THE NEW IEEE-754 STANDARD FOR FLOATING POINT ARITHMETIC

Peter Markstein

Woodside, CA 94062, USA
peter@markstein.org

1. INTRODUCTION

The IEEE-754 standard for Floating Point Arithmetic[1] that was in effect at the time of this seminar was adopted in 1985. That standard was intended for hardware implementation, although provisions were made for software implementation for operations. In addition to required operations, an appendix of recommended functions was also specified. Default exception handling was specified, and provisions for alternate exception handling were also provided. Lacking in the standard were means to access many of the features from higher level languages.

Because of the standardization which IEEE-754 (1985) provided, it became possible to write algorithms using floating point arithmetic which could be executed on a variety of platforms and which would produce identical results. It became possible to prove statements about the behavior of floating point programs.

In 2000, the IEEE chartered a new committee to examine the IEEE-754 standard with the goals of including decimal floating point arithmetic, incorporating good existing practice, providing for reproducible results, and clarifying the standard, while not invalidating conforming implementations of the IEEE-754(1985) standard.

2. EXISTING PRACTICE ADDED TO THE STANDARD

Fused multiple-add (FMA), which computes $a \times b + c$ with only one rounding operation, has been available on several computer architectures, including IBM Power architecture and Intel Itanium. FMA is now an operation in the new IEEE-754 standard. In hardware, the instruction is usually implemented to take about the same time as a multiplication alone, and so offers a performance advantage in many common applications. With FMA, the low order bits of a product can be obtained in only one additional operation, and so becomes a powerful bridge toward higher precision arithmetic. Because IBM Power and Intel Itanium treat exception handling for the case of $\infty \times 0 + \mathrm{NaN}$, that one aspect of FMA is left unspecified.

The quad precision data type, using a 128-bit container with 113 bit significand (112 represented) and a 15-bit exponent, has been adopted into IEEE-754(2008).

The 1985 standard provided for homogeneous operations (operands and result types were of the same arithmetic type.) Furthermore, it advised against providing operations which produced results in a format which differed from the input format. Existing practice has shown that it is often desirable to have operations which produce results in a format which differs from the inputs. Such operations are especially useful in providing software implementations of operations, where the intermediate computations are performed in a wider precision. Software-implemented division and square root are examples of this practice. Therefore the 2008 version of the standard provides for inhomogeneous operations, in which the result format can differ from the input formats.

3. DECIMAL ARITHMETIC

In business applications, the use of decimal data is the norm. Especially important is that a computer performs rounding in the same manner as is common practice in financial calculations. To meet this need, decimal floating point has been incorporated into the 2008 standard, in addition to binary floating point.

For a system supporting decimal arithmetic, at least one of the two computational formats shown in Table 1 must be included. Because two existing representations of decimal data already exist, both

Dagstuhl Seminar Proceedings 08021
Numerical Validation in Current Hardware Architectures
http://drops.dagstuhl.de/opus/volltexte/2008/1448

representations are described in the standard. An implementation only provides one of the representations, but must provided conversions between the two representations.

The Decimal Encoded representation represents three decimal digits within the trailing significand by 10-bit subfields (not the obvious encoding, but one that lends itself to easy hardware interpretation), whereas the Binary encoded representation represents the trailing significand as a binary integer. The leading significand digit in either representation is encoded together with the exponent bits.

Table 1 - Decimal Computational Types

	Decimal64	Decimal 128
Digits in significand	16	34
Maximum Exponent	384	6144

Decimal formats permit leading zeros in the significand, which allows the position of the decimal point to be kept in a fixed position within a datum. Therefore, values may have more than one representation in decimal floating point. The different representations of the same quantity are called a cohort. Members of a cohort compare as being equal, but provisions have been made to distinguish between members of a cohort. Operations all have preferred exponents for their results, which have the effect of keeping the decimal point aligned.

An additional rounding mode is required in decimal arithmetic: round-to-nearest with ties rounded to the larger magnitude, which corresponds to a frequently used rounding in financial calculations. Binary floating point is not required to support this rounding mode.

4. NUMBER FORMATS

The new floating point standard defines five basic computational types, characterized by radix *b*, precision *p*, and maximum exponent *emax*, and storage width *w*. The decimal computational types are shown in Table 1, and the binary computational types are shown in Table 2. In addition, reference types of width 16 in binary, and width 32 in decimal are also defined.

Table 2 – Binary Computational Types

	Binary32	Binary64	Binary128
Bits in significand	24	53	113
Maximum Exponent	127	1023	16383

The standard also provided for extended types and extendable types; neither is required for conformance. The extended types are used to extend the basic computational types and have fixed width, but of greater precision and exponent range than the basic type. They are useful for supporting some classes of computation in a basic type, and for supporting the next wider basic type in software, if it is not provided by the hardware. Extendable formats have precision and range under program control.

5. ADDITIONAL FUNCTIONALITY

The standard requires that control of various floating point environment parameters or modes, be under the user’s control. This control may be specified statically, on a block or entire program text basis. Alternatively, the control may be specified dynamically. Control over rounding modes is an example of a mode which the user can specify. Additional modes include alternate exception handling and expression evaluation modes.

Expression evaluation modes control whether a wider format is used to represent intermediate results. Implementations may provide such modes (such as the original C mode of evaluating all floating point computations in double-precision, even if the target of a computation were single precision.) A similar widening mode allows the last computation before storing into a variable to round to the width of the target so as to avoid double-rounding. These modes are not required for a conforming implementation.

An implementation may claim that components of an elementary function library are conforming if the results of the function are correctly rounded. The standard lists functions for which such claims may be made, including behavior for special cases. Claims for conformance are made for each function and precision in which the implementation conforms.

Product and sum reduction operators are also defined, to allow fast implementations of these operators. A common example is the dot product. The object is to allow fast implementation by means of parallelism and/or exploitation of the associative rules for multiplication and addition (which do not really apply to floating point arithmetic). As defined in this standard, use of the reduction operators are not necessarily reproducible from one system to another.

6. REPRODUCIBILITY

Even under the 1985 version of IEEE-754, if two implementations of the standard executed an operation on the same data, under the same rounding mode and default exception handling, the result of the operation would be identical. The new standard tries to go further to describe when a program will produce identical floating point results on different implementations.

The operations described in the standard are all reproducible operations. The recommended operations, such as library functions or reduction operators are not reproducible, because they are not required in all implementations. Likewise dependence on the underflow and inexact flags is not reproducible because two different methods of treating underflow are allowed to preserve conformance between IEEE-754(1985) and IEEE-754(2008). The rounding modes are reproducible attributes. Optional attributes are not reproducible.

The use of value-changing optimizations is to be avoided for reproducibility. This includes use of the associative and disributative laws, and automatic generation of fused multiply-add operations when the programmer did not explicitly use that operator.

7. CONCLUSION

At this writing, there are still minor revisions in progress for IEEE-754(2008), so some details in this account may be changed in the final version.

Some aspects of floating point arithmetic are not addressed by the standard. Complex arithmetic has not been specified, although some languages do support complex arithmetic, using pairs of floating point numbers to represent a complex quantity. There have been some requests to include interval arithmetic and exact arithmetic in IEEE-754(2008), but none of the proposals have been adopted. Instead, it has been proposed that separate standards be developed for interval arithmetic and exact arithmetic, and that perhaps at a later time, such a standard would be merged with IEEE-754. (The sense of the Dagstuhl Seminar 08021 was to to form a working group to develop a new interval arithmetic standard under IEEE.)

The new floating point standard should be adopted sometime in 2008. Within a short time thereafter, a new committee should be formed to start to consider further developments of the standard for later adaptation.

8. BIBLIOGRAPHY

1. IEC 60559: 1989, Binary floating-point arithmetic for microprocessor systems (previously designated IEC 559:1989, and equivalent to IEEE-754(1985))