IEEE Task P854

Minutes, 12 April 1985

The radix-free floating-point working group of the Microprocessor Standards Subcommittee of the IEEE Computer Society met from 9:43 a.m. to 5:30 p.m. at the Fairchild Camera and Instrument Corporation in Palo Alto. Nineteen people attended.

Minutes from 1 March. Change "Aimes" to "Ames" twice in the second paragraph on page 3. Approved as thus amended.

Several people had not yet received the mailing for the current meeting. Karpinski provided additional copies for attendees who were so lacking.

Announcements.

P754 is dead; long live 754! P754 passed its final hurdle along with four other draft standards on March 21 and is now an IEEE standard applause". It will be roughly six months before one will be able to get official copies of 754 from the IEEE. Meanwhile David Stevenson seems to have disappeared and so is not readily available for congratulations. (Anybody know how to get in touch with him?) The final draft is "essentially" P754 Draft 10.1 (cf. IEEE P854/85-2.25).

Kahan is collecting information on 754 implementations. He particularly solicits more information from companies (non-proprietary -- for purposes of publication).

Reprints of August 1984 Micro article. Cody finally received 540 copies (against a request for 500). Of these, 200 have gone to Karpinski (against a current backlog of 186 orders to him) and 50 to Bob Stewart. Cody is retaining 50, and each of the eight other authors is to receive 30 copies. The article has been reprinted in SIGNUM Newsletter, Vol. 20 No. 1 (January 1985) from the original Micro plates with an appropriate credit.

Meetings. Triennial IEEE Computer Arithmetic meeting, Champaign-Urbana, June 4-6. SMU Conference on Computer Arithmetic for Scientific Computation -- preliminary announcement in mailing (IEEE P854/85-2.22) -- April 26-27. Program by invitation only. Cody has been invited to and will make presentation on IEEE standards (30 minutes). Final program will be included in next mailing.

Statements made by Kahan for the record (cf. IEEE P854/85-2.24): (1) Anything Kulisch/Miranker arithmetic can do can be done equally by 754/P854 provided that the extended format is made sufficiently wide. However, 754/P854 does not make this obligatory. (2) No published theory of computation to date (including Kulisch/Miranker) is at odds with 754/P854. (3) Kulisch/Miranker theory, on the other hand, regards exponent overflow as a disaster from which no recovery is possible. 754/P854 deal with exceptional computational events including exponent overflow as gracefully as is known how.

Bob Stewart: There are 32 TCs in the IEEE Computer Society; none are related to numerical analysis. Should there be one? Silence." Any volunteers? None." Any in favor of idea? Few." Alternative suggestion (Kahan): Cannot SIGNUM be jointly sponsored by ACM and IEEE? Cody: SIGNUM has in the past considered approaching SIAM about possible co-sponsorship.

Correspondence.

Further to the correspondence with Kulisch (IEEE P854/85-2.17 and 2.18), a subsequent letter from Kulisch to Cody (IEEE P854/85-3.5) points out that while extended arithmetic may be sufficient for accumulating arbitrarily precise inner products, its full capabilities are not necessary, as only extended precision addition and some mechanism for feeding exact products to it are required. Further, the exponent range for exact products need not be extended explicitly.

Kahan: There are semantic (software) and architecture (hardware) problems in requiring double length results or, equivalently, two results from a multiplication. However, if an operation of the form (A*B - C) can be obtained to the same precision as A, B, and C, life could be ameliorated somewhat. In particular, this allows easy computation of quotient and square root from addition and multiplication primitives (cf. current Weitek dilemma). Somewhere we need an implementation note that the trailing part of a product is essential to obtain for reasons including but not limited to support of precise inner products. How this is done must be left to the implementer; it is the capability that is essential, not the form. A linguistic problem likewise exists with a superaccumulator, but is somewhat reduced, as is the function. Whether the consequent reduction in capability is palatable is a matter of taste.

Kahan volunteers to take lead on implementation note covering above to aim at TOMS publication. Cody to draft acknowledgement to Kulisch mentioning preparation of this note.

Wilkinson correspondence (continued from 15 November 1984 meeting): David Gay to prepare response on behalf on committee.

Brakefield correspondence (P854/85-2.14 and 2.15) revisited: Gradual underflow succeeds when it does because rounding error is insignificant compared to previous or subsequent rounding errors. Unfortunately, gradual overflow does not have this property. Cody will issue clarification.

Proposed Revisions to P854.

Ris distributed a second edition of his proposal on rounding modes (P854/85-2.21) which was not considered at the March meeting (to be included in a future mailing).

Kahan: running suspect code with different rounding modes might be a useful diagnostic tool. If results turn out same or close, nothing has been established; but if results are substantially different, this can show where one might look for difficulties. In such a regime, if rounding modes are static, one must recompile. If source code is not available, only dynamically adjustable modes will do (assuming "black box" code does not save/set/restore modes). Problems with decimal constant conversion at compile time can be solved statically by invoking loophole in "should" of last paragraph of section 5.6, so this is a reduced difficulty. Run-time conversion can always be enforced through an intrinsic function; e.g., "ATOF ('3.7')".

After some discussion, it was proposed, to make the latter explicit, to add to the appendix of recommended functions and predicates:

"(11). CONV ('X') converts the string 'X' as though at run-time (rather than compile-time) to a floating-point value in the widest format supported by the implementation, paying due heed to exceptions and the current rounding direction and precision as specified in Section 5.6." Agreed. Additional wording in Forward to signal additional function above 754 also to be added.

The consensus on the main proposal was that potential reduction of code portability is too troublesome to justify increasing the variety of implementation options. Defeated 8-1.

Thomas distributed a set of four proposals (text to be included in the next mailing).

Proposal 1 would make optional the signalling of invalid operation on encountering an unrecognizable input decimal string.

Consensus: The standard should not single out either high-level languages or hardware. No change from past view that the strengthening over 754 is for the better. Hence, not adopted.

Proposal 2 would relax the requirement that an input NaN be signalling unless explicitly recognized to be silent.

The intent of the third to last paragraph in section 5.6 is that implementations which can output quiet NaNs and recover them on input should do so when possible. When not possible -- for whatever reason -- the sentence in question requires delivery of a signaling NaN as the safest course. The consensus was that the sentence read in the context of the paragraph did indeed have meaning and while further help to the implementer might be appropriate it was not critical. Defeated 7-4.

Proposal 3 would delete the recommendation to use "1/0" for infinity in output.

After some discussion, passed 9-3. Further proposal to make "Inf" an acceptable substitute for "Infinity"; passed without dissent.

Proposal 4 was slightly reworked as follows. Replace (4) in Appendix by: "logb (x) returns the exponent of x, as though x were represented with infinite range, as a signed integer in the precision of x, except that logb(NaN) is a NaN, logb(Infinity) is +Infinity, and logb(0) is -Infinity and signals the division by zero exception. For x nonzero and finite, 1 .LE. abs (scalb (x, -logb (x))) .LT. b. Agreed unanimously.

Wording will also be added to the Forward to warn of slight difference with corresponding function in 754.

Kahan proposed to eliminate the inexact exception in round floating-point to integral value (Section 5.5). Here inexact serves solely to indicate that the input to the operation was not an integer.

For historical reasons, 754 raises inexact due to non-critical adherence to the original Coonen implementation guide reinforced by the standard test vectors. The present proposal would thus make P854 directly opposed to common practice in 754 (in which there is sufficient ambiguity in specification to lead to honest differences in interpretation of what should be done) on this narrow question, and therefore precludes the possibility of implementations which conform both to 754 (in its present consensus implementation) and P854. In earlier cases of ambiguity, the Coonen implementation guide was consulted for guidance, being roughly the Federalist Papers to the 754 Constitution.

Because inexact is discussed separately in the context of rounding (Section 4) and in the context of exceptions (Section 7), the question of how to interpret the situation from the text of the standard alone seems to hinge on which of these discussions is felt to be dominant. A straw poll was taken, and there turned out to be three camps. One felt that the standard as drafted was unambiguous in specifying that inexact not be raised. Another felt that the standard as drafted was unambiguous in specifying that inexact must be raised. The third, and largest, camp felt that the standard was ambiguous (adducing as evidence the existence of the honestly held opinions of the other two camps) and that having uncovered this ambiguity it was necessary to make some concrete resolution.

It was understood that all known conforming implementations of 754 signal inexact in this operation and so a decision to adopt Kahan's proposal as the "right" thing might in consequence make these implementations suddenly non-conforming to P854. After much individual and collective soul searching on the matter, Cody closed the discussion and called the question.

Passed 9-4. Sentence to be added "This operation never signals inexact."

Reference to section 5.5 to be added to Forward.

Proposal to further add: "This operation leaves zeros and infinities unchanged." Agreed without dissent.

In the same vein: proposal to add to the second sentence of section 5.3, ", and consequently may signal inexact." Agreed without dissent.

And again, in section 5.4, "When no other exception arises, this operation signals inexact whenever its result differs in value from its operand." Agreed without dissent.

And yet again in section 5.5, "Except for not signalling inexact, the rounding shall be as specified ..." Agreed without dissent.

For scalb: "Over/underflow should be handled in the same way as for decimal string to floating-point conversion (Section 7.3)" Agreed without dissent.

Interpretation Queries (Ris).

Does the fourth sentence of Section 7 imply that a system with traps implemented shall start in an initial state with traps masked off? Yes.

Apparently there is a validation suite of programs for Unix (TM) version 5 which requires that the Unix floating-point trap be entered on deliberately generated overflows and underflows. The default setting for trapping in Unix is thus the opposite of that required for 754/P854. Is this apparent conflict real? Yes.

Balloting.

The number of voting members stands at 31. Cody will make a revised draft, and send it to the 31 with a ballot offering alternatives to approve, disapprove, abstain, or not be considered a voting member. Thirty days will be allowed for return of ballots, which will be out before May 1.

Next Meeting.

July 8 at Apple, Cupertino, 10:00. Jim Thomas to be host.

F. N. Ris