Overview of New Floating-Point Standards: IEEE P3109 (May 2024 interim report) and Open Compute Project OCP8/MX 1.0

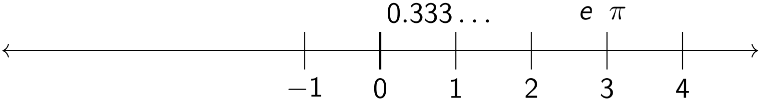
Mantas Mikaitis

School of Computer Science, University of Leeds, Leeds, UK

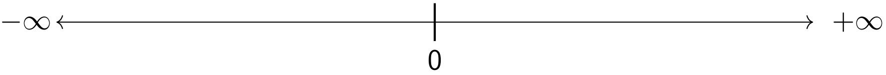
Scientific Computation Group Seminar
22 May, 2025
Slides: `mmikaitis.github.io`

Working with real numbers on computers

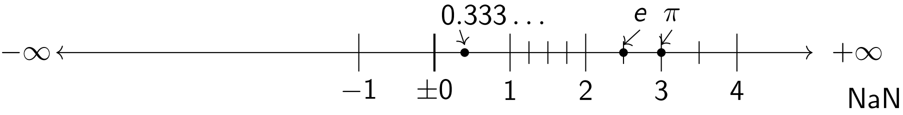
$\mathbb{R}$



$\mathbb{R} \cup \{-\infty, +\infty\}$



$\mathbb{F}$



What is floating point? Picking subsets of reals

A floating-point system $\mathbb{F} \subset \mathbb{R} \cup \{\pm\infty, -0, \mathrm{NaN}\}$ is described with β, t, e_{min}, e_{max} with elements

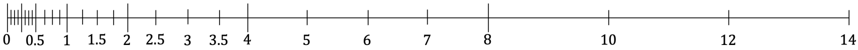
$$\pm m \times \beta^{e-t+1}.$$

Virtually all computers have $\beta = 2$ (binary FP).

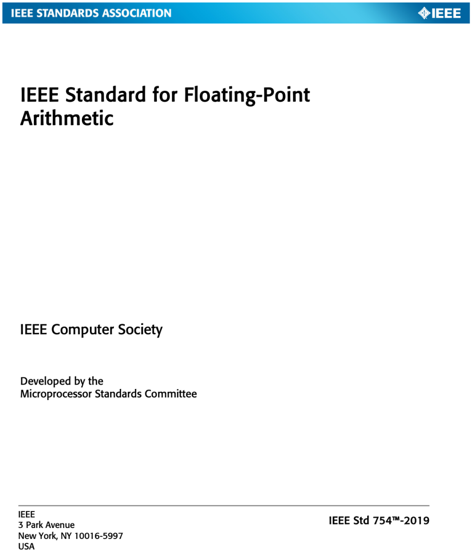
Here t is precision (binary digits), $e_{min} \leq e \leq e_{max}$ an exponent, $m \leq \beta^t - 1$ a significand (m, t, $e \in \mathbb{Z}$).

Toy FP system

Below: the positive numbers in $\mathbb{F}(\beta = 2, t = 3, e_{min} = -2, e_{max} = 3)$.



Part 1: IEEE 754 Standard for Floating-Point Arithmetic


IEEE STANDARDS ASSOCIATION IEEE

IEEE Standard for Floating-Point Arithmetic

IEEE Computer Society

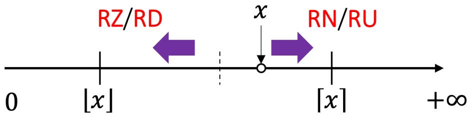
Developed by the
Microprocessor Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 754™-2019

IEEE 754 standard FP arithmetic: rounding

- Round-to-nearest (**RN**) (ties even)
- Round-toward-zero (**RZ**)
- Round-down (**RD**)
- Round-up (**RU**)

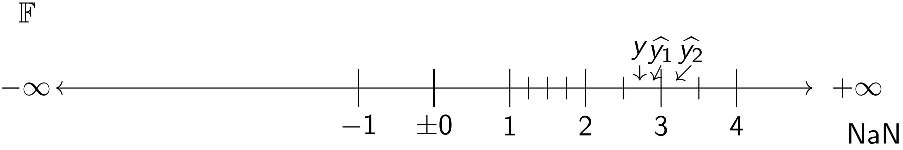


Use of rounding modes

RN is a default. *Directed modes* used for special cases, such as **interval arithmetic**.

Why a standard is needed?

- Rounding modes: result lies between two FP numbers—return one of them consistently.
- *Correct rounding* (directed): exact result y and approximation $\widehat{y}$ are surrounded by the same two FP numbers, consistently.
- Correct rounding (nearest): nearest FP number to y is also a nearest FP number to $\widehat{y}$.

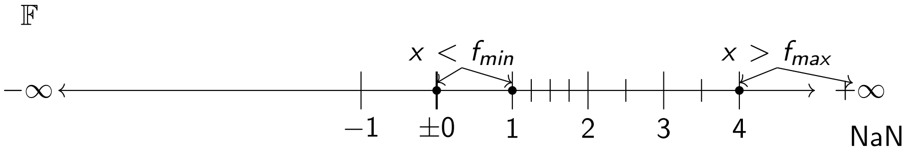


Accuracy is therefore not a consideration in IEEE 754—compliant operations are as accurate as precision allows.

Why a standard is needed? Overflows and underflows

The standard sets out behaviour for edge cases:

- Overflow is when rounding produces $\pm\infty$.
- Underflow occurs when the rounded result is between zero and smallest magnitude normal FP value.



Why a standard is needed? Operations

IEEE 754 requires *correctly rounded*

- $+$, $-$, $\times$, $\div$
- Square root
- FMA: $a \times b + c$

However, it does not require x^y and mathematical functions exp, sin, cos, log, and others.

Nevertheless, it recommends providing them, and they *shall be* correctly rounded—most mathematical function libraries do not assure this today.

Why a standard is needed? Invalid operations

A NaN is generated by

- $0/0$
- $0 \times \infty$
- ∞/∞
- $(+\infty) - (-\infty)$
- $\sqrt{-1}$

Once a NaN is generated, it propagates so the user sees it in the result, or causes an exception from the floating-point unit.

Why a standard is needed? Infinities

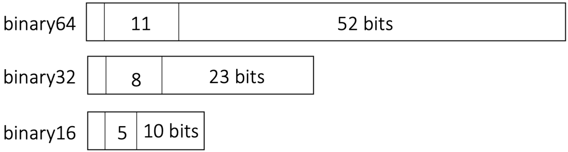
∞ obeys mathematical rules

- $\infty + \infty = \infty$
- $(-1) \times \infty = -\infty$
- $x/\infty = 0$ where x is a finite number

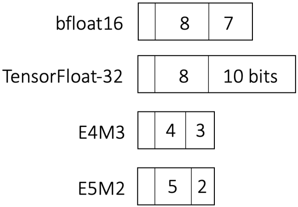
Floating-point format encoding

Numbers are held in memory using bits (convenient when $\beta = 2$).

Main IEEE 754 formats (**double**, **single**, **half**):



Some ubiquitous non-standard formats:



Floating point formats and their ranges

Format	precision	min pos. (f_{min})	max pos. (f_{max})	u
binary64	53	2^{-1022}	$\sim 1.798 \times 10^{308}$	2^{-53}
binary32	24	2^{-126}	$\sim 3.403 \times 10^{38}$	2^{-24}
tf32 (19-bit)	11	2^{-126}	$\sim 3.401 \times 10^{38}$	2^{-11}
bfloat16	8	2^{-126}	$\sim 3.389 \times 10^{38}$	2^{-8}
binary16	11	2^{-14}	65504	2^{-11}
fp8-E4M3	4	2^{-6}	448	2^{-4}
fp8-E5M2	3	2^{-14}	57344	2^{-3}
fp6-E2M3	4	2^{0}	7.5	2^{-4}
fp6-E3M2	3	2^{-2}	28	2^{-3}
fp4-E2M1	2	2^{0}	6	2^{-2}

Limitations of standardisation: ordering of operations

Some of the known properties of floating point:

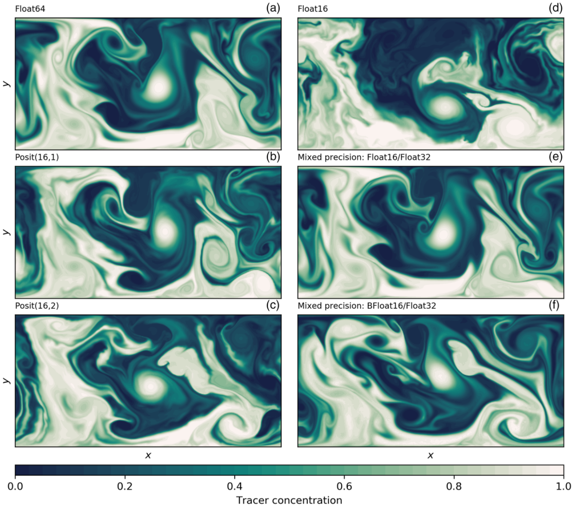
- ✓ Commutativity: $a \times b = b \times a$.
- × Associativity: $(a + (b + c)) \neq (a + b) + c)$.
- × Distributivity: $(a \times (b + c)) \neq (a \times b) + (a \times c)$.

Even though $+$ is correctly rounded, IEEE 754 does not mandate the order in $\sum_{i=0}^{n} x_i$—results implementation dependent.

But that is OK: ordering is in programmer's control, and two systems can match their implementations if needed.

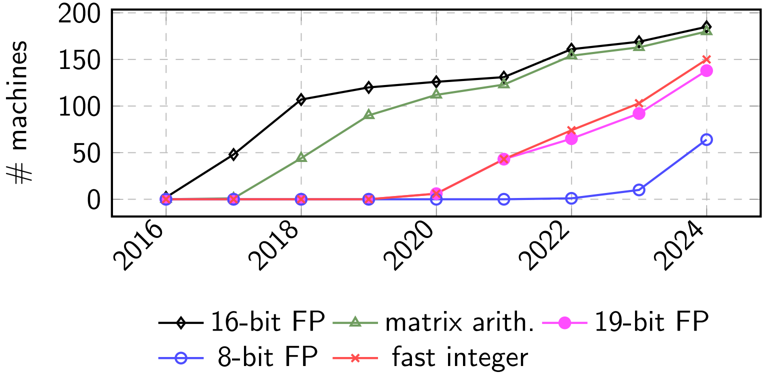
(except when race conditions and thread scheduling change ordering)

Part 2: Low-Precision Floating Point (made for AI, but people use it for scientific computation)



Shallow water model simulation [Klower et al. 2020].

Low-precision floating point on the TOP500



Devices counted: P100, V100, A100, H100, MI210, MI250X, MI300X, Intel Data Center GPU, from `https://www.top500.org`.

With NVIDIA Blackwell 4/6-bit FP will appear.

Mixed-precision matrix multipliers

Non-standard low-precision formats are available in matrix multiply form.

$$
\begin{matrix} D & = & C & + & A & \times & B, \end{matrix}
$$

$$
\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\substack{\text{binary16 or} \\ \text{binary32}}} = \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\substack{\text{binary16 or} \\ \text{binary32}}} + \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\text{8-bit FP}} \times \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\text{8-bit FP}}
$$

Example above is 4×4, but dimensions differ across architectures.

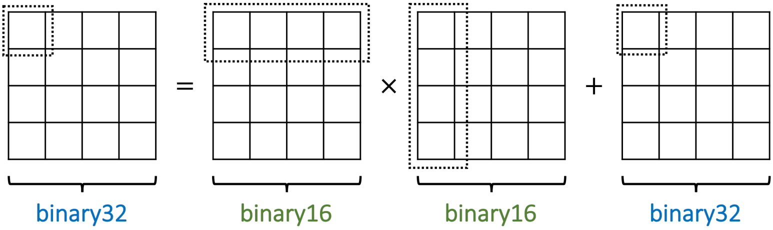
Mixed-precision matrix multipliers

Architecture	Input format	Accumulation format
NVIDIA PTX ISA	fp8-E5M2	binary32
	fp8-E4M3	binary32
	binary16	binary16
	binary16	binary32
	bfloat16	binary32
	19-bit FP	binary32
AMD MI300 ISA	fp8-E5M2	binary32
	fp8-E4M3	binary32
	binary16	binary32
	bfloat16	binary32
	19-bit FP	binary32

NVIDIA Blackwell throughputs (FLOPS)

fp8 (9×10^{15}) fp16 (4.5×10^{15}) fp64 (0.04×10^{15}).

Nonstandard features of matrix multipliers



$$a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} + a_{14} \times b_{41}$$

- We are used to **normalize-round after each op**.
- In hardware it is not necessarily the case.
- Normalize at the end? Savings in circuit area and latency.
- Round or drop bits? Savings.
- Accumulate in higher precision? Can do with 1-bit granularity.

Why test the mathematical hardware

IEEE 754 does not define strict rules on dot products:

"Implementations may associate in any order or evaluate in any wider format."

- Implementations might differ.
- Not documented in detail by vendors.
- Massive speed means we are using MMAs in other areas.
- No control over ordering means we may not be able to match two systems.

For now we test to determine features.

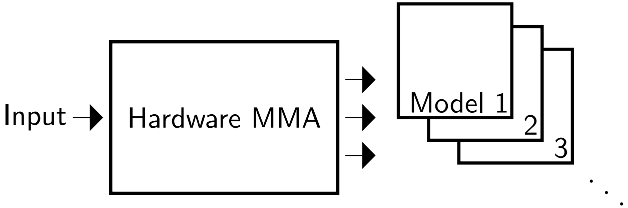
How do we test mathematical hardware

[Fasi, Higham, Mikaitis, Pranesh, 2021]

Technique goes back to software called `Paranoia` from the 1980s.

MMAs more complicated than $+, \times, \div$

Find floating-point **inputs that will yield different outputs on different hardware**.



Our main findings from testing

$$a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} + a_{14} \times b_{41}$$

1-bit difference in accumulators of NVIDIA V100 (2018) and T4/A100 (2020).

Normalization at the end, not at intermediate adds.

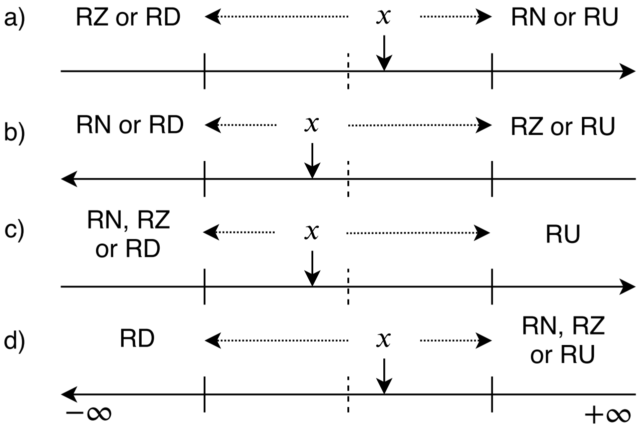
Rounding is not to nearest.

Nonmonotonicity: increase one input, do not change order, dot product decreases. See [Mikaitis, 2024].

No fixed order.

Currently a manual process

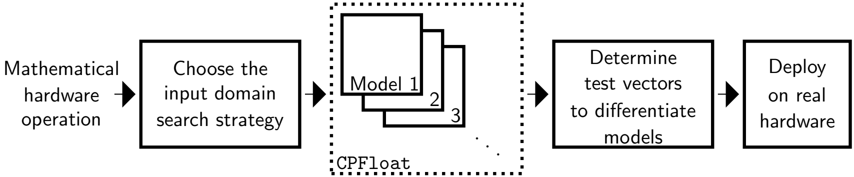
Consider rounding:



New, more systematic approach

3-year EPRSRC project: 2025-2028. Intel and Argonne National Lab as partners.

- Make the search for testing vectors automatic, or at least partially.
- Remove the need for very specialized floating-point knowledge.
- Make a library of behaviours and maintain as new hardware comes.



Simulating custom precision with CPFloat in MATLAB/Octave

Paper in ACM TOMS provides details. [Fasi and Mikaitis, 2023]

https://github.com/north-numerical-computing/cpfloat

Example use in MATLAB:

```
>> input.format = 'q43';
>> input.subnormal = 0;
>> accum.format = 'binary16';
>> accum.subnormal = 0;
>> cpfloat(pi, input)
ans =
   3.25000000000000
>> cpfloat(cpfloat(
    cpfloat(pi,input)*cpfloat(pi,input),accum)+0.5,accum)
ans =
  11.06250000000000
```

Part 3: Open Compute Project (OCP) standards

Open Compute Project floating-point standards

Two standards:

- Standard 1: OCP 8-bit Floating Point Specification
- Standard 2: OCP Microscaling Formats (MX) Specification

Version 1.0 released June and Septermber 2023.

OPEN Compute Project

OCP 8-bit Floating Point Specification (OFP8)

Revision 1.0

Date Submitted: May 26, 2023
Date Approved: June 20, 2023

Author: Paulius Micikevicius and Stuart Oberman, NVIDIA
Author: Pradeep Dubey, Marius Cornea and Andres Rodriguez, Intel
Author: Ian Bratt and Richard Grisenthwaite, Arm
Author: Norm Jouppi, Chiachen Chou and Amber Huffman, Google
Author: Michael Schulte and Ralph Wittig, AMD
Author: Dharmesh Jani and Summer Deng, Meta

OCP floating-point standard 1

Key aspects in the Standard 1

E4M3

$$\underbrace{\boxed{s}}_{1}\underbrace{\boxed{\quad E \quad}}_{4 \text{ bits}}\underbrace{\boxed{\quad M \quad}}_{3 \text{ bits}}$$

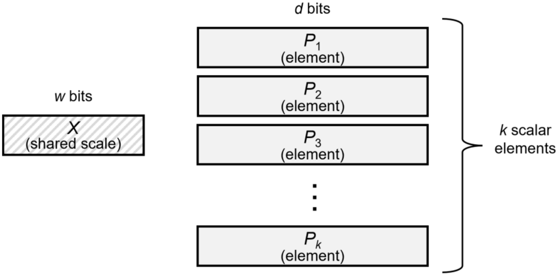
E5M2

$$\underbrace{\boxed{s}}_{1}\underbrace{\boxed{\quad E \quad}}_{5 \text{ bits}}\underbrace{\boxed{\; M \;}}_{2 \text{ bits}}$$

- Defines two formats: OFP8. E4M3 no $\pm\infty$, one NaN.
- Defines conversion from higher precision formats (binary32, binary16, bfloat16) to OFP8.
- Conversion includes
  - Round to nearest (no other rounding modes required)
  - Saturation mode: after rounding, if $> f_{max}$, return f_{max} instead of ∞.

Arithmetic operations are not in scope of standard 1.

OCP standard 2 (screenshot from the MX standard)



- Characterized by scale (X) type, data (P_i) type, and block size k
- Layout in memory not prescribed.
- $w + kd$ bits required. $w = 8$, $k = 32$ for all configurations.
- Value i encodes $X \times P_i$.

Benefits of shared scale: cheaply extend dynamic range (extra w/k bits per element).

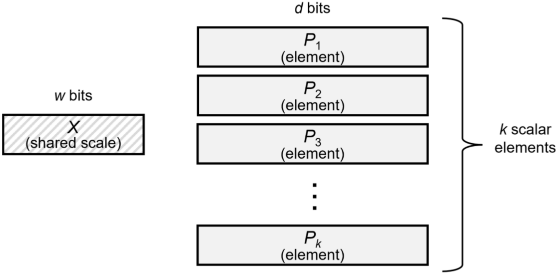
OCP standard 2 (screenshot from the MX standard)

Format Name	Element Data Type	Element Bits (d)	Scaling Block Size (k)	Scale Data Type	Scale Bits (w)
MXFP8	FP8 (E5M2)	8	32	E8M0	8
	FP8 (E4M3)				
MXFP6	FP6 (E3M2)	6	32	E8M0	8
	FP6 (E2M3)				
MXFP4	FP4 (E2M1)	4	32	E8M0	8
MXINT8	INT8	8	32	E8M0	8

INT8 is a fixed-point format encoding values $(-2, 2)$ in steps of 2^{-6}.

Conversion from standard vectors of size 32, to the MX formats, must be provided.

OCP standard 2 (screenshot from the MX standard)



Dot product of two MX vectors must be provided

$$C = Dot(A, B) = X^A X^B \sum_{i=1}^{32} (P_i^A \times P_i^B)$$

(*"internal precision of the dot product and order of operations is implementation-defined"*)

Part 4: IEEE P3109 Standard for Arithmetic Formats for Machine Learning (Oct. 2024 state)

IEEE floating-point standardisation work: IEEE P3109

Standard for Arithmetic Formats for Machine Learning

New IEEE standard for computer arithmetic for AI is in progress. We are also actively participating in it: meeting fortnightly.

Standardises:

- Small floating-point subsets of reals,
- encoding in 8-bit words,
- rounding behaviour,
- arithmetic operations needed in AI workloads,
- conversion to/from 754,
- exception handling.

Interim report available: `http://bit.ly/42gPWcy`

IEEE floating-point standardisation work: IEEE P3109

Current draft outlines these key aspects:

- Defines formats as binary$K$pP with $K = 8$ and $P = \{1, 2, 3, 4, 5, 6, 7\}$.
- No -0 (would have been 0x80)
- Only one NaN (0x80) - note IEEE 754 numbers have many bit patterns for NaNs.
- $\pm\infty$ 0x7F and 0xFF.
- Saturation mode: on overflow, return maximum finite value.
- Several rounding modes, operations.

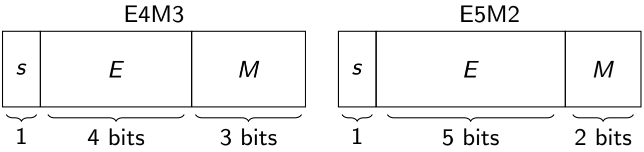
Format	minSubnormal	maxSubnormal	minNormal	maxNormal
binary8p1	N/A	N/A	1×2^{-62}	1×2^{63}
binary8p2	1×2^{-32}	1×2^{-32}	1×2^{-31}	1×2^{31}
binary8p3	1×2^{-17}	$3/2 \times 2^{-16}$	1×2^{-15}	$3/2 \times 2^{15}$
binary8p4	1×2^{-10}	$7/4 \times 2^{-8}$	1×2^{-7}	$7/4 \times 2^{7}$
binary8p5	1×2^{-7}	$15/8 \times 2^{-4}$	1×2^{-3}	$15/8 \times 2^{3}$
binary8p6	1×2^{-6}	$31/16 \times 2^{-2}$	1×2^{-1}	$31/16 \times 2^{1}$
binary8p7	1×2^{-6}	$63/32 \times 2^{-1}$	1×2^{0}	$63/32 \times 2^{0}$

IEEE P3109: 8-bit format on one slide

C.4 Value Table: P4, emin = −7, emax = 7

0x00 = 0_0000_000 = 0.0	0x40 = 0_1000_000 = +0b1.000×2^0 = 1.0	0x80 = 1_0000_000 = NaN	0xc0 = 1_1000_000 = −0b1.000×2^0 = -1.0
0x01 = 0_0000_001 = +0b0.001×2^-7 = 0.0009765625	0x41 = 0_1000_001 = +0b1.001×2^0 = 1.125	0x81 = 1_0000_001 = −0b0.001×2^-7 = -0.0009765625	0xc1 = 1_1000_001 = −0b1.001×2^0 = -1.125
0x02 = 0_0000_010 = +0b0.010×2^-7 = 0.001953125	0x42 = 0_1000_010 = +0b1.010×2^0 = 1.25	0x82 = 1_0000_010 = −0b0.010×2^-7 = -0.001953125	0xc2 = 1_1000_010 = −0b1.010×2^0 = -1.25
0x03 = 0_0000_011 = +0b0.011×2^-7 = 0.0029296875	0x43 = 0_1000_011 = +0b1.011×2^0 = 1.375	0x83 = 1_0000_011 = −0b0.011×2^-7 = -0.0029296875	0xc3 = 1_1000_011 = −0b1.011×2^0 = -1.375
0x04 = 0_0000_100 = +0b0.100×2^-7 = 0.00390625	0x44 = 0_1000_100 = +0b1.100×2^0 = 1.5	0x84 = 1_0000_100 = −0b0.100×2^-7 = -0.00390625	0xc4 = 1_1000_100 = −0b1.100×2^0 = -1.5
0x05 = 0_0000_101 = +0b0.101×2^-7 = 0.0048828125	0x45 = 0_1000_101 = +0b1.101×2^0 = 1.625	0x85 = 1_0000_101 = −0b0.101×2^-7 = -0.0048828125	0xc5 = 1_1000_101 = −0b1.101×2^0 = -1.625
0x06 = 0_0000_110 = +0b0.110×2^-7 = 0.005859375	0x46 = 0_1000_110 = +0b1.110×2^0 = 1.75	0x86 = 1_0000_110 = −0b0.110×2^-7 = -0.005859375	0xc6 = 1_1000_110 = −0b1.110×2^0 = -1.75
0x07 = 0_0000_111 = +0b0.111×2^-7 = 0.0068359375	0x47 = 0_1000_111 = +0b1.111×2^0 = 1.875	0x87 = 1_0000_111 = −0b0.111×2^-7 = -0.0068359375	0xc7 = 1_1000_111 = −0b1.111×2^0 = -1.875
0x08 = 0_0001_000 = +0b1.000×2^-7 = 0.0078125	0x48 = 0_1001_000 = +0b1.000×2^1 = 2.0	0x88 = 1_0001_000 = −0b1.000×2^-7 = -0.0078125	0xc8 = 1_1001_000 = −0b1.000×2^1 = -2.0
0x09 = 0_0001_001 = +0b1.001×2^-7 = 0.0087890625	0x49 = 0_1001_001 = +0b1.001×2^1 = 2.25	0x89 = 1_0001_001 = −0b1.001×2^-7 = -0.0087890625	0xc9 = 1_1001_001 = −0b1.001×2^1 = -2.25
0x0a = 0_0001_010 = +0b1.010×2^-7 = 0.009765625	0x4a = 0_1001_010 = +0b1.010×2^1 = 2.5	0x8a = 1_0001_010 = −0b1.010×2^-7 = -0.009765625	0xca = 1_1001_010 = −0b1.010×2^1 = -2.5
0x0b = 0_0001_011 = +0b1.011×2^-7 = 0.0107421875	0x4b = 0_1001_011 = +0b1.011×2^1 = 2.75	0x8b = 1_0001_011 = −0b1.011×2^-7 = -0.0107421875	0xcb = 1_1001_011 = −0b1.011×2^1 = -2.75
0x0c = 0_0001_100 = +0b1.100×2^-7 = 0.01171875	0x4c = 0_1001_100 = +0b1.100×2^1 = 3.0	0x8c = 1_0001_100 = −0b1.100×2^-7 = -0.01171875	0xcc = 1_1001_100 = −0b1.100×2^1 = -3.0
0x0d = 0_0001_101 = +0b1.101×2^-7 = 0.0126953125	0x4d = 0_1001_101 = +0b1.101×2^1 = 3.25	0x8d = 1_0001_101 = −0b1.101×2^-7 = -0.0126953125	0xcd = 1_1001_101 = −0b1.101×2^1 = -3.25
0x0e = 0_0001_110 = +0b1.110×2^-7 = 0.013671875	0x4e = 0_1001_110 = +0b1.110×2^1 = 3.5	0x8e = 1_0001_110 = −0b1.110×2^-7 = -0.013671875	0xce = 1_1001_110 = −0b1.110×2^1 = -3.5
0x0f = 0_0001_111 = +0b1.111×2^-7 = 0.0146484375	0x4f = 0_1001_111 = +0b1.111×2^1 = 3.75	0x8f = 1_0001_111 = −0b1.111×2^-7 = -0.0146484375	0xcf = 1_1001_111 = −0b1.111×2^1 = -3.75
0x10 = 0_0010_000 = +0b1.000×2^-6 = 0.015625	0x50 = 0_1010_000 = +0b1.000×2^2 = 4.0	0x90 = 1_0010_000 = −0b1.000×2^-6 = -0.015625	0xd0 = 1_1010_000 = −0b1.000×2^2 = -4.0
0x11 = 0_0010_001 = +0b1.001×2^-6 = 0.017578125	0x51 = 0_1010_001 = +0b1.001×2^2 = 4.5	0x91 = 1_0010_001 = −0b1.001×2^-6 = -0.017578125	0xd1 = 1_1010_001 = −0b1.001×2^2 = -4.5
0x12 = 0_0010_010 = +0b1.010×2^-6 = 0.01953125	0x52 = 0_1010_010 = +0b1.010×2^2 = 5.0	0x92 = 1_0010_010 = −0b1.010×2^-6 = -0.01953125	0xd2 = 1_1010_010 = −0b1.010×2^2 = -5.0
0x13 = 0_0010_011 = +0b1.011×2^-6 = 0.021484375	0x53 = 0_1010_011 = +0b1.011×2^2 = 5.5	0x93 = 1_0010_011 = −0b1.011×2^-6 = -0.021484375	0xd3 = 1_1010_011 = −0b1.011×2^2 = -5.5
0x14 = 0_0010_100 = +0b1.100×2^-6 = 0.0234375	0x54 = 0_1010_100 = +0b1.100×2^2 = 6.0	0x94 = 1_0010_100 = −0b1.100×2^-6 = -0.0234375	0xd4 = 1_1010_100 = −0b1.100×2^2 = -6.0
0x15 = 0_0010_101 = +0b1.101×2^-6 = 0.025390625	0x55 = 0_1010_101 = +0b1.101×2^2 = 6.5	0x95 = 1_0010_101 = −0b1.101×2^-6 = -0.025390625	0xd5 = 1_1010_101 = −0b1.101×2^2 = -6.5
0x16 = 0_0010_110 = +0b1.110×2^-6 = 0.02734375	0x56 = 0_1010_110 = +0b1.110×2^2 = 7.0	0x96 = 1_0010_110 = −0b1.110×2^-6 = -0.02734375	0xd6 = 1_1010_110 = −0b1.110×2^2 = -7.0
0x17 = 0_0010_111 = +0b1.111×2^-6 = 0.029296875	0x57 = 0_1010_111 = +0b1.111×2^2 = 7.5	0x97 = 1_0010_111 = −0b1.111×2^-6 = -0.029296875	0xd7 = 1_1010_111 = −0b1.111×2^2 = -7.5
0x18 = 0_0011_000 = +0b1.000×2^-5 = 0.03125	0x58 = 0_1011_000 = +0b1.000×2^3 = 8.0	0x98 = 1_0011_000 = −0b1.000×2^-5 = -0.03125	0xd8 = 1_1011_000 = −0b1.000×2^3 = -8.0
0x19 = 0_0011_001 = +0b1.001×2^-5 = 0.03515625	0x59 = 0_1011_001 = +0b1.001×2^3 = 9.0	0x99 = 1_0011_001 = −0b1.001×2^-5 = -0.03515625	0xd9 = 1_1011_001 = −0b1.001×2^3 = -9.0
0x1a = 0_0011_010 = +0b1.010×2^-5 = 0.0390625	0x5a = 0_1011_010 = +0b1.010×2^3 = 10.0	0x9a = 1_0011_010 = −0b1.010×2^-5 = -0.0390625	0xda = 1_1011_010 = −0b1.010×2^3 = -10.0
0x1b = 0_0011_011 = +0b1.011×2^-5 = 0.04296875	0x5b = 0_1011_011 = +0b1.011×2^3 = 11.0	0x9b = 1_0011_011 = −0b1.011×2^-5 = -0.04296875	0xdb = 1_1011_011 = −0b1.011×2^3 = -11.0
0x1c = 0_0011_100 = +0b1.100×2^-5 = 0.046875	0x5c = 0_1011_100 = +0b1.100×2^3 = 12.0	0x9c = 1_0011_100 = −0b1.100×2^-5 = -0.046875	0xdc = 1_1011_100 = −0b1.100×2^3 = -12.0
0x1d = 0_0011_101 = +0b1.101×2^-5 = 0.05078125	0x5d = 0_1011_101 = +0b1.101×2^3 = 13.0	0x9d = 1_0011_101 = −0b1.101×2^-5 = -0.05078125	0xdd = 1_1011_101 = −0b1.101×2^3 = -13.0
0x1e = 0_0011_110 = +0b1.110×2^-5 = 0.0546875	0x5e = 0_1011_110 = +0b1.110×2^3 = 14.0	0x9e = 1_0011_110 = −0b1.110×2^-5 = -0.0546875	0xde = 1_1011_110 = −0b1.110×2^3 = -14.0
0x1f = 0_0011_111 = +0b1.111×2^-5 = 0.05859375	0x5f = 0_1011_111 = +0b1.111×2^3 = 15.0	0x9f = 1_0011_111 = −0b1.111×2^-5 = -0.05859375	0xdf = 1_1011_111 = −0b1.111×2^3 = -15.0
0x20 = 0_0100_000 = +0b1.000×2^-4 = 0.0625	0x60 = 0_1100_000 = +0b1.000×2^4 = 16.0	0xa0 = 1_0100_000 = −0b1.000×2^-4 = -0.0625	0xe0 = 1_1100_000 = −0b1.000×2^4 = -16.0
0x21 = 0_0100_001 = +0b1.001×2^-4 = 0.0703125	0x61 = 0_1100_001 = +0b1.001×2^4 = 18.0	0xa1 = 1_0100_001 = −0b1.001×2^-4 = -0.0703125	0xe1 = 1_1100_001 = −0b1.001×2^4 = -18.0
0x22 = 0_0100_010 = +0b1.010×2^-4 = 0.078125	0x62 = 0_1100_010 = +0b1.010×2^4 = 20.0	0xa2 = 1_0100_010 = −0b1.010×2^-4 = -0.078125	0xe2 = 1_1100_010 = −0b1.010×2^4 = -20.0
0x23 = 0_0100_011 = +0b1.011×2^-4 = 0.0859375	0x63 = 0_1100_011 = +0b1.011×2^4 = 22.0	0xa3 = 1_0100_011 = −0b1.011×2^-4 = -0.0859375	0xe3 = 1_1100_011 = −0b1.011×2^4 = -22.0
0x24 = 0_0100_100 = +0b1.100×2^-4 = 0.09375	0x64 = 0_1100_100 = +0b1.100×2^4 = 24.0	0xa4 = 1_0100_100 = −0b1.100×2^-4 = -0.09375	0xe4 = 1_1100_100 = −0b1.100×2^4 = -24.0
0x25 = 0_0100_101 = +0b1.101×2^-4 = 0.1015625	0x65 = 0_1100_101 = +0b1.101×2^4 = 26.0	0xa5 = 1_0100_101 = −0b1.101×2^-4 = -0.1015625	0xe5 = 1_1100_101 = −0b1.101×2^4 = -26.0
0x26 = 0_0100_110 = +0b1.110×2^-4 = 0.109375	0x66 = 0_1100_110 = +0b1.110×2^4 = 28.0	0xa6 = 1_0100_110 = −0b1.110×2^-4 = -0.109375	0xe6 = 1_1100_110 = −0b1.110×2^4 = -28.0
0x27 = 0_0100_111 = +0b1.111×2^-4 = 0.1171875	0x67 = 0_1100_111 = +0b1.111×2^4 = 30.0	0xa7 = 1_0100_111 = −0b1.111×2^-4 = -0.1171875	0xe7 = 1_1100_111 = −0b1.111×2^4 = -30.0
0x28 = 0_0101_000 = +0b1.000×2^-3 = 0.125	0x68 = 0_1101_000 = +0b1.000×2^5 = 32.0	0xa8 = 1_0101_000 = −0b1.000×2^-3 = -0.125	0xe8 = 1_1101_000 = −0b1.000×2^5 = -32.0
0x29 = 0_0101_001 = +0b1.001×2^-3 = 0.140625	0x69 = 0_1101_001 = +0b1.001×2^5 = 36.0	0xa9 = 1_0101_001 = −0b1.001×2^-3 = -0.140625	0xe9 = 1_1101_001 = −0b1.001×2^5 = -36.0
0x2a = 0_0101_010 = +0b1.010×2^-3 = 0.15625	0x6a = 0_1101_010 = +0b1.010×2^5 = 40.0	0xaa = 1_0101_010 = −0b1.010×2^-3 = -0.15625	0xea = 1_1101_010 = −0b1.010×2^5 = -40.0
0x2b = 0_0101_011 = +0b1.011×2^-3 = 0.171875	0x6b = 0_1101_011 = +0b1.011×2^5 = 44.0	0xab = 1_0101_011 = −0b1.011×2^-3 = -0.171875	0xeb = 1_1101_011 = −0b1.011×2^5 = -44.0
0x2c = 0_0101_100 = +0b1.100×2^-3 = 0.1875	0x6c = 0_1101_100 = +0b1.100×2^5 = 48.0	0xac = 1_0101_100 = −0b1.100×2^-3 = -0.1875	0xec = 1_1101_100 = −0b1.100×2^5 = -48.0
0x2d = 0_0101_101 = +0b1.101×2^-3 = 0.203125	0x6d = 0_1101_101 = +0b1.101×2^5 = 52.0	0xad = 1_0101_101 = −0b1.101×2^-3 = -0.203125	0xed = 1_1101_101 = −0b1.101×2^5 = -52.0
0x2e = 0_0101_110 = +0b1.110×2^-3 = 0.21875	0x6e = 0_1101_110 = +0b1.110×2^5 = 56.0	0xae = 1_0101_110 = −0b1.110×2^-3 = -0.21875	0xee = 1_1101_110 = −0b1.110×2^5 = -56.0
0x2f = 0_0101_111 = +0b1.111×2^-3 = 0.234375	0x6f = 0_1101_111 = +0b1.111×2^5 = 60.0	0xaf = 1_0101_111 = −0b1.111×2^-3 = -0.234375	0xef = 1_1101_111 = −0b1.111×2^5 = -60.0
0x30 = 0_0110_000 = +0b1.000×2^-2 = 0.25	0x70 = 0_1110_000 = +0b1.000×2^6 = 64.0	0xb0 = 1_0110_000 = −0b1.000×2^-2 = -0.25	0xf0 = 1_1110_000 = −0b1.000×2^6 = -64.0
0x31 = 0_0110_001 = +0b1.001×2^-2 = 0.28125	0x71 = 0_1110_001 = +0b1.001×2^6 = 72.0	0xb1 = 1_0110_001 = −0b1.001×2^-2 = -0.28125	0xf1 = 1_1110_001 = −0b1.001×2^6 = -72.0
0x32 = 0_0110_010 = +0b1.010×2^-2 = 0.3125	0x72 = 0_1110_010 = +0b1.010×2^6 = 80.0	0xb2 = 1_0110_010 = −0b1.010×2^-2 = -0.3125	0xf2 = 1_1110_010 = −0b1.010×2^6 = -80.0
0x33 = 0_0110_011 = +0b1.011×2^-2 = 0.34375	0x73 = 0_1110_011 = +0b1.011×2^6 = 88.0	0xb3 = 1_0110_011 = −0b1.011×2^-2 = -0.34375	0xf3 = 1_1110_011 = −0b1.011×2^6 = -88.0
0x34 = 0_0110_100 = +0b1.100×2^-2 = 0.375	0x74 = 0_1110_100 = +0b1.100×2^6 = 96.0	0xb4 = 1_0110_100 = −0b1.100×2^-2 = -0.375	0xf4 = 1_1110_100 = −0b1.100×2^6 = -96.0
0x35 = 0_0110_101 = +0b1.101×2^-2 = 0.40625	0x75 = 0_1110_101 = +0b1.101×2^6 = 104.0	0xb5 = 1_0110_101 = −0b1.101×2^-2 = -0.40625	0xf5 = 1_1110_101 = −0b1.101×2^6 = -104.0
0x36 = 0_0110_110 = +0b1.110×2^-2 = 0.4375	0x76 = 0_1110_110 = +0b1.110×2^6 = 112.0	0xb6 = 1_0110_110 = −0b1.110×2^-2 = -0.4375	0xf6 = 1_1110_110 = −0b1.110×2^6 = -112.0
0x37 = 0_0110_111 = +0b1.111×2^-2 = 0.46875	0x77 = 0_1110_111 = +0b1.111×2^6 = 120.0	0xb7 = 1_0110_111 = −0b1.111×2^-2 = -0.46875	0xf7 = 1_1110_111 = −0b1.111×2^6 = -120.0
0x38 = 0_0111_000 = +0b1.000×2^-1 = 0.5	0x78 = 0_1111_000 = +0b1.000×2^7 = 128.0	0xb8 = 1_0111_000 = −0b1.000×2^-1 = -0.5	0xf8 = 1_1111_000 = −0b1.000×2^7 = -128.0
0x39 = 0_0111_001 = +0b1.001×2^-1 = 0.5625	0x79 = 0_1111_001 = +0b1.001×2^7 = 144.0	0xb9 = 1_0111_001 = −0b1.001×2^-1 = -0.5625	0xf9 = 1_1111_001 = −0b1.001×2^7 = -144.0
0x3a = 0_0111_010 = +0b1.010×2^-1 = 0.625	0x7a = 0_1111_010 = +0b1.010×2^7 = 160.0	0xba = 1_0111_010 = −0b1.010×2^-1 = -0.625	0xfa = 1_1111_010 = −0b1.010×2^7 = -160.0
0x3b = 0_0111_011 = +0b1.011×2^-1 = 0.6875	0x7b = 0_1111_011 = +0b1.011×2^7 = 176.0	0xbb = 1_0111_011 = −0b1.011×2^-1 = -0.6875	0xfb = 1_1111_011 = −0b1.011×2^7 = -176.0
0x3c = 0_0111_100 = +0b1.100×2^-1 = 0.75	0x7c = 0_1111_100 = +0b1.100×2^7 = 192.0	0xbc = 1_0111_100 = −0b1.100×2^-1 = -0.75	0xfc = 1_1111_100 = −0b1.100×2^7 = -192.0
0x3d = 0_0111_101 = +0b1.101×2^-1 = 0.8125	0x7d = 0_1111_101 = +0b1.101×2^7 = 208.0	0xbd = 1_0111_101 = −0b1.101×2^-1 = -0.8125	0xfd = 1_1111_101 = −0b1.101×2^7 = -208.0
0x3e = 0_0111_110 = +0b1.110×2^-1 = 0.875	0x7e = 0_1111_110 = +0b1.110×2^7 = 224.0	0xbe = 1_0111_110 = −0b1.110×2^-1 = -0.875	0xfe = 1_1111_110 = −0b1.110×2^7 = -224.0
0x3f = 0_0111_111 = +0b1.111×2^-1 = 0.9375	0x7f = 0_1111_111 = +Inf	0xbf = 1_0111_111 = −0b1.111×2^-1 = -0.9375	0xff = 1_1111_111 = −Inf

Standard 8-bit floating-point formats: summary



- OCP standard (`https://bit.ly/3G9EsyG`). E5M2 similar to IEEE 754 formats. E4M3: no infinities, one signed NaN.
- IEEE p3109 (interim report `https://bit.ly/42gPWcy`). No -0. One unsigned NaN. More rigorous and complete.
- NVIDIA (PTX ISA 8.7 `https://bit.ly/3RNElve`) follows the OCP spec.
- AMD does not fully follow the OCP spec. for NaN and inf.
- **Standardisation work ongoing (P3109 is changing significantly).**

Summary

- Standardisation work ongoing until approx 2029.
- We are developing tools and methodologies to probe hardware.
- We maintain a base of hardware knowledge, which we input into standardisation committees.

Key takeaway

OCP is different from IEEE P3109 in terms of defining subsets of reals (formats). For the foreseeable future, two 8-bit standards will be driving the progress in hardware.

Paper that started this work

M. Fasi, N. J. Higham, M. Mikaitis, S. Pranesh. *Numerical Behavior of NVIDIA Tensor Cores*. **Peer J. Comput. Sci., 7**. Feb. 2021. `https://bit.ly/442IIGT`.

References I

M. Klöwer , P. D. Düben , and T. N. Palmer
Number formats, error mitigation, and scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow water model
J. Adv. Model. Earth Systems, 12. 2020.

M. Fasi, N. J. Higham, S. Pranesh, and M. Mikaitis
Numerical behavior of NVIDIA tensor cores
PeerJ Comput. Sci., 7. 2021

M. Mikaitis
Monotonicity of Multi-Term Floating-Point Adders
IEEE Trans. Comput., 73. 2024.

M. Fasi and M. Mikaitis
CPFloat: A C library for Simulating Low-Precision Arithmetic
ACM Trans. Math. Software, 49. 2023