Representing and Manipulating Information: Floating point mastery

Yipeng Huang

Rutgers University

March 2, 2023

Table of contents

Quizzes and programming assignments

Short quiz 4

- Due Friday. All about integers.

Programming assignment 3

- Has been out, due Friday before spring break.

Table of contents

`monteCarloPi.c` Using floating point and random numbers to estimate PI

Table of contents

Floating point numbers

Avogadro's number

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

Scientific notation

- sign
- mantissa or significand
- exponent

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

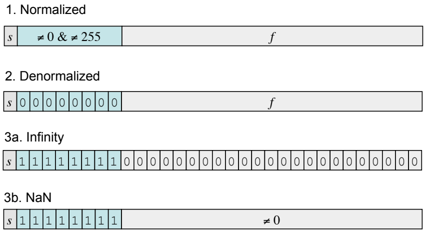

Figure: Different cases within a floating point format. Image credit CS:APP

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,
$0 < \exp < 2^k - 1$

- 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

The IEEE 754 number line

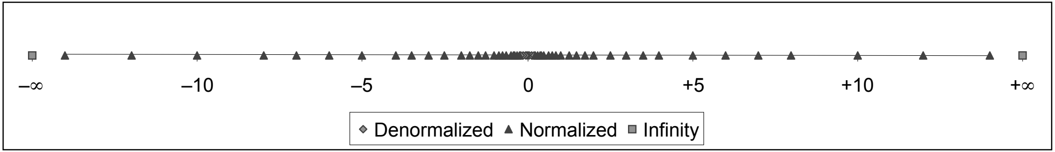

Figure: Full picture of number line for floating point values. Image credit CS:APP

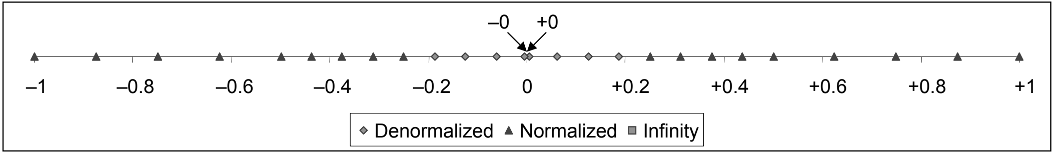

Figure: Zoomed in number line for floating point values. Image credit CS:APP

Denormalized: exp field

For denormalized numbers, 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

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

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