STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

DONALD E. KNUTH
FLETCHER JONES PROFESSOR
OF COMPUTER SCIENCE

April 28, 1980

Mr. Richard H. Delp
Four-Phase Systems
10700 De Anza Blvd.
Cupertino, California 95014

Dear Mr. Delp,

I have now read the proposed standard, Draft 5.11, twice; so I seem to be entitled to comment on it, according to the rules! The point that I have thought about most seriously is the question of gradual underflow, since it disagrees with the view I had expressed in my book The Art of Computer Programming, Volume 2, and I am just now preparing the second edition of this book for the press.

As you can imagine, I was inclined to be skeptical of this part of the proposal, because it appears to be needlessly complicated just to gain a few bits at the low end of the range. My initial idea was that all numbers should be normalized, and that all exponent underflow and overflow should cause error flags and should define a normalized result to the trap handler, where the exponent of the result would be modulo 256 or whatever. The advantages of this scheme are that it can be stated simply and that error analyses are readily performed using simple formulas for relative error.

However, I have now come around to the opposite view, and I believe that gradual underflow is a wonderful invention! The modified formula for errors, namely $x(1+\delta)+\varepsilon$ instead of simply $x(1+\delta)$, is actually not much harder to deal with, and gradual underflow gives significantly more freedom to the careful programmer who wants to tune up a beautiful piece of numerical software as well as to the casual programmer who wants the computer to do the right thing without his bothering to work out any tricky math.

The fallacy in my earlier reasoning was that I was thinking of gradual underflow as merely a way to squeeze out a few drops of accuracy in a range that is rare anyway. Why should I worry about that range, when there clearly has to be an arbitrary cutoff at some point? The thing I missed was that gradual underflow adds an element of completeness to the system that seems impossible to achieve in any other way. A large variety of common problems can now be solved satisfactorily whenever their inputs and outputs are representable numbers. For example, we can divide a

complex number z by any other complex number w, whenever z and w and z/w are all representable. Such completeness would be impossible without great difficulty if gradual underflow was not used, and this would mean that nearly all numerical software would either have to report "Sorry, your numbers tweaked a bad case" on occasion or they would be extremely complicated.

In other words, I had failed to think of floating point numbers as a working system of values that should be closed under the important operations; I had held the more traditional view that they were just approximations to the real number system over a certain range, where we would hope that our problems were simple enough not to lead to anomalies outside the range. Without gradual underflow, there were lots of anomalies; these problems could be handled by extending the range of exponents, but then other anomalous situations would appear. The beauty of gradual underflow is that it closes off the anomalies nicely, making it possible to write lots of anomaly-free subroutines; the extra extension of the range doesn't lead to extra troubles, it magically fixes the ones that were already there.

Thus I strongly hope that the IEEE standard will be based on the concept of gradual underflow. At last we have a chance to get floating-point arithmetic that will attract scientific minds, instead of repelling them. Maybe in ten years we will even be able to have portable programs that routinely give the same answers on different computers. Unfortunately I am now having to change any TEX program to do only integer arithmetic, because as things stand in 1980 I dare not use floating point.

I would appreciate it if you can circulate my comments to the members of your committee.

Sincerely,

Donald E. Knuth

DEK/pw

cc: W. M. Kahan