

CS321-002

Introduction to Numerical Methods

Lecture 1

Number Representation and Errors

Professor Jun Zhang

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046

August 24, 2000

Number in Different Bases

human beings use number base 10 for reasons of the number of fingers

computers use number base 2 for reasons of on-off switch

number based on 8 can be used to facilitate conversion between the base 2 and base 10

Base 10 Expansion:

$$\begin{aligned} 76321 &= 1 + 20 + 300 + 6000 + 70000 \\ &= 1 \times 10^0 + 2 \times 10^1 + 3 \times 10^2 \\ &\quad + 6 \times 10^3 + 7 \times 10^4 \end{aligned}$$

the general formula for a base 10 integer is

$$\begin{aligned} a_n a_{n-1} \dots a_2 a_1 a_0 &= a_0 \times 10^0 + a_1 \times 10^1 \\ &\quad + a_2 \times 10^2 + \dots + a_{n-1} \times 10^{n-1} + a_n \times 10^n \end{aligned}$$

Nonintegers and Fractions

an example of fractional number

$$\begin{aligned} 0.6321 &= \frac{6}{10} + \frac{3}{100} + \frac{2}{1000} + \frac{1}{10000} \\ &= 6 \times 10^{-1} + 3 \times 10^{-2} + 2 \times 10^{-3} \\ &\quad + 1 \times 10^{-4} \end{aligned}$$

the general formula for a decimal fraction

$$\begin{aligned} 0.b_1b_2b_3\dots &= \\ &b_1 \times 10^{-1} + b_2 \times 10^{-2} + b_3 \times 10^{-3} + \dots \end{aligned}$$

decimal fractional numbers can be repeating or nonrepeating (rational or irrational numbers)

$$(a_n a_{n-1} \dots a_1 a_0 . b_1 b_2 b_3 \dots)_{10} =$$

$$\sum_{k=0}^n a_k \times 10^k + \sum_{k=1}^{\infty} b_k \times 10^{-k}$$

Base β Numbers

convert a number from 8 system to 10 system

$$\begin{aligned}(7632)_8 &= 2 \times 8^0 + 3 \times 8^1 + 6 \times 8^2 + 7 \times 8^3 \\ &= 2 + 8(3 + 8(6 + 8(7))) \\ &= 3994\end{aligned}$$

convert a number between 0 and 1

$$\begin{aligned}(0.763)_8 &= 7 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3} \\ &= 8^{-3}(3 + 8(6 + 8(7))) \\ &= \frac{499}{512} \\ &= 0.9746\end{aligned}$$

general formula

$$\begin{aligned}(a_n a_{n-1} \dots a_1 a_0 . b_1 b_2 b_3 \dots)_\beta &= \\ \sum_{k=0}^n a_k \times \beta^k + \sum_{k=1}^{\infty} b_k \times \beta^{-k}\end{aligned}$$

Conversion between Systems

from system α to β with $\alpha < \beta$

- express $(N)_\alpha$ in nested form using powers of α
- replace each digit by corresponding base β number
- carry out indicated arithmetic in base β .

Two examples were given in the previous page

note that one needs to convert the integer and the fractional parts separately

the method is in theory can be used to convert a number between any two bases. But the third step is not easy for humans if β is not 10

Division Algorithm

use remainder-quotient-split method to convert an integer, best if $\alpha > \beta$

$$\begin{aligned}(N)_{\alpha} &= (c_m c_{m-1} \dots c_1 c_0)_{\beta} \\ &= c_0 + \beta(c_1 + \beta(c_2 + \dots + \beta(c_m) \dots))\end{aligned}$$

1. divide the representation by β ;
2. take the remainder as the next digit in the base β representation;
3. repeat the procedure on the quotient.

this procedure is easier to carry out by hand

note that the first digit obtained is the one closest to the radix point

Multiplication Algorithm

use integer-fraction-split process for converting a fractional part

$$x = \sum_{k=1}^{\infty} c_k \times \beta^{-k} = (0.c_1c_2c_3 \dots)_\beta$$

- 1.) multiply the (fractional) number by β ;
- 2.) take the integer part as the first (next) digit;
- 3.) repeat the process on the remaining *fractional part*.

again, the first digit obtained is the one closest to the radix point

terminating fractional number may become non-terminating in different base systems, and vice versa

Use Intermediate Base

we can convert a base 10 number to a base 8 (octal) number, then to base 2 (binary) number, and vice versa

binary-octal table

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

groups of three binary digits can be translated to octal using the binary-octal table

$$\begin{aligned} & (110\ 101\ 001.011\ 100\ 111)_2 \\ & = (651.347)_8 = (?)_{10} \end{aligned}$$

$$(57.321)_{10} = (?)_8 = (?)_2$$

Base 16 Numbers

certain computers with word length being multiples of 4 use hexadecimal system, need A, B, C, D, E, and F to represent 10, 11, 12, 13, 14, and 15.

binary-hexadecimal table

0000	0001	0010	0011	0100	0101
0	1	2	3	4	5
0110	0111	1000	1001	1010	1011
6	7	8	9	A	B
1100	1101	1110	1111		
C	D	E	F		

$$\begin{aligned} & (010\ 111\ 100.110\ 100\ 101\ 111)_2 \\ & = (1011\ 1100.1101\ 0010\ 1111)_2 \\ & = (BC.D2F)_{16} \end{aligned}$$

Normalized Scientific Notation

write a number in the form of

$$728.359 = 0.728359 \times 10^3$$

a normalized floating-point number has the form

$$x = \pm 0.d_1d_2d_3 \dots \times 10^n$$

where $d_1 \neq 0$, n is an integer

in a simple notation

$$x = \pm r \times 10^n \quad \left(\frac{1}{10} \leq r < 1 \right)$$

r is called normalized mantissa and n is the exponent. Also for binary representation

$$x = \pm q \times 2^m \quad \left(\frac{1}{2} \leq q < 1 \right)$$

Machine Numbers

the real numbers that can be represented exactly in a computer are called the *machine numbers* for this computer

most real numbers are not machine numbers

if a computer has word length of the form $0.d_1d_2d_3d_4$, then 0.1011 is a machine number, but 0.10101 is not

machine numbers are machine dependent. The use of normalized floating-point numbers creates a phenomenon of *hole at zero*, a bunch of numbers close to 0 are not representable

A 32-bit Machine

single precision floating-point numbers with 32-bit word length

how to store a floating-point number

$$\pm q \times 2^m$$

with 32 bits?

sign of q needs 1 bit

integer $|m|$ needs 8 bits

number q needs 23 bits

single precision IEEE standard floating-point

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

32-bit Representation

the exponent number c is actually stored, so $m = c - 127$, this is an *excess-127 code*

with normalized representation, the first bit is always 1 and needs not be stored. The mantissa actually contains 24 binary digits with a *hidden bit*

with a mantissa of 32 bit, a machine can have about six significant decimal digits of accuracy, since $2^{-23} \approx 1.2 \times 10^{-7}$

the smallest number ϵ such that $1 + \epsilon = 1$ is called the *machine epsilon* or *unit roundoff error*

for single precision, $\epsilon = 1.2 \times 10^{-7}$

for double precision, $\epsilon = 2.2 \times 10^{-16}$

Representation Procedure

how to represent a real number x ?

- 1.) if x is zero, stored by a full word of zero bits, (a possible sign bit)
- 2.) for nonzero x , first consider sign bit and then consider $|x|$
- 3.) convert both integer and fractional parts of $|x|$ from decimal to octal, then to binary
- 4.) one-plus normalize $(|x|)_2$ by shifting the binary point
- 5.) find the 24 bit 1-plus normalized mantissa
- 6.) find the exponent of 2 by setting it equal to $c - 127$ and determine c
- 7.) denote the 32-bit representation as 8 hexadecimal digits

Summary and Examples

a 32-bit single-precision pattern

$$b_1 b_2 b_2 \cdots b_9 b_{10} b_{11} \cdots b_{32}$$

can be interpreted as the real number

$$(-1)^{b_1} \times 2^{(b_2 b_3 \cdots b_9)_2} \times 2^{-127} \times (1.b_{10} b_{11} \cdots b_{32})_2$$

examples

find the 32-bit representation of -52.234375 .

integer part

$$(52.)_{10} = (64.)_8 = (110\ 100.)_2$$

fractional part

$$(.234375)_{10} = (.17)_8 = (.001\ 111)_2$$

Examples - cont.

$$\begin{aligned}(52.234375)_{10} &= (110\ 100.001\ 111)_2 \\ &= (1.101\ 000\ 011\ 110)_2 \times 10^5\end{aligned}$$

the exponent is $(5)_{10}$, we need to write it as $c - 127 = 5$ so $c = 132$

the stored exponent is

$$(132)_{10} = (204)_8 = (10\ 000\ 100)_2$$

the representation of -52.234375 is

$$\begin{aligned}(1\ 10\ 000\ 100\ 101\ 000\ 011\ 110\ 000\ 000\ 000\ 00)_2 \\ = (1100\ 0010\ 0101\ 0000\ 1111\ 0000\ 0000\ 0000)_2 \\ = (C250F000)_{16}\end{aligned}$$

what number has the representation
 $(45DE4000)_{16}$?