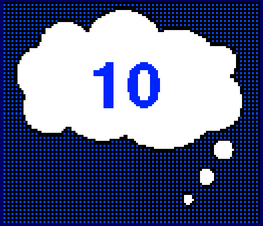
Ecma/TC39/2008/083

IEEE 754-2008

ECMA TC39/TG1 – 25 September 2008

Mike Cowlishaw
IBM Fellow



Overview

- Summary of the new standard
- Binary and decimal specifics
- Support in hardware, standards, *etc.*
- Questions?

IEEE 754 revision

- Started in December 2000 – 7.7 years
  - 5.8 years in committee (92 participants + e-mail)
  - 1.9 years in ballot (101 voters, 8 ballots, 1200+ comments)
- Removes many ambiguities from 754-1985
- Incorporates IEEE 854 (radix-independent)
- Recommends or requires more operations (functions) and more language support

Formats

- Separates sets of floating-point numbers (and the arithmetic on them) from their encodings (‘interchange formats’)

- Includes the 754-1985 basic formats:
  - binary32, 24 bits (‘single’)
  - binary64, 53 bits (‘double’)

- Adds three new basic formats:
  - binary128, 113 bits (‘quad’)
  - decimal64, 16-digit (‘double’)
  - decimal128, 34-digit (‘quad’)

Why decimal? A web page…

- Parking at Manchester airport…

- £4.20 per day …

  … for 10 days …

  … calculated on-page using ECMAScript

  Answer shown:

Why decimal? A web page…

- Parking at Manchester airport…
- £4.20 per day …

  … for 10 days …

  … calculated on-page using ECMAScript

  Answer shown: £41.99

  (Programmer must have truncated to two places.)

Where it costs real money…

- Add 5% sales tax to a $ 0.70 telephone call, rounded to the nearest cent

- 1.05 x 0.70 using binary double is exactly

  0.73499999999999998667732370449812151491641998291015625

  (should have been 0.735)

- rounds to $ 0.73, instead of $ 0.74

Interchange formats

- Encodings (bit-field layouts) for binary and decimal floating-point numbers:
  - binary16, 11 bit
  - decimal32, 7 digits
  - all five basic formats
  - longer formats, to be used as required

- Decimal encodings use compression to achieve similar precision to binary

Rounding

- Same four rounding algorithms as 1985:
  - round to nearest, ties to even (Banker's)
  - round towards +∞ (ceiling)
  - round towards –∞ (floor)
  - round towards zero (down)
- Adds:
  - round to nearest, ties away from zero

Operations

- More required operations added, notably:
  - fused multiply-add (FMA)
  - minnum and maxnum (ignore one NaN)
  - correctly-rounded conversions
  - copy operations (abs, negate…)
  - classification (isNaN, isFinite, *etc*.)
  - quantize and sameQuantum
  - compareTotal for reproducible sort (–0 < +0)

Decimal floating-point

- Unnormalized (1.20 and 1.2 are different but have the same value) and compatible with:
  - manual processes (algorism)
  - legal requirements (1 Euro = 340.750 drachmas)
  - current programming language data types (COBOL, PL/I, C#, Java, Rexx, JDBC, *etc.*)
- No integer type needed: 12 x $ 9.99
- Operations fully define results

NaNs

- Copy operations are NaN-neutral, so signs on NaNs are now predictable in some cases, and ‘payload’ is a stronger concept
- Signaling NaNs are required
- Fewer variations in binary NaN encodings
  – no variations in decimal NaNs

Exceptions: flags and traps

- As in 754-1985, flags (or equivalent) are required, but traps (handlers) are not
- Results after exceptions now well-defined
- Only one underflow rule for decimal
  - still two ways to detect underflow for binary
- Recommendations for exception handling

Recommended functions

- 36 trigonometric and other functions, and their special cases
- Must be correctly rounded to conform
- Need not report exact/inexact results

Expression evaluation

- Aimed at improving languages’ support of floating-point operations
- Recommends how to specify the evaluation of compound expressions (more than one operation)
- Points out pitfalls and problems caused by optimizations

Reproducible results

- Spells out how to achieve reproducible results (*i.e.*, the same results regardless of platform and hardware)
- Requirements for a language that supports reproducible programming
- Lists what to do and what not to use (in general good advice for all programs)

Annexes

- Bibliography
- Recommendations for debuggers

Standards and Industry support

- Changes in binary support will be incremental

- All the ‘action’ is in decimal support

Standards, *etc.*

- ISO C and C++ are jointly adding decimal floating-point as first-class primitive types
  - entering final ballots
  - in GCC since 2007, plus IBM compilers
- Java 5 BigDecimal (arithmetic) in 2004
  - MathContext for IEEE 754 decimal types
- C# and .Net ECMA and ISO changed 2006
  - arithmetic tweaked; allows use of 754 formats

Standards, *etc.* [2]

- COBOL 2002 has floating-point decimal
  - 754 128-bit type in COBOL 2008
- SAP changed to decimal128 in 2007
- XML Schema 1.1 draft now has *pDecimal*
- Strong support expressed by users, open source projects (Python, Eiffel, *etc.*), SHARE, academia, and many others

Hardware

- power.org (PowerPC, *etc.*) architecture
  - in POWER6 (2007)
  - first decimal FP hardware for 40 years
- IBM System z mainframes
  - System z9 (assists + emulation) 2007
  - System z10 (full hardware) Feb. 2008
- Other vendors' plans not yet announced

Software – decFloats C package

- Fixed-precision decimal package
  - computes directly on the IEEE decimal64/128
  - fastest decimal package
- Open source cross-platform ANSI C
  - very free licence (three sentences)
  - widely used (IBM, SAP, GCC, *etc*.)
  - included in the decNumber package, at: `http://speleotrove.com/decimal`

Software – Gnu C Compiler (GCC)

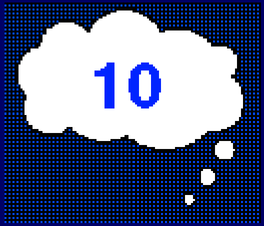
- Open source C compiler for many platforms
- Includes the proposed ISO C decimal types, `_Decimal64` and `_Decimal128`
  - uses decFloats for decimal arithmetic when no hardware available
  - System z and PowerPC hardware code generation is in GCC 4.3

Benefits of the new types

- Standard data types for decimal bring all the benefits that binary applications enjoy:
  - known, standard, formats for interchange
  - faster processing (especially with hardware)
  - no conversions (only one type of decimal)
  - well-defined arithmetic, rounding rules, *etc.*
  - long-term: a single numeric type (no binaries)

Questions?

Google: decimal arithmetic



IEEE 754 Participants

- Ad hoc (grandfathered) process until 2006
- Many users and academics participated by mailing list (140+ registered)
- Voting members (36) were mostly hardware vendors (Apple, AMD, HP, IBM, Intel, Sun, *etc.*)
  - Intel (9), IBM (7), and HP (4) sent most

Format details

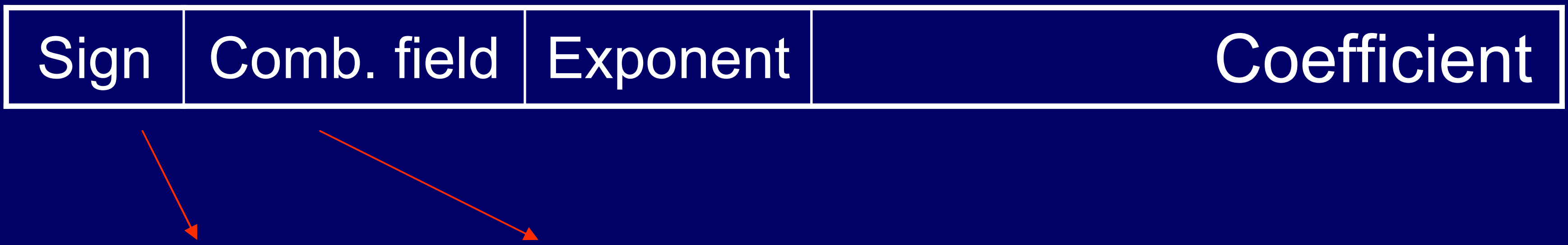
Formats

IEEE 754r formats

size (bits)	digits	exponent range
32	7	-95 to +96
64	16	-383 to +384
128	34	-6143 to +6144

(range always larger than same-size binary format)

IEEE 754r: common 'shape'



- Sign and combination field fit in first byte
  - combination field (5 bits) combines 2 bits of the exponent (0−2), first digit of the coefficient (0−9), and the two special values
  - allows 'bulk initialization' to zero, NaNs, and ± Infinity by byte replication

Exponent continuation

Sign	Comb. field	Exponent	Coefficient

Simple concatenation

Format	exponent bits	bias	normal range
32-bit	2+6	101	-95 to +96
64-bit	2+8	398	-383 to +384
128-bit	2+12	6176	-6143 to +6144

(All ranges larger than binary in same format.)

Coefficient continuation

Sign	Comb. field	Exponent	Coefficient

- Densely Packed Decimal – 3 digits in each group of 10 bits (6, 15, or 33 in all)

- Derived from Chen-Ho encoding, which uses a Huffman code to allow expansion or compression in 2–3 gate delays

Unexpected behaviour

Another typical problem; the Java loop:

```
for (double d = 0.1; d <= 0.5; d += 0.1)
    System.out.println(d);
```

displays five numbers, whereas:

```
for (double d = 1.1; d <= 1.5; d += 0.1)
    System.out.println(d);
```

displays only four.

How computers compute

- Binary arithmetic will continue to be used, but, perhaps …

  “in the relatively distant future, the continuing decline in the cost of processors and of memory will result (in applications intended for human interaction) in the displacement of substantially all binary floating-point arithmetic by decimal”

Professor W. Kahan, UCB