IEEE 754: An Interview with William Kahan

If you were a programmer using floating-point computations in the 1960s and 1970s, you had to cope with a wide variety of configurations, with each computer supporting a different range and accuracy for floating-point numbers. While most of these differences were merely annoying, some were very serious. One computer, for example, might have values that behaved as nonzero for additions but behaved as zero for division. Sometimes a programmer had to multiply all values by 1.0 or execute a statement such as $X = (X + X) - X$ to make a program work reliably. These factors made it extremely difficult to write portable and reliable numerical computations.

In 1976, Intel began to plan for a floating-point coprocessor for the Intel i8086/8 and i432 microprocessors. John Palmer convinced Intel that they needed to develop a thorough standard to specify the arithmetic operations for their coprocessor so that all Intel processors would produce the same results. Because William Kahan had extensive experience with the IBM, Cray, and Control Data Corp. (CDC) floating point, he was one of the few who understood the challenges of writing accurate numerical code. In 1976, Kahan's influence on floating-point processing escalated when Intel hired him as a consultant to help design the arithmetic for the 8087 processor.

As a result, he had a hand in the birth of the IEEE 754 specification for floating-point computations.

—Charles Severance

Editor: Charles Severance, Michigan State University, Department of Computer Science, 1338 Engineering Bldg., East Lansing, MI 48824; voice (517) 353-2268; fax (517) 355-7516; crs@egr.msu.edu; http://www.egr.msu.edu/~crs

I think that it is nice to have at least one example—and the floating-point standard is one—where sleaze did not triumph.

THE BEGINNING

Charles Severance: When Intel hired you as a consultant in 1976, what did they want you to do?

William Kahan: The folks at Intel decided that they wanted really good arithmetic. The DEC VAX was really not that bad, so my reasoning went: Why not copy the VAX? Intel wanted the best arithmetic, so Palmer and I got together to think about what the best arithmetic should be. One of the things Palmer told me was that Intel anticipated selling these coprocessors in very large numbers. The best arithmetic was what was best for a large market, which subsequently started to frighten Silicon Valley because of rumors that Intel was building floating point on a single chip, the i8087. And when they heard rumors of what was going to be on that chip, they were aghast.

CS: Out of this thinking grew IEEE 754?

WK: People have said from time to time (as a joke) that the other Silicon Valley companies got worried and joined the IEEE 754 working group. I realized at this first meeting that the members of the committee were very serious. CDC didn't bother to attend that meeting in November 1977 because it was a microprocessor committee—they had no idea that microprocessors would mean anything at all. Cray felt the same way. IBM was only there in an observer capacity—they knew microprocessors were coming but they couldn't say much.

CS: What were the meetings like?

WK: One of my friends said that attending one of these meetings was like a visit to the Grand Canyon: just awesome. In the usual standards meeting everybody wants to grandfather in his own product. I think that it is nice to have at least one example—and the floating-point standard is one—where sleaze did not triumph. Cray, CDC, and IBM could have weighed in, if they wanted to, and destroyed the whole thing. But CDC and Cray must have thought, "Microprocessors. Why worry?"

CS: What happened next?

WK: After the first meeting, I went back to Intel and asked to participate in the standards effort. Then Gerome Kunan, Harold Stone, and I prepared a draft document of the Intel specification in the format of an IEEE standard and brought it back to an IEEE 754 meeting.

CS: Were there any complications?

WK: I got Palmer's verbal permission to disclose the specifications for the nontranscendental functions on the chip, but not the specifications for the architecture. I could describe the precision, exponent ranges, special values, and storage formats. I could also disclose some of the reasoning behind the decisions. We didn't say a word about the i8087's transcendental functions—I had to bite my tongue. [*Commonly used transcendental functions include sine, cosine, loga-*

rithms, and exponentials. —CS] We were going to put the transcendental functions on the 8087 chip, and it was going to have an interesting architecture. We really didn't want to give away the whole ball of wax. Intel was going to spring a real surprise on the world. We were going to have a chip that had most of the essentials of a math library using only 40,000 transistors.

THE PROPOSALS

CS: So you brought the draft back to the IEEE 754 group, but there were multiple proposals being put forward. DEC was suggesting that their format be adopted and there were other proposals as well. The initial reaction to your document was mixed, wasn't it?

WK: Initially, it looked pretty complicated. But what distinguished our proposal from the others was that we had reasoned out the details. What we had to do was enhance the likelihood that the code would get correct results and we had to arrange it so that the people who were really experts in floating point could write portable software and prove that it worked. Also, the design had to be feasible. I had to be reasonably confident that when floating-point arithmetic was built into hardware it would still run at a competitive speed. At the same time I had to be careful. There were things going on at Intel that I couldn't talk about with the committee. This was particularly the case of the *gradual underflow*—the subnormal numbers. I had in mind a way to support gradual underflow at high speeds, but I couldn't talk about that.

CS: What happened with the proposals?

WK: The existing DEC VAX format had the advantage of a broadly installed base. Originally, the DEC double-precision format had the same number of exponent bits as its single-precision values, which turned out to be too few exponent bits for some double-precision computations. DEC addressed this by introducing its G double-precision format, which supported an 11-bit exponent and which was the same as the CDC floating-point format. With the G format, the major remaining difference between the Intel format and the VAX format was gradual underflow.

THE BATTLE OVER UNDERFLOW

[*Gradual underflow provides a number of advantages over abrupt underflow. Without it, the gap between zero and the smallest floating-point number is much larger than the gap between successive small floating-point numbers. Without gradual underflow one can find two values, X and Y (such that X is not equal to Y), and yet when you subtract them their result is zero. While a skilled numerical analyst could work around this limitation in many situations, this anomaly would tend to cause problems for less skilled programmers.—CS*]

CS: Given the advantages of underflow, why was anyone opposed to it?

WK: The primary reason that some committee members were opposed to gradual underflow was the claim that it would slow performance. After my confidentiality obligations expired, I could talk about ways of doing gradual underflow in hardware without slowing down all floating-point operations.

At one of the meetings in the late 1970s, DEC came in with a hardware engineer who said that it was going to be impossible to build fast hardware to support the proposed standard. It just so happened that we had a student, George Taylor, who had taken up the task of producing a new floating-point board for a VAX. We were going to remove the floating-point boards and substitute our own with IEEE standard arithmetic. Otherwise, it conformed to the DEC VAX instruction set. We were going to compare a good arithmetic (the VAX arithmetic) with the IEEE arithmetic and see what it was going to be like. So George came to a meeting, showed how it was going to work, and it was perfectly clear to everyone there that this was eminently feasible.

CS: Wasn't there also an attempt to prove that gradual underflow was bad from a theoretical viewpoint?

WK: Yes, DEC had been struggling to persuade us that gradual underflow was a bad thing. If they could prove it was unnecessary, there was no reason not to use DEC's exponent bias. The exceptional handling and other details could be done with small tweaks. DEC finally commissioned one of the most prominent error analysts in the east, G.W. (Pete) Stewart, to perform the study. He was to look into the error analysis aspects to demonstrate that gradual underflow was not all that I had cracked it up to be.

CS: And what happened?

WK: At a meeting in Boston in 1981, Stewart reported that, on balance, he thought gradual underflow was the right thing to do. The DEC folk who had commissioned the report were rather disappointed and they said, "OK, we'll publish this later." They were really annoyed because this was on their home turf. Having suffered that rather substantial defeat, they got disheartened.

CS: With all the success of IEEE 754, what's missing?

WK: Compilers and programming languages new and old—from Java to Fortran—still lack competent support for features of IEEE 754 so painstakingly provided by practically all hardware nowadays. SANE, the Standard Apple Numerical Environment, on old Motorola 68K-based Macs is the main exception. Programmers seem unaware that IEEE 754 is a standard for their programming environment, not just for hardware.

The new C9X proposal before ANSI X3J11 is a fragile attempt to insinuate their support into the C and C++ language standards. It deserves the informed consideration of the programmers it tries to serve, not indifference.

As new microprocessor-based computers have become widespread, we have clearly benefited from the widely available floating-point standard. Users and programmers alike need to thank William Kahan and the others involved in IEEE 754 for their efforts. For more detail on the subject, see http://www.egr.msu.edu/~crs/ieee/wkahan. ❖

William Kahan won the ACM Turing Award in 1989 and is currently professor of computer science at the University of California, Berkeley. He can be contacted at wkahan@cs.berkeley.edu.