IEEE Task P854

Minutes, 17 January 1984

The radix-free floating-point working group of the Microprocessor Standards Subcommittee of the IEEE Computer Society met from 10:15 a.m. to 4:30 p.m. at Evans Hall, University of California at Berkeley. Fourteen people were in attendance.

Minutes from the 20 July meeting in Cupertino were approved.

The next meeting of P854 is tentatively scheduled for Thursday, 22 March, at National Semiconductor in Santa Clara.

Relations with the IEEE. Tom Pittman was kind enough to visit us to explain the current line-up at the Computer Society. Bob Stewart, recently elected First Vice-President of the Computer Society, is taking the lead in standards work. Stewart has appointed Pittman the Chairman of the Computer Standards Committee (former Chairman, H. Hecht), which has oversight responsibility for all Computer Society standards efforts. Working group P854 is subordinate to a parallel organization in the Computer Society, the Technical Committee on Microprocessors and Microcomputers. Mike Smolin is the newly appointed Chairman of this Technical Committee (former Chairman, D. Allison). A rough organization chart is thus:

```
IEEE
| --    ...
| --    Computer Society
        | --    ...
        | --    Computer Standards Committee (Pittman)
        | --    Microprocessors and Microcomputers Tech. Com. (Smolin)
                | --    ...
                | --    P754
                | --    P854
```

The next step in the consideration of P754 is the formation of a balloting committee, which Smolin is to form. He and Stewart are seeking nominations for this committee, some of whom will preferably have not served on P754.

Publication of P854 is now subject to the approval of Smolin and Pittman, whose recommendation is that we publish an informal draft interspersed with technical commentary.

Publications. Draft 1.0 of P854 had been submitted to the editor of Computer, who felt that standards are inherently uninteresting to his readers, and in any event Draft 1.0 was too mathematical. However, he would welcome a 1-page summary.

Draft 1.0 had also been submitted to the editor of Micro, who felt that standards are interesting to his readers and would be pleased to publish an augmented draft in the April issue, but subject to a January 15 deadline. As there may be a few days' window, it was resolved that we attempt to submit a draft during the current week (e.g. by January 20). Cody had prepared a preliminary draft by interspersing the technical notes which have been accumulating at the back of the working drafts among the text of the proposed standard, and he, Kahan, and Ris had done some additional writing and editing the previous day. He was given leave to continue the pursuit vigorously. (That pursuit continued after the adjournment of the meeting for a few hours.)

For the invited 1-page summary, Ris had prepared an outline which was generally felt to be more than could fit on a page. Kahan proposed instead that we try to publish the reading list he had prepared. Hough wrote an introduction to the list which was approved.

Kahan wished to invite all comers to submit additional material which might be added to the reading list. Product literature is acceptable provided it has some technical content. (A copy of the current list was included in the last mailing.)

Programming Language Issues. The next meeting of X3J3 (ANSI Fortran Standards) will be in Austin the week of February 6. Items of interest on the agenda include events and exception handling. Kahan proposed that P854 should be represented to address the proposals already on the table with our perspective and share our concerns and views. Boney and Kahan were given leave to represent P854 at this meeting.

Kahan distributed a paper on arithmetic exceptions (to be included in the next mailing). He proposed we pursue a fresh approach to the questions of flags, modes, etc., by attempting a bottom-up infiltration from which languages could take a uniform lead rather than starting with each individual language with its unique set of problems and vested interests. The only practical way to make this happen is through a common run-time library to which all language processor implementors could obtain efficient access. One starting point could be the set of functions proposed in late 1982 of the form

```
v = FFLAG (exception-type)      /* a simple read */
v = FFLAG (exception-type, new-value)    /* a swap */
v = FMODE (...
```

in which case a small set of common keywords is required: e.g.,

```
FFLAG, FMODE, other generic functions
names for exception-types and mode-types
names for common flag and mode settings
DEFAULT, ABORT, other standard actions
```

Further, Kahan posited that, notwithstanding the optional implementation of precise traps specified in the draft standards, that we need to be prepared for the possibility that imprecise interrupts will become a common way of life, about which we will hear more in terms of potential approaches in due course.

Paul Hilfinger (U. C. Berkeley) was our guest to lead a discussion about Ada. He told us that among the Ada standardization and certification efforts there is currently a vaccum on the subject of arithmetic which could use some filling. An earlier set of consultants on these matters appears to have returned to the proverbial woodwork. While the standards effort paid some lip-service to being IEEE compatible (if loosely), the implications were not sufficiently understood, so there remains today a gulf between Ada and IEEE conformability. However, Hilfinger believes that this gulf contains but a few subtle issues.

The Ada standard only specifies that Brown's model be applicable for m-bit binary where m is adequate to support a declared decimal precision and the exponent range cover at least $-4m \le E \le 4m$. Extensions in precision or range as defined by Brown's model are allowed. The standard is silent on questions of 1/0, 0/0, etc. Overflow and like conditions are so vaguely handled that IEEE support could be accommodated, although Inexact may possibly constitute an illicit extension, provoking side-effects at too low a level of granularity. Most other functions seem plauible through a combination of implementation-standard modules and overloading where necessary.

Other views in a more general language-issue discussion follow.

Boney is particularly concerned about standard names for infinities, NaNs, modes, exceptions, etc. and about how one specifies whether expressions like 1/0 and 0.3 (in a binary implementation) which raise exceptions (overflow and inexact in these examples) handle things at compile time or run time. The question of compile-time accounting for rounding mode is particularly vexing.

Kahan advocated that no floating-point optimizations should be permitted except at with the user's explicit permission, leading to a form of "what you see is what you get". Optimization of addressing and (non-floating-point) loop control are fair game.

Palmer agreed with the plan to unilaterally lay down standard interfaces now so that there is a chance for uniformity by the time the compiler writers get around to thinking about these issues. However, approaching a compiler group at the wrong time (which is most of the time) is counter-productive.

Kahan argued that the job of infiltrating the purveyors of run-time libraries ought to be easier than dealing with compiler or language standards groups, and he was therefore in favor of commencing this process. The alternative would be to adopt the current Apple scheme as a baseline which is at least being implemented. Jim Thomas promised to provide the Apple internal standards to Karpinski for inclusion in the next mailing so that this proposition could be considered at the March meeting.

Inexact Exception in Conversion/Round to Integer

Immediately following the meeting, Cody, Hough, Kahan, Palmer, and Ris met to continue drafting commentary to P854. We started by considering a problem presented earlier to us by Thomas concerning exceptions arising from conversion of floating-point value to integer format (Section 5.4 of both P754 version 10.0 and P854 version 1.0). The original question concerned the appropriateness of signaling Overflow during such a conversion. It seemed clear to the five of us that what both standards intend is that if an Integer Overflow signal (outside the scope of the floating-point standard) should be available, then Integer Overflow is the appropriate signal; if not, some indication of untoward behavior must be given, and Invalid Operation is the catch-all for such things. It seemed reasonably clear to us that floating-point Overflow is explicitly precluded from being raised in this operation because there is not a floating-point destination, and the language of Section 7.3 seems adequate for this purpose. Nevertheless, adding further explanatory material in the commentary to Section 5.4 to explain what was intended seemed appropriate, given that a real implementor had a genuine confusion.

In the course of drafting this explanation we uncovered a rather more serious problem. When floating-point values have integer components extracted from them either with an integer destination (Section 5.4) or a floating-point destination (Section 5.5), it appears there are some real questions about under what circumstances the Inexact exception applies.

Consider first the round to integral value operation (Section 5.5), because the discussion is simpler. A signaling NaN operand will provoke Invalid Operation because of the NaN and independent of the operation; Invalid Operation cannot occur otherwise. Overflow is precluded in all cases, because for sufficiently large operands, the operation is an identity. Underflow is precluded because all integers lie above the underflow threshold. Divide by Zero is clearly precluded. This leaves only the question of Inexact.

The elements on which the controversies are based are found in Sections 4 and 5.5. If the production of an integral value is considered the operation producing an "intermediate result", and the packaging of that result into the destination format is what is associated with the Inexact exception, then Inexact would never be raised in this operation. On the other hand, if the operand itself is considered the "intermediate result" and the rounding is applied to a virtual format in which there is exactly enough room for significant bits of integer but not fraction, then Inexact would apply.

For conversion to integer format (Section 5.4) there is an additional complication in that a very large operand may not fit into the integer destination format (the original overflow question), and then some rather more serious exception than Inexact is required. In this case, the last sentence in the introductory material in Section 7 would seem to preclude raising Inexact concurrently, irrespective of whether Inexact might be raised in cases where there is not a destination overflow.

We found a lack of unanimity on several points, with no particular pattern about which individuals were in the majority or the minority on each question.

First, we could not agree whether the standard as drafted admits to more than one good-faith interpretation of the Inexact question.

Second, we did not agree how the standard as drafted should be interpreted. If the Draft Standard is the Constitution, Coonen's implementation algorithms (Computer, January 1980) are the Federalist Papers, and our majority reading on this particular question was that, whether originally intended or not, an im-

plementer following Coonen's guidelines would be likely to signal Inexact unless specifically warned not to.

The third point of disagreement concerns what the standard should have said, independently of how it might now be interpreted. The alternatives are not complicated, but discussion served to elucidate that there are differences of understanding what the general interpretation of the Inexact condition should be. The prevailing view in this case was that Inexact should not be raised; i.e., that the integer value is the "intermediate result".

Fourth, the last sentence in the introductory material in Section 7 would seem to indicate that Inexact must not be raised if overflow occurs in round to integer format and Invalid Operation will be signalled. While all agreed that this sentence was inserted in the late days of P754 in an attempt to be helpful rather than as a specification, the majority felt that it now has the force of specification. However, it might be permissible to signal Inexact in case Invalid Operation did not arise as a consequence of overflow.

Finally, we were not unanimous on how to deal with the situation as it now exists. The majority felt that raising these issues in the commentary to P854 Draft 1.0 would not be productive. The following wording emerged: "For a non-integral value, conversion to floating-point integer (Section 5.5) is always Inexact, and conversion to a fixed-point integer (Section 5.4) is Inexact unless Invalid must be signalled (Section 7)." Note that this is interpretive commentary, and not considered a part of the standard.

The consolidated majority opinions, therefore, is that the sensible definition is that both the conversion of floating-point number to integer format and the rounding of floating-point value to floating-point integral value are intended to be exact results which do not raise the Inexact exception under any circumstances. Notwithstanding, the standard as it now stands admits to legitimately differing interpretations. Both practice in existing implementations and Coonen's implementation guide suggest that raising Inexact is the current pluralistic interpretation. Whether or not Inexact is raised in some circumstances, the observation in Section 7 about Inexact being only coupled with Overflow and Underflow clearly precludes Inexact from being raised in conversion to integer format should a destination overflow arise which must be handled with an Invalid Operation exception.

I believe it is fair to say that none of the five of us is completely happy with this situation. But I also do not think that any two of us advocate quite the same approach for clearing up the confusion.

F. N. Ris
/lar

25 January 1984