Can You Count on Your Computer?

By IVARS PETERSON

Illustration by Donna Ward

William M. Kahan is picky. It bothers him when a calculator shows 10 consecutive 3s as the answer for 1 divided by 3, and 11 consecutive 3s for 9 divided by 27. The answers should be identical. He worries that on some computers the order in which numbers are multiplied sometimes makes a difference to the answer, when everyone knows, for example, that 3 x 2 is the same as 2 x 3. Moreover, Kahan says calculators and computers occasionally give "completely erroneous results for innocent-looking problems, ... and sometimes the false answers look quite plausible."

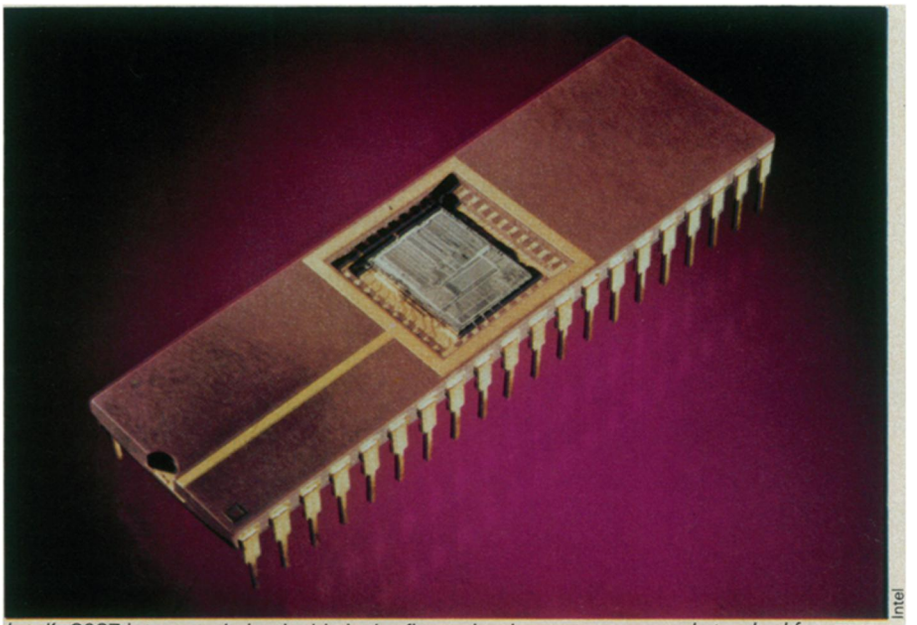
In his office at the University of California at Berkeley, Kahan keeps a drawerful of calculators. For any visitor, he can select one of the calculators and nimbly manipulate the keys to unveil an error or inconsistency in the answer to a particular problem. Kahan says, "Funny things can happen on even the best products available." This applies to computers, too.

For more than 20 years, Kahan has complained about the quality of the arithmetic computers perform. The errors most difficult to identify and eradicate involve flaws in the design of computer systems both in the hardware (the electronic circuits that make up a computer's brain) and the software (the programs of instructions that govern a computer's manipulations). Last year, a subcommittee of the Computer Society of the Institute of Electrical and Electronics Engineers proposed standards for microprocessor arithmetic, and a new subcommittee is looking at arithmetic standards for larger computers.

Manufacturers are starting to listen to the complaints of scientists and mathematicians who value accurate, reliable computer arithmetic. A few months ago, David S. Walonick, a computer programmer and consultant in Minneapolis, discovered a particularly blatant mistake. He was astonished to find that, on his new IBM personal computer, 0.1 divided by 10 equaled 0.001, instead of 0.01. "It's the type of thing where you sit and look at the terminal, and your mouth just drops open," Walonick says. "I was running test data through my package, and it was coming up with wrong answers."

At first, Walonick had difficulty in persuading IBM that the computer was making a mistake. "When I called them, I was told that beginning programmers have

By handling numbers in unexpected ways, computers and calculators create headaches for computer programmers, set traps for unsuspecting users and sometimes produce wrong answers



Intel's 8087 integrated-circuit chip is the first to implement a proposed standard for microprocessor arithmetic.

problems like that," Walonick says. "They weren't even willing to try it when I first called." However, a story in the New York Times brought a response, and the company released a corrected version of the operating system program to circumvent this problem.

Many computer mistakes arise out of decisions computer designers make on how to handle numbers that can't be represented exactly on a computer or a calculator. For example, the fraction one-third (1/3) can't be represented exactly as a decimal fraction. No matter how many 3s are added to the end of 0.3333 ..., it is never exactly equal to one-third. Calculators, and large computers, too, work with numbers having a definite number of digits, frequently 10 or 12 decimals. Therefore, many numbers and the results of calculations must be rounded or chopped off, introducing unavoidable error.

A further complication is that most computers actually operate in the binary number system, in which all numbers are represented by strings of 1s and 0s. (For instance, in the binary system, two becomes 10, three is 11, four is 100, five is 101, and so on.) Certain numbers in the decimal system, like 0.1 and 0.99, are impossible to represent exactly as binary numbers except as infinitely long strings of binary digits. More errors accumulate when the computer has to translate its internal binary numbers into decimal notation for the benefit of users.

In the IBM personal computer, a designer made a mistake in the conversion of binary to decimal notation in one special case. Jeanette A. Maher of IBM points out that the computer calculated accurately in binary although it erred in what it displayed as a decimal. "It was only in very isolated situations when users were using the BASIC language interpreter that the decimal point was misplaced by one position," says Maher, "and only under unusual and narrow conditions."

Kahan says that despite its limitations, a calculator or computer need never deliver misleading answers. "Calculators can be designed in such a way that if a user does encounter strange output he can be sure that it is not a consequence of anything capricious that his machine has done to him but must be attributed to his data, his problem, or his procedures," Kahan says. "Somehow, we have to get mathematical ideas into packages that can be used safely without obliging users to understand all the details."

Kahan asks rhetorically, "What would happen to our society if everybody who wished to use a telephone, a television set, a car, a detergent, a plastic toy or a computer were obliged first to learn at least a little about how it was made and how it works internally, and then to test it himself for hazards and other surprises?"

William J. Cody, a mathematician at the Argonne National Laboratory, says, "Numerical analysts have been trying to tell computer designers for the last 20 years how to do the job right, and they've never got the message." He says most designers are electrical engineers who are proud of the fact they can do something a little bit faster than their competition, and they don't seem to worry about whether it's right or not. "I don't think they sometimes realize the errors they've made until those things are cast in concrete, and then it's too late," says Cody.

Often, there are several different ways of calculating mathematical quantities. Sometimes, designers choose a procedure or formula (called an algorithm) that doesn't work in all cases. One example is the calculation of the slope of a straight line that runs through several given points. Some calculators that allow this calculation at the touch of a key occasionally fail to give a correct result for reasonable data. The formula the designers chose allowed rounding errors that affected the answer. Kahan says, "The mistake made by the designers of the calculators ... was their assumption that a standard formula found in many texts was *the* way to do the calculation."

In some ways, large computers are even more fallible than handheld calculators. Kahan tells this "horror story" from several years ago to illustrate his point.

A graduate student in aeronautical engineering was testing his ideas for enhancing an aircraft's lift at low speeds by solving a complicated system of differential equations on an IBM 7090 computer. After running the simulation, the student found that his modified airplane crashed. At the time, Kahan was concerned that the IBM program for calculating logarithms seemed to be inaccurate in several places. He wrote a substitute program and tested it on previously run sets of results.

"There were only two sets of results that were different," says Kahan. "One set belonged to a psychologist. He didn't care because he had already published something." In the other case, the graduate student's airplane no longer crashed. After further checking, the student accepted the new result, was delighted to have a thesis topic and went on to get his degree.

Later, after the university obtained an IBM 7094, the plane crashed again when the calculations were done to high accuracy. The student found he had to use an arithmetical subterfuge to save the air-

"A little neglect may breed mischief...
for want of a nail the shoe was lost;
for want of a shoe the horse was lost;
and for want of a horse the rider was lost."
— Benjamin Franklin

William M. Kahan

plane. Whenever the computer had to subtract 1 from a number slightly less than 1, he programmed it to subtract 0.5 twice instead. Apparently, in the first case, the computer prematurely discarded the last digit in order to do the subtraction, while this was unnecessary in the second case. The final answers were substantially different. The lack of a guard digit, an extra digit that is kept to ensure accuracy during computations but is not displayed, is a common problem in both calculators and computers.

After graduation, the student went to work for an aircraft manufacturer. When he ran his program on the company's Univac 1107 computer, the aircraft crashed again. This time, the computer had automatically transformed 0.5 subtracted twice into the number 1, a more efficient expression (it thought), and the original guard digit problem reappeared.

Kahan says, "Regard this as being more nearly typical of what happens in engineering practice; namely, that something goes wrong, you believe the computer, you say the device doesn't work, and you decide you've got to do something else."

This story also illustrates how different one computer can be from another. This presents headaches for computer programmers who want to write programs that will run on a variety of machines. A program written for one company's computer will not necessarily run on another's computer because of seemingly trivial differences in how they perform arithmetic, round off numbers and handle computations that result in numbers larger or smaller than the computer can hold. At the same time, programmers must watch for each computer's idiosyncrasies and program around them by providing tests and alternate routes to make sure that the computer does not stop in its tracks, unable to cope with an unexpected division by zero or some other intractable problem.

Examples of anomalies abound for every brand of computer. For example, some numbers on several large Control Data Corp. computers test as being different from zero, yet if you multiply by them, the product vanishes. The numbers act as zeros in multiplication and division. The Cray 1, one of the world's fastest computers, has a similar set of numbers. These work for multiplication and division, but if you add one of these to itself, the sum is zero. Cody says, "Admittedly, some of these numbers are on the very fringes of arithmetic systems. They're not things that you would run into in everyday computation, and yet their mere existence means that you have to write general-purpose software in such a way that it is tolerant of such numbers."

"What makes tests and branches expensive is that programmers must decide *in advance* where and what to test," Kahan says. "They must anticipate *every* undesirable condition in order to avoid it, even if that condition cannot arise on any but a few of the machines over which the program is to be portable."

"Programmers take pride in coding around these perversities," says Kahan. "Unfortunately, we may not have reckoned the cost." In addition to the programmer's wasted time, some programs accept an unexpectedly limited range of data; some are less accurate and more complicated to use than they should be; some are less helpful than users would like when things go wrong; and some are slow.

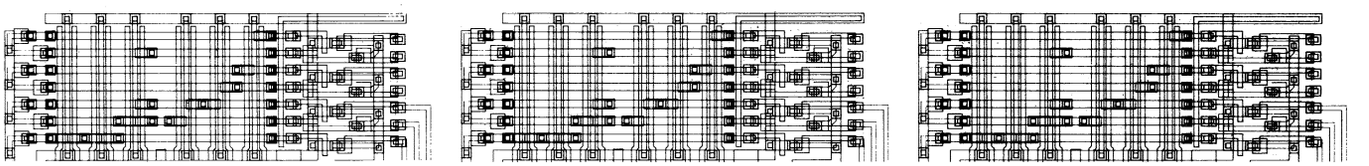
In many business and government applications, computers do not perform complex computations, but rather process data by sorting or classifying information, or control operations. Because arithmetic errors are much less likely to arise in these applications, designers have been less concerned about the quality of arithmetic in computers dedicated to these duties. Problems arise when such computers are also used for scientific and engineering calculations, or for business purposes, like the calculation of mortgage interest rates and payments, which requires arithmetically subtle algorithms. For instance, Kahan suggests that businessmen should not always trust all the figures displayed on some financial calculators. Although discrepancies are likely to be only a few dollars out of sums of millions of dollars, the differences may be big enough to attract a bank examiner's eye and waste his time.

Cody says that when IBM brought out its highly successful 360 line of computers, the company seemingly ignored the scientific market and concentrated on the commercial market. Fred N. Ris, manager of computation-intensive systems at the IBM Research Center, admits that the early 360 models had major flaws in their arithmetic, now corrected. One involved the lack of guard digits for high-precision computations. "It suffered from things that we know now ought to be done a little bit nicer," Ris says.

Cody is also concerned about the new computer language Ada being developed for the Department of Defense. "I think there are features in the language that, in my opinion, are somewhat questionable," he says. "It really was not intended originally as a scientific programming language, and unfortunately it is being regarded in that light right now." As a result, many arithmetic operations are not defined or specified properly, and do not meet an arithmetic standard like the one recently proposed for microprocessors.

A draft of the proposed standard for "binary floating-point arithmetic" was published in the March 1981 COMPUTER, along with comments on the standard's merits. The term "floating-point" refers to the format computers and calculators use to represent numbers internally. In the decimal system, for example, the floating-point (or scientific notation) form of the number 0.032 is 3.2×10^{-2}. This particular standard applies specifically to arithmetic performed in the binary system, with 32 places or "bits" set aside to represent each number.

John Palmer, a mathematician at Intel Corp., indirectly sparked the standard's development. He persuaded management to adopt a company-wide arithmetic standard for all its lines of microprocessors so they would be compatible and at the same time to develop the best possible floating-point arithmetic. When competitors heard rumors about the development, they became a little nervous, and as Kahan facetiously puts it, said, "Let's slow them down by forming a committee."



Ris says part of the motivation for producing a standard is that it allows manufacturers to compete strictly on a price-performance basis without "endless verbal contests about whose arithmetic is better." Products from different manufacturers also would be compatible so that the whole industry would benefit and grow.

The subcommittee, instead of surveying what had been done in the past and coming up with something close to past and present industrial practice, started from scratch. Ris says, "They got away with it because of the excessive grubbiness of many floating-point arithmetics already in the field, all the way from super-computers down to pocket calculators."

The effort and many of the subsequent recommendations were controversial. Ris says many people were skeptical and worried that the committee was going off to define something nobody wanted to build. "People were jumping up and down on the table and saying one couldn't possibly implement various functions at a reasonable cost," he says. Intel cut the debate short when its 8087 microprocessor arithmetic chip appeared showing that the principles in the proposed standard could be built into an integrated-circuit chip reasonably and efficiently. "I don't think it [the standard] would have happened without that," says Ris.

Although the standard is still a proposal and a final draft has yet to appear, a variety of microprocessor manufacturers and software developers have adopted it. Now a new working group, headed by Cody, is beginning to develop a more general arithmetic standard that accommodates a wider variety of computer formats and bases. Getting such a standard accepted once it's developed, however, may be much more difficult.

Daniel W. Lozier, a mathematician at the National Bureau of Standards, says, "Manufacturers [of computers] have a strong vested interest in doing things the way they have developed internally." Hardware designers may have valid reasons for their choices, or the company may wish to maintain compatibility with previous product lines. Performance may be affected, and extensive software libraries would have to be rewritten.

Even the proposed binary floating-point arithmetic standard will not guarantee correct results from all numerical programs. "But the standard weights the odds more in our favor," says Kahan. "It does so by meticulous attention to details that hardly ever matter to most people but matter, when they do matter, very much."

Errors, although rare, will still appear, Ris says, "We know it happens all the time, and we know that a certain amount of it is inevitable. The question is what can we do to provide environments in which we reduce the inevitable to its irreducible minimum, and at the same time try to provide mechanisms for warning users they may be treading on thin ice and ought to investigate the results they got a little bit more closely."

Kahan goes further. "The real horror of this situation is that the incidence of error in one's final conclusion is unknowable," he says. "We do not know how often numerical results are considerably more wrong than is believed by the people who use them."

Lozier says computer users tend to look for grosser errors than those likely to be caused by faulty arithmetic. "But in any kind of careful application of the computer, there are independent checks," he says. "In other words, they look for independent confirmation that what they're computing is reasonable."

Kahan says, "Those funny things computers do can cause confusion. Some of the confusion can be alleviated by education, whereby we come to accept and cope with these anomalies that are inescapable consequences of the finiteness of our machines. But education cannot mitigate the demoralizing effects of anomalies when they are unnecessary or inexplicable, when they vary capriciously from machine to machine, when they occur without leaving any warning indication, or when no practical way exists to avert them."

Today's calculators, compared with those of five years ago, make far fewer mistakes. New microprocessors built to the standard will have more reliable arithmetic than earlier models. Errors by computers occur only occasionally, under special circumstances, but exactly when and where they appear adds a tiny, maddening uncertainty to any computation. The benefit of careful attention to detail in the new standards that are developing, says Kahan, is "that consequences follow from what we have done rather than from what has been done to us." □

Calculators can do funny things

You can check your own calculator or personal computer to see what kinds of arithmetic anomalies arise. Some calculators are better than others; older calculators, even in the same product line, tend to make more mistakes.

Try this. Start with 1, divide by 3 and then multiply your answer by 3. Subtract 1. What answer do you get? On a TI-55 (made by Texas Instruments), for example, the answer is zero. On a TI-25, it's $-1\text{x}10^{-8}$. Repeat the process, but this time in the last step, instead of subtracting 1, subtract 0.5 twice. On a TI-25, the new answer is $-1\text{x}10^{-9}$, while the TI-55 gets $-1\text{x}10^{-11}$.

Although the differences in the answers appear trivial, the difficulty is that if a programmable calculator or computer performs arithmetic in the same way (and many do), then it may come across division by zero depending on how the machine has rounded off earlier results. The problem is the lack of a proper guard digit.

Is 2^3 exactly equal to 8? Calculate 2^3, then subtract 8. On a TI-55, the answer is $2\text{x}10^{-10}$. On your calculator, does $\pi x e$ give the same result as $e x \pi$? Is 1 divided by 3 the same as 9 divided by 27?

If your calculator has trigonometric functions like sin, cos and tan, check to see whether trigonometric identities (mathematical relationships among the functions) are preserved. For example, $\tan 20° = \tan 200° = \tan(2\text{x}10^n)°$ (for any power of ten). Compare your answer for tan 20° to, say, $\tan(2\text{x}10^6)°$. Does your calculator give the correct values for $\sin\pi = 0$, $\cos\pi = 1$ and $\tan\pi = 0$, although the calculator cannot represent the number π exactly?

In more complicated calculations, other peculiar behaviors can show up. Although today's calculators are very good and in most applications will give accurate, reliable results, users must be wary of exceeding the sometimes arbitrary limits of any machine. *I. Peterson*