Some thoughts on Floating-Point

Jean-Michel Muller, CNRS/LIP

FPTalks 2025 – July 10, 2025

In this brief presentation

- just a few thoughts on what we should require, specify, standardize. . . in FP arithmetic;
- goal: preserve consistance, portability, re-usability, reproducibility as much as possible.

44 years ago... a jungle of incompatible systems

Machine	Underflow	Overflow
DEC PDP-11, VAX, F and D formats	$2^{-128} \approx 2.9 \times 10^{-39}$	$2^{127} \approx 1.7 \times 10^{38}$
DEC PDP-10; Honeywell 600, 6000; Univac 110x single; IBM 709X, 704X	$2^{-129} \approx 1.5 \times 10^{-39}$	$2^{127} \approx 1.7 \times 10^{38}$
Burroughs 6X00 single	$8^{-51} \approx 8.8 \times 10^{-47}$	$8^{76} \approx 4.3 \times 10^{68}$
H-P 3000	$2^{-256} \approx 8.6 \times 10^{-78}$	$2^{256} \approx 1.2 \times 10^{77}$
IBM 360, 370; Amdahl; DG Eclipse M/600; ...	$16^{-65} \approx 5.4 \times 10^{-79}$	$16^{63} \approx 7.2 \times 10^{75}$
Most handheld calculators	10^{-99}	10^{100}
CDC 6X00, 7X00, Cyber	$2^{-976} \approx 1.5 \times 10^{-294}$	$2^{1070} \approx 1.3 \times 10^{322}$
DEC VAX G format	$2^{-1024} \approx 5.6 \times 10^{-309}$	$2^{1023} \approx 9 \times 10^{307}$

Source: W. Kahan, *Why do we need a Floating-Point Standard*, 1981.

44 years ago... a jungle of incompatible systems

- many vastly different FP environments (radix, precision, roundings, exception handling);
- some pretty good, many pretty poor;
- extremely poor portability of numerical software;
- almost impossible to prove anything rigorously (a proof needs to start from assumptions... what can you prove if you don't know what $a + b$ is?);
- need to know "wizard tricks" such as `(0.5 - x) + 0.5` often better than `(1 - x)` for small `x`;

Then came IEEE 754-1985

- single precision;
- double precision;
- correctly-rounded operations + FP $\rightarrow$ a uniform bound on relative error $\rightarrow$ the key to clean and rigorous numerical analysis *à la* Wilkinson;
- well-specified exception handling.

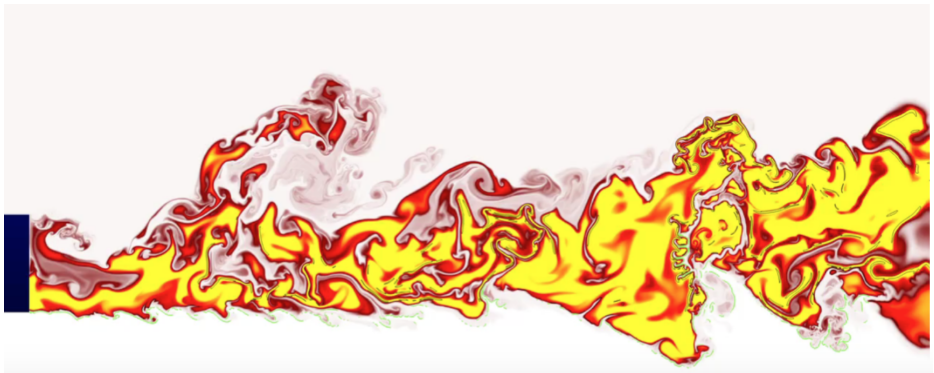
. . . and just like that, the jungle became a well-kept garden!

The jungle became a well-kept garden?

- Slight exaggeration. There were also the extended formats (essentially the80-bit x87 double extended) with their good and bad consequences:
  - generally more accurate calculations;
  - frequently allows one to avoid spurious underflow or overflow in intermediate calculations;
  - the curse of double roundings;
  - difficult to predict the accuracy of a calculation (e.g. on x86, depending on whether intermediate results are stored in 80 bit registers or 64 bit memory);
  - the curse of the C long double data type (will be double, double extended, or double-double depending on the environment).

But, anyway, IEEE-754 has been the greatest progress in numerical computing since Konrad Zuse's Z3.

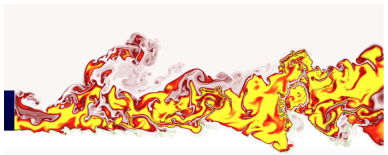
This was facilitated by the small number of application domains



Numerical simulation

Financial calculations

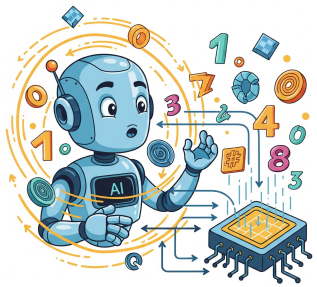
Now the situation is quite different



Numerical simulation

Financial calculation

Games, entertainment



Machine learning

Embedded computing

. . .

Extremely diverse requirements and characteristics in terms of speed, reliability, accuracy, dynamic range, provability, reproducibility, energy consumption, amount of data transfer between memory and processor, . . .

The other big changes

- The ratio

$$\frac{\text{time to read/write in memory}}{\text{time to perform } +, \times, \div, \sqrt{}}$$

  has increased by more than 150 since 1985;

$\rightarrow$ strong incentive to use small formats, even for applications that need much accuracy and/or wide range.

- heterogeneity of computing platforms: CPUs, GPUs, Tensor cores, etc.

Welcome back to the jungle?

Format	Precision	smallest normal (black) or smallest nonzero (orange)	max
binary8p3	3	3.052×10^{-5}	49152
binary8p4	4	7.812×10^{-3}	224
OCP MX FP8E5M2	3	6.104×10^{-5}	57344
OCP MX FP8E4M3	4	0.0156	448
Posit8	1–4	5.96×10^{-8}	1.677×10^{7}
binary16	11	6.1035×10^{-5}	65504
BFloat16	8	1.175×10^{-38}	3.389×10^{38}
Posit16	1–12	1.387×10^{-17}	7.205×10^{16}
binary32	24	1.175×10^{-38}	3.402×10^{38}
Posit32	1–28	7.723×10^{-37}	1.329×10^{36}
TensorFloat32	11	1.175×10^{-38}	3.402×10^{38}

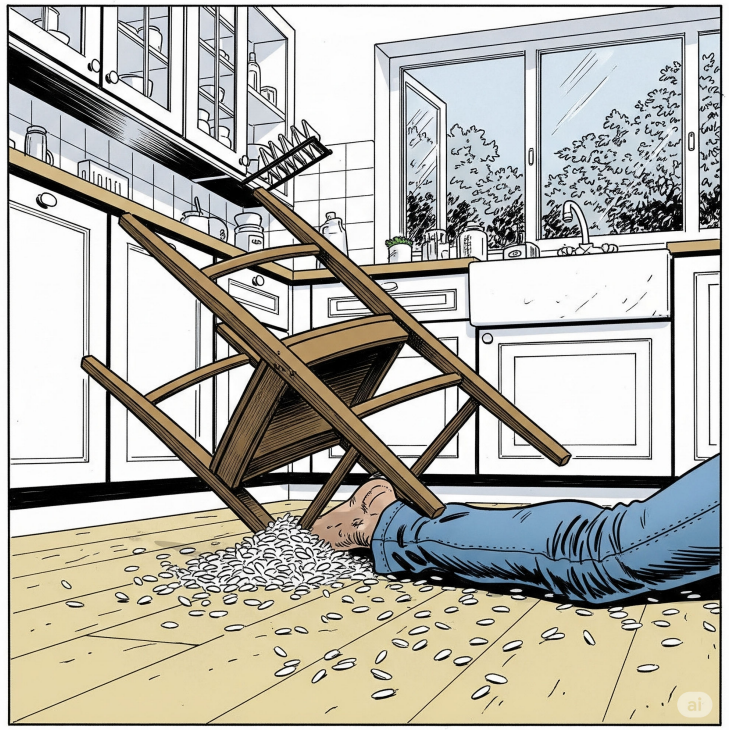
What can we do to avoid complete chaos?

Specify, specify and specify $\rightarrow$ standardize.

- help interoperability between these various formats and environments;
- don't have only one application in mind (*design thinking*);
- clear, simple and complete specification of arithmetic operations, conversions, etc.

A lesson from Design thinking

$\Rightarrow$



- those who design chairs know that they will also be used as stepladders;
- warn these "careless" users, call them idiots?... they are there anyway;
- it's best to take this use into account when designing a chair.

What does it mean for us?

There is nothing such as...

- an arithmetic (formats, operations) dedicated to one application: if it's fast, it will inevitably be used elsewhere anyway: success of mixed-precision computing;
- a purely storage format: some will do arithmetic with it;
- an extended format that only serves for implementing functions for the corresponding basic format.

Some applications need uniform relative error bound (error control in numerical analysis), possibility of formal proof (avionics, automated transportation) and therefore full specification. Must be carefully considered.

Correct rounding

- Tensor cores, "matrix" operators of the form $ab + cd + \cdots$ and similar: if not fully specified, it's difficult to prove anything. The lack of specification undermines portability and reproducibility efforts;
- the specification must endure over time $\rightarrow$ software reusability;
- the only specification that can stand the test of time is correct rounding;

  (and for example, a correctly-rounded $ab + cd + e$ has many very nice properties, see *Useful applications of correctly-rounded operators of the form* $ab + cd + e$, ARITH 2024)
- we believe the same holds for the most "fundamental" elementary functions: exp, log, sin, cos...

$\rightarrow$ see *Correctly-rounded evaluation of a function: why, how, and at what cost?*, to appear in ACM Computing Surveys (`https://dl.acm.org/doi/10.1145/3747840`).

Mixed-precision computing

- not just a passing trend: it's here to stay.
- in all complex-enough numerical calculations, all parts do not need the same precision (initial vs final steps of iterative algorithms, residuals, etc.)
- Facilitate interoperability between formats, for instance:
  - provide Round-to-Odd in the wider ones to avoid double-rounding issues;
  - promote consistency between ranges: maybe generalize the *FP32 vs BFloat16* idea and have for each word length w, beyond the standard "FPw" format:
    - a "high range" version with the exponent range of the next wider std format?
    - an "accurate" version with the exp. range of the next smaller std format?
- a numerical analyst needs to know what a 16-bit tensor core actually computes without having to do reverse engineering (again: clear, unambigous specifications are essential).

Thank you!