Representing and Manipulating Information: Floating point denormalized numbers and mastery

Yipeng Huang

Rutgers University

February 29, 2024

Table of contents

Programming assignment 3

Programming assignment 3

1. Due Friday 3/8.
2. Get started early! Plenty of background already for Parts 1 through 3.

Table of contents

Floating point numbers

Avogadro's number

$+6.02214 \times 10^{23}\, mol^{-1}$

Scientific notation

- sign
- mantissa or significand
- exponent

Floating point numbers

Before 1985

1. Many floating point systems.
2. Specialized machines such as Cray supercomputers.
3. Some machines with specialized floating point have had to be kept alive to support legacy software.

After 1985

1. IEEE Standard 754.
2. A floating point standard designed for good numerical properties.
3. Found in almost every computer today, except for tiniest microcontrollers.

Recent

1. Need for both lower precision and higher range floating point numbers.
2. Machine learning / neural networks. Low-precision tensor network processors.

Floats and doubles

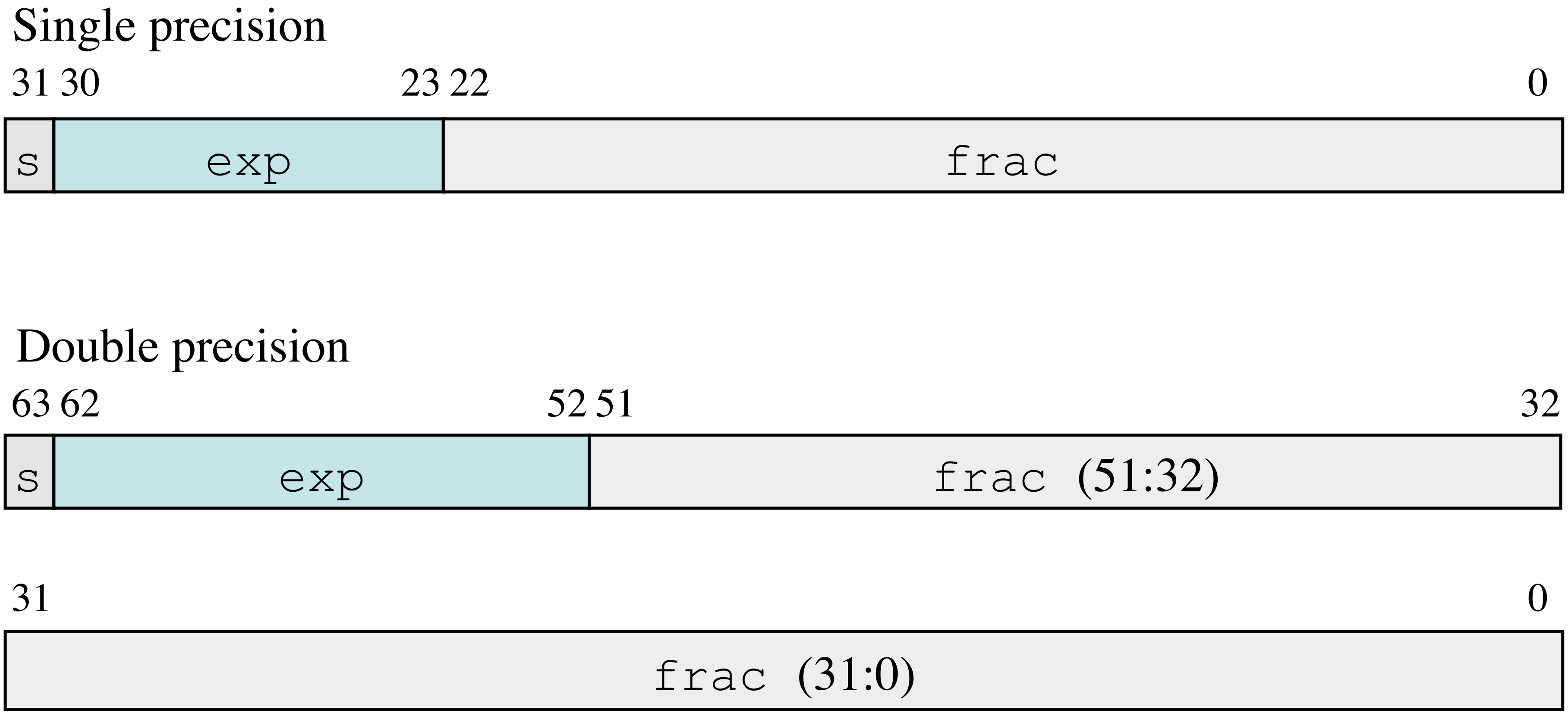


Figure: The two standard formats for floating point data types. Image credit CS:APP

Floats and doubles

property	half*	float	double
total bits	16	32	64
s bit	1	1	1
exp bits	5	8	11
frac bits	10	23	52
C printf() format specifier	None	"%f"	"%lf"

Table: Properties of floats and doubles

The IEEE 754 number line

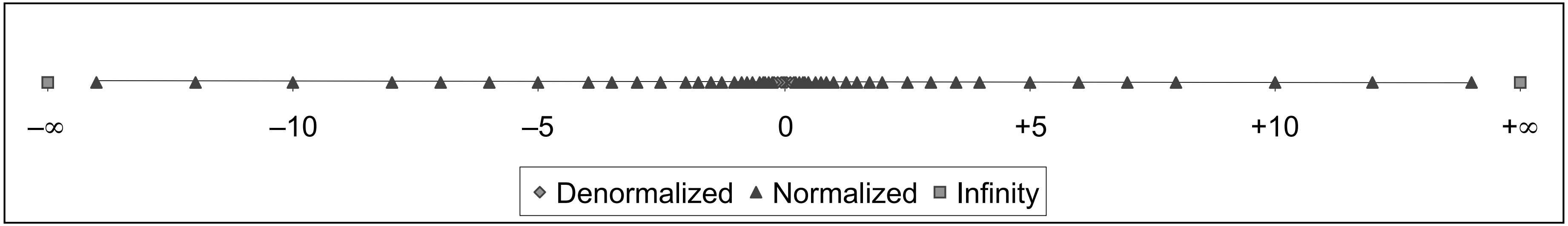


Figure: Full picture of number line for floating point values. Image credit CS:APP

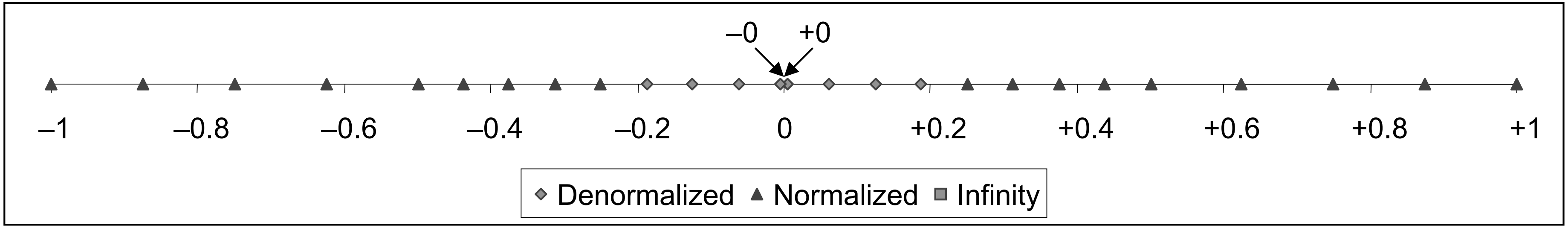


Figure: Zoomed in number line for floating point values. Image credit CS:APP

Different cases for floating point numbers

Value of the floating point number = $(-1)^s \times M \times 2^E$

- E is encoded the exp field
- M is encoded the frac field

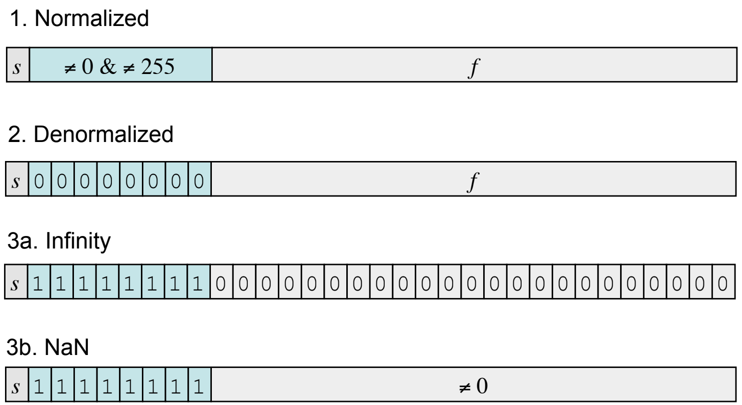


Figure: Different cases within a floating point format. Image credit CS:APP

Normalized and denormalized numbers

Two different cases we need to consider for the encoding of E, M

Table of contents

Normalized: exp field

For normalized numbers,
$0 < \text{exp} < 2^k - 1$

- exp is a *k*-bit unsigned integer

Bias

- need a bias to represent negative exponents
- bias = $2^{k-1} - 1$
- bias is the *k*-bit unsigned integer: 011..111

For normalized numbers,
E = exp-bias

In other words, exp = E+bias

property	float	double
k	8	11
bias	127	1023
smallest E (greatest precision)	-126	-1022
largest E (greatest range)	127	1023

Table: Summary of normalized exp field

Normalized: frac field

M = 1.frac

Normalized: example

- 12.375 to single-precision floating point
- sign is positive so s=0
- binary is 1100.011_2
- in other words it is $1.100011_2 \times 2^3$
- $\text{exp} = E + \text{bias} = 3 + 127 = 130 = 1000\_0010_2$
- M = 1.100011_2 = 1.frac
- frac = 100011

Table of contents

6.02×10^{23} } 10^{21}

6.03×10^{23}

0.

The IEEE 754 number line

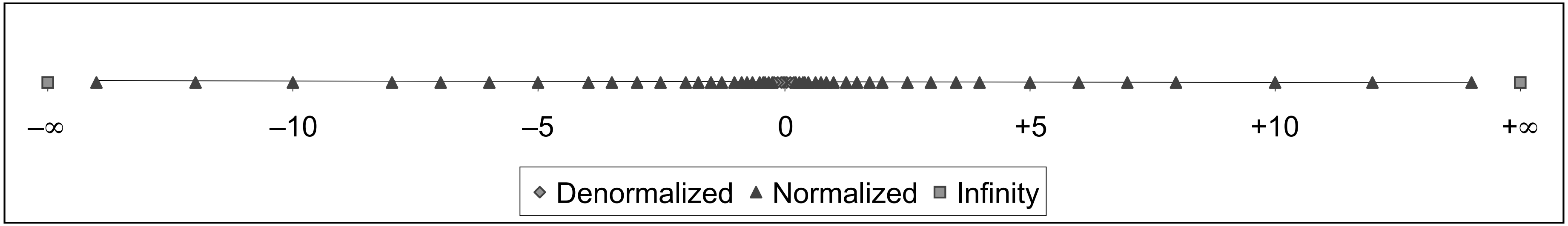


Figure: Full picture of number line for floating point values. Image credit CS:APP

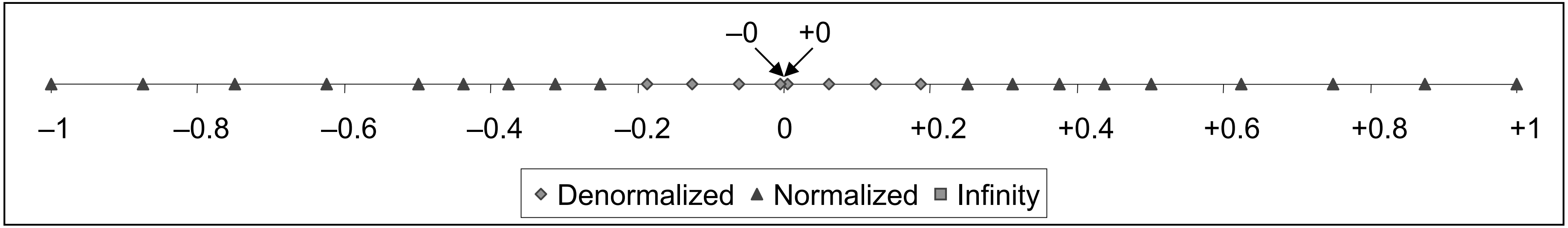


Figure: Zoomed in number line for floating point values. Image credit CS:APP

Denormalized: exp field

For denormalized numbers, $\text{exp} = 0$

Bias

- need a bias to represent negative exponents
- bias = $2^{k-1} - 1$
- bias is the k-bit unsigned integer: 011..111

For denormalized numbers,
E = 1-bias

property	float	double
k	8	11
bias	127	1023
E	-126	-1022

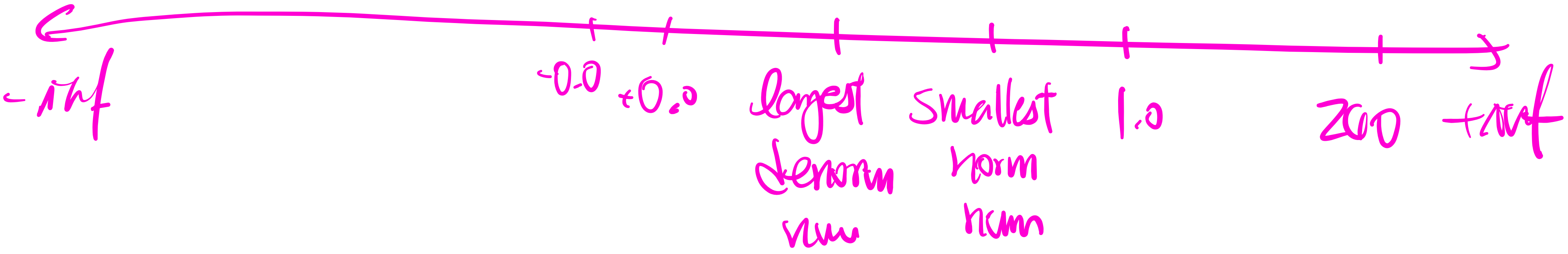
Table: Summary of denormalized exp field

Denormalized: frac field

M = 0.frac
value represented leading with 0

Denormalized: examples

8-bit, 1 bit sign & 4-bit exp field
3 bit frac.



$(-1)^S \cdot M \cdot 2^E$

+inf. 0_1111_000

largest pos number 0_1110_111 $\Rightarrow (-1)^0 \cdot M \cdot 2^E = +1.875 \cdot 2^7$

$= 240$

$E = exp - bias$

$= 14 - (2^{k-1} - 1)$

$= 14 - (2^3 - 1)$

$= 7$

$M = 1.111_2$

$= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$

$= 1.75 + .125$

$= 1.875$

$+1.0 = (-1)^S \cdot M \cdot 2^E$

$= (-1)^0 \cdot 1.0 \cdot 2^0$

1.000 = 1.frac

0_0111_000

$E = exp - bias$

$0 = 7 - 7$

smallest positive norm number

0_0001_000

$(-1)^0 \cdot M \cdot 2^E$

$= +1.0 \cdot 2^E$

$= +1.0 \cdot 2^{-6}$

$= \frac{1}{64}$

E = exp - bias

= 1 - 7

= -6

largest denorm. pos number

0_0000_111

$\Rightarrow (-1)^s \cdot M \cdot 2^E = .875 \cdot 2^{-6}$

$M = 0.111 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = .875$

E = 1 - bias = -6

smallest pos num (denorm)

0_0000_001

$M = 0.001_2 = \frac{1}{8}$

E = -6

$\Rightarrow .125 \cdot 2^{-6} = \frac{1}{8} \cdot \frac{1}{64} = \frac{1}{512}$

+0.0 — 0_0000_000

-0.0 — 1_0000_000

$\frac{1.0}{inf} = +0.0$

$\frac{-1.0}{inf} = -0.0$

Commutative :

$A+B = B+A$ $\qquad$ $\overset{\infty}{\overset{\shortparallel}{238+239}} = \overset{\infty}{\overset{\shortparallel}{239+238}}$

$A+(-B) = (-B)+A$

$A \cdot B = B \cdot A$ $\qquad$ < In the tiny float number system >

Associative

$(A+B)+C = A+(B+C)$

$(240+1)+(-239)$	$240+(1+(-239))$
$= \infty + (-239)$	$= 240+(-238)$
$= \infty$	$= +2$

Distributive

$A(B+C)$	$AB+AC$
$\frac{1}{2}(240+1)$	$\frac{1}{2} \cdot 240 + \frac{1}{2} \cdot 1$
$= \frac{1}{2} \cdot \infty$	$= 120 + \frac{1}{2}$
$= \infty$	$= 120.5$

Table of contents

What's the best that a double precision float can do to represent $1+10^{12} = (10^3)^4 + 1$

$\approx 1024^4 + 1$

$= (2^{10})^4 + 1$

$(-1)^S \cdot M \cdot E$

$S = 0$

$E \approx 40$

$E = exp - bias$

$exp = E + bias$

$= 40 + (2^{(11-1)} - 1)$

$= 40 + (2^{10} - 1)$

$= 40 + 1023$

$= 1063$

52 bits

$M = 1.000 \cdots 000$

$1.000 \cdots 001$

52 bit frac

52 > 40 ?

Floats: Special cases

number class	when it arises	exp field	frac field
+0 / -0		0	0
+infinity / -infinity	overflow or division by 0	$2^k - 1$	0
NaN not-a-number	illegal ops. such as $\sqrt{-1}$, inf-inf, inf*0	$2^k - 1$	non-0

Table: Summary of special cases

Table of contents

Floats: Summary

	normalized	denormalized
value of number	$(-1)^s \times M \times 2^E$	$(-1)^s \times M \times 2^E$
E	E = exp-bias	E = -bias + 1
bias	$2^{k-1} - 1$	$2^{k-1} - 1$
exp	$0 < exp < (2^k - 1)$	$exp = 0$
M	M = 1.frac	M = 0.frac
	M has implied leading 1	M has leading 0
	greater range	greater precision
	large magnitude numbers	small magnitude numbers
	denser near origin	evenly spaced

Table: Summary of normalized and denormalized numbers

Table of contents

Deep understanding 1: Why is exp field encoded using bias?

exp field needs to encode both positive and negative exponents.

Why not just use one of the signed integer formats? 2’s complement, 1s’ complement, signed magnitude?

Deep understanding 1: Why is exp field encoded using bias?

exp field needs to encode both positive and negative exponents.

Why not just use one of the signed integer formats? 2's complement, 1s' complement, signed magnitude?

Answer: allows easy comparison of magnitudes by simply comparing bits.

Deep understanding 1: Why is exp field encoded using bias?

exp field needs to encode both positive and negative exponents.

Why not just use one of the signed integer formats? 2's complement, 1s' complement, signed magnitude?

Answer: allows easy comparison of magnitudes by simply comparing bits.

Consider hypothetical 8-bit floating point format (from the textbook)

1-bit sign, $k = $ 4-bit exp, 3-bit frac.

What is the decimal value of 0b1_0110_111?

What is the decimal value of 0b1_0111_000?

Deep understanding 1: Why is exp field encoded using bias?

exp field needs to encode both positive and negative exponents.

Why not just use one of the signed integer formats? 2's complement, 1s' complement, signed magnitude?

Answer: allows easy comparison of magnitudes by simply comparing bits.

Consider hypothetical 8-bit floating point format (from the textbook)

1-bit sign, $k = 4$-bit exp, 3-bit frac.

What is the decimal value of 0b1_0110_111?

-1.875×2^{-1}

What is the decimal value of 0b1_0111_000?

-2.000×2^{-1}

Table of contents

Deep understanding 2: Why have denormalized numbers?

Why not just continue normalized number scheme down to smallest numbers around zero?

Answer: makes sure that smallest increments available are maintained around zero.

Suppose denormalized numbers NOT used.

What is the decimal value of 0b0_0000_001?

1.125×2^{-7}

What is the decimal value of 0b0_0000_111?

1.875×2^{-7}

What is the decimal value of 0b0_0001_000?

2.000×2^{-7}

Deep understanding 2: Why have denormalized numbers?

Why not just continue normalized number scheme down to smallest numbers around zero?
Answer: makes sure that smallest increments available are maintained around zero.

Suppose denormalized numbers ARE used.

What is the decimal value of 0b0_0000_001?
0.125×2^{-6}

What is the decimal value of 0b0_0000_111?
0.875×2^{-6}

What is the decimal value of 0b0_0001_000?
1.000×2^{-6}

Table of contents