28 Jan '57

THE STRETCH NUMBER SYSTEM

The basic word length and word format assigned to a calculator impose certain limits on the range and precision of the numbers which the machine is to handle. The effect of these limits can be minimized by expressing the numbers in floating point form and by performing the four elementary operations in this mode. In addition, both the range and the precision can be further increased by, respectively, scaling operations and expansion (multiple precision) techniques. The Stretch calculator must have these general capabilities.

We assume, in what follows, that 55 binary bits suffice for representing a finite non-zero number (e,m), and that a 9-bit exponent to the base 2 (sign σ,e) coupled with a 46-bit mantissa (sign s, m) represents a good balance between range and precision. This number format allows for up to 13.5 significant decimal digits in m, $0 < m < 1$, and, since $|e| < E = 256$, for a large range, represented approximately by $(10^{-77}, 10^{77})$. If m is in the range (1/2,1) then (e,m) is said to be normalized.

The limits discussed above lead, at times, to the generation of so-called exceptions, i.e., to the appearance of 0-like and ∞-like quantities. Specifically, exceptions arise: (1) from complete mantissa cancellation on addition of truncated numbers, (2) when e becomes large and negative (underflow), $e \leq -E$, (3) when e becomes large and positive (overflow), $e \geq E$, and (4) by division by a zero-like quantity. It is convenient to identify the results of (1) and (2) with a signless mathematical zero, and the results of (3) and (4) with a signless symbol for infinity. This means, of course, that the cancellation of exact numbers and the division by an exact zero are thrown in with the exceptions. This is regarded as acceptable.

Since arithmetic with exceptions, no matter how these and the associated arithmetic are defined, may lead to nonsense results, it is important that the programmer be warned of the generation of exceptions and be provided with means for taking the steps he thinks proper if such events take place.

We, therefore, suggest that a trigger be associated with each exception and that, if an exception is generated, the corresponding trigger is set. The status of the triggers shall be subject to test by instructions of the transfer type. At the programmer's option, such instructions may alter the triggers tested, i.e., reverse, set or reset the triggers.

In addition, a break-in register with bits corresponding to the triggers is recommended, so that the program can exit automatically to a fixed location (one for each exception case) if an exception is generated and if the corresponding bit in the break-in register is present. In such cases, no triggers are set but the location of the instruction which generated the exception is transferred to a special register.

We now define arithmetic with the symbols 0, ∞, and x where x is an arbitrary finite non-zero number. The tables given below suffice for the purpose if subtraction is regarded as the addition of a negative number and division as the multiplication by the reciprocal. For this purpose we also let ∞ be the reciprocal of 0 and visa versa.

Addition

	0	x	∞
0	0	x	∞
x	x	x	∞
∞	∞	∞	∞

Multiplication

	0	x	∞
0	0	0	∞
x	0	x	∞
∞	∞	∞	∞

If one of the operands is 0 or ∞, the machine should, of course, not go through the complicated steps involved when both operands are x's, it should, in other words, be able to recognize exceptions. We propose, therefore, that the generation of an exception sets m to zero and s to either zero (for zero exceptions) or to unity (for ∞ exceptions). As an alternative to setting m = 0, a special i-bit (normally unity) may be set to zero.

Schematically, the word format (60 bits) now proposed looks as follows:

Quantity→	i	s	σ	e	m	p
Bits ⟶	1	1	1	8	45	4

where p represents bits yet to be assigned. If, for some reason it seems desirable to have a 64-bit word, one could, for example, increase m by three and p by one.

To enable the programmer to scale numbers before the allowed range has been exceeded we require break-in facilities similar to those described above for exceptions if (5) $e < -E'$, and (6) $e > E'$, where E' is an arbitrary exponent, $E' < E$, specified by the program.