Number Representation

Representation of Numbers

- Computer represent all numbers, other than integers and some fractions with imprecision.
- Numbers are stored in some approximation which can be represented by a fixed number of bits or bytes.
- There are different types of "representations" or "data types".

Representation of Numbers

- Information in the computer are stored as a groups of binary digits.
- An individual digit is called a **bit.**
- Bits are grouped into 8-bit collections called **bytes**.
- Memory is normally measured in terms of bytes.

- Bytes are further grouped into 4 or more byte groupings to make up a computer **word**.
- The most common word size in most modern computers is 32 bits (4 8-bit bytes)
- Many programs do scientific calculations using double (64-bit) words.
- What is the largest integer value that can be expressed in 32 bits?

32-bit Unsigned Integers

Maximum Value

11111111111111111111111111111111

$2^{32}-1 = 4,294,967,295$

Fixed point Integer Representation

- Signed Magnitude: Leading bit represents sign and remaining bits represent the corresponding unsigned number.
- One's Complement: Negative representation is obtained by flipping all the bits of the unsigned number.
- Two's Complement: Negative representation is obtained by flipping all the bits and adding one.

Fixed point Integer Representation

- A common method of integer representation is sign and magnitude rep.
- One bit is used for the sign and the remaining bits for the magnitude.
- For example, let 0 denote positive and 1 denote negative
- Clearly there is a restriction to the numbers which can be represented.

Fixed point Integer Representation

- The range of numbers represented is
  - $-2^{m-1}-1$ to $+2^{m-1}-1$.
- With an 8 bit byte, the largest and smallest numbers represented are +127 and –127.

Sign bit (+ve number)

+127	=	0	1	1	1	1	1	1	1
-127	=	1	1	1	1	1	1	1	1

Sign bit (-ve number)

Fixed point Integer Representation

- Things to note:

1. Fixed point numbers are represented as exact.
2. Arithmetic between fixed point numbers is also exact provided the answer is within range.
3. Division is also exact if interpreted as producing an integer and discarding any remainder.

Ones and twos compliment representation

- Can be used for negative numbers.
- The ones compliment form of a negative number is the bitwise NOT applied to it
- Ie. The compliment of its positive counterpart
- The twos compliment form of a negative number is the bitwise NOT applied to it with a 1 added to it.

- The range of signed numbers using ones' compliment is represented by
  - $-2^{m-1}-1$ to $2^{m-1}-1$

Floating point Representation

Floating point Representation

- In floating point representation, numbers are represented by a sign bit s, an integer component e, a positive integer mantissa M.

Computer Representation of Real (floating point) Numbers

$$m * b^e$$

mantissa

base
(10 or 8 or 2)

exponent
(integer)

Floating point Representation

- Most computers use the IEEE representation where the floating point number is normalized.
- The two most common IEEE rep are:

1. IEEE Short Real (Single Precision): 32 bits – 1 for the sign, 8 for exponent and 23 mantissa
2. IEEE Long Real (Double Prec): 64 bits – 1 sign bit, 11 for exp and 52 for the mantissa

Floating Point Numbers

Example 1.
Represent 27.25 using the IEEE short real rep
First Let us convert 27.25 to binary:
11011.01
This is equal to:
0.1101101×2^5
In normal form

Floating Point Numbers

Using the IEEE Short Real : 32 bits – 1 for the sign, 8 for exponent and 23 mantissa
Internal representation of 27.25 is then:

0 00000101 11011010000000000000000

Sign of number

exponent (with sign)

Note: limited number of digits to represent mantissa

Floating Point Numbers

Example 2.
Represent 0.1 using the IEEE short real representation
First Let us convert 0.1 to binary:
0.000110011001100110011001100110110110…

This is equal to:
$0.11001100110011001100110011001100110011\ldots \times 2^{-3}$

In normal form

Floating Point Numbers

Using the IEEE Short Real : 32 bits – 1 for the sign, 8 for exponent and 23 mantissa
Internal representation of 0.1 is then:

0 10000011 10011001100110011001100

Sign of number

exponent (with sign)

Note: limited number of digits to represent mantissa

Floating Point Numbers

Example 3.
Represent -78.5 using the IEEE short real rep
First Let us convert 78.5 to binary:
1001110.1
This is equal to:
0.10011101×2^{7}

In normal form

Floating Point Numbers

Using the IEEE Short Real : 32 bits – 1 for the sign, 8 for exponent and 23 mantissa
Internal representation of 0.1 is then:

1 00000111 10011101000000000000000

Sign of number

exponent (with sign)

Note: limited number of digits to represent mantissa

Errors in computations

- There five types of errors in computation:

1. Mistakes
2. Random error
3. Truncation error
4. Roundoff error
5. Propagated error

Errors and Uncertainties

- Mistakes: are typographical errors entered with program or maybe running the program using the wrong data etc.

Errors and Uncertainties

- Random errors: these are caused by random fluctuations in electronics due to for example power surges.
- The likelihood is rare but there is no control over them.

Errors and Uncertainties

- Truncation or approximation errors: these occur from simplifications of mathematics so that the problem may be solved.
- For example replace of an infinite series by a finite series.
- Eg: $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \approx \sum_{n=0}^{N} \frac{x^n}{n!} = e^x + \zeta(x, N)$

Errors and Uncertainties

- Where $\zeta(x, N)$ is the total absolute error.
- The truncation error vanishes as N is taken to infinity.
- For N much larger than x, the error is small.
- If x and N are close then the truncation error will be large.

Errors and Uncertainties

- Roundoff error: since most numbers are represented with imprecision by computers (and general restrictions) this leads to a number being lost.
- The error as result of the roundoff or truncation of digits is known as the roundoff error.
- Eg1: $2\left(\frac{1}{3}\right) - \frac{2}{3} = 0.6666666 - 0.6666667 = -0.0000001 \neq 0$

Round-Off Error

If we represent the perfectly good decimal fraction 0.1 as a binary fraction we get:

0.00011001100110011001100110011011 (0110 repeats)

Suppose we have a computer that stores 11 bits in the mantissa. The binary fraction (after rounding) becomes:

$0.110011001101 \cdot 2^{-3} = 0.1000061_{10}$

Errors and Uncertainties

- Propagated error: this is defined as an error in later steps of a program due to an earlier error.
- This error is added the local error(eg. to a roundoff error).

- Propagated error is critical as errors may be magnified causing results to be invalid.
- The stability of the program determines how errors are propagated.

Calculating the Error

- A simple way of looking at the error is as the difference between the true value and the actual value.
- Ie:

  Error (e) = True value – Approximate value

Calculating the Error

- Three other ways of defining the error are:
- Absolute error
- Relative error
- Percentage error

Calculation the Error

- Absolute error.

  e_a = |True value – Approximate value|

$$e_a = |X - X'| = |Error|$$

Calculating the Error

- Absolute error:

  e_a = |True value – Approximate value|

$$e_a = |X - X'| = |Error|$$

- Relative error is defined as:

$$e_r = \left|\frac{Error}{True\ Value}\right| = \left|\frac{X - X'}{X}\right|$$

Calculating the Error

- Percentage error is defined as:

$$e_p = 100\, e_r = 100\left|\frac{X - X'}{X}\right|$$

Examples

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.

$$\sqrt{2} = 1.41421356$$

Examples

$$e_a = |\text{True value} - \text{Approximate value}| \quad (\textit{absolute error})$$

$$\therefore e_a = |1.41421356 \; - 1.414|$$

$$= 0.00021356$$

Examples

$$e_r = \left| \frac{Error}{True \quad Value} \right| \quad (\textit{relative error})$$

$$\therefore e_r = \left| \frac{0.00021356}{\sqrt{2}} \right| = 0.151 \times 10^{-3}$$

Examples

$$\therefore e_r = \left| \frac{0.00021356}{\sqrt{2}} \right| = 0.151 \times 10^{-3}$$

$$\therefore e_p = e_r \times 100 \quad = 0.151 \times 10^{-1}\,\% \quad (\textit{percentage error})$$