

Chapter 3

Representing Real Numbers

Fractions

base: b any integer > 1

digits: $0, 1, \dots, b-1$

number $d_{n-1}d_{n-2}\dots d_2d_1d_0 \cdot d_{-1}d_{-2}d_{-3}$

its definition

$$d_{n-1} \times b^{n-1} + \dots + d_1 \times b^1 + d_0 \times b^0 + d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + d_{-3} \times b^{-3}$$

Examples:

$$3.14 = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} = 3 + .1 + .04$$

Conversions between Decimal and Binary

Binary to Decimal

Technique

- use the definition of a number in a positional number system with base 2
- evaluate the definition formula using decimal arithmetic

Example

$$\begin{aligned} 10.1011 &= 1 \times 2^1 + 0 \times 2^0 + \\ &\quad 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} \\ &= 1 \times 2 + 0 \times 1 + \\ &\quad 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 + 1 \times 0.0625 \\ &= 2.6875 \quad (\text{decimal}) \end{aligned}$$

Decimal to Binary

Technique

- integer part: convert separately, as described before
- fraction part:
 - repeatedly multiply by 2
 - integer part (with is always 0 or 1) is the next digit
 - binary fraction is developed left to right

Example

3.14579

- convert integer part:

11

- convert fraction part: keep multiplying fraction by 2

.14579 × 2 =	0.29158	.0
.29158 × 2 =	0.58316	.00
.58316 × 2 =	1.16632	.001
.16632 × 2 =	0.33264	.0010
etc.		

$$3.14579 = 11.0010\dots \quad (\text{binary})$$

Exercise: Convert .1 (decimal) to binary

Floating Point Numbers

Real numbers represented on a computer are called floating-point numbers.

Scientific Notation

- the following are all equivalent representations of 1234.56

123456.0	$\times 10^{-2}$
12345.6	$\times 10^{-1}$
1234.56	$\times 10^{+0}$
123.456	$\times 10^{+1}$
12.3456	$\times 10^{+2}$
1.23456	$\times 10^{+3}$
0.123456	$\times 10^{+4}$
0.0123456	$\times 10^{+5}$

- the representations differ in that the decimal point “floats” to the left or right (with the appropriate adjustment in the exponent)
- in general, any real number x can be written as

$$x = f \times b^e$$

where b is an integer > 1 (the base), e is any integer,
and $1/b \leq f < 1$ (normalized)

Excess notation

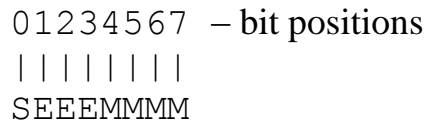
- another way of representing integers using natural numbers
- shift the interval of integers up the number line to the interval 0 to whatever
- work backwards from the number of digits (bits) available for the natural number representatives
- want an equal number of positive and negative integers
- 3-bit representation implies 8 numbers from 0 to 7 \rightarrow -4 to +3

Typical floating-point format in binary (single precision)



- S is the sign of the overall number
- the exponent is stored in excess-128 notation (8-bit exponent \Rightarrow 256 values)
- the mantissa (or fraction) has an implied radix point at the left end (just before bit position 9)

Simplified 8-bit floating-point format



- S is the sign of the overall number
- the exponent is stored in excess-4 notation (3-bit exponent \Rightarrow 8 values)
- the mantissa (or fraction) has an implied radix point at the left end (just before bit position 4)

Example

$$\begin{aligned}
 3.14579 &= 11.0010\dots \quad (\text{binary}) \\
 &= 11.0010\dots \times 2^{+0} \\
 &= .110010\dots \times 2^{+2}
 \end{aligned}$$

representation: 01101101 (rounded)

Exercise: find the floating point representation for .1 (decimal)

IEEE 754 Floating point Standard

- the definitive standard for representing floating point numbers
- single precision (32-bits)
 - sign (1 bit)
 - exponent (8 bits)
 - mantissa/fraction (23 bits)
- double precision (64-bits)
 - sign (1 bit)
 - exponent (11 bits)
 - mantissa/fraction (52 bits)

floating point arithmetic used to be very inconsistent, not anymore.