Cairo University
Electronics and Communications Department

Computer Arithmetic: Decimal and the 'fine print' of the standard

Hossam A. H. Fahmy

Do we need decimal?

$(1/10)_{\beta=10} = (0.1)_{\beta=10}$ but in binary it is $(0.000110011001100\ldots)_{\beta=2}$ which the computer rounds into a finite representation.

For a computer using binary64, if $y = 0.30$ and $x = 0.10$ then $3x - y = 5.6 \times 10^{-17}$. Furthermore, $2x - y + x = 2.8 \times 10^{-17}$. Leading to the wonderful surprise that

$$\left.\frac{3x-y}{2x-y+x}\right|_{(x=0.1,y=0.3)} = 2 \text{ !}$$

For a human, is 0.050 kg = 0.05 kg*?*

Humans and decimal numbers

If both measurements are normalized to 5×10^{-2} and stored in a format with 16 digits as $(5.000000000000000 \times 10^{-2})$ they are

- indistinguishable and
- give the incorrect impression of a much higher accuracy (0.050000000000000 kg).

To maintain the distinction, we should store

$0.000000000000050 \times 10^{12}$ first measurement
$0.000000000000005 \times 10^{13}$ second measurement

with all those leading zeros. Both are members of the same *cohort*.

IEEE decimal formats

	Sign	Combination	Trailing Significand
	±	exponent and MSD	$t = 10J$ bits
64 bits:	1 bit	13 bits, bias = 398	50 bits, 15 + 1 digits
128 bits:	1 bit	17 bits, bias = 6176	110 bits, 33 + 1 digits

IEEE decimal64 and decimal128 formats.

Note that $(-1)^s \times \beta^e \times m = (-1)^s \times \beta^q \times c$ when

$$\begin{aligned} m &= d_0.d_{-1}d{-2}\ldots d_{p-1}, \\ c &= d_0d_{-1}d{-2}\ldots d_{p-1}, and \\ q &= e-(p-1). \end{aligned}$$

The combination field encodes the exponent q and four significand bits.

Back to addition

Decimal examples:

$$\begin{array}{rll} & 1.324 & \times 10^5 \\ + & 1.576 & \times 10^3 \end{array} \left\{ \begin{array}{rll} & 1.324 & \times 10^5 \\ + & 0.01576 & \times 10^5 \\ \hline & 1.33976 & \times 10^5 \\ \approx & 1.340 & \times 10^5 \end{array}\right.$$

Back to addition

Decimal examples:

$$\begin{array}{rll} & 1.324 & \times 10^5 \\ + & 1.576 & \times 10^3 \end{array} \left\{ \begin{array}{rll} & 1.324 & \times 10^5 \\ + & 0.01576 & \times 10^5 \\ \hline & 1.33976 & \times 10^5 \\ \approx & 1.340 & \times 10^5 \end{array}\right.$$

$$\begin{array}{rll} & 9.853 & \times 10^7 \\ + & 1.466 & \times 10^6 \end{array} \left\{ \begin{array}{rll} & 9.853 & \times 10^7 \\ + & 0.1466 & \times 10^7 \\ \hline & 9.9996 & \times 10^7 \\ \approx & 1.000 & \times 10^8 \end{array}\right.$$

Back to addition

Decimal examples:

$$\begin{array}{rll} & 1.324 & \times 10^5 \\ + & 1.576 & \times 10^3 \end{array} \left\{ \begin{array}{rll} & 1.324 & \times 10^5 \\ + & 0.01576 & \times 10^5 \\ \hline & 1.33976 & \times 10^5 \\ \approx & 1.340 & \times 10^5 \end{array}\right.$$

$$\begin{array}{rll} & 9.853 & \times 10^7 \\ + & 1.466 & \times 10^6 \end{array} \left\{ \begin{array}{rll} & 9.853 & \times 10^7 \\ + & 0.1466 & \times 10^7 \\ \hline & 9.9996 & \times 10^7 \\ \approx & 1.000 & \times 10^8 \end{array}\right.$$

$$\begin{array}{rll} & 1.324 & \times 10^3 \\ - & 1.321 & \times 10^3 \end{array} \left\{ \begin{array}{rll} & 1.324 & \times 10^3 \\ + & 8.679 & \times 10^3 \\ \hline & 0.003 & \times 10^3 \\ \stackrel{?}{=} & 3.000 & \times 10^0 \end{array}\right.$$

Multiplication

1. No alignment is necessary.
2. Multiply the significands.
3. Add the exponents.
4. The sign bit of the result is the *XOR* of the two operand signs.

Is it really that simple?

Division

1. No alignment is necessary.
2. Divide the significands.
3. Subtract the exponents.
4. The sign bit of the result is the *XOR* of the two operand signs.

You know it is not that simple!

Where is the nearest number?

Humans add 1/2 of the *LSD* position of the desired precision to the *MSD* of the portion to be discarded.

For a sign-magnitude representation this gives RNA but not RNE:

38.5	$X\ X\ X\ X$	←Number to be rounded
0.5	0 0 0 0	←Add 0.5
39.0	$X\ X\ X\ X$	←Result
39		←Truncate

The *sticky bit* is the *OR* function of all the bits we want to check.

The *round digit* is the *MSD* of the discarded part.

M.		L	G	R	S

← desired precision →

Shall we normalize then round?

Even in decimal, if you have leading zeros and there are digits to discard then: *Yes, shift to the left first.*

Consider binary with a possibility of a single position shifting:

Left shift: S does not participate but G is shifted into the number and R into the old position of G.

Right shift: S and R guard bits are ORed into S (i.e., $L \to G$ and $G + R + S \to S$).

1.		L	G	S
← desired precision →			a	

The proper action to obtain unbiased rounding-to-even (RNE) is:

L	G	S	Action	a
X	0	0	Exact result, no action.	0
X	0	1	Inexact result, but no action needed.	0
0	1	0	Tie with even significand, no action.	0
1	1	0	Tie with odd significand, round to nearest even.	1
X	1	1	Round to nearest by adding 1.	1

The sticky is important

Example 1 Let us see the importance of the sticky bit to Directed Upward Rounding when we round to the integer in the following two cases.

Case 1: No sticky bit is used;

38.00001 → 38

38.00000 → 38

Case 2: Sticky bit is used:

38.00001 → 39 (sticky bit = 1)

38.00000 → 38 (sticky bit = 0, exact number).

When the sticky bit is one and we neglect using it, the result is incorrect.

Exceptions

The IEEE standard specifies five exceptional conditions that may arise during an arithmetic operation:

1. invalid operation, ($\infty - \infty$, $\infty \times 0$, $\sqrt{-3}, \ldots$)
2. division by zero,
3. overflow,
4. underflow, and
5. inexact result.

Overflow and infinities

The overflow flag is raised whenever the magnitude of what would be the result exceeds **max** in the destination format.

In default exception handling, the rounding mode and the sign of the intermediate result determine the final result:

	RNE	RNA	RZ	RP	RM
+ve	$+\infty$	$+\infty$	**+max**	$+\infty$	**+max**
-ve	$-\infty$	$-\infty$	**-max**	**-max**	$-\infty$

Furthermore, under default exception handling for overflow, the overflow flag shall be raised and the inexact exception shall be signaled.

Gradual underflow

The gradual underflow preserves an important mathematical property: if M is the set of representable numbers according to the standard then

$$\forall x, y \in M, \qquad x - y = 0 \Longleftrightarrow x = y.$$

Example 2 Assume that a system uses the single precision format of IEEE but without denormalized numbers. In such a system, what is the result of $1.0 \times 2^{-120} - 1.1111 \cdots 1 \times 2^{-121}$?
Solution: The exact result is obviously

$$\begin{array}{rll} & 1.000 \cdots 0 & \times 2^{-120} \\ - & 0.111 \cdots 1|1 & \times 2^{-120} \\ \hline & 0.000 \cdots 0|1 & \times 2^{-120} = 2^{-144} \end{array}$$

which is not representable in this system. Hence the returned result is zero although the two numbers are not equal.

Going for the speed: Cray

As before, the format ($\beta = 2$) consists of sign bit, biased exponent and fraction (mantissa):

S	E	F
1	$\leftarrow$ 15 $\rightarrow$	$\leftarrow$ 48 $\rightarrow$

where

$$\begin{aligned} S &= \text{sign bit of fraction} \\ E &= \text{biased exponent} \\ F &= \text{fraction} \end{aligned}$$

then

$$\begin{aligned} e &= \text{true exponent} = E\text{-bias} \\ f &= \text{true mantissa} = 0.\text{F} \end{aligned}$$

A normalized nonzero number X is

$$X = (-1)^S \times 2^{E-bias} \times (0.F)$$

with a bias $= 2^{14} = 16384$.

Cray: overflow and underflow

- $\mathbf{max} = 2^{2^{13}-1}(1-2^{-48}) = 2^{8191}(1-2^{-48})$
- Any result with an exponent containing two leading ones indicates overflow.
- $\mathbf{min} = 2^{-(2^{13})} \cdot 2^{-1} = 2^{-8193}$
- Any result with an exponent containing two leading zeros indicates underflow. (Flush to zero)
- Testing for over and underflow is done *before* normalization.
- Inputs are *not* tested.

Penalty for speeding

A number $s < \mathbf{min}$ can participate in computations:

- $(\mathbf{min} + s) - \mathbf{min} = s$, where s is 2^{-2} to 2^{-48} times $\mathbf{min}$, since $\mathbf{min} + s > \mathbf{min}$ *before postnormalization*.

  The machine normalizes such results producing a number up to 2^{-48} smaller than $\mathbf{min}$. This number is not set to zero.
- $s \times Y = 0$ if the exponent of Y is not positive enough to bring $\exp(s) + \exp(Y)$ into range.
- $s \times Y = s \times Y$ if $\exp(s) + \exp(Y) \geq \exp(\mathbf{min})$.

Does it really matter?

- In 3D graphics animation, an error in a few pixels in a frame that flashes on the screen is tolerable.
- In general, audio and video signal processing tolerates a number of errors.
- However, if fast and inaccurate results are delivered in scientific or financial computations catastrophes might occur.

Looking back

- Comparison of the different systems
- Rounding
- Is $\frac{1}{3} \times 3 = 1$?
- Does $(x - y = 0) \Rightarrow (x = y)$?
- Penalty for speeding!