Kolsky

Stretch Floating Point Operations

11/9/56

by Dura Sweeney

This paper is a replacement for a previous paper dated 10/10/56 and contains more detailed information about the operations as well as new definitions for exception cases.

The general design objectives are:

1. Automatic sign control.
2. Preservation of the maximum number of information bits.
3. Automatic transfer to subroutines (break-in) or machine generated definitions of exception cases.
4. Facilities are provided to test any condition or part of a floating point number.
5. Facilities are provided so that multiple precision operations can be easily programmed with automatic sign control using the floating point system.

Floating point numbers will be defined as consisting of 64 bits assigned as follows:

1 bit	exponent sign
12 bits	exponent
1 bit	mantissa sign
48 bits	mantissa
2 bits	exception indicators

If bits are required for general assignment by the programmer, the exponent can be cut to 10 bits.

A normalized floating point number will consist of a signed exponent in the range $-2^{12} < e < 2^{12}$ with a signed mantissa in the range $\frac{1}{2} \leq |m| < 1$. A unnormalized floating point number will have the same exponent range but the mantissa can be in the range $0 \leq |m| < 1$.

The two exception indicators will be assigned the symbols I and E if they are on. I denotes a number in which the resultant exponent would have been greater than 2^{12}. E denotes a number in which the resultant exponent would have been less than -2^{12} or in certain cases where the resultant mantissa is zero. The various conditions by which I and E are indicated will be discussed under the various operation codes.

It is assumed that there are five registers associated with the arithmetic unit such that before the execution of any instruction, the previous result and both operands are available. These registers will be labeled S, A, B, A', and B'. S will hold all data flowing to or from the memory. A and B will hold the result of the operation. A' and B' will in general hold the contents of A and B before the operation was executed. It is assumed that S, A, B, A', and B' are addressable.

It is assumed that there is a sign manipulator, M, which can be part of most instructions. It will have four states which are use sign, invert sign, set sign plus, and set sign minus.

It is assumed that there is a normalization operator, N, which can be part of most instructions. It will have two states, normalize or donot normalized (n and $\bar{n}$).

In general all instructions will have the following sequence of events:

1. Data into S.
2. AB into A'B'
3. Execute operation
4. Set results in AB

Store operations imply:

1. Load A with appropriate register
2. Store S in specified location

All additions and multiplications will result in a double length mantissa in AB. The signs of A and B will agree and the exponent of B will always be 48 less than the A exponent except in case of exponent underflow. All division operations will result in a quotient in A and a remainder in B.

Load and Store operations.

M Load A:	1.	Data into S
	2.	AB into A'B'
	3.	MS into A(B undisturbed)
M Load B:	1.	Data into S
	2.	AB into A'B'
	3.	MS into B (A undisturbed)
NM Load AB:	1.	Data into S
	2.	AB into A'B'
	3.	MS into AB (B cleared)
	4.	Normalize AB (See add operations for exception handling.)
M Store S:		MS into data location
M Store A:	1.	A into S
	2.	MS into data location
M Store B:	1.	B into S
	2.	MS into data location.

There will probably be M Store A' and M Store B' operations.

M (Round and Store A)	1.	AB into A'B'
	2.	Round A with respect to high order bit of B. (Follow rules for mantissa overflow and exponent overflow for Add)
	3.	A into S
	4.	MS into data location
Clear AB:	1.	AB into A'B'
	2.	Clear A and B exponents, mantissas, and signs, set E on in A and B.

Store E:

1. Clear S, exponent, mantissa, and signs, set E on in S.
2. S into data location.

Add operations.

NM Add S to A	implies	MS≠A into AB
NM Add A to S	implies	MA≠S into AB
NM Add S to AB	implies	MS≠(AB) into AB
NM Add AB to S	implies	M(AB)≠ S into AB
NM Sub S from A	implies	MA-S into AB
NM Sub A from S	implies	MS-A into AB
NM Sub S from AB	implies	M(AB)-S into AB
NM Sub AB from S	implies	MS-(AB) into AB

If either or both operands is I the result will be I. If either operand is E the result will be the other operand. If both operands are E the result will be E.

If neither I or E in the operands the following sequence of events takes place.

1. Data into S.
2. AB into A'B'
3. Find difference of the exponents of S and A'
4. Enter the mantissa (or mantissas) with the smaller exponent into AB shifted right (0 to 95 positions) by the exponent difference. A and B signs will be determined by the original sign, M, and the operation.
5. Add the number with the larger exponent to A. Its sign will be determined by the original sign, M, and the operation. Set A exponent to the larger exponent value. The signs of A and B mantissas are the same and will be the final result signs.
6. If A mantissa overflows, shift AB right one place, insert one in high under A mantissa, add one to A exponent.
   If exponent overflows, clear AB, set I in A and B.
   If exponent in range, set B exponent to A exponent minus 48.
   If B exponent underflows, clear B set E in B.

or 6. If the operation specifies $\overline{n}$, set B exponent to A exponent minus 48. If B exponent underflow, clear B, set E in B.

or 6. If the operation specifies n, shift AB mantissas left (0 to 95 positions) until there is a bit in the high order of A mantissa. Subtract the amount of shift from A exponent.
   If A exponent underflows, clear AB and set E in A and B.
   If A exponent in range, set B exponent to A exponent minus 48.
   If B exponent underflows clear B, set E in B.
   If both A and B mantissas are zero test break-in for zero mantissa, if no break-in clear AB, set E in A and B.

7. Test break-in for I or E.

Multiplication

NM mply A·S	implies	MA.S into AB
NM mply S·A	implies	MS·A into AB

If I in either or both operands the result is I. If E in either or both operands the result is E. (If I·E the result is I)

If neither I nor E in the operands the following sequence of events takes place:

1. Data into S
2. AB into A'B'
3. Multiply A' times S placing the product in AB. The signs will be a combination of the original signs of A' and S and M.
4. Set A exponent to A' exponent plus S exponent.
5. If A exponent overflows, clear AB, set I in A and B.

or 5. If A exponent underflows, clear AB, set E in A and B.

or 5. If the operation specifies N, set B exponent to A exponent minus 48. If B exponent underflows, clear B and set E in B.

or 5. If the operation specifies n, normalize one leading zero in A if necessary. (follow above rule if A exponent underflows) Set B exponent to A exponent minus 48. If B exponent underflows clear B, set E in B.

6. Test break-in if I or E in A.

Divide Operations

M Div AB	implies	M(AB)÷S
M Div A	implies	MA÷S
M Div S	implies	MS÷A

If I (or E) in dividend, quotient is I (or E) and remainder is I (or E). If I (or E) in divisor quotient is E (or I) and remainder is E (or I). If neither I nor E in the operands the following sequence takes place.

1. Data into S
2. AB into A'B'
3. Divide M (X) by (Y) putting quotient in A and remainder in B.
4. Set A exponent to (X) - (Y) exponents, set B exponent to (X) exponent minus 48.
5. If A exponent overflows, clear A set I in A.

or 5. If A exponent underflows, clear A, set E in A.

or 5. If B exponent underflows, clear B, set E in B.

6. Test break-in is I or E in A

Special operations

Square root.
Round
Normalize
Exponent into Floating Point number
Mantissa into exponent
Borrow

Interchange exponents
Interchange mantissas
Set exponent
Set mantissa
Compare exponents
Compare mantissas
Compare floating point numbers
Compare exponent against limits
Count mantissa zeros
Add to exponent
Add to mantissa
Test mantissa sign
Test exponent sign
Test mantissa zero
Test exponent overflow
Test exponent underflow