KIND(X)

Description

Returns the kind type value of X.

Kind

Inquiry function.

Argument

X A scalar or array of any numeric type.

Result Type

Scalar of type INTEGER*4.

Result Value

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
INTEGER*4	32
INTEGER*8	64
REAL*4	32
REAL*8	64
COMPLEX*8	32
COMPLEX*16	64

Note

The INTEGER*8 type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

LASTLOC(MASK *[*, DIM *]*)

Description

Locates the last true element (in array element order) of a logical mask array, or locates the last true element of each of the complete rank-one sections parallel to a given dimension of an array of rank two or greater, effectively locating the "trailing edge" of the set of true elements along the given dimension. The result is a logical mask that may be used to select elements of the original array.

Kind

Transformational function.

Arguments

`MASK` An array of type logical.

`DIM` (optional)
A scalar expression of type integer, with a value in the range $1 \leq$ `DIM` $\leq n$, where n is the rank of `MASK`.

Result Type

The result is an array of type logical with the same shape as `MASK`.

Result Value

Case 1 `LASTLOC ( MASK )`
`DIM` is absent:

The result has at most one true element. If `MASK` is all false, the result is all false. If `MASK` contains one or more true elements, the result has a single true element and it is in the position corresponding to the last true element (in array element order) in `MASK`.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `LASTLOC( MASK, DIM )`
`DIM` is present:

The result is formed by applying `LASTLOC` to each of the rank-one array sections of `MASK` that lie parallel to dimension `DIM`. Thus the value of section

`R(`s_1`,`s_2`,...,`s_{DIM-1}`,:,`s_{DIM+1}`,...,`s_n`)`

of a result array `R` is equal to

`LASTLOC( MASK`$_{DIM}$ `)`

where `MASK`$_{DIM}$ is array section

`MASK(`s_1`,`s_2`,...,`s_{DIM-1}`,:,`s_{DIM+1}`,...,`s_n`)`.

Examples

Case 1 `DIM` is absent:

$$\texttt{LASTLOC(}\begin{bmatrix} . & . & T & . \\ . & T & T & T \\ . & T & . & . \end{bmatrix}\texttt{)} \rightarrow \begin{bmatrix} . & . & . & . \\ . & . & . & T \\ . & . & . & . \end{bmatrix}$$

`LASTLOC([.,`T`,.,`T`,`T`,.,.]) → [.,.,.,.,`T`,.,.]`

Case 2 `DIM` is present:

The "bottom edge" of a two-dimensional mask is obtained by

$$\texttt{LASTLOC(}\begin{bmatrix} . & . & T & . \\ . & T & T & T \\ . & T & . & . \end{bmatrix}\texttt{, DIM = 1)} \rightarrow \begin{bmatrix} . & . & . & . \\ . & . & T & T \\ . & T & . & . \end{bmatrix}$$

while the "right edge" is obtained by

$$\texttt{LASTLOC(}\begin{bmatrix} . & . & T & . \\ . & T & T & T \\ . & T & . & . \end{bmatrix}\texttt{, DIM = 2)} \rightarrow \begin{bmatrix} . & . & T & . \\ . & . & . & T \\ . & T & . & . \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

LBOUND(ARRAY *[*, DIM *]*)

Description

Returns all the lower bounds of an array, or the lower bound of a given dimension.

Compatibility Note

For back compatibility, CM Fortran also accepts the intrinsic function DLBOUND, which is equivalent to LBOUND.

Kind

Inquiry function.

Arguments

ARRAY An array of numeric or logical type. It must not be an assumed-size array.

DIM (optional)
A scalar expression of type integer, with a value in the range $1 \le$ DIM $\le n$, where n is the rank of ARRAY.

Result Type

The result is of type INTEGER*4. It is a scalar if DIM is present; if DIM is absent, the result is a CM array of rank one and size n, where n is the rank of ARRAY.

Result Value

Case 1 LBOUND (ARRAY, DIM)
DIM is present:

The result is a scalar value equal to the lower bound for subscript DIM of ARRAY if dimension DIM of ARRAY does not have size zero; the result is 1 if dimension DIM has size zero. For an array section or other array expression, the result is 1.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `LBOUND ( ARRAY )`
`DIM` is absent:

For an `ARRAY` of rank *n*, the result is a CM array with the value

$[e_1, e_2, \ldots, e_n]$

where e_i = `LBOUND(ARRAY,`*i*`)`, for *i* = 1, 2, ..., *n*.

Examples

If `A` is declared by the statement

```
REAL A(2:3,7:10)
```

then

```
LBOUND(A) → [2,7]
LBOUND( A, DIM = 1 ) → 2
LBOUND( A, DIM = 2 ) → 7
```

Also,

```
LBOUND( [-2,3,10] ) → [1]
LBOUND( [-2,3,10], DIM = 1 ) → 1
LBOUND( [1:-1] ) → [1]
LBOUND( [1:-1], DIM = 1 ) → 1
```


Copyright © 1989–1994 Thinking Machines Corporation

LEADZ(X)

Description

Returns the number of leading zeros in the bit-level representation of a value.

Kind

Elemental function.

Argument

X	A scalar or array of type logical or integer.

Result Type

If the argument is LOGICAL or INTEGER*4, the result is INTEGER*4. If the argument is INTEGER*8, the result is INTEGER*8.

Result Value

The result is the number of consecutive bits set to zero, beginning from the high-order (left-most) bit position, before a bit set to 1 is encountered.

If X is an array argument, the result is an array of the same shape. Each element of the array returned contains the result of LEADZ for the corresponding element of X.

Examples

```
LEADZ(1) → 31
```

$$\texttt{LEADZ}\left(\begin{bmatrix}0 & 1 & 2\\ 3 & 4 & 5\\ 6 & 7 & 8\end{bmatrix}\right) \rightarrow \begin{bmatrix}32 & 31 & 30\\ 30 & 29 & 29\\ 29 & 29 & 28\end{bmatrix}$$

```
LEADZ(0) → 32

LEADZ(X) → 0 for all X < 0

LEADZ(.FALSE.) → 0

LEADZ(.TRUE.) → 32
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

LEN(STRING)

Description

Returns the length of a scalar character entity.

Kind

Inquiry function.

Argument

`STRING` A scalar or array of type character.

Result Type

Integer scalar.

Result Value

The result has a value equal to the number of characters in `STRING`.

Examples

If `C`, `D`, and E are declared by the statement

```
CHARACTER*11 C, D(100), E*4
```

then

```
LEN(C) → 11
LEN(D(1)) → 11
LEN(E) → 4
```

Also,

```
LEN("1234") → 4
LEN( "x" // "yy" // "zzz" ) → 6
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

LGE(STRING_A, STRING_B)

Description

Tests whether a string is lexically greater than or equal to another string, based on the ASCII collating sequence.

Kind

Elemental function (restricted to scalar arguments).

Arguments

`STRING_A`	A scalar expression of type character.
`STRING_B`	A scalar expression of type character.

Result Type

Logical.

Result Value

If the strings are of unequal length, the comparison is made as if the shorter string were extended on the right with blanks to the length of the longer string. The result is true if the strings are equal or if `STRING_A` follows `STRING_B` in the ASCII collating sequence.

If either string contains a character not in the ASCII character set, the result is undefined.

Examples

```
LGE( "ONE", "TWO" ) → F
LGE( "Y", "Y" ) → T
LGE( "XXX", "XX" ) → T
LGE( "x", "X" ) → T
```


Copyright © 1989–1994 Thinking Machines Corporation

LGT(STRING_A, STRING_B)

Description

Tests whether a string is lexically greater than another string, based on the ASCII collating sequence.

Kind

Elemental function (restricted to scalar arguments).

Arguments

`STRING_A`	A scalar expression of type character.
`STRING_B`	A scalar expression of type character.

Result Type

Logical.

Result Value

If the strings are of unequal length, the comparison is made as if the shorter string were extended on the right with blanks to the length of the longer string. The result is true if `STRING_A` follows `STRING_B` in the ASCII collating sequence; otherwise, the result is false.

If either string contains a character not in the ASCII character set, the result is undefined.

Examples

```
LGT( "ONE", "TWO" ) → F
LGT( "Y", "Y" ) → F
LGT( "XXX", "XX" ) → T
LGT( "x", "X" ) → T
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

LLE(STRING_A, STRING_B)

Description

Tests whether a string is lexically less than or equal to another string, based on the ASCII collating sequence.

Kind

Elemental function (restricted to scalar arguments).

Arguments

`STRING_A`	A scalar expression of type character.
`STRING_B`	A scalar expression of type character.

Result Type

Logical.

Result Value

If the strings are of unequal length, the comparison is made as if the shorter string were extended on the right with blanks to the length of the longer string. The result is true if the strings are equal or if `STRING_A` precedes `STRING_B` in the ASCII collating sequence; otherwise, the result is false.

If either string contains a character not in the ASCII character set, the result is undefined.

Examples

```
LLE( "ONE", "TWO" ) → T
LLE( "Y", "Y" ) → T
LLE( "XXX", "XX" ) → F
LLE( "x", "X" ) → F
```


Copyright © 1989–1994 Thinking Machines Corporation

LLT(STRING_A, STRING_B)

Description

Tests whether a string is lexically less than another string, based on the ASCII collating sequence.

Kind

Elemental function (restricted to scalar arguments).

Arguments

`STRING_A`	A scalar expression of type character.
`STRING_B`	A scalar expression of type character.

Result Type

Logical.

Result Value

If the strings are of unequal length, the comparison is made as if the shorter string were extended on the right with blanks to the length of the longer string. The result is true if `STRING_A` precedes `STRING_B` in the ASCII collating sequence; otherwise, the result is false.

If either string contains a character not in the ASCII character set, the result is undefined.

Examples

```
LLT( "ONE", "TWO" ) → T
LLT( "Y", "Y" ) → F
LLT( "XXX", "XX" ) → F
LLT( "x", "X" ) → F
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

LOG(X)

Description

Computes the natural logarithm of a number.

Kind

Elemental function.

Argument

X	An expression of type real or complex. If X is real, its value must be greater than zero; if X is complex, its value must not be zero.

Result Type Same as X.

Result Value

The result is an approximation to the natural logarithm function $\log_e$ X. If X is of type complex, the result is a principal value in one of the forms:

(*a*,*b*)	where *b* is in the range $-\pi < b < \pi$
(*a*,π)	if X has the value (*x*,0), where $x < 0$.

LOG is the inverse of EXP.

Examples

```
LOG(EXP(10.)) → 10.
LOG(EXP(-10.d0)) → -10.d0
LOG((0,1)) ⇒ (0,1.570796)
LOG((-1,0)) ⇒ (0.,3.14159)
LOG((-3,0)) ⇒ (1.09861,3.14159)
```

$$\texttt{LOG}\left(\begin{bmatrix}10.0000 & 0.69315\\ 1.00000 & 1.57080\end{bmatrix}\right) \Rightarrow \begin{bmatrix}2.30259 & -0.36651\\ 0.00000 & 0.45158\end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

LOG10(X)

Description

Computes the common logarithm of a number.

Kind

Elemental function.

Argument

X An expression of type real, with a value greater than zero.

Result Type

Same as X.

Result Value

The result is an approximation to the common logarithm function $\log_{10} X$.

Examples

```
LOG10(10.d0) → 1.d0

LOG10(10.**-5) → -5.

LOG10( [ 1001.0   0.30103 ] ) ⇒ [ 3.00043 -0.52139 ]
       [ 1.0000   2.00000 ]     [ 0.00000  0.30103 ]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

MATMUL(MATRIX_A, MATRIX_B)

Description

Performs matrix multiplication of numeric or logical matrices.

Kind

Transformational function.

Arguments

`MATRIX_A`

A numeric or logical array of rank one or two.

`MATRIX_B`

An array of numeric type if `MATRIX_A` is of numeric type, or of type logical if `MATRIX_A` is of type logical.

If `MATRIX_A` has rank one, `MATRIX_B` must have rank two; if `MATRIX_B` has rank one, `MATRIX_A` must have rank two. The size of the first (or only) dimension of `MATRIX_B` must equal the size of the last (or only) dimension of `MATRIX_A`.

Result Type

If the arguments are of type logical, the result is of type logical. If the arguments are of numeric type, the result type is determined using the rules for expressions given in Table 2, page 86.

The shape of the result depends on the shapes of the arguments as follows:

Case 1 If `MATRIX_A` has shape `[`*n*, *m*`]` and `MATRIX_B` has shape `[`*m*, *k*`]`, the result has shape `[`*n*, *k*`]`.

Case 2 If `MATRIX_A` has shape `[`*m*`]` and `MATRIX_B` has shape `[`*m*, *k*`]`, the result has shape `[`*k*`]`.

Case 3 If `MATRIX_A` has shape `[`*n*, *m*`]` and `MATRIX_B` has shape `[`*m*`]`, the result has shape `[`*n*`]`.

Copyright © 1989–1994 Thinking Machines Corporation

Result Value

Case 1 If the arguments are of numeric type, element `R(`*i*`,`*j*`)` of the result `R` is

```
SUM( MATRIX_A(i,:) * MATRIX_B(:,j) )
```

If the arguments are of logical type, element `R(`*i*`,`*j*`)` of the result `R` is

```
ANY( MATRIX_A(i,:) .AND. MATRIX_B(:,j)
```

Case 2 If the arguments are of numeric type, element `R(`*j*`)` of the the result `R` is

```
SUM( MATRIX_A(:) * MATRIX_B(:,j) )
```

If the arguments are of logical type, element `R(`*j*`)` of the result `R` is

```
ANY( MATRIX_A(:) .AND. MATRIX_B(:,j)
```

Case 3 If the arguments are of numeric type, element `R(`*j*`)` of the result `R` is

```
SUM( MATRIX_A(i,:) * MATRIX_B(:) )
```

If the arguments are of logical type, element `R(`*j*`)` of the result `R` is

```
ANY( MATRIX_A(i,:) .AND. MATRIX_B(:) )
```

Examples

Case 1 `MATMUL(`$\begin{bmatrix}1&2&3\\2&3&4\end{bmatrix}$`,`$\begin{bmatrix}1&2\\2&3\\3&4\end{bmatrix}$`)` $\rightarrow$ $\begin{bmatrix}14&20\\20&29\end{bmatrix}$

`MATMUL(`$\begin{bmatrix}T&T&.\\.&T&.\end{bmatrix}$`,`$\begin{bmatrix}T&T&.&T\\T&.&T&.\\T&T&T&.\end{bmatrix}$`)` $\rightarrow$ $\begin{bmatrix}T&T&T&T\\T&.&T&.\end{bmatrix}$

Case 2 `MATMUL( [1,2],`$\begin{bmatrix}1&2&3\\2&3&4\end{bmatrix}$`)` $\rightarrow$ `[5,8,11]`

Case 3 `MATMUL(`$\begin{bmatrix}1&2&3\\2&3&4\end{bmatrix}$`, [1,2,3] )` $\rightarrow$ `[14,20]`

Note

When the arguments to `MATMUL` are of type real, double precision, complex, or double complex, the function's performance is significantly improved if the program is linked with the CM Scientific Software Library, `-lcmssl`.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

MAX(A1, A2 *[*, A3, *]*...)

Description

Computes the maximum value of two or more numbers.

Kind

Elemental function.

Arguments

`A1, A2, A3, . . .`

The arguments must all be expressions of the same type, which must be integer or real. Argument `A3` and any following arguments are optional.

Result Type

Same type as the arguments.

Result Value

The value of the result is that of the largest argument.

Examples

`MAX( 17.1,-3., 2., 25.,4. )` → `25.`

$$\texttt{MAX(}\begin{bmatrix}-5 & -2 & 8\\ 7 & -4 & 3\end{bmatrix}\texttt{,}\begin{bmatrix}-3 & 0 & 7\\ -1 & -5 & 4\end{bmatrix}\texttt{)} \rightarrow \begin{bmatrix}-3 & 0 & 8\\ 7 & -4 & 4\end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

MAXEXPONENT(X)

Description

Returns the maximum exponent in the model representing numbers of the same type and kind type as the argument.

Kind

Inquiry function.

Argument

`X` A scalar or array of type `REAL`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result has the value e_{max}, as defined in Section 17.15 for the model representing numbers of the same type and kind type as `X`.

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
`REAL*4`	128
`REAL*8`	1024

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

MAXLOC(ARRAY *[*, MASK *]*)

Description

Locates the first element (in array element order) of the subset of elements of a numeric array having the maximum value of all the array elements, possibly under the control of a mask. The result is an array of values that may be used in subscript expressions to choose the selected element from the original array, or an array of zeros if no maximum element is found.

Kind

Transformational function.

Arguments

ARRAY An array of type integer or real.

MASK (optional)
A scalar or an array of type logical that is conformable with ARRAY.

Result Type

The result is a rank-one array of type INTEGER*4.

Result Value

The result is an array of size equal to the rank n of ARRAY. If a maximum element is found, the result is an array $[s_1, s_2, \ldots, s_n]$, each of whose element values s_i lies in the range 1 to e_i, where e_i is the extent of the ith dimension of ARRAY. If a maximum element is not found, the result is an array of zeros.

Case 1 MAXLOC (ARRAY)
MASK is absent:

If ARRAY has size greater than zero, the result array elements $s_1, s_2, \ldots, s_n$ designate the first element (in array element order) of the subset of elements of ARRAY having the maximum value of all the elements of the array. If ARRAY has size zero, the value of the result is an array of zeros.

Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `MAXLOC ( ARRAY, MASK )`
`MASK` is present:

If `ARRAY` has size greater than zero and if at least one element of `MASK` is true, the result array elements $s_1, s_2, ..., s_n$ designate the first element (in array element order) of the subset of elements of `ARRAY` having the maximum value of all the elements of the array and for which the corresponding elements of `MASK` are true. If `ARRAY` has size zero or if every element of `MASK` is false, the value of the result is an array of zeros.

NOTE: If the lower bounds of each dimension of `ARRAY` are equal to `1`, then the returned values s_i form valid subscripts of a maximum element `ARRAY` ($s_1, s_2, ..., s_n$). If some of the lower bounds of `ARRAY` are not equal to `1`, then the s_i must be translated to the required range using

t_i = `LBOUND ( ARRAY, i )` + s_i − `1`

and `ARRAY` ($t_1, t_2, ..., t_n$) is the array element designated by the result.

Examples

Case 1 `MASK` is absent:

```
MAXLOC ( [2,4,7,3,-2,5] ) → [3]

MAXLOC ( [1:0] ) → [0]
```

If `A` and `B` are 3×4 arrays declared as

```
INTEGER A(3,4), B(-3:-1,-2:1)
```

both with the value $\begin{bmatrix} 7 & 5 & 3 & 19 \\ 34 & 16 & 62 & 6 \\ 8 & 19 & 10 & 40 \end{bmatrix}$, then

```
MAXLOC(A) → [2,3]
```

and

```
MAXLOC(B) → [2,3]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `MASK` is present:

If $A = \begin{bmatrix} 0 & -5 & 8 & -3 \\ 3 & 4 & -1 & 2 \\ 1 & 5 & 6 & -4 \end{bmatrix}$, then

```
MAXLOC( A, MASK = A.LT.6 ) → [3,2].
```

If C is the 3×4×2 array $\begin{bmatrix} \begin{bmatrix} 7 & 5 & 62 & 19 \\ 34 & 16 & -3 & 6 \\ 8 & 19 & 10 & 40 \end{bmatrix} \\ \begin{bmatrix} 8 & 14 & 2 & 14 \\ 13 & -8 & 5 & 15 \\ 61 & 4 & 8 & -3 \end{bmatrix} \end{bmatrix}$, then

```
MAXLOC( C, MASK = C.LT.0 ) → [2,3,1]
```


Copyright © 1989–1994 Thinking Machines Corporation

MAXVAL(ARRAY *[*, DIM *]* *[*, MASK *]*)

Description

Computes the maximum value of all elements of a numeric array, or computes the maximum value of each of the complete rank-one sections parallel to a given dimension of an array of rank two or greater, possibly under the control of a mask.

Kind

Transformational (reduction) function.

Arguments

`ARRAY` An array of type integer or real.

`DIM` (optional)
A *constant* scalar expression of type integer, with a value in the range $1 \le$ `DIM` $\le n$, where n is the rank of `ARRAY`.

`MASK` (optional)
An array or scalar expression of type logical; if an array, it must be conformable with `ARRAY`.

Result Type

The result has the same type as `ARRAY`. It is a scalar if `DIM` is absent or if `ARRAY` has rank one; otherwise, it is an array with shape equal to that of `ARRAY` with dimension `DIM` removed.

Result Value

Case 1a `MAXVAL( ARRAY, DIM = 1 [, MASK ] )`
`ARRAY` has rank `1`, `DIM` is present

or

Case 1b `MAXVAL( ARRAY [, MASK = MASK ] )`
`ARRAY` has any rank, `DIM` is absent:

The result is a scalar equal (approximately) to the maximum value of the elements of `ARRAY` corresponding to the true elements of `MASK`, if present. If `MASK` is absent, the result is the maximum value of all elements of `ARRAY`.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

If ARRAY is a zero-sized array, or if MASK is present and no elements of MASK are true, the result is the scalar value equal to the negative number of largest magnitude of the type of ARRAY.

Case 2 MAXVAL(ARRAY, DIM [, MASK])
ARRAY has rank $n > 1$, DIM is present:

For an ARRAY of rank n with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM}, d_{DIM+1}, ..., d_n]

the result is an array of rank $n-1$ with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM+1}, ..., d_n].

The result is formed by computing the maximum value of each of the complete rank-one sections of ARRAY that lie parallel to dimension DIM . If MASK is present, only elements of ARRAY corresponding to the true elements of MASK are used in the computation; if MASK is absent, all elements of ARRAY are used. Thus, the value of element

R(s_1, s_2, ..., s_{DIM-1}, s_{DIM+1}, ..., s_n)

of a result array R is equal to

MAXVAL($ARRAY_{DIM}$ [, MASK = $MASK_{DIM}$])

where $ARRAY_{DIM}$ and $MASK_{DIM}$ are complete rank-one sections of ARRAY and MASK, both with shape [d_{DIM}]. $ARRAY_{DIM}$ is the array section

ARRAY(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n)

and $MASK_{DIM}$ is the array section

MASK(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n)

If MASK is present and all its elements are false, or if only dimension DIM of ARRAY is zero-sized, the result is an array with all elements equal to the negative number of largest magnitude of the type of ARRAY. If any dimension of ARRAY other than DIM is zero-sized, the result is a zero-sized array.

Examples

Case 1a ARRAY has rank 1, DIM is present (with value 1):

MAXVAL([2,3,-5,7], DIM = 1) → 7

If A = [2,3,-5,7] and M = [.,*T*,*T*,], then

MAXVAL(ARRAY = A, DIM = 1, MASK = M) → 3

MAXVAL([1:0], DIM = 1) → -2147483648

Copyright © 1989–1994 Thinking Machines Corporation

Case 1b `ARRAY` has any rank, `DIM` is absent:

If $\texttt{A} = \begin{bmatrix} -4 & 25 & -1 \\ 15 & 5 & -2 \end{bmatrix}$, then

```
MAXVAL(A) → 25
```

and

```
MAXVAL( ARRAY = A, MASK = A < 0 ) → -1
```

$\texttt{MAXVAL(}\begin{bmatrix} -4 & 25 & -1 \\ 15 & 5 & -2 \end{bmatrix}\texttt{, MASK =}\begin{bmatrix} T & . & T \\ . & T & T \end{bmatrix}\texttt{)} \rightarrow 5$

Case 2 `ARRAY` has rank $n > 1$, `DIM` is present:

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$, then

```
MAXVAL( ARRAY = B, DIM = 1 ) → [17,1,14,18]
```

and

```
MAXVAL( ARRAY = B, DIM = 2 ) → [18,17]
```

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} . & . & . & T \\ T & . & T & . \end{bmatrix}$, then

```
MAXVAL( ARRAY = B, DIM = 1, MASK = M ) →
    [17,-2147483648,14,18]
```

and

```
MAXVAL( ARRAY = B, DIM = 2, MASK = M ) → [18,17]
```

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} . & . & . & . \\ T & T & T & T \end{bmatrix}$, then

```
MAXVAL( ARRAY = B, DIM = 1, MASK = M ) → [17,-7,14,0]
```

and

```
MAXVAL( ARRAY = B, DIM = 2, MASK = M ) →
   [-2147483648,17]
```


Copyright © 1989–1994 Thinking Machines Corporation

MERGE(TSOURCE, FSOURCE, MASK)

Description

Returns one of two alternative values according to the value of a mask.

Kind

Elemental function.

Arguments

`TSOURCE` An expression of numeric or logical type.

`FSOURCE` An expression of the same type as `TSOURCE`.

`MASK` An expression of type logical.

Result Type

Same as the arguments.

Result Value

The result is `TSOURCE` if `MASK` is true and `FSOURCE` otherwise.

Examples

$$\texttt{MERGE(}\begin{bmatrix}1 & 6 & 5\\ 2 & 4 & 6\end{bmatrix}\texttt{,}\begin{bmatrix}0 & 3 & 2\\ 7 & 4 & 8\end{bmatrix}\texttt{,}\begin{bmatrix}T & . & T\\ . & . & T\end{bmatrix}\texttt{)} \rightarrow \begin{bmatrix}1 & 3 & 5\\ 7 & 4 & 6\end{bmatrix}.$$

`MERGE( 0, 1, MOD([1:7],2)==1) )` → `[0,1,0,1,0,1,0].`

For any two conformable arrays `A` and `B` of the same integer or real type,

`MERGE( A, B, A>B )` → `MAX(A,B).`

If $\texttt{A} = \begin{bmatrix}-4 & 25 & 1\\ 15 & 5 & 30\end{bmatrix}$ and $\texttt{B} = \begin{bmatrix}3 & 19 & -5\\ 7 & 4 & 17\end{bmatrix}$, then

$$\texttt{MERGE(A, B, MOD(A+B,2)==0)} \rightarrow \begin{bmatrix}3 & 25 & 1\\ 15 & 4 & 17\end{bmatrix}.$$

If `A` is defined as above, then

$$\texttt{MERGE(2, MERGE(20, A, A>20), A<2)} \rightarrow \begin{bmatrix}2 & 20 & 2\\ 15 & 5 & 20\end{bmatrix}.$$

Copyright © 1989–1994 Thinking Machines Corporation

MIN(A1, A2 *[*, A3 *]*...)

Description

Computes the minimum of two or more numbers.

Kind

Elemental function.

Arguments

`A1, A2, A3, . . .`

The arguments must all be expressions of the same type, which must be integer or real. Argument `A3` and any following arguments are optional.

Result Type

Same type as the arguments.

Result Value

The value of the result is that of the smallest argument.

Examples

`MIN( 17.1,-3., 2., 25.,4. )` → `-3.`

$$\texttt{MIN}\left(\begin{bmatrix}-5 & -2 & 8\\ 7 & -4 & 3\end{bmatrix},\begin{bmatrix}-3 & 0 & 7\\ -1 & -5 & 4\end{bmatrix}\right) \rightarrow \begin{bmatrix}-5 & -2 & 7\\ -1 & -5 & 3\end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

MINEXPONENT(X)

Description

Returns the minimum (most negative) exponent in the model representing numbers of the same type and kind type as the argument.

Kind

Inquiry function.

Argument

`X` A scalar or array of type `REAL`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result has the value e_{min}, as defined in Section 17.15 for the model representing numbers of the same type and kind type as `X`.

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
`REAL*4`	–125
`REAL*8`	–1021

Copyright © 1989–1994 Thinking Machines Corporation

MINLOC(ARRAY *[*, MASK *]*)

Description

Locates the first element (in array element order) of the subset of elements of a numeric array having the minimum value of all the array elements, possibly under the control of a mask. The result is an array of values that may be used in subscript expressions to choose the selected element from the original array, or an array of zeros if no minimum element is found.

Kind

Transformational function.

Arguments

ARRAY An array of type integer or real.

MASK (optional)
A scalar or an array of type logical that is conformable with ARRAY.

Result Type

The result is a rank-one array of type INTEGER*4.

Result Value

The result is an array of size equal to the rank n of ARRAY. If a minimum element is found, the result is an array $[s_1, s_2, \ldots, s_n]$, each of whose element values s_i lies in the range 1 to e_i, where e_i is the extent of the ith dimension of ARRAY. If a minimum element is not found, the result is an array of zeros.

Case 1 MINLOC (ARRAY)
MASK is absent:

If ARRAY has size greater than zero, the result array elements $s_1, s_2, \ldots, s_n$ designate the first element (in array element order) of the subset of elements of ARRAY having the minimum value of all the elements of the array. If ARRAY has size zero, the value of the result is an array of zeros.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `MINLOC( ARRAY, MASK )`
`MASK` is present:

If `ARRAY` has size greater than zero and if at least one element of `MASK` is true, the result array elements $s_1, s_2, \ldots, s_n$ designate the first element (in array element order) of the subset of elements of `ARRAY` having the minimum value of all the elements of the array and for which the corresponding elements of `MASK` are true. If `ARRAY` has size zero or if every element of `MASK` is false, the value of the result is an array of zeros.

NOTE: If the lower bounds of each dimension of `ARRAY` are equal to `1`, then the returned values s_i form valid subscripts of a minimum element `ARRAY(`$s_1, s_2, \ldots, s_n$`)`. If some of the lower bounds of `ARRAY` are not equal to `1`, then the s_i must be translated to the required range using

t_i = `LBOUND( ARRAY, i ) +` s_i `- 1`

and `ARRAY(`$t_1, t_2, \ldots, t_n$`)` is the array element designated by the result.

Examples

Case 1 `MASK` is absent:

```
MINLOC( [2,4,7,3,-2,9] ) → [5]
```

```
MINLOC( [1:0] ) → [0]
```

If `A` and `B` are `3×4` arrays declared as

```
INTEGER A(3,4), B(-3:-1,-2:1)
```

both with the value $\begin{bmatrix} 7 & 5 & 3 & 19 \\ 34 & 16 & 62 & 6 \\ 8 & 19 & 10 & 40 \end{bmatrix}$, then

```
MINLOC(A) → [1,3]
```

and

```
MINLOC(B) → [1,3]
```


Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `MASK` is present:

If `A` = $\begin{bmatrix} 0 & -5 & 8 & -3 \\ 3 & 4 & -1 & 2 \\ 1 & 5 & 6 & -4 \end{bmatrix}$, then

```
MINLOC( A, MASK = A.GT.0 ) → [3,1]
```

If `C` is the 3×4×2 array $\begin{bmatrix} \begin{bmatrix} 7 & 5 & 62 & 19 \\ 34 & 16 & -3 & 6 \\ 8 & 19 & 10 & 40 \end{bmatrix} \\ \begin{bmatrix} 8 & 14 & 2 & 14 \\ 13 & -8 & 5 & 15 \\ 61 & 4 & 8 & -3 \end{bmatrix} \end{bmatrix}$, then

```
MINLOC( C, MASK = C.GT.0 ) → [1,3,2]
```


Copyright © 1989–1994 Thinking Machines Corporation

MINVAL(ARRAY *[*, DIM *] [*, MASK *]*)

Description

Computes the minimum value of all elements of a numeric array of rank one, or computes the minimum value of all complete rank-one sections parallel to a given dimension of an array of rank two or greater, possibly under the control of a mask.

Kind

Transformational (reduction) function.

Arguments

`ARRAY` An array of type integer or real. It must not be scalar.

`DIM` (optional)
A *constant* scalar expression of type integer, with a value in the range $1 \le$ `DIM` $\le n$, where n is the rank of `ARRAY`.

`MASK` (optional)
An array or scalar expression of type logical; if an array, it must be conformable with `ARRAY`.

Result Type

The result has the same type as `ARRAY`. It is a scalar if `DIM` is absent or if `ARRAY` has rank one; otherwise, it is an array with shape equal to that of `ARRAY` with dimension `DIM` removed.

Result Value

Case 1a `MINVAL( ARRAY, DIM = 1 [, MASK ] )`
`ARRAY` has rank `1`, `DIM` is present (with value `1`)

or

Case 1b `MINVAL( ARRAY [, MASK = MASK ] )`
`ARRAY` has any rank, `DIM` is absent:

The result is a scalar equal (approximately) to the minimum value of the elements of `ARRAY` corresponding to the true elements of `MASK`, if present. If `MASK` is absent, the result is the minimum value of all elements of `ARRAY`.

Copyright © 1989–1994 Thinking Machines Corporation

If ARRAY is a zero-sized array, or if MASK is present and no elements of MASK are true, the result is the scalar value equal to the positive number of largest magnitude of the type of ARRAY.

Case 2 MINVAL(ARRAY, DIM [, MASK])
ARRAY has rank $n > 1$, DIM is present:

For an ARRAY of rank n with shape

$[\ d_1, d_2, \ldots, d_{DIM-1}, d_{DIM}, d_{DIM+1}, \ldots, d_n\ ]$

the result is an array of rank $n-1$ with shape

$[\ d_1, d_2, \ldots, d_{DIM-1}, d_{DIM+1}, \ldots, d_n\ ]$.

The result is formed by computing the minimum value of each of the complete rank-one sections of ARRAY that lie parallel to dimension DIM . If MASK is present, only elements of ARRAY corresponding to the true elements of MASK are used in the computation; if MASK is absent, all elements of ARRAY are used. Thus, the value of element

$R(\ s_1, s_2, \ldots, s_{DIM-1}, s_{DIM+1}, \ldots, s_n\ )$

of a result array R is equal to

$MINVAL(\ ARRAY_{DIM}\ [, MASK = MASK_{DIM}\ ]\ )$

where $ARRAY_{DIM}$ and $MASK_{DIM}$ are complete rank-one sections of ARRAY and MASK, both with shape $[d_{DIM}]$. $ARRAY_{DIM}$ is the array section

$ARRAY(\ s_1, s_2, \ldots, s_{DIM-1}, :, s_{DIM+1}, \ldots, s_n\ )$

and $MASK_{DIM}$ is the array section

$MASK(\ s_1, s_2, \ldots, s_{DIM-1}, :, s_{DIM+1}, \ldots, s_n\ )$

If MASK is present and all its elements are false, or if only dimension DIM of ARRAY is zero-sized, the result is an array with all elements equal to the positive number of largest magnitude of the type of ARRAY. If any dimension of ARRAY other than DIM is zero-sized, the result is a zero-sized array.

Examples

Case 1a ARRAY has rank 1, DIM is present (with value 1):

MINVAL([2,3,-5,7], DIM = 1) → -5

If A = [2,3,-5,7] and M = [.,*T*,.,*T*], then
MINVAL(ARRAY = A, DIM = 1, MASK = M) → 3

MINVAL([1:0], DIM = 1) → 2147483647

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 1b `ARRAY` has any rank, `DIM` is absent:

If $\texttt{A} = \begin{bmatrix} -4 & 25 & -1 \\ 15 & 5 & -2 \end{bmatrix}$, then

```
MINVAL(A) → -4
```

and

```
MINVAL( ARRAY = A, MASK = A > 0 ) → 5
```

$\texttt{MINVAL(}\begin{bmatrix} -4 & 25 & -1 \\ 15 & 5 & -2 \end{bmatrix}\texttt{, MASK =}\begin{bmatrix} T & . & T \\ . & T & T \end{bmatrix}\texttt{)} \rightarrow -4$

Case 2 `ARRAY` has rank $n > 1$, `DIM` is present:

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$, then

```
MINVAL( ARRAY = B, DIM = 1 ) → [3,-7,5,0]
```

and

```
MINVAL( ARRAY = B, DIM = 2 ) → [1,-7]
```

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} . & . & . & T \\ T & . & T & . \end{bmatrix}$, then

```
MINVAL( ARRAY = B, DIM = 1, MASK = M ) →
   [17,2147483647,14,18]
```

and

```
MINVAL( ARRAY = B, DIM = 2, MASK = M ) → [18,14]
```

If $\texttt{B} = \begin{bmatrix} 3 & 1 & 5 & 18 \\ 17 & -7 & 14 & 0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} . & . & . & . \\ T & T & T & T \end{bmatrix}$, then

```
MINVAL( ARRAY = B, DIM = 1, MASK = M ) → [17,-7,14,0]
```

and

```
MINVAL( ARRAY = B, DIM = 2, MASK = M ) →
    [2147483647,-7]
```


Copyright © 1989–1994 Thinking Machines Corporation

MOD(A, P)

Description

Returns the remainder obtained on dividing one number by another.

Kind

Elemental function.

Arguments

`A`	An expression of type integer or real.
`P`	An expression of the same type as `A`.

Result Type

Same type as `A`.

Result Value

The result is the remainder obtained on dividing `A` by `P`. If `P` is not equal to zero, the value of the result is `A - INT(A/P) * P`. If `P` is equal to zero, the result is undefined. `MOD` always has the same sign as `A`.

Examples

```
MOD( 101, 2 ) → 1
```

$$\texttt{MOD}\left(\begin{bmatrix} -5 & -1 & 2 \\ -3 & 0 & 3 \\ -2 & 1 & 4 \end{bmatrix}, 2\right) \rightarrow \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\texttt{MOD}\left(\begin{bmatrix} -5 & -1 & 2 \\ -3 & 0 & 3 \\ -2 & 1 & 4 \end{bmatrix}, -3\right) \rightarrow \begin{bmatrix} -2 & -1 & 2 \\ 0 & 0 & 0 \\ -2 & 1 & 1 \end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

MODULO(A, P)

Description

Returns the modulo function.

Kind

Elemental function.

Arguments

`A`	A scalar or array of type `INTEGER` or `REAL`.
`P`	Must be of the same type and kind type as `A`.

Result Type

Same as `A`.

Result Value

Case 1 `A` is of type integer. If `P` is not equal to 0, the value of the result is `A - FLOOR(REAL(A)/REAL(P)) * P`. If `P` equals 0, the result is undefined.

Case 2 `A` is of type real. If `P` is not equal to 0, the result is `A - FLOOR(A/P) * P`. If `P` equals 0, the result is undefined.

`MODULO` always has the same sign as `P`.

Examples

```
MODULO(8, 5) → 3

MODULO(-8, 5) → 2

MODULO(8, -5) → -2

MODULO(-8, -5) → -3
```


Copyright © 1989–1994 Thinking Machines Corporation

MVBITS(FROM, FROMPOS, LEN, TO, TOPOS)

Description

Copies a subfield of bits from an integer value to a subfield of an integer variable.

Kind

Subroutine.

Arguments

`FROM` An expression of type integer.

`FROMPOS` An expression of type integer, with a nonnegative value satisfying the inequality `FROMPOS + LEN ≤ 32`.

`LEN` An expression of type integer.

`TO` A variable of type integer, to receive the result. (The same variable may be passed as an actual argument to both `TO` and `FROM`.)

`TOPOS` An expression of type integer, with a nonnegative value satisfying the inequality `TOPOS + LEN ≤ 32`.

`FROMPOS` specifies the starting position of a source subfield of `FROM` to be copied, and `TOPOS` specifies the starting position of a destination subfield of `TO` to receive a value.

A sequence of bits of length `LEN` is copied from the source subfield of value `FROM` to the destination subfield of variable `TO`. No other bits of `TO` are altered. On return from the procedure, the destination subfield of `TO` is equal to the value that the source subfield of `FROM` had on entry.

The low-order (right-most) bit of an integer value has position `0`, and the high-order (left-most) bit has position `31`.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Examples

In the examples below, N is a scalar integer variable whose value is given in binary and decimal representation, before and after references to MVBITS. Note that in the last reference, N is used in both the FROM and TO argument positions.

```
N → 00 000 000 000 000 000 000 010 000 000 111₂ → 1031₁₀
    CALL MVBITS ( FROM = -1, FROMPOS = 0, LEN = 3, TO = N, TOPOS = 4 )
N → 00 000 000 000 000 000 000 010 001 110 111₂ → 1143₁₀
    CALL MVBITS ( FROM = -1, FROMPOS = 7, LEN = 5, TO = N, TOPOS = 25 )
N → 00 111 110 000 000 000 000 010 001 110 111₂ → 1040188535₁₀
    CALL MVBITS ( FROM = N, FROMPOS = 3, LEN = 9, TO = N, TOPOS = 9 )
N → 00 111 110 000 000 010 001 110 001 110 111₂ → 1040260215₁₀
```


Copyright © 1989–1994 Thinking Machines Corporation

NEAREST(X, S)

Description

Returns the nearest different processor-representable number in a given direction.

Kind

Elemental function.

Arguments

X	A scalar or array of type REAL.
S	A scalar or array of type REAL. Must not equal zero.

Result Type

Same as X.

Result Value

The result has a value equal to the processor-representable number distinct from X and nearest to it in the direction of the infinity with the same sign as S.

Example

If X is represented by the model described in Section 17.15, then

NEAREST(3.0, 2.0) $\rightarrow 3+2^{-22}$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

NINT(A *[*, KIND *]*)

Description

Returns the integer nearest to a given real number.

Compatibility Note

When a program is not compiled with the `-f90syntax` switch, it is necessary to use the `KIND=` keyword if the `KIND` argument is used. If the `-f90syntax` switch is used, no keyword is necessary.

Kind

Elemental function.

Arguments

`A` An expression of type real.

`KIND` (optional)
A scalar integer constant expression indicating the length of the result type. When providing this argument, it is necessary to use the keyword `KIND=`, unless the code will be compiled with the `-f90syntax` switch.

Result Type

Integer. If `KIND` is present, the kind type parameter is 32 for an `INTEGER*4` result or 64 for an `INTEGER*8` result; otherwise, the kind type parameter is 32.

Result Value

The result is the integer nearest to `A`, rounded away from zero. If `A > 0`, the result is equal to `AINT(A+0.5)`. If `A ≤ 0`, the result is equal to `AINT(A-0.5)`.

The result is undefined if the processor cannot represent the result in the specified integer type.

Examples

```
NINT(0.999) → 1
NINT(-3.5) → -4
NINT(3.49d0) → 3
```


Copyright © 1989–1994 Thinking Machines Corporation

```
NINT(0.7, KIND=64) → 1_64
NINT(-4.6, KIND=64) → -5_64
NINT( [  0.6   27.9 ] ) ⇒ [ 1   28 ]
     [ -0.4   -5.6 ]      [ 0   -6 ]
```

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

NOT(I)

Description

Performs the bitwise logical complement of an integer value.

Kind

Elemental function.

Argument

`I` An expression of type integer.

Result Type

Integer.

Result Value

The result is the value obtained by complementing `I` bit-by-bit according to the following truth table:

I	NOT(I)
1	0
0	1

The low-order (right-most) bit of an integer value has position `0`. If the type is `INTEGER*4`, the high-order (left-most) bit has position `31`; if the type is `INTEGER*8`, the high-order bit has position `63`.

Examples

```
NOT(999)+1 → -999   ! twos complement negation
NOT( [0:10] ) → [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,-11]
```

$$\texttt{NOT(}\begin{bmatrix}0 & 3 & 6\\ 1 & 4 & 7\\ 2 & 5 & 8\end{bmatrix}\texttt{)} \rightarrow \begin{bmatrix}-1 & -4 & -7\\ -2 & -5 & -8\\ -3 & -6 & -9\end{bmatrix}$$

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

PACK(ARRAY, MASK *[*, VECTOR *]*)

Description

Gathers selected elements from an array of any rank into a rank-one array, under the control of a mask.

Kind

Transformational function.

Arguments

ARRAY An array of numeric or logical type.

MASK A logical scalar or array that is conformable with ARRAY.

VECTOR (optional)
A rank-one array of the same type as ARRAY. VECTOR must have at least as many elements as there are true elements in MASK. If MASK is a scalar with value true, VECTOR must have at least as many elements as there are in ARRAY.

Result Type

The result is an array of rank one with the same type as ARRAY. If VECTOR is present, the result size is that of VECTOR; otherwise, the result size is the number *t* of true elements in MASK, unless MASK is a scalar with the value true, in which case the result size is the size of ARRAY.

Result Value

The result is a rank-one array consisting of values from a subset of the elements of ARRAY selected by MASK, possibly supplemented by elements from VECTOR.

If *t* is the total number of true elements of MASK, then the value of element R(*i*) of a result array R is

the element of ARRAY corresponding to the *i*th true element of MASK, considering elements in array element order.

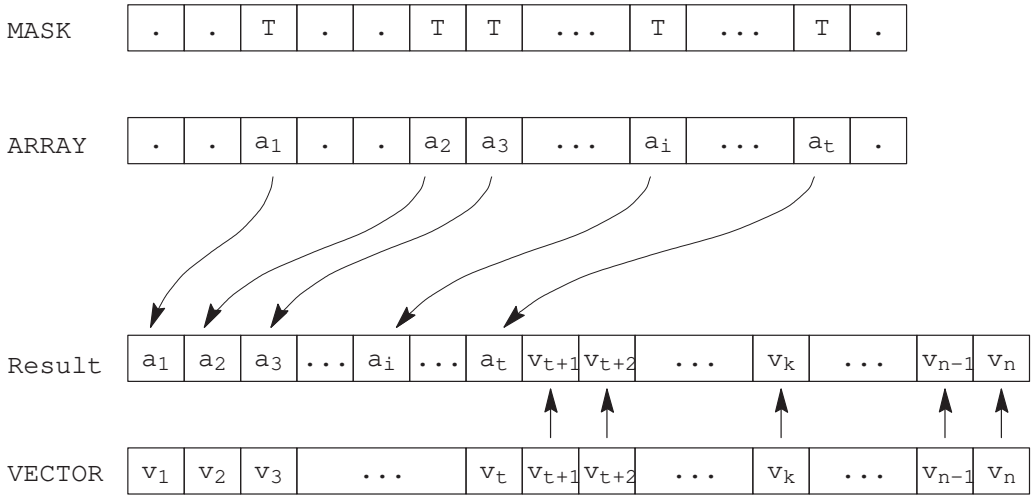
If VECTOR is present and has size $n > t$, then element R(*i*) of the result is

VECTOR(*i*)

for *i* ranging from *t*+1 to *n*.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

The action of the PACK function is illustrated graphically below:



Examples

If $A = \begin{bmatrix} 0 & -1 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 7 \end{bmatrix}$, the nonzero elements of A may be "gathered" by the function:

```
PACK( A, MASK = A .NE. 0 ) → [9,-1,7]
```

and the result can be "padded" with elements from a VECTOR argument:

```
PACK( A, A .NE. 0, [1:10] ) → [9,-1,7,4,5,6,7,8,9,10]
```

Finally, if $B = \begin{bmatrix} 7 & 14 & -9 \\ -5 & 23 & 56 \end{bmatrix}$, then

```
PACK( B, B>0 ) → [7,14,23,56]
```

and

```
PACK( B, B<0, [5[0]] ) → [-5,-9,0,0,0]
```


Copyright © 1989–1994 Thinking Machines Corporation

POPCNT(X)

Description

Returns the number of bits that contain one (the population count) in the bit-level representation of a value.

Kind

Elemental function.

Argument

X	A scalar or array of type logical or integer.

Result Type

The result is of type `INTEGER*4` for any argument of type `LOGICAL`, `INTEGER*4`, or `INTEGER*8`.

Result Value

The result is the number of bits in `X` that contain 1.

If `X` is an array argument, the result is an array of the same shape. Each element of the array returned contains the result of `POPCNT` for the corresponding element of `X`.

Examples

`POPCNT(3)` → 2

$$\texttt{POPCNT}\left(\begin{bmatrix}0 & 1 & 2\\ 3 & 4 & 5\\ 6 & 7 & 8\end{bmatrix}\right) \rightarrow \begin{bmatrix}0 & 1 & 1\\ 2 & 1 & 2\\ 2 & 3 & 1\end{bmatrix}$$

`POPCNT(0)` → 0

`POPCNT(.FALSE.)` → 0

`POPCNT(.TRUE.)` → 32

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

POPPAR(X)

Description

Returns 0 if the population parity of the argument value is even, 1 if odd.

Kind

Elemental function.

Argument

X	A scalar or array of type logical or integer.

Result Type

The result is of type INTEGER*4 for any argument of type LOGICAL, INTEGER*4, or INTEGER*8.

Result Value

The result is 0 if the parity of X is even or 1 if the parity of X is odd.

If X is an array argument, the result is an array of the same shape. Each element of the array returned contains the result of POPPAR for the corresponding element of X.

Examples

```
POPPAR(4) → 1
```

$$\texttt{POPPAR}\left(\begin{bmatrix}0 & 1 & 2\\ 3 & 4 & 5\\ 6 & 7 & 8\end{bmatrix}\right) \rightarrow \begin{bmatrix}0 & 1 & 1\\ 0 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}$$

```
POPPAR(0) → 0

POPPAR(.FALSE.) → 0

POPPAR(.TRUE.) → 0
```


Copyright © 1989–1994 Thinking Machines Corporation

PRECISION(X)

Description

Returns the decimal precision in the model representing real numbers of the same kind type as the argument.

Kind

Inquiry function.

Argument

`X` A scalar or array of type `REAL` or `COMPLEX`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result has the value `INT(LOG10(`2^{p-1}`))`, where p is as defined in Section 17.15 for the model representing real numbers of the same kind type as `X`.

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
`REAL*4`	6
`REAL*8`	15
`COMPLEX*8`	6
`COMPLEX*16`	15

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

PRODUCT(ARRAY *[*, DIM *]* *[*, MASK *]*)

Description

Computes the product of all elements of a numeric array of rank one, or computes the product of each of the complete rank-one sections parallel to a given dimension of an array of rank two or greater, possibly under the control of a mask.

Kind

Transformational (reduction) function.

Arguments

`ARRAY` An array of numeric type. It must not be scalar.

`DIM` (optional)
A *constant* scalar expression of type integer, with a value in the range $1 \le$ `DIM` $\le n$, where n is the rank of `ARRAY`.

`MASK` (optional)
An array or scalar expression of type logical; if an array, it must be conformable with `ARRAY`.

Result Type

The result has the same type as `ARRAY`. It is a scalar if `DIM` is absent or if `ARRAY` has rank one; otherwise, it is an array with shape equal to that of `ARRAY` with dimension `DIM` removed.

Result Value

Case 1a `PRODUCT(ARRAY [, MASK = MASK ])`
`ARRAY` has any rank, `DIM` is absent

or

Case 1b `PRODUCT(ARRAY, DIM = 1 [, MASK ])`
`ARRAY` has rank `1`, `DIM` is present:

The result is a scalar equal (approximately) to the product of the elements of `ARRAY` corresponding to the true elements of `MASK`, if present. If `MASK` is absent, the result is the product of all elements of `ARRAY`.

Copyright © 1989–1994 Thinking Machines Corporation

If ARRAY is a zero-sized array, or if MASK is present and no elements of MASK are true, the result is the scalar value 1.

Case 2 PRODUCT(ARRAY, DIM [, MASK])
ARRAY has rank $n > 1$, DIM is present:

For an ARRAY of rank n with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM}, d_{DIM+1}, ..., d_n]

the result is an array of rank $n-1$ with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM+1}, ..., d_n].

The result is formed by computing the product of each of the complete rank-one sections of ARRAY that lie parallel to dimension DIM . If MASK is present, only elements of ARRAY corresponding to the true elements of MASK are used in the computation; if MASK is absent, all elements of ARRAY are used. Thus, the value of element

R(s_1, s_2, ..., s_{DIM-1}, s_{DIM+1}, ..., s_n)

of a result array R is equal to

PRODUCT($ARRAY_{DIM}$ [, MASK = $MASK_{DIM}$])

where $ARRAY_{DIM}$ and $MASK_{DIM}$ are complete rank-one sections of ARRAY and MASK, both with shape [d_{DIM}]. $ARRAY_{DIM}$ is the array section

ARRAY(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n)

and $MASK_{DIM}$ is the array section

MASK(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n) .

If MASK is present and all its elements are false, or if only dimension DIM of ARRAY is zero-sized, the result is an array with all elements equal to 1. If any dimension of ARRAY other than DIM is zero-sized, the result is a zero-sized array.

Examples

Case 1a ARRAY has any rank, DIM is absent:

If A = [2,3,-5,7], then

PRODUCT(A) → -210

PRODUCT(ARRAY = A, MASK = A .GT. 0) → 42

$$\text{PRODUCT}\left(\begin{bmatrix} -4 & 2 & -1 \\ 3 & 5 & -2 \end{bmatrix}, \text{MASK} = \begin{bmatrix} T & T & . \\ . & T & T \end{bmatrix}\right) \rightarrow 80$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 1b `ARRAY` has rank `1`, `DIM` is present:

```
PRODUCT( [2,3,-5,7], DIM = 1 ) → -210
```

If `A = [2,3,-5,7]` and `M = [T,.,.,T]`, then

```
PRODUCT( ARRAY = A, DIM = 1, MASK = M ) → 14
```

```
PRODUCT( [1:0], DIM = 1 ) → 1
```

Case 2 `ARRAY` has rank $n > 1$, `DIM` is present:

If $\mathtt{B} = \begin{bmatrix} 3 & 2 & 5 \\ 4 & 1 & -7 \end{bmatrix}$, then

```
PRODUCT( ARRAY = B, DIM = 1 ) → [12,2,-35]
```

If $\mathtt{B} = \begin{bmatrix} 3 & 2 & 5 \\ 4 & 1 & -7 \end{bmatrix}$ and $\mathtt{M} = \begin{bmatrix} T & T & . \\ . & T & T \end{bmatrix}$, then

```
PRODUCT( ARRAY = B, DIM = 2, MASK = M ) → [6,-7]
```

If $\mathtt{B} = \begin{bmatrix} 3 & 2 & 5 \\ 4 & 1 & -7 \end{bmatrix}$ and $\mathtt{M} = \begin{bmatrix} . & . & . \\ T & T & T \end{bmatrix}$, then

```
PRODUCT( ARRAY = B, DIM = 1, MASK = M ) → [4,1,-7]
```

and

```
PRODUCT( ARRAY = B, DIM = 2, MASK = M ) → [1,-28]
```


Copyright © 1989–1994 Thinking Machines Corporation

PROJECT(ARRAY, MASK, FIELD *[*, DIM *]*)

Description

Projects a selected array element from an array onto a scalar, or projects selected elements from each of the complete rank-one sections parallel to a given dimension of an array of rank two or greater, with element selection under the control of a mask. A default value is chosen if no source array element is selected.

Kind

Transformational function.

Arguments

`ARRAY` An array of numeric or logical type.

`MASK` An array of type logical with the same shape as `ARRAY`. If `DIM` is absent, `MASK` must have at most one true element; if `DIM` is present, each complete rank-one section along dimension `DIM` of `MASK` must have at most one true element.

`FIELD` A scalar or array of the same type as `ARRAY`. It must be scalar if `DIM` is absent. If `DIM` is present, `FIELD` must have shape equal to that of `ARRAY` with dimension `DIM` removed.

`DIM` (optional)
A scalar *constant* expression of type integer, with value in the range $1 \leq$ `DIM` $\leq n$, where n is the rank of `ARRAY`.

Result Type

The result has the same type as `ARRAY`. It is scalar if `DIM` is absent or `ARRAY` has rank one; otherwise, the result is an array with shape equal to that of `ARRAY` with dimension `DIM` removed.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Result Value

Case 1a `PROJECT( ARRAY, MASK, FIELD, DIM = 1 )`
`ARRAY` has rank one, `FIELD` is scalar, `DIM` is present

or

Case 1b `PROJECT( ARRAY, MASK, FIELD )`
`ARRAY` has any rank, `FIELD` is scalar, `DIM` is absent:

`MASK` must have at most one true element. The result is a scalar with the value of the element of `ARRAY` corresponding to the true element of `MASK` if there is one, and is the value of `FIELD` otherwise. If `MASK` has zero size, the result is `FIELD`.

Case 2 `PROJECT( ARRAY, MASK, FIELD, DIM )`
`ARRAY` has rank $n > 1$, `FIELD` is an array, `DIM` is present:

If `ARRAY`, `MASK`, and `FIELD` have shape

$[\ d_1, d_2, \ldots, d_{\mathtt{DIM}-1}, d_{\mathtt{DIM}}, d_{\mathtt{DIM}+1}, \ldots, d_n\ ]$

then the result is an array of rank $n-1$ with shape

$[\ d_1, d_2, \ldots, d_{\mathtt{DIM}-1}, d_{\mathtt{DIM}+1}, \ldots, d_n\ ]$.

Each complete rank-one section of `MASK` along dimension `DIM` must have at most one true element.

The result is formed by projecting each of the complete rank-one sections of `ARRAY` that lie parallel to dimension `DIM`. Thus, the value of element

$\mathtt{R}(s_1, s_2, \ldots, s_{\mathtt{DIM}-1}, s_{\mathtt{DIM}+1}, \ldots, s_n)$

of a result array `R` is equal to

$\mathtt{PROJECT}(\ \mathtt{ARRAY}_{\mathtt{DIM}}, \mathtt{MASK}_{\mathtt{DIM}}, \mathtt{FIELD}_{\mathtt{VAL}}\ )$

where $\mathtt{ARRAY}_{\mathtt{DIM}}$ and $\mathtt{MASK}_{\mathtt{DIM}}$ are complete rank-one sections of `ARRAY` and `MASK`, both with shape $[d_{\mathtt{DIM}}]$. $\mathtt{ARRAY}_{\mathtt{DIM}}$ is the array section

$\mathtt{ARRAY}(s_1, s_2, \ldots, s_{\mathtt{DIM}-1}, :, s_{\mathtt{DIM}+1}, \ldots, s_n\ )$

and $\mathtt{MASK}_{\mathtt{DIM}}$ is the array section

$\mathtt{MASK}(s_1, s_2, \ldots, s_{\mathtt{DIM}-1}, :, s_{\mathtt{DIM}+1}, \ldots, s_n\ )$

while $\mathtt{FIELD}_{\mathtt{VAL}}$ is the scalar array element

$\mathtt{FIELD}(s_1, s_2, \ldots, s_{\mathtt{DIM}-1}, s_{\mathtt{DIM}+1}, \ldots, s_n\ )$

Note that if `ARRAY` has zero size because $d_{\mathtt{DIM}}$ has value zero, the result may have nonzero size with all its values coming from `FIELD`.

Copyright © 1989–1994 Thinking Machines Corporation

Examples

Case 1a ARRAY has rank one, FIELD is scalar, DIM is present:

If V = [1,2,3,4] and P = [.,.,*T*,.], then

PROJECT(V, MASK = P, FIELD = 0) → 3

and

PROJECT(V, MASK = V .GT .5, FIELD = 99) → 99

Case 1b ARRAY has any rank, FIELD is scalar, DIM is absent:

$$\texttt{PROJECT(}\begin{bmatrix}1&4&7&10\\2&5&8&11\\3&6&9&12\end{bmatrix}\texttt{, MASK =}\begin{bmatrix}.&.&.&.\\.&.&T&.\\.&.&.&.\end{bmatrix}\texttt{, FIELD = 0)} \rightarrow 8$$

Case 2 ARRAY has rank $n > 1$, FIELD is an array, DIM is present:

If $\texttt{A} = \begin{bmatrix}1&4&7&10\\2&5&8&11\\3&6&9&12\end{bmatrix}$ and $\texttt{L} = \begin{bmatrix}.&.&.&.\\.&.&T&.\\.&.&.&.\end{bmatrix}$, then

PROJECT(A,L, [0,0,0], DIM = 2) → [0,8,0]

and

PROJECT(A, L, [0,0,0,0], DIM = 1) → [0,0,8,0].

The first nonzero number in each column of the table

$\texttt{TABLE} = \begin{bmatrix}0&0&0&0\\0&2&0&0\\0&3&5&0\\1&4&6&0\end{bmatrix}$ is located by the mask $\texttt{M} = \begin{bmatrix}.&.&.&.\\.&T&.&.\\.&.&T&.\\T&.&.&.\end{bmatrix}$.

A vector that contains those nonzero numbers can be extracted from TABLE by the PROJECT function. Thus, the value of

PROJECT(TABLE, M, [-1,-1,-1,-1], DIM = 1)

is that vector, namely [1,2,5,-1] .

Note that M itself is the value of

FIRSTLOC(TABLE .NE. 0, DIM = 1) .

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

RADIX(X)

Description

Returns the base of the model representing numbers of the same type and kind type as the argument.

Kind

Inquiry function.

Argument

`X` A scalar or array of type `INTEGER` or `REAL`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
`INTEGER*4`	2
`INTEGER*8`	2
`REAL*4`	2
`REAL*8`	2

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

RANDOM_NUMBER(HARVEST)

Description

Returns one pseudo random number or an array of pseudo random numbers from the uniform distribution over the range `0 ≤ x < 1`.

Kind

Subroutine.

Argument

`HARVEST` A scalar or CM array of type `REAL` or `DOUBLE PRECISION`. It is an `INTENT(OUT)` argument.

Examples

```
REAL X, Y (10, 10)
CALL RANDOM_NUMBER (HARVEST = X)
! INITIALIZES X WITH A PSEUDO RANDOM NUMBER
CALL RANDOM_NUMBER (Y)
! X AND Y CONTAIN UNIFORMLY DISTRIBUTED RANDOM NUMBERS
```

Note

The performance of this subroutine is significantly improved if the program is linked with the CM Scientific Software Library, `-lcmssl`.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

RANDOM_SEED(*[* SIZE *]* *[*, PUT *]* *[*, GET *]*)

Description

Restarts or queries the pseudo random number generator.

Kind

Subroutine.

Arguments

There must be exactly one or no argument present, i.e., no more than one of the arguments listed below. If no argument is present, the processor sets the seed to a processor-dependent value.

`SIZE` (optional)
: The number of integers used to hold the value of the seed. `SIZE` is always 1.

`PUT` (optional)
: Sets the seed value for both the CM and front-end generators.

`GET` (optional)
: `INTEGER*4` array of rank one, set to the current value of the seed.

(Each of these arguments specifies a different incompatible operation — one or none.)

Copyright © 1989–1994 Thinking Machines Corporation

RANGE(X)

Description

Returns the decimal exponent range in the model representing integer or real numbers of the same kind type as the argument.

Kind

Inquiry function.

Argument

`X` A scalar or array of type `INTEGER`, `REAL`, or `COMPLEX`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

Case 1 For an integer argument, the result has the value `INT` (`LOG10` (*huge*)), where *huge* is the largest positive integer in the model representing integer numbers of the same kind type as `X` (Section 17.15).

Case 2 For a real or complex argument, the result has the value `INT` (`MIN` (`LOG10` (*huge*) – `LOG10` (*tiny*))), where *huge* and *tiny* are the largest and smallest positive numbers in the model representing real numbers of the same kind type as `X` (Section 17.15).

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
`INTEGER*4`	9
`INTEGER*8`	18
`REAL*4`	37
`REAL*8`	307
`COMPLEX*8`	37
`COMPLEX*16`	307

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

RANK(SOURCE)

Description

Returns the rank of an array or a scalar.

Kind

Inquiry function.

Argument

SOURCE A scalar or array of any type.

Result Type

A scalar of type INTEGER*4.

Result Value

The result is the rank of SOURCE, which is zero if SOURCE is a scalar.

Examples

If N is a named constant and A is a seven-dimensional array, then

```
RANK(A) → 7
```

Also,

```
RANK(13) → 0
RANK([1:10]) → 1
RANK( RESHAPE( MOLD=[3,4,5], SOURCE=[1:60] ) ) → 3
RANK( [ 1  4  7  10 ] ) → 2
      [ 2  5  8  11 ]
      [ 3  6  9  12 ]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

REAL(A *[*, MOLD *]*)

Description

Converts a numeric argument to type real, with either single or double precision.

Compatibility Note

For Fortran 90 compatibility, `REAL` accepts the keyword `KIND` for the second argument when a program is compiled with the `-f90syntax` switch.

Invoked with:	**`REAL` is interpreted as:**
`-f90syntax`	`REAL(A [, KIND])`
`-nof90syntax` (default)	`REAL(A [, MOLD])`

Kind

Elemental function.

Arguments

`A` An expression of numeric type.

`MOLD` (optional)
An expression of type real or complex.

Result Type

The result is of type single-precision real if `MOLD` is absent, or if `MOLD` is present with single precision; the result is double-precision real if `MOLD` is present with double precision.

The result is undefined if the processor cannot represent the result in the specified type.

Copyright © 1989–1994 Thinking Machines Corporation

Result Value

Case 1 `REAL( A [, MOLD ] )`
`MOLD` is absent, or present with single precision:

The value of the single-precision real result depends on the type of `A`, as given in the following table:

Type of A	Result Value
`INTEGER*4`	`A` approximated as single precision
`INTEGER*8`	`A` approximated as single precision
`REAL`	`A` unchanged
`DOUBLE PRECISION`	`A` narrowed to single precision
`COMPLEX`	real part of `A`
`DOUBLE COMPLEX`	real part of `A` narrowed to single precision

Case 2 `REAL( A, MOLD )`
`MOLD` is present with double precision:

The value of the double-precision real result depends on the type of `A`, as given in the following table:

Type of A	Result Value
`INTEGER*4`	`A` converted to double precision
`INTEGER*8`	`A` approximated as double precision
`REAL`	`A` widened to double precision
`DOUBLE PRECISION`	`A` unchanged
`COMPLEX`	real part of `A` widened to double precision
`DOUBLE COMPLEX`	real part of `A`

Examples

Case 1 `MOLD` is absent, or present with single precision:

```
REAL(10) → 10.
REAL(10,0.) → 10.
REAL((40,50)) → 40.
```

$$\texttt{REAL}\left(\begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}\right) \rightarrow \begin{bmatrix}1. & 3.\\ 2. & 4.\end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 2 `MOLD` is present with double precision:

```
REAL(10,0.d0) ⟶ 10.d0
REAL(-10.,0.d0) ⟶ -10.d0
REAL( (40,50), 0.d0 ) ⟶ 40.d0
REAL([1 3],0.d0) ⟶ [1.d0 3.d0]
                   [2.d0 4.d0]
```

Note

The `INTEGER*8` type is supported only on CM-5 systems with vector units.

Copyright © 1989–1994 Thinking Machines Corporation

REPLICATE(ARRAY, DIM, NCOPIES)

Description

Extends an array along a dimension by replicating copies of the array along the dimension.

Kind

Transformational function.

Arguments

ARRAY	An array of numeric or logical type.
DIM	A scalar *constant* expression of type integer, with value in the range $1 \le \texttt{DIM} \le n$, where n is the rank of ARRAY.
NCOPIES	A scalar expression of type integer.

Result Type

The result has the same type and rank as the ARRAY argument.

Result Value

The result of REPLICATE(ARRAY, DIM, NCOPIES) for an ARRAY of rank n with shape

$$[\ d_1, \ldots, d_{\texttt{DIM}-1}, d_{\texttt{DIM}}, d_{\texttt{DIM}+1}, \ldots, d_n\ ]$$

is an array of the same rank with shape

$$[\ d_1, \ldots, d_{\texttt{DIM}-1}, d_{\texttt{DIM}} * k, d_{\texttt{DIM}+1}, \ldots, d_n\ ]$$

where k = MAX(NCOPIES,0).

Thus, the value of element

$$\texttt{R}(\ s_1, \ldots, s_{\texttt{DIM}-1}, s_{\texttt{DIM}}, s_{\texttt{DIM}+1}, \ldots, s_n\ )$$

of a result array R is equal to

$$\texttt{ARRAY}(\ s_1, \ldots, s_{\texttt{DIM}-1}, t, s_{\texttt{DIM}+1}, \ldots, s_n\ )$$

where t = MOD($s_{\texttt{DIM}}$, $d_{\texttt{DIM}}$) + 1

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Examples

If $\texttt{A} = \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}$, then

$$\texttt{REPLICATE(A, DIM = 2, NCOPIES = 3)} \rightarrow \begin{bmatrix} 2 & 3 & 2 & 3 & 2 & 3 \\ 3 & 4 & 3 & 4 & 3 & 4 \end{bmatrix}$$

and

$$\texttt{REPLICATE(A, DIM = 1, NCOPIES = 2)} \rightarrow \begin{bmatrix} 2 & 3 \\ 3 & 4 \\ 2 & 3 \\ 3 & 4 \end{bmatrix}$$

Note that for an argument of `NCOPIES = 1`, the result is the same as the input; for a value of `0`, the result is a zero-sized array.

Copyright © 1989–1994 Thinking Machines Corporation

RESHAPE(MOLD, SOURCE *[*, PAD *]* *[*, ORDER *]*)

Description

Creates an array with specified shape from elements of a source array, with additional elements supplied from an optional padding array, if necessary. The order of elements of the result array may be permuted with respect to the element order of the source array.

Compatibility Note

RESHAPE becomes compatible with Fortran 90 when a program is compiled with the -f90syntax switch. Two changes take place: the keyword SHAPE is used instead of the equivalent MOLD, and the argument order is changed. When the function is invoked without argument keywords, CM Fortran interprets the argument order as:

Invoked with:	**RESHAPE is interpreted as:**
-f90syntax	RESHAPE(SOURCE, SHAPE *[*,PAD*]* *[*,ORDER*]*)
-nof90syntax (default)	RESHAPE(MOLD, SOURCE *[*,PAD*]* *[*,ORDER*]*)

The use of keywords allows the arguments to appear in any order; however, RESHAPE requires different keywords, depending on whether the program is compiled with -f90syntax.

Kind

Transformational function.

Arguments

MOLD An integer array of rank one, with a size in the range 1 to 31. It must not have an element whose value is negative.

SOURCE An array of numeric or logical type. If PAD is absent or of size zero, the size of SOURCE must be greater than or equal to PRODUCT(MOLD). The size of the result is the product of the values of the elements of MOLD.

PAD (optional)
An array of the same type as SOURCE.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

ORDER (optional)

An array of type integer, with the same shape as MOLD. Its value must be a permutation of [1:*n*], where *n* is the size of MOLD. If absent, it is as if it were present with value [1:*n*].

Result Type

The result has the same type as SOURCE and the shape specified by MOLD, so that

```
SHAPE( RESHAPE( MOLD, SOURCE, PAD, ORDER ) )
```

is equal to MOLD.

Result Value

The elements of the result array, taken in permuted subscript order

```
ORDER(1),...,ORDER(n)
```

are those of SOURCE in normal array element order followed (if necessary) by those of PAD in array element order, followed (if necessary) by additional copies of PAD in array element order.

Examples

```
RESHAPE( MOLD=[2,3], SOURCE = [1:6] ) →
```

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

```
RESHAPE(MOLD=[2,4],SOURCE=[1:5],PAD=[-7,-8],ORDER=[2,1]) →
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & -7 & -8 & -7 \end{bmatrix}$$

```
RESHAPE(MOLD=[3,2], SOURCE=
```

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

```
,ORDER=[2,1]) →
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

```
! aka TRANSPOSE
```

```
RESHAPE( MOLD=[4,5], SOURCE=
```

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

```
, PAD =
```

$$\begin{bmatrix} -1 & -4 \\ -2 & -5 \\ -3 & -6 \end{bmatrix}$$

```
) →
```

$$\begin{bmatrix} 1 & 5 & -1 & -5 & -3 \\ 2 & 6 & -2 & -6 & -4 \\ 3 & 7 & -3 & -1 & -5 \\ 4 & 8 & -4 & -2 & -6 \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

If C = $\begin{bmatrix}\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \\ \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}\end{bmatrix}$, then

```
RESHAPE(MOLD=[5,4], SOURCE=C, ORDER=[2,1]) →
```
$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \end{bmatrix}$

```
 RESHAPE(MOLD=[3,4,2], SOURCE=[1:15], PAD=[-16:-23:-1],
&        ORDER=[1,3,2]) →
```
$\begin{bmatrix}\begin{bmatrix} 1 & 7 & 13 & -19 \\ 2 & 8 & 14 & -20 \\ 3 & 9 & 15 & -21 \end{bmatrix} \\ \begin{bmatrix} 4 & 10 & -16 & -22 \\ 5 & 11 & -17 & -23 \\ 6 & 12 & -18 & -16 \end{bmatrix}\end{bmatrix}$

```
 RESHAPE(MOLD=[3,4,2], SOURCE=[1:15], PAD=[-16:-23:-1],
&        ORDER=[3,2,1]) →
```
$\begin{bmatrix}\begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ -17 & -19 & -21 & -23 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 & 6 & 8 \\ 10 & 12 & 14 & -16 \\ -18 & -20 & -22 & -16 \end{bmatrix}\end{bmatrix}$

```
 RESHAPE(MOLD=[3,4,2], SOURCE=[1:15], PAD=[-16:-23:-1],
&        ORDER=[3,1,2]) →
```
$\begin{bmatrix}\begin{bmatrix} 1 & 7 & 13 & -19 \\ 3 & 9 & 15 & -21 \\ 5 & 11 & -17 & -23 \end{bmatrix} \\ \begin{bmatrix} 2 & 8 & 14 & -20 \\ 4 & 10 & -16 & -22 \\ 6 & 12 & -18 & -16 \end{bmatrix}\end{bmatrix}$

Copyright © 1989–1994 Thinking Machines Corporation

RRSPACING(X)

Description

Returns the reciprocal of the relative spacing of model numbers near the argument value.

Kind

Elemental function.

Argument

X A scalar or array of type REAL.

Result Type

Same as X.

Result Value

The result has the value $|X \times 2^{-e}| \times 2^{p}$, where e and p are as defined in Section 17.15 for the model representation of X, provided this result is within range.

Spacing is the same for all numbers with a given exponent. Since those numbers can differ greatly in size, RRSPACING provides information about the relative spacing.

Example

If X is represented by the model described in Section 17.15, then

```
RRSPACING(-3.0) → .75 x 2^24
```


Copyright © 1989–1994 Thinking Machines Corporation

SCALE(X, I)

Description

Multiplies a real times its base to an integer power.

Kind

Elemental function.

Arguments

`X`	A scalar or array of type `REAL`.
`I`	A scalar or array of type `INTEGER`, conformable to `X`.

Result Type

Same as `X`.

Result Value

The result has the value $X \times 2^{I}$, provided this result is within range.

Example

```
SCALE(3.0, 2) → 12.0
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SELECTED_INT_KIND(R)

Description

Returns the smallest value of the kind type parameter of an integer data type that represents all integer values n with $-10^R < n < 10^R$.

Kind

Transformational function.

Argument

`R` A scalar of type `INTEGER`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result has a value equal to the smallest value of the kind type parameter of an integer data type that represents all values n in the range of values n with $-10^R < n < 10^R$, or if no such kind type is available on the processor, the result is –1.

Copyright © 1989–1994 Thinking Machines Corporation

SELECTED_REAL_KIND(*[*P*] [*, R*]*)

Description

Returns the smallest value of the kind type parameter of a real data type with decimal precision of at least `P` digits and a decimal exponent range of at least `R`.

Kind

Transformational function.

Arguments

`P` (optional)
A scalar of type `INTEGER`.

`R` (optional)
A scalar of type `INTEGER`.

Result Type

Scalar of type `INTEGER*4`.

Result Value

The result has a value equal to the smallest value of the kind type parameter of a real data type with decimal precision, as returned by the function `PRECISION`, of at least `P` digits and a decimal exponent range, as returned by the function `RANGE`, of at least `R`. If no such kind type is available on the processor, the result is –1 if the precision is not available, –2 if the exponent range is not available, and –3 if neither is available.

Note

At least one argument must be present.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SET_EXPONENT(X, I)

Description

Returns the model number whose fractional part is the fractional part of the model representation of X and whose exponent part is I.

Kind

Elemental function.

Arguments

X	A scalar or array of type REAL.
I	A scalar or array of type INTEGER, conformable to X.

Result Type

Same as X.

Result Value

The result has the value X x 2^{I-e}, where *e* is as defined in Section 17.15 for the model representation of X, provided this result is within range. If X is zero, the result is zero.

Example

If X is represented by the model described in Section 17.15, then

```
SET_EXPONENT(3.0, 1) → 1.5
```


Copyright © 1989–1994 Thinking Machines Corporation

SHAPE(SOURCE)

Description

Returns the shape of an array or a scalar.

Compatibility Note

For back compatibility, CM Fortran also accepts the intrinsic function DSHAPE, which is equivalent to SHAPE.

Kind

Inquiry function.

Argument

SOURCE — A scalar or array of numeric or logical type. It must not be an assumed-size array.

Result Type

The result is of type INTEGER*4. It is a CM array of rank one whose size is equal to the rank of SOURCE.

Result Value

The value of the result is the shape of SOURCE. For an ARRAY of rank n, the result is a CM array with the value

$[e_1, e_2, \ldots, e_n]$

where e_i is the extent of the ith dimension of SOURCE, for $i = 1, 2, \ldots, n$.

Examples

If A is declared by the statement

```
REAL A(7,-5:5,0:9)
```

then

```
SHAPE( A ) → [7,11,10]
SHAPE( A(:,4,:) ) → [7,10]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Also,

```
SHAPE( [1:10] ) → [10]
SHAPE(17) →[1:0]     ! (the zero-size array of rank 1)
SHAPE(SHAPE(17)) → [0]
```


Copyright © 1989–1994 Thinking Machines Corporation

SIGN(A, B)

Description

Given two numbers, returns a number with the magnitude of the first and the sign of the second.

Kind

Elemental function.

Arguments

`A`	An expression of type integer or real.
`B`	An expression of the same type as `A`.

Result Type

Same type as `A`.

Result Value

The value of the result is $|A|$ if $B \geq 0$ and $-|A|$ if $B < 0$.

Examples

```
SIGN( 17.1, -3. ) → -17.1

SIGN( 17.1, 3. ) → 17.1
```

$$\texttt{SIGN(}\begin{bmatrix} -5 & -2 & 8 \\ 7 & -4 & 3 \end{bmatrix}\texttt{,}\begin{bmatrix} -3 & 0 & 7 \\ -1 & -5 & 4 \end{bmatrix}\texttt{)} \rightarrow \begin{bmatrix} -5 & 2 & 8 \\ -7 & -4 & 3 \end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SIN(X)

Description

Computes the sine trigonometric function.

Kind

Elemental function.

Argument

X An expression of type real or complex.

Result Type

Same as X.

Result Value

The result has the value equal to an approximation to the `sin(X)` trigonometric function. If X is of type real, it is regarded as a value in radians. For complex values of X with the form `(a,b)`, the result is calculated using the identity

```
SIN(X) = CMPLX( SIN(a)*COSH(b), COS(a)*SINH(b), MOLD = X )
```

The real part *a* is regarded as a value in radians.

Examples

```
SIN(0.) ⇒ 0.
SIN(ASIN(0.25d0)) → 0.25d0
SIN((π,0)) → (0,0)
SIN((π,1)) ⇒ (0,-1.175201)
```

$$\mathtt{SIN}\left(\begin{bmatrix} 0.52359 & 1.57080 \\ 1.04720 & 2.35619 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} 0.50000 & 1.00000 \\ 0.86603 & 0.70711 \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

SINH(X)

Description

Computes the hyperbolic sine function.

Kind

Elemental function.

Argument

X	An expression of type real.

Result Type

Same as X.

Result Value

The result is an approximation to the `sinh(X)` mathematical function.

Examples

```
SINH(0.) → 0.

SINH( [0.5  2.0  5.0]  ) ⇒ [0.52109  3.62686  74.2032]
     [1.0  3.0  9.0]       [1.17520  10.0179  4051.54]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SIZE(ARRAY *[*, DIM *]*)

Description

Returns the total number of elements in an array, or the extent of an array along a specified dimension.

Compatibility Note

For back compatibility, CM Fortran also accepts the intrinsic function `DSIZE`, which is equivalent to `SIZE`.

Kind

Inquiry function.

Arguments

`ARRAY` An array of numeric or logical type. It must not be an assumed-size array.

`DIM` (optional)
A scalar expression of type integer, with a value in the range $1 \leq$ `DIM` $\leq n$, where n is the rank of `ARRAY`.

Result Type

`INTEGER*4` scalar.

Result Value

The result is the total number of elements of `ARRAY` if `DIM` is absent, and is the extent of dimension `DIM` of `ARRAY` if `DIM` is present.

Examples

If `A` is declared by the statement

```
REAL B(7,-5:5,0:9)
```

then

```
SIZE(B) → 770
SIZE(B,DIM=1) → 7
SIZE(B,DIM=3) → 10
```


Copyright © 1989–1994 Thinking Machines Corporation

```
SIZE( B(2:5,-1:1,:), DIM = 2 ) → 3
SIZE( B(2:5,0,0) ) → 4
```

Also,

```
SIZE( [1:0] ) → 0
SIZE( [1:10] ) → 10
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SPACING(X)

Description

Returns the absolute spacing of model numbers near the argument value.

Kind

Elemental function.

Argument

X A scalar or array of type REAL.

Result Type

Same as X.

Result Value

The result has the value 2^{e-p}, where e and p are as defined in Section 17.15 for the model representation of X, provided this result is within range. Otherwise, the result is the same as that of TINY(X).

The result is the representable difference between two successive numbers near X. This spacing is constant for all arguments with a given exponent.

Example

If X is represented by the model described in Section 17.15, then

```
SPACING(3.0) → 2^-22
```


Copyright © 1989–1994 Thinking Machines Corporation

SPREAD(SOURCE, DIM, NCOPIES)

Description

Broadcasts copies of a source array (or scalar) along a dimension (as in forming a book from copies of a single page), yielding an array of rank one greater than the source, with values replicated from the source.

Kind

Transformational function.

Arguments

`SOURCE` A scalar or array of numeric or logical type, with rank less than 31.

`DIM` A scalar *constant* expression of type integer, with value in the range $1 \leq \texttt{DIM} \leq n+1$, where n is the rank of `SOURCE`.

`NCOPIES` A scalar of type integer.

Result Type

The result is an array with the same type as `SOURCE`.

Result Value

Case 1 `SOURCE` is a scalar:

The result is a rank-one array of length `MAX(NCOPIES,0)`. Each element of the result array is equal to `SOURCE`.

Case 2 `SOURCE` is an array:

If `SOURCE` has rank n and shape

$[d_1, d_2, \ldots, d_{\texttt{DIM}-1}, d_{\texttt{DIM}}, d_{\texttt{DIM}+1}, \ldots, d_n]$

the shape of the result is an array of rank $n+1$ and shape

$[d_1, d_2, \ldots, d_{\texttt{DIM}-1}, k, d_{\texttt{DIM}}, d_{\texttt{DIM}+1}, \ldots, d_n]$

where k = `MAX(NCOPIES,0)`

The value of element

$\texttt{R}(s_1, s_2, \ldots, s_{\texttt{DIM}-1}, s_{\texttt{DIM}}, s_{\texttt{DIM}+1}, \ldots, s_{n+1})$

of a result array `R` is equal to

$\texttt{SOURCE}(s_1, s_2, \ldots, s_{\texttt{DIM}-1}, s_{\texttt{DIM}+1}, \ldots, s_{n+1})$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Examples

Case 1 `SOURCE` is a scalar:

```
SPREAD( 7, DIM = 1, NCOPIES = 5 ) → [7,7,7,7,7]

SPREAD( 13, DIM = 1, NCOPIES = -2 ) → [1:0]
                                    ! (the zero-sized array of rank 1)
```

Case 2 `SOURCE` is an array:

If `A = [2,3,4]`, then

```
SPREAD( A, DIM = 1, NCOPIES = 2 ) →
```

$$\begin{bmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \end{bmatrix}$$

and

```
SPREAD( A, DIM = 2, NCOPIES = 4 ) →
```

$$\begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

SQRT(X)

Description

Computes the square root of a number.

Kind

Elemental function.

Argument

X — An expression of type real or complex. If X is real, its value must not be negative.

Result Type

Same as X.

Result Value

The result is an approximation to the square root of X. If X is of type complex, the result is a principal value in one of the forms:

(*a*,*b*) where $a \geq 0$

(0,*b*) where $b \geq 0$

Examples

```
SQRT(4.0) → 2.0
SQRT(-25.) →  ? ! undefined
SQRT((-25,0)) → (0,5)
SQRT((3,4)) → (2,1)
```

$$\texttt{SQRT}\left(\begin{bmatrix}10. & 2.\\ 1. & 4.\end{bmatrix}\right) \Rightarrow \begin{bmatrix}3.16228 & 1.41421\\ 1.00000 & 2.00000\end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

SUM(ARRAY *[*, DIM *] [*, MASK *]*)

Description

Computes the sum of all elements of a numeric array, or computes the sum of each of the complete rank-one sections parallel to a given dimension of an array of rank two or greater, possibly under the control of a mask.

Kind

Transformational (reduction) function

Arguments

`ARRAY` An array of numeric type. It must not be scalar.

`DIM` (optional)
A *constant* scalar expression of type integer, with a value in the range $1 \le$ `DIM` $\le n$, where n is the rank of `ARRAY`.

`MASK` (optional)
An array or scalar expression of type logical; if an array, it must be conformable with `ARRAY`.

Result Type

The result has the same type as `ARRAY`. It is a scalar if `DIM` is absent or if `ARRAY` has rank one; otherwise, it is an array with shape equal to that of `ARRAY` with dimension `DIM` removed.

Result Value

Case 1a `SUM( ARRAY, DIM = 1 [, MASK ] )`
`ARRAY` has rank `1`, `DIM` is present

or

Case 1b `SUM( ARRAY [, MASK = MASK ] )`
`ARRAY` has any rank, `DIM` is absent:

The result is a scalar equal (approximately) to the sum of the elements of `ARRAY` corresponding to the true elements of `MASK`, if present. If `MASK` is absent, the result is the sum of all elements of `ARRAY`.

Copyright © 1989–1994 Thinking Machines Corporation

If ARRAY is a zero-sized array, or if MASK is present and no elements of MASK are true, the result is the scalar value 0.

Case 2 SUM(ARRAY, DIM [, MASK])
ARRAY has rank $n > 1$, DIM is present:

For an ARRAY of rank n with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM}, d_{DIM+1}, ..., d_n]

the result is an array of rank $n-1$ with shape

[d_1, d_2, ..., d_{DIM-1}, d_{DIM+1}, ..., d_n].

The result is formed by computing the sum of each of the complete rank-one array sections of ARRAY that lie parallel to dimension DIM. If MASK is present, only elements of ARRAY corresponding to the true elements of MASK are used in the computation; if MASK is absent, all elements of ARRAY are used. Thus, the value of element

R(s_1, s_2, ..., s_{DIM-1}, s_{DIM+1}, ..., s_n)

of a result array R is equal to

SUM($ARRAY_{DIM}$ [, MASK = $MASK_{DIM}$])

where $ARRAY_{DIM}$ and $MASK_{DIM}$ are complete rank-one sections of ARRAY and MASK, both with shape [d_{DIM}]. $ARRAY_{DIM}$ is the array section

ARRAY(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n)

and $MASK_{DIM}$ is the array section

MASK(s_1, s_2, ..., s_{DIM-1}, : , s_{DIM+1}, ..., s_n) .

If MASK is present and all its elements are false, or if only dimension DIM of ARRAY is zero-sized, the result is an array with all elements equal to 0. If any dimension of ARRAY other than DIM is zero-sized, the result is a zero-sized array.

Examples

Case 1a ARRAY has rank 1, DIM is present:

SUM([2,3,-5,7], DIM = 1) → 7

If A = [2,3,-5,7] and M = [*T*,.,.,*T*], then

SUM(ARRAY = A, DIM = 1, MASK = M) → 9

SUM([1:0], DIM = 1) → 0

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

Case 1b `ARRAY` has any rank, `DIM` is absent:

If $\texttt{A} = \begin{bmatrix} -4 & 2 & -1 \\ 3 & 5 & -2 \end{bmatrix}$, then

```
SUM(A) → 3
```

and

```
SUM( ARRAY = A, MASK = A > 0 ) → 10
```

$\texttt{SUM(}\begin{bmatrix} -4 & 2 & -1 \\ 3 & 5 & -2 \end{bmatrix}\texttt{, MASK =}\begin{bmatrix} T & T & . \\ . & T & T \end{bmatrix}\texttt{)} \rightarrow 1$

Case 2 `ARRAY` has rank $n > 1$, `DIM` is present:

If $\texttt{B} = \begin{bmatrix} 3.5 & 2.0 & 5.0 \\ 4.5 & 1.5 & -7.0 \end{bmatrix}$, then

```
SUM( ARRAY = B, DIM = 1 ) → [8.,3.5,-2.]
```

If $\texttt{B} = \begin{bmatrix} 3.5 & 2.0 & 5.0 \\ 4.5 & 1.5 & -7.0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} T & T & . \\ . & T & T \end{bmatrix}$, then

```
SUM( ARRAY = B, DIM = 2, MASK = M ) → [5.5,-5.5]
```

If $\texttt{B} = \begin{bmatrix} 3.5 & 2.0 & 5.0 \\ 4.5 & 1.5 & -7.0 \end{bmatrix}$ and $\texttt{M} = \begin{bmatrix} . & . & . \\ T & T & T \end{bmatrix}$, then

```
SUM( ARRAY = B, DIM = 1, MASK = M ) → [4.5,1.5,-7.]
```

and

```
SUM( ARRAY = B, DIM = 2, MASK = M ) → [0.,-1.]
```


Copyright © 1989–1994 Thinking Machines Corporation

SYSTEM_CLOCK([COUNT] [, COUNT_RATE] [, COUNT_MAX])

Description

Returns integer data from a real-time clock.

Kind

Subroutine.

Arguments

`COUNT` (optional)

A scalar of type `INTEGER*4`. It is an `INTENT(OUT)` argument. `COUNT` returns the current value of the clock. It is set to zero at an arbitrary point and incremented by one each millisecond until `COUNT_MAX` is reached; the next millisecond, `COUNT` is reset to zero.

`COUNT_RATE` (optional)

A scalar of type `INTEGER*4`. It is an `INTENT(OUT)` argument. `COUNT_RATE` is always 1000.

`COUNT_MAX` (optional)

A scalar of type `INTEGER*4`. It is an `INTENT(OUT)` argument. `COUNT_MAX` is always 200000000.

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

TAN(X)

Description

Computes the tangent trigonometric function.

Kind

Elemental function.

Argument

X An expression of type real.

Result Type

Same as X.

Result Value

The result is an approximation to the `tan(X)` trigonometric function, with X regarded as a value in radians.

Examples

```
TAN(0.d0) → 0.d0
TAN(π) → 0.
TAN(π/4) → 1.
TAN(π*45/180) → 1.
TAN(ATAN(.25)) → 0.25
```

$$\texttt{TAN}\left(\begin{bmatrix} 0.52359 & 3.14159 \\ 1.04720 & 2.35619 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} 0.57773 & 0.00000 \\ 1.73205 & -1.00000 \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation

TANH(X)

Description

Computes the hyperbolic tangent function.

Kind

Elemental function.

Argument

X An expression of type real.

Result Type

Same as X.

Result Value

The result is an approximation to the `tanh(X)` mathematical function.

Examples

```
TANH(0.) → 0.

TANH( [ .5 2. 5. ] ) ⇒ [ 0.462117  0.964028  0.999909 ]
      [ 1. 3. 9. ]     [ 0.761594  0.995055  1.000000 ]
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

TINY(X)

Description

Returns the smallest positive number in the model representing numbers of the same type and kind type as the argument.

Kind

Inquiry function.

Argument

X	A scalar or array of type REAL.

Result Type

Scalar of the same type and kind type as X.

Result Value

The result has the value $2^{e_{min}-1}$, where e_{min} is as defined in Section 17.15 for the model representing numbers of the same type and kind type as X.

The result is one of the following values, depending on the kind type of the argument.

Data Type	Value
REAL*4	2^{-126}
REAL*8	2^{-1022}

Copyright © 1989–1994 Thinking Machines Corporation

TRANSPOSE(MATRIX)

Description

Returns the transpose an array of rank two.

Kind

Transformational function.

Argument

`MATRIX` A rank-two array of numeric or logical type.

Result Type

The result is an array of the same type as `MATRIX`, with rank two and shape `[`*n*`,`*m*`]`, where `[`*m*`, `*n*`]` is the shape of `MATRIX`.

Result Value

Element `R(`*i*`,`*j*`)` of a result array `R` has the value

```
MATRIX(i,j)
```

for *i* = 1,2,...,*n* and for *j* = 1,2,...,*m*

Examples

$$\texttt{TRANSPOSE}\left(\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}\right)\rightarrow\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

$$\texttt{TRANSPOSE}\left(\begin{bmatrix}T&.&T\\.&T&.\end{bmatrix}\right)\rightarrow\begin{bmatrix}T&.\\.&T\\T&.\end{bmatrix}$$

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

UBOUND(ARRAY *[*, DIM *]*)

Description

Returns all the upper bounds of an array, or the upper bound of a specified dimension.

Compatibility Note

For back compatibility, CM Fortran also accepts the intrinsic function `DUBOUND`, which is equivalent to `UBOUND`.

Kind

Inquiry function.

Arguments

`ARRAY` An array of numeric or logical type. It must not be an assumed-size array.

`DIM` (optional)
A scalar expression of type integer, with a value in the range $1 \leq$ `DIM` $\leq n$, where n is the rank of `ARRAY`.

Result Type

The result is of type `INTEGER*4`. It is scalar if `DIM` is present; if `DIM` is absent, the result is a CM array of rank one and size n, where n is the rank of `ARRAY`.

Result Value

Case 1 `UBOUND(ARRAY,DIM)`
`DIM` is present:

The result has a value equal to the upper bound for subscript `DIM` of `ARRAY` if dimension `DIM` does not have size zero; it has the value `0` if dimension `DIM` has size zero. For an array section or other array expression, its value is the number of elements in the corresponding dimension.

Case 2 `UBOUND(ARRAY)`
`DIM` is absent:

For an `ARRAY` of rank n, the result is a CM array with the value

$[e_1, e_2, \ldots, e_n]$

where e_i = `UBOUND(ARRAY,`i`)`, for $i = 1, 2, \ldots, n$.

Copyright © 1989–1994 Thinking Machines Corporation

Examples

If A is declared by the statement

```
REAL A(2:3,7:10)
```

then

```
UBOUND(A) → [3,10]
UBOUND( A, DIM = 1 ) → 3
UBOUND( A, DIM = 2 ) → 10
```

Also,

```
UBOUND( [-2,3,10] ) → [3]
UBOUND( [-2,3,10], DIM = 1 ) → 3
UBOUND( [1:-1] ) → [0]
UBOUND( [1:-1], DIM = 1 ) → 0
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

UNPACK(VECTOR, MASK, FIELD)

Description

Scatters selected elements from a rank-one array into an array of any rank, under the control of a mask, with alternate elements provided by a default field.

Kind

Transformational function.

Arguments

`VECTOR`	A rank-one array of numeric or logical type. Its size must be at least *t*, where *t* is the number of true elements in `MASK`.
`MASK`	An array-valued expression of type logical.
`FIELD`	A scalar or array of the same type as `VECTOR` that is conformable with `MASK`.

Result Type

The result is an array of the same type as `VECTOR` and the same shape as `MASK`.

Result Value

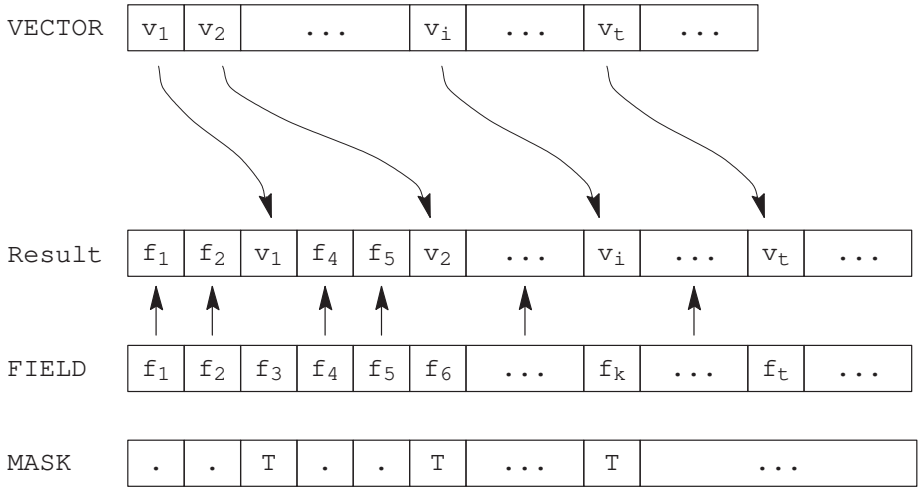
The result is an array consisting of values from `VECTOR` selected by `MASK`, interspersed by values supplied by `FIELD`. If *t* is the total number of true values in `MASK`, then the value of the element of the result that corresponds to the *i*th true element of `MASK` has the value

```
VECTOR(i)
```

for *i* ranging from `1` to *t*, considering elements in array element order. Each other result element has a value equal to `FIELD` if `FIELD` is scalar, or to the element of `FIELD` corresponding to the *i*th true element of `MASK`, if `FIELD` is an array.

Copyright © 1989–1994 Thinking Machines Corporation

The action of the UNPACK function is illustrated graphically below:



Examples

Specific values may be "scattered" to specific positions in an array by using UNPACK:

$$\texttt{UNPACK([11,22,33], MASK =} \begin{bmatrix} . & T \\ T & . \\ . & T \end{bmatrix} \texttt{, FIELD =} \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \texttt{)} \rightarrow \begin{bmatrix} 1 & 22 \\ 11 & 5 \\ 3 & 33 \end{bmatrix}$$

$$\texttt{UNPACK([11,22,33], MASK =} \begin{bmatrix} . & T \\ T & . \\ . & T \end{bmatrix} \texttt{, FIELD = 7)} \rightarrow \begin{bmatrix} 7 & 22 \\ 11 & 7 \\ 7 & 33 \end{bmatrix}$$

Copyright © 1989–1994 Thinking Machines Corporation