. They did not and, since most of the technical people had been told they wouldn't be needed, and had left. Xerox needed help fast.

In September, after obtaining a letter of permission from Honeywell, I accepted a position as sales analyst at Xerox. The promise of 20 hours a week overtime had the opposite effect from what the Xerox service manager had planned. At 38 I had better plans for that 20 hours than spending them in computer rooms. So my career as a fixer officially came to an end and my sales career started. I would still do some fixing and experience the most rewarding 10 years of my career. But as I've said before, that's another story.

You Want To Hear It???