Decimal Floating-Point: Algorism for Computers

Arith16 — 16 June 2003

Mike Cowlishaw
IBM Fellow

http://www2.hursley.ibm.com/decimal

Overview

• Why decimal arithmetic is increasingly important
• Why hardware support is needed
• How decimal arithmetic is done

Origins of decimal arithmetic

• Decimal (base 10) arithmetic has been used for thousands of years
• Algorism (Indo-Arabic place value system) in use since 800 AD
• Many early computers were decimal

Decimal computation today

• Pervasive in financial, commercial, and human-centric applications — often a legal requirement
• Benchmarks do not reflect actual use
• 55% of numeric data in databases are decimal (and a further 43% integers)

Why floating-point?

• Traditional integer arithmetic with ‘manual’ scaling is awkward and error-prone
• Floating-point is increasingly necessary
  – division and exponentiation
  – interest calculated daily
  – telephone calls priced by the second
  – calculators, financial analysis, etc.

Why not use binary FP?

• Binary fractions cannot exactly represent all decimal fractions
• 1.2 x 1.2 ⟷ 1.44 ?
  – 1.2 in a 32-bit binary float is actually: 1.2000000476837158203125
  – and this squared is: 1.440000057220458984375

Copyright © IBM Corporation 2003. All rights reserved.
A financial example...

• 5% sales tax on a $0.70 telephone call, rounded to the nearest cent
• 1.05 x 0.70 using binary double type is 0.73499999999999998667732370449812151491641998291015625 (should have been 0.735)
• rounds to $0.73, instead of $0.74

Hence...

• Binary floating-point cannot be used for commercial applications
  – cannot match values computed by hand
  – cannot meet legal and financial requirements, which are based on 2,500+ years of decimal arithmetic
• So applications use decimal software floating-point packages...

...but decimal software is slow...

some measurements ...

<table>
<thead>
<tr>
<th>times in µs</th>
<th>x=y+z</th>
<th>x=y*z</th>
<th>x=y/z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java BigDecimal</td>
<td>1.250</td>
<td>1.100</td>
<td>8.440</td>
</tr>
<tr>
<td>Binary hardware</td>
<td>0.006</td>
<td>0.006</td>
<td>0.078</td>
</tr>
<tr>
<td>Decimal penalty</td>
<td>208x</td>
<td>183x</td>
<td>108x</td>
</tr>
</tbody>
</table>

(These are 9-digit timings. 27-digit calculations are 9x worse for multiply and divide.)

Effect on real applications

• The ‘telco’ billing application
  – 1,000,000 calls read from file, priced, taxed, and printed (two minutes-worth)

<table>
<thead>
<tr>
<th>time in decimal operations</th>
<th>fastest C package</th>
<th>Java BigDecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>72.2%</td>
<td>93.2%</td>
</tr>
</tbody>
</table>

The path to hardware...

• A 2x to 10x performance improvement in applications makes hardware support very attractive
• IEEE 854 tells us how to compute the value of floating-point results
• We can use redundant encodings to allow fixed-point and integer arithmetic, too

Traditional two-integer form

• Allows integer, fixed-point, and floating-point numbers in one representation
  – integers always have exponent = 0
  – in general: numbers with the same number of decimal places have the same exponent, and need no alignment for addition
    e.g., 1.23 and 123.45 both have exponent -2
    [123, -2] and [12345, -2]
Example: multiplication

• The significands are multiplied (an integer operation), and the exponent is the sum of the operand exponents
  
  - $123E^{-2} \times 45E^{-1}$ gives $5535E^{-3}$
  - $122E^{-2} \times 45E^{-1}$ gives $5490E^{-3}$

• Independent calculations for the two parts

• No further processing is necessary unless rounding (etc.) is needed

Rounding

• Correct rounding, as in IEEE 754/854
  
  - additional rounding mode (round-half-up)

• A rounded normal number will always have maximum digits (the first digit will be non-zero)

• Subnormals may have leading zero digit(s)

Note …

• The core operations when no rounding occurs are simple integer operations; integer arithmetic is a subset

• Comparison does not distinguish between redundant encodings of the same value

• The rules are base-independent

Integer-based floating-point

• Compatible with:
  
  - IEEE 754/854
  - manual processes (algorism)
  - legal requirements
  - programming language data types
    (COBOL, PL/I, Java, C#, REXX, Visual Basic, etc.)
  - databases (DB2, SQL Server, Oracle, etc.)
  - application testcase data formats
  - mixed-type arithmetic: $12 \times \$9.99$

Summary

• Hardware two-integer arithmetic (or the equivalent) gives same results as software
  
  - allows the hardware to be used to accelerate existing applications and processes
  
  - e.g., a typical large bank has 1,480 programmers, 65 application subsystems, 900+ IT projects/year, and 10,000+ programs in use

• This does not conflict with IEEE 754/854
  
  - allows integer and FP in the same unit

Questions?

http://www2.hursley.ibm.com/decimal

(Google: decimal arithmetic)