Representing and Manipulating Information: Fixed point and floating point

Yipeng Huang

Rutgers University

February 27, 2023

Table of contents

Quizzes and programming assignments

Short quiz 4

- Has been out as of this morning. Due Friday. All about integers.

Programming assignment 3

- Has been out, due Friday before spring break.

`toBin.c`: Printing the binary representation

- Shifting and masking
- Try modifying to print octal.

Bit shifting

$<< N$ Left shift by N bits

- multiplies by 2^N
- $2 << 3 = 0000\_0010_2 << 3 = 0001\_0000_2 = 16 = 2 * 2^3$
- $-2 << 3 = 1111\_1110_2 << 3 = 1111\_0000_2 = -16 = -2 * 2^3$

$>> N$ Right shift by N bits

- divides by 2^N
- $16 >> 3 = 0001\_0000_2 >> 3 = 0000\_0010_2 = 2 = 16/2^3$
- $-16 >> 3 = 1111\_0000_2 >> 3 = 1111\_1110_2 = -2 = -16/2^3$

Table of contents

Unsigned fixed-point binary for fractions

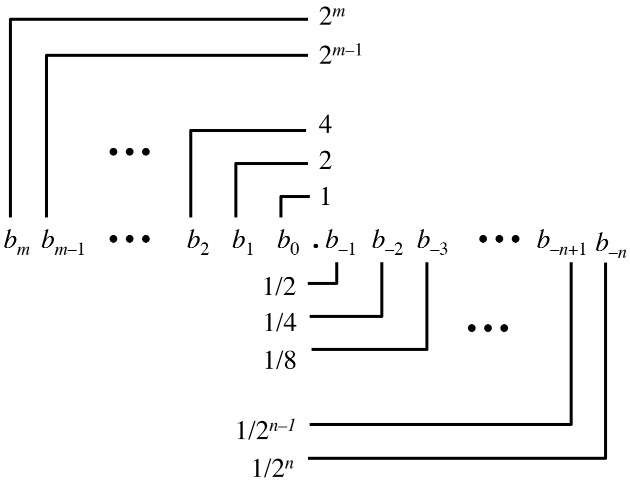

Figure: Fractional binary. Image credit CS:APP

Unsigned fixed-point binary for fractions

unsigned fixed-point char example	weight in decimal
1000.0000	8
0100.0000	4
0010.0000	2
0001.0000	1
0000.1000	0.5
0000.0100	0.25
0000.0010	0.125
0000.0001	0.0625

Table: Weight of each bit in an example fixed-point binary number

- $.625 = .5 + .125 = 0000.1010_2$
- $1001.1000_2 = 9 + .5 = 9.5$

Signed fixed-point binary for fractions

signed fixed-point char example	weight in decimal
1000.0000	-8
0100.0000	4
0010.0000	2
0001.0000	1
0000.1000	0.5
0000.0100	0.25
0000.0010	0.125
0000.0001	0.0625

Table: Weight of each bit in an example fixed-point binary number

- $-.625 = -8 + 4 + 2 + 1 + 0 + .25 + .125 = 1111.0110_2$
- $1001.1000_2 = -8 + 1 + .5 = -6.5$

Limitations of fixed-point

- Can only represent numbers of the form $x/2^k$
- Cannot represent numbers with very large magnitude (great range) or very small magnitude (great precision)

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

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

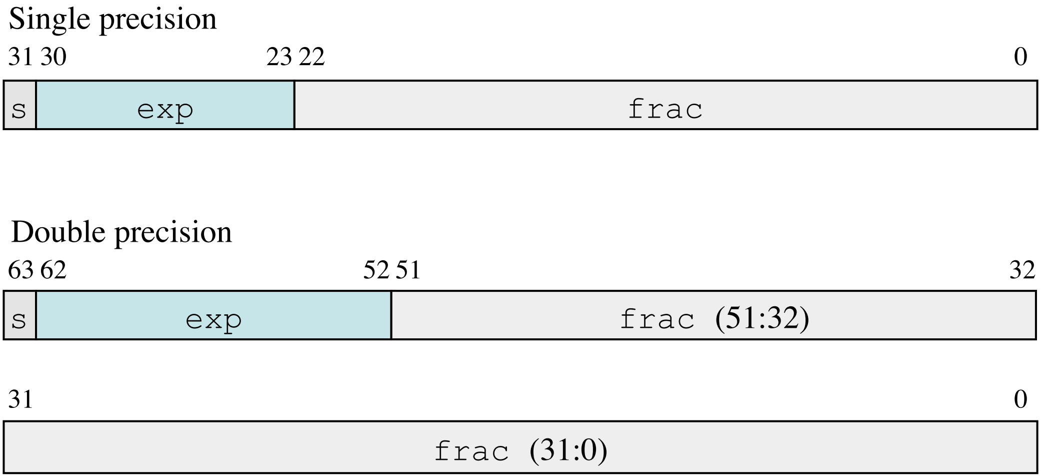

Figure: The two standard formats for floating point data types. Image credit CS:APP

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

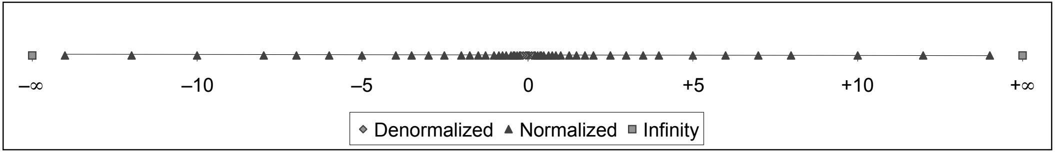

Figure: Full picture of number line for floating point values. Image credit CS:APP

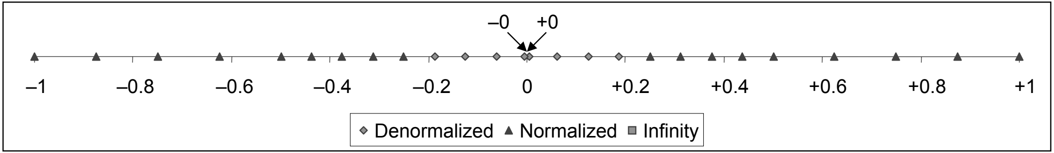

Figure: Zoomed in number line for floating point values. Image credit CS:APP

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

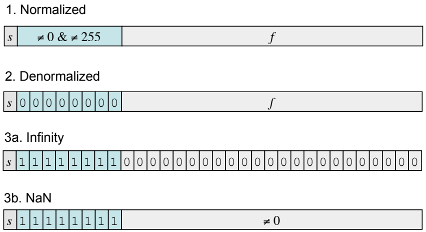

Figure: Different cases within a floating point format. Image credit CS:APP

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

- 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

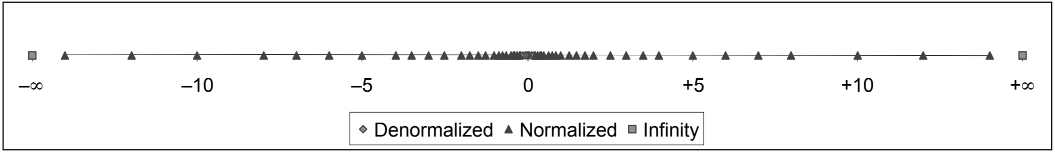

Figure: Full picture of number line for floating point values. Image credit CS:APP

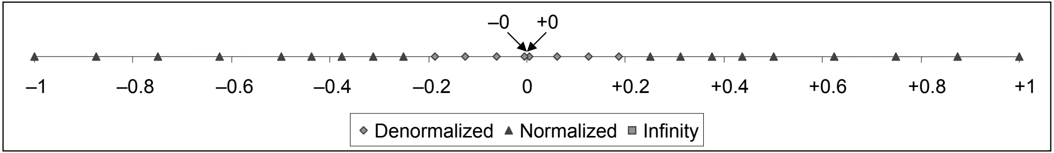

Figure: Zoomed in number line for floating point values. Image credit CS:APP

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

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

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