FLOATING POINT ARITHMETIC

The purpose of the floating point instruction set is to perform calculations using data with a wide range of magnitude and yielding results scaled to preserve precision.

Floating point numbers consist of an exponent E and a fraction F. The quantity expressed by this number is the product of the fraction and the number 2 raised to the power of the exponent, that is:

$$\text{value} = \pm F \times 2^{E}$$

Instruction Format

Most floating point instructions have the following format:

where j and k designate the arithmetic registers containing the source operands and i specifies the result register(s). The remaining floating point operations ignore the k field of the instruction.

The arithmetic registers containing the source operands are not changed as a result of floating point instructions unless they are also specified by the i field to be result registers.

Number Representation

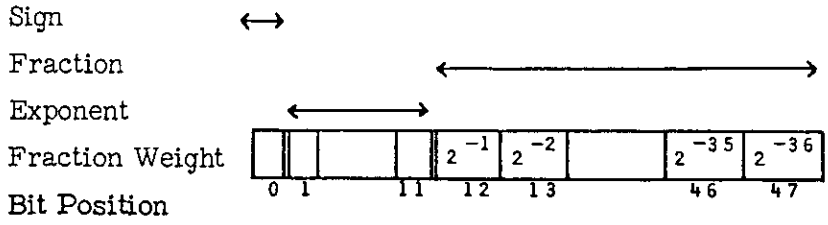
Floating point numbers may be in 48 bit single precision form or in 96 bit double precision form. Single precision numbers may occupy any of the arithmetic registers. Double precision numbers may occupy any even-odd pair of arithmetic registers. Arithmetic register A^0 is specified to be a source of 0's. When A^0 is specified as a source operand, 48 or 96 0's will be provided depending on whether a single or double precision operand was called for by the instruction. If A^0 is specified as the result register, the result will be lost, and the only effect of the operation will be a possible change in the exception register.

Whenever a double precision number is specified by an instruction, the value of the i-, j-, or k-field (as appropriate) is assumed to be even. If it is not, the low order bit of the field is forced to 0, exception bit RS is set, and the operation proceeds. Thus fifteen non-zero double precision quantities are specifiable; namely, the data in register pairs specified by $2, 4, 6, \ldots, 30$. Note that register A^1 is not the low order half of any double precision quantity.

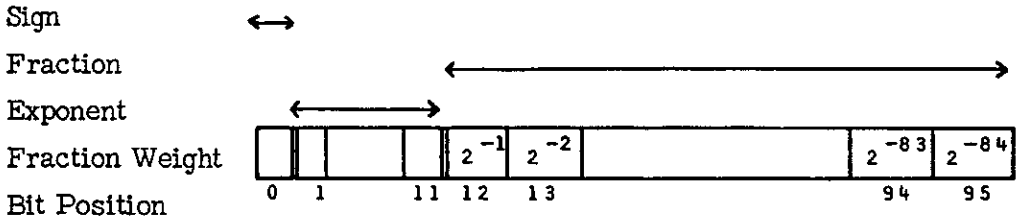
The representation of a floating point number consists of a one bit sign field, an 11 bit exponent field, and a 36 bit or 84 bit fraction field. The sign field occupies positions 0 of the floating point number. A 0 signifies a positive fraction and a 1 a negative fraction. The exponent field occupies positions 1 through 11. The exponent field contains the sum of the exponent, E, and the number 1024, the bias value. The fraction field occupies positions 12 through 47 for a single precision number, and positions 12 through 95 for a double precision number.

Pictorially, with arithmetic bit weights given for the fraction, the formats are:

48 bit floating point number



96 bit floating point number



Number Range

The range of exponents which can be represented is +1023 through -1024. Whenever a floating point operation results in an exponent which cannot be represented in this range, an exception condition exists. If the exponent exceeds +1023, the condition is called overflow. If the exponent is less than -1024, the condition is called underflow. An appropriate exception bit is set to 1 on occurrence of these exceptional conditions.

Figure 3.1 illustrates the range of normalized results possible from a single precision floating point operation. The range of result magnitude is approximately 2.8×10^{-309} to 9.0×10^{307}.

Numbers in the exponent overflow range have the properties of undefined numbers and will be symbolized by u. Furthermore, results in this overflow range are changed to a specific bit configuration: bit zero is set to 1 and the remaining bits are set to 0.

Numbers in the exponent underflow range may be considered to have the properties of the number zero, and are symbolized by 0. Hence, results in the exponent underflow range are changed to the bit configuration which is all 0's.

Since the bit configuration of a leading 1 followed by all 0's represents the u range, special definition of arithmetic using this configuration as an operand is necessary. Figure 3.2 summarizes the results for initial operands as specified. N represents a non-zero value in the valid number range. N* represents a result which is normally in the N range but may be in the u or 0 ranges due to exponent overflow or underflow. Results when using these operands in the compare and sign change operations are given in the sections of this manual dealing with those instructions.

In order to inform the programmer when results approach to the limits of representability, there are overflow and underflow warning bits. These bits are set when a machine with a ten bit exponent would overflow or underflow. The resulting operand is not affected by the setting of these exception bits.

Normalization

A quantity can be represented with the greatest precision by a floating point number, with a given fraction length, when that number is normalized. A floating point number is normalized when it is zero or when $1/2 \leq |F| < 1$. Thus, a non-zero floating point number is normalized if bit 12 (the high order fraction bit) is a 1. The process or normalization consists of shifting the fraction left until the high order fraction bit is a 1, and reducing the exponent by the number of bits shifted.

Instructions are provided which allow floating point arithmetic to be performed with either normalized or unnormalized results. The normalized addition and subtraction instructions yield a normalized result regardless of whether or not the input operands were normalized. The normalized multiplication instructions only guarantee a normalized result if both input operands were normalized. Division may not be done with an unnormalized divisor and does not guarantee a normalized result if the dividend is not normalized.

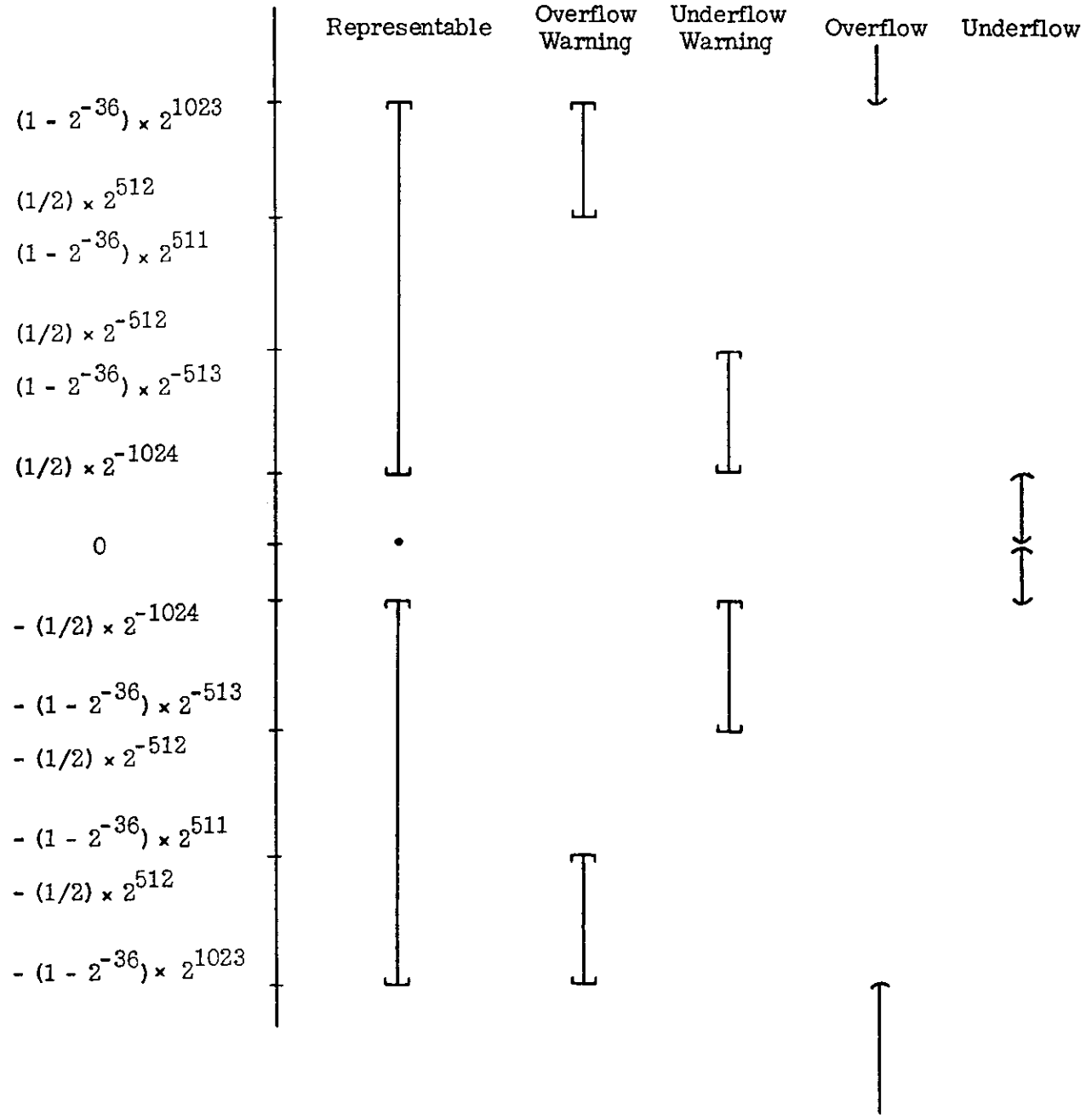


Figure 3.1. Range of Normalized Numbers

Addition Table

Augend	Addend O	Addend N	Addend u
O	O	N	u
N	N	N*	u
u	u	u	u

Multiplication Table

Multiplier	Multiplicand O	Multiplicand N	Multiplicand u
O	O	O	u
N	O	N*	u
u	u	u	u

Division Table

Divisor	Dividend O	Dividend N	Dividend u
O	u	u	u
N	O	N*	u
u	u	u	u

Figure 3.2. Result Ranges

Truncation and Rounding

During the execution of floating point arithmetic operations, low order bits may be truncated in intermediate calculations or when placing the result in the result register. In single precision arithmetic the intermediate result fraction is truncated with the high order 37 bits retained. The low order bit of the retained information is called the guard bit. Normalization, if specified by the instruction, takes place by shifting the intermediate fraction including the guard bit. Following normalization the fraction is truncated to 36 bits and placed in the result register. Double precision normalized arithmetic is similar except the intermediate fraction is truncated with 85 bits retained, and the final result is truncated to 84 bits.

Normalized floating point arithmetic instructions are also provided which specify the result fraction to be statistically rounded. If any of the bits truncated, during either of the truncation processes described above, were a 1, the low order bit of the fraction field is forced to a 1.

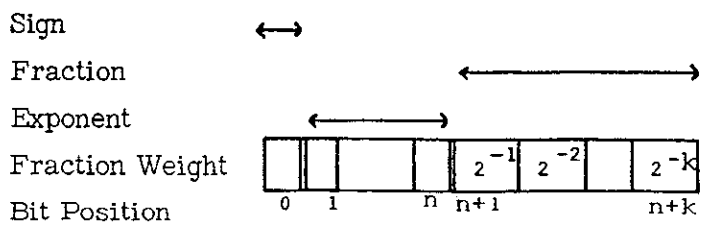
Rounding does not take place if a zero fraction, exponent overflow, or exponent underflow exception occurred.

Low Significance and Zero Fraction

Addition and subtraction may cause a loss of significant bits in cases where the operands are of nearly the same magnitude and differ in sign or when the operands have large numbers of leading 0's, so that the result, before normalization, has a large number of leading 0's. A warning is given when this occurs by setting a "low significance" exception bit to 1. The bit is set to 1 for both single and double precision operations when the leading 1 bit of the result fraction, prior to normalization, is in one of the eight least significant bit positions or is in the guard bit. The low significance exception bit is not set to 1 when the result fraction is all 0's. For this occurrence, a "zero fraction" exception bit is set to 1.

Short Word Floating Point Format

Special provision is made for allowing floating point numbers to be packed into less than 48 bits. In the short floating point format, the sign bit occupies the leading bit position, the exponent field the next n positions (where n is between one and eleven), and the fraction field the remaining positions. The following diagram illustrates this format:



Quantities in this short format are not acceptable operands for arithmetic operations and must be converted to the standard 48 bit format before use. Two special instructions CVF and CVS are provided for converting from short word to full word floating point and vice versa. These instructions which contract and expand the exponent will be described further in the instruction section.

In the short format the exponent, sign, and fraction fields have the same interpretation as in the long format except that the exponent is stored with a bias of 2^{n-1} in the short format (in the regular format, the bias is 2^{10}).

Operation Summary

The following table summarizes the floating point arithmetic instructions:

	Add	Subtract	Multiply	Divide
Normalized, Truncated				
Single	AN	SN	MN	DN
Double	ADN	SDN	MDN	DDN
Mixed	-	-	MMN	DMN
Normalized, Rounded				
Single	AR	SR	MR	DR
Double	ADR	SDR	MDR	DDR
Mixed	-	-	-	DMR
Unnormalized, Truncated				
Single	AU	SU	MU	-
Double	ADU	SDU	MDU	-
Mixed	-	-	MMU	-

Addition and Subtraction Instructions

All the addition and subtraction operations are performed as follows:

The notation 37/85 indicates 37 in the case where the result is to be in single precision and 85 in the case where the result is to be in double precision.

1. If either operand is a u, the result is set to u and the remaining steps of the operations are omitted.

2. If either operand is 48/96 0's, step 3 is omitted. If both of the operands are 48/96 0's, the result is set to 48/96 0's and the remaining steps are omitted.

3. The exponents of the operands in A^j and A^k are compared and the fraction of the operand with the smaller exponent is shifted right a number of positions equal to the difference in exponents. 0's are inserted in the high order vacated bits.

4. If the instruction specifies subtraction, the sign of the second operand is changed.

5. A signed fraction addition then takes place with the high order bit of the shifted fraction aligned with the high order bit of the unshifted fraction. Conceptually, addition takes place before any truncation and the intermediate result is then truncated to 37/85 bits. In fact, truncation takes place first, with sufficient information retained from the truncated bits to make the above property hold.

6. The larger of the two operand exponents is taken as the exponent of the intermediate result.

7. If the fraction addition caused a fraction overflow, the intermediate fraction is shifted right one position and a 1 is inserted in the high order position. One is then added to the intermediate exponent. If this causes the exponent to exceed 1023 an exponent overflow has occurred. The result is then set to u, the AO (add overflow) exception bit is set to 1, and the remaining steps are omitted.

8. If normalization is specified and if the 37/85 bit intermediate result fraction was all 0's, the result is set to zero (i. e. 48/96 0's), the ZF (zero fraction) exception bit is set to 1, and the remaining steps are omitted.

9. If normalization is not specified and the 36/84 bit intermediate result fraction was all 0's, the ZF exception bit is set to 1 and step 10 is omitted.

10. If the high order 28/76 bits of the fraction are all 0's the LS (low significance) exception bit is set to 1.

11. If the instruction specified normalization, the intermediate fraction is now normalized. The intermediate fraction is shifted left until its high order bit is a 1. 0's are inserted into the low order vacated bits. The intermediate exponent is decreased by the amount of the shift. If this causes the exponent to be less than -1024, an exponent underflow has occurred. The AU (add underflow) exception bit is set to 1, the result is set to 48/96 0's and the remaining steps are omitted.

12. If the resulting exponent is greater than 511, the OW (overflow warning) exception bit is set to 1. If the resulting exponent is less than -512, the UW (underflow warning) exception bit is set to 1.

13. This intermediate result fraction is truncated to 36/84 bits.

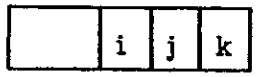
14. If rounding was specified by the operation, the low order bit of the fraction is forced to a 1 if any of the truncated bits, including the bit shifted out in step 7, were a 1.

Add Normalized AN

The contents of register A^i are replaced by the normalized sum of the single precision floating point numbers in A^j and A^k.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF

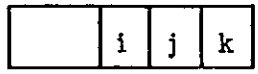
Add Double Normalized ADN



The contents of register pair $A^{i,i+1}$ are replaced by the normalized sum of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

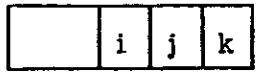
Add Rounded AR



The contents of register A^i are replaced by the normalized and statistically rounded sum of the single precision floating point numbers in A^j and A^k.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF

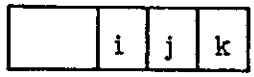
Add Double Rounded ADR



The contents of register pair $A^{i,i+1}$ are replaced by the normalized and statistically rounded sum of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

Add Unnormalized AU



The contents of register A^i are replaced by the unnormalized sum of the single precision floating point numbers in A^j and A^k.

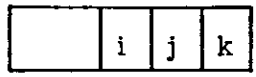
Exceptions	Exception bit
result exponent $>$ +1023	AO
+511 $<$ result exponent $\leq$ +1023	OW
low significance	LS
zero fraction	ZF

Add Double Unnormalized ADU

The contents of register pair $A^{i,i+1}$ are replaced by the unnormalized sum of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent $>$ +1023	AO
+511 $<$ result exponent $\leq$ +1023	OW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

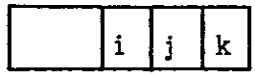
Subtract Normalized SN



The contents of register A^i are replaced by the normalized result of the subtraction of the single precision floating point number in A^k from the single precision floating point number in A^j.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF

Subtract Double Normalized SDN



The contents of register pair $A^{i,i+1}$ are replaced by the normalized result of the subtraction of the double precision floating point number in $A^{k,k+1}$ from the double precision floating point number in $A^{j,j+1}$.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

Subtract Rounded SR

The contents of register A^i are replaced by normalized and statistically rounded result of the subtraction of the single precision floating point number in A^k from single precision floating point number in A^j.

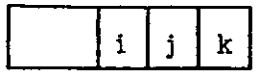
Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF

Subtract Double Rounded SDR

The contents of register pair $A^{i,i+1}$ are replaced by the normalized and statistically rounded result of the subtraction of the double precision floating point number in $A^{k,k+1}$ from the double precision floating point number in $A^{j,j+1}$.

Exceptions	Exception bit
result exponent > +1023	AO
result exponent < -1024	AU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

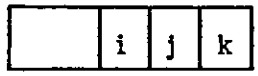
Subtract Unnormalized SU



The contents of register A^i are replaced by the unnormalized result of the subtraction of the single precision floating point number in A^k from the single precision floating point number in A^j.

Exceptions	Exception bit
result exponent > +1023	AO
+511 < result exponent ≤ +1023	OW
low significance	LS
zero fraction	ZF

Subtract Double Unnormalized SDU



The contents of register pair $A^{i,i+1}$ are replaced by the unnormalized result of the subtraction of the double precision floating point number in $A^{k,k+1}$ from the double precision floating point number in $A^{j,j+1}$.

Exceptions	Exception bit
result exponent > +1023	AO
+511 < result exponent ≤ +1023	OW
low significance	LS
zero fraction	ZF
i, j, or k odd	RS

Multiplication Instruction

The multiplication operations are performed as follows:

1. If either operand is u, the result is set to u and the remaining steps of this operation are omitted.

2. If either operand is 48/96 0's, the result is set to zero and the remaining steps are omitted.

3. If either operand is unnormalized and the instruction calls for normalization, the UO is set to 1.

4. The exponents of the operands are added to form an intermediate exponent.

5. The 36/84-bit fraction of the operand in A^j and the 36/84-bit fraction of the operand in A^k are multiplied to form a 72/168-bit product which is then truncated with the high order 37/85 bits being retained.

6. If normalization was called for, the high order bit of the intermediate fraction is compared to 0. If it is a 0, a left shift of one position takes place and the intermediate exponent is decreased by one. This is sufficient to normalize the result if the operands were normalized.

7. If the intermediate exponent is greater than 1023, the MO (multiply overflow) exception bit is set to 1 and the result is set to u. The remaining steps are omitted.

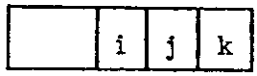
8. If the intermediate exponent is less than -1024, the MU (multiply underflow) exception bit is set to 1 and the result is set to 48/96 0's. The remaining steps are omitted.

9. If the exponent is greater than 511, the OW bit is set to 1. If it is less than -512, the UW bit is set to 1.

10. The intermediate fraction is truncated to 36/84 bits.

11. If rounding is called for and if any of the truncated bits were a 1, the low order bit of the fraction is set to 1.

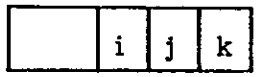
Multiply Normalized MN



The contents of register A^i are replaced by the normalized product of the single precision floating point numbers in A^j and A^k.

Exceptions	Exception bit
result exponent $>$ +1023	MO
result exponent $<$ -1024	MU
+511 $<$ result exponent $\leq$ +1023	OW
-512 $>$ result exponent $\geq$ -1024	UW
unnormalized operand	UO

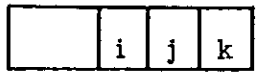
Multiply Double Normalized MDN



The contents of register pair $A^{i,i+1}$ are replaced by the normalized product of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent $>$ +1023	MO
result exponent $<$ -1024	MU
+511 $<$ result exponent $\leq$ +1023	OW
-512 $>$ result exponent $\geq$ -1024	UW
unnormalized operand	UO
i, j, or k odd	RS

Multiply Rounded MR



The contents of register A^i are replaced by the normalized and statistically rounded product of the single precision floating point numbers in A^j and A^k.

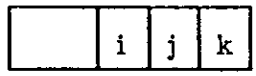
Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized operand	UO

Multiply Double Rounded MDR

The contents of register pair $A^{i,i+1}$ are replaced by the normalized and statistically rounded product of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized operand	UO
i, j, or k odd	RS

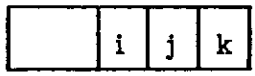
Multiply Unnormalized MU



The contents of register A^i are replaced by the unnormalized product of the single precision floating point numbers in A^j and A^k.

Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW

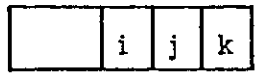
Multiply Double Unnormalized MDU



The contents of register pair $A^{i,i+1}$ are replaced by the unnormalized product of the double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
i, j, or k odd	RS

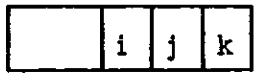
Multiply Mixed Normalized MMN



The contents of register pair $A^{i,i+1}$ are replaced by the double precision normalized product of the single precision floating point numbers in A^j and A^k.

Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
i odd	RS

Multiply Mixed Unnormalized MMU



The contents of register pair $A^{i,i+1}$ are replaced by the double precision unnormalized product of the single precision floating point numbers in Aj and A^k.

Exceptions	Exception bit
result exponent > +1023	MO
result exponent < -1024	MU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
i odd	RS

Division Instructions

The division operations are performed as follows:

1. If either operand is a u, the result is set to a u and the remaining steps are omitted.

2. If the fraction of the divisor is zero, the result is set to u, the DO bit is set to 1, and the remaining steps are omitted.

3. If the divisor is unnormalized, the result is set to u, the UD bit is set to 1, and the remaining steps are omitted.

4. If the fraction part of the dividend is zero, the result is set to zero, and the remaining steps are omitted.

5. The exponent of the divisor is subtracted from the exponent of the dividend to form an intermediate exponent.

6. The 36/84 bit dividend fraction is divided by the 36/84 bit divisor fraction to form the intermediate quotient fraction. Conceptually the quotient is computed to infinite precision and then truncated to 36/84 bits.

7. Since the divisor was normalized, the intermediate quotient fraction must be less than 2.0. If it is greater than or equal to 1.0, it is shifted right one position, a 1 is inserted in the high order position, and the intermediate exponent is increased by one. If the dividend was normalized, the fraction is now normalized.

8. If the intermediate exponent is greater than 1023, the result is set to u, the DO exception bit is set to 1, and the remaining steps are omitted.

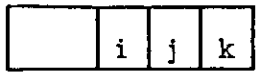
9. If the intermediate exponent is less than -1024, the result is set to all 0's, the DU exception bit is set to 1, and the remaining steps are omitted.

10. If the exponent is greater than 511, the OW bit is set to 1. If the exponent is less than -512, the UW bit is set to 1.

11. If rounding was specified and any of the bits truncated in steps 6 or 7 was a 1, the low order bit of the fraction is forced to a 1.

12. The sign of the result, determined by the rules of algebra, together with the intermediate exponent and fraction form the result.

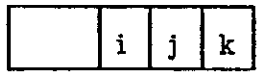
Divide Normalized DN



The contents of register A^i are replaced by the quotient formed by dividing the single precision floating point dividend in A^j by the single precision floating point divisor in A^k.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO

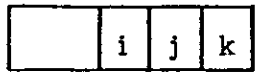
Divide Double Normalized DDN



The contents of register pair $A^{i,i+1}$ are replaced by the quotient formed by dividing the double precision floating point dividend in $A^{j,j+1}$ by the double precision floating point divisor in $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO
i, j, or k odd	RS

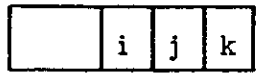
Divide Rounded DR



The contents of register A^i are replaced by the statistically rounded quotient formed by dividing the single precision floating point dividend in A^j by the single precision floating point divisor in A^k.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO

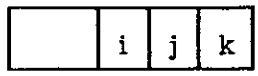
Divide Double Rounded DDR



The contents of register pair $A^{i,i+1}$ are replaced by the statistically rounded quotient formed by dividing the double precision floating point dividend in $A^{j,j+1}$ by the double precision floating point divisor in $A^{k,k+1}$.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO
i, j, or k odd	RS

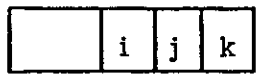
Divide Mixed Normalized DMN



The contents of register A^i are replaced by the quotient formed by dividing the double precision floating point dividend in $A^{j,j+1}$ by the single precision floating point divisor in A^k.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO
j odd	RS

Divide Mixed Rounded DMR

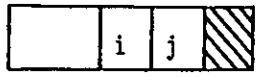


The contents of register A^i are replaced by the statistically rounded quotient formed by dividing the double precision floating point dividend in $A^{j,j+1}$ by the single precision floating point divisor in A^k.

Exceptions	Exception bit
result exponent > +1023	DO
result exponent < -1024	DU
+511 < result exponent ≤ +1023	OW
-512 > result exponent ≥ -1024	UW
unnormalized divisor	UD
divisor fraction = 0	DO
j odd	RS

Miscellaneous Instructions

Round RND



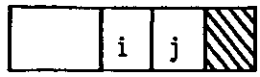
The double precision floating point number is register pair $A^{j,j+1}$ is rounded to form a single precision number which replaces the contents of A^i.

This round which is a "true round", rather than the statistical round which was described earlier, is performed as follows:

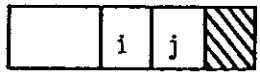
1. If the input operand is a double precision representation of u, the result is set to a single precision u and the remaining steps are omitted.
2. The exponent of the input forms the intermediate exponent.
3. If bit A^{j+1}_0 (i. e. bit 37 of the 84 bit fraction) is a 1, the magnitude of the fraction of A^j is increased by 2^{-36} to form an intermediate fraction.
4. If the addition in step 3 caused a fraction overflow, the intermediate fraction is shifted right one position, with its low order bit lost, and a 1 is inserted into the high order position of the fraction. Then the intermediate exponent is increased by one. If this causes an exponent overflow, the AO exception bit is set to 1, the result is set to u, and the remaining steps are omitted.
5. If the intermediate fraction is 36 0's, the result is set to 48 0's and the ZF exception bit is set to 1, and the remaining steps are omitted.
6. If there are no 1's in the high order 28 bits of the fraction, the LS bit is set to 1.
7. If the intermediate exponent is greater than 511 the OW exception bit is set to 1.
8. The sign of the input together with the intermediate exponent and fraction form the result.

Exceptions	Exception bit
result exponent $>$ +1023	AO
+511 $<$ result exponent $\leq$ +1023	OW
low significance	LS
zero fraction	ZF
j odd	RS

Set Positive, Floating SPF



Set Negative, Floating SNF

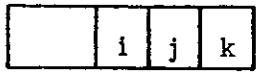


The 47 low-order bits of A^i are replaced by the 47 low-order bits of A^j. Bit A^i_0 is set to 0 for SPF; it is set to 1 for SNF.

If A^j is the u configuration, then A^i is set to the u configuration. If A^j is true zero (all 0's), then A^i is set to true zero.

Exceptions: none

Convert to Short Floating CVS



The single precision floating point number in A^j is converted to a short floating point number and replaces the contents of A^i. The length, n, of the exponent field of the short floating point number is given by the 5-bit 2's complement literal in the k field.

The conversion is performed as follows:

1. If n is negative, zero, or greater than eleven, the result is set to a u, the ILO (illegitimate operand) exception bit is set to 1, and the remaining steps are omitted.
2. If the input is u, the result is set to u, and the remaining steps are omitted.
3. If the input is 48 0's, the result is set to 48 0's and the remaining steps are omitted.
4. The 46 bits A^j_2, A^i_3, ..., A^j_{47} are shifted left 11-n positions. 0's fill the vacated positions. If any of the bits shifted out through A^j_2 are the same as A^j_1, this indicates that the exponent cannot be represented in 11-n bits. The SO bit is then set to 1, the result is set to u, and the remaining steps are omitted.
5. The two unshifted bits A^j_0, A^j_1 together with the 46 bits of the shifted quantity form the result.

Exceptions	Exception bit
$n > 11$	ILO
$n < 1$	ILO
unrepresentable exponent	SO

Convert to Full Floating CVF

The short, left justified, floating point number in A^j is converted to a single precision floating point number and replaces the contents of A^i. The length n, of the exponent field of the short floating point number is given by the 5-bit 2's complement literal in the k field.

The conversion is performed as follows:

1. If n is negative, zero, or greater than eleven, the result is set to u, the ILO exception bit is set to 1, and the remaining steps are omitted.
2. If the input is a u, the result is set to u, and the remaining steps omitted.
3. If the input is 48 0's, the result is set to 48 0's and the remaining steps are omitted.
4. The 46 bits A^j_2, A^j_3, ..., A^j_{47} are shifted right 11-n positions. The vacated bit positions are filled with the complement of the bit A^j_i.
5. The two unshifted bits A^j_0, A^j_1 together with the high order 46 bits from the shifted quantity form the result.

Exceptions	Exception bit
$n > 11$	ILO
$n < 1$	ILO