IEEE Task P854

Minutes, 27 August 1982

A meeting of the IEEE Standards Task P854 for format-independent floating-point arithmetic was held from 13:30 until 17:55 on Friday, August 27, 1982 at Evans Hall, the University of California at Berkeley. Seven people were in attendance.

Minutes from the April meeting were agreed to.

The next meeting of P854 will be Wednesday, October 20 in Palo Alto at 10:00 a.m.

Formats. Balance between overflow and underflow thresholds continues to be troublesome. The committee removed the restrictions that the product of the two not be less than 4 nor greater than 100, leaving the recommendation that the product "should be the smallest power of the base not less than 4". This is due to the lack of an airtight technical justification for what the bounds should be, but in realization that caveats will have to be attached to codes about operating ranges which might not have been necessary with the restrictions.

Substantial time was spent considering what ought to be the relationship specified between the precision of basic and extended formats. Consideration of the two most important functions of an extended format, conversion and computation of x**y, lead to the consideration that the requirements in binary are dominated by binary<-->decimal conversion, while the requirements in decimal are dominated by x**y and are much more severe.

Consideration of what ought to be the appropriate generalization of tables 2 and 3 in Section 5.6 of P754 led to the following points of consensus. There should be a range of values for which conversions will be done with but a single rounding error. This range is characterized by those values which in decimal are expressed as an M-digit integer times a power of ten ranging from -N to N. The values of M and N are derived as follows. M is the largest integer such that 10**M - 1 is representable exactly in the extended precision format but 10**(M+1) - 1 is not. N is the largest integer such that 10**N is representable exactly in the extended precision format but 10**(N+1) is not. N is in general bigger than M because 10**N contains factors of two whereas 10**M - 1 does not; the real test is how large can N be and have 5**N representable. Thus for P754 single-extended with its 32-bit (at least) format, M = 9 and N = 13.

There is another practical constraint on M. For a p-bit basic format there will be a number K = floor (2 + p * log 2) which is the minimum number of digits having the property that when p-bit floating-point is converted to K-digit decimal floating point and the K digits are converted back to p-bit binary the original value can be recovered. (Note that for floating-point values in the range 1000 to 1001 ten bits will be used to the left of the binary point, leaving 2**(p-10) distinct values. In K-digit decimal, four digits will be used to the left of the decimal point leaving 10**(K-4) distinct values in the decimal format. The recovery property requires therefore that 10**(K-4) > 2**(p-10) from which the above follows for modest p.) If we require that M have at least the value K, we can then show that the relationship between the number of bits in extended and the number of bits in basic requires a difference of log2 100 = 6.64... . Thus, it is resolved that the extended formats in binary must have at least seven more bits than the basic formats they support for the purposes of binary<-->decimal conversion.

Reanalysis of the x**y error story show that P754 formats do not have quite enough bits to guarantee without luck of happenstance that x**y can be computed in a straightforward manner with an error of less than one half ulp. The decimal case is more difficult; it is not difficult to construct plausible formats for which double precision is required to make x**y work properly, and therefore extended cannot be entirely motivated by x**y considerations. A similar problem will exist in all large radices.

Conversions to external decimal strings. There are issues in conversion of decimal floating-point to external strings. One critical one seems to be that the inexact exception applied to decimal floating-point ought to be precise, whereas in binary to decimal it can be fuzzed for decimal arguments with more than

M significant decimal digits. For example, P754 allows the conversion of the decimal string 1073741824 to signal inexact despite that its value is exactly 2**30, to which it must convert; here the exactness is regarded as coincidental for the pain of determining otherwise can be substantial. For input of decimal strings to decimal floating-point systems, detection of inexact simply means looking at significant digits presented after the p'th to check whether they are all zero or not, and this does not seem an onerous requirement.

P754 has so far ducked the question of dealing with infinities and NaNs in external string conversions; P854 resolved to give that question fresh consideration at its next meeting.

Underflow. P754 had agreed to put a revised definition of underflow to mail ballot at their meeting the previous day; P854 decided to incorporate the latest wording of the proposition directly. Accordingly, the current draft of P854 reads as follows (notwithstanding required format generalizations):

7.4 Underflow. Two correlated events contribute to underflow. One is the creation of a tiny nonzero result between +/- 2**Emin which, because it is so tiny, may cause some other exception later, such as overflow upon division. The other is extraordinary loss of accuracy during the approximation of such tiny numbers by denormalized numbers. The implementer may choose how these events are detected, but shall detect these events in the same way for all operations.

Tinyness may be detected either

a. "After rounding": when a nonzero result computed as though exponent range were unbounded would lie strictly between +/- 2**Emin;

or

b. "Before rounding": when a nonzero result computed as though both exponent range and precision were unbounded would lie strictly between +/- 2**Emin.

Loss of accuracy may be detected as either

c. A denormalization loss: when the delivered result differs from what would have been computed were exponent range unbounded;

or

d. An inexact result: when the delivered result differs from what would have been computed were both exponent range and precision unbounded. (This is the condition called inexact in Section 7.5.)

When an underflow trap is not implemented or is not enabled (the default case) underflow shall be signaled (via the underflow flag) only when both tinyness and loss of accuracy have been detected. The method for detecting tinyness and loss of accuracy does not affect the delivered result which might be zero, denormalized, or +/- 2**Emin. When an underflow trap has been implemented and is enabled underflow shall be signaled when tinyness is detected regardless of loss of accuracy. Trapped underflows on all operations except conversions shall deliver to the trap handler the result obtained by multiplying the infinitely precise result by 2**a and then rounding. The bias adjust a is 192 in the single, 1536 in the double, and 3*2**(n-2) in the extended format, where n is the number of bits in the exponent field. The trap handler should also be sent information (one more bit) to allow it to return the same result as if the trap had been disabled, should the user so choose. Trapped underflows on conversions shall be handled analogously to overflows.

Affine/Projective. If P754 votes to remove projective mode, P854 will probably do likewise, but is presently reserving judgement on the subject.

Fred Ris
29 August 1982