Solved Example on Floating-Point/Int/Fixed-Point Number

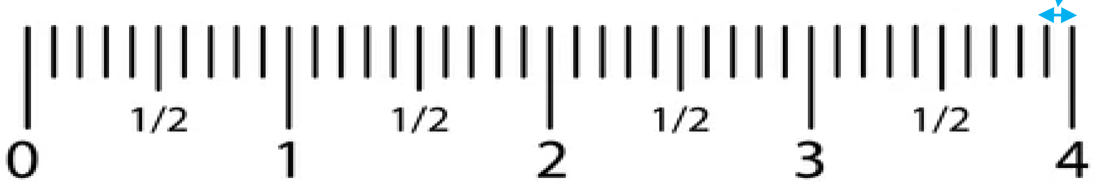
Sparsh Mittal

FP – Floating point
FxP – Fixed-point
Int – Integer

Range vs precision

Comparison of Weighing machines

Precision



Range

High precision
Low range

Low precision
High range

Comparing Int, FP, and FxP (All Unsigned)

- They have same number of total bits

Example 1

- Int: 4 integer bits
- FxP: 2 int and 2 fractional bits

2	2

- FP: 2 exp and 2 mantissa bits

2	2

Now, let's generate all the combinations

Comparing Int, FP, and FxP

string	int	FxP	FP
0000	0	0	0
0001	1	0.25	Denormal number 0.25
0010	2	0.5	Denormal number 0.5
0011	3	0.75	Denormal number 0.75
0100	4	1	1
0101	5	1.25	1.25
0110	6	1.5	1.5
0111	7	1.75	1.75
1000	8	2	2
1001	9	2.25	2.5
1010	10	2.5	3
1011	11	2.75	3.5
1100	12	3	+infinity
1101	13	3.25	NAN
1110	14	3.5	NAN
1111	15	3.75	NAN

Here, difference between consecutive numbers = 0.25

Here, difference between consecutive numbers = 0.5

Range = 1 to 3.5

NaN – Not a number

Observation

- Given a FxP number, we can multiply it by 4 to get corresponding integer number.
- We multiplied by 4 because FxP had 2 bits after decimal.

➔ Int and FxP are related.

- But, there is no such correlation between FxP and FP value of a binary representation.

Example 1: Let's Do the Math for FP

$$x = (1 + \text{Fraction}) \times 2^{(\text{Exponent}-\text{Bias})}$$

- Find bias.
- Bias = 2^(NumberOfExpBits-1)-1 = 1
- Find least normal number
  - Fraction= 00, Exponent=01, so actual exponent = 1-bias = 0
  - Number = 1 * 2^0 = 1
- Find largest number
  - Fraction = 11, Exponent = 10, so actual exponent = 2-bias = 1
  - Number = 1.11_2 * 2^1 = 1.75*2 = 3.5
- Which combination is infinity
  - Exponent is all 1, fraction is all-zero, which means 1100

Example 1: Let's Do the Math for FP

- Which combination is zero?
  - 0000
- Which combination is NaN
  - Fraction is not all-zero, exponent is all 1
  - Three combinations: 1101, 1110, 1111
- Denormal number (exp bits =0, fraction not all-zero)

$$x = (0 + \text{Fraction}) \times 2^{e}$$

  - Here, e = 1-bias =0
  - Three combinations: 0001, 0010, 0011

Let's Translate One Number 1001 to FP

- The number in binary is 1001, so Exponent (or ExcessExponent) is 10 and mantissa (fraction) is 01.

  ActualExponent = ExcessExponent-Bias = 10-01 = 1 (in base2 format)
  Thus, actual exponent is 1

  Fraction = .01(base2) = 0.25 (base10)

  Thus, overall number
  1.25*2 = 2.5 (decimal)

Comparing Int, FP, and FxP (All Unsigned)

- They have same number of total bits

Example 2

- Int: 4 integer bits
- FxP: 2 int and 2 fractional bits

2	2

- FP: 3 exp and 1 mantissa bits

3	1

Now, let's generate all the combinations

Example 2

Bias = 3

Range = 0.25 to 12

string	integer	fixed point	FP
0000	0	0	0
0001	1	0.25	Denormal 0.125
0010	2	0.5	0.25
0011	3	0.75	0.375
0100	4	1	0.5
0101	5	1.25	0.75
0110	6	1.5	1
0111	7	1.75	1.5
1000	8	2	2
1001	9	2.25	3
1010	10	2.5	4
1011	11	2.75	6
1100	12	3	8
1101	13	3.25	12
1110	14	3.5	+infinity
1111	15	3.75	NAN

Here, difference between consecutive numbers = 0.125

Here, difference between consecutive numbers = 0.25

Difference = 0.5

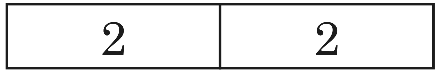
Difference = 4

Comparing Int, FP, and FxP (All Unsigned)

- They have same number of total bits

Example 3

- Int: 4 integer bits
- FxP: 2 int and 2 fractional bits



- FP: 4 exp and 0 mantissa bits

Now, let's generate all the combinations

Example 3

Range = 0.015625 to 128

String	integer	fixed point	FP
0000	0	0	0
0001	1	0.25	0.015625
0010	2	0.5	0.03125
0011	3	0.75	0.0625
0100	4	1	0.125
0101	5	1.25	0.25
0110	6	1.5	0.5
0111	7	1.75	1
1000	8	2	2
1001	9	2.25	4
1010	10	2.5	8
1011	11	2.75	16
1100	12	3	32
1101	13	3.25	64
1110	14	3.5	128
1111	15	3.75	+infinity

Here, difference between consecutive numbers = 0.015625

Here, difference between consecutive numbers = 0.25

Difference = 1

Difference = 64

There is no mantissa bit, so we can represent only power-of-two numbers

Comparing Int, FP, and FxP (All Unsigned)

- They have same number of total bits

Example 4

- Int: 4 integer bits
- FxP: 2 int and 2 fractional bits

2	2

- FP: 1 exp and 3 mantissa bits

1	3

Now, let's generate all the combinations

Example 4

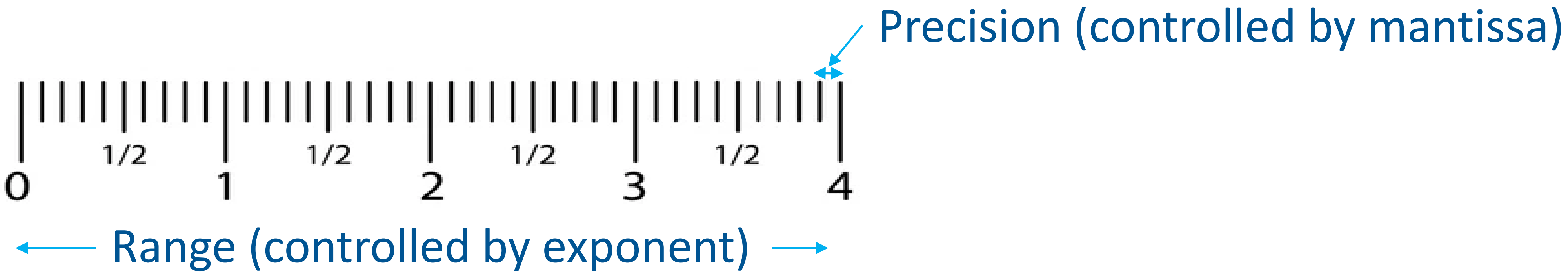
Comb.	Integer	Fixed Point	FP
0000	0	0	0
0001	1	0.25	Denormal number 0.25
0010	2	0.5	Denormal number 0.5
0011	3	0.75	Denormal number 0.75
0100	4	1	Denormal number 1
0101	5	1.25	Denormal number 1.25
0110	6	1.5	Denormal number 1.5
0111	7	1.75	Denormal number 1.75
1000	8	2	+infinity
1001	9	2.25	NAN
1010	10	2.5	NAN
1011	11	2.75	NAN
1100	12	3	NAN
1101	13	3.25	NAN
1110	14	3.5	NAN
1111	15	3.75	NAN

This number format is of not much use (too many NaNs)

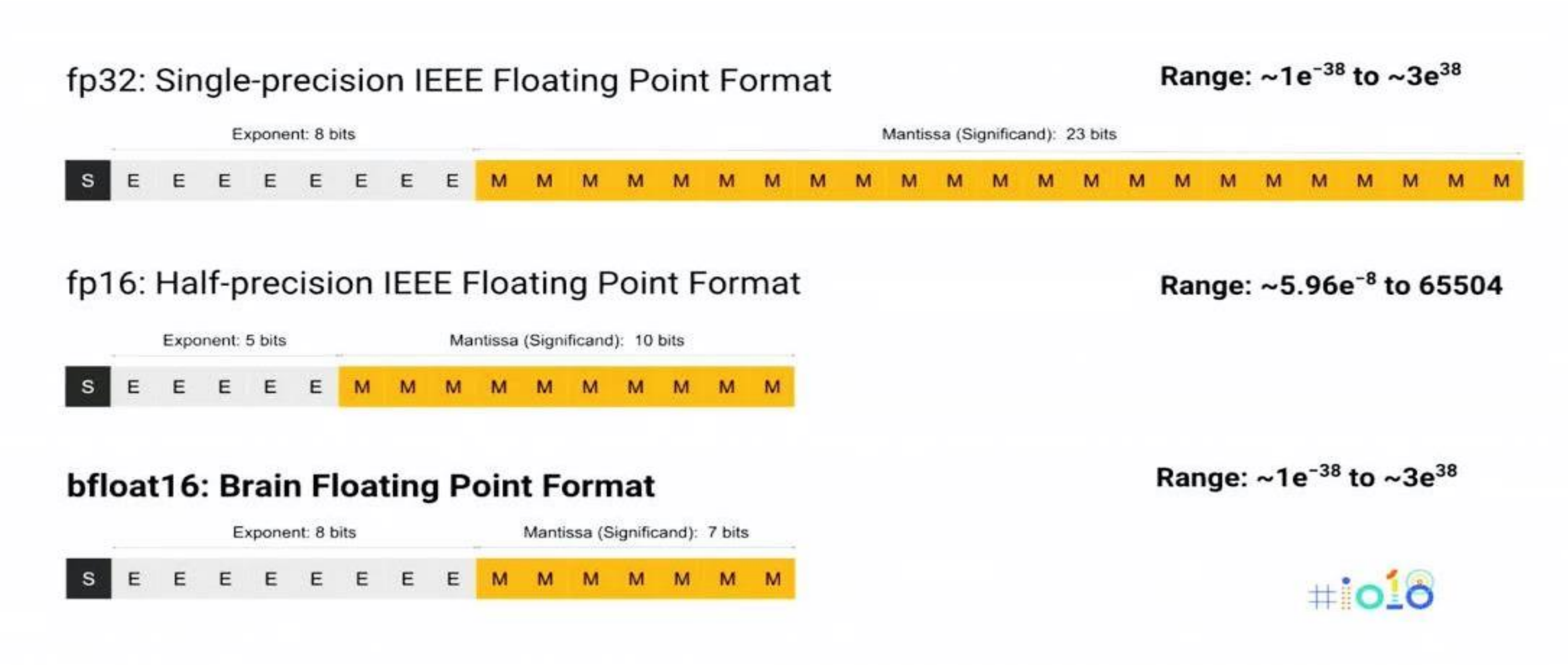
Need to have a minimum number of exponent bits

Insights

- Exponent controls the range.
- Mantissa decides the precision.
- Needs to balance them. For a fixed total bit-width
    - ➢Having too many mantissa bits will lead to overflow due to small range.
    - ➢Having too many exponent bits will lead to approximate many values to nearby representable value.



FP16 vs Bfloat16



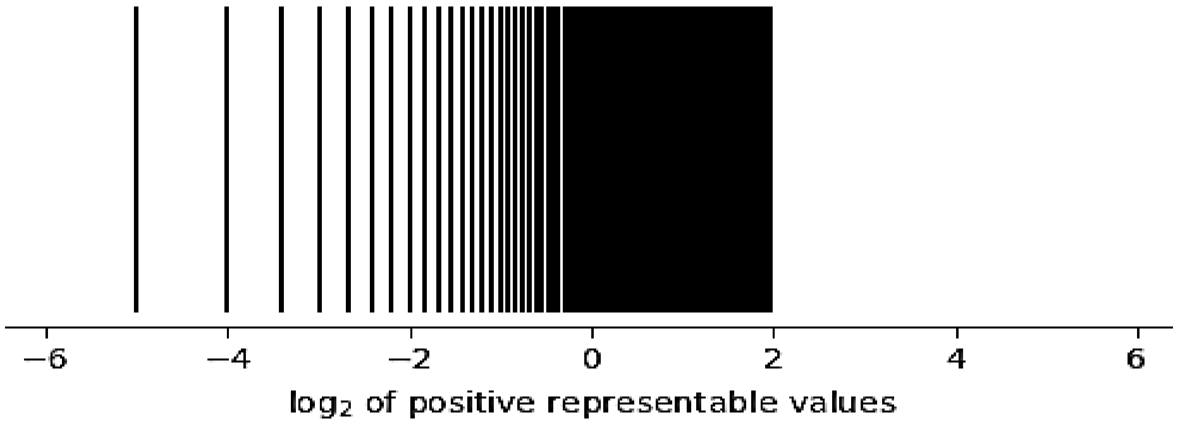
FP16 has only 5 exponent bits, hence, it has very low range

BFloat16 has 8 exponent bits, same as FP32. Hence, it can be used as a replacement of FP32 (although it has low precision)

Gaps between numbers in FP vs FxP

- FxP: gaps between adjacent numbers is fixed
- FP: gaps are not uniformly spaced. Large gaps between large numbers and small gaps between small numbers

Illustration of 8b FP format on number line



Source: https://www.johndcook.com/blog/2018/04/15/eight-bit-floating-point/

Range of Integer, FP, and fixed-point

- A signed 32-bit integer variable has a maximum value of $2^{31} - 1 = 2{,}147{,}483{,}647$,
- An IEEE 754 32-bit base-2 floating-point variable has a maximum value of $(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$.
- A floating-point variable can represent a wider range of numbers than a fixed point (or integer) variable of the same bit width at the cost of precision.

Relative costs

Bit-width	Operation	Energy	Relative costs
32-bit	Floating-point **ADD**	0.9pJ	30
	Floating-point **MUL**	3.7pJ	123.33
	Fixed-point **ADD**	0.1pJ	3.33
	Fixed-point **MUL**	3.1pJ	103.33
	DRAM access (Average)	0.65~1.3nJ	21667~43333
16-bit	Floating-point **ADD**	0.4pJ	13.33
	Floating-point **MUL**	1.1pJ	36.67
	*Fixed-point **ADD**	0.05pJ	1.67
	*Fixed-point **MUL**	1.55pJ	51.67
	DRAM access (Average)	0.33~0.65nJ	10000~21667
8-bit	Fixed-point **ADD**	0.03pJ	1
	Fixed-point **MUL**	0.2pJ	6.67
	DRAM access (Average)	0.16~0.33nJ	5333~10000

Zhao et al., Cambricon-Q: A Hybrid Architecture for Efficient Training, ISCA 2021