IEEE Task P854

Minutes, 14 April 1982

A meeting of the IEEE Standards Task P854 for format-independent floating-point arithmetic was held from 16:00 until 19:30 on Wednesday, April 14, 1982 at Four-Phase Systems Inc, Cupertino, CA. The general tone of the meeting was somewhat subdued, owing to an earlier P754 meeting lasting until 1:30 the previous morning. In attendance were:

Werner Buchholz	IBM, Poughkeepsie
Jim Cody (Chairman)	Argonne
Chris Horne	Zilog, Campbell
David Hough	Apple, Cupertino
Vel Kahan	U.C., Berkeley
Fred Ris	IBM, Yorktown Heights
Charles White (Host)	Four-Phase, Cupertino

Minutes from the previous meeting (IEEE P854/82-2.2) were agreed to by Cody and Ris, the only attendees who had been there.

Ris agreed to be secretary for the present meeting.

Discussion of format specifications opened with specific remarks on Cody's working draft (IEEE P854/82-2.6) on examples of floating-point formats, both of existing machines and of possible P854 implementations. Kahan and Ris felt that the tables should be completed and augmented to cover as many machines as we reliably know about. It was agreed to solicit, through these minutes, such information. Readers of this paragraph should consider themselves so solicited.

The stock subject of how to characterize exponent range balance was then addressed. Cody felt that restrictions expressed in terms of emax + emin were the most straightforward; Kahan and Ris preferred bounds on the product of the overflow and underflow thresholds, since this exhibits less dependence on the radix. The latter formulation was extensively wordcrafted and resulted in the following proposal for some version of draft (about which more below):

"Exponent Range Balance. The balance between the overflow threshold (b ** emax+1) and the underflow threshold (b ** emin) is characterized by their product which is subject to the constraints:

b ** (emax + emin + 1)
shall be .GE. 4,
should be the smallest integral power of b .GE. 4, and
shall be .LE. 100."

Note that this formulation allows emax + emin to be either 0 or 1 in decimal (0 preferred), and allows the sum to be 1, 2, 3, 4, or 5 in binary (1 preferred). The intent is to give a decimal implementer sufficient freedom in representation so that the cardinality of the set of exponent values may be either even or odd, and to allow comparable functional latitude for binary, although sums of 3, 4, or 5 are clearly inferior to a sum of 1 or 2 which amounts to nothing more than an adjustment in exponent bias.

The question of minimal precision was yet again addressed and the present restriction b ** (p-1) .GE. 10**5 was reaffirmed, specifically to permit implementations of 6-digit decimal in 32-bit words without necessitating digit encoding more complicated than BCD. (At issue was a lower limit of 10**6.)

Finally, returning to the inequality characterizing the relationship of exponent range to precision, it was agreed to drop reference to a named constant of proportionality and to say simply: "(emax - emin) / p shall exceed 5 and should exceed 10."

Discussion of whether anything should be said about storage representations resulted in a stand-off, with nothing presently to be specified. At question was the proposition that mapping from values into storage

representations "should" be unique. Kahan argued that the standard should be about a mathematical model not a programming paradigm and that implementations such as B5500 which exploit redundant representations for performance (?) should not be discouraged; Ris, White, and others felt there are values in knowing that comparisons for equality (admittedly with different semantics when NaNs are involved) can be made on storage representations in a machine code environment. Kahan argued that such was subject matter for manufacturers' manuals and not standards. References in the working draft to storage formats were removed.

Kahan proposed wording clarifying the sign bit associated with the value zero. The subject had been rather thoroughly aired in P754, and was thus agreed with minimal discussion. The proposition is to specify in the enumeration of values within a format: plus infinity, minus infinity, (signless) NaNs, (signed) non-zero values, and zero, followed by the sentence, "A sign bit shall provide one additional bit of information about variables which have the value zero."

Discussion about format specification being finished for this meeting, Kahan proposed that P854 make a formal request to P754 soliciting active co-ordination between the two groups on draft language, with P854 attempting to keep as closely parallel a draft as possible to the P754 draft of the day, giving each the opportunity to borrow wording from the other for the sake of clearer exposition. It was felt that such co-operation would serve to advance both drafts more surely and quickly. The proposition received unanimous assent.

Cody and Ris had slight reservations about casting the P854 specification entirely in the P754 format. It was agreed that P854 will attempt to have two working drafts; one as closely tied to P754 as possible, and one which attempts greater clarity by abandoning format and organization conventions when that seems appropriate.

Ris made a specific proposal to start along those lines, based on a formulation originally proposed by Kahan to P754 in December 1981. The essence of the notion is to define (more rigorously than what will follow here) for an arithmetic operation in floating-point three values:

r0 the (infinitely precise) result of the operation
r1 r0 rounded to p digits (unrestricted exponent)
r2 the delivered result (restricted exponent)

Each format has an exponent adjustment value associated with it, say q, whose value is roughly 1.5 * emax.

Condition (a) is defined as r0 .LT. b ** emin
Condition (b) is defined as r1 .LT. b ** emin
Condition (c) is defined as r2 .NE. r1 and r1 .LT. b ** (emax+1)
Implementer will choose one of conditions (a), (b), or (c) and stick with it throughout. Call the choice (x).

If r2 .NE. r0 inexact will be signalled
If r1 .GE. b ** (emax+1) overflow will be signalled
If (x) underflow will be signalled

Upon entry to an enabled overflow trap handler, the value delivered is r1 * b ** -q. Upon entry to an enabled underflow trap handler, the value delivered is r1 * b ** q.

It was noted that P754 had shied away from tabular expressions, perhaps to the detriment of clarity. P854 is in a good position to explore alternatives without distracting P754.

Discussion turned to time and place of the next P854 meeting. There was no strong sentiment that P854 should meet close to the next P754 meeting on June 3. Kahan felt it would be imperative for P754 to have a publicly announced and open meeting during the SIAM national meeting in Palo Alto the week of July 19. (There will be a panel discussion at SIAM on floating-point arithmetic with which to tie in.) Ris expressed a concern that P754 may not choose to have such a meeting if it cannot complete its present

agenda on June 3, and therefore that P854 should pick up the public-relations obligation if P754 should default.

It was therefore concluded that P854 will tentatively plan to meet in Palo Alto in July, but will probably not do so if P754 chooses to. In any event, a time and place of next meeting will be determined prior to the conclusion of the P754 meeting on June 3.

The remainder of the evening was devoted to strategic discussion of programming language related issues. Kahan observed that of the implementations which today support the standard there is wide variation, some needless, in both nomenclature and protocol; there is therefore a clear danger that some significant part of the benefits of the standard may be lost as translation from one set of local syntax conventions to another intrudes on the program porting process. At the least, it would be helpful if implementers could try to reconcile usage conventions for dealing with modes and flags and in any event inform the language standards committees what they are doing and why.

Kahan felt that implementers might be more motivated to standardize to a body of code than to a group consensus process. He thus asks how can large, useful bodies of code be created with a single set of interfacing conventions to the arithmetic. He proposed that P754/P854 should serve as a clearing house to publicize those portions of existing language processors which are related to the support of the floating-point standard. This should at least cut down on the amount of gratuitous new invention. The committees in this way would be the natural focal points for suggestions how best to handle these features in Fortran, Pascal, Basic, Ada, ...

On a more ambitious note, Kahan offered to undertake the production of standardized code such as LINPACK provided only that the manufacturers provide basic support in terms of hardware and financial support of the masters' students who would do the work; Kahan would provide space, students, and supervision. He opined that others might well be inclined to make similar offers, especially in times of tight student economy. Again, suggestions are solicited from those of you dear readers who have gotten this far.

Kahan also announced his intention to run another elementary function workshop in Berkeley starting in late spring. Objective: mathematical subroutine libraries for IEEE arithmetic implementations in a co-operative, supervised, hands-on way. Contributors to provide own bodies and hardware. Numeric goal: beat Cody and Waite. Secondary objectives: explore algorithmic variants which become attractive when division is very fast (like multiplication) or very slow, variations in accuracy/time/space trade-offs with and without extended formats, ...

By this point we were all quite out of steam.

Fred Ris
15 April 1982