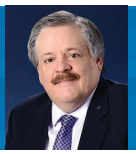
EDITOR **F. DON WRIGHT**
Standards Strategies; don@standards-strategies.com



The IEEE Standard 754: One for the History Books

David G. Hough, IEEE 754 Working Group

IEEE Standard 754, Standard for Floating Point Arithmetic, *had its beginnings more than 40 years ago. Implementations of the standard have flourished in many commercial microprocessors and other computer platforms. In June, a revision of the standard was approved by the IEEE Standards Association Standards Board. This column recounts some of the interesting history behind the standard.*

Each IEEE standard must be revised every 10 years or it will be withdrawn. IEEE Standard 754-2008, *Standard for Floating-Point Arithmetic*, was scheduled to expire in 2018; therefore, a bug-fix-and-minor-enhancements revision activity began in 2015 and with an IEEE Standards Association ballot in early 2019. That balloter-approved draft was accepted by the IEEE Standards Board as IEEE Standard 754-2019 in June 2019.

The simplified scope of the new draft is as follows:

> *This standard specifies formats and operations for floating-point arithmetic in computer systems. Exception conditions are defined, and handling of these conditions is specified.*

The IEEE 754 Working Group considered 50 drafts over a period of three and one-half years. That's a lot of work for bug fixes and minor enhancements, such as augmented arithmetic and payload operations! To put that effort into perspective, let's review the history of IEEE Standard 754.

ARITHMETIC DIVERSITY

The 1960s saw the flowering of the mainframe era, epitomized by the IBM 360, CDC 6600, and Digital Equipment Corporation (DEC) 10, but with many other contenders as well. In the 1970s, there was a corresponding boom in minicomputers, epitomized by the DEC PDP-11 and, again, many

Digital Object Identifier 10.1109/MC.2019.2926614
Date of current version: 22 November 2019

FROM THE EDITOR

In the early 1980s, this young (at the time) engineer worked for IBM and had the opportunity to first learn about many of the concepts of IEEE Standard 754 while implementing the Intel 8087 floating-point coprocessor instruction set via macros. Although the original IBM PC had a socket to add the 8087 chip, there was no way to access the 8087's floating-point instruction set from the original macro assembler. As always, actually doing drove learning. – *F. Don Wright*

competitors. These systems all had floating-point arithmetic, usually programmed with Fortran compilers, but that was about all one could generalize. Each system's arithmetic was different in greater or lesser ways, sometimes even within the same instruction-set architecture. Quite an engineering art was developed to write programs that would run equally correctly when compiled with differing compilers targeted at different hardware. The "portable" numerical programs that resulted were often much more complicated than corresponding programs targeted at just one system.

The motivation for IEEE Standard 754-1985 was to make it easier to provide portable, robust mathematical software.

STANDARDS EMERGE

Meanwhile, microprocessor development was taking over the steepest part of the development curve. Although the single-chip processors available around 1976 had no floating-point hardware and often not much integer arithmetic hardware beyond addition and subtraction, technologists and executives could foresee that powerful arithmetic engines would be feasible in a few years. To address this emerging need, Intel hired John Palmer to gather customer requirements and coordinate a specification for floating-point software for Intel processors, with the expectation that hardware would be the next step. Intel also consulted with William Kahan from the University of California, who, in the course of mathematical error analysis for scientific computing, had unearthed and publicized the consequences of many bad choices made in mainframe and minicomputer floating-point hardware, system software, and compilers. At the same time, Bob Stewart and others in the IEEE standards community decided it would be more efficient and timely to standardize aspects of microprocessors in advance, by achieving consensus among individual technical experts, rather than trying to standardize in the traditional way, after the fact, by achieving consensus among commercial organizations. Thus, the IEEE Microprocessor Standards Committee was commissioned to promote "timely development of great technical standards." In 1977, these diverse threads converged in the IEEE 754 Working Group for binary floating-point arithmetic. Under Dick Delp and Dave Stevenson, the working group labored to produce IEEE Standard 754-1985, *Binary Floating-Point Arithmetic*.

1985

The motivation for IEEE Standard 754-1985 was to make it easier to provide portable, robust mathematical software. It is far more efficient to put a little more effort into the hardware and system software specifications than to put much more effort into keeping the entire world's portable mathematical software working. IEEE Standard 754-1985 specifies formats, operations, rounding, and exceptions that affect all levels of hardware and software; but in several places, it reads mostly like a hardware specification, referring to condition codes, traps, global mode, and status bits. Mathematical software providers would have preferred to standardize high-level language facilities, but there was no chance of getting language standards interested until there were some hardware implementations and some compilers and libraries proving the concepts. Therefore, the intent was to build some hardware implementations, subroutine libraries to provide access, compilers to generate the obvious mappings of language constructs to hardware, and finally, compilers to add language features that would allow access to the novel aspects of the standard.

Typical contentious issues of this effort included questions, such as the following:

- Why not standardize DEC VAX arithmetic, which was closest to the proposed standard?
- How should underflow be handled?
- Why not do something really novel, such as symbols for overflow and underflow intervals or variable-length exponent fields?

Over time, these issues were resolved, and the final consensus was not very different from the early "CKPPS": the proposal drafted by Jerome Coonen, Kahan, Palmer, Tom Pittman, and Harold Stone. Dave Goldberg wrote an excellent summary of IEEE Standard 754-1985 for *ACM Computing Surveys*[1]; see also a 2001 article by Overton.[5]

1987

In the midst of the IEEE 754 Working Group's endeavors, a parallel effort under Jim Cody tackled decimal

floating-point arithmetic and word lengths other than 32 and 64 bits. Although potentially of greater importance to ordinary computer users who think in decimal rather than binary, this process was not as contentious because the main issues had been settled in IEEE Standard 754-1985. IEEE Standard 854-1987, *Radix-Independent Floating-Point Arithmetic*, was the result, but it had little industrial impact compared to IEEE Standard 754-1985.

2008

IEEE Standard 754-1985 was adopted in whole or in part by most microprocessor architectures as they began to incorporate hardware floating-point instructions. Because IEEE Standard 754-1985 couldn't be implemented entirely in hardware, manufacturers developed subroutine libraries to provide access to the novel aspects of the standard, in particular, modes and exception flags. The libraries tended to be different, so that once again portable mathematical software development was inhibited. Unfortunately, the latitude allowed to implementers in IEEE Standard 754-1985 had more negative consequences than anticipated, as described by Doug Priest.[9]

By now, C had become an important system and application programming language, so Rex Jaeschke convened a subgroup of the C Language Standards Committee to start developing proposals for numerical C extensions, including full support of IEEE Standard 754-1985. Much of its work was incorporated in the 1999 standard for C. Meanwhile, IEEE Standard 754-1985 had expired and been quietly renewed several times. In 2001, a new IEEE 754 Working Group was convened under Bob Davis and Dan Zuras to produce a comprehensive revision to supersede IEEE Standards 754-1985 and 854-1987. The goal was ambitious—encompassing incorporating insights from IEEE Standard 854-1987, decimal arithmetic proposed by Mike Cowlishaw of IBM, the fused multiply-add operation, a new 16-bit binary format, specifications for elementary transcendental functions, higher-level language facilities for dealing with modes and exceptions, expression evaluation, and optimization.

IEEE Standard 754-1985 was adopted in whole or in part by most microprocessor architectures as they began to incorporate hardware floating-point instructions.

Such an ambitious project became quite contentious on a number of technical points, but the biggest challenge was a dispute that led to standardizing two encodings for decimal floating-point formats. No end user asked for that, but the compromise was accepted to prevent stalling out on the rest of the standard. The most significant change from IEEE Standard 754-1985 was turning from a specification close to hardware to a specification close to higher-level language constructs. This was the ultimate intention all along, but it seemed premature in IEEE Standard 754-1985. Now, however, ISO/International Electrotechnical Commission Technical Specification 18661 specifies extensions to C to support nearly all of IEEE Standard 754-2008.[8] The extensions are 18661-1:2014, Part 1: Binary floating-point arithmetic; 18661-2:2015, Part 2: Decimal floating-point arithmetic; 18661-3:2015, Part 3: Interchange and extended types; 18661-4:2015, Part 4: Supplementary functions; and 18661-5:2016, Part 5: Supplementary attributes. These specifications are candidates for inclusion in the next version of the C standard, C2X. You can read more about IEEE Standard 754-2008 in Muller et al.[7]

2019

IEEE Standard 754-2008 was due to expire in 2018. The many IEEE Standard 754-2008 veterans, eager to avoid another eight-year ordeal, intended the next version to be an upwardly compatible bug-fix-and-minor-enhancements revision. The bug fixes are many, mostly refinements of language and the elimination of accidental inconsistencies; the description of comparison operations was substantially rewritten to better clarify the original intent. The minor enhancements are new recommended operations:

- quantum for decimal formats
- tanPi, aSinPi, and aCosPi
- augmented{Addition,Subtraction,Multiplication}
- {min,max}imum{,Number,Magnitude,MagnitudeNumber}; Not-a-Number (NaN) and signed zero handling are changed from 754-2008 5.3.1.
- {getPayload,setPayload,setPayloadSignaling}.

To retain compatibility with IEEE Standard 754-2008, all new operations are "recommended," even though the intent is for them to be universally implemented. Several additions were inspired by existing features of the C language support for IEEE Standard 754-2008 arithmetic. For various reasons, these had been omitted from IEEE Standard 754-2008. The new Pi operations complete the set defined in IEEE Standard 754-2008. The payload operations allow applications to read and write payloads of NaN representations in an implementation-independent way.

Somehow, IEEE Standard 754-2008 incorporated a defective definition of the {min,max} {Num,NumMag} operations, which weren't associative in the presence of NaNs. To address this deficiency, that definition was removed from IEEE Standard 754-2019. Instead, new operations

{min,max}imum{Number,Magnitude, MagnitudeNumber} are defined that are associative. Implementations can conform to both IEEE Standard 754-2008 and IEEE Standard 754-2019 by providing all of these operations, but the defective ones are deprecated.

THE FUTURE—2029 AND BEYOND

IEEE Standard 754-2019 was self-constrained to be compatible, with only minor extensions, with the previous IEEE Standard 754-2008. It is expected that new kinds of computational demands might eventually encompass new kinds of standards, particularly for fields such as artificial intelligence, machine vision, speech recognition, and machine learning. Some of these fields obtain greater accuracy by processing more data faster rather than by computing with more precision—rather different constraints from those for traditional scientific computing. Old ideas like block floating point have become new again.[6] There might be arithmetic standards dedicated to very specific application areas, rather than compromises intended to be suitable for a wide range of diverse applications.

IEEE Standard 754-2019 was self-constrained to be compatible, with only minor extensions, with the previous IEEE Standard 754-2008.

The most interesting new feature of IEEE Standard 754-2019 is the augmented arithmetic operations. These provide the exact result of an addition, subtraction, or multiplication in two parts that add up to the exact result. These operations were added because hardware implementations of similar functionality appeared imminent, and we hoped to have them work identically and provide the most useful functionality. To that end, a new rounding method—round to nearest, ties toward zero—was defined for these operations. So defined, they are useful for two target applications: bitwise reproducible vector summation independent of the order of summation (for example, in the presence of varying numbers of threads[2,3]) and "double-double" software that extends the topmost hardware precision,[4] used in high-precision mathematical calculations and some physics simulations, such as climate prediction and solar system stability.

More extensive background discussions of the rationales for the new operations and explanations for some confusing parts retained from IEEE Standard 754-2008 may be found in the document at http://grouper.ieee.org/groups/msc/ANSI_IEEE-Std-754-2019/background/.

The next generation of application programmers and error analysts will face new challenges and have unique requirements for standardization. Good luck to them as they develop the next revision of IEEE Standard 754!

REFERENCES

1. D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991. doi: 10.1145/103162.103163.
2. J. Demmel, J. Riedy, and P. Ahrens, "Reproducible BLAS: Make addition associative again!" *SIAM News*, vol. 51, no. 8, p. 8, Oct. 2018. [Online]. Available: https://sinews.siam.org/Details-Page/reproducible-blas-make-addition-associative-again
3. J. Riedy and J. Demmel, "Augmented arithmetic operations proposed for IEEE-754 2018," in *Proc. 25th IEEE Symp. Computer Arithmetic (ARITH 25)*, June 2018. doi: 10.1109/ARITH.2018.8464813.
4. D. H. Bailey, "High-precision floating-point arithmetic in scientific computation," *Comput. Sci. Eng.*, vol. 7, no. 3, pp. 54–61, May–June 2005. doi: 10.1109/MCSE.2005.52.
5. M. L. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*, vol. 76. Philadelphia, PA: SIAM, 2001.
6. J. H. Wilkinson, *Rounding Errors in Algebraic Processes. Notes on Applied Science No. 32, Her Majesty's Stationery Office, London*. Englewood Cliffs, NJ: Prentice Hall, 1963.
7. J.-M. Muller et al., *Handbook of Floating-Point Arithmetic*, Basel: Birkhäuser, 2018.
8. ISO/IEC TS, "Information technology—Programming languages, their environments, and system software interfaces—Floating-point extensions for C," 18661-1:2014, Part 1: Binary floating-point arithmetic.
9. D. Priest, "Differences among IEEE 754 implementations," IEEE, Piscataway, NJ. [Online]. Available: http://grouper.ieee.org/groups/msc/ANSI_IEEE-Std-754-2019/background/addendum.htm

DAVID G. HOUGH, retired from Sun Microsystems, chaired the IEEE 754 Working Group during the drafting of the IEEE Standard 754-2019 revision. Contact him at 754r@ucbtest.org.