A Floating-Point Standard for Balanced Ternary

Aresh Pourkavoos

June 5, 2022

Balanced ternary is a base 3 number system, i.e. numbers in balanced ternary are written as a string of digits whose contributions to the number are scaled by powers of 3. However, rather than the usual 0, 1, and 2, the digits of balanced ternary are -1, 0, and 1. -1 is usually written as T, as it looks like a 1 with a bar or negative sign on top. It can represent negative integers without the use of a negative sign, e.g. T1 $= (-1) \times 3 + 1 \times 1 = -2$.

In the early days of computers, this system was considered as an alternative to binary, and many balanced ternary computers were built, most notably the Setun in Russia in the early 1960s. A brief side-by-side comparison follows. First of all, since there is no real distinction between positive and negative integers, there is no need for additional circuits to handle each case. However, the same is also true for all modern binary computers thanks to the two's complement implementation of signed integers, which takes advantage of the fact that logic gates on a finite number of bits behaves like modular arithmetic. In two's complement, the larger half of what would be unsigned integers, e.g. 8 through 15 (inclusive) for 4-bit integers, are reassigned to negative numbers (-8 through -1 in this case). So in this case, the advantage evens out.

Next, addition and multiplication in balanced ternary both rely on their respective tables for a single digit. In binary, adding two bits requires a carryover $\frac{1}{4}$ of the time, namely when performing $1 + 1$. In balanced ternary, however, this occurrs only $\frac{2}{9}$ of the time, where $1 + 1 = 1T$ and $T + T = T1$, giving it a marginal advantage.

The efficiency of a base in representing numbers can be calculated using radix economy, which measures the limiting behavior of the length of large numbers stored in base b, where each digit is b units long, as if it were stored by a one-hot vector of bits. In base b, the radix economy is given by $\frac{b}{\ln(b)}$, and a lower radix economy is more efficient. Binary and quaternary (base 4) both have a radix economy of approximately 2.89, whereas base 3's is 2.73 - again, a slim advantage over binary. However, the practical usefulness of radix economy is debated, since it makes assumptions about the distribution of numbers that computers typically store which may not be in accordance with reality.

Where I believe balanced ternary shines is in how it handles floating point numbers.

Float f comprised of exponent e (int) and significand s (-1.5 to 1.5)
If s starts with 1 or T, $f = 3^e \times s$ (normal)
Else if $e = e_{min}$ (all Ts), f is denormal (see formula for normal)
Else, $f =$ sNaN, qNaN, or $\pm\infty$ (TBD when operations are implemented)
0 is unique, no ± 0 like in binary floats
Example with 1 trit for e and 2 trits for s:

Memory	TTT	TT0	TT1	T0T	T00	T01	T1T	T10	T11
Formula	$10^{\mathrm{T}} \times$ T.T	$10^{\mathrm{T}} \times$ T.0	$10^{\mathrm{T}} \times$ T.1	$10^{\mathrm{T}} \times$ 0.T	$10^{\mathrm{T}} \times$ 0.0	$10^{\mathrm{T}} \times$ 0.1	$10^{\mathrm{T}} \times$ 1.T	$10^{\mathrm{T}} \times$ 1.0	$10^{\mathrm{T}} \times$ 1.1
Result	.TT	.T0	.T1	.0T	.00	.01	.1T	.10	.11
Decimal	$-4/9$	$-1/3$	$-2/9$	$-1/9$	0	1/9	2/9	1/3	4/9
Memory	0TT	0T0	0T1	00T	000	001	01T	010	011
Formula	$10^0 \times$ T.T	$10^0 \times$ T.0	$10^0 \times$ T.1	?	?	?	$10^0 \times$ 1.T	$10^0 \times$ 1.0	$10^0 \times$ 1.1
Result	T.T	T.0	T.1	?	?	?	1.T	1.0	1.1
Decimal	$-4/3$	-1	$-2/3$	?	?	?	2/3	1	4/3
Memory	1TT	1T0	1T1	10T	100	101	11T	110	111
Formula	$10^1 \times$ T.T	$10^1 \times$ T.0	$10^1 \times$ T.1	?	?	?	$10^1 \times$ 1.T	$10^1 \times$ 1.0	$10^1 \times$ 1.1
Result	TT.	T0.	T1.	?	?	?	1T.	10.	11.
Decimal	-4	-3	-2	?	?	?	2	3	4

Note that in order to avoid a special case for denormal numbers, this format effectively wastes a third of possible code words. It is possible to do the same for binary floats, storing the extra 1 in the mantissa which is usually omitted and changing it to 0 for denormal numbers, but the IEEE decided against it.