The Analytical Engine
JOURNAL OF THE COMPUTER HISTORY ASSOCIATION OF CALIFORNIA

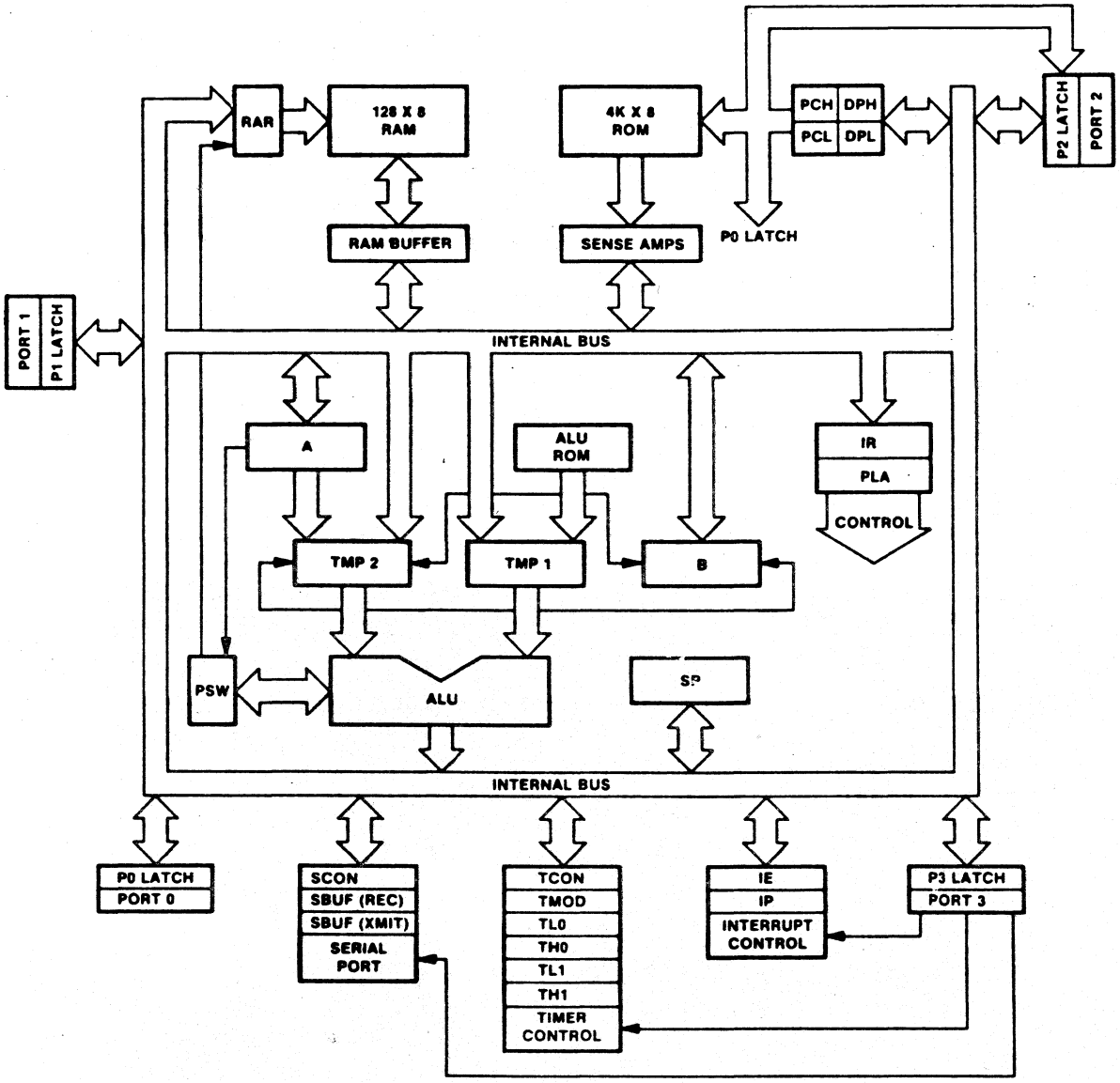



**Figure 3. Block Diagram of 8051 Internal Structure**

Winter 1997 Volume 4.1

# The Analytical Engine

JOURNAL OF THE COMPUTER HISTORY ASSOCIATION OF CALIFORNIA

## Editorial: PULLING TOGETHER

Since the fall of 1996, we've had several occasions to publicize new initiatives in technical history in the Bay Area. The Computer Museum History Center is making steady progress towards its interim goal of a fully organized and inventoried collection. The San Francisco Computer Museum is raising funds and negotiating for an exhibit site. The Perham Foundation, longtime custodian of the history of broadcasting in Northern California, looks forward to the construction of a museum at Kelley Park in San Jose. The Intel Museum continues to be an unpretentious miracle, as both education and entertainment; and the well-regarded Hewlett-Packard Archive sets the local standard for curatorship. As for other local collections formerly on display, especially the collection of the Lawrence Livermore National Laboratories, we can at least say that they're safe in good hands, even if they're not open to the public at the moment.

Okay, *what next?*

Everything in the CHAC's experience says that the best progress has been made—and will continue to be made—through collaboration. Our efforts at collecting have been made much easier by the cooperation and generosity of the Perham Foundation; we hope that we, in turn, have been a ready resource for Perham while they organize and document their own collection. Through TCMHC we have met many of the South Bay's other technical historians; through SFCM we have contacted several members of the Macintosh community in the East Bay, whose knowledge is steadily more important to us as the CHAC builds up a world-class collection of Apple artifacts and manuals. *Cooperation counts. Collaboration works.*

The institutions that safeguard technical history in Northern California collectively represent really formidable experience and intellectual strength. Each of them makes a unique contribution to the general purpose. But if what we want is to make the Bay Area into a world-class resource for technical history—while we attract a few more tourist dollars into the region—*we will all be pouring energy into this creation.* And the sooner we start, the better off we'll be.

So far, most of the cooperation between institutions has been accomplished on a handshake. From now on, we hope, arrangements to combine our planning and effort will be more global and more formal. In any case, the ENGINE will keep you posted, but be sure: The great days are coming.

## TAKING THE HEAT

During this summer, and we hope before the *end* of the summer, tactical teams under the leadership of Edwin El-Kareh and Frank McConnell will complete the rescue of the vast collection of Apple (and other) hardware, software, manuals and ephemera now sitting in a disorganized warehouse outside Sacramento. When we're finished, the CHAC will probably—proudly—own the best collection of Apple-related material in captivity, outside Apple itself. The collection shows particular strength in Lisas and early Macs, and includes a lot of material that will be a gold mine to restorers and hobbyists.

Mundane aspects of this rescue are real stoppers. The round trip from the South Bay to the site is over 200 miles; it beats up vehicles and, worse, burns out volunteers. If we try to accelerate the process, we'll quickly end up spending more money than we have; if we stick to our present pace, we may spend a fortune in warehouse rent. All we know is that, the more money we have *now*, the more effectively we can put it to use.

This rescue is the CHAC's biggest project in almost two years—and bigger, in ways, than the fabled rescue of the SDS 930. Please help by *donating today* and making this phenomenal effort possible. See the box on page 38 for our postal and electronic addresses....and thanks!

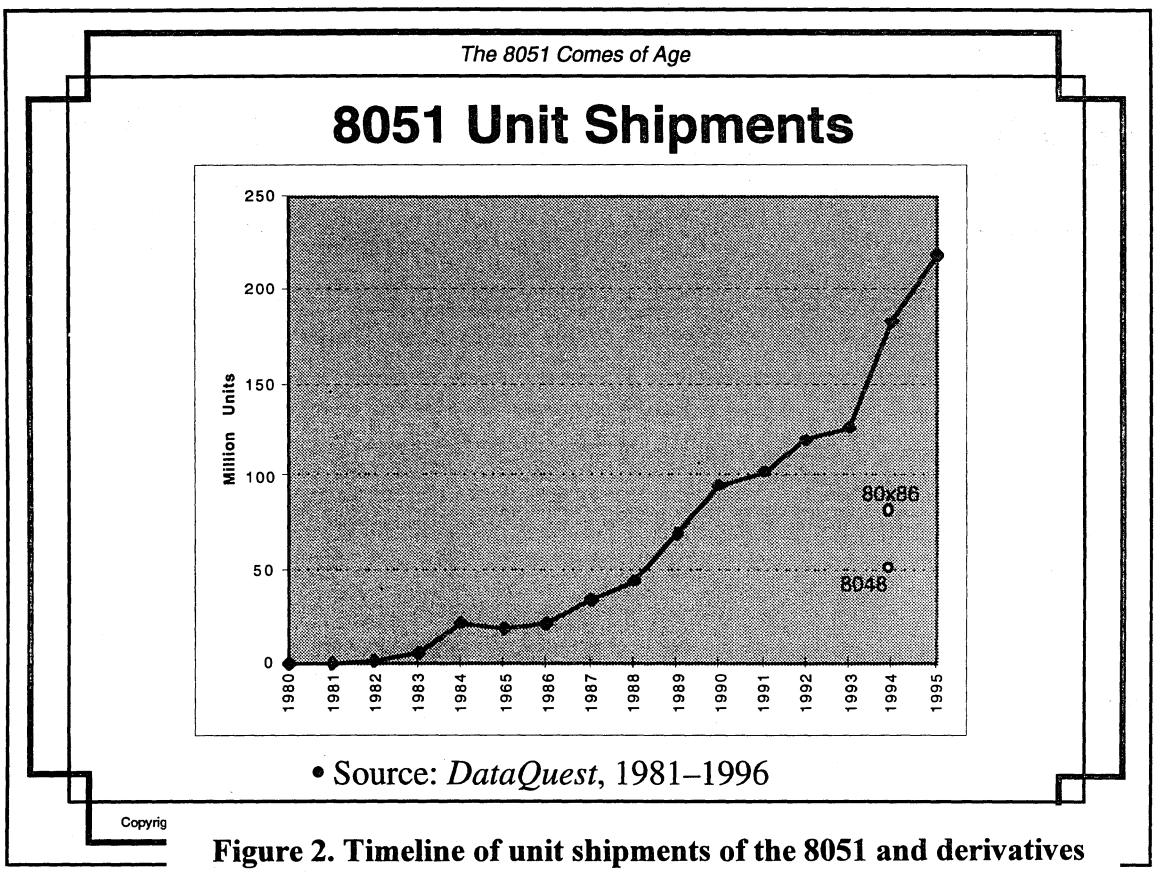



**Figure 2. Timeline of unit shipments of the 8051 and derivatives**

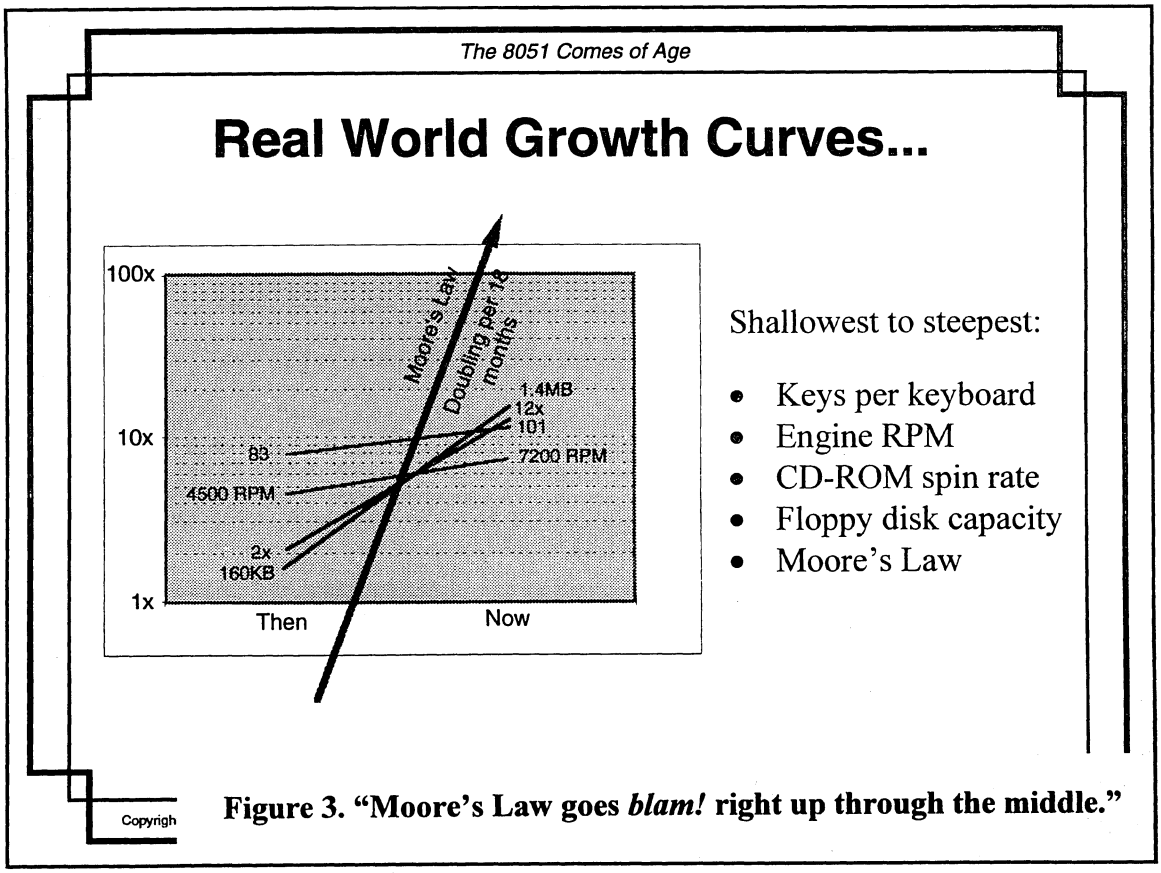



**Figure 3. "Moore's Law goes *blam!* right up through the middle."**

# SLIGHTLY MORE NUMEROUS THAN PEOPLE:
## A Quarter-Century of Microcontrollers

### An Interview with John Wharton

*KC:* Let's begin by talking about how Intel got into the microcontroller business. It's my understanding that the 4004, which was the first microprocessor, was also the ancestor of all Intel microcontrollers.

*JW:* You could say that the 4004 was the ancestor of all microprocessors, microcontrollers, DSP chips and microengines in general. In the years after the 4004 came out, the concept of the microengine evolved in several different directions, one of which led to the market for microcontrollers—or embedded controllers. But the link with the 4004 in particular is unusually direct, because the 4004 was designed for and primarily used in this embedded market. It's important to remember that, when the earliest microprocessors were developed, the desktop computer as a paradigm leading to a market didn't exist. Even when the 8086 was being designed [Intel's first 16-bit microprocessor, in 1976-77,] the target market was not desktop computing as we know it now; it was industrial automation, machine control systems, intelligent controllers for factories, for which the advantage was that the 8086 could provide sixteen or more virtual machines, each of which was comparable to an 8080.

### 8048: AN ERA BEGINS

*KC:* So, what was the first chip that Intel designed specifically as a microcontroller? Was it the 8048?

*JW:* The terminology, markets and products developed in an intertwined fashion. The 4004—if you look at the old ads from 25 years ago—was promoted as a logic replacement, as a way of simplifying hardwired controllers for things like postage scales, traffic light switches, security systems, and gasoline pumps; it was intended to inspire new design techniques for implementing circuits used to control a wide variety of machinery or appliances. The 4004 was targeting that market from day one, as were the 8008, the 8080, the 8085, and even the 8086, but they weren't microcontrollers by today's definition.

Today when we think of microcontrollers we envision a highly integrated single-chip system, a complete microcomputer that includes the processor, the input/output, the memory, the timing, control and interrupt circuitry, counters, serial ports, and numerous other things, all rolled into one die—one microchip. The first Intel chip that aspired to system-level integration on a single die was the 8048, introduced in 1976-1977.

*KC:* What were the characteristics of that chip?

*JW:* Let's back up a little bit. Intel came of age, in a way, with an 8-bit microprocessor called the 8080. This was a gratifyingly popular processor, but to build a system using it you needed to start with the 8080 itself as the "brain" and surround it with easily a dozen chips to get it to function at all—one that handled the timing circuitry, one that created control signals that would go elsewhere, one that would handle interrupt requests, one containing program memory, one containing data memory, one to perform input operations, one to perform output operations. An 8080 in working form would cover a board at least six by six inches, maybe six inches by a foot.

*KC:* This design paradigm had begun with the 4004, which was one chip of a four-chip set?

*JW:* Exactly—one processor, which was the 4004 itself, one chip to handle program memory, one to handle data memory and one to handle I/O. The 4004 actually had a few gimmicks to increase the integration level, in that some of those chips could do multiple functions, that then vanished and didn't resurface for six or eight years. But still, a working 4004-based test computer would occupy perhaps a quarter to a third of a square foot of circuit board.

Furthermore, with regard to use in embedded systems, this wasn't the only complexity or even the most significant one. Every Intel microprocessor prior to the 8085 and the 8048 required multiple power supplies—power voltages—which was an inheritance of development from multichip systems. To design dense logic for a microprocessor, you would employ one set of design rules. Design of a program memory chip would imply an entirely different process that perhaps used different voltages, and design of a data memory chip according to entirely different parameters again. These wafers would be run through different factories,

sometimes in different states or even continents, using different process technologies. All these chips, manufactured separately, would have as a common denominator only the circuit board you were soldering them to.

When I was in college we were carefully taught that there was no way you could ever produce a single-chip computer because of the fundamental inherent differences in process technology. In the design stage of the 8048 Intel had determined that that wasn't true; that a single process technology, a single wafer, could accommodate logic circuitry, program memory, data memory, and I/O drivers. Now, it would not perform all of these functions optimally—or even any of them optimally—but the central implication was written in letters of fire: We can create a single die that does it all, *if* we're careful not to ask too much of the design. So the designers made certain compromises. They cut corners to force the circuitry to be as trivial as possible. There wasn't a whole lot of program or a whole lot of data. Even so this was challenging from a lot of different perspectives, especially in terms of circuitry being crammed onto one device. But the results, the 8048 and its little brother, the 8021, worked perfectly for an astounding spectrum of applications involving relatively small sets of functions that weren't especially time-critical. The 8048 was a salute to the creativity of the device physicists, a miracle. It's like the joke about the dancing pigs—what's amazing is not that they can do it very well, but that they can do it at all.

The 8085 and the 8048 were Intel's first generation of processors that ran on the standard voltages of 0 and +5 volts. Everything prior to that had required dual supplies at least. Being able to run on a single voltage was very appealing to markets which were cost-sensitive and wanted to micro-control something that didn't have lots of different power supplies available to it. Think of the engine of a car, which in theory is a steady-state 12 volt system. Actually it's anywhere from 6 or 7 volts if you're trying to start the engine when the battery is cold, maybe 8 volts if the battery is low, to as much as 14 or 15 volts when the battery is charging, or much higher in surges introduced by the starter motor. But whatever a car battery is putting out, it's very easy to stick a regulator in there and bring the voltage down to +5. Contrast that with trying to tell Ford that every car would now need two extra little batteries in order to have a computerized dashboard.

*KC:* Yeah, they wouldn't like that at all.

## CARBURETOR CONTROL

*JW:* When I worked with Ford on the feedback carburetor—the EEC-3 product, which was the first important commercial application of the 8048—I met an engineer who worked for Philco-Ford, the Ford electronics subsidiary. He had built a career on trying to make car radios cheaper; and he explained to me that if Ford is making 10 million car radios a year, and each year he can eliminate one more penny from the cost of building a radio, he's justified his own salary. 10 million units times a penny is $100,000, which more than pays the cost of keeping him on the payroll. It's funny—we think of cars as being expensive things, and yet the car manufacturer's perspective would call it success if over the space of 10 years you managed to make the car radio a dime more cheaply. Imagine trying to persuade such a market to install computers under their dashboard that included extra power supplies or batteries. I think that would be a very hard sell.

*KC:* In the absence of external constraints, certainly. But just at that time, the government was saying to the manufacturers that okay, now you will need to control your input and output mixtures much more carefully than was hitherto the case. Sorry, guys, the only way you're going to do it is with a microcontroller.

*JW:* Government-mandated emissions control was indeed Intel's foot in the door in the automotive marketplace. The 8048 became Ford's initial application of microprocessor technology in the automotive world, for a feedback carburetor system; and it solved the problem that there was no way to tune a car for all situations, all speeds, all air pressures, all humidities, all temperatures, fuel grades, oxygen levels—

*KC:* Because to tune a car is to set a fixed series of parameters within the engine and say "Okay, I hope this works adequately over the broadest possible spectrum of conditions." The microcontroller changed all that by saying "When the conditions are *x*, the response of the engine can be *x*-prime, and when the conditions are *y* the response can be *y*-prime."

*JW:* Exactly. When the car is moving fast you can have one set of parameters. When the car is moving slowly you can have a different set. The same engine in different cars can have different sets of parameters. But the next level of refinement was to make the whole process dynamic. By putting sensors into the exhaust gas stream and measuring its oxygen level, we created the possibility of fine-tuning on the fly. The oxygen level is too high, that means we could be burning a little more fuel efficiently....the oxygen level is too low, we've been burning too much fuel, we should turn the ratio back down a little bit.

*KC:* So that the engine's microcontroller actually listened to the exhaust composition?

*JW:* Yes. That gimmick was the 8048's entry into the automotive market and I think it was probably the first very-high-volume sale. In those days, high volume to Intel meant millions of units. The Ford carburetor contract was on the order of four million units spread over a couple of model years, at a time when Intel found it absolutely incredible to sell four million of anything. It was considered a real coup.

*KC:* What was the approximate unit price?

*JW:* I'm guessing right now, but I'd put it somewhere in the five- to seven-dollar range for the processor itself and then a few dollars for the circuitry pertinent to the rest of the computer.

*KC:* For Intel that was perhaps between 20 and 40 million dollars right in the door—

*JW:* That's probably the order of magnitude, yes.

*KC:*—which must have been very gratifying at that time in history.

*JW:* It no doubt helped pay the bills while they were waiting for the 8086 market to take off! I think that—going back to the gentleman who saves a penny from radios—Intel figured that I had justified my salary by managing to get that deal done. At that point Ford had actually selected and designed in another processor, a competing product.

*KC:* Which product?

*JW:* The chip was a 3870, which was a single-chip implementation of the Fairchild F-8, I think by Mostek. There may have been a couple of different vendors supplying the same processor. That was another chip in the same time frame that was accomplishing what the 8048 accomplished, and Intel saw it as primary competition. Strategically it was very important to displace that design and install an Intel chip in its place.

*KC:* Why was the 8048 found superior to the 3870?

*JW:* Why did Ford find it superior? I'm sure a big part of it was that Intel was seen as a bigger company. Intel had perhaps more track record of selling processors in general. Intel was willing to work with Ford very closely on defining future products that would fit Ford's needs more precisely. Intel is a very far-seeing company, and when they decided they wanted Ford's business, the attack was probably at least three-pronged. The first was to show them that, technically, the chip would work. The second was to reassure them that Intel would give them the support other companies wouldn't. The third was to demonstrate an advantage in doing long-term business with Intel.

My task, in pursuit of the first prong, was to make sure that the technical issues could go away and that Ford would have confidence in the chip, which required a couple of months of engineering effort and a couple of trips to Detroit. I totally destroyed a pair of Birkenstocks showing up in Detroit in mid-winter, not remembering to bring real shoes, and walking through the salty slush of the parking lot getting in and out of the Ford building.

*KC:* Be glad you didn't destroy your feet. Now the 8048 *in situ*, in the feedback carburetor, was located where?

*JW:* Actually in the engine compartment. It was put in a little aluminum box, probably about the size of this tape recorder, bolted to the front surface of the firewall, if I recall correctly.

*KC:* Was there any problem with, for example, temperature range?

*JW:* There were a lot of problems, and much of Ford's concern was with the environment in which the device would operate. Microprocessor engineers think in terms of everything running at 72°, plugged into a wall with steady AC current, and if the system malfunctions you reset it, and if all else fails, well, you power it down from time to time anyway.

*KC:* But in an engine compartment running at anything from subzero to well over 100°—

*JW:* I grew up in Wisconsin. I know what it's like to go to your car and not be able to get the key into the lock, because the tumblers are frozen in place. Or not be able to shift gears on a manual transmission because the grease in the case has solidified. And Ford was saying, not unreasonably but with great insistence, that regardless of conditions, it's not okay to be driving down the highway and suddenly have your engine turn off.

*KC:* So, you have this tremendous temperature range; you're trying to cope with everything from 6 to about 14 volts from the battery; and you don't have reset capability.

*JW:* That's the summary of the challenge.

*KC:* How was this challenge coped with?

*JW:* Well, the customer had defined pretty well what the chip had to do, what the sensors were, how to distinguish between different input states, and how to respond accordingly. All of this was in the specification for the module. So the first step was to design hardware that would drive their actuators appropriately, and develop software that would implement the algorithms to drive them. At that point we flew out—that's the imperial "we," I suppose I flew out—to Ford to demonstrate this, and make sure it was doing what they wanted it to do.

But one of the rules of car design is that you can't have single-point failures—or if you do, they have to be few in number and very well constrained. Ford shies away from anything that would cause a car to burst into flames, for instance. So standard procedure for electronic stuff is to take some system, abuse it until it fails, and then see what happens to the rest of the vehicle. And what's desirable is to have the product malfunction, recover gracefully on its own, and continue with what it was doing.

## LIGHTNING STRIKES

*KC:* Did the 8048 in fact do that?

*JW:* Were we able to induce failures? Oh, yes, very easily. Part of their test was to introduce the chip into a very noisy environment, and it's easy to create a high level of electrical noise with spark coils and spark gaps and things of that sort. They had a huge ignition coil, about the size of a Quaker Oats box, with a heavy insulated cable and an adjustable spark gap at the end. Normal automotive spark plug gap is what, a sixteenth, eighth of an inch? This spark gap could be opened to half an inch or two inches wide, until the thing would shoot lightning around their test bench. To induce failures, all you had to do was wave the spark gap in the vicinity of the circuit board and something would go out to lunch, die, stop, malfunction, start twittering. We replied with hardware solutions—bigger capacitors, isolation, ferrite beads on the cables. There are a lot of things you can do that will increase resistance to electrical noise. The spark wasn't doing physical damage. We were pretty consistently able to kill this.

Their answer was simply to make the spark bigger and more powerful and get it closer. They never hit the chip itself with a lightning bolt, but they were able to make the program malfunction repeatedly. So it was back to Santa Clara to try to figure out what was going on and how to handle the software malfunctions. I don't know if we ever really had direct evidence, since it's awfully hard to see what's going on inside a chip, but one of our hunches was that the program counter was being corrupted, causing execution to suddenly jump around in the program. This was disappointing because, obviously, software can't continue to run if you start executing instructions at random.

*KC:* but did it recover appropriately?

*JW:* The original program, no, because we didn't realize that was a design constraint. The description of the algorithms didn't include this provision for inducing failures and then seeing how it responded. But the 8048 had 1,024 bytes of program space on-chip, and the average instruction was about twelve bits, so on the order of 700 instructions could be implemented in one device. Initially we were using about half of that space, certainly not much more than that, to do all of the algorithms that Ford had requested. We had room for innovation.

We took the existing program and re-implemented it in a modular fashion. The foreground—interrupt-driven—portion was actually controlling the actuators, while the background portion ran to monitor the inputs and so forth. In the redesigned program, the two different sections of software were each evaluating the progress of the other; the background program could monitor the interrupt portion to make sure the interrupts were continu-

ing to run properly, while the interrupt routine was able to monitor the background program to make sure that it was continuing to progress in legitimate program space. We filled the remaining space with error-checking and recovery mechanisms such that, whenever there was extra time, the program would make sure the software was still operating the way it was supposed to. And at any time, if anybody was unhappy with the machine state, a software mechanism could boot the processor, reinitialize variables, reset in exactly the fashion of a hardware reset, and continue the program from the beginning. The effect might have been that, for a second or two, the engine might have run a bit roughly with an imperfect mixture. Within several seconds the fuel/air mixture would be optimal again, from the driver's perspective that probably wouldn't be noticeable, and for sure the engine wouldn't shut down.

*KC:* And this was only in the case where you were throwing electrical noise from two-inch sparks at it, which is not entirely a real-world condition.

*JW:* It was very unrealistic, I thought. After one round of hardware fixes that made it particularly noise-immune, they sealed up the box that contained the microprocessor and actually ran the spark gap onto the aluminum shell, at whatever the equivalent of engine speed was. We're talking tens or hundreds of sparks per second against this thing.

*KC:* What was the box like?

*JW:* My recollection is of a metal pan, with the circuit board inside it, and the whole thing potted in some acrylic-epoxy mix and bolted against the front of the firewall. They turned it upside down and hit the metal pan with lightning directly from the spark generator. It was frightening, seeing the poor little chip being abused by this.

*KC:* Now, even if this was a real-world condition, it was not something that would occur repeatedly in a well-functioning system.

*JW:* Ford had a very expansive notion of the real world. They argued, for instance, that you might be driving down the highway and have one of your spark plug cables disconnect from the spark plug and flop around inside the engine, back to the point where it's in physical contact with the engine controller. Then it would have to tolerate the sparks to the shell.

*KC:* Any driver with any proportion of awareness, upon losing a spark plug lead, would realize that his firing order had gone so wonky that you would think he would pull over and stop.

*JW:* You would think so, but probability wasn't the primary concern. The overriding goal here was to do due diligence, and not intentionally go into production with something that might conceivably fail. Partly they just wanted to be able to say that they had covered their bases.

## KEY QUESTIONS

*KC:* Okay, that was how Intel got its start in the automotive industry. What were some other early adoptions of the 8048?

*JW:* There was a company in Illinois, Microswitch I think it was, that built a keyboard. Traditionally keyboards were fairly simple things used only on things like electric typewriters and CRT terminals. Now, an 80-character CRT used to cost $2,000 for the terminal and the keyboard, so there was room in there for development expense. A few companies developed keyboard controller chips, 24-pin packages with custom algorithms inside, which when connected to an array of keys would serve the function of being a keyboard controller.

That turned out to be one market the 8048 was ideally suited for. It had enough program space to do basic keyboard scanning and encoding functions, with enough left over to hold translation tables which would allow up to 126 or 127 keys; each key position could return up to four different values, which were typically character, character+shift, character+control, or character+alternate. Naturally, permutations of those control keys and the character could report a different 8-bit code for each case. Any key could be auto-repeating or not repeating, and you could have multiple-key rollover up to about eight keys down. Multiple keys down would be sensed as transitions, and each key would be reported to some host processor as a bit-parallel value with strobe values, saying in effect "Here's the 8-bit code and here's a strobe to tell you it's valid," as well as a serial connection using—if I remember—300 baud standard ASCII format. We developed this as a favor for Microswitch; they adopted the chip and started using it in a line of custom computer control keyboards. Intel then adopted the same technique for a development

system that it was working on at that time, and other keyboard companies followed suit. The 8048 seemed to be perfectly suited to that in terms of speed, of on-chip data memory, on-chip program memory, number of I/O pins.

*KC:* That raises a point. How many pins was the package of the 8048?

*JW:* 40 pins.

*KC:* How did the 8048 then manage to be less expensive than the 24-pin dedicated keyboard controllers?

*JW:* It wasn't necessarily less expensive, but it was far more flexible. The dedicated 24-pin controllers were hard-programmed inside for particular functions and key layouts. If you as a manufacturer committed to these chips, you had to have keyboards with the following keys in the following patterns, and there was no way to distinguish products; different vendors using the same controller would necessarily produce keyboards with the same number of keys and the same encodings. No gimmicks, no features, no bullets on a data sheet—no sizzle.

By moving to custom ROM code in an 8048 you were able to expand the character set, change the character positions, and accommodate different types of technology—simple contacts with diodes, for example, or Hall-effect magnetic sensors. If the 8048 cost $7 and the TI dedicated chip cost $3, that extra four bucks represented good value in the context of the full unit cost. By the time you installed 128 keyswitches, even if each switch was only 20 cents, you were already spending over $25 just on the switches. And keyboards were, or were part of, very high-margin products.

*KC:* Right. I remember that the first microcomputer keyboard that I purchased as a separate part was $350, and that was in late 1983 or early 1984.

*JW:* So spend four more dollars to get the 8048, and you can have more bullets on your data sheet, different features, this year's keyboard looks different than last year's keyboard. An embedded keyboard controller became widely accepted, and you'll notice that the IBM PC—which was no one's definition of a daring machine—used a dedicated controller to do the scanning and encoding functions and report the key detections back through a serial cable.

When the IBM PC came out, it was quite outlandish for the keyboard to be detachable. That was not standard for the day. CRT's often had the keyboard hardwired or built into the same shell, or there might be a ribbon cable from the keyboard back into the guts of the machine.

## ALLURE OF THE PLASTIC PACKAGE

*KC:* So that was pretty much the story of the 8048, those two applications?

*JW:* Those were two that I was involved in developing, but by no means the whole story. It was also used in things like thermostats and little matrix printers, in cash registers, in telephones. I think one plotter used it. I'm sure there were many hundreds of applications at the time—most of them done primarily by our customers. Within Intel there was a decision that the 8048 was to be promoted by targeting key accounts, then doing the engineering work that would allow these accounts to adopt the product. Intel through its Strategic Design group, which I was part of for about a year, started with standard chips and completed the package with custom firmware. But the client in effect received custom chips, received the added value of chips tailored to specific applications. Thus the carburetor and the keyboard controller.

*KC:* And Intel, at or near this point, must have realized that given the breadth of application and the potential longevity of some of these applications, the 40-pin microcontroller promised to be a cash cow of rare order.

*JW:* I think what they realized was the allure of reduced complexity. What Ford was buying from Intel wasn't a computer; it wasn't even a microprocessor. It was a 40-pin plastic package that would make their carburetor work. What Microswitch was buying from Intel wasn't a computer; it was a 40-pin keyboard controller chip. If anybody changed their mind about how this 40-pin plastic package should work, in short order there could be a slightly different package that filled the new requirement.

That this was a microprocessor was incidental—not to Intel, but to the customer, who cared about flexibility, versatility, and quick turnaround. Texas Instruments and Mostek were selling processors and redesigning logic. Intel was tweaking firmware and selling plastic packages. Intel's deciding genius

here, if you will, was to realize—I think before its competitors did—exactly how generic a microcontroller could be. Everything else was margin.

*KC:* And the potential in customizing and re-customizing the 8048 was making the materials guys real happy.

*JW:* It was making the customers happy. From their perspective this was now a universal chip. It was easy to design into things. In this way it followed Intel chips like the 4040 that had been adopted by technologically creative companies—startups—who wanted to get a competitive edge in their markets through innovation. And it blew those markets wide open.

The 8048 was irresistible to a postage or a grocery scale company that craved an edge over competing postage or grocery scale companies. The application of the microcontroller would go beyond the electronic sensing mechanism for the scale, although that was interesting and had its own merit. A microcontrolled scale could do postage computations, in which the user would enter the zone—

*KC:* Enter the postage class, determine the weight and compute it all—

*JW:* And print out a little tag that summarizes all this information and serves as the stamp itself, and, finally, even do the record keeping. Pitney Bowes periodically comes back and finds out what to charge you for the stamps you've printed so far. Microcontrollers made that order of innovation possible without resorting to multi-chip microprocessors, which would have been impossibly expensive.

Intel saw this as a market that they should work tirelessly to exploit, which they did. The effort to expand the use of microcontrollers was fairly high-profile within the company—not only for the 8048, but for the 8085, which was finding its way into more complex control systems. The 8086 wasn't out yet, it would appear in mid-1978, and certainly hadn't been built into a computer, which would happen in the spring of 1979.[1] Intel meanwhile was openly disdainful of what they called the "minicomputer" market because they didn't see enough volume there to keep the fab lines busy. Gordon Moore made this point repeatedly at shareholder's meetings; he'd say that if we could replace every minicomputer ever built with an Intel processor, it would only keep our fab lines busy for two weeks.

*KC:* Yeah, but he was looking through the wrong end of the telescope.

*JW:* These are smart people. I'm not trying to make fun of them at all. I'm using this as an example of how different the world was, and how different our perspective was, twenty years ago before the PC revolution took off.

*KC:* Exactly.

## LINES OF INHERITANCE

Now, at what point did the 8048 and its success give rise to the 8051? Was there a direct descendancy?

*JW:* Well, when you discuss technical inheritance, you have to bear in mind the big difference between Intel of the nineties and Intel of the seventies. Intel has a high-profile introduction of a new microprocessor roughly every three years. Just this month we've seen the introduction of the Pentium II, which is sort of a Pentium Pro for the mass market, without the in-package L2 cache. Pentium Pro was formally introduced in October or November of '95. Before that, the Pentium came out in May '93, the original 486 in April '89 and the original 386 in October '85. Of course numerous derivations, like the DX2's, SX's and SL's, were interspersed between those, but the high-profile chips—processor cores—come out at two- to four-year intervals. And these are all members of basically a single processor family, the x86.

In contrast, in the seventies Intel was coming out with whole new processor families roughly nine months apart. The 4004 was introduced in November '71 and followed by the 4040, the 8008, the 3000 series—which were bit-slice processors—then the 8080 in June 1974, and we finally arrive at the 8085, the 8048, the 8021 and 8022 in 1976 and 1977. So in a six-year interval Intel introduced about eight or nine significantly different microprocessors for different markets, different philosophies, different instruction sets.

[1] This was the date of the prototype 8086 computer built by Tim Patterson at Seattle Computer Products; 8086-based Intel "blue box" development systems may have existed earlier.

In this time frame it was appropriate for Intel to identify markets, and design new processors to target each market, so that's what they did—and this was not the more mature Intel of today that puts a high priority on upward compatibility with each generation. This was, if you will, the "wild West" Intel that had gone from 4-bit processors to 8-bit processors and was determined to stake out a strong position in the next generation of 16-bit processors. There was a task force set down to develop a new 16-bit single-chip microcontroller.

*KC:* What was the approximate date?

*JW:* The philosophical discussions trying to define the markets and products took place in the autumn of '77. At much the same time Intel was negotiating with Ford to develop new chips for automotive applications. In the autumn of '77, spring of '78 I was doing the feedback carburetor work.

*KC:* Your architectural definition there is dated mid-January 1978.

## A LUCRATIVE FREE LUNCH

*JW:* My recollection of events was that in December of '77 I was part of the Strategic Design Group, which was a function of the Applications Engineering department. Applications Engineering was sort of technical support for the marketing group—we served as the translators between the marketing gurus and the actual customers, so we were engineers that were paid through the marketing department. By that time I had a fairly broad spectrum of 8048 experience. I had done the Ford carburetor project. I had done the Microswitch keyboard project. I had helped write a couple of application notes. I had worked on some other generic keyboard scanning algorithms. I had done bids on an energy management system, a video game system, and an audio sound effects system all using 8048's—which would involve taking a week to dummy up a simple version of the hard parts, to decide whether this was in fact feasible before we would commit to the project. So I had done a lot of very quick-turn design proposals, evaluations, carried a couple through to fruition.

One day in December of '77 I showed up for work and I was broke. My boss Lionel Smith, a superb manager and good friend, owed me a lunch, so I asked him if he'd buy lunch for me that day; he said he couldn't, because he had to go to a planning committee meeting, but there would probably be extra food there. So if I wanted to come to the meeting, I could eat for free as well, which sounded good.

The policy at Intel was that, generally, anybody that wanted to attend a meeting could do so, whether it involved them or not and whether or not it was in their own department. And there were lunch meetings virtually every day—it was a good way to cross-pollinate, to understand people's issues. The other thing that was sort of remarkable was that you could wander into a meeting uninvited and, even if you didn't have prior professional interest, you became a peer by courtesy. When a question came up and people would vote, the head of the department, the vice president of the division, the head of marketing, and you would all have equal votes. This was very heady stuff for somebody fresh out of grad school.

This meeting that my boss invited me to, in pursuit of lunch, was a product planning meeting for the microcontroller group. They had identified a need for two different classes of single-chip microcomputer. One would be a high-performance, high-end 16-bit device—a revolutionary new design—targeting, for example, automotive applications. (It would later be defined and designed in very close cooperation with Ford.) We didn't have a product name, so it was given the name 80XR—80 as the generic form for microprocessors, X because we didn't know what the number was, and R for the Revolutionary design that wasn't constrained. But that would take a couple of years to develop, and we were in the stage of very preliminary discussion. In the meantime it was vital to keep our grip on the 8048 market which, as you say, made a significant contribution to Intel's revenues.

The 8048 at that point had several variations. The first was the 8048 itself. Then the upgraded version, the 8049, offered twice as much program memory and twice as much data memory. A stripped-down version called the 8021 was in a cheaper 24-pin package with only a subset of the instruction set implemented. The 8022 was a slightly enhanced version of the 8021 which included analog-to-digital converters on chip—this was a new technology they were playing with. There was also an EPROM version, the 8748, and a special-purpose version with no memory, the

8035, as well as peripheral controllers, the 8041, 8741, and 8042. So, counting the different memory types and sizes, different pinouts, different A-to-D, there were between seven and ten different part numbers that were all considered "8048 family," all within that first year or two. It was an impressive product line.

## WHERE FROM HERE?

The dirty secret was that we were running out of space. The architecture allowed for no more than 4,000 bytes of program memory—on-chip, off-chip, total. And with the 8049 and 8042, we were already selling chips that had half of that space full. We knew that the next generation would fill it up entirely, and we needed someplace to go from there. There was also confusion about wanting to add new peripherals—a serial port, more I/O capacity. The 8048 was in a 40-pin package, but 16 of those pins were overhead—power, grounds, status, clock-in, reset and so on—which left only 24 pins for simple I/O....or actually, with a couple of test and interrupt pins, there might have been 27 pins on that package that were useful for communicating with the rest of the system. This was beginning to be a serious bottleneck. Complex control systems needed more I/O and, with other designs of the period, it was necessary to throw in off-chip I/O expansion devices of some sort.

That cost more money, and it was money that Intel was not getting. Intel's strategy is to increase the perceived value of products, then charge more for them. Intel will say to the user, "Our chip is 50 cents more, but then you don't have to spend 50 cents for I/O expansion." The user ends up with the same functionality in a smaller—hence cheaper—package that requires less power and gives better reliability. Intel ends up with the extra fifty cents. So, when we ask "For fifty cents, can we add an extra port to the processor?," obviously we can add an extra port for lots less than fifty cents, so that's what we should be doing.

The grand scheme was to proliferate the chips, add more peripherals, add a serial port, an extra parallel port or two, more program memory, more data memory—whatever would fit into the packages we contemplated moving to. The customers that we had already established with the 8048 were high-volume, long-term customers that meshed very well with Intel's way of doing business. During the development interim of this new chip, the 80XR, we needed to keep them from going elsewhere. The product proposed for this holding action was called the 80XE: E standing for "evolutionary," growing out of the 8048, whereas the XR would start from scratch.

When I wandered into this meeting, the XE was already defined as sharing the basic architecture of the 8048, to appeal to the same customers. It would still be an 8-bit machine. It would have the same instruction set, the same registers, the same on-chip data memory. But the existing instructions would be paired with escape codes to do double duty. (This was somewhat the philosophy that Zilog adopted with the Z80.) In the 8048 there was an 8-bit opcode corresponding to every function, controlled through tables. The operation and the operand were both encoded within 8 bits, to keep the decoding simple and the program dense. It was a priority, with such a small amount of on-chip memory, to do as much as humanly possible in that space.

So, for instance, the status word and the I/O ports and the peripherals had a separate opcode to read each of 5 entities—PSW, port 0, port 1, port 2, and a timer. Then there were 5 more opcodes to write to those devices. Still more were needed in order to do logical functions, and then there was an individual opcode to set or clear each of the flags or complement it or to test whether it was true. This was kind of inefficient. For the XE, when I got involved, the plan was to increase the resources by adding an escape code. For example, the instructions that would read or write the timer, if prefixed with an escape, now would read or write the alternate timer. You could in essence double the number of opcodes and still keep opcode compatibility with existing programs.

*KC:* And how many bits did that escape take?

*JW:* "Escape" was posited as one of the several 8-bit opcodes still available in the instruction set. Then, for instance, the second timer would be invoked with Esc+the instructions for the first timer; a fourth parallel port would prefix Esc to the opcodes recycled from port 0. A serial port with both control and data registers would recycle the existing opcodes for ports 1 and 2. That would be a clean growth path—sort of; code would still be dense, and if you didn't use these auxiliary functions, you wouldn't carry the overhead of the

escape byte. If you did, you should be willing to accept it.

So I was trying to do these applications under contract and under pressure—slinging code as quickly as possible, using today's vernacular—and I could not get it out of my mind that the 8048's real problems were not the ones that the committee was addressing. Yes, extra ports or an extra timer would be nice for a lot of customers, but even if you weren't using them, the 8048 made your life hard in several ways. The addressing modes were very difficult. The branch destinations were restrictive. The stack was tiny. The amount of on-chip RAM was minimal. It seemed to me that to attack some of the problems in a way that would be inherently short-lived, while ignoring the others, wasn't a winning solution. It was as though their goal was to squeeze one more generation out of the architecture—to double the resources once more—and then accept that the 8048 was screwed. The plan was, of course, that within a year Intel would have moved to the 16-bit microcontroller and the market would move with it. The 80XR under active development was only a case in point.

## THE 80XE AND THE 432

*KC:* In other words, the 80XE architecture as the committee conceived it was only a stopgap?

*JW:* Yes, just to tide us over! Incidentally it's sort of funny—and I'm not trying to be at all derogatory here—that the 8086 was similarly intended to plug a gap for one year while a chip called the iAPX-432 was completed. The 8086 development was by different people working in different buildings at different times, more or less independently, but the philosophy that they adopted was analogous to my proposal to change the XE to be something different than the 8048.

*KC:* And of course the dreaded 432—

*JW:* I actually think the 432's gotten a bad rap. It was a good thing that Intel had to suffer through and learn from, and some of the ideas first considered for the 432 had results still visible in the Pentium Pro today.

*KC:* Didn't it have one of the world's most complex instruction sets?

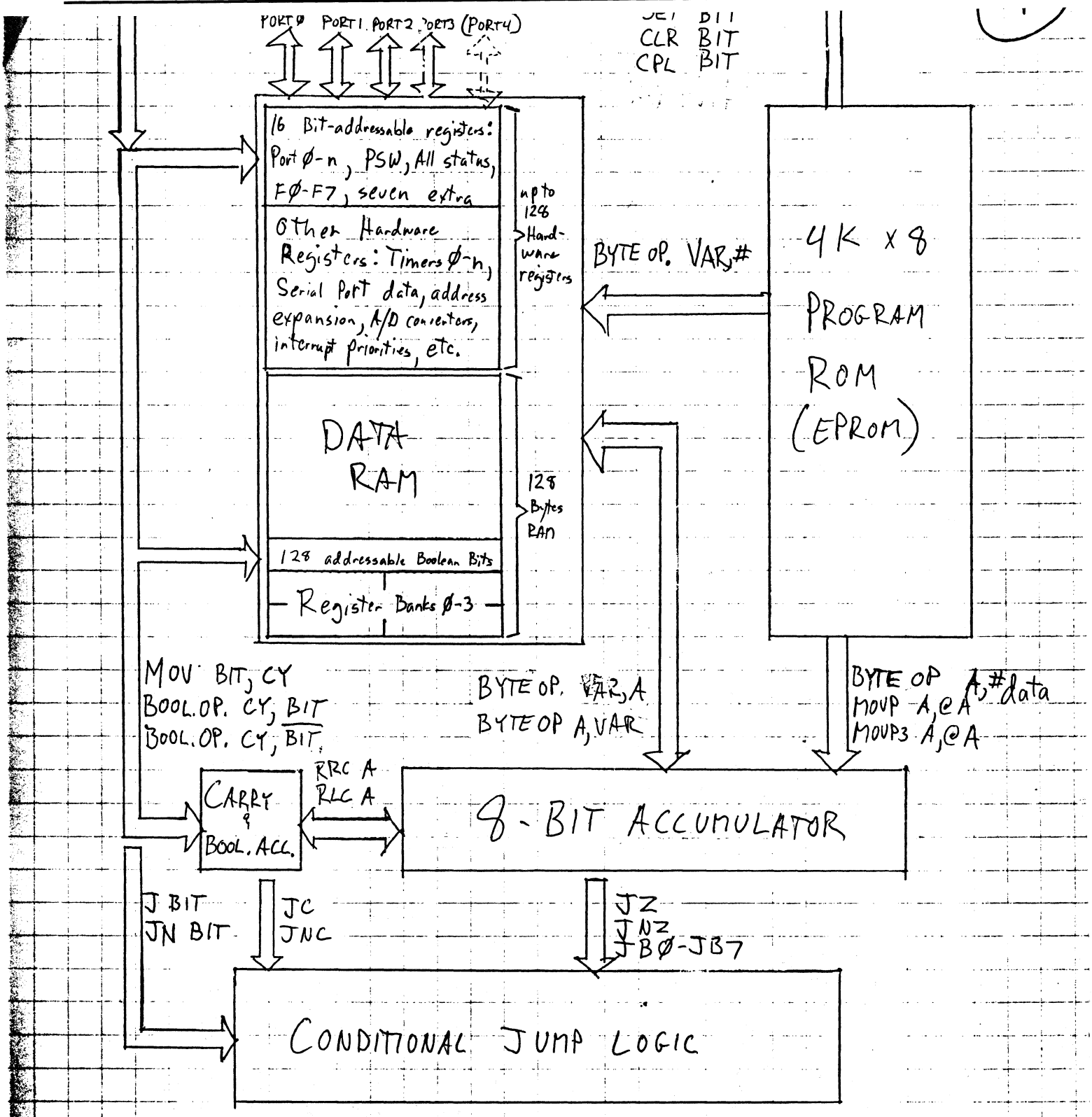
*JW:* Oh, easily. It was bizarre, arcane, incredibly esoteric, and in many ways ill-conceived. The brilliance of the 432 was in the problems and solutions that the architects defined very, very much ahead of their time. This was a chip that supported object-oriented programming at the hardware level. Each operand carried tags indicating the data type and access privileges. The 432 could do a task switch in one instruction, which was incredibly elegant, given that the alternative was to call a whole subroutine that took maybe 200 instructions. Now, the "single instruction" to do a task switch might have taken 5 milliseconds to complete. But some of the core concepts enshrined in the 432 are enjoying a resurgence of popularity, at vastly higher clock rates, that in my opinion justifies the whole experiment.

*KC:* Gotcha. Now, leaving aside the 432—did the 80XE become the 8051?

*JW:* Well, at that meeting we discussed the approach underway at the time, at the level of writing up specs, agreeing on what the opcodes should be, and defining how the serial ports should work. I was concerned that a lot of problems would go unaddressed. On the other hand, this was Intel, I was an outsider, I was there for a free lunch and I felt out of place. After the meeting my boss asked me what I thought, and I indicated that if this was what the next generation was going to be, I was disappointed in its potential. He said "Well, what do you think we should be doing instead?" and I outlined a half-dozen areas in which I thought the 8048 needed improvement in order to be useful past the current generation. He asked me to write it up and propose it at the next meeting, a week later, as an alternative path to be followed. So at the next meeting we had discussions of the 80XE path 1, which was already underway, versus 80XE path 2, which I was proposing.

*KC:* Which is this document right here? [*Figure 1*]

*JW:* Well, this is the more formal version from January [1978] sometime, and it's sort of confusing, because I remember the terminology "path 1" and "path 2," but this spec. is titled "Definition of 80XE Version 2 Architecture." I'm not sure that calling it a version or a path makes a whole lot of difference. Note that I spelled "architecture" wrong




## IV. Complete Design of Version II.

**Figure 1. Hand-drawn block diagram of the 8051**

and had to correct it with white-out—this was back when human frailty was immortalized for all time, before the days of desktop publishing. But this spec. depicts a transitional design state after a bunch of these discussions, and it mentions a "version 1.5," which is a terminology I no longer remember.

The problem in my view was that the 8048 already had a bloated architecture—if you can say that about a chip with 1K memory—which arose from an overlapping series of short-term goals. We had modified the instruction set to support the 8021, modified it again to support the 8022, and yet again to support the peripheral version, the 8041. Every turn of the crank increased the complexity and stuck us with producing new documentation and new assemblers. This sort of proliferation didn't make sense, especially loaded on top of a chip that, as I say, had its own ways of not working too well to begin with. I strongly disagreed with the notion of turning the crank one more time to hold onto the existing market for another year.

The basic philosophy of my rebuttal, which became Path 2, was that much of the functionality needed for the next generation wasn't part of the 8048 at all. What I was suggesting was sort of a unified approach—a greatly expanded set of instructions, of peripherals, of addressing modes, register sets, additional test pins, additional ports—all adding up to a much more open-ended architecture that would define all the anticipated functions, so that future versions could be expanded without rethinking the CPU core itself.

*KC:* Did the powers on high recognize that the [8-bit] 80XE path 2, or version 2, architecture might then survive even past the introduction of the [16-bit] 80XR?

*JW:* I think that was what the marketing people saw as the advantage. The head of the marketing department at that point, a fellow named George Adams, seemed intrigued by the growth prospects. What was sort of ironic was that, for the space of a month or six weeks in December and January, there was debate within this group which in our minds was "wasting an entire month" trying to figure out what to do next. It's comical in retrospect to think that, in the space of four or five weeks *and* over the Christmas-New Year's holiday, we actually persuaded all these different departments and different people to reach consensus on changing direction and trying the new approach. My boss, by the way, Lionel Smith, is still with Intel, and he deserves an awful lot of credit for sucking me into this and supporting me and going to bat for me when there were objections.

*KC:* So in return for "wasting an entire month," you crystallized the direction of development for a product that has lasted almost twenty years and only gone from strength to strength.

*JW:* Looking back, it was a good use of those four weeks!

## THE ARCHITECTURE SHIFTS

Now, backing up to the way the 8048 worked, there were perhaps a dozen opcodes devoted just to reading and writing the I/O ports, one each for "read port 0," "read port 1," "read port 2," "write port 0," "write port 1," "write port 2," and so on. With a dozen opcodes devoted to port transactions, we could do almost no more than read and write two or three ports. My approach was to define a *single* opcode as "read port" and follow it with a byte that said which port to read. A second opcode would become "write port" followed by a byte saying which port to write. Then other opcodes might do logical function between the accumulator and the port; another one might copy a working register to a port—so that you wouldn't have to go through the accumulator—and another one would do the opposite; yet another one would push the port content. The same dozen opcodes would expand to handle a dozen different *functions,* and the port would always be specified by the suffix byte.

Along the same line, there were about a dozen opcodes devoted to clearing or complementing status flags, or to reading pins and testing if they're true or false. A better use of the opcode space would be as a generic function, "jump if a certain bit is true," followed by a second byte indicating which bit to test. Then a second instruction would be jump-if-false, and so on. This basic approach did more than add functionality—it actually shrank the number of opcodes, opening up opcode space that we could use for new functions, like copying the bit into the carry flag, or copying the carry flag to that pin, conditional setting, conditional clearing, things of this sort. For example, the 8048 couldn't subtract in the strict sense. By reading a series of values,

complementing, incrementing, and so forth, you could do something that looked a lot like subtraction.

But Intel of the day had a philosophy called the Principle of Least Astonishment, which my boss explained to me, which says that sometimes it's easier to do what the customer expects than to convince the customer why he is stupid to want that. In a sales pitch, if somebody says "How come there's no subtract instruction?", you can give him a short tutorial on why he doesn't really want a subtract instruction, or you can say "Well, there is a subtract instruction." And the latter is the cleaner thing to do.

*KC:* So the 8051, with a sigh of relief, obviously had a subtract instruction.

*JW:* For instance, and a multiply and a divide. The multiply and the divide finally were quirky little instructions and not all that universal, but they were bullets on the data sheet that the competition didn't have, and that helped get the chip established.

But the 8051—or at that time 80XE path 2 or version 2,—as the philosophical successor to the 8048, inherited all of its functions. So each of the 250 or so opcodes in the 8048 was paralleled by an 8051 instruction that did exactly the same thing to the exact same resource with the exact same semantics in exactly one step, even though it might take two bytes instead of one, or three instead of two. But it was possible to take an existing program, without understanding it, and replace each 8048 instruction more or less mechanically with an 8051 equivalent.

*KC:* And any increased overhead on the micro level was made up for by the increased regularity and versatility of the instruction set on the macro level.

*JW:* Also by increased resources. The 8048 had 1,024 bytes of on-chip memory, the 8049 had 2,048 bytes, but the 8051 had 4,096 bytes. So we could tell people that, even in the absolute worst case [of reusing 8048 code,] we would double the byte count and still give them twice the storage. Clearly any program that did exist [in the 8048] would exist in the 8051. But chances were that the extra space in the 8051 would tempt people to develop new programs and that some of the additional opcodes—for example, bypassing the accumulator—would actually make those programs tighter and faster.

*KC:* In other words, slam your existing 8048 code into the ROM connected to the 8051, and that'll run without alteration while you bum down code for the real 8051 instruction set.

*JW:* Exactly, and Intel actually promoted that approach with a fairly neat hack, a program called CONV51—CONV for converter—which would read in any 8048 assembly language source that met Intel standards and convert that into 8051 assembly language source that was functionally equivalent, instruction by instruction. So take your whatever 8048-based device, run your source through this converter, put it into an 8051, wire an adapter socket into your existing 8048 design. Lo and behold, it will work, it'll probably run faster, and you can start adding new features into all that code space.

Okay, that's what you can do with basically zero engineering work, which is certainly good enough for evaluation. Now buckle down and optimize by refining your critical inner loops. If you had done a software multiply, you could replace the entire subroutine with one hardware multiply instruction. If you were doing I/O in the 8048, you had to copy into the accumulator and then copy to the port—which really slowed things down if you had to save and then restore the state of a busy accumulator—but the 8051 would allow direct move of contents from register. Whether you were optimizing your old code or writing from scratch, the 8051 was a clear win, and the embedded market so voted.

## FOLLOWING THE TERRAIN

*KC:* And what was your title in all this?

*JW:* Depending on which business card I had in my wallet that day, I was either an applications engineer or a strategic design engineer. My responsibility was to provide technical support to the marketing people and to be a liaison between designers and the rest of the company. Which, by the way, I think was not typical. This industry is sometimes faulted for having people with grandiose titles like "architecture specialist," whose job—according to the detractors—is to remain in an ivory tower thinking grand thoughts and occasionally promulgating the future of the chip. I think I was rela-

tively unusual in having had hands-on experience with the prior generation while I attempted to solve the problems of the new one, hoping that the rest of the world would benefit from those solutions, rather than sitting in a vacuum trying to predict what the problems of the outsiders would be.

Back in the summer of '77, fall of '77, in anticipation of all this I would be writing application notes, doing customer training, doing field sales engineer training. New products would come out, I would write articles about them. That sort of thing. I ended up having a lot of customer contact because somebody would have trouble getting a certain chip to work, they would call the factory and the call might get routed to me. Or somebody in academia would find an instruction that didn't work—at least not the way they expected it to—and call in to complain. "Hey, there's a bug in your chip, I tried to do *x* and it didn't work the way I wanted it to." And I'd have to explain, "Well, nobody ever said it would work the way you expected it to. I'm not surprised that your results are unsatisfactory. Here's a description of how to use this instruction and if you use it in that form it does work." The point is that I would talk to several people a week, calling in to talk about something. And I got into the habit of saying, towards the end of these calls, "Tell me a little bit about this chip—other than this problem you're having what's your general reaction? What do you like about the chip? What's most or least useful to you? What should be different?" Now, this was a statistically very unsound way of surveying customers, but based on people's replies you could almost always identify or infer what their immediate problem was.

*KC:* You were real close to the terrain.

*JW:* The 8048, for instance, had two software flags, zero and one, and you could clear each flag or complement each flag or you could test for the flag being set. You couldn't set it initially. You had to clear it first and then complement it. And you couldn't test for it being false, but you could work around that. I'd ask people what they thought about the chip and they'd say "Oh, I really like those software flags, they've saved me a lot of trouble." So I'd poke at what they were saying and it might turn out that what they really wanted on the next chip was five software flags—one for the shift key and one for the control key and one for each of three other functions.

Now, I couldn't take that and say "Okay, for the next generation what we need is five flags." But what I could do was infer that everybody wanted *more* flags. It was the same situation with I/O pins; the solution wasn't to say very specifically "Let's give them four pins instead of two," the solution was to say that if people wanted more pins, we'd give them as many pins as they could use. So on the 8051 we arranged it such that any bit—all 32 I/O pins, all eight bits of the accumulator, of an ancillary register, of the status word, of the interrupt logic, of the serial port control logic, or of the timer logic—any of those bits could be set, or cleared, or complemented, or tested for true or for false, or tested for true and then cleared....you get the idea. We gave them each 8-bit addresses. So a whole bunch of generic instructions could be used in conjunction with any of those pins or bits. And then just for good measure we took 128 of the bit-identification values and mapped them onto bits of RAM....

*KC:* Oh, boy....

*JW:* ....so now we had 128 software flags, and when people said "I like two but I need five," we said "Okay, we'll give you those five and here's 123 more for free." Our philosophy was to figure out what the problems were and solve them in such a way that they'd never be problems again.

*KC:* You created, in essence, an almost infinitely flexible chip.

*JW:* Certainly we over-designed the one-year, stop-gap measure that the committee first envisioned.

*KC:* Did it take extra pins to implement all this?

*JW:* That was a consideration. One of the amusing things about the 8051 was that from the start it targeted two packages. The 40-pin package that had been in production was going to be introduced simultaneously with a 48-pin package that we had great hopes for. The early presentations all had figures with both versions. We had cut the overhead pins from 16 down to just 8, which meant that the 40-pin package would offer 32 pins of I/O usable either as four 8-bit ports, or as three 8-bit ports plus eight configurable ancillary functions like interrupt requests, or read and write strobes if you were doing memory expansion, that sort of

thing. Then the 48-pin package would have five 8-bit ports—40 pins for I/O—plus 8 pins for overhead. And if you look at the early documentation it actually defines five ports, zero through four. To my knowledge Intel never went into production with this 48-pin device, certainly not while I was there. This fifth port I don't think was ever used in the way it was conceived. And it's sort of funny—if you look at the port map there is very clearly space reserved for port 4 that never got used. In fact, that may have been implemented on the original die in anticipation of the new package.

Let me back up a little bit. This practice of breaking I/O instructions into two halves, the generic operation and then the port number on which it operated, had two advantages. One was more efficient use of opcodes, which left you room to create new operations affecting the port—one of which, for instance, was initialize a port to an 8-bit constant value just by copying a number to the port, without having to compute first or go through the accumulator. But the other, perhaps more important advantage was freedom from worry about things like: Okay, are we going to add an extra port? Will we need a second timer? Are we going to have an A-to-D converter someday? If so, how do we talk with those? A whole eight bits to specify the port address gave us the potential for a hundred or more ports—the half dozen that we knew about now, and nearly unlimited growth in the future with no impact on the architecture itself. Someday, even if we dropped this in a 68-pin package, we could accommodate new ports, different combinations of hardware features, and simply assign each one an unused port number. No need for adjusting the assembler. No need for developing new compilers, or rewriting the documentation, or changing the customer training curriculum.

*KC:* Meanwhile, back at the ranch, where production and marketing were pacing the floor, it can only have been construed as an advantage that thanks to this spiffy new two-layer instruction set, the 8051 contrived to fit in the 40-pin package of the 8048.

*JW:* That certainly helped customer adoption. It meant that the automated assembly techniques in their factories, or their logic analyzers with 40-pin probes, or their employees trained to solder in 40-pin sockets, didn't require substantial adjustment. We were replacing a 40-pin package with a significantly more powerful 40-pin package and the only thing that changed was the number on the cap.

*KC:* What did it mean about the cost of the chip?

*JW:* I never paid a lot of attention to manufacturing costs and, although someone must have, I'm not sure who it was. I think the 8048, in the late seventies and depending on volume and customer, was selling in the $5 to $12 price band. The 8051 was introduced with a very clear price premium, probably at $40 to $60. The package cost would be the same because the package was the same. The die was larger, with a processor core maybe two or three times the size of the 8048, and then more ROM and more RAM. So it was a more expensive device, but we argued that it was more capable, more flexible, more capacity, worth the premium. Certain customers at least seemed to agree with us.

## TAKING OVER THE WORLD

*KC:* The 8051 was introduced when?

*JW:* We had working parts and started showing them to customers, started making samples available, in the spring of 1980—along with the usual app notes, seminars and articles. If you look at the graph of sales, the numbers are kind of hugging the horizontal axis for the first two or three years—which is partly just a matter of scale, and partly reflective of somewhat slow acceptance. It takes a few years for chips to go into high volume production, and in any case, the numbers shipped of the 8051 were significant from the beginning; they only pale when compared with the 200 million units being shipped now.

*KC:* Okay, so unit shipments per year seem to have reached 25 million in 1984, didn't get to 50 million until sometime in 1988, but then 100 million in 1991, 150 million in 1993 and 200 million sometime in 1994. [*Figure 2*]

*JW:* And oh, let's say, 225 million units per year today. This is not a typical growth curve. For a microprocessor to remain in production for 10 years, not to mention high-volume production, is unheard of. I suppose you could go to Fry's today and buy an 8088, which was roughly contemporary with the 8051, but the 8088 certainly is not a first-echelon microprocessor in this decade. It's been superseded by six or seven later generations in the same product line. The idea that a chip

wouldn't come into its own until 10 years after introduction and that thereafter the growth would, essentially, monotonically increase for 15 years, is a shocking turn of events for the industry.

## INFINITE VARIETY

*KC:* There must be an incredible number of variations shipping.

*JW:* Yes, and that was a partly unforeseen corollary of the breadth of the architecture, which was to 8051's benefit—I mean in terms of becoming a standard—and perhaps, ultimately, to Intel's detriment. Intel has historically blazed the trail introducing new products and then modifying their instruction sets, so that other manufacturers would follow suit. The 8086, used in the early IBM PC compatibles, had its instruction set adopted by NEC for the V20. Intel added some new instructions to the 286 and everybody else had to follow along. The 386 included a bunch of new instructions and new modes, virtual machines, everybody else had to add those. The 486 added some new instructions, then the Pentium, now the Pentium with MMX extensions and the Pentium Pro's conditional moves and new flag settings. Everybody has to follow that. But Intel at each stage was the first to develop a new generation and everybody else had to accommodate Intel's plans. Our goal for the 8051 was to define an architecture with room for extra timers, extra counters, more ports, A-D circuitry, all without having to rethink the basic instruction set. But the capacious architecture allowed companies to introduce their own variants, graft on their own extensions, without waiting for Intel to get there first. So other companies second-sourced the 8051 core architecture and added more memory, more ports, whatever they were good at.

Philips was promoting a serial bus for interacting among processors within an appliance. Other companies added A-to-D converters, changed the core frequency, did low-power versions, offered different package types—filled almost every niche they could and still leverage the development tools and prototyping experience that could be derived from the Intel family line. The irony is that Intel, which has obviously made many fortunes from variations on this core, quickly lost control of the architecture—to the point that most of this growth probably reflects Intel's competitors coming on line with new subtypes.

So, just to put things in perspective, as of last year's data Intel had 64 different flavors of 8051 class products—different part numbers with different sets of capabilities. Philips had 101, with really incredible variety, and Siemens had 24. Then two or three other vendors had very special versions, like Dallas Semiconductor's 8051 with its memory backed up by a battery right in the package—a non-volatile 8051 which could, for example, be software-programmable to compete with flash EPROM.

The 8051 as we originally specified it had four kilobytes of code, 128 bytes of data, 12MHz operating frequency, 32 I/O pins, two timers and one UART. Today, by selecting the appropriate "8051-type" descendant, you can have zero kilobytes of code—that's a device with no ROM on board—or one with one, or two, or four, or six, or eight, 16, 32, 48, up to 64 kilobytes of on-chip program memory; and you can start with less data RAM than the original and go all the way up to two or three KB. Available speeds range from slower than the original up to about 50MHz. You can get chips with only two ports, with anywhere from two to 68 I/O pins. Yet all of this is on the 8051 core with the same basic opcodes and instructions.

One version, in a six-pin package, I find quite comical and amazing. This is a device—I think it's from Philips—designed expressly for smart cards, credit cards with a computer system built into the plastic. They target low cost, small physical size, and the highest reliability possible. So in a six-pin package it's got power, ground, clock in, reset in, one serial bit in and one serial bit out. No parallel ports at all. I mean this is hilarious. People make transistors with more pins than that. And yet it addresses all the issues that are important in context.

That may be a lesson that things have come full circle. Initially a chip like the 8048, that could be programmed for a variety of functions, meant that economy of scale, the ability to defray tooling and development costs broadly, would let a general-purpose microcontroller compete squarely with a custom keyboard control chip developed by Texas Instruments—a general-purpose computer achieve dominance of one tiny little niche market. But this market is so huge today that it's worth targeting any one niche with an incredibly specific version of the product.

*KC:* The latest statistic I've seen, and this has to be pretty loose, says that microcontrollers are slightly more numerous than people—that there are seven or eight billion currently installed in the world. We've pretty well covered why the 8051 ain't dead yet; it's an absolutely entrenched commodity part available in such a variety of configurations that you can say how many bytes of code you need, how many pins you need, whether it requires A-D capability, and find somebody's chip with exactly that combination of resources. If you can get that much clock rate, you're really not worried about speed.

*JW:* Don't forget the comfort factor. If you get ready for production and guess wrong, you discover that your functions won't fit in 6KB, upgrading to 8KB isn't much more than an annoyance. It's not as though you have to start from scratch. Multiple vendors are compatible with the tools you've already got, and your learning curve has been paid for over the years by thousands of people.

*KC:* And all this without the dreaded incantation "ASIC." So what does come next?

## THE DRAGSTRIP AND THE ELEVATOR

*JW:* Let's see. I think we can find our way into that by exploring the fundamental difference between a microcontroller and a microprocessor. If you're dealing with other machines, or memory, or raw math, faster is always better. 200MHz is better than 100MHz, 400MHz would be better than 200MHz, within a very special environment that lives inside a box. Of course consumer demand also influences the performance curve of microprocessors—in a way that's ceased to be entirely rational, it's like the horsepower race of the sixties and seventies. You get perceived added value from making your buddy's jaw drop.

A microcontroller enjoys the phenomenal advantage of dealing with the real world, which means it cooperates on a just-in-time basis with devices whose performance parameters don't change a whole lot. If you're controlling a VCR, it only matters that you turn the motors on and off more quickly than the mechanism is able to respond physically. If you're dealing with electric motor control or dimming light bulbs, you just need to switch your current at the right phase of a 60Hz signal.

*KC:* When the machine becomes ready to respond, the next instruction is sitting there waiting for it.

*JW:* For instance, car engines today might turn fifty per cent faster than car engines turned 20 years ago. But a computer that was fast enough to compute fuel injection requirements 20 years ago is by and large fast enough to do the same thing today, because the requirements haven't changed significantly.

*KC:* And if it's necessary for the controller to be fifty per cent faster for your application, that's been taken care of through perfectly natural reductions in die size.

*JW:* Absolutely. The chips have improved by a factor of four in speed and, generally, that's more than enough. The other thing that's interesting is that, for most of this stuff, 8 bits is more than adequate to handle the required computation. If you're trying to control a motor by one part in a hundred, 8 bits will easily represent a hundred separate steps. The number of keys on a keyboard hasn't increased a whole lot. The rate at which LED displays have to be multiplexed in order to appear non-flickering hasn't changed—our eyes work the same way they always did. Modem frequencies haven't gone up by more than a factor of eight and at some point aren't going to go up any more. Thermostats don't need to be any more precise now than they were 20 years ago. Scales in a bathroom don't need to read higher weights than they used to. Clocks don't have to be more precise or faster. Scoreboards for sports don't show scores that are especially higher now than they were. Therefore, a microcontroller that could keep up with application *x* a few years ago can probably keep up with it today. Most of the processes that are appropriately controlled in this way have nearly flat rates of increase over twenty years or so—

*KC:* And then Moore's Law goes *blam!* right up through the middle. [*Figure 3*]

*JW:* Almost a vertical line compared with growth in what we consider to be the real world. So there are certain things that just don't need to get any better. What 8 bits would run in the beginning, 8 bits can continue to run.

*KC:* Which puts us ninety degrees out from the microprocessor market since, of course, what sells microprocessors is increased clock rate and word width for that dragstrip inside the box. Meanwhile, your average microcontroller is plugging along stamped "Acme 8-Bit Plastic Package." What, then, accounts for this stunning expansion of sales?

## TAXES, RICE, AND MAGIC WANDS

*JW:* New applications in old markets, old applications in new markets. For instance, one big market developing today is taxation machines in China. China has a huge unregulated economy created by local villagers transacting on a cash or barter basis with each other. There's no good mechanism—no infrastructure, no mail system, no telephones—which will allow the government any perspective on the economy of the Chinese interior. One application that I've heard of is a tax taking, like a census taking, where people drive around to the different villages and figure out how much sales the merchants and general stores have had, and punch it into a data logging device that keeps track of sales and tax liabilities. The 8051 is a chip they're using for these things because it does everything it needs to do. Nothing pertinent has changed significantly in the last twenty years—cash registers don't have to add any faster, totals aren't materially higher, the number of keys on the keyboard is almost identical, the rate at which human beings press keys is pretty much the same, as is the rate at which numbers flash on the screen. But with the devices getting steadily cheaper, and the rest of the system integration improving, the potential market for the 8051 finds itself gaining a fifth of the world's population. That may be an exaggeration in raw numbers but it's not in principle.

*KC:* In other words, the level of development of the 8-bit microcontroller stays largely flat, the level of development of other stuff rises; pretty soon the curve of your average developing country crosses the flat plane of the microcontroller....

*JW:* Suddenly a market that used to be priced out is now in line to buy. A cash register used to be a big bulky expensive mechanical thing. You had one in each store. The price and size fall to the point that a cash register is a battery-powered device based on an 8051. You can now have 100,000 tax agents traveling through the Chinese interior collecting statistics from the individual merchants and bolstering the national economy.

*KC:* Asia in particular is an interesting case—because, in a sense, that's where I saw the first instance of a microprocessor-style horsepower race applied to microcontrollers. I think it's statistically provable that in Asia, whenever electrification reaches a new region, the first line-powered device that most households buy is a rice cooker, which typically uses a microcontroller to profile the temperature depending on the water content of the rice. It happened that I myself needed a new one, so I was in Marina Market in Cupertino looking at an entire wall of rice cookers; the biggest, fanciest one was made by Supentown, which isn't a brand you see a lot of in the United States, and it had a yellow-and-black sticker on the box that said "Latest 32-Bit Fuzzy Logic!" And I said to myself, why on earth would a rice cooker need thirty-two bits—

*JW:* So that was the one you had to buy, obviously.

*KC:* No, I got a National [Panasonic] that was distinctly eight-bit. But it's clear that the time is ripe in developing economies for wholesale adoption of microcontrollers, not only by governments but also by retail consumers, and there goes that curve again. So what about other new applications? One, of course, is smart cards. What kind of personal communication is coming on line now for which the 8051 might be useful?

*JW:* Oh, identification systems, cell phones or pagers or navigation systems, devices that report the physical position of your children, for instance.

*KC:* An 8051 will actually handle personal positioning?

*JW:* Not by itself. But the sorts of appliances that are developing, the ways for people to wander around without losing their connectedness to the rest of the world through wireless, through cell phones, through wireless modems, through the Global Positioning System, navigation systems in cars—certainly some of those devices involve or will involve an 8051, possibly as one of several processors.

*KC:* So that we may not be far from the point when an 8051 is something you wear—as normal a part of your gear as your shoelaces?

*JW:* Well, not just yet. For example, complex digital watches probably don't need more than four bits, because watches are a particularly non-challenging application. But if I had a heart monitor or some sort of a diagnostic device, or this little pump for diabetics that periodically adds measured insulin to your bloodstream through a permanently installed catheter. I'm not sure those necessarily use 8051's, but clearly it's a microcontroller application in a broader sense.

Another potentially big application is public security and identification, modules that verify who you say you are. There was a story on ABC News last week about a little ID wand which would be issued to individuals, who could set up accounts with their gas stations such that simply waving this thing in front of the gas pump tells the gas pump who to charge the following transaction to. You don't have to slide a card or punch a PIN; when you're in physical proximity of a dispensing device, it will do what it does and charge you accordingly. That could be the size of a lipstick, sitting on your key ring as you wander through. Interactive toll taking, a little module on your dashboard that gets interrogated by the transponder as you drive through the toll gate, and your account gets debited the cost of traveling on that freeway that day. I was also thinking of security in transactions, something that might be a little safer than a credit card, where if you lose the card and don't cancel it, it wouldn't work unless it was in custody of the proper user; it would probably have to verify some metric of your body.

Last year Philips sponsored a design contest, asking people to suggest new applications for 8051's, and they got hundreds of submissions. I'm not trying to do a Philips plug here, but some of the applications were fascinating. Remote heart monitors. Electric dice for strategy games, that use the microcontroller as a random number generator. A guidance system for a telescope that would store time of day, latitude and longitude, North Pole base and levelness—all of this from sensors and controlling a motorized gimbaling system. You'd just give it a request, I'd like to see Jupiter, and the machine could figure out where Jupiter was from your space-time coordinates and automatically point your telescope in that direction. That was one of the applications. Robotics—there's a scheme for an autonomous insect robot where each leg would have its own 8051-based control module taking orders from a centralized core. The CPU would set destination, direction, and speed, and then leave it to the individual leg controllers to figure out their motion based on the states of the other legs. This is sort of the way the nervous system works in insects anyway, maybe in people.

*KC:* And it turns out to be an incredibly powerful model in all sorts of networks—have a central processor as a traffic cop and surround it with—

*JW:* small, localized, autonomous controllers—

*KC:* and then the more processors you have at the end points the more powerful the network gets.

*JW:* And the more fault-tolerant. Remove one leg and the other seven compensate, which is a real-world situation. Now, currently, in such a network, if you remove the CPU the peripheral processors won't gang together and compensate—although that's a different topic, and we are moving there.

## MICROCONTROLLED RECREATION

One of my favorite applications for an 8051 in this book was as a peripheral for bowling balls.

*KC:* Smart bowling balls?!

*JW:* This device would be a dime-sized thing that you stick into the thumb hole of your bowling ball, deeper than where your thumb goes, and it can sense with accelerometers and infrared transmitters and receivers if the ball is spinning versus if it's sliding, if it's rolling, how fast is it rolling? It can measure the time from release to when it touches the ground, which lets it determine direction; then it can sense the time before it strikes the first pin, which then tells it how fast it was moving, how much of that time was sliding and how much rolling. The ball comes back having logged a complete diagnostic of your bowling technique, then uses the same infrared hardware to report back the data to an external unit.

*KC:* And then, of course, no sooner do we produce this bowling ball than we want to implant [the device] into a tennis ball and then a golf ball.

*JW:* Now, see, I hadn't even thought in those terms but that makes perfect sense.

*KC:* Well, believe me, if you could create that functionality in a golf ball you would make a fortune.

*JW:* I was just astounded at the creativity. Who could look at a bowling ball, and identify a market opportunity for a computer system inside the thumb hole?

## THE CREDITS

*KC:* You have a foil here called "8051 Development Credits" which is a list of names and titles—would you take this opportunity to say more about these people?

*JW:* Lionel Smith had a remarkable effect on my career, by both hiring me into Intel and encouraging me to get involved in a lot of things. I owe him a whole lot. He's still working for Intel, down in the Chandler operation in Arizona. I talk with him a couple of times a year. I hope someday soon he retires, Lord knows he deserves it. He's the one who said "Why don't you come to this [80XE] planning meeting?" and afterwards encouraged me to codify my thoughts in a specification. I thought then that I was wasting my time, and I might have been, except that Lionel went to bat for me and did the behind-the-scenes lobbying that made this happen.

George Adams was his boss at the time. In the winter of '77-'78, during these four or six weeks of debate, we identified two polarized paths. The battle lines were the reverse of what you would expect. George Adams, the marketing head, was pushing perhaps most strongly for changing the architecture as a bet on future growth. Meanwhile the design engineers were arguing in favor of retaining byte-level compatibility with the existing product.

*KC:* Because, if it ain't busted, don't fix it?

*JW:* And it's usually the other way around. It's usually the marketing people that are accused of saying that we can't do anything to rock the boat, that we have to do exactly what's been proven to work. And usually the design engineers are being a little more progressive, saying "Let's do something different and rationalize it this way." Engineers are willing to reassemble with the right software tools. Engineers are typically trying to make their jobs more interesting by being aggressive. And marketing people would normally want to keep the old architecture in order to give themselves a greater challenge in selling it....

*KC:* Right, and they're saying, we know we can sell *a* so let's keep *a'* as close to *a* as possible.

*JW:* Yeah, and so it was sort of refreshing that George Adams, the marketing manager, was lobbying in favor of change while the design engineers were lobbying to keep it the same. Bob Kaehler and Peter Jones were the product marketing managers for the chip itself. Gene Hill was the engineering manager in charge of the project within the design team. The three engineers actually working at the bare metal that I knew were Marty Palowski, Steve Sample and Bob Wickersham. If I remember correctly, soon after the 8051 came out, Marty left Intel and formed a company called MetaLink that created development tools expressly for 8051's. Steve Sample left Intel and I believe he's a vice president at QuickTurn currently doing tools for chip designers. Bob Wickersham last I heard was still at Intel in Chandler, working on 16-bit and 32-bit embedded controllers.

## MYSTERY SOLVED

*KC:* One more unanswered question. What happened to the 80XR?

*JW:* The 80XR was developed under contract with Ford with a collaborative architecture. It was used for high-end engine control, initially targeting eight-cylinder engines and eventually used in the Ford Taurus was one of the big chips. It was basically proprietary for four or five years. When it became a commercial product it was first given the part number 8061, I think, which didn't last long and it was finally introduced as the Intel 8096, a 16-bit single-chip controller that in a lot of ways anticipated some aspects of RISC processing, although I don't want to get into that here. It also had a very complex I/O system, a bunch of logic related to acquiring events and causing things to happen autonomous to the processor. So, for instance, setting up engine control, spark time and fuel injection wasn't a matter of writing a program that at the right instant would execute an output operation. The program would maintain a table of values managed by counters, timers and comparators and certain I/O bits such that an injection sequence should start at a certain degree position of the camshaft and continue for so many clock cycles. The program was to maintain this table and tweak these values, but autonomous of the engine

itself, and then as the engine was spinning the appropriate control signals would be generated to cause the fuel injection to happen or not happen, as a corollary of the table that was being read in.

The '96 evolved into something called the 80196, which was popular in modems and in disk controllers. It was very good at math, and had DSP-type functions to the extent that you could do servomechanical control for high-speed positioning of disk read heads, that sort of thing. I think Intel's still selling it and has deemed it a successful product. It's developed new markets.

But sixteen bits and thirty-two bits, powerful as they are, didn't much disrupt the ascendancy of the 8051. Not every 8-bit microcontroller in the world is waiting in fear and trembling to be obsoleted by a 16-bit version. Intel has discovered markets that need a particular level of capability and will steadily buy the cheapest product that provides it. A refrigerator need never go beyond eight bits, if that's enough.

*John Wharton, president of Applications Research in Palo Alto CA, is a free-lance technology analyst and writer, and lectures in EE and CS at Stanford University. He lives with a dog, Samantha, a cat, Zero, and three geriatric cars. In 1995 he appeared as "The Shower Guy" on* The Late Show with David Letterman.

## 1.3 IS HISTORY....

We've advised you that some back issues of the ENGINE are threatening to go out of print, and issue 1.3 (January-March 1994) is now unavailable. Since it's a corner-stapled "flyer" issue, we *could* reprint it without much trouble, and we will if there's enough demand. Meanwhile "Land of the Silent Giants," that issue's article about the Livermore collection, is available as a reprint for $2.

But be warned: 1.3 will be followed into the shadows by several early issues now in very short supply. 1.4 and after, the saddle-stapled offset copies, would be much more difficult to reprint and we're not sure we will. If you're missing any ENGINEs from your own set, you'd be prudent to order them soon; 1.1 is $3 per copy, and 1.2, 1.4, or any issue from volumes 2 and 3 is $6 each, first class postage included.

## PLACES IN THE SUN....
### Examining a 2/170

One thing about micros is, they're easy, right? Put the beast on a handy table, pop the case, look for magic numbers like "Z80A," "41264," or "EV-332," count a few chips, catch the dates on the corners of the boards, and before long you have a working idea of the little dear's place in history.

The Sun 2/170 generously donated to the CHAC by Dolby Laboratories in December was quite a different story—apart from the fact that it wasn't going on any tables, or not easily. (The CPU is about the size of a small dorm refrigerator, which it remarkably resembles in appearance, but that doesn't count the two disks and two tape drives that fill up most of the rest of a nineteen-inch rack.) We poked around inside the case, timidly, and concluded that this box was cram-full of goodies that we scarcely understood.

Luckily for us, James Birdsall—a steady supporter of the CHAC and the author of the invaluable Sun Hardware Reference—was in the South Bay this spring and stopped by to take a look at it. We were right that it's loaded for bear. The CPU is a Multibus Prime board with a 10MHz 68010; fast floating-point is taken care of by a Sky coprocessor board. Memory is on two Helios boards, one 3MB and one 4MB (that's a whole lot of 256K DRAM!) which, together with the one meg on the mono framebuffer, makes up the maximum of 8MB that this backplane will support. Ports, of which there are lots, belong to a Systech MTI-1600 serial I/O controller and a Sun SCSI-serial board; networking is through Sun Ethernet as usual. Storage is by a CLIC-II tape drive connected to a Sysgen SC-4000 controller; a Cipher tape drive connected to a Xylogics 472 nine-track controller; and two Fujitsu Eagles driven by a Xylogics 450 SMD. There's also a prototyping board and a breakout board.

All the above are known quantities but the other four boards are not; they're called Zip boards, made by Mercury Computer Systems, and they all have different layouts. If any ENGINE reader can tell us what these boards do, we'd love to know. Meanwhile, thanks to Dolby Laboratories, to Scott Robinson—whose idea this donation was—and to James for helping us figure it all out. This massive black rack is an imposing and pertinent addition to the CHAC's collection.

## IN MEMORIAM: JEAN HOERNI

Jean Amedée Hoerni, inventor of the world's most widely used technique for integrated circuit fabrication, died of complications from leukemia in Seattle on January 12, 1997. He was seventy-two.

Hoerni was pre-eminently a scientist and had little tolerance for the detail and protocol of business, yet his ideas were so innovative and bankable that again and again he found himself participating in startups. He was a founder of Fairchild Semiconductor, Intersil, and Telmos, and served as the first general manager of Teledyne's semiconductor division. He often recalled that he had tried to interest Swiss backers of Intersil in manufacturing watches that depended on semiconductors to keep time, but that the idea had been too radical for them.

In Silicon Valley he will be remembered best as one of the "Traitorous Eight" who founded Fairchild, along with Robert Noyce, Julius Blank, Victor Grinich, Eugene Kleiner, Gordon Moore, Jay Last and Sheldon Roberts. At Fairchild his contribution was the "planar process" for IC production, which was much simpler and more straightforward than the methods of competitors, and which Robert Noyce used in the company's fab on Charleston Road in Palo Alto to produce what have been called the first "commercially practicable" integrated circuits. This process has, of course, proved superbly congenial to further development and resulted in the almost ludicrously cheap and abundant chips of today. For this accomplishment he was awarded the IEEE Computer Society's W. Wallace McDowell award in 1972, as well as the Edward Longstreth Medal of the Franklin Institute.

Hoerni was born in Geneva in 1924 and earned two degrees in physics from the University of Geneva, then a second doctorate from the University of Cambridge in 1952. Shortly thereafter he had his first experience of California as a research fellow with Linus Pauling at the California Institute of Technology.

Four years later, he moved to Bell Labs to work on the first transistor with William Shockley, but bounced back to Palo Alto as a staff member of Shockley Semiconductor Laboratories. Fairchild and fame, or such fame as he would concede, were around the corner.

By repute a shy and formal man, Hoerni had copious energy and often drove himself to do difficult things. In later life he became an avid mountaineer and trekker and, on one trip to the Karakoram Mountains, was struck by the pitiless poverty and isolation of the Balti people—local Muslim tribes of Tibetan origin. Working with mountaineer Greg Mortenson, Hoerni created the Central Asia Foundation to contribute to their education and welfare. A year ago, the Foundation completed construction of a school in the village of Korphe.

Hoerni's adventures at high altitude had to end in the summer of 1995 when he was diagnosed with the early stages of acute leukemia. Surgery in July 1996 slowed the progress of the disease but did not arrest it. He spent his last months primarily with his wife, children and grandchildren.

A memorial service for Jean Hoerni was held on Wednesday, February 12, at Stanford University Memorial Chapel. The ANALYTICAL ENGINE extends condolence to his wife, Jennifer Wilson, of Seattle; his son, Michael Hoerni, of Nochistlan, Mexico; his daughters, Anne Blackwell of Half Moon Bay and Susan Killham of El Granada; his five grandchildren; and his brother, Marc Hoerni, of Geneva.

## OLIVER IN HALL OF FAME

Dr. Bernard More Oliver, lead designer of the HP 9100 and 35 calculators and Hewlett-Packard's Director of Research for decades, has been elected to the Silicon Valley Engineers' Hall of Fame and was inducted at the annual SVEC Engineers' Week Banquet on February 20th, 1997.

Consideration for the Hall of Fame is based on the quality, scope and duration of a nominee's contribution to the art of engineering. For testimony to Dr. Oliver's prowess in the field, consult "A Core Plane in Amber," the interview with him in ENGINE 2.3. His nomination to the Hall of Fame originated with the CHAC and we are honored that the Selection Committee concurred with our opinion.

# IN MEMORIAM: T. VINCENT LEARSON

T. Vincent "Vin" Learson, former chief executive of IBM, died at New York Hospital-Cornell Medical Center in Manhattan on November 4, 1996. He was eighty-four.

Learson joined IBM in 1935 and held numerous positions in sales, marketing and administration, which gave him formidable command of the company's operations and priorities in the most minute detail. With a blunt manner, imposing physical presence, and encyclopedic memory, Learson was the terror of subordinates, who knew they would suffer scathing criticism if their reports to him were even trivially inaccurate. Yet his love of tactics never obstructed a fierce appetite for strategic success, and as the 1950s turned to the 1960s, he became convinced that IBM was losing crucial ground in the computer industry just when corporate and military orders for computing equipment promised to increase dramatically. He championed a new, internally consistent product line of mainframes, built from semiconductor technology which was then scarcely tested, and which would share a common software architecture.

He was able to convince Thomas J. Watson jr. of the merits of this approach, but when he took it to the IBM division chiefs and senior engineers, they complained that an immense body of work in progress would have to be scrapped. To Learson, the sacrifice was easily bearable by the greater good. He met their opposition first with an iron grip on concept—Fred Brooks referred to him as an "incredibly sharp" manager who could attend technical seminars and understand "everything immediately"—and, when more leverage was needed, with considerable force of personality. At last, in the waning days of 1961, he banished the members of IBM's internal SPREAD committee to a motel in Connecticut and told them not to return to their IBM offices until their conclusions were presentable.

The committee members offered the eighty-page SPREAD Report, which became the blueprint for the IBM System/360, then the single most expensive American industrial initiative in history. *Fortune* magazine called it "the most crucial and portentous—as well as perhaps the riskiest—business judgment of recent times," and Watson admitted in his autobiography that the simultaneous introduction of the System/360 line "was the biggest, riskiest decision I ever made, and I agonized about it for weeks, but deep down I believed there was nothing IBM couldn't do....Vin was the father of the new line of machines. His intention was to make all other computers obsolete."

He may not have accomplished that, but certainly his firm hand on the tiller was qualitative to IBM's greatest and most memorable success. After the April 1964 introduction of System/360, he continued to rise within the company, and served as its chairman from 1971 to his retirement in 1973. High office did nothing to mellow his opinions, and he was quite ready to disagree vehemently with anyone else in the company, including Tom and Dick Watson. But his strategic talents were formidable, and as early as 1972—while IBM was one of the most profitable and admired companies in the United States—he warned of the entrenched corporate complacency and self-satisfaction which would take its toll decades later. He earned the wholehearted respect of his company and, among the world's managers, became a legend in his own time.

Born in Boston, Vin Learson graduated from Boston Latin School in 1931 and was noted for his pride as an alumnus. He earned a bachelor's degree in mathematics from Harvard in 1935 in the same year that he joined IBM's sales force. He was single-minded and aggressive about everything that interested him—especially yachting, which he pursued with ferocity that almost wrecked his relationship with his boss and fellow sailor, Tom Watson—but his ultimate concern was the success and supremacy of IBM, to which he contributed materially.

The ANALYTICAL ENGINE extends condolence to Mr. Learson's wife Gladys and his daughters, Martha Allen, Elizabeth Daniels, Kate Emmerling, and Elaine Schoch, and to colleagues and friends.

# A EULOGY FOR CHARLES EDWIN MOLNAR

**by Wesley Clark**

*[Dr. Charles Molnar, lead developer of the pioneering LINC minicomputer and a legendary figure in the biomedical community, died in Sunnyvale CA on December 13, 1996. It was our intention to publish an obituary for Dr. Molnar that followed this journal's usual practice.*

*We find, however, that an unusually vivid and congenial impression of the scientist and his career is given by this eulogy, the work of Dr. Molnar's long-time colleague and friend, Dr. Wesley Clark. It was originally read at the "Memorial Service Celebrating the Life of Charles Edwin Molnar" held on December 18, 1996, at the Lima Mortuary in Sunnyvale.*

*Dr. Clark advises that this eulogy "was rather specifically intended for Charlie's immediate family and closest co-workers," but while we see his point, we believe that its qualities commend it to a broader audience in any case. He has accordingly permitted us to publish, asking us to note that Donna is Dr. Molnar's wife, Chris and Steven are his sons, and Ivan and Bert are the Sutherland brothers. —Ed.]*

Last night Charlie looked over the draft of what I was planning to "read into the record" this morning, and his eyebrow went up in that inimitable way of his when he wants you to know that, if called to account, your effort will be found lacking.....A bit too heavy in voice and tone.... inappropriate use of the present tense....and....some third uneasiness we couldn't put our finger on....

We agreed to defer to circumstance, and compromised.

All right, Charles. But even though I am surely the most senior of the many colleagues, students, associates and close friends who have had the opportunity—sometimes troubled, sometimes joyful, always challenging and rewarding—to know and to work with you over so many years, I am going to speak today in a very different role: that of the older brother you never had.

Yes, I know; even though I was once made an honorary Hungarian by your wonderful old grandmother—a delightful lady, as I knew her, who would occasionally bring into sharp display the folk wisdom of many generations—I can't, of course, lay familial claim to any such relationship as that of brother, senior or otherwise.

And, yes, right now the most compelling aspect of your rich legacy to us is that we must all, regardless of relationship, somehow come to grips with the immutable fact that our lives can never be quite the same again without your warmth and immediacy. I would not presume to speak for Donna, for Steven, for Chris, for your grandchildren; their loss is beyond any words. I cannot speak for the many of us for whom your wit and your insight and your commanding intellect and your infectious charm so very often made the difference between a day that was ordinary and one that was unusually memorable and important. And no one can speak for your own loss, the greatest of all: the life you were living so energetically and with so much love; the enjoyment of the happiness of your devoted family; the culmination and the re- charging of a long scientific enquiry that was just now productively rewarding you with the fruits of your most recent twenty years of intensive work.

But I can speak of my own wrenching ache as I realize that I can't pick up the telephone and hear your "Got a minute?" and your careful exposition, your chuckle; that I can't give you a giant bear-hug of a greeting and then settle in for the pleasure of an extended visit while we futilely try to rearrange our chairs so that the smoke from my pipe won't so unerringly find its way to exactly the spot you're sitting in.

And yet, Charles, as so many of us have come to appreciate, you were gifted with an extraordinary ability to live a good part of your life in the heads and hearts of those of us who were fortunate enough to be close to you, quite literally sharing our anxieties, our triumphs and disappointments and, yes, even our pain; so it was with me, as with so many of us here today. That is why, in our deepest sensibilities, you are still here with us and always will be; and that is why, after our present grief has receded into a gentle past, your enduring legacy to each one of us will continue to be one of great and sustaining enrichment.

—

The more I try to put into perspective our work together, our long resonant companionship over the many, many years, the better I understand that in my own case this amazing empathy of Charlie's was so strong and so demanding of response that it often amounted to a remarkable sharing of the same psychic space, one that blurred the boundaries between what he was thinking and feeling and what at that moment was in my own head and in my own heart. So often he even knew what I was *about* to think or say or feel, and would adjust his own thought accordingly without a micro-moment's intervention; and I know that the reverse was true as well, though I was, and forever will be, slower than Charlie, whose sensitivity and spontaneity were breathtaking. It was always clear to the two of us that there was more going on here than a simple meeting of minds, a thinking together along similar lines, though of course there was that too. No, there was always something startlingly profound in the quality of our interaction. Both of us understood and accepted this as just the way things were, and were glad. It was the way things sometimes are between brothers; Chris and Steven will understand, and Ivan and Bert.

The unmatchable wit of this man! This morning we've all laughed affectionately over some of the many amusing incidents that will have us telling and re-telling Molnarian anecdotes as long as we have breath to do so. I'll add only a few more, honoring both Charlie and our compromise:

It seems that the handsome young fellow who joined my small group at Lincoln Laboratory in 1957, a fresh product of Rutgers University and now a graduate student at MIT, had apparently been put into a temporary quandary by having to choose between two alternative career paths. He found that he had been offered positions both by my group in advanced computer development and by another, equally interesting group in communications theory and practice. His solution was characteristically efficient if a bit whimsical. He wrote of it later in these words: "I chose between them by a coin-flip. Since it was an important decision, I used a half-dollar."

I have often reflected on the happy circumstance, and in awe of the power of fate, that that toss of a coin—a fifty-fifty chance!—came out the way it did; for I very quickly learned that this remarkable young man not only shared my own deepest convictions about the proper use of the computer as a scientific tool in brain research, but was also on a first-name basis with every known electron in the near universe! The combination was entrancing and I immediately fell under the Hungarian spell.

Charlie set to work mastering the TX-0 and the complicated TX-2 computer as well, which was then taking shape. He brought a degree of self-discipline that was gratifying to behold and quite new to me in my own efforts to deal with complexity. Yet I seem to have been the source of some frustration, especially in those first few months, by refusing to tell him what I wanted him to do—and he admitted to me recently that I seemed to deliberately frustrate him in this way throughout all the years we worked together; but then how do you even begin to direct the professional activities of someone of his extraordinary intellect and wisdom? I was out-classed. Charles, I knew what I was doing!

He did seek my counsel and help when the times occasionally seemed a bit too problematic, his weekend trips back and forth to New Jersey taking their toll as he and Donna worked out their wedding plans, his increasing involvement in campus activities and academic politics, and his military service at a neighboring air force laboratory, all requiring more and more attention. But what are big brothers for if not this sort of thing? I always tried to do my best.

Charlie creatively busied himself with everything, expert on all matters large and small. He catalyzed the growing interaction between the group at Lincoln and the Communications Biophysics group on campus, where he was beginning his doctoral research in auditory neurophysiology. And with no less zeal he tracked down the mysterious bone-jarringly loud bangs that randomly shook the computer room: the TX-2 had the world's first xerographic printer attached to it, an awkward gadget that dumped an endless strip of symbol-infested paper into a separate metal storage bin for later retrieval—a combination that turns out to be a very efficient Van de Graaff high-voltage generator. Each bang was apparently the result of an enormous static charge that built up in the storage bin, every so often producing a bolt of lightning that zapped the printer cabinet. To fix the problem, Charlie strapped the separate pieces together

with a thick interconnecting cable—and then waggishly mounted a prominent sign on the whole thing that read Please Record All Explosions in the Log!

But over the following few years, it was the development of the LINC that was the Grand Enterprise. Some of you know, with the weight of first-hand experience, just how crucial to its success were the sound design and engineering and the enormous effort that Charlie put into it. Perhaps some of you have learned of this work at second or third hand—Steven, you and the personal computer were both launched in that very same historic year. What you may not know, however, is that the pressure of time and commitment stressed all of us on the LINC team to our ultimate limits, and Charlie was no exception.

We were about half-way through the LINC design effort when Charlie made, so far as I am aware, the only misstep of his entire career in which he seriously disappointed himself. It seems that despite his meticulous, back-to-fundamentals analysis of the tortuous chain of eleven phase-reversals in the exacting magnetic tape circuits he had so carefully designed, he had the unexpected shock of discovering, when he tried it all out, that there were actually twelve! We couldn't coax him out of a nearly disabling depression that lasted throughout the following day, though the rest of us were delighted that only one more signal inversion made the whole thing work perfectly—ahead of schedule—and took the rest of the night off to celebrate, waiting in vain for him to join us.

But once he recovered from his deep chagrin, his dependable wit was back in full force. He insisted that at the end of the project any member of the team who had been responsible for a design goof would have to pay off every other member of the team in non-canceling martinis, one per goof. No one quite remembers how our merry final accounting all turned out.

Despite the intensity of that incredible summer of '63, Charlie kept us all on course, solving one difficult technical problem after another and always brightening up momentary hardships with a bit of Molnarian playfulness. On one occasion he stopped by the lab of one of the team's most talented engineers, who had just inadvertently managed to annihilate one of the electronic components on a circuit board for the LINC display unit, from which a wisp of smoke was now wafting into the room. Charlie sniffed the air and proclaimed, "Humph! Smells like a thousand-ohm, half-watt, one-percent resistor!" The engineer's jaw dropped in amazement, for Charlie had identified the fault with perfect precision! Charlie told me later that he'd often blown out that very component in similar circuits while repairing television sets during his college years, and knew it well.

We finished our LINC design work very successfully—NIH still considers it to be among its most important sponsored research accomplishments—only to find that for irremediable political reasons the project had to leave MIT. Senior members of the team visited many prospective new academic homes around the country, met with many university presidents and august boards of directors, and finally narrowed the choice down to the University of Rochester in New York and Washington University in St. Louis. Charlie and I quietly revisited Rochester together one last time to "kick the tires," as he put it, without the encumbrance of official escort. On our thoughtful trip home, the right decision already clarifying itself wordlessly in our minds, Charlie lightened the gravity of the occasion by observing that at the airport there *had* seemed to be many more people leaving Rochester than arriving.

It troubled Charlie that he couldn't yet move with most of the rest of us to St. Louis, not until he had completed both his military tour of duty and his doctoral program at MIT. But so began the years at Washington University, as our small transplanted group put down whatever new roots it could without the benefit of our absent colleague's energy and creativity and wonderfully annealing sense of humor. Charlie did manage to visit us fairly frequently throughout that busy time, though often he seemed just to stand in my doorway with his raincoat on and that perennially overloaded briefcase in his hand, vacillating in visible distress, knowing that he couldn't yet allow himself to become involved in the new effort to develop what we were calling macromodular systems. But soon enough he did complete his service and his doctorate—though the latter might not have been nearly so timely if his thesis advisor hadn't locked the two of them into a hotel room in Cambridge until a final critical part of the dissertation was written.

And so Dr. Molnar rejoined the group at Washington University, accepting a *starting* position as Associate Professor of Biophysics—an appointment without precedent at one of the world's best medical schools. He moved his growing family to St. Louis into what would become his residence for the next thirty years, and embarked on a distinguished academic career that over the years would profoundly affect many activities and many lives.

Progress in our work immediately took a giant leap forward as Charlie began to commit his incomparable intellect and competence to assuring that the macromodule development program's many difficult conceptual and technical problems would be dealt with soundly. That the effort I had thought might take about two years would ultimately take more than seven never seemed to dismay him; he always multiplied my time estimates by what he called Clark's constant, a number he had empirically determined to be fairly close to pi.

These very creative early years were not easy ones for Charlie, whose exacting standards of professional excellence did not permit what to his associates seemed to be merely the occasional pragmatic compromise. "What's the point of having principles," he said, "if you don't use them when the going is rough? That's what you're paying them for!" He divided his time and energies among his many commitments, not only taking charge of macromodular engineering development but also continuing his research in auditory neurophysiology and devoting countless hours to his students, on whose behalf no effort ever seemed too great.

At the end of one of his particularly hard days, I offered to drive the two of us off to dinner at some quiet restaurant so that we could talk things over. Exhausted though he was, his irrepressible wit was very much intact. After he had wearily climbed into my old automobile and fastened his seat belt, he waited for me to establish a destination. "Whither away?" I asked him, and his response was an immediate "Gladly!"

He also began to travel, first to South America and later to Europe and to other parts of the world, developing an international outlook and concern and involvement that, even today, Chris carries forward in his own work in the Foreign Service of the State Department. Everywhere Charlie went he established new friendships and sometimes even new scholastic enterprises—always leaving distinctive Molnarian imprints of insightfulness and challenge along the way.

And Charlie's thoroughness was legendary. When a new Brazilian acquaintance failed for months to respond to repeated requests by telephone and by letter for confirmation of a return visit to St. Louis, Charlie decided on one last try. In a little book on Voodoo he'd bought in Rio, he found a recipe for summoning up a distant friend. It directed him to locate some old rainwater in a tree stump, add a bit of cat fur and a few other unsavory ingredients, and then recite certain prescribed words over the whole thing in the dead of night under a full moon. Charlie told me that he had carried out all these steps very carefully, omitting nothing. Sure enough, the long-awaited confirmation letter arrived two days later. "Well," he explained, "I'd tried everything else."

Charlie also found time to lecture at other universities around the country. My son Douglas told me that at Carnegie Mellon, where Doug was then a student, he had once shown Charlie a drawing he'd made of a complicated asynchronous logic circuit for some function or other. After studying it for a few minutes, Charlie had said only, "Very interesting. How do you know it'll work?" Doug remembers that he shrugged off his disappointment at what seemed to be the casual dismissal of a brash young graduate student, and said that only with the experience of further years did he come to realize just how penetrating the question Charlie had asked really was, which he now sometimes has occasion to put to his own students.

In fact, it was a question to whose answer Charlie had begun to devote what would be more than two decades of research in an intellectual adventure that frequently took him to various parts of the world—most notably and productively, to work with new friends in the Netherlands and in Canada, and ultimately to join and work with both new and old friends here in California.

Following my return to the East Coast, after my own seven-year sojourn at Washington University, Charlie and I remained in close contact with one another through frequent visits and telephone chats. He began to use e-mail as well, although he told me that he occasionally sent off communiqués to imaginary recipients, with content-free messages that were merely strings of randomly chosen five-

letter groups. "Keeps the government eavesdroppers on their toes," he said. But then, even in Charlie's most substantive notes he almost always embedded a treasurable fragment of humor. Just yesterday, a close St. Louis colleague of Charlie's expressed this perfectly: Charlie made you smile and think at the same time.

Over the years we would continue to discuss progress and difficulties in his auditory system research, agonize together over the fate of a disappointing graduate student or about the disposition of research proposals he was reviewing on behalf of distant scientists he'd never met—a responsibility he always discharged with characteristic care and thoughtful analysis and immeasurable hours of personal effort. We joined forces on several worthy projects, once even spending parts of a fascinating year together at Caltech taking advantage of an opportunity to work further with Ivan. We talked not only about the glitch phenomenon and the difficulty of building a sound theory of asynchronous systems, but also about family and about the amusing adventures and misadventures of his cats. We talked about politics and viewpoints and friends, and about the vicissitudes of living in New York; about his travels to Africa to visit Chris and his trips to North Carolina to visit Steven and his new grandchildren; about those snippets of Bartok that he hummed, accurately and often, simple folk melodies whose original words he'd learned as a boy; about a camping trip that went awry but nevertheless generated several new humorous stories to add to his collection.

Yes, Charlie always managed to find some amusing aspect to even the most commonplace of events and situations, which he delighted in recounting and generally illustrated his view that our ordinary world was often slightly askew. Yet to every uncommon challenge that he accepted with serious intent, he brought an integrity and capability that were so great that whenever he told you some projected outcome or course of action had his confidence, you could bet the empire on it—and I did so more than once.

This is the Charles Edwin Molnar I knew, whose students throughout the world are now writing the book he never gave himself a large enough block of time to put together; whose insight and accomplishments gave new life to so very many scientific and technological explorations; whose warmth and generosity and concern for doing what was right suffused us all.....Charles Molnar, who loved people and intellectual challenge and humor and music and complexity and canoeing and cats.....Charlie, the younger brother *I* never had, whom I loved dearly.

So long as my head and heart continue to serve me competently, he and I will still review things together from time to time, and I know I will always be the better for it.

—

By the way, Charles, about your new asynchronous arbiter circuit—there's one subtlety in there I don't quite understand yet. And how do you know it'll work?

Give me a call when you have a minute.

# THE SWAC:
## First Computer on the West Coast

**By David Rutland**

### THE PROJECT

In late 1948 a friend asked me "Say, David, why don't you go out to UCLA and get a job with the new computer project there?" He had heard of the project through the UCLA engineering department.

I immediately left to apply. When I arrived I found Dr. Harry Huskey in charge of the project at the Institute for Numerical Analysis at UCLA. He had been asked by John Curtiss, Chief of the National Applied Mathematics Laboratory of the National Bureau of Standards (now known as NIST, the National Institute for Science and Technology), to build a new type of general purpose electronic digital automatic calculator. This machine was originally referred to as the INA machine but later was named the (National Bureau of) Standards Western Automatic Computer, or SWAC. It was going to be a stored-program computer unlike any other built at that time—a parallel and synchronous machine using Williams cathode-ray tubes for high-speed memory. Of course, all logic elements were built from vacuum tubes designed for radios. The only solid state components were point-contact germanium diodes which were used as gates; but these were unreliable, so many were later replaced with vacuum-tube diodes. I was pleased to join the small team of a dozen people on such a pioneering project. Besides myself there were only two other engineers on the project, Biagio Ambrosio and Ed Lacey.

### SWAC ORGANIZATION

Today's computers are organized in three major sections: the ALU (Arithmetic Logic Unit), the BIOS (Basic Input-Output System), and the memory. This organization hasn't changed since the idea of a general purpose computer was invented in 1945 by the computer pioneers J. Presper Eckert, Jr., John Mauchly and John von Neumann. But in the first computers, like the SWAC, the ALU was divided into the Arithmetic and Control, the BIOS was referred to simply as the Input-Output, and the memory was the memory. Initially all SWAC data was inputted and outputted through an electric typewriter and punched paper tape; later, a punch-card reader and tape punch were added.

### THE ARITHMETIC UNIT

The SWAC was intended to be the fastest computer in the world, and the design of the arithmetic unit was crucial to that speed. Huskey decided to work from the Whirlwind computer, sponsored by the Air Force and then being built at MIT, which had the fastest arithmetic unit then operating so it could be used in real-time operations for tracking friendly and enemy aircraft. The arithmetic unit added bits in parallel, like those in modern computers, rather than serially as in the UNIVAC and other early computers. It was therefore many times faster. The Whirlwind used a 16-bit word, while the SWAC was to use a 37-bit word, so Lacey had to redesign it for even faster operation. The final design was very rapid for its time, taking 5.6 microseconds to add two 37-bit numbers.

The arithmetic unit used 22 tubes for each bit which were mounted on two long thin chassis, each about 3 feet long—12 tubes on one chassis and 10 on the other. These two chassis plugged one above the other into the electronic racks. The whole 37 bits of the arithmetic unit, 814 tubes, used 74 chassis and occupied the full 12-foot length of the SWAC from one side to the other. All this for what now wouldn't begin to fill a silicon chip!

### THE MEMORY UNIT

Prior to taking on the SWAC project, Huskey had spent a year in England at the National Physical Laboratory. While working on the ACE computer with Alan Turing, he had visited the computer project at Manchester University under the direction of F. C. Williams. Williams had invented a cathode-ray tube memory for his computer that stored data serially. Curtiss and Huskey decided that a Williams tube array modified for parallel operation, like modern RAM, would meet the speed requirements of the SWAC. It was able to store 256 words (about 1200 bytes) using 37 5-inch CRT's displaying 256 bits each; with auxiliary circuits, the array occupied a space seven feet high, two feet deep and five feet wide. That seems a lot of space, but the only alternative at that time would have been to use a vacuum tube for each bit,

9472 tubes! The CRT storage represented a giant step forward in 1950, even though a tiny RAM chip can now store thousands of times as much.

The SWAC was a synchronous machine, timed by an electronic "clock", like today's PCs. A machine cycle of 16 microseconds required a 125 kHz clock. Like most modern RAMs, the CRT memory had to be refreshed; this was done during half of each cycle while the other half was used to random access the memory. Two numbers to be added were each read from memory to the arithmetic unit, which took two cycles, one for each number. At the end of the second cycle, the arithmetic unit would add the numbers, and on the third memory cycle, the result would be written back to the proper addresses on the CRTs. On the fourth cycle the next instruction was read from memory to the control unit. The resulting total addition time was 64 microseconds, or 15,625 instructions per second. Multiply and divide took 384 microseconds, 2,600 per second. Floating point arithmetic had to be programmed if used at all. By any modern measure of speed in floating point instructions per second, the SWAC was terribly slow; but in 1950 it was the fastest computer in the world.

## THE CONTROL UNIT

By the time I joined the project, preliminary design of both the memory and arithmetic units were under way. Ambrosio was responsible for the memory and Lacey for the arithmetic. By default, I inherited responsibility for the control and input-output units. The input-output consisted of the control console, an electric typewriter and a paper tape reader and punch.

Huskey planned to keep the instruction set to the minimum which would allow the SWAC to solve problems efficiently. This he was able to do with only 13 different instructions defined by 4 bits. Together with the four 8-bit addresses—two operands, result, and the next address for branching—they made up 36 bits of SWAC's 37 bit word.

In designing the control I made extensive use of a delay line that was developed for radar use and was commercially available from Raytheon. It was a specially constructed co-axial cable with a tightly wound helical inner conductor. By sending the pulses from the master clock down this cable, we could delay them so they would appear at the proper times to sequence the operations of the SWAC.

I also had to design the circuits for flip-flops and gates, choosing the tube types and calculating the resistor and capacitor values. These then had to be built and checked for reliability.

Finally I was able to sit down and draw the detailed diagrams of the control unit. These diagrams showed how each control pulse was generated, so that the pulses would sequence the operations exactly according to the rules of arithmetic that had been carefully laid out by Huskey. My hand-drawn sketches covered several desk-size sheets of paper. I remember spending many days and evenings checking and double-checking the pulse sequences against Huskey's specifications. I also had to be sure that pulses were properly timed to operate the memory and arithmetic units within their specifications. The delay cables saved many tubes that would have been required if I had used a state counter. Even so I ended up using "only" a few hundred tubes in the control unit.

## HOW IT WAS BUILT

Everything was hard-wired in the SWAC, which was built before printed circuit boards became available. Every tube plugged into a socket with seven or eight connections. The 2700 tubes had over 20,000 connections on their sockets alone, each wired by hand. The wiring job was compounded further by the fact that each wire or component attached to the sockets had a similar number of connections on its other end. The soldered components included 3700 crystal diodes which were very temperature sensitive; great care was needed in installation so that the heat wouldn't damage them. Finally, the chassis were interconnected by large cables, each containing many hundreds of wires, that ran up, down and between the racks. Our small team of skilled technicians and women assemblers had their hands full for a solid year wiring up the SWAC.

The memory occupied the center of the front row of racks, just a few feet in front of the arithmetic unit. A rack to the right of the memory housed the auxiliary memory circuits, the master clock and input-output units. A rack to the left of the memory contained the control unit. The two rows of racks—front row with memory and control, and

back row with the arithmetic unit—were closely aligned so that the wires carrying the 37 bits of information from the memory to the arithmetic unit would be as short as possible. The overall depth of the SWAC was a little over four feet, so it occupied about 50 square feet of floor space. This did not include the power supplies, which were located on a wall outside the building.

## CHECKING IT OUT

For those days the pulses in the SWAC were very fast and short, about 100 nanoseconds, giving a bandwidth of only 10 MHz. The pioneering Type 511 Tektronix oscilloscopes were hard pressed to display such short pulses. Low accelerating voltage made them hard to see and all viewing had to be in the dark or using a viewing hood. During the summer of 1950, while the SWAC was nearing completion and the pressure was on to get the control unit working, I spent most of my waking hours looking intently at the pulses on the oscilloscope screen—so intently for so long that I would still see them dancing before my eyes as I went to sleep.

## THE DEDICATION

Toward the end of 1949, the projected dates of completion of computer projects came and passed so often that John von Neumann came up with a law: *A computer will be completed six months after the day on which you happened to ask.* But a day had to arrive when the computer was pronounced completed and a dedication ceremony scheduled. Usually "completion" required that the machine run a program for a short time, say a minimum of half an hour, without error. When this criterion seemed within reach, the SWAC dedication was set for August 17, 1950.

Dedication day began with the usual speeches. The Director of NBS, Edward Condon, came out from Washington to be the first speaker. He was followed by Air Force Colonel F. S. Seiler, Chief of the Office of Air Research, a major contributor of funds for the SWAC, who had flown out from Wright Field. Then came Dr. L.N. Ridenour, Dean of the Graduate School of the University of Illinois, and finally, the Chief of NAML, John Curtiss. Lastly our leader, Harry Huskey, now officially Chief of the INA Machine Development Unit, described the SWAC and gave a short demonstration. We all held our breath, but the hours of work had paid off and the SWAC did run.

The SWAC continued to run, first at the INA and then after it was moved to the Engineering Department at UCLA. It was finally dismantled in December 1967, 17 years after its dedication, and at that time the oldest of the pioneer computers still in operation.

## PROBLEMS SOLVED BY THE SWAC

The day after the Dedication a symposium gave a glimpse of what sort of problems were being planned to run on the SWAC. First there were the problems in pure and applied mathematics, including statistics and number theory. Then there were a host of papers on solutions to engineering problems. Scientists and mathematicians gave papers on topics such as the flight of an airplane when it starts a turn, nuclear reactor physics, perturbations of an earth satellite, rocket engine research and a problem in astronomy. The computer was going to help solve all these problems. We could scarcely have imagined then that computers would not only solve scientific problems, but eventually help millions of people in homes and offices.

Some months after the dedication, the SWAC was running well enough to engage in problem-solving for hours without error. The mathematicians at INA, like mathematicians throughout the world, were interested in prime numbers. As primes get bigger they keep getting further and further apart and are more difficult to find. Each number must be tested to see if a smaller number will divide into it exactly, and the larger the potential prime being tested, the more tests are necessary. Mathematicians who deal with the theory of numbers are very interested in finding large ones, so they programmed the SWAC to start with a known large prime and find a larger one. It was set to this task and, after hours of calculating, it came up with a prime number larger than any yet known. This was an achievement of real interest only to mathematicians, but to all of us that had built a computer from "scratch" in less than 18 months, it was a milestone in the history of computers.

## THE SWAC AND THE PC

With its small memory the SWAC was limited to calculating one problem at a time. Like an astronomer using a telescope, each programmer requested time to run his or her problem alone. In this way it was a "personal computer" like our own PCs. It was also more like our PCs than other computers of its time in that it had a parallel arithmetic unit, parallel RAM, and ran synchronously from a clock. In 1953 a 4,096-word magnetic drum, forerunner of our hard disks, was added. Although CRT displays for operators were not then practical, the SWAC's data could be entered and typed on its electric typewriter. It therefore had all the basic features of our PCs.

This similarity extended beneath the skin. The SWAC and the PC are *stored-program computers* with their programs stored in the memory with the data. They are both "von Neumann machines" carefully advancing through their memories, looking for instructions, and executing them one at a time.

Almost fifty years later the apparent changes in computers—their vastly smaller physical size, their great increase in memory and speed, and, of course, modern software—have transformed them almost beyond recognition. Yet their fundamental principles remain the same and have taken form in the most flexible machines that man has yet invented, performing an uncountable number and variety of tasks. Whenever we see or use a computer we should all remember three great men, the inventors of the computer: J. Presper Eckert, Jr., John Mauchly and John von Neumann.

*David Rutland can be reached by e-mail at wren@peak.org. For much more information about the SWAC and how computers were invented, browse to http://www.peak.org/~wren, or obtain his book "Why Computers are Computers: The SWAC and the PC," published by Wren Publishers.*

## Book Review: THE COMPLETE COLLECTOR'S GUIDE TO POCKET CALCULATORS

**by Guy Ball and Bruce Flamm**

Wilson/Barnett Publishing, 1997
204 pp., 500 b&w illustrations
ISBN-1-888840-14-5
$US23.95 paper ($20.95 to ENGINE readers)
plus sales tax if ordered from within CA

**Reviewed by Erich W. Schienke**

My first glimpse of *The Complete Collector's Guide to Pocket Calculators* propelled me back through time to a point when the simple four-function calculator—a "way-back machine" if ever there was one—represented the pinnacle of portable computing. Product of "new math" and hand-held electronics, I have literally never used a slide rule; but I have fond memories of my first calculator, the "Quiz Wiz," which didn't have a numerical display at all, just a red and green LED to signify an incorrect or correct answer. When I was nine I learned Reverse Polish Notation (RPN) on my father's HP-41CV, a truly cool tool which I learned to love. I could play a sub-hunt simulation with a bar-code reader. I wrote small programs which would beep or play other little tricks. My math career blossomed with my discovery that I could program in all those ridiculous equations I was supposed to remember... Eventually *they* caught on and I was compelled to use a "standard" (non-RPN) calculator for tests, which I bitterly resented...

One look at *Pocket Calculators* will remind you, as it did me, that the pocket calculator has become a ubiquitous accessory, in both school and daily life, for millions if not billions of people. The book's first startling artifact is a timeline of the evolution of the pocket calculator during its "heyday"—the early sixties to 1979—which will have you saying "Oh, right, once upon a time there were no pocket calculators....although I don't really remember not having one..." The first one you had is almost certainly one of the 1,500-plus calculators from over 220 different manufacturers covered by the "guide" part of the book. As well as models, pictures and prices, *Pocket Calculators* contains plenty of history, and some great reprints of early calculator advertisements.

The guide is sorted alphabetically by manufacturer and contains all known information about every calculator the authors have been able to find. Models are listed chronologically under each manufacturer and about a third of the models are pictured. Subordinate information can include functionality, display, special functions, size, original price, year of the model, desirability, and collector's value—although this, with old calculators as with old micros, is only rule-of-thumb since almost any item, regardless of "catalog" value, can with luck be picked up at a flea market or swap meet for a few bucks.

Between the two of them, Ball and Flamm have decades of calculator collecting experience, and it shows. As far as I can tell, the pictures in the book are of items from their own massive collection of calculators, and include some cool pieces; I can't decide whether my favorite is the Kosmos Astro, a bubble-dome four-function-and-biorhythm calculator, or the Star Trekulator with its strange flashing LEDs and sound effects. (My major criticism of the book, though, is that these low-contrast grayscale photos sometimes lack detail and could have gained a lot with minor touch-up. I also wish that the dates of manufacturing were more complete, but Ball and Flamm obviously used all the information they had. This book can't be faulted as the pioneering work in a field still ripe for research.)

The book ends with a series of reprinted calculator ads from the sixties and seventies, when features like square root were major selling points. Sinclair advertised its Scientific as heralding the demise of the mechanical slide rule, which it did. HP modestly called its Model 80 "the most revolutionary financial device of our time," and for a mere $356.75. One ad even speaks of the "New Calculator Revolution." I wish the authors had included more of these ads, which are telling testimony to the wild popular reception for these marvelous little devices.

Whether you use—or used—HP or TI, Sinclair or Bowmar, *Pocket Calculators* belongs in your library of interesting books about collectibles. The Calculator Revolution, no longer "New," is here to stay and well served by this remarkably comprehensive reference.

## CORRECTION

Woops! When we printed Erich's review of *Where Wizards Stay Up Late* in issue 3.4 we forgot the publication information. If we had a buck it would stop here, so here's the heading as it should have read:

**WHERE WIZARDS STAY UP LATE**

by Katie Hafner and Matthew Lyon
Simon & Schuster, 1996
304 pp., b&w illustrations
ISBN 0-684-81201-0
$US24.00 cloth

## LETTERS

### REPLY TO SELL ON SUN

Some comments on John Sell's comments:

1. Only two Sun systems ever used 256K SIMMs: the 3/60LE, a cut-down version of the 3/60 which used mostly 256K SIMMs but had some 1M slots; and the 4/1xx, which could take either 256K or 1M SIMMs. Of Sun-3's, the only ones that took SIMMs at all were the 3/60 (and LE) and the 3/80, both latecomers. Everything else used custom memory boards. Sun only really started using SIMMs with SPARC systems, and even then some still used custom memory boards.
2. Sun's 9U VME boards are 15+" square as well.
3. Fujitsu Eagles have 10" platters and are typically encountered in 19" rackmount chassis. The complete chassis, which has a hefty 120VAC power supply in it, weighs about 150 pounds. They have a formatted capacity of about 400M. There was also a Super Eagle with a formatted capacity of about 600M. As far as I know, all Eagles were SMD units.
4. On the one hand, it is in fact true that most Sun-3's were limited in memory capacity. On the other hand, that was generally because of form factor or the capacity of memory boards that Sun supplied, not due to limitations inherent in the architecture. The 3/50 had no provision for memory expansion and hence was officially fixed at 4M, period. Third-party daughterboards could get it up to at least 12M.

The 3/60 has 24 SIMM slots and takes only 1M SIMMs. Period. The 3/60LE has the same number of SIMM slots but most of them take 256K SIMMs, so it has a max of 12M. The 3/1xx series and the 3/75, which was a 1xx CPU in a two-slot pizzabox, were officially limited to 16M due to the limited capacity of Sun memory boards. I know 16M third-party boards were available, but I'm not sure if more than one could be used. The 3/80 could, with a late-model ROM, accept 4M SIMMs for up to 64M. The real memory monsters were the 3/2xx and 3/4xx series. In their initial incarnations, they were limited to 32M because the memory boards only held 8M, and only four of them could be used. Later on, however, compatible 16M and 32M boards became available, and up to six can be used, if your chassis has enough slots with P2 bussed for memory. I don't know if anybody has ever assembled a Sun-3 with that many, but within arm's reach I have a 3/2xx configured with three 32M boards for a total of 96M of memory.

*James W. Birdsall*
jwbirdsa@picarefy.picarefy.com

## INFO WANTED: LISA, VICTOR, COMPUCORP

I have three pieces of California hardware about which I would be glad to receive any further information.

First, an Apple Lisa computer—one of three I have—which was upgraded to a Lisa 2/5; I received it with full documentation including upgrade instructions which give detailed steps for the removal of the original bezel, replacement of the Twiggy drives with the 400k 3.5" Sony, and installation of the replacement bezel. By checking in the Apple Module Identification manual, I have verified that this is indeed an upgraded original Lisa. It has a Profile external hard drive which is a 5MB Seagate ST-506 (another piece of California computer history, from Shugart Associates). Question. Did Apple require Lisa users given free upgrades to Lisa 2 to return the old bezel and Twiggy drives to Apple? I would like to restore my upgraded Lisa to an original Lisa. If Apple didn't require the return of the Twiggys and bezel, then maybe my quest has a slim chance.

Next, I have a Victor Technologies model 420 with one 360k floppy drive and an internal 10 megabyte hard drive. The green monochrome monitor receives its power through the same cable that supplies video signal. The keyboard is detachable via a RJ-45 jack.

I have partially dismantled this PC, in order to reseat cables and check the manufacturer of the hard drive; I susupect it is a Seagate drive but it is so buried in the system unit that I haven't yet figured how to pull it. This unit isn't intuitive regarding servicing, that's for sure. I have repaired PC's for a living for the past 12 years and never saw one of these Victor models before.

I would greatly appreciate any information on the marketing history of this unit. I have visited Victor Technologies' web site but they scarcely mention this period of their history. I found some helpful information about Victor in a book entitled *The Computer Entrepreneurs*, but I am more interested in knowing what was produced, when it was produced, and so on.

Lastly I have what looks like an overgrown calculator, a Compucorp® 340 Statistician manufactured by Computer Design Corporation in Los Angeles. The rear of this unit reads Compucorp Microcomputer and has operating instructions which imply that some models were available with a programmable option. The unit requires 7 volts via battery or AC adapter. Again, any information on this unit, its cpu, its period of manufacture, etc., would be greatly appreciated. The individual I bought it from told me that it once lived at the University of Virginia in Charlottesville. I have no more info on this and have searched the Web in vain.

*Marty Mintzell*
5635 Heming Avenue
Springfield, Virginia 22151
703-569-2380
marty@itgonline.com

[Marty,

Our Lisa guru, who was a Northern California Apple repair parts distributor for many years, says that Apple did indeed require the Lisa One bezel and Twiggys to be returned to their stock in exchange for the Lisa 2 bezel and the Sony drive. The majority of Lisa Ones were upgraded because

their owners were so frustrated with the erratic Twiggys. If you could find the bezel and drives, which are quite scarce, the deconversion would be more than worth doing; according to our latest information, the street price of a Lisa One is ten to twenty times the street price of a Lisa 2. Be aware, though, that the question of getting the Twiggys to work is an entirely separate matter.—Ed.]

### AES DATA: ANYTHING?

I just got a very old computer. On the front it says AES 7100, on the back AES Data Inc., Montreal Canada, Model 230. As far as I can find out from the handbook, it dates from the early eighties. The screen is built onto the computer on one side, and two 5.25" floppy drives on the other. I wonder:

1) Does anyone have the technical specifications of this machine? (I tried to find AES Data Inc. on the net, without success) 2) Does anyone have any software for it? It came with some floppies with a "wordprocessor", which you have to put in when booting; it directly boots into the program, and if you leave it out or put a wrong diskette in, you get either a black screen or all 1's. 3) Does anyone know if AES Data Inc. still exists, maybe under another name?

Any help greatly appreciated,

*Sergej*
c/o info@chac.org

## NEXT ISSUE / COVER ART

An interview with one of the true pioneers of semiconductor research in the Valley. Who? Ah, we'll keep you guessing! And lots more, of course.

*Cover:* The published architectural block diagram for the Intel MCS-51 (8051) microcontroller.


## SPOTTER FLASH

*Computerworld* for January 20, 1997 featured "Archival Rivals," by Lisa Picarille, which highlighted the different approaches of the CHAC, the SFCM and the Computer Museum History Center to creating a context for the display of artifacts.

David Noack lauded our Web site's "rich mix of resources....detailed information about personal computer hardware, software, computer museums, and even computer folklore" in his article "The Origin of PCs," which appeared in *Internet World* for February. This article also says nice things about many of your friends and ours, and includes a whole bunch of engaging URLs.

In January MSNBC's *The Site* broadcast "Silicon Valley Mystery Tour," a seven-minute gallop past the Valley's semi-sung or unsung landmarks—Dave Packard's garage, the adult video store that was once the Byte Shop, Shockley's fab now a home theater demo room. We even played Pong in Al Alcorn's garage! Our only regret is that the world will never see hours of great shots produced by two days' shooting....

## THANKS TO....

Wesley Clark for allowing us to publish his eulogy to Charlie Molnar.

Frank Freeman and Evelyn Sprada for their generosity and support.

Judy Goddess for counseling on fundraising.

Stan Kibby, Gary Mercer, Craig Miller and Jenn Rogers for great work on the SITE shoot; and Al Alcorn and Len Shustek for their participation.

Frank McConnell for his donation.

David Noack, Bob Parks, and Lisa Picarille for writing.

## ADDRESSES OF CORRESPONDING ORGANIZATIONS

Amateur Computer Group of New Jersey (ACGNJ), P. O. Box 135, Scotch Plains NJ 07076. Joe Kennedy, president.

Australian Computer Museum Society, PO Box 103, KILLARA 2071, NSW, Australia. Michael Chevallier, secretary.

Charles Babbage Institute, 103 Walter Library, 117 Pleasant Street SE, Minneapolis MN 55455. Bruce Bruemmer, archivist.

Commercial Computing Museum, 220 Samuel Street, Kitchener ON N2H 1R6, Canada. Kevin Stumpf, president.

Computer Conservation Society, 15 Northampton Road, Bromham, Beds. MK43 8QB, UK. Tony Sale, secretary.

The Computer Museum History Center, Box 3038, Stanford CA 94309-3038. Dag Spicer, collections manager.

*The Computer Journal*, P. O. Box 3900, Citrus Heights CA 95611. Dave Baldwin, editor.

Computer Preservation Society (Inc.), Ferrymead Historic Park, 369 Bridle Path Road, Christchurch, New Zealand. Abraham Orchard, secretary.

East Bay FOG, 5497 Taft Avenue, Oakland CA 94618. Tom Lewis, president.

Hewlett-Packard *Journal*, Hewlett-Packard Company, Box 51827, Palo Alto CA 94303-0724. Richard P. Dolan, editor.

Historical Computer Society, 2962 Park Street, #1, Jacksonville FL 32205. David Greelish, president.

International Association of Calculator Collectors, Box 345, Tustin CA 92781-0345. Guy Ball, Bruce L. Flamm, directors.

IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos CA 92640. Bob Carlson, director.

Lexikon Services, Box 1328, Elverta CA 95843. lexikon2@aol.com. Mark Greenia, director.

Perham Foundation, 101 First Street #394, Los Altos CA 94022. Don Koijane, president.

San Francisco Computer Museum, Box 420914, San Francisco CA 94142-0914. Erich W. Schienke, manager.

Santa Clara Valley Historical Association, 580 College Avenue, Palo Alto CA 94306. John McLaughlin, director. Note change of address.

*Vacuum Tube Valley*, 1095 E. Duane Avenue, Suite 106, Sunnyvale CA 94086-2601. Eric Barbour, staff editor.

## GUIDELINES for DISTRIBUTION and SUBMISSION

To conserve space, guidelines for distributing and submitting to the ANALYTICAL ENGINE have been moved to our Web page, http://www.chac.org/. If you do not have Web access and you need a copy of these guidelines, please e-mail a request to rules@chac.org or snail-mail to the address below.

**The ANALYTICAL ENGINE**
**Volume 4, Number 1,**
**Winter 1997**
**ISSN 1071-6351**

journal of the Computer History Association of California, is published four times a year at Palo Alto, California.

Basic, domestic subscriptions are $35, with $25 deductible as a charitable donation. For details of institutional, international, and low-income subscription, contact the Association at:

**4159-C El Camino Way**
**Palo Alto, CA 94306-4010 USA**
**Internet: engine@chac.org**
**WWW: http://www. chac.org/**

## NINES-CARD

### HACKER TAROT CARDS
### (origin lost in the miasma of USENET)

0. **The FOOL:** A manager using a MIPS R10000 to run a screensaver.

1. **The MAGICIAN:** A hacker with a Mac, a Pentium box, a SPARC, and an Alpha on the table in front of him—all running the same program with the same GUI. An infinity sign is over his head.

2. **The HIGH PRIESTESS:** A woman holding the Documentation, closed and concealed. The crescent moon is showing on an Indigo behind her.

3. **The EMPEROR:** Steve Jobs sitting on a NeXT Cube, holding an optical disk vertically in his hand.

4. **The EMPRESS:** A secretary with a NeXT pizzabox.

5. **The HEIROPHANT:** Bill Gates with two flunkies kneeling before him, their faces averted, offering him floppy disks. He wears a laptop computer on his head.

6. **The LOVERS:** A PowerMac and an IBM PowerPC sit exchanging software as an angel bathed in glory regards them.

7. **The CHARIOT:** A man in a chariot, hurtling up an exponential curve, drawn by the twin sphinxes of Technology (black) and Culture (white).

8. **STRENGTH:** A woman holding the entire design and implementation of Microsoft Excel in her mind as she corrects the final error. An infinity sign is over her head.

9. **The HERMIT:** An old hacker, white-bearded, burns the midnight oil; its Star-of-David flame illuminates his keyboard.

10. **The WHEEL OF FORTUNE:** A rotating wheel. Cray is on the side going down, despite good technology; Smalltalk is opposite it, and C++ is sitting on top. Four winged beings—a mouse, a turtle, a dogcow, and a human—look on.

11. **JUSTICE:** A cold-faced woman holds a calculator in one hand and a delete key in the other.

12. **The HANGED MAN:** A programmer is tied by his ankle to a cable duct. His phase is completely shifted; he awakens at sunset, he sleeps at dawn. His monitor is reverse-video. He programs on, flawlessly, oblivious to his circumstances.

13. **DEATH:** A skeleton wielding a scythe surveys a parking lot, on which are scattered PDP-11/23's, Apple Lisas, an IBM 360/91, a Xerox Alto, and many other machines.

14. **TEMPERANCE:** An angel stands with one foot on her chair and one on the floor, as she copies files from one disk to another. A cursor blinks from her chest.

15. **The DEVIL:** The goat-headed Lord of the Pit stands on a pile of Windows manuals, holding an inverted torch in one hand. Two humans, male and female, are in chains at his feet.

16. **The TOWER:** An ivory tower is struck by a bolt of lightning. Two robed figures, denied tenure, are hurtled to the ground.

17. **The STAR:** A Mac is running its "warp" screen saver, in a transient fragile moment of peace.

18. **The MOON:** A wolf and a jackal are typing at two PC's. A crayfish crawls out of a pool, offering suggestions that may ultimately prove deadly. The moon shines through a window.

19. **The SUN:** A naked child riding a winged rocking horse programs clever applications on an HP PA/RISC workstation.

20. **JUDGMENT:** An angel blows a trumpet; all over the net, Web pages arise, to be rated Cool or not.

21. **The WORLD:** A woman dances in the sky, unclothed, unencumbered, in a ring of clouds, a 3-D mouse in each hand. The four winged creatures from the Wheel of Fortune surround her.

# CONTENTS

US$5.00 UK£3.50 10DM ¥700

CHAC, 4159-C El Camino Way, Palo Alto CA 94306 USA FIRST CLASS MAIL

# SO YOU WANT TO BUY YOUR FIRST COMPUTER

**....again? That IMSAI, Apple ][e, or Poly-88 that changed your life a few years ago?**

Well, you've come to the right place—or, more exactly, it's come to you. Beginning with issue 4.2, the **ANALYTICAL ENGINE** will accept classified ads from buyers and sellers of fine legacy hardware, software, archives and ephemera. Everybody wins; *we* launch a modest pilot project to incorporate advertising in the magazine, while *you* present your garage treasures to one of the world's most technically sophisticated audiences of collectors. And, of course, the artifacts pass from loving hands to loving hands without getting rained on in a parking lot....

Place your ad at only US$10 for a 2"x2.5" (50x63mm) space. Four insertions are a bargain at $8 each. Give the **ENGINE**'s sharp-eyed, sharp-minded readership a chance to un-closet exactly the micro, or whatever, that lights *your* LED's; e-mail your ad copy to classads@chac.org or snail-mail it to the address at the top of this page. Thanks!

**Computer History Association of California**