H. Kolsky

October 10, 1956

STRETCH Floating Point Operations

by Sweeney

The following paper is a description of a set of floating point operations for the senior machine. Some, but probably not all, of these operations will be necessary in the input-output computer.

The general design objectives are:

1. Automatic sign control.
2. Preserve the maximum number of data bits.
3. All exception cases are capable of causing automatic transfers without programmed testing.
4. Manipulation commands allow testing of any part of floating point numbers.
5. Facilities are provided so that multiple precision operations can be programmed with a minimum of effort.

Floating point numbers will be defined as consisting of a signed twelve bit exponent and a signed forty-eight bit mantissa. For comparison purposes, the relative importance of the information is mantissa sign, exponent sign, exponent, and mantissa. Floating point numbers will be represented as $\pm x, \pm y$ where x is the exponent in the range $-X \leq x \leq X$ and y is the mantissa in the range $1/2 \leq y < 1$. The integer, 1, therefore, will be represented as +1, + 1/2. In the case of unnormalised numbers, the mantissa can be in the range $0 \leq y < 1$. The largest possible number (LPN) is +X, $\pm$(1) where (1) means a string of 48 ones in the mantissa. The smallest possible normalized number (SPN) is -X, $\pm$ 1/2. Floating point zero (FPO) is -X, + 0. Other zero mantissa cases ($\pm$ 0) may occur as the result of operations and these may be preserved.

The assumption is that there will be three registers, S, A, and B, where S is the register which holds all information flowing to or from the main memory bus. A and B are result registers whcih hold the final result of any operation. Two more registers may be necessary if all operands are to be preserved until after an operation is executed.

It is assumed that there are data addresses (or indicators) such that the previous contents of S, A, or B can be the operand in S for this operation. It is also assumed that there will be an additional address part such that

any number appearing in S can be stored in any one of the transistor registers as well as used as the operand in this instruction.

Define a sign manipulator, m, which can be part of any operation and can have four states as follows:

1. Use sign
2. Invert sign
3. Set sign plus
4. Set sign minus

The sign manipulator will operate on the sign of S, A, or B as the factor is used in the operation, but the actual sign in the register will not be disturbed so that the factor with its original sign may be used in subsequent operations.

Define a normalization operator, N, having two states as follows:

1. n means normalize the result.
2. $\tilde{n}$ means do not normalize the result.

In all operations only the registers specified will be changed. Any register not specified will be undisturbed. Denotes the contents of the location specified by the data address (assume the possibility of S, A, or B)

Transfer and manipulation operations -

Load A	D → S, mS → A
Load A normal	D → S, FPO → B, mS → NA
Load B	D → S, mS → B
Normal AB	m (AB) → N (AB)
Normal AB zero	m (AB) → N (AB), FPO → B
Reset B	FPO → B
Reset AB	FPO → (AB)
Store A	A → S, mS → D
Store A Reset	A → S, mS → D, FPO → A and B
Store FPO	FPO → S, S → D
Store B	B → S, mS → D
Store and Reset B	B → S, mS → D, FPO → B
Store B Reset	B → S, mS → D, FPO → A and B

Add operations -

Add A	D → S, FPO → B, mS + A → N (AB)
Subtract A	D → S, FPO → B, mS - A → N (AB)
Add D	D → S, FPO → B, mA + S → N (AB)
Subtract D	D → S, FPO → B, mA - S → N (AB)
Add AB	D → S, mS + (AB) → N (AB)
Subtract AB	D → S, mS - (AB) → N (AB)
Add D to AB	D → S, m (AB) + D → N (AB)
Subtract D from AB	D → S, m (AB) - D → N (AB)

(Note that certain sign operations are redundant, these can be culled later)

Automatic break-in can be specified for the following cases, but if break-in is not specified, the machine definition is in the column on the right.

1.	Exponent overflow A	LPN→ A (maintain sign), FPO→ B.
2.	Exponent underflow A	SPN→ A (maintain sign), FPO→ B.
3.	Exponent underflow B	FPO→ B.
4.	Zero mantissa A	If n, normalize bits from B if specified.
5.	Zero mantissa AB	If n, set exponent A to (x-96) and exponent B to x of A-144. If $\tilde{n}$, do not change A or B.

The results in A and B will have the same sign. If the operation specifies n, the mantissa of A will be normalized to the range $1/2 \leq y < 1$, the exponent of B will be set to x of A -48 (this implies a mantissa in B not necessarily normal). If S, A, or B is detected to contain FPO before the execution of the operation, it will not be considered as a factor in the operation. If the operation specifies n and either A or S contains a zero mantissa before the operation, its exponent will not be used to determine a shift amount. If both S and A contain zero mantissas, no operation will be performed but post operations such as break-in and normalization will take place. If the operation specifies $\tilde{n}$ zero mantissas (not FPO) will be treated as if the mantissa were not zero.

Multiply operations -

Multiply D	D→S, mS·A→N (AB)
Multiply A	D→S, mA·S→N (AB)

The prior contents of B will not affect the resultant product. Normalization of only one leading zero is possible if the operation specifies n. If the mantissa of S or A is zero, the result will be zero but the normal rules of exponent addition and normalization will apply.

Break-in cases are as follows:

1.	Exponent overflow A	LPN→ A (maintain sign), FPO→B.
2.	Exponent underflow A	SPN→ A (maintain sign), FPO→B.
3.	Exponent underflow B	FPO→B
4.	Zero Mantissa A	Normalize one place if specified.
5.	Zero Mantissa AB	Normalize one place if specified.

Divide operations -

Divide by D	D→S AB ÷ mS→ NQ in A, FPO in B
Divide by D with R	D→S, AB ÷ mS R in A, NQ in B.
Divide A	D→S, FPO→B, A ÷ mS→NQ in A, FPO in B
Divide D	D→S, FPO→B, S ÷ mA→NQ in A, FPO in B

If the operation specifies n, this implies the generation of a quotient with a leading bit not zero. (This can be achieved by partial or complete pre-normalization of the dividend or by taking extra divide sub-cycles until $1/2 \leq |Q| < 1$. If the operation specifies $\overline{n}$, this implies that if the dividend has p more leading zeros than the divisor, the quotient will have p leading zeros.

Break-in can be specified as follows:

1.	Exponent overflow Q	LPN→ Q, R undisturbed
2.	Exponent underflow Q	SPN→ Q, R undisturbed
3.	Exponent underflow R	FPO→ R, Q undisturbed
4.	Divisor mantissa zero	LPN→ Q, FPO→R
5.	Divisor has more leading zeros than dividend.	LPN→ Q, FPO→R

If the dividend mantissa [S, A, or (AB)] is zero the quotient will be zero but ordinary exponent differencing rules apply.

Special operations -

Square Root D, D→S, $\sqrt{mS}$→A, FPO→B, break-in for mS negative or FPO→A, FPO→B.

Round A, If B contains a normal mantissa add one to the magnitude of mantissa A. FPO→B, NA.

Make exponent of S into a normalized floating point number in A, FPO →B. If exponent S is zero, A would contain the number FPO.

Put mantissa of S (considered as an integer) plus one into the exponent position of A, set mantissa A to 1/2, FPO→B. Break-in for exponent overflow or underflow.

Borrow one from the low order position of mantissa A and replace mantissa B by 1/2 and same sign as A. Set exponent B to x of A-47. If x of A-47 would cause exponent underflow, do not execute set FPO→B.

Interchange exponent S and exponent A.

Interchange mantissa S and mantissa A.

Compare exponent S to exponent A.

Compare mantissa S to mantissa A.

Special operations (continued) -

Compare S to A.

Compare exponent A with limits specified.

Count leading zeros of A and possibly B; do not normalize.

Add or subtract from exponent A, normal break-in rules for exponent overflow or underflow apply.

The set of operations shown here is not complete (in fact, only a few conditional transfers are shown) but will serve as a sound basis for the senior computer floating point system. Other operations may be specified after consultation with Los Alamos. The class of special operations is susceptible to the greatest change as the investigation of the desirability and feasibility of functions such as $\sin x$, $\cos x$, e^x, $\ln x$, $\arctan x$, $\sum a_i b_i$, etc. proceeds.

Operations have been defined for FPO and zero mantissa cases. Further investigation should be made for the feasibility or necessity of defined operations with LPN and SPN to denote infinity operators. (e.g. LPN · X = LPN, X − LPN = SPN, LPN · SPN, etc.)

D. W. Sweeney