

Brakefield's correspondence with IEEE 854 committee

1st page sent to Cody RE IEEE854

W. J. Cody
MCS-221/G223
Argonne National Laboratory
Argonne, Illinois 60439

16 Jan 1985

Subject: Comments on IEEE 854 Floating Point Standard

Dear Sirs:

The comments are that if gradual underflow is to be permitted, the gradual overflow should also be provided. In the spirit of G. W. Gerrity's Transactions on Computers article, the numbers represented in gradual underflow should have multiplicative inverses.

The following presents an encoding which allows efficient implementation of arithmetic for both gradual series. Only one code is unused and again following Gerrity it would be the null or undefined value. It could also be used for an unsigned infinity.

The scheme uses a novel encoding for the underflow numbers:

When the underflow exponent occurs, the mantissa is examined least significant bit for a one. The number of zeros before the first one bit becomes a scale factor subtracted from the exponent. The remaining mantissa bits (less the zeros and the first one bit) and the hidden most significant one bit make up the mantissa. Note that except for masking out the least significant one bit and adjusting the exponent the number retains the normal format.

The scheme for overflow numbers is very similar:

When the overflow exponent occurs, the mantissa is examined in the same way. The number of zeros is then added to the exponent.

As an example, take the case of 8-bit floating point numbers:

There will be a sign bit, four exponent bits using offset binary, and three mantissa bits in addition to the hidden mantissa bit. Let 01000000 represent 1.0, s1111xxx will be the overflow numbers, and s0000xxx will be the underflow numbers.

The following are the underflow/overflow codes and their values:

00000000 zero	10000000 undefined or infinity
00000100 $1/1024$	01111000 1024
00000010 $2/1024$	01111100 512
00000110 $3/1024$	01111110 384
00000001 $4/1024$	01111010 256
00000011 $5/1024$	01111111 224
00000101 $6/1024$	01111101 192
00000111 $7/1024$	01111011 160
	01111001 128
00001000 $8/1024$	01110111 120

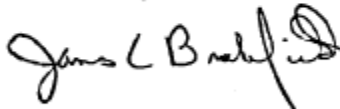
2nd page sent to Cody RE IEEE854

Where the underlined parts are the non-hidden mantissa bits. In the above $1/1024$ and 1024 have only the hidden mantissa bit. $2/1024$, $3/1024$, 512 , and 384 have one mantissa bit in addition to the hidden bit. The rest have two mantissa bits. Note that many of the fractions have exact reciprocals and $7/1024$ and $6/1024$ reciprocate to the same number (160).

If a larger range of underflow and overflow numbers is desired additional exponent values can be reserved. I personally would like to see the floating point standard as simple as possible and prefer not to include signed infinity, NaNs, and reserved operands. Let the single code 10000000 bear the burden via exception routines.

Your standard does not specify encodings, which is good. The above encodings are presented as one possibility and to show a tentative implementation.

Sincerely



JAMES C. BRAKEFIELD
Technology Incorporated
300 Breesport
San Antonio, Texas 78216
AC 512 - 533-1228

George W. Gerrity, Computer Representation of Real Numbers, IEEE
Transactions on Computer, August 1982, p709:8:C-31.

IEEE Task P854

Minutes, 1 March 1985

James C. Brakefield (Technology Incorporated, San Antonio) on a scheme for "gradual overflow", allowing values modestly above the overflow threshold to be represented with decreasing relative precision as the magnitude increases.

IEEE Task P854

Minutes, 12 April 1985

Brakefield correspondence (P854/85-2.14 and 2.15) revisited: Gradual underflow succeeds when it does because rounding error is insignificant compared to previous or subsequent rounding errors. Unfortunately, gradual overflow does not have this property. Cody will issue clarification.