Harwood

FLOATING POINT DATA WORD FORMAT (STRETCH)

SCHEME II

E. A. Voorhees (11/30/56)

INTRODUCTION

After the development and presentation of Scheme I, it was felt by a considerable number of experienced programmers that it was possible to design a much simpler scheme- preferably one without I's or e's. It is the purpose of this paper to try to make such a scheme specific.

CHARACTERISTICS OF SCHEME II.

1. Scheme is designed for normalized and unnormalized modes of operation.

2. Exponent overflows and underflows can occur only when at least one of the operand mantissas is non-zero.

3. Let Q represent any floating point number with a zero mantissa. (It may be that Q should be further restricted.)

4. Let N represent any floating point number with a non-zero mantissa.

5. If one or both of the operands is Q, the primary operation is not performed. Again, it may be desirable to perform result normalization (if normalized operation) as in N + Q. This is done in this write-up.

6. Operand (argument) mantissas are assumed in the following [redacted] to be either normalized or unnormalized.

7. Each arithmetic instruction contains a bit called the Break-in bit (BI). This bit is relevant only on exponent underflow situations (described later). Assume that the absence of BI means that Break-in does not take place, i.e. is usually ignored.

8. Each Load or Store (relevant to arithmetic registers) has a Reset bit, (R), instead of the corresponding BI bit. The presence of an R bit resets the three triggers (described in 10 below) and clears the "Instruction Location Preserve Register" (described in 9 below). If the R bit is not present in the instruction, neither of these events take place.

9. It is assumed that the machine contains an "Instruction Location Preserve Register", (ILPR). This register is self-explanatory and the conditions for its being set are indicated later in the paper. This register contains the last "interesting" location (or possibly no location).

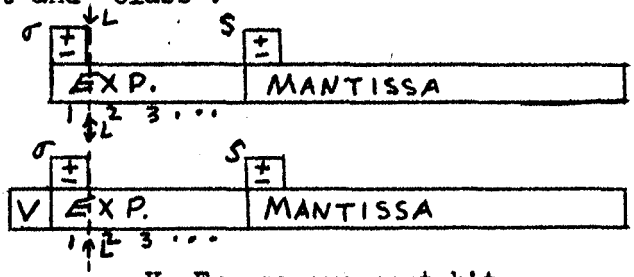
10. The machine is assumed to have three triggers:
    a. "Hi-Ov" : Indicates high order exponent overflow.
    b. "Hi-Un" : Indicates high order exponent underflow.
    c. "ZD" : Indicates division by zero. In this respect the division, N/N, is always assumed to take place regardless of whether or not there are more lead zeroes in the divisor than in the dividend.

CHARACTERISTICS OF SCHEME II (Continued).

11. It is assumed that there are two types of Division operations (both having BI provisions) to allow the programmer either intervention or no intervention due to attempted division by zero (Q). Let us call the first type "Divide or Transfer" and the second type "Divide or Bypass". In "Divide or Transfer", the transfer is assumed to be to FM 3 (Fast Memory, location 3), and the "ZD" trigger is not set. In "Divide or Bypass", the "ZD" trigger is set (when appropriate) and the program continues to the next instruction. In all zero division cases the dividend is left as the result in the A and B registers, i.e. "N/Q = N" and "Q/Q = Q".

12. Assume (for convenience) that signed exponents are used.

13. Format and "Class":



V: Excess exponent bit.

(The sign of the exponent distinguishes exponent overflow from exponent underflow.)

Let the dotted line, L (for limit), between bits 1 and 2 (for the time being) represent the upper exponent boundary. When L is "crossed" due to exponent underflow or overflow, the event will be designated as a " 1st Class" underflow or overflow. The result of an operation on ordinary arguments which produces a 1st class event can be completely represented in a data word. Triggering, Break-ins, etc. which are performed are done to provide advance warning.

Exponent carries from bit 1 to bit V are referred to as "2nd Class" exponent underflows and overflows. In this case more drastic action is called for.

TYPES OF EXPONENT EVENTS

Exponent Overflow (Note: Whenever low order overflow occurs, then high order must also have exponent overflow and hence high order exponents only are considered.)

A. 1st Class Exponent Overflow (+ exponent).
"Hi-Ov" trigger is set. Trigger is examined by separate instruction such as "Transfer on Hi-Ov" and not by BI (Break-in) in the offending instruction. (BI bit is ignored.) The ILPR is set.

B. 2nd Class Exponent Overflow (+ exponent).
The machine does an immediate and mandatory transfer to some fixed memory location, say FM 1. No triggers are set. (BI bit is ignored.) The ILPR is set.

TYPES OF EXPONENT EVENTS (Continued)

Exponent Underflow

A. 1st Class Underflow (- exponent).
The BI bit of the instruction is examined and if 1st Class exponent underflow occurred during the execution of the instruction, Break-in to FM 2 takes place. No trigger (namely "Hi-Un" trigger) is set and no clearing of the high order result takes place. The ILPR is set in either case.

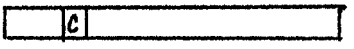
B. 2nd Class Underflow (- exponent).
The number is cleared and "Hi-Un" trigger is set. Trigger is examined by "Transfer on Hi-Un" instruction and not by BI which only applies to 1st Class underflows at time of formation. The ILPR is set.

Clearing of low order underflows is done and there is no setting of triggers, no BI's, no setting of ILPR, etc.

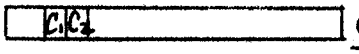
RANDOM OBSERVATIONS

1. Possibly some type of "internal switch" could be designed to allow positioning of L instead of having it rigidly fixed between exponent bits 1 and 2, "L = # Mode".

2. Cancellation indication could be an additional bit (or two) in data word.

1 bit: 

c = 0: Less than $\frac{1}{2}$ word cancellation.
c = 1 More than $\frac{1}{2}$ word.

2 bits: 

c_1	c_2	
0	0	: Less than $\frac{1}{4}$ word.
0	1	: More than $\frac{1}{4}$, less than $\frac{1}{2}$.
1	0	: More than $\frac{1}{2}$, less than 3/4.
1	1	: More than 3/4.

(Would Break-in techniques be desirable--more bits from instruction-?)

3. Should an internal switch be provided to halve "effective" mantissa lengths to provide unused bits to be used by the programmer in any way he desires and which are not disturbed or modified by arithmetic operations? (Would this be more flexible if done in quarters?)

4. The operands are always assumed to be in the S-register and the A',B' registers after the execution of any operation.

5. The result of an operation involving operands, both of which are Q's, is assumed to produce a Q whose exponent is computed by the usual rules for exponents.