7

THE END OF AN ERA 1976–84

The legendary Zohrab Kaprielian – Football game at the Rose Bowl – Creating a top department at USC – How Harvard grants tenure – The first book on concurrent programming – Doctor technices – Surviving the executive vice president – Designing the Edison multiprocessor for Mostek – United Technologies kills the project – Let no man complain to me – Brush fire and mud slides in Altadena – Magical simplicity – What we achieved.

In the fall of 1975, I started looking for a permanent job as a tenured professor. I was encouraged to apply for a new professorship in datalogy at the Technical University of Denmark, but the deadline was too short for Milena and me to decide to fold our tent and return to Europe.

I visited universities in Washington, Utah, California, Colorado, Wisconsin, North Carolina, New York and Ontario. At each university, I stayed for two days, gave a talk and spent the rest of the time meeting individually with local faculty members and joining them for lunch and dinner. These were interesting, but exhausting trips since I was constantly being evaluated by my peers.

At the beginning of 1976, these universities received letters of recommendation from Edsger Dijkstra, Don Knuth, Butler Lampson, Bill Lynch, Harlan Mills, Peter Naur, John Reynolds, and Niklaus Wirth. Of the five offers I received, I chose the University of Southern California (USC) in downtown Los Angeles.

After living in rented houses for five years, Milena and I had finally bought an idyllic ranch house in Altadena. My decision to join USC was heavily influenced by our desire to stay in Altadena and create a home of our own where our children could grow up.

At the time, computer science at USC was just a program in electrical engineering. The tenured faculty consisted of Seymour Ginsburg, an early

pioneer in formal languages, and Ellis Horowitz, who was becoming a prolific writer of textbooks. In addition, there were a few assistant professors. With such a small faculty, USC had a unique opportunity to develop a first-rate department from scratch. But, to do that, they would need new leadership. The program was headed by Jack Munushian, one of the nicest people I ever met. However, as a professor of material science, he was not an effective leader of computer science.

Before accepting an offer from USC, I met with the legendary Zohrab Kaprielian, who had turned USC into a major research institution. Kaprielian joined electrical engineering in 1958. After four years, he became chairman of electrical engineering. By 1970, he was dean of engineering, and, two years later, promoted to senior vice president. Shortly thereafter, he became executive vice president of the university.

Every time Kaprielian was promoted to higher office, he kept all his previous positions. When I first met him, he was in his early fifties and was clearly in charge of the university. Solomon Golomb, professor of electrical engineering at USC, commented on the five levels of administration between himself and the president of the university: "All of them were Zohrab Kaprielian. We operated on the principle of one man, one vote, and Kaprielian was the one man who had the one vote."

Kaprielian's visionary leadership showed what Regnecentralen's Niels Ivar Bech might have achieved if he had lived in the United States. But, in contrast to Bech, Kaprielian was a ruthless politician who made many enimies, which didn't seem to bother him (as they say in Washington, "If you want a friend in this town, get a dog!"). I remember an instance, where a delegation of university administrators and senior faculty from USC visited China. One of the participants was the head of the university's news bureau. For some reason, this small office was not controlled by Kaprielian. As soon as the plane left Los Angeles International Airport, Kaprielian announced that he had "reorganized" the news bureau. From now on, it would work directly under his office. When the plane landed in Beijing many hours later, the announcement had already appeared in the Los Angeles Times, and it was too late for the former head to do anything about it.

In 1968, Kaprielian pioneered distance-education by establishing an instructional television network under Jack Munushian's leadership. The broadcasting of lectures via television made it possible for engineers to complete graduate class work at the corporate offices of Hughes Aircraft and other companies. Today, USC broadcasts over one hundred engineering courses

by satellite and internet webcast to 1,000 graduate students.

Kaprielian had an uncanny talent for recognizing opportunity and making fast decisions. In 1972, he was approached by Keith Uncapher, who was director of the computer science division at RAND Corporation in Santa Monica. His research on packet-switching led to the military's Arpanet and then the Internet. He was now trying to create a university-based research institute in Southern California. The University of California at Los Angeles (UCLA) told him it would take 18 months to work out a deal. However, Kaprielian jumped at the chance, and, within a week, Uncapher was able to start the Information Sciences Institute (ISI) at USC.

My meeting with Kaprielian took place in his large office, which was kept in semi-darkness by heavy drapes. He was a short man who looked supremely confident, with a smile that was both friendly and slightly devious. He spoke so softly that you had to be very quiet to hear what he said. He asked what it would take to make me join USC. He immediately offered me a tenured position as full professor of computer science and agreed to fund a minicomputer lab with a PDP 11/55 computer, so I could continue my work with Concurrent Pascal without interruption.

He would have preferred to let computer science continue for a while as a program headed by his friend Munushian. However, under that scenario, I could not see USC becoming a national leader. He then agreed to let my appointment coincide with the establishment of a computer science department chaired by me. The department would be housed in a new building financed by Henry Salvatori, a prominent Southern California industrialist. At the end of my fifteen minute meeting with Kaprielian, I came away with a promise of two endowed chairs and fifteen professorships for computer science.

On September 1, 1976, I started working at USC. Walking around campus with my son during the Christmas break, we met Kaprielian, who gave us two tickets to the New Year's college football game at the Rose Bowl Stadium in Pasadena, which seated 100,000 spectators. At that time, these tickets were selling for $100 each on the street. My ten-year old son, Thomas, who had grown up with American football, loved every minute of the game, as we watched the USC "Trojans" defeat the Michigan "Wolverines" 14–7. As a young man in Denmark, I enjoyed watching soccer and tennis on TV. But I didn't have a clue about the rules of American football. To me it looked like each team first stuck their heads together and shouted something. Then the two teams proceeded to tackle each other and fall. The cycle of

shouting and stumbling then started all over. Having an inquisitive nature, I asked my son what was going on. Thomas, who was embarrassed by his ignorant immigrant father, studiously ignored me. Twenty years later, he gave me "The Complete Idiot's Guide to Understanding Football." I struggled bravely with it before giving up. You have to grow up with a sport to appreciate the subtleties of the game.

It was now up to Ginsburg, Horowitz, and me to create a top department. Although we were hardly Caltech, I decided to act as if we were. I am sure my colleagues thought I was nuts when I said that "North America is too small a continent for our recruiting!"

We started systematic recruiting of new faculty by asking leading departments in the United States and Europe to name their best PhD students. Based on letters of recommendation from their advisors, we invited perhaps five out of twenty candidates to visit USC and give seminars about their research and be interviewed by the faculty. We then asked the dean (who, of course, was Kaprielian) to send offers to one or two candidates.

By talking to colleagues at other universities, we got a clear idea that all of us were competing for the same top candidates. The department spent years selecting faculty candidates in this time-consuming fashion. Most of them, of course, accepted offers from universities like Stanford, MIT, and Berkeley. Nevertheless, there were limits to how many positions these few universities had to offer. As long as USC tried a little harder than other departments, we would usually hire one person per year.

I remember calling Tony Hoare at Queen's University in Belfast, who recommended one of his research associates, Nissim Francez. With Hoare and others, he had done research on the semantics of concurrency. He would eventually publish a book about the tricky question of fair scheduling of concurrent processes. In 1977, Nissim became an assistant professor at USC. He is now a professor at the Technion in Israel.

In 1980, we were fortunate to attract Len Adleman, who was an assistant professor at MIT. In 2002, Rivest, Shamir, and Adleman received the Turing Award for their invention of the RSA cryptosystem.

At one point, we offered one of our endowed chairs to Zohar Manna, who was being considered for tenure at Stanford. My senior colleagues felt that we had no chance of attracting him to USC. My response was that that was for *him* to tell us! So, we wined and dined Zohar, who turned us down when Stanford offered him tenure. I believe I was correct in assuming that Zohar would add to our reputation by telling his colleagues to watch these

ambitious guys at USC.

Now, hiring first-rate people is difficult enough. But evaluating them for tenure is even harder. If you start promoting weak researchers, you will some day find that most of your faculty belong to this category. Once that happens, a department has no future. Unproductive researchers naturally judge other researchers by their own standards and often see outstanding researchers as a threat to their own local reputation and influence. So average researchers have a tendency to hire and promote other researchers of the same kind.

To avoid falling into that trap, we did not consider anybody for tenure unless they had letters of recommendation from the top ten people in the world in their field. During the first seven years, we only promoted one assistant professor. In the same period, we probably hired ten others who left the department without tenure.

One of our assistant professors—I will call him Joe—was a difficult promotion case. His PhD thesis had attracted international attention. However, at the end of his term as assistant professor, his thesis still remained his best work. Since we had hired him based on his PhD work, I did not think we should also grant him tenure for the same ideas. My decision to deny Joe tenure upset Ginsburg and Horowitz, who felt that he met the standards of tenure at USC and did not want to vote against a friend they had known for several years. The human conflict between friendship and professional standards often prevents faculty from reaching tenure decisions that are in the best interest of the university.

Towards the end of my years at USC, I learned how Harvard deals with this dilemma. Sometime in 1983, I received a letter from Henry Rosovsky, the renowned dean of Arts and Sciences at Harvard University, inviting me to serve on an ad hoc committee to advise president Derek Bok on a proposed tenure appointment in Computer Science. The department had already obtained letters from leading computer scientists ranking the candidate as one of the top specialists in the world. As an experienced dean, Rosovsky (1990) knew "all too well that departments frequently present a somewhat misleading impression of enthusiasm and unanimity...Private and confidential letters [from each member of the departmental committee] provide a superb check on the extravagances of official case statements."

The ad hoc committee, chaired by president Bok, included dean Rosovsky and two faculty members from other departments at Harvard. I was invited as one of three computer scientists from other universities. At the ad hoc

meeting, members of Harvard's computer science department faced the committee, one at a time, for about half an hour each. Prior to the meeting, the identities of the panel members had been kept confidential.

According to dean Rosovsky, "It is not at all unusual for positive witnesses to turn slighly negative under the stress of interrogation." When I was there, Derek Bok asked each witness the same question: "Is the candidate, in your opinion, a major intellect?" The witnesses were obviously well prepared to answer questions about the significance of the candidate's work. However, when asked this embarrassing question by Harvard's president, every one of them admitted that he was not a major intellect. During lunch, Derek Bok asked each of us to offer his or her personal advice. The final decision was his. In this case, he denied the department's recommendation of tenure.

Dean Rosovsky makes it clear why Harvard succeeds where lesser universities fail: "Few, if any university presidents play as great a role in the appointment process. Harvard's Derek Bok considers this role to be the most important and interesting part of his work. It is the most direct way for him to control the quality of the faculty...We seek the best scholar-teachers, and if they happen to have abominable personalities, why then we claim joyfully to suffer in the name of learning."

Five years before my visit to Harvard, I had already reached the same conclusion as the bestselling author Nevil Shute, who started a small airplane manufacturing company, named Airspeed, in the 1930s. In his fascinating "Autobiography of an Engineer" (1954), he writes:

> I would divide the senior executives of the engineering world into two categories, the starters and the runners, the men with a creative instinct who can start a new venture and the men who can run it to make it show a profit. They are very seldom combined in the same person...I was a starter and useless as a runner.

When it became evident that we were on the right track towards becoming first-rate, I discovered that I did not much like the administrative aspects of my job. I wanted to do research instead of meeting the mother of a promising high-school senior from Pasadena. And my strong desire to chart the future course of the department undoubtedly made me insensitive to the personal agendas of some of my colleagues.

In 1978 I stepped down as department head. Two years later, a survey conducted by the National Research Council in 1980 ranked computer science at USC as one of the top ten departments in the country in terms of reputation and faculty publications. Our reputation was based on the opinions of 5,000 faculty members at 228 universities (Computerworld 1983).

Seymour Ginsburg said that the main effect of my tenure as the first chair had been to raise the standards of the department and make bold decisions. As long as I was in charge, he was a strong supporter of my relentless drive for success. He was then in his early fifties and had been at USC for ten years. He had the highest possible standards in his research. In private, he would make surprisingly sharp observations about his colleagues. I remember him saying: "Professor X can't tell the difference between the great and near-great." Ginsburg was not interested in succeeding me as department chair. He was content to influence the department as a gray eminence behind the scenes. When I stepped down, he supported a succession of acting chairs of less and less academic stature and vision.

Over the years, USC dropped to a still respectable position in the second rank of computer science departments.

⋆ ⋆ ⋆

Before joining USC, I had developed the programming language Concurrent Pascal at Caltech. I knew the time was now ripe for a book on the principles of abstract parallel programming. My second book, *The Architecture of Concurrent Programs*, included the complete text of the model operating systems I had written in Concurrent Pascal (Brinch Hansen 1977). Thanks to my editor, Karl Karlstrom, it was also translated and published in Japanese (1980) and German (1981).

The mathematician Harlan Mills, who was well-known for his efforts to introduce structured programming at IBM, and his associate Roy Maddux studied my book carefully. In a review they wrote (Maddux 1979):

> This is, as far as we know, the first book published on concurrent programming. Previously, this topic has been included in books on operating systems, a closely related but different subject. Books on operating systems usually consist of a survey of such topics as processor allocation, memory management, interrupts, I/O, file systems, process synchronization, batch and multiprogramming systems, scheduling, deadlock, and protection.

> Even after reading several books of this nature, the reader is left feeling that he has been exposed to a number of complex problems yet has learned very little about designing and implementing even a modest operating system. If you have shared these feelings with us, you will welcome Brinch Hansen's most recent book.

I agree with this criticism of operating system texts. Over the years they have often been reduced to the level of "Popular Mechanics." By making such superficial courses "required," universities have a convenient excuse to lower their standards and attract marginal students. I finally stopped teaching the subject ten years ago.

Maddux and Mills were particularly pleased with the Solo system:

> Here, an entire operating system is visible, with every line of program open to scrutiny. There is no hidden mystery, and after studying such extensive examples, the reader feels that he could tackle similar jobs and that he could change the system at will. Never before have we seen an operating system shown in such detail and in a manner so amenable to modification.

In conclusion, they wrote:

> The book cannot be called a textbook; it is, rather, a thorough technical monograph that requires sustained concentration. The importance of Concurrent Pascal as the first language for concurrent programming makes the effort worthwhile.

While the book was still in production, I submitted the manuscript to my alma mater, the Technical University of Denmark, as a thesis for the Doctor Technices degree. This Danish degree (which requires no course work) is awarded about once a year to a researcher who has moved engineering and applied science a significant step forward.

After twenty years in civil engineering, my father, Jørgen Brinch Hansen, earned the Dr. techn. degree in 1953. He was then chief engineer at the internationally known engineering firm, Christiani & Nielsen. His doctoral thesis, Earth Pressure Calculation, developed the first generally applicable method for the solution of most earth pressure problems in practice.

Now, a quarter of a century later, it was my turn. In September 1976, the Technical University appointed a committee to read my thesis. The committee consisted of three distinguished Scandinavian professors of computer

science: Ole-Johan Dahl (University of Oslo), Christian Gram (Technical University of Denmark), and Peter Naur (University of Copenhagen). Seven months later, they submitted a five-page evaluation to the university recommending that it be accepted for the defense of the technical doctoral degree.

In January 1977, I satisfied one of the official requirements for the degree by asking Prentice Hall to ship 200 copies of the thesis to the Technical University, for general distribution to various places (I have no idea where they went).

The leading Danish newspapers, Berlingske Tidende and Politiken, interviewed me about the practical significance of my work and announced the time and place of the official defense of my thesis. This event took place on January 23, 1978, in the largest auditorium at the Technical University of Denmark. It began at 2 p.m. and lasted about four hours. The Swedish computer magazine Data wrote (February 1, 1978):

> All [three opponents, Dahl, Gram, and Naur] gave the doctoral candidate an extremely positive reception. In the auditorium, where no less than 400 people were present, the spirit of Niels Ivar Bech seemed to be present, while his associates engaged in discussion at a higher level. (English translation by me.)

Each opponent gave a summary and evaluation of my thesis. In his remarks, Peter Naur said:

> The text is characterized by great clarity and convincing arguments. In my opinion, it culminates in the description of the Solo operating system, the job stream problem, and the real-time scheduler. In these chapters, the description proceeds fluently with an apparent ease that is quite overwhelming. Here, above all, Per Brinch Hansen demonstrates his mastery. Everything looks so easy, as it always does in the hands of the master. For those who will attempt to do the same, it will probably turn out to be fraught with problems and traps, but as a source of inspiration, these sections will be of enormous value. For the work as a whole, the significance of these sections is that they demonstrate the value of the new programming language concepts they are based on. The discussion of these new concepts [monitor and process types] must also be praised for its convincing clarity.

So far, so good. However, as an opponent, Naur was also expected to point out weaknesses of my thesis. It was not by chance that he focussed on the definition of the programming language Concurrent Pascal. He said:

> I see that on page 245 you define a process type as a form of data type, while on page 236 you define a data type as a set of values. Can you tell me in what sense a process is a set of values?

I was well aware that my first language report was the weakest part of my work. Since I have always made it a rule never to defend the indefensible, I turned to Peter and said:

> The English computer scientist Tony Hoare once said that the Algol 60 report, which you wrote, was a considerable improvement over its successors. Well, my report is one of the successors.

Everybody laughed and Peter smiled saying: "You got it!"

It would be another seven years before Tony Hoare (1985) introduced a mathematical model which identifies a process with all the possible sequences of actions (known as "traces") in which it can participate. In 1974, Roy Campbell and Nico Habermann had introduced an early notation, called "path expressions," for this idea.

If I had to single out an event that marked the peak of my research career, it would be that day in 1978 when I became the first computer scientist to receive the Dr. techn. degree. I was then 39 years old and had worked at the cutting edge of operating systems and concurrent programming for ten years. Never again would I have a similar streak of luck.

In most instances, scientific creativity peaks around age forty. Nobody knows why it should be so. In his study of *Genius, Creativity, and Leadership*, Simonton (1984) suggests that "True creativity demands the right combination of enthusiasm and experience. . .Enthusiasm tends to peak rather early in life and then steadily decline, whereas experience gradually increases with age. . .Thus the age-40 floruit is a consequence of this uniquely balanced juxtaposition of youth's rapture and maturity's sagacity."

Speaking of honors: When I joined USC, Kaprielian offered me an endowed chair. I didn't think that my first act as department head should be to accept an honor for myself. So I suggested that he offer it to Seymour Ginsburg. In the spring of 1978, Ginsburg became the first Fletcher Jones Professor of Computer Science. Kaprielian, who was not used to being turned down, never offered me another endowed chair.

Three years later, the new president of USC, James Zumberge, removed Kaprielian from his position as executive vice president. Driving home from a New Year's party, Kaprielian suffered a fatal heart attack and crashed through the living room of a house in Beverly Hills. The following year, on September 15, 1982, I was named the first Henry Salvatori Professor of Computer Science at USC.

My last act as chair of computer science was to nominate Tony Hoare for an honorary doctorate at USC. My colleagues would have preferred to honor an American computer scientist, but, since I was the chair, they went along with my nomination. In 1979, Hoare became the first computer scientist to be awarded the degree of Doctor of Sciences *Honoris Causa* by an American University.

⋆ ⋆ ⋆

In 1978, L. J. Sevin, chairman of Mostek Corporation in Dallas, and one of his young engineers, Steve Goings, paid me a visit at USC. Mostek was then the world's largest manufacturer of semiconductor memories. Goings, who had read my book, "The Architecture of Concurrent Programs," had suggested to L.J. that they should meet with me to discuss what Mostek should be doing in computing.

Mostek predicted that VLSI technology soon would make it possible to put an IBM/360 mainframe computer on a single chip! So L.J. wanted to know if I thought it would be a good idea for Mostek to develop such a powerful chip. As far as I could see, the problem was not the chip, but the notoriously unreliable IBM software that would run on it. Once they had sold a large number of System/360 microprocessors, I feared their customers would expect them to correct errors in OS/360—a task that taxed even the expertise of IBM itself.

At the 1969 Nato Conference on Software Engineering, Martin Hopkins, IBM, admitted that, "We face a fantastic problem in big systems. For instance, in OS/360 we have about 1000 errors in each release and this number seems to be reasonably constant."

In a letter to me, thirty years later, Dijkstra wrote:

> I always felt that IBM's inability to make a decent operating system for its own hardware played a significant role in the recognition of the "software crisis" in 1968. In that sense, OS/360 has been significant.

Before our meeting, Steve Goings had already told L. J. Sevin that he did not think there was much future in designing chips that emulate obsolete computer architectures:

> I urged that we abandon the IBM emulator, and create a microprocessor that could work effectively in a multiprocessor architecture, and provide for more direct support of high level programming languages. Further, we needed the help of top level software engineers in the field, He asked me if I had anyone in mind. My reponse was, "I do not know him personally, but I have a high regard for the publications of Per Brinch Hansen." (Letter from Steve Goings, July 19, 2004.)

That is when they decided to see me. During our conversation, I said, "While you are here, I would like to tell you about an inexpensive multiprocessor I have proposed."

A multiprocessor consists of identical processors that run in parallel and communicate through common memory. The challenge is to make sure that the common memory does not become a serious bottleneck for the processors. When Bill Wulf and Gordon Bell (1972) developed their pioneering C.mmp multiprocessor at Carnegie-Mellon, they made the bold decision of making every memory location accessible to every processor. They did this by connecting sixteen PDP 11 minicomputers to sixteen independent memory modules via a crossbar switch. Since a switch that connects n processors to n memory modules has a hardware complexity of order n^2, this is a rather expensive solution.

What I outlined was a simpler multiprocessor with two to ten microprocessors. The processors would have their own local memories and would share a single common memory (Brinch Hansen 1978b). This architecture was intended for dedicated real-time applications programmed in a language with concurrent processes and monitors. Each processor and its local memory would be dedicated to the execution of a single process. The processes would communicate by means of monitors stored in the common memory.

Since a monitor only performs one operation at a time, it is per definition a bottleneck in a concurrent program. To make a concurrent program as fast as possible, a wise programmer will make sure that each process spends most of its time accessing its own code and local variables and uses as little time as possible inside monitors. If that assumption was correct, it would make sense to replace the crossbar switch with a single common memory module.

This would make the complexity of the multiprocessor proportional to the number of processors.

While I was explaining all of that, L. J. Sevin looked immensely bored. When I was finished, he turned to Goings and said: "I think we ought to build his machine!"

In the fall of 1978, Mostek started a research project managed by Steve Goings. As project consultant, I would be responsible for designing a concurrent programming language and a multiprocessor architecture tailored to the language. A team of Mostek engineers, headed by Nick Matelan, would be responsible for the hardware implementation. Several times a year, Goings, Matelan, and I would spend a weekend at the Pasadena Hilton Hotel discussing technical details.

My first task was to develop a concurrent programming language, named Edison, which included concurrent statements and conditional critical regions. It was as powerful as the combination of Pascal and Concurrent Pascal, but much simpler (Brinch Hansen 1981). At my suggestion, Mostek signed a consulting agreement with Peter Naur to review my definition of Edison. Naur made almost no comments about my choice and design of language features. His main concern was the clarity of the language report. I would write a complete draft of the report and Naur would then point out what the weaknesses were and suggest broadly how they might be removed in my next draft. Between January 1979 and September 1980, I wrote four versions of the Edison report from scratch. About the second version, Naur wrote (Brinch Hansen 1981):

> The report is a vast improvement over the previous version in clarity, consistency, and completeness. The remaining weaknesses, described below in detail, are to a large extent concerned merely with finer matters of conceptual clarity.

After this pleasant introduction, he went on to enumerate 79 conceptual problems. The writing of the Edison report was far more difficult and time consuming than the selection of language features and the design of the first compiler.

A key element in the development of the Edison multiprocessor was our decision from the beginning to define the function of every piece of hardware by an equivalent Edison program. Such a description was far more precise than a mixture of circuit diagrams, timing examples and prose. Not only could an Edison algorithm be subject to compile-time checking of consistency, but it could also be tested on an existing computer. More importantly,

Edison would serve as a formal specification language that was understood by both hardware and software engineers.

In the spring and summer of 1979, I wrote a report (revised after discussions with Nick Matelan) that defined the multiprocessor architecture by Edison algorithms. These algorithms closely mirrored the exchange of data and signals that took place in interactions between processors, memories, busses, peripheral devices, and arbiters. We also specified how to build a distributed system as a cluster of multiprocessors.

The specification of hardware by means of algorithms was not yet widely used in the computer industry and certainly not for something as complex as a multiprocessor with a hierarchy of bus lines. Our Edison algorithms enabled the hardware engineers to discover several logical errors in my original proposals of buslines *before* the circuits were implemented. In at least one case, they also made a hardware designer realize that an intermittent error in a circuit design was caused by his deviation from my Edison specification of what the circuit was supposed to do.

In a letter to me, Matelan wrote: "We got a 4-node Edison multiprocessor working, Monday morning, May 12 [1980]." During the summer, I wrote the first Edison compiler in Edison and tested it on a PDP 11/55 computer at USC. When I demonstrated this compiler for Mostek in November, L. J. Sevin agreed buy it for $100,000. However, before I had a chance to see the multiprocessor or sell my compiler, United Technologies bought Mostek and cancelled the multiprocessor project. Many years later, Steve Goings told me that the technical documents for the Edison project disappeared on that unfortunate occasion.

A few years ago, I discovered that Nick Matelan had started his own company, Flexible Computer Corporation, which developed a multiprocessor called the Flex/32. Since Matelan (1985) neither acknowledged Mostek nor me, it is difficult to say how much this machine owed to the Edison multiprocessor. At one point, Purdue had a 7-processor Flex configuration, while a 20-processor machine was installed at the NASA Langley Research Center in Virginia. Matelan's company no longer exists.

At the beginning of the Edison multiprocessor project, L. J. Sevin told me that Mostek was funding a dozen research projects that gambled on future computer technology. He expected most of them to fail. Even though nothing came of our project, I am glad I met L.J. who always thought big. During a dinner at a Dallas restaurant, he asked if I would be interested in starting a software research center for Mostek. "Can I do it anywhere in the

world?" I asked. "We hope you will do it in Texas," he said, "but, if you prefer, you can also build it in Denmark." I said I would need to think about it before I gave up my tenured university position. L.J. responded with the immortal words, "Let no man complain to me about the size of his balls!"

In the end, L.J. did all right. When United Technologies bought Mostek, he joined a venture capital partnership and invested his money in a little-known manufacturer of PCs named Compaq. Steve Goings did some consulting work for L.J. The first day he walked into this new business, L.J. introduced him around and said, "Steve is a pioneer. You can always tell who the pioneers are. They are the ones with all the arrows in their backs."

⋆ ⋆ ⋆

Our one-story house on 1351 Pleasant Ridge, Altadena, was located at the top of a steep street in the foothills of the Sierra Madre mountains, at the entrance to a steep, narrow canyon covered with oak trees and brush. The house was built in Spanish style with hardwood floors and beam ceilings. The living room was dominated by a large fireplace embedded in a brick wall. It had full-length floor-to-ceiling windows facing south. On the opposite wall, sliding doors led to a covered patio facing the canyon. After we had furnished it with pine furniture and rya rugs, our home looked like a cozy hunting cabin. On a clear night, we had a panoramic view of the endless carpet of lights in Pasadena and beyond. On a smoggy day, it looked as if we lived above the clouds.

Our patio was like a zoo with frogs, lizards, rabbits, hummingbirds, and a small pond with Japanese koi fish. Sometimes a deer would come all the way down from the canyon, and at night packs of coyotes would howl nearby in the mountains. One evening, Milena went to bed without closing the screen door to the patio. I found her sleeping while a tarantula the size of a child's hand was crawling on the floor. Although they look dangerous, tarantulas are fairly harmless and won't bite if you leave them alone. Their painful bite is apparently no worse than a bee sting.

We were much more concerned about the rattlesnakes that occasionally found their way into our garden looking for water and mice. The previous owner was a doctor who kept snake serum in his refrigerator. He said, "Don't worry about snakes—the kids always spot them first!" And he was right, they did. Before getting into my car in the garage, I always knelt down to see if a rattler was hiding under the car. I killed several small ones by cutting their heads off with a shovel. However, when I found a big rattler,

as thick as my arm, on the driveway, I called the fire department. The fire fighters drove up to our house in a huge fire engine with flashing lights and sirens on, killed the snake and cut the rattle off as a present to my son.

Although we learned to live with the snakes, I could never get used to the black widows—the poisonous spiders that were found throughout the house, in our potted plants, behind book shelves, and underneath our beds. Fortunately, none of us was ever bitten by one.

The most dramatic event we experienced in Southern California was a natural catastrophy that nearly ruined us and almost killed me. (However, as the Danes say, "nearly" and "almost" never threw anybody off his horse.)

Sometime in September 1979, children set fire to a trash can several miles from our home. This started the largest brush fire in the area in forty years. The mountains were covered with dry vegetation that burns like torches. When it starts burning, there isn't much anyone can do other than waiting until it burns itself out.

Our house was completely surrounded by brush that grew right down to the edge of the patio. On the first day of the fire, we saw smoke rising above the mountains east of our house. During the night, the flames reached the top of the mountains and started creeping down towards the homes of our neighbors. Soon the whole mountain side was burning with a faint crackle lighting the terrain with a deep red color. It was both beautiful and cozy unless you lived right next to it.

The next day, the fire slowly burned away from us and moved up into a large canyon that was separated from our small canyon by a mountain ridge. In the middle of the night I woke up and saw, for the first time, the sun rising in the north with a shiny glow high up in our canyon.

On the morning of the third day, the fire was burning through our canyon towards our house. For several days, the air above Los Angeles had been stagnant making the air pollution worse and worse. However, since the fire burned downhill in the still air, it moved relatively slowly. Late in the afternoon, the fire reached our property. On this quiet day, it was no problem for the firefighters to prevent it from spreading to our house. Two fire fighters spent the night in lawn chairs in our driveway, guarding a tree that, if it were to catch fire, would explode and burn like a torch.

When it was finally contained, the brush fire had destroyed 30,000 acres and had occupied 3,000 fire fighters for a week. Where our house had been surrounded by green hillsides we saw only scorched ridges covered with soil and gravel.

In California, it often rains for days in December and January, as storms move in from the Pacific. Where we lived, most of the runoff water from the hills had been stopped by brush and tree roots before it reached the bottom of our canyon. However, when it rained heavily, some water would flow through a small storm drain under our house. This storm drain was owned by the county who was responsible for maintaining it.

However, since the mountains were now stripped of vegetation and covered with debris, it was a virtual certainty that our house would be destroyed by mudslides during heavy rain. The only hope of saving the house was to build a deflector wall that would direct the debris flow across the patio continuing past the garage and through the driveway onto the street (Fig. 7.1). But we were unable to find an engineering firm that would help us calculate the dimensions of such a wall. The consulting firm of Alderman, Swift & Lewis described their main concern:

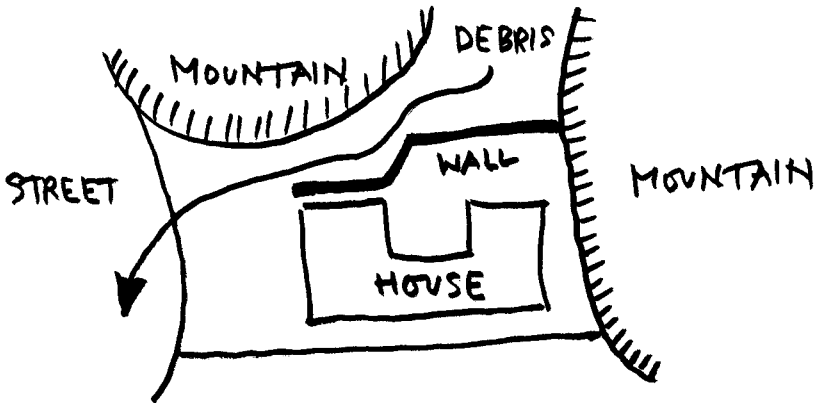

Figure 7.1 Debris flow in Altadena.

> Because of the location of your house, patio and garage, it would be necessary for a diversion wall to alter the direction of flow nearly 90 degrees in a very short distance. Unlike clear flow, debris flow cannot be diverted this quick. As a result, the diversion wall may be topped.

Two professors of civil engineering at USC volunteered to help us for free

without any guarantees. In the meantime I started calling local building contractors and discovered again that none of them were willing to take the risk of being sued. I finally found a Danish bricklayer, Knud Balling, who put me in contact with an American builder, Ed Sylvis. With no contract other than a handshake I agreed to pay this man and his Mexican crew on an hourly basis, without knowing in advance what it would end up costing.

Since there was no time left to worry about minor details, such as a building permit, we decided to call our wall a "timber and pipe fence." When it was finished in late November, it was 150 feet long and 6 feet tall. It was built of 2×6-inch lumber bolted to fifty 12 foot steel posts with a diameter of 6 inches, embedded in six feet of reinforced concrete. After I gave it several coats of dark red paint, it didn't look all that bad. However, we were still listed by the police as one of a dozen families who would need to be evacuated to save our lives during a big storm. In the last forty years, such storms had typically occurred twice a year.

Nothing happened in December. But in January 1980 it started raining for days. Early one morning, Milena was driving home after taking the children to school, when the house suddenly started shaking, as if a large helicopter was hovering right above it. From the bedroom window, I saw that our red wall suddenly had turned grey on the side facing the house. I ran outside in the rain. On the other side of the wall, an avalanche of mud and boulders from the canyon had piled up close to the top of the wall and was flowing through our driveway and down the steep Rubio Vista Drive into the gardens of a dozen other homes. Milena called from the local police station and told me that she was unable to drive up the street.

Although the Flood Control District had no funds to build our wall, they were still technically responsible for keeping their tiny (useless) storm drain open. They used this as an official excuse to send one of their bulldozers and an army of dump trucks to prepare us for the next storm. For twenty four hours, the bulldozer and the trucks worked continuously to remove debris behind the wall.

The storm had dumped two inches of rain. While this went on, an even bigger storm (which occurs about once every three years) was moving towards Southern California from the Pacific. Students and faculty from USC came to our house and piled hundreds of sandbags around the house inside the wall, while three carpenters covered all doors and windows with plywood. At lunch time, I went to the drive-in entrance of Burger King and casually ordered "20 whoppers, please."

The next morning, January 11, the headline on the frontpage of the Pasadena Star-News read "Altadenans may evacuate." The article included two photos showing how "Altadenan Per Brinch Hansen readies for rain" while "Storm clouds hang over L.A. as seen from Pleasantridge Drive in Altadena." According to Bill Hardy of the Flood Control District, "If any house in Altadena is in danger, it is 1351 Pleasant Ridge, which is directly in the mouth of a gorge."

We now got four inches of rain, and again the wall held up. This happened several times over the next four months. When a big storm was forecast, we spent the night in a motel, while flood control workers protected our property against looters. When spring finally came, we had survived the worst rainy season in ten years. In May, we were able to remove the plywood and half-rotten sandbags and let the daylight into our rooms. After some minor repair, the house looked as good as new.

Had we not built the wall, I would almost certainly have been killed in the ruins of our house during the first mudslide and my family would have been ruined. When the wall was finished, Milena and I cooked steaks on the patio for Ed Sylvis and his crew. Over a beer, I said to Ed: "You knew I was completely at your mercy—how come you didn't take advantage of me?" He answered: "I can always make more money, but I can only lose my reputation once!"

⋆ ⋆ ⋆

While these natural catastrophies threatened our home, the pioneering era of concurrent programming was coming to an end. It is time to look at what we had achieved.

In the first survey paper on concurrent programming I had cited 11 papers only, written by four researchers. None of them described a concurrent programming language (Brinch Hansen 1973b). The development of monitors and Concurrent Pascal started a wave of research in concurrent programming languages. Fifteen years later, there were close to 20 monitor languages and 100 languages for distributed computing (Brinch Hansen 1993, Bal 1989).

Two of my former Ph.D. students recalled their experience of working with Concurrent Pascal at USC (Brinch Hansen 1993):

> *Jon Fellows*: The beauty of the structures you created using Concurrent Pascal created an aura of magical simplicity. While

> working with my own programs and those of other graduate students, I soon learned that ordinary, even ugly, programs could also be written in Concurrent Pascal... My current feeling is that the level of intellectual effort required to create a beautiful program structure cannot be reduced by programming language features, but that these features can more easily reveal a program's beauty to others who need to understand it.
>
> *Charles Hayden*: I think the significance of the system was ... that one could provide a protected environment for concurrent programming—a high-level language environment which could maintain the illusion that there was no "machine" level. It was remarkable that through compile time restrictions and virtual machine error checking ... you could understand the program behavior by looking at the Pascal, not at the machine's registers and memory. It was remarkable that the machine could retain its integrity while programs were being developed, without hardware memory protection.

In the fall of 1981, when Microsoft had just implemented DOS in assembly language for the first IBM Personal Computer, my students and I had already used high-level languages for seven years to write portable single-user operating systems for minis and micros, and had published the complete program text of some of these systems. Charles Hayden wrote no less than three operating systems on his own: a single-user system, a multiuser system, and another one with a Unix-style I/O system.

During the summer of 1981, I tested a single-user operating system for a PDP 11/23 microcomputer, written in the programming language Edison. The Edison system was able to compile itself and its compiler in 56K bytes of memory using two 8-inch floppy diskettes of 250K bytes each as the only form of backing store.

In the fall of 1982, I moved the Edison system to the IBM PC by rewriting a kernel of 4K bytes in assembly language. The Edison-PC system compiled itself in a 64K byte memory using dual $5\frac{1}{4}$-inch floppy diskettes.

In 1983, I published a book about the Edison system, entitled *Programming a Personal Computer*. According to Peter Naur (1984):

> In this book the author carries through an entirely fresh attack on the problem of programming language and operating system

> design, the incentive being the availability of microcomputers. Within the compass of the 388 pages of the book, the author manages to present in every detail: Edison, a new programming language suitable for concurrent programming; Edison system, an operating system; Edison code, an intermediate language designed to be suitable as intermediary between Edison and the machine languages of microcomputers; Alva, an assembly language for PDP 11 computers specially designed for supporting Edison; the complete programs for implementing each of these languages and systems; extensive discussions of the argument that lie behind the designs adopted throughout. While most of the detailed argumentation of the presentation is found similarly in the author's earlier work, the new development serves as a convincing demonstration of the power of the principles and methods employed in solving a problem having basically new constraints, those of a microcomputer.

It was now obvious to any casual observer that a programming revolution had taken place in programming languages and operating systems.

Looking back, what am I most proud of? The answer is simple: We did something that had not been done before! We demonstrated that it is possible to write nontrivial concurrent programs exclusively in a secure programming language.

In retrospect, the monitor concept was the first example of *object-oriented concurrent programming* (although I never used that term). However, the particular paradigm we chose (monitors) was a detail only. The important thing was to discover if it was possible to add a new dimension to programming languages: *modular concurrency*.

Every revolution in programming language technology introduces abstract programming concepts for a new application domain. Fortran and Algol 60 were the first abstract languages for numerical computation. Pascal was used to implement its own compiler. Simula 67 introduced the class concept for simulation.

Before Concurrent Pascal it was not known whether operating systems could be written in secure programming languages without machine-dependent features. The discovery that this was indeed possible for small operating systems and real-time systems was far more important (I think) than the introduction of monitors.

Monitors made process communication abstract and secure. That was,

of course, a breakthrough in the art of concurrent programming. However, the monitor concept was a detail in the sense that it was only one possible solution to the problem of making communication secure. Today we have three major communication paradigms: monitors, remote procedures, and message passing.

The development of abstract language notation for concurrent programming started in 1971. Fifteen years later Judy Bishop (1986) concluded:

> It is evident that the realm of concurrency is now firmly within the ambit of reliable languages and that future designs will provide for concurrent processing as a matter of course.

So passed an exciting era.