Floating Point Numbers

Floating Point Numbers

1) Wish to represent wide range of numbers (very small fractions to very large integers)
2) Range is limited by the number of bits
3) Trade precision (accuracy of representation) for range

Scientific Notation

$$3{,}155{,}760{,}000 = \underbrace{3.155760000}_{\text{fraction}} \times \underbrace{10^9}_{\text{exponent}}$$

$$100100 = 1001 \times 2^2$$

Scientific Notation

1001000×2^0

100100×2^1

10010×2^2

1001×2^3

100.1×2^4

10.01×2^5

1.001×2^6

All represent the same value!

Normalized Numbers

Limit the number of digits to the left of the point.

Example: normalize to 1 digit

$$1001000 = 1.001 \times 2^6$$

Assuming all numbers have same base and number of digits to left of the point, only need digits in fraction and exponent.

Normalized Numbers

Exercises: normalize to 1 digit

100100×2^1

111010×2^4

1010.1×2^3

00.01001×2^4

Binary Floating Point

Assign Fixed number of bits for the:

Sign
Exponent
Fraction

Normalize such that 1 is to the left of the decimal point:
$1001101 = 1.001101 \times 2^6$

Since there is always a 1 to the left,
drop the one when storing the number (hidden one)

Single Precision Floating Point

1 bit sign (0 -> positive, 1 -> negative)
8 bit exponent, 00000001 to 11111110 excess 127
23 bit fraction, hidden 1

Example:

1 01111110 10000…0 what is the value?

Sign = -
Exponent = 01111110 = 126 (excess 127) = -1_{10}
Fraction = 1.100…0

$\mathbf{-1.1 \times 2^{-1}}$

Single Precision Floating Point

1 bit sign (0 -> positive, 1 -> negative)
8 bit exponent, 00000001 to 11111110 excess 127
23 bit fraction, hidden 1

Exercise:

0 10000001 11000..0
convert Single Precision Floating Point to Decimal

Single Precision Floating Point

1 bit sign (0 -> positive, 1 -> negative)
8 bit exponent, 00000001 to 11111110 excess 127
23 bit fraction, hidden 1

Example:

-5.5_{10} encode into a Single Precision Floating Point

Single Precision Floating Point

1 bit sign (0 -> positive, 1 -> negative)
8 bit exponent, 00000001 to 11111110 excess 127
23 bit fraction, hidden 1

Exercise:

13.25_{10} encode into a Single Precision Floating Point

Single Precision Floating Point

1 bit sign (0 -> positive, 1 -> negative)
8 bit exponent, 00000001 to 11111110 excess 127
23 bit fraction, hidden 1

Exercise:

-7.75_{10} encode into a Single Precision Floating Point

Trade Precision for Range

32 bits used to encode a float.

Largest Float: 0 11111110 11111…1 = + 1.111 x 2^{127}

Largest Int: 11111111…11 = 1.111 x 2^{32}

Gaps between floats

$10 \times 2^0 = 2$
$11 \times 2^0 = 3$

$10 \times 2^1 = 4$
$11 \times 2^1 = 6$

$10 \times 2^2 = 8$
$11 \times 2^2 = 12$

Trade Precision for Range

Represent 20,000,001 in single precision

Cannot do it.

2^{24} is roughly 16 million. We only have 24 bits to represent the fraction (including the hidden bit) We need more than 24 bits for 20,000,001

Double Precision Floating Point

1 bit sign (0- positive, 1 - negative)
11 bit exponent, 00000000001 to 11111111110 excess 1023
52 bit fraction, hidden 1

Example:

-5.5_{10} encode into a Double Precision Floating Point

Denormalized Numbers

Sign	Exponent	Fraction	
0/1	11111111	000…0	+/- Infinity
0/1	11111111	Nonzero	NaN / Not a Number
0/1	Not all 0 or 1	Anything	Hidden1, excess 127, normalized
0/1	00000000	Nonzero	Denormalized, no hidden 1, 2^{-126} fixed exponent
0/1	00000000	000..0	0

Exercises: What values are encoded in the following?

0 00000000 1000..0

0 00000001 1000..0