Pete's Unsung Contribution to IEEE Standard 754 for Binary Floating-Point

Prepared for the Conference to Celebrate
Prof. G.W. "Pete" Stewart's 70th Birthday
July 19-20, 2010, at the University of Texas at Austin

by Prof. W. Kahan
Mathematics Dept. & Computer Science Dept.
University of California @ Berkeley

This is posted at
<www.eecs.berkeley.edu/~wkahan/19July10.pdf>

Pete's Unsung Contribution to IEEE Standard 754 for Binary Floating-Point

Abstract

The near-universal portability, after recompilation, of numerical software for scientific, engineering, medical and entertaining computations owes a lot to the near-universal adoption of IEEE Standard 754 by computer arithmetic hardware starting in the 1980s. But in 1980, after forty months of dispute, the committee drafting this Standard was still unable to reach a consensus.

The disagreement seemed irreconcilable.

That was when Pete helped to close the divide.

This is posted at
<www.eecs.berkeley.edu/~wkahan/19July10.pdf>

"Three Removes are as bad as a Fire."

Benjamin Franklin, Preface to *Poor Richard's Almanack* (1758)

My office has been moved twice since 1980, so now my notes for the events in question are buried in one of a few dozen full cartons,— I know not which.

Pete was asked about his contribution to the events in question. He replied …

> "… some of my papers were lost in a move and that report was among them. Moreover, I cannot remember what was in it."

Three removes … .

What I have to say is based upon my recollections now of an IEEE-sponsored conference *ELECTRO/80* in Boston in May of 1980. However, …

… Old men's recollections are notoriously unreliable.

If anybody else can recall what happened better than I can,
their corrections will be received gratefully.

What is "Gradual Underflow" ? Much ado about very little.

It is the issue that stubbornly divided the committee drafting IEEE 754 in 1980.

To describe it, consider all the finite floating-point numbers available to a chosen
Floating-Point Format's { *Radix*, *Precision*, *Exponent Range* **}** :

Radix β : β = Two for *Binary*, β = Ten for *Decimal*, … .

Precision P : P = The number of *Significant Digits* of Radix β .

Exponent Range [Ë, **Ê**] : $Ë \le$ (*Exponent* e) $\le$ **Ê** ; else Over/Underflow**!**

Given {β, P, Ë, **Ê**}, ***every*** finite floating-point number x has one of these values:

$$x = m \cdot \beta^{e+1-P} \quad \text{for } \textit{all} \text{ integers } |m| < \beta^P \text{ and } e \text{ in } Ë \le e \le \hat{E} .$$

If *Underflows* are *Gradual*, ***all*** these values are available to x ; no exceptions.

If Underflows *Flush to Zero* nonzero values x get *normalized*: $\beta^{P-1} \le |m| < \beta^P$.

Example …

Floating-Point $x = m \cdot \beta^{e+1-P}$ for integers $|m| < \beta^P$ and e in $\ddot{E} \le e \le \hat{\mathbf{E}}$.

Example: 6 sig. dec. calculator has radix β = Ten, P = 6 , $-\ddot{E} = \hat{\mathbf{E}} = 99$.

$3.14160 \cdot 10^0 = 31416 \cdot 10^{-4} = 314160 \cdot 10^{-5}$ (*Non-unique* m and e .)

Biggest finite magnitude is $999999 \cdot 10^{\hat{\mathbf{E}}+1-P} = 9.99999 \cdot 10^{+99}$.

Least nonzero magnitude is $1 \cdot 10^{\ddot{E}+1-P} = 0.00001 \cdot 10^{-99}$ ("*subnormal*")

But if Underflow is Flushed to Zero (as *almost* all calculators do), then …

Least *Normal* nonzero magnitude is $100000 \cdot 10^{\ddot{E}+1-P} = 1.00000 \cdot 10^{-99}$.

~ ~

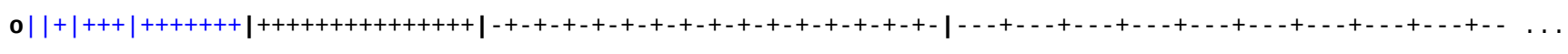
Flt.-Pt. $x = m \cdot \beta^{e+1-P}$ for *all* integers $|m| < \beta^P$ and e in $\ddot{E} \le e \le \hat{\mathbf{E}}$.
The value of x need not determine *Significand* m and Exponent e uniquely.

Nonuniqueness can be advantageous for Fixed-Point performed in Floating-Pt. registers. e.g., old Burroughs B5x00 (β = 8), latest IBM mainframes (β = Ten).

IEEE 754 Binary x determines m and e uniquely; Floating-Pt. magnitudes in memory are *Lexicographically Ordered*, speeding up comparisons and sorting.

Binary Flt.-Pt. $x = m \cdot 2^{e+1-P}$ for integers $|m| < 2^P$ and e in $\ddot{E} \le e \le \hat{\mathbf{E}}$.

Example: For P = 5 sig. bits, the first 59 nonnegative Floating-Pt. numbers on the real axis are shown below as "+" except "o" is for zero, and "|" is for powers of 1/2 . Monotonic gaps between adjacent Flt-Pt. numbers.



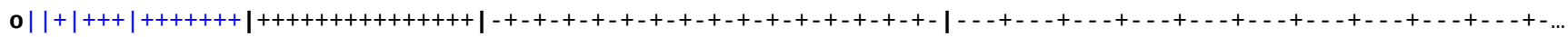
Subnormal Normal(ized) Floating-Point Numbers

The smallest $2^{P-1}-1$ nonzero magnitudes are present if Underflow is **Gradual**, but absent if Underflow Flushes to Zero and thereby surrounds 0.0 by a gap vastly bigger than gaps between other nearby *Normal* floating-point numbers.

Consequently, Flushing Underflows to Zero allows them to corrupt computation with errors huge compared with roundoff in tiny neighboring normal numbers.

When is Gradual Underflow Better than Flushed-to-Zero ? When Worse ?

Substantial experience was accumulated with both treatments of underflow on the Univ. of Toronto's IBM 7094 in the mid-1960s; see W. Kahan [1966-8].



Subnormal Normal(ized) Floating-Point Numbers

The smallest $2^{P-1} - 1$ nonzero magnitudes are present if Underflow is Gradual.

When is Gradual Underflow Better than Flushed-to-Zero ? When Worse ?

Better:

- Scalar Products: $\sigma := \sum_j x_j \cdot y_j$ correct despite underflows if σ is normal, and likewise for Matrix Products.
- Polynomials: $\Phi(x) := \sum_j c_j \cdot x^j$ despite underflows if $\Phi(x)$ or c_0 is normal.
- *Compensated Summations* of slowly convergent series or integrals or ODEs (because differences that mostly cancel remain exact even if subnormal).

Worse:

- Quotients $q := \alpha/\beta$ if underflow has made either α or β subnormal AND the program detects underflow by testing for *Zero* instead of *Negligible* or a raised *Flag*; such subnormals can render a nonzero q arbitrarily inaccurate.

o||+|+++|+++++++|+++++++++++++++|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|---+---+---+---+---+---+---+---+---+---+-…

IEEE 754 equips programmers with an Underflow Flag raised by Underflow:

5 Flags: [Inexact], [Underflow], [Overflow], [Divide-by-0], [Invalid Op'n]

- A program should be able to Sense, Clear, Save and Restore a Flag as a stored variable with a localized scope, not just a bit in a register. *Cf. APL*.
- Flag-raising is a *Side-Effect* abhorred by programming language purists, so some languages, like FORTRAN and C, have been slow to support flags. Other languages afford no support for flags or, like JAVA, prohibit them.

No surprise, then, that …

Regardless of how Underflow is treated, it will almost always be ignored.

A programmer is unlikely to be punished for Ignorance of Underflow, not even when underflows occur later and punish his program's users.

Regardless of how Underflow is treated, it will almost always be ignored.

Challenge: in 1975-6

Design an environment to support floating-point computation, by novices as well as experts, that (nearly) minimizes the risk of serious harm from frequent errors like … overlooked or ignored underflows, among other Exceptions; choices of inadequate precision and/or range; … .

Response: Intel [1980], Palmer & Morse [1984]

Intel i8087 Numeric Coprocessor to accompany i8086/88 in IBM PCs:

- Three floating-point formats: 4-byte, 8-byte, ≥10-byte *default* in registers
- +, –, ·, ÷, √, |…|, sign, compare, Float ↔ Integer Conversion, …
- Binary ↔ Decimal Conversion assistance; Directed roundings; …
- Coherent treatment of ±0, ±∞, *NaN* (*N*ot-*a*-*N*umber), and Exceptions
- Kernel of Math. Library's exp, log, tan, arctan, …

In 1977 Intel donated all but its Math. Library to IEEE p754's Committee as a draft proposal by W. **K**ahan, J.T. **C**oonen & H. **S**tone; the "**KCS Proposal**".

IEEE p754's Only Credible Alternative to the KCS Proposal :

The DEC VAX

See Payne & Bhandarkar [1980] .

This computer family had the best binary floating-point arithmetic architecture commercially significant in the late 1970s. And the VAX arithmetic included or could be retrofitted with every feature of the **KCS** proposal ***except its default Gradual Underflow***.

DEC's main advocate on the IEEE p754 Committee was a Mathematician and Numerical Analyst Dr. Mary H. Payne. She was experienced, fully competent,

and misled by DEC's hardware engineers

when they assured her that Gradual Underflow was unimplementable as the default (no trap) of any computer arithmetic aspiring to high performance.

DEC's hardware engineers declared Gradual Underflow unimplementable as the default in any computer arithmetic aspiring to high performance. But…

George Taylor, then a Berkeley grad. student of EE&CS, designed a complete implementation of the **KCS** proposal to replace the VAX 780's two floating-point accelerator boards and run about as fast; see Taylor & Patterson [1981].

Next George implemented **KCS** faster (in ECL) for the ELXSI 6400 super-minicomputer that eclipsed DEC's fastest VAXs for a while; see Taylor [1983].

DEC's engineers' precipitate misjudgment undermined the credibility and self-confidence of Dr. Payne, but not her persistent advocacy of VAX arithmetic.

Instead, what should DEC's hardware engineers have advised Mary Payne?

"Fast floating-point hardware's speed will be slowed badly by default Gradual Underflow and Subnormal Operands unless extra engineering design time is spent taking them into account from the outset, rather than retrofitting them at the end of the design."

Persistence:

Both of IEEE p754's main proposals, **KCS** and DEC VAX, evolved a little as their advocates struggled to win votes of committee members, but they remained divided persistently by **G**radual **U**nderflow *vs.* **F**lush-to-**Z**ero .

Mary Payne argued that, even if it could run fast, *GU* was a bad idea. She offered several examples of numerical software that malfunctioned under *GU* .

KCS's advocates countered with an analysis of each example to reveal that …

- Equally likely data made her program malfunction under *FtoZ* too, or …
- Her program's method was very suboptimal for both *FtoZ* and *GU* , or … .

Other examples were merely less likely to malfunction under *GU* than *FtoZ* .
How much less likely? Lacking statistics, we could only speculate.

These disputes were too exotic for most of the committee's hardware experts and compiler writers. Neither were they swayed much by letters of support for **KCS** from luminaries, like J.H. Wilkinson, unknown to most of them.

The committee's deadlock seemed insurmountable.

"Nihil est ab omni
Parte beatum."

"Nothing is an unmixed blessing." Horace (65 - 8 BC) *Odes* II.xvi.27

DEC was the host for the IEEE p754 committee's meeting in May 1980 at the *ELECTRO/80* conference in Boston. Advocates of the competing proposals prepared presentations for what turned out to be a large audience. One of the presentations commissioned by DEC was from a highly-regarded error-analyst

Prof. G.W. "Pete" Stewart III .

"… . On balance, the **KCS** proposal seems the better."

DEC was taken aback. (They never made Pete's full report public.)

Votes shifted. A consensus was deemed reached upon a draft of **KCS** slightly different from Coonen's [1979]; see Stevenson [1980]. The draft was sent out for approval by an IEEE balloting body, and then forwarded for official assent

in 1981.

1981: IEEE receives p754 Draft approved by ballot.
1985: IEEE issues 754 as an official standard.
What caused four years' delay?

My surmise is based upon fragments of (mis?)information:

DEC sent lawyers to protest to the IEEE that p754 was just an untested design, rather than a practice common to a significant fraction of the computer industry.

Their protest was overtaken by events as p754 was implemented by …

Apple Macintosh *Pascal* and S.A.N.E. (Standard Apple Numeric Environment)
Intel i8087, i80287, i80387, …, various others' clones; i960KB; Itanium
Motorola 68881/2, 68040, 88110
ELXSI 6400
Weitek 3167
National Semiconductor 160xx, 320xx, …
IBM *Power* Architecture
…

… and ultimately the DEC *ALPHA* too.

What has IEEE Standard 754 done for us?

- Averted arithmetic anarchy by teaching designers of arithmetic circuits to perform *correctly rounded* floating-point +, −, •, ÷, √ without much loss of speed, though at the cost of more transistors and longer design times.
- Facilitated development and promulgation of portable numerical software like Math. libraries, packages like LAPACK, environments like MATLAB, …
- Simplified education in Numerical Analysis & Programming by eradicating numerous arithmetical arcana. See Kahan [1992]

Thanks, Pete, for your help!

What has IEEE Standard 754 not yet done for us?

A long story for another day.

Citations:

J. Coonen *et al*. [1979] "A proposed standard for binary floating point arithmetic" Draft 5.11, pp. 4-12 of *ACM SIGNUM Newslwtter* **14** #2 <portal.acm.org/citation.cfm?id=1057521>

Intel [1980] *The 8086 Family User's Manual Numerics Supplement*, #121586-001 Rev A

W. Kahan [1996-8] "7094-II System Support for Numerical Analysis" IBM *SHARE Secretarial Distribution* SSD #159 Item C-4537 (1966); & in "Error in Numerical Computation" Univ. of Mich. Eng'g Summer Conf'ce #6818 *Numerical Analysis* (1968) <eecs.berkeley.edu/~wkahan/7094II.pdf>

W. Kahan [1992] "Analysis and Refutation of the *LCAS*" pp. 61-74 of *ACM SIGPLAN Notices* **27** #1 (Jan. 1992) <portal.acm.org/citation.cfm?id=130722>

Mary Payne & D. Bhandarkar [1980] "VAX floating point: a solid foundation for numerical computation" pp. 22-33 of *ACM SIGARCH Computer Architecture News* **8** #4, Presented at *ELECTRO/80*, May 14 1980, Boston <portal.acm.org/citation.cfm?id=641849>

J.F. Palmer & S.P. Morse [1984] *The 8087 Primer* (Wiley, N.Y.)

D. Stevenson [1980] "A Report on the Proposed IEEE Floating Point Standard (IEEE Task p754)" <portal.acm.org/ft_gateway.cfm?id=859513&type=pdf>

G.S. Taylor & D.A. Patterson [1981] "VAX Hardware for the Proposed IEEE Floating-Point Standard" pp. 190-6 (plus pp. 127-133) of *Proc. 5th IEEE Symposium on Computer Arithmetic*

G.S. Taylor [1983] "Arithmetic on the ELXSI System 6400" pp. 110-5 of *Proc. 6th IEEE Symposium on Computer Arithmetic*

Appendix: 8 FREQUENTLY ASKED QUESTIONS

FAQ #1:
How is Gradual Underflow better than Flush-to-Zero with an Exponent Range 1 unit wider?

FAQ #2:
Why could DEC VAX arithmetic not retrofit Gradual Underflow?

FAQ #3:
Does IEEE Standard 754 forbid flushing underflows to zero?

FAQ #4:
What are examples of programs that fare badly because underflow is Gradual?

FAQ #5:
What is *Compensated Summation*, and how does Flush-to-Zero harm it?

FAQ #6:
How much less likely is Gradual Underflow than Flush-to-Zero to do damage?

FAQ #7:
What has IEEE Standard 754 not yet done for us?

FAQ #8:
How does the revised IEEE Standard 754 (2008) differ from 754 (1985) ?

Every great idea, good or bad, generates a bandwagon effect that attracts adherents including inevitably also fanatics and crackpots.

"A fanatic is one who can't change his mind and won't change the subject."
Sir Winston L.S. Churchill (1874-1965)

A recent publication worthy of attention:

Handbook of Floating-Point Arithmetic by Jean-Michel Muller *et al*.
572 pp., Birkhäuser Boston (Springer), 2010
