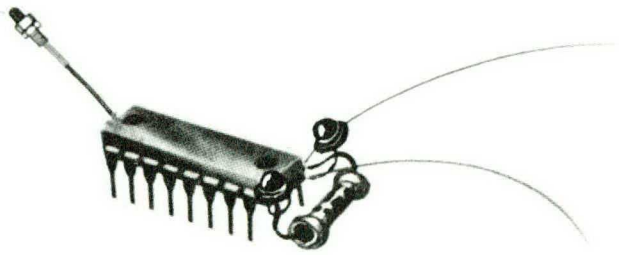


Debugging the computer "Eagle"

THE ULTIMATE TOY

BY TRACY KIDDER

"LA MACHINE," SAID ED RASALA, ONE DAY IN THE spring of 1979. "*The* machine. It's what everyone calls it. That's the whole thing, to build *the* machine."

Rasala's machine was the new 32-bit minicomputer, nicknamed Eagle, that Data General's Eclipse Group was building. By January of 1979, after about six months of intense labor, the team had designed Eagle and two prototypes had been assembled. But computers are complicated, and theirs was more complicated than many. As expected, Eagle's design was far from perfect. The prototypes did not work. The Eclipse Group had to find and repair the flaws in their creation; they had, as they said, to "debug" it. A debugging could be harrowing, in part because it had to be performed with great thoroughness. The hardware, the actual circuitry, of modern computers is remarkably reliable, and needs to be. Eagle would do a cycle of work every 220 billionths of a second. If they left in the machine bugs that caused it to fail only once in every billion cycles, Eagle would be a very unreliable contraption indeed.

The debugging of Eagle did not go well at first, and the Eclipse Group's leader, Tom West, fretted, his stomach queasy, for a couple of months. Gradually, the team surmounted the first barriers to success, and in March West declared, "Most of the fear is gone now." But the debugging was not nearly finished. In effect, West merely passed on the responsibility for worrying to his lieutenant, Ed Rasala.

Computer engineers, like poets, usually blossom early, and Ed Rasala was at thirty-five an old and experienced tradesman. He was the captain of "the Hardy Boys," a dozen or so engineers who worked on Eagle's hardware. When the debugging began, Rasala's Hardy Boys virtually went to live with the prototype Eagles, in a small and windowless laboratory, behind locked doors, at Data General's headquarters in Westborough, Massachusetts. Soon most of the crew were spending the majority of their waking hours in the lab, for six days of every week, and even when they went elsewhere, in their minds they continued to grope around inside the new, still defective machine. For the people who were building it, the incipient computer defined a small world. Rasala was the leader of the debugging; it was, in effect, his job to see that this little world was rid of uncertainty.

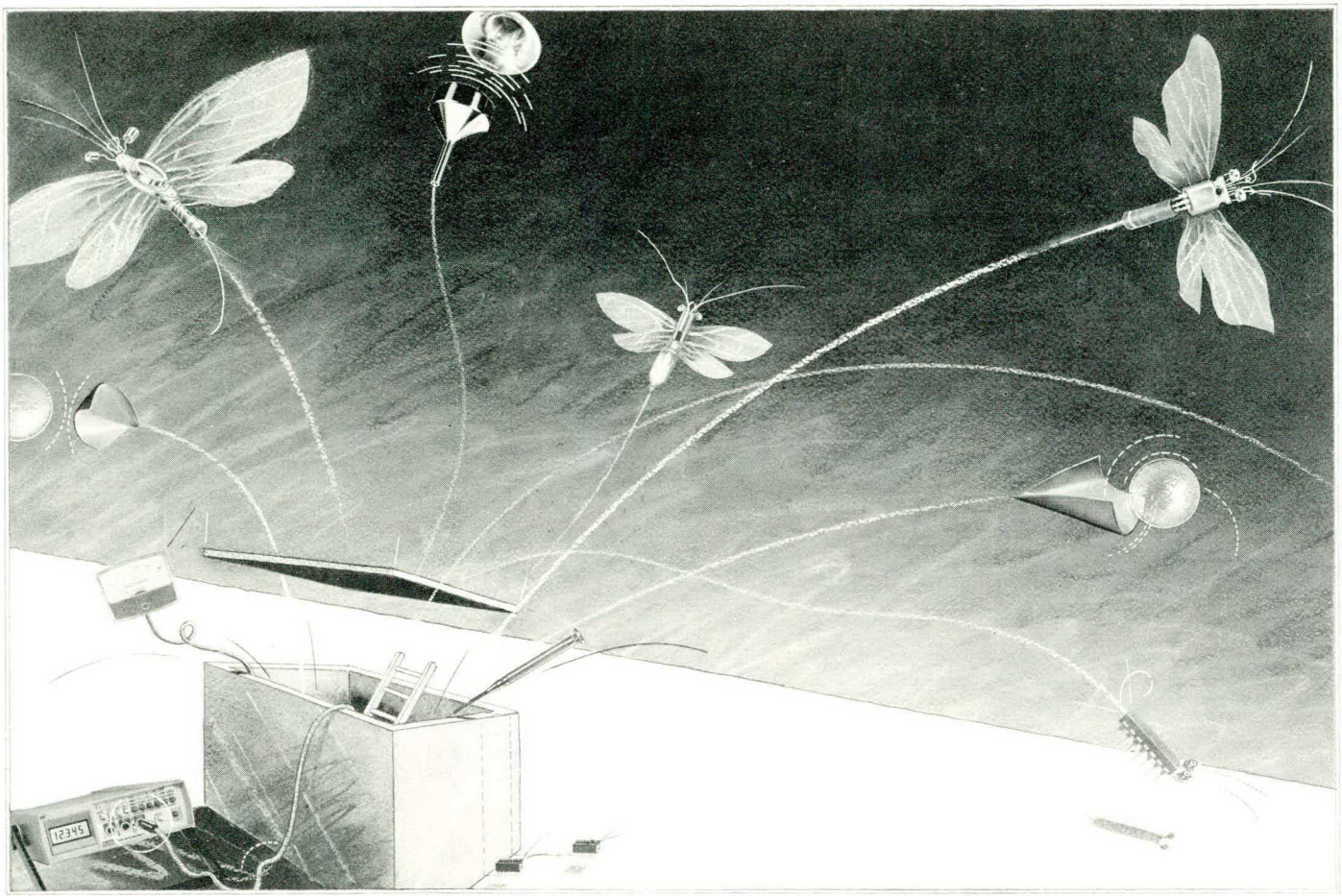
From the top of his head, which was bald, to his feet, which were usually shod in construction boots, Rasala looked big. He wore a thin beard. He delivered firm handshakes, and he spoke rapidly. "It's almost a speech impediment," said one of his oldest professional friends. "Sometimes I feel like giving him a shot of Valium and then telling him, 'Okay, Ed, say it again.' "

Of Ed Rasala Tom West once said, "His biggest strength is, the guy does not give up." Rasala's parents were immigrants from Poland; he grew up in modest circumstances in Brooklyn. It was his misfortune to have a brilliant older brother, against whom all of his teachers measured him. "From grade school on, it was always my rap, I never lived up to my potential," Rasala told me.

Rasala had been reluctant to work on Eagle when the project began; he had just completed another computer and he wanted a rest. Some months later, I asked him why he had changed his mind. He ticked off the reasons on his fingers. "I was looking for opportunity, responsibility, visibility."

Tracy Kidder is a frequent contributor to this magazine. His new book, The Soul of a New Machine, *will be published this month by Atlantic-Little, Brown.*

ILLUSTRATIONS BY ANDRZEJ DUDZINSKI



What did those words mean to him?

Rasala shrugged his shoulders. "I wanted to see what I was worth," he said.

Rasala got along well with most of the Hardy Boys. Often he jokingly insulted them, and they replied in kind. Rasala liked a slightly contentious atmosphere. "Smart, opinionated, and *non-sensitive*, that's a Hardy Boy," he declared. Above all, he wanted around him engineers who took an interest in the entire computer, not just in the parts they had designed.

One evening at the end of January, on the lower level of Data General's headquarters building, Rasala walked up to a doorway labeled "RESTRICTED AREA." He unlocked it and led me down a corridor, around a bend, through another door, and into a chamber about the size of a living room. The floor was linoleum-tiled; the ceiling was a network of heating pipes and steel girders, from which many thick black electrical cables hung; the furnishings were gray metal. The chief attractions, standing side by side at shoulder height along a wall of cement blocks, and surrounded by a variety of adjunctive devices, were the two prototype Eagles. Their blue metal frames housed a multitude of wires. "Here are two state-of-the-art computers, sitting there," Rasala said. "They're not too impressive at this point in time." In each machine was a shelf filled with "wire-wrapped" prototype circuit boards. On these boards, which were rectangular and about the size of an atlas, were implanted the essential hardware of the computer. They rendered an accurate visual impression of what the debuggers confronted. One side of each board contained orderly looking rows of small boxes. A nest of tiny wires covered the other side.

The Hardy Boys were debugging Eagle by exercising the prototypes and by fixing them when they didn't work. "Exercising" meant running programs, which are the linked lists of instructions and data that direct a computer's actions. These exercise programs were called "diagnostics." They were designed to make the prototypes fail on account of both glaring and subtle bugs. When a failure occurred during the running of a diagnostic, the Hardy Boys searched for its cause and invented a repair. Often, their repairs required alterations in the wiring of one or more boards. Rasala had made it a rule that they perform this labor promptly, and he had come to the lab this evening to enforce his commandment and to participate in the process itself.

Everyone, including Rasala, disliked this part of the job. Altering the wires was dreary labor that strained the eyes. Rasala sat down to work on a board. In a little while he retrieved with a pair of tweezers a tiny strand of wire. Holding the tweezers aloft, he explained that it was very easy to drop a piece of wire into the nest of wires on

a board, and an extraneous piece of wire could cause the machine to commit failures of the "flaky" variety; debuggers could spend hours, even days, searching for the cause of such a problem.

"We do this ourselves rather than have a wire person do it," said Rasala. "We may be a little more cautious." He turned back to the board he was working on, and added, with characteristic humility toward the gods of chance, "Then again, we may not be."

Rasala maintained that he was not the most inventive of the team's engineers. "But I'm a good designer, I think, and I'm a better debugger. I may not be the smartest guy in the world . . . but I'm dumb enough to stick with it to the end."

In the first months of debugging, Rasala's main technical role was to play the brake on the Hardy Boys' enthusiasm. The sheer physical complexity of the computer required that they plod sometimes; otherwise, they would never gain control over this machine. Were signals just barely making their appointed rounds in the 220 nanoseconds that lay between two ticks of the computer's clock? If so, Rasala would make the engineers return to problems they thought they had solved. Was there a lot of noise inside the prototypes? "Noise," Rasala explained, is what makes the TV go haywire when the blender is turned on; it consists of stray voltages that are propagated through the machine. Too much noise fouls performance.

Once in a while, a Hardy Boy would find a problem, fix it temporarily, and move on. The engineer would plan to make a proper repair later, but, absorbed in new bugs, he would forget to do so, and weeks afterward his oversight would cause some mysterious failure. Such omissions were inevitable. By April, constant handling of the prototype was beginning to make unreliable some of the thousands of connections among components on the boards; loose wires and sockets began to cause some flaky failures. These were often hard to diagnose; as the debuggers liked to say, it's hard to fix something when it's working. Now and then an improperly manufactured component impeded their progress. "Just stuff you never account for in a schedule," Rasala said. "You assume it's not going to happen, and it always does."

In addition, the Hardy Boys were unearthing many flaws in the logic of their design. "We went with an imperfect design," said Rasala, early in the spring. "We knew we were pushing it." When debugging had started, in January, Rasala's boss, West, had promised the front office that Eagle would be finished in April. Rasala was skeptical, but, believing in the need for haste and being a loyal lieutenant, he had worked up a debugging schedule for that date. However, by March it was clear that they would not finish by April. So the group's leader had named a new deadline, and then another, and for each revised deadline Rasala had drawn up a new debugging schedule. In May, when his third revised schedule was hanging on a wall in his cubicle, Rasala remarked, "The way to stay on schedule is to make another one." Asked to evaluate the team's progress, he said, "It's coming, it's coming." He added, "There's still a good chance that we've totally blown it."

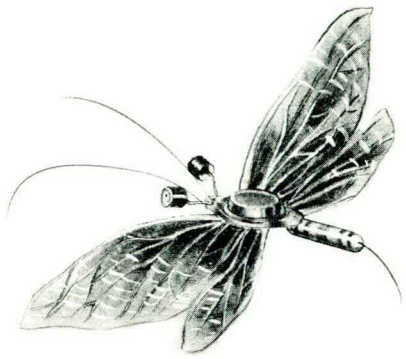
Near the beginning of the debugging he had said, "I believe there's always a focal point in any machine." And by May, the focal point of the problems had identified itself to him. It was the part of Eagle known as the instruction processor, or IP.

WHEN A COMPUTER RUNS A PROGRAM, THE MACHINE has to perform, again and again, two general sorts of tasks. First, it has to find and prepare an instruction; then it has to execute that instruction. Often a great deal more time is consumed in the preparation than in the execution. In order to make Eagle a fast computer, its designers arranged for it to perform the two operations simultaneously. The IP was, in the local idiom, an "accelerator." Instructions fall into oft-repeated patterns; the IP was designed to recognize the pattern and to figure out what, at any moment, the next instructions would be. In the 220 billionths of a second that lay between two ticks of Eagle's clock, the IP would send one instruction out for execution and at the same time make ready the next several instructions that the program is going to require.

The IP was a complex device, obviously; it was a sort of computer inside the computer, and like every computer, the IP had its own storage compartment, called the "I-cache." Another part of Eagle, called the "system cache," also contained a storage compartment, considerably larger than the I-cache. The system cache kept on hand commonly used instructions, so that if the IP made a wrong assumption and did not have the next instruction in its I-cache, it could get that instruction quickly from the system cache. Eagle had other storage compartments, including a "main memory," and all of these storage compartments had to be kept consistent with one another. Consistency, however, was not easily achieved. As Eagle maneuvered through a program, the IP and the system cache were constantly throwing out blocks of instructions and bringing new ones into their storage compartments. Although Eagle's designers had created mechanisms for preserving consistency among storages, they could not foresee all the possible sources of trouble. And the bugs that arose were particularly troublesome and devilishly hard to diagnose.

If some unforeseen electronic event should cause the I-cache to contain a block of instructions that "looked" the same but was in fact slightly different from a block of instructions in the system cache, Eagle would be bound to fail. Sooner or later, the IP would draw upon the outdated, inconsistent block of instructions in its cache and would order the computer to execute a wrong instruction.

Usually, the IP would place this wrong order a long time after its cache had become inconsistent with the rest of the memory system. Then the debuggers would not be able to identify the problem by examining the failure. It would be as if a time bomb had been placed inside the machine.



NOT LONG AFTER DAWN ONE MORNING IN THE MIDdle of May, Ken Holberger turned his brown Saab down the road toward the red-brick, fortress-like Data General building. At the beginning of the debugging, it had been dark when he drove in and dark when he left. The slow daily ascent of the morning sun across his windshield was one of the principal means by which he kept track of outside, planetary time. For the past few months, he had found himself incapable of reading newspapers. "We're deep in the debug. Yeah, underground," said Holberger. "Burn-out city."

Holberger was short and very handsome. He wore a neatly trimmed black beard. Rasala called him "a sharp cat." He was considered to be a first-rate circuit designer, and at this point the only member of the group whose understanding of the details of the hardware approached completeness. Though only in his mid-twenties, he had chief responsibility under Rasala for the creation of Eagle's hardware, and he shared Rasala's general attitude toward an engineering project. "It doesn't matter how hard you work on something," Holberger once said. "What counts is finishing and having it work."

Holberger had devised significant parts of Eagle, including some of the IP. He said he was not worried about that piece of hardware. The machine was like a crossword puzzle; he and the other designers had invented it, surely they could solve it. "I'm getting quite good at it," Holberger said. "I can track a problem back into the twilight zone quite well."

Holberger went right to the lab, to see how the machines had performed the night before.

Some weeks previously, the Hardy Boys had run a diagnostic program called "Eclipse 21," and the prototypes had failed occasionally. The debuggers had hypothesized that the problem was likely of the flaky sort, the result of a loose connection or noise, and they had moved on to other diagnostic programs. Now, however, the prototypes had successfully completed all of the basic diagnostics except for Eclipse 21. So Holberger had decreed that it was time to return and clean up that bug. Accordingly, the previous night he had asked that when they departed the lab, the second shift of debuggers leave both prototypes running Eclipse 21.

One characteristic of the diagnostic programs was extreme repetitiousness. They directed the computer to perform many sequences of operations and to do so over and over again. When the machine completed the entire program, it was said to have performed one "pass," and if left running all night, it would perform many passes. If the computer failed to execute an instruction properly, the program would tell the machine to send an error message to the system console, a large device with a typewriter-like keyboard. Then the program would tell the computer to continue its exercise. In the morning, the debuggers could find out right away, by placing an order through the console, how many passes a machine had completed during the previous night and how many times it had failed.

When Holberger came into the lab, a tall and fairly husky young man named Jim Veres was sitting at the console of the prototype called Gollum. (The Hardy Boys had christened one of their prototypes "Coke," and Holberger had named the other "Gollum," after the sinister, spidery creature in Tolkien's *Lord of the Rings*.) Veres had already examined the results of the nighttime exercise. Turning to Holberger, he said that Gollum had run 921 passes of Eclipse 21 and had committed only thirty failures. Coke, Veres added, had performed in identical fashion.

Holberger made a dour face. In 921 passes, thirty was a very small number of failures. Noise or a loose connection could produce such a result, but flaky problems usually created failures in no discernible pattern. If the problem was noise, then why had Coke and Gollum performed identically? Veres was thinking along the same lines. To himself, Veres said, "Either that noise is remarkably consistent, or we've got a real problem in the logic somewhere."

Holberger and Veres swapped a few theories about noise, but they were really just taking deep breaths. "Okay," said Holberger at length. "Time to fix it."

Between Holberger and Veres there existed a technical understanding of these machines that speech could not encompass. Most of the debuggers shared this specialist's ESP, but not all debugged well together. Holberger and Veres did. Like many of the Hardy Boys, Veres was just a year out of engineering school; Holberger called him "one of the stars." Veres had designed a large part of the IP, with Holberger's assistance, and he had quickly developed his own technique of debugging.

In talking to Veres, one found him listening intently, sometimes with a stern look on his face. He had his own small computing system. Sometimes, after a long day in

the lab, he would go home and play with it. "I like to tinker," he said once. "I like to build things." In his senior year at Georgia Tech, he became interested in digital clocks. "I built four or five. Then it was computer terminals. I built one. Then I decided I ought to have a computer to hook it up to. So I got a microprocessor and then I figured it was not worth much without an operating system, so I wrote a small operating system. I did a number of all-nighters building this computer junk." Veres was enjoying the Eagle project; his only complaint was that lately his managers had been scheduling work in the lab in such a way that he could not always get his hands on Gollum for as much time as he desired.

Holberger and Veres hooked the probes of two logic analyzers to various parts of Gollum. They "put on a trace." The analyzers were the Hardy Boys' windows into the computers. Each analyzer had a little screen; in essence, the analyzer served as a camera. Used artfully, it could take pictures of what was going on inside and between the circuit boards.

Holberger and Veres backed up the diagnostic program just a short distance from the point of failure. They ran the program. Gollum did not fail.

This was, as Veres put it, "another clue." In a machine with accelerators, history is important. Often it is some complex combination of previous operations that leads to a later failure. So they set Gollum running the diagnostic program from the beginning, and went to the cafeteria for coffee. About fifteen minutes later, when they were sitting in front of Gollum again, they saw flashes on the screens of their analyzers. Gollum had failed. They pulled up their chairs and began to study pictures.

What instruction did Gollum fail to execute? That was the first question.

"Okay. It's doing a JSR and Return."

The diagnostic program was telling Gollum to take a short detour off the main road of the program. Gollum was supposed to "Jump" while remembering how to get back ("Save Return") the stream of instructions that it was executing and get a new instruction. This new instruction should have directed Gollum to return to the mainstream of the program. This small series of operations was a spot quiz, as it were. Further study of the pictures on the analyzers' screens and of the various documents scattered across the top of the table told them that Gollum had in fact jumped to the right instruction and had returned to the right place. But when it returned, it executed the wrong instruction.

"Is it hitting the I-cache?" asked Holberger. Was the instruction that Gollum was supposed to execute, after making its jump, residing in the IP's storage? They examined more pictures and decided that Gollum was indeed hitting the I-cache. When they examined the contents of the I-cache, they discovered that it contained a wrong instruction where the right one should have been.

HOLBERGER AND VERES SAT BEFORE GOLLUM. THEY had completed their preliminary investigation of the bug in Eclipse 21, and now the main question lay before them: Why was there a wrong instruction at the right place? The morning and half the afternoon had slipped away. It was two o'clock, and Jim Guyer had just come into the lab. He put down his motorcycle helmet, pulled up a chair, and started asking questions.

Guyer, at twenty-six, was a relatively old hand among Hardy Boys. His brown beard made an oval frame around his face. As a rule, he was very cheerful, much given to laughter, and he had become one of Rasala's favorites, largely because he took an interest in every

DURING THE WAR

I was one of those who sees something cross the moon
and is never again the same person.
I was one of those who fought for an envelope, for a stamp.
I was one who felt the pain of the potted plant
and the loneliness of the tree.

It was all in my mind. It was an idea I had!
Was it such a bad idea?
Did it hurt anyone? Did it even delay for a minute
a father thousands of miles from me
fleeing a tank from my own beloved country?

Did it hasten the deaths by bombing?
Did it quicken the suffering of the homeless?

Certainly not. Absolutely, unequivocally, the answer is No.
My love for the tree did not interfere
with one decision of a committee, one staff meeting.
My joining forces with the mysterious
did not blot a single marching order, nor break a rifle.

The world went on without me. To its mind,
I was no part of it, but to my mind I was too much a part.
I put it all down on paper but in code.
Even at night, I watched the skies like a spy
for the intersection of a planet and the future.

—Marvin Bell

part of the machine. Although he had had nothing to do with designing it, Guyer had been studying the IP assiduously of late. Indeed, he had been blaming the IP, sometimes incorrectly, for practically everything that went wrong inside the prototypes.

In the past month or so, the IP had been found responsible for a number of bugs. This fact bothered Veres. It did not matter that the IP was a much more complex piece of hardware than many others in Eagle, nor that when he had helped to design it Veres had been a novice and pressed for time. Like most members of the team, Veres felt what Holberger called "the peer pressure": "If I screw up this, I'll be the only one, and I'm not gonna be the only one."

Though they were friends, Veres was a little annoyed with Guyer, and Guyer knew it. So Guyer did not feel inclined to blame this latest bug on the IP. It appeared that either the IP or the system cache was responsible for the failure. Another engineer in the team had designed the system cache, and he wasn't in the lab that afternoon. Laughing about their choice, Veres, Holberger, and Guyer decided to interrogate the system cache first.

They hooked up analyzers to the circuits of the system cache and took some pictures, but they received no immediate enlightenment. Weary after more than ten hours in the lab, Holberger and Veres departed, leaving Guyer alone with Gollum. He spent the night with it.

Some time after midnight, Guyer was sitting in front of a couple of logic analyzers, peering into their screens, when suddenly he touched his mouth, wheeled around in his chair, and started flipping through one of the large bound volumes on the lab table. He had discovered that the diagnostic program periodically changed the location of the target instruction, the one that Gollum failed to execute. Maybe the IP wasn't getting the word to change the instruction's location. The system cache was supposed to see to it that the IP did so. Maybe the system cache was indeed to blame. With mounting enthusiasm, Guyer recorded his theory in the log book and returned to the machines to gather evidence. By three that morning, however, he had found none, and his enthusiasm for his theory had waned. He felt, he said, "sort of neutral." It was beginning to look more complex, he thought on his way out of the lab.

Around dawn, Veres sat down in front of Gollum again. He examined Guyer's entries in the log book. These deepened Veres's suspicions that the problem involved one of the figurative time bombs that they had been encountering frequently in the past several weeks. One way to approach such a difficult bug was to follow clues back through the diagnostic program. First, Veres found out exactly where in the program the failure initially occurred. Next, he examined "addresses." One could imagine the machine's memory system as a collection of mailboxes, organized in neighborhoods. Like each mailbox, each neighborhood has a unique address; that address is a number, which is called a "tag." At the moment of failure, Veres discovered, the I-cache contained a collection of mailboxes identified by the tag 21. But in the system cache at that moment, the tag for what should have been the corresponding collection of mailboxes was 45. Which was the right number? The answer would reveal whether the IP or the system cache was at fault. Veres hooked up logic analyzers to Gollum once again.

Old hands in the group had used oscilloscopes for debugging previous machines. Veres once said, "An oscilloscope is what cavemen used for debugging fire." Analyzers were newer, more versatile tools; one of their important features was that they had memories. They could take and save pictures of electronic events that occurred in 256 different cycles of the computer's operation. Veres ran the diagnostic program all over again. Then he started looking back through the pictures that his analyzers had taken and saved. He had scarcely begun this tedious search when Holberger arrived, and when Holberger saw what Veres was up to, he retreated, to work with another Hardy Boy on Coke.

Nothing turned up. By the time Guyer came in that afternoon, Veres had not found a single new clue. Holberger held a short conference. "We need ideas. We're going to defer it," he said. So Guyer worked on other problems that night. But Veres went home with the two tags on his mind. One was right, the other wrong. Obviously, there had to be a way of finding out which was which.

Veres was up early the next morning. He didn't want any interference; he wanted time alone with this bug. He got into the shower and his working day commenced. "I get quite a lot of work done in the morning while taking a shower," Veres remarked later on. "Showers are kind of boring things, all things considered." That morning, he conceived a new approach. Evidently, the cause of the failure lay further back in the diagnostic program than a logic analyzer could look. So why not search from the other direction, forward through the program, instead of backward? He would run the diagnostic up to the start of the fourth pass, and then every time Gollum performed a JSR and Return, he'd have the computer halt so that he could get a picture with the analyzer and certain printed information on the system console. This technique would take some time; but he was willing to try it.

Holberger entered the lab a few hours after Veres did. The scene that greeted him made him smile wryly. Nearby Gollum, on the floor, lay a great heap of computer paper, and Veres was sitting beside the pile. "I found it," he said to Holberger.

At iteration 122 of the subtest in question, the I-cache contained the block of instructions numbered 21. Millions of ticks of the computer's clock and thousands of instructions later, at iteration 151 of the subtest, Veres had observed the system cache telling the IP to replace tag 21

with tag 45. He had exonerated the system cache. The IP must have disobeyed the order, because at iteration 158 Veres had found the I-cache still harboring tag 21. "Which, I'm very sorry to say, is wrong."

Holberger and Veres moved swiftly. They hooked up analyzers. Seen at such a moment, among their machines, flicking switches, speaking cryptically in a language that even another computer engineer, from a different project, would have found largely incomprehensible, they resembled airline pilots in a cockpit preparing for takeoff. In fairly short order, the crucial picture appeared on the small blue screen of one of the analyzers.

"There it is."

"Yup."

They saw the IP throwing out tag 45 and keeping the old, invalid tag 21. A few more pictures showed that the IP was, quite literally, getting its signals crossed. The IP received from the system cache the signal to throw away tag 21, but before the IP could obey, the signal from the system cache was altered, by another signal coming from another part of the machine. The solution lay in delaying the arrival of that second signal, so that the IP would always have time to clear out an old collection of mailboxes before it was asked to perform some other task.

The solution took the material form of a circuit that cost eight cents wholesale. This circuit produced a signal. Writing up the engineering change order, Holberger christened the signal "NOT YET." Other engineering teams used formal, technical names for signals. The Eclipse Group usually looked for something simple that fit, and if they couldn't come up with an appropriate title they'd use their own given names. That, Holberger noted, defined the Eclipse Group's style. It was a way—a small one, to be sure—of leaving something of oneself inside one's creation.

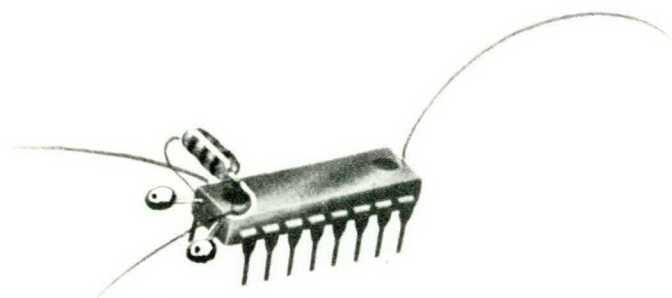
They were having fun now. They installed the new circuit. They ran the diagnostic. Holberger wrote in the logbook, "With this ECO installed, Eclipse 21 runs 10 passes." Just one more routine chore remained. They had to make sure that the new circuit didn't foul up some other operation. So they started running the other diagnostic programs that Gollum had already mastered, and everything was proceeding satisfactorily when all of a sudden the console started scratching out an error message.

"We didn't do it," said Holberger. "We didn't do it right."

It seemed to Holberger that they were on the brink of another lengthy search, and he had no appetite for it. They hooked up analyzers and studied some pictures, but in a desultory way. The problem looked complex. Then Veres remembered that they had forgotten to do something basic.

He took out the new circuit. They ran the failing program. Gollum committed the new failure anyway. So it was not the new circuit that had caused this problem. Greatly relieved, smiling now, Holberger pointed out that they had placed the IP circuit board out on the "extender." The IP was hooked up to Gollum, but it was sitting in a small frame of its own, outside the main frame of the machine. Use of the extender is standard debugging practice; it makes the board easy to get at. But computers aren't created with extenders in mind, and in some cases a perfectly good circuit board won't function correctly while out on an extender. They put the IP board back in its proper place, among the other boards, and the failure no longer occurred.

That afternoon, Gollum was made to perform all of the basic diagnostics, including Eclipse 21. The machine did not fail once. They had reached a milestone, but it was one that they had thought they had reached before. The trickiest diagnostic programs lay ahead. Veres said he had "a feeling of accomplishment." He added, "But then again, there's lots more feeling of accomplishment to go."



MANY OF THE ENGINEERS IN THE ECLIPSE GROUP were tinkerers when they were growing up. They remembered taking apart and then reconstructing—often just for the fun of it and sometimes to their parents' chagrin—all manner of things in their childhood homes: lamps, telephones, clocks, radios, TVs, lawnmowers. Some still found a complicated-looking, storebought object an irresistible temptation. Ken Holberger, for instance, had recently bought a new digital watch and a new programmable calculator. He had taken both of them apart, in order to figure out how they worked. "I usually get things back together, too," he remarked. Another of the Hardy Boys called computers "the ultimate toy." One did not have to look far for the source of the delight with which many of them delved into Eagle. Engineering at its best resembles play.

Some members of the team cheerfully professed ignorance about, and little interest in, the ultimate fruits of their labor. Ken Holberger remarked, with a smile, "A sense of the applications is somewhat missing. But it doesn't matter. We say that the ultimate goal is to build a machine to run a multiprogramming reliability test. But I understand that people who buy computers do run other

programs on them . . ." Some worried about what would be done with their machine, but for all, such issues remained comfortably remote; they would have no control over Eagle's ultimate destination.

One of the young engineers remarked, in the middle of the long debugging, "You have narrowed your field of vision to a small world . . . When you're concentrating on that little world, you leave everything else out." He did not entirely approve of this state of affairs, he said, but he did not ask that it be altered. The world that Eagle made for many of them was narrow, but it was also a world of clarity, in which good and bad could be defined. All they had to do was to make this computer come to life, and that was itself such a challenging task that beside it most other considerations seemed dull.

When they had started debugging, the wires on the backs of the prototypes' boards had all been blue. They made their changes with red wires. The backs of the boards got redder and redder.

As the summer wore on, new boards arrived at the lab, ones in which the functions of wires were assumed by metal engravings. Two new prototypes were assembled. One part of the Eclipse Group continued to refine and expand the "microcode," the basic vocabulary of signals that endowed Eagle's circuits with the ability to follow instructions. In another division of the company, a team of software engineers was working to create for Eagle the complex set of programs called an operating system. All phases of the project were proceeding well, but the debugging of the hardware still wasn't at an end. In August, an executive from the engineering division asked Rasala when he thought the Hardy Boys would finish. Rasala looked squarely at the executive and said, "I *don't know*."

Rasala no longer felt able to participate in the scheduling game. It troubled him to make promises that he could not fulfill. On the other hand, he thought they had better finish by the end of September. "Our credibility, I think, is running out," he said, one evening early in August.

They still had a long way to go; they were tired. So was Rasala. He had not been doing much of the debugging, but he had usually participated in the dreary work of altering the boards, and for the past seven months he had made it his policy to be on hand for most of each day's two shifts. Now even the speed at which he talked had decreased. Walking toward the lab, he spoke in short sentences and paused in between them. "The momentum has slowed down dramatically . . . Back in the early days, when nothing worked, it was easy to find things to do . . . Now almost everything works . . . The problems are harder to find . . . They take days . . . The people are more tired, I think . . . The problems may be less interesting . . . Some are more complicated . . . This is the grind part, I think." Experienced members of the Eclipse Group held that when a computer reaches the last stages of its incubation, progress usually comes slowly.

ON SATURDAY, THE FIFTEENTH OF SEPTEMBER, Rasala inaugurated "the last big push." "We're gonna work 'round the clock, Saturday, Sunday," he said. "We're gonna do it all, and bathe in glory, guys." Late on Saturday afternoon, however, while running one of the last diagnostic programs, Gollum came down with what appeared to be a terrible bug.

Guyer, Holberger, and Rasala took it on. After an hour or so, Rasala felt stymied. "It's some complex interaction of the IP, the ATU, microsequencer, and IOC. And probably the ALU, for that matter," he said to himself. In effect, he was thinking, "The whole damn machine is involved." They could not find the cause of the failure, and after many hours of searching, they all went home. Rasala's big push came to a halt.

On Sunday morning, all three engineers gathered around Gollum and Coke again. They discovered that Coke did not commit the error in question, in spite of the fact that Coke was supposed to be identical to Gollum. Rasala and Holberger invented a hypothesis, which they discussed. In the lab, three-way conversations often led to confusion, so Jim Guyer pushed his chair away from his colleagues. He rocked back and gazed absently at Coke. He was staring at the circuit boards that were lined up like a shelf of books inside the machine's open frame. Suddenly, he realized that Coke had one more board on its shelf than Gollum had. Coke had an extra board of main memory. Guyer swore.

"What?" said Rasala.

Guyer was already leafing through the listing that described the steps in the diagnostic program that Gollum was failing. "I don't want to tell you yet," he called over to Rasala. "But I think I've got it."

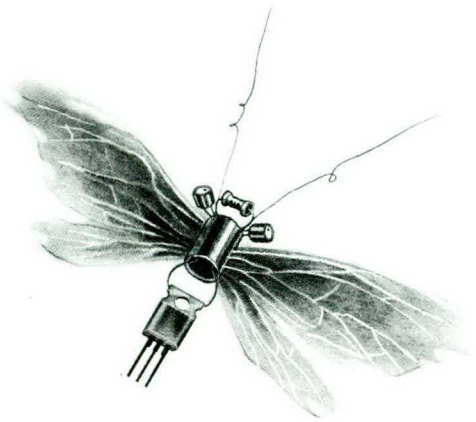
"You tell me now!" said Rasala.

Guyer brought over the document. He reminded Rasala and Holberger that Coke had two boards of main memory installed, whereas Gollum had only one.

Holberger looked at the listing of the diagnostic. "Yup," he said. The person who had written the diagnostic program had inadvertently directed that the computers cross from a section of storage located at the end of one main memory board to a section at the beginning of a second. But in Gollum there was no second board. So when Gollum came to this instruction, it failed. "It fell off the end of the world," said Rasala.

The ratiocination continued. They had stepped into a well-known trap. Diagnostic programs, which are designed to locate bugs, sometimes have bugs themselves. But mainly, the three engineers agreed, they themselves were at fault. They had failed to recognize the error promptly because they were used to thinking in the numbering system that they had employed in their last machine. Eagle used a different system. "And we didn't change our way of thinking," said Rasala.

The problem was easily rectified, but they had lost most of a day. "It's kind of embarrassing," said Guyer.



BY THE EVENING OF SEPTEMBER 20, THE PROTOTYPES had mastered the first but not the second of the multiprogramming reliability tests. That night in the lab, in a confident moment, Rasala and some of the others on the team attempted to make one of the prototypes run the program for a well-known computer game called "Adventure." If the machine ran this program, it would be the first time, after more than a year and a half of labor, that an Eagle had done more than run tests. It would be a launching of sorts. Everyone in the lab gathered around the prototype's system console. Rasala sat at the controls. He typed some instructions, in order to initiate the running of the program. In a moment the console scratched out this message, in reply: "FATAL ERROR."

What was wrong?

"We don't know," said Rasala. "We don't know everything about the machine."

Reaching for some comforting words, an observer said, "Well, at least some of it works."

"Yeah," said Jim Guyer. "And someday we'll know which part."

For the next several days Eagle failed its last multiprogramming test mysteriously. In the jargon of the lab, the computer was "blowing away," "crashing," and "going to never-never land" after every four hours or so of smooth running. But on September 25, Rasala announced, "As of this morning, Eagle ran multiprogramming twelve hours overnight without failing."

The cause of the failures had been trivial; noise had been to blame. The debugging was not finished, however. The most advanced of the prototypes was running all of the toughest diagnostic programs, but it was failing occasionally when asked to perform some of the easier ones. The Hardy Boys would rush to their analyzers, run the low-level test again, and the failure would not occur. Clearly, it was a "flake." But where did the problem reside?

The executive from the engineering division often visited the lab. One evening he came in and was informed about the "flake." At that moment, one of the circuit boards of the most advanced prototype was sitting out on an extender. The executive walked over to the computer and, to the engineers' horror, grasped the circuit board by its edges and shook it. At that instant, the computer failed. The problem lay in that board's sockets. The engineers replaced all of them; when they were done, the flake was gone forever.

Holberger paid this tribute to the executive: "He got it to fail by beating it up."

On October 8, a maintenance crew came into the lab with a large dolly. Onto it they loaded a prototype Eagle, and, accompanied by several members of the Eclipse Group, they wheeled the machine down a hallway to the software department.

The veterans of the Eclipse Group subscribed to a hypothesis called "the theory of the last bug." It stated that no art could remove from any computer all the flaws in its design; a computer would become obsolete or would stop functioning, some centuries hence, because of dust in its circuits, before the last of its bugs appeared. By their own theory, they would never finish debugging Eagle. Nevertheless, when they shipped it to software, they reached an approximate end. In Eagle, in "la machine," order now reigned over chaos. "It's a computer," Rasala declared.

Several months passed before it became apparent that Eagle was indeed a very good and reliable computer and that it was going to bring in a great deal of money. In the interim, Rasala held to his cautious line, as if by naming the worst he could prevent it from happening. "There's still a small but finite chance that there won't be an Eagle," he'd say. Indeed, only a few weeks before the machine's scheduled public debut, a truly terrible bug appeared.

The Hardy Boys spent several days attempting to isolate its cause. They got close enough to know that they were near to the source of the problem, but they could not fully identify it. Then Holberger said that though he could not name the origin of the bug, he knew how to fix it. His solution involved the addition of only one wire to a circuit. Holberger said he didn't know why his repair would solve the problem; he knew only that it would. He added his magical wire to Eagle, in the software department's lab. One software engineer remarked that hardware design certainly was a peculiar art. Jim Veres offered a better explanation. He said, "A computer to the people who built it, it's part of them. You can almost feel what's wrong with it." □