Computer representation of floating point numbers

Jochen Martin Eppler
eppler@biologie.uni-freiburg.de

September 30, 2004

Motivation

As the computer can only store finite numbers, an efficient way to store them is needed. Several problems may occur when dealing

with extreme small or large numbers on a computer.

> *If you don't have that understanding, get advice, take the time to learn, or [. . .] hope for the best.*
>
> — *Bjarne Stroustrup.*

Integer binary numbers

Definition: A *binary number* is a finite sequence of digits $d_i \in \{0, 1\}$, $i = 0, \dots, n-1$. The value $\phi(d_i)$ of each digit is

defined as $\phi(d_i) = 2^i d_i$. The value $\phi(d)$ of a binary number $d = d_{n-1} d_{n-2} \dots d_1 d_0$ is given by

$$\phi(d) = \sum_{i=0}^{n-1} \phi(d_i)$$

smallest number: 0
largest number: $2^n - 1$

Fixed-point numbers

How can real numbers be represented? **Definition:** A *fixed-point*

number consists of $n+1$ pre-decimal and k post-decimal digits, $n, k \geq 0$.

A variety of systems exist to represent fixed-point numbers. Two examples are 'sign and magnitude' and 'two's complement'.

Sign and magnitude

The highest bit is used as sign bit. The value $\phi(d)$ of a number $d = d_n d_{n-1} \dots d_0 . d_{-1} \dots d_{-k}$ is given by

$$\phi(d) = (-1)^{d_n} \sum_{i=-k}^{n-1} \phi(d_i)$$

smallest number: $-(2^n - 2^{-k})$
largest number: $2^n - 2^{-k}$ **Note:** Two representations for 0 (e.g.

100 and 000)
Note: Neighbouring numbers have distance 2^{-k}

Problems with fixed-point numbers

Some problems with fixed-point numbers:

- Very small and very large numbers cannot be represented
- Operations are not algebraically closed:
  2^{n-1} is representable, $2^{n-1} + 2^{n-1}$ is not!
- Associative and distributive law are not applicable:
  $(2^{n-1} + 2^{n-1}) - 2^{n-1} \neq 2^{n-1} + (2^{n-1} - 2^{n-1})$

IEEE 754: Floating-point numbers

The position of the binary point is not fixed!
$\Longrightarrow$ Larger range of numbers with same number of digits.

S	E	M
Sign	Exponent	Mantissa

Value of a number: $(-1)^S \cdot M \cdot 2^E$.

- Single precision (32 bit): S: 1 bit, E: 8 bit, M: 23 bit
- Double precision (64 bit): S: 1 bit, E: 11 bit, M: 52 bit

IEEE 754: Normalized numbers - Mantissa

Observation: representation of a number is not unique.
Definition: If $1 \leq \phi(M) < 2$ the floating-point number is called

normalized, i.e. $M = 1.m_{-1} \dots m_{-k}$. The leading 1 needs not to be saved ("hidden bit"). For every normalized floating-point

number the value of M is calculated as $\phi(M) = 1 + \sum_{i=-1}^{-k} \phi(m_i)$.
Note: 0 is not representable!

Note: Normalized numbers are unique.

IEEE 754: Normalized numbers - Exponent

IEEE 754 defines that

- the exponent bits are interpreted as an unsigned number $(E = e_{n-1} \dots e_0)$
- to be able to represent negative exponents, the so called *bias* is subtracted from E
- the bias B is 127 for single precision, 1023 for double precision $(B = 2^{n-1} - 1)$

For n bits in the exponent the value of E is defined as $\phi(E) = \sum_{i=0}^{n-1} \phi(e_i) - B$.

IEEE 754: Denormalized numbers and special cases

Definition: If all exponent bits are 0, the hidden bit is also interpreted as 0. This way much smaller numbers can be represented. These numbers are called *denormalized*. The value of such a number is

$$\sum_{i=-1}^{-k} \phi(m_i) \cdot 2^{-126}$$

Note: 0 is representable again! If all bits in E are 1 and all bits in M are 0, ∞ is represented.

IEEE 754: Overview of representable numbers

	single precision	double precision
Sign bits	1	1
Exponent bits	8	11
Mantissa bits	23	52
Bits altogether	32	64
Bias	127	1023
Exponent range	-126 to 127	-1022 to 1023
Smallest normalized	2^{-126}	2^{-1022}
Largest normalized	$\sim 2^{128}$	$\sim 2^{1024}$
Smallest denormalized	2^{-149}	2^{-1074}
Decimal range	$\sim 10^{-38}$ to 10^{38}	$\sim 10^{-308}$ to 10^{308}

IEEE 754: Accuracy

The distance from one number to the next varies from $\sim 10^{-45}$ to $\sim 10^{31}$ over the full range of single precision numbers. **Definition:**

This fact is expressed by the *machine epsilon* ε, which is the maximum relative error when representing a real number as a floating-point number. The machine epsilon is the smallest number such that $1 + \varepsilon \neq 1$ still holds. Only fractions whose denominator

is a power of 2 (e.g. $\frac{1}{2}$, $\frac{3}{8}$, $\frac{127}{256}$) can be represented exactly.

IEEE 754: Summary

The properties of IEEE 754 numbers are

- numbers are unique when only normalized numbers are used
- not every number between the smallest and the largest is representable
- numbers are more dense around 0
- operations are still not algebraically closed.
- associative and distributive law are still not applicable!

Byte order and length

The IEEE standard leaves it open, in which direction the numbers are stored in memory. Two possibilities are obvious:

- Big-endian, used by Sparc, Mac, PowerPC machines: The most significant byte is saved first
- Little-endian, used by Intel and Alpha machines: The least significant byte is saved first

The default length of some data types also depends on the architecture.

Integer data types

char:
signed: two's complement (8 bits)
unsigned: binary number (8 bits) **int:**

signed: two's complement (16, 32 or 64 bits)
unsigned: binary number (16, 32 or 64 bits) **long:**

signed: two's complement (32 or 64 bits)
unsigned: binary number (32 or 64 bits)

Floating point data types

Real numbers:
float: IEEE 754 with single precision (32 bit)
double: IEEE 754 with double precision (64 bit) There also exists a

'long double' type, which provides IEEE 754 numbers with extended precision (80 bit).
These provide numbers up to an excess of $\sim 10^{4932}$ and are mainly needed for rounding-free calculations in the hardware itself, but can also be used within C++ programs.

Representation details in C++

Information about the used number representation can be obtained via the template numeric_limits<typename T> which is defined in the header <limits>:

- int radix - base of exponent (usually 2 for binary)
- int digits - number of bits in the mantissa
- T min() - the minimal representable number
- T max() - the maximal representable number
- T epsilon() - the machine epsilon (ε)

What to remember?

- Operations are not algebraically closed.
- Associative and distributive law are not applicable.
- Not every real number is representable.
- Information about the used data types are available via numeric_limits.

References

- Andrew S. Tanenbaum: *Structured Computer Organization*
  Fourth Edition, ISBN: 0-13-020435-8, Prentice Hall 1999
- Bjarne Stroustrup: *The C++ Programming Language*
  Third Edition, ISBN: 0-201-70073-5, Addison-Wesley 1997
- Michael Knorrenschild: *Numerische Mathematik*
  First Edition, ISBN: 3-446-22169-7, Fachbuchverlag Leipzig
- Institute of Electrical and Electronics Engineers: *Homepage*
  http://grouper.ieee.org/groups/754/