The
Connection Machine
System

Change Pages to Paris Reference Manual Supplement

Update for Version 5.2
October 1989

Add these change pages to the
Paris Reference Manual Supplement,
which was distributed with Version 5.1

Thinking Machines Corporation
Cambridge, Massachusetts

First printing, October 1989

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts 02142–1214
(617) 876–1111

About Paris Version 5.2 Change Pages

Purpose of These Change Pages

Change pages correct and update a manual. The change pages in this packet provide corrections to dictionary entries in the *Paris Reference Manual Supplement*, Version 5.1.

What Has Changed?

The Version 5.2 *Paris Release Notes* include descriptions of the documentation errors corrected by the change pages included in this packet.

What to Do with These Pages

By page number, replace the existing pages in the *Paris Reference Manual Supplement*, Version 5.1. In each case, simply tear out the existing page and replace it with the new one.

Placement of Change Pages

Change Page Sequence	Replace pages
17, 18	17–18
43, 44	43–44

After inserting the change pages, this explanatory page and the title page for this change pages packet may be discarded.

Contents for Supplement Change Pages

CHANGE-FIELD-ALIAS

Changes the referent of the specified field alias.

Formats CM:change-field-alias *alias-id*, *field-id*

Operands *alias-id* An alias field-id. This must be an alias field-id returned by CM:make-field-alias. It need not be in the current VP set.

field-id A field-id. This must be a field id returned by CM:allocate-stack-field or CM:allocate-heap-field; it may *not* be an offset into a field. The field need not be in the current VP set.

Context This operation is unconditional. It does not depend on the *context-flag*.

The alias field id *alias-id* is made to reference the field identified by *field-id*. This function allows field aliases to be recycled.

After a call to CM:change-field-alias, the field length and the physical length associated with *alias-id* are exactly what they would be if CM:make-field-alias had been called with *field-id*.

An error is signaled if the physical length of the aliased field is not exactly divisible by the VP ratio of the VP set to which *field-id* belongs. (For more on the physical length associated with an alias field see the dictionary entry for CM:make-field-alias.)

The alias field-id can be used in all the same ways as a regular field-id can, with the following exceptions:

- It cannot be passed to CM:deallocate-heap-field.
- It cannot be passed to CM:deallocate-stack-through.

C-F-CIS

Calculates the cosine and sine for the floating-point source field and stores the result in the complex destination field.

Formats CM:c-f-cis-2-1L *dest*, *source*, *s*, *e*

Operands

dest The complex destination field.

source The floating-point source field.

s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of the *dest* field in this format is $2(s+e+1)$. The total length of the *source* field in this format is $s+e+1$.

Overlap The *source* field must be either identical to *dest*, identical to $(dest+s+e+1)$, or disjoint from *dest*.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $\textit{context-flag}[k] = 1$ then
$\textit{dest}[k].\textit{real} \leftarrow \cos \textit{source}[k]$
$\textit{dest}[k].\textit{imag} \leftarrow \sin \textit{source}[k]$

The result is a complex number whose real part is the cosine of the *source* and whose imaginary part is the sine of the *source*. The term cis signifies $\cos + i \sin$.

MAKE-FIELD-ALIAS

Creates a new field-id that points to an existing field.

Formats result ← CM:make-field-alias *field-id*

Operands *field-id* A field-id. This must be a field id returned by CM:allocate-stack-field or CM:allocate-heap-field; it may *not* be an offset into a field. The field need not be in the current VP set.

Result A field-id, the alias field-id. This id initially resides in the current VP set.

Context This operation is unconditional. It does not depend on the *context-flag.*

The return value is a *field alias.* It is a new field-id that identifies the same area of memory as does *field-id.*

The field identified by *field-id* can be in a VP set other than the current VP set. The returned alias field-id initially resides in the current VP set. The alias field-id can be used in all the same ways as a regular field-id can, with the following exceptions:

- It cannot be passed to CM:deallocate-heap-field.
- It cannot be passed to CM:deallocate-stack-through.

Associated with a field alias is a *physical length*: the number of bits that the field occupies in each physical processor. Also associated with a field alias is a *field length*: the number of bits the field occupies in each virtual processor. The physical length is equal to the field length multiplied by the VP ratio of the current VP set. It is an error if the physical length is not exactly divisible by the VP ratio of the current VP set.

It is possible for the field length of an alias field to be different from the field length of the original field. This is the case when make-field-alias is called on a field in a VP set that has a VP ratio different from the VP ratio of the current VP set. Suppose, for example, the current VP ratio is 32. If we make an alias for a 32-bit field that resides in a VP set with a VP ratio of 1, the resulting alias field is a 1 bit field (in a VP ratio of 32).

F-MOD

The residue of one floating-point source value divided by another is placed in the destination field. Overflow is also computed.

Formats

CM:f-mod-2-1L	*dest/source1*, *source2*, *s*, *e*
CM:f-mod-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:f-mod-constant-2-1L	*dest/source1*, *source2-value*, *s*, *e*
CM:f-mod-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*

Operands

dest The floating-point destination field. This is the quotient.

source1 The floating-point first source field. This is the dividend.

source2 The floating-point second source field. This is the divisor.

source2-value A floating-point immediate operand to be used as the second source.

s, *e* The significand and exponent lengths for the *dest*, *source1*, and *source2* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

Flags *test-flag* is set if division by zero occurs; otherwise it is cleared.

overflow-flag is set if floating-point overflow occurs; otherwise it is unaffected.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\mathit{context\text{-}flag}[k] = 1$ then
    if $\mathit{source2}[k] = 0$ then
      $\mathit{dest}[k] \leftarrow \langle\text{unpredictable}\rangle$
      $\mathit{test\text{-}flag}[k] \leftarrow 1$
    else
      $\mathit{dest}[k] \leftarrow \mathit{source1}[k] - \mathit{source2}[k] \times \left\lfloor \frac{\mathit{source1}[k]}{\mathit{source2}[k]} \right\rfloor$
      $\mathit{test\text{-}flag}[k] \leftarrow 0$
    if $\langle$overflow occurred in processor $k\rangle$ then $\mathit{overflow\text{-}flag}[k] \leftarrow 1$

The
Connection Machine
System

Change Pages to Paris Reference Manual

Update for Version 5.2

October 1989

Add these change pages to the *Paris Reference Manual* only after adding those distributed with Version 5.1

Thinking Machines Corporation
Cambridge, Massachusetts

First printing, October 1989

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts 02142–1214
(617) 876–1111

About Paris Version 5.2 Change Pages

Purpose of These Change Pages

Change pages correct and update a manual. The change pages in this packet provide

- dictionary entries for Paris instructions new with Version 5.2
- dictionary entries for Paris instructions changed with Version 5.2
- corrected dictionary entries for Version 5.0 Paris instructions

What Has Changed?

The Version 5.2 *Paris Release Notes* describe the new and changed features that are documented by these pages. The release notes also include descriptions of all the documentation errors corrected by change pages included in this packet.

What to Do with These Pages

By page number, insert the change pages into your copy of the *Paris Reference Manual*, Version 5.0.

Additional Pages

Any change page with a page number ending in a letter must be added to the existing manual. Find the page whose number matches the number part of the change page number and insert the change page behind it.

Replacement Pages

Any change page with a normal page number replaces an existing Paris manual page. Tear out the existing page and replace it with the new one.

Note that many of the replacement pages are included only to preserve the order of the Paris dictionary entries.

Placement of Change Pages

Change Page Sequence	Add after page	Replace pages
45, 46		45–46
51, 52		51–52
63, 64		63–64
83, 84, 85, 86		83–86
89, 90, 91, 92		89–92
92a	92	
93, 94, 95, 96, 97, 98		93–98
106a	106	
107, 107a, 107b, 107c, 107d, 107e, 107f		107
108		108
113, 113a, 113b, 113c, 114		113–114
117,117a, 117b, 118		117–118
123, 124		123–124
133, 134, 135, 136		133–136
142a, 142b	142	
143, 143a, 143b, 143c, 144		143–144
157, 157a, 158, 159, 160, 161, 162		157–162
181, 182		181–182
185, 186, 187, 188		185–188

Placement of Change Pages (continued)

Change Page Sequence	Add after page	Replace pages
211, 212		211–212
271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286		271–286
301, 302, 303, 304		301–304
307, 308, 309, 310, 311, 312		307–312
340a, 340b, 340c, 340c, 340d	340	
341, 342		341–342
373, 374, 375, 376, 377, 378, 379, 380		373–380
435, 435a		435
449, 449a, 449b, 449c, 450		449–450
455, 456		455–456

After inserting all change pages, these explanatory pages and the title page for this change pages packet may be discarded.

Contents for Reference Change Pages

inclusion

One of the values CM_exclusive or CM_inclusive, indicating the boundaries of a scan instruction.

smode

One of the values CM_none, CM_start_bit, or CM_segment_bit, indicating how a scan operation is to be partitioned.

There are other symbolic values as well, but these are the most important. All names are formed by the standard rule: starting from a Lisp name such as :start-bit, add "CM" to the front and then convert colons and hyphens to underscores, yielding CM_start_bit.

6.3 C/Paris Configuration Variables

The configuration variables provide access to information about the configuration of the Connection Machine system. See section 3.6 for a list. The C/Paris interface makes these variables accessible through variables declared in the C/Paris header file. They are initialized in an application program by a call to the subroutine CM_init and should not be changed by an application program.

Each configuration variable is a numeric value that is constant over the course of a session (from one cold boot operation to the next), or varies from one Connection Machine configuration to another. For example, CM_physical_processors_limit is a value that depends upon the size of the Connection Machine to which the application is attached.

Numeric values that are constant for a given release of the CM System Software are given in #define statements.

6.4 Calling Paris from C

This section describes how to build C programs that access the Paris instruction set using the C/Paris interface. Such programs must manage the dynamic allocation and deallocation of Connection Machine fields directly. This section describes the form of C main programs and subprograms that call the C/Paris interface, as well as the steps involved in compiling and linking such programs.

The following code fragment illustrates the structure of a C main program that calls Paris instructions.

```
#include <cm/paris.h>
⋮
main() {
  CM_init();
  ⋮
  CM_paris_instruction(...);
  ⋮
  if ( CM_configuration_variable > limit ) ...
```

```
  ⋮
}
```

Note that the call to CM_init is required prior to any other calls to Paris instructions.

The following code fragment illustrates the structure of a C subroutine subprogram that calls Paris instructions.

```
#include <cm/paris.h>
⋮
float test() {
  ⋮
  CM_paris_instruction(...);
  ⋮
  if ( CM_configuration_variable > limit ) ...
  ⋮
}
```

It looks exactly like a main program in its use of Paris, *except* that a subprogram should not call CM_init.

Use the following command to compile and link these program units:

```
% cc main.c test.c -lparis -lm
```

Note that there should be no space between the -l option and its argument.

smode

One of the values CM_none, CM_start_bit, or CM_segment_bit, indicating how a scan operation is to be partitioned.

There are other symbolic values as well, but these are the most important. All names are formed by the standard rule: starting from a Lisp name such as :start-bit, add "CM" to the front and then convert colons and hyphens to underscores, yielding CM_start_bit.

7.3 Fortran/Paris Configuration Variables

The configuration variables provide access to information about the configuration of the Connection Machine system. See section 3.6 for a list. The Fortran/Paris interface makes these variables accessible through variables declared in the common block named cmval, defined by the Fortran/Paris header file. They are initialized in an application program by a call to the subroutine CM_init and should not be changed by an application program.

Each configuration variable is a numeric value that is constant over the course of a session (from one cold boot operation to the next), or varies from one Connection Machine configuration to another. For example, CM_physical_processors_limit is a value that depends upon the size of the Connection Machine to which the application is attached. Most of these configuration variables are declared to be of Fortran type INTEGER.

Numeric values that are constant for a given release of the CM System Software are also given in PARAMETER statements.

7.4 Calling Paris from Fortran

This section describes how to build Fortran programs that access the Paris instruction set using the Fortran/Paris interface. Such programs must manage the dynamic allocation and deallocation of Connection Machine fields directly. This section describes the form of Fortran main programs and subprograms that call the Fortran/Paris interface, as well as the steps involved in compiling and linking such programs.

The following code fragment illustrates the structure of a Fortran main program that calls Paris instructions.

```
      PROGRAM main
C     VAX Fortran or Sun Fortran
      ⋮
      INCLUDE '/usr/include/cm/paris-configuration-fort.h'
      CALL CM_init()
      ⋮
      CALL CM_paris_instruction(...)
      ⋮
      IF ( CM_configuration_variable .GT. limit ) ...
      ⋮
      END
```

Note that the call to CM_init is required prior to any other calls to Paris instructions.

The following code fragment illustrates the structure of a Fortran subroutine subprogram that calls Paris instructions.

```
      SUBROUTINE test
C     VAX Fortran or Sun Fortran
      ⋮
      INCLUDE '/usr/include/cm/paris-configuration-fort.h'
      ⋮
      CALL CM_paris_instruction(...)
      ⋮
      IF ( CM_configuration_variable .GT. limit ) ...
      ⋮
      END
```

It looks exactly like a main program in its use of Paris, *except* that a subprogram should not call CM_init.

Using VAX Fortran, the following command compiles and links these program units to run on the Connection Machine Model 2:

```
% fort main.for test.for -lparisfort -lparis
```

Note that there should be no space between the -l option and its argument.

Using Sun Fortran, the following command compiles and links these program units to run on the Connection Machine Model 2:

```
% f77 main.f test.f -lparisfort -lparis
```

Note that there should be no space between the -l option and its argument.

F-ABS

Computes, in each selected processor, the absolute value of a floating-point source field and stores it in the destination field.

Formats CM:f-abs-1-1L *dest/source*, *s*, *e*
CM:f-abs-2-1L *dest*, *source*, *s*, *e*

Operands

dest	The floating-point destination field.
source	The floating-point source field.
s, *e*	The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    if $\textit{source}[k] \geq 0$ then $\textit{dest}[k] \leftarrow \textit{source}[k]$
    else $\textit{dest}[k] \leftarrow -\textit{source}[k]$

The absolute value of the *source* operand is placed in the *dest* operand.

For floating-point numbers, absolute value is calculated by changing the sign bit to 0 (positive). All other bits in the number are unchanged. As a result, the absolute values of negative infinities, denormalized numbers, and NaN's are their positive counterparts.

S-ABS

Computes the absolute value of a signed integer source field and stores it in the destination field.

Formats

CM:s-abs-1-1L *dest/source*, *len*
CM:s-abs-2-1L *dest*, *source*, *len*
CM:s-abs-2-2L *dest*, *source*, *dlen*, *slen*

Operands

dest The signed integer destination field.

source The signed integer source field.

len The length of the *dest* and *source* fields. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

dlen The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen The length of the *source* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Flags *overflow-flag* is set if the result cannot be represented in the destination field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    if *source*$[k] \geq 0$ then *dest*$[k] \leftarrow$ *source*$[k]$
    else *dest*$[k] \leftarrow -$*source*$[k]$
    if $\langle$overflow occurred in processor $k\rangle$ then *overflow-flag*$[k] \leftarrow 1$
    else *overflow-flag*$[k] \leftarrow 0$

The absolute value of the *source* operand is placed in the *dest* operand. (If the length of the *dest* field equals the length n of the *source* field, overflow can occur only if the *source* field contains -2^n. If the length of the *dest* field is greater than the length of the *source* field, then overflow cannot occur.)

ALLOCATE-HEAP-FIELD-VP-SET

Allocates a new heap field of the specified length in the specified VP set and returns a unique identifier.

Formats result ← CM:allocate-heap-field-vp-set *len*, *vp-set-id*

Operands	*len*	An unsigned integer, the length in bits of the field to be allocated.
	vp-set-id	A vp-set-id.
Result	An unsigned integer, the new field-id.	
Context	This operation is unconditional. It does not depend on the *context-flag*.	

A new field of length *len* is allocated on the heap within the specified VP set. A field-id for the newly created field is returned.

ALLOCATE-STACK-FIELD

Allocates a new stack field of specified length in the current VP set and returns a unique identifier.

Formats result ← CM:allocate-stack-field *len*

Operands *len* An unsigned integer, the length, in bits, of the field to be allocated.

Result An unsigned integer, the new field-id.

Context This operation is unconditional. It does not depend on the *context-flag.*

A new field of length *len* is allocated on the stack within the current VP set. A field-id for the newly created field is returned.

ALLOCATE-STACK-FIELD-VP-SET

Allocates a new stack field of the specified length in the specified VP set and returns a unique identifier.

Formats result ← CM:allocate-stack-field-vp-set *len*, *vp-set-id*

Operands	*len*	An unsigned integer, the length in bits of the field to be allocated.
	vp-set-id	A vp-set-id.
Result	An unsigned integer, the new field-id.	
Context	This operation is unconditional. It does not depend on the *context-flag*.	

A new field of length *len* is allocated on the stack within the specified VP set. A field-id for the newly created field is returned.

ALLOCATE-VP-SET

Create a new VP set, within which fields may be allocated.

Formats result ← CM:allocate-vp-set *geometry-id*

Operands *geometry-id* A geometry-id.

Result A vp-set-id, identifying the newly allocated VP set.

Context This operation is unconditional. It does not depend on the *context-flag*.

This operation returns a vp-set-id for a newly created VP set. This may be given to other Paris operations in order to create memory fields in which data may be stored. The size and shape of the VP set is determined by the geometry specified by the *geometry-id*. It is possible to alter the geometry later (by using CM:set-vp-set-geometry), but the total number of virtual processors in the VP set remains forever fixed.

AREF32

Fetches array elements specified by a per-processor index and copies them to a fixed destination. The array is stored in a special format that allows fast access.

Formats

CM:aref32-2L *dest, array, index, dlen, index-len, index-limit*
CM:aref32-always-2L *dest, array, index, dlen, index-len, index-limit*

Operands

dest The destination field.

array The source array field. This must contain data stored in a special format by either CM:aset32 or CM:transpose32.

index The unsigned integer index field. This is used as the per-processor index into the *array*.

dlen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This is taken as the *array* element length and must be a multiple of 32.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the *array* extent.

Overlap The fields *array* and *index* may overlap in any manner. However, the *array* and *index* fields must not overlap the *dest* field.

Context The non-always operations are conditional. The destination may be altered only in processors whose *context-flag* is 1.

The always operations are unconditional. The destination may be altered regardless of the value of the *context-flag*.

Definition

For every virtual processor k in the *current-vp-set* do
  if (always or $\textit{context-flag}[k] = 1$) then
    if $\textit{index}[k] < \textit{index-limit}$ then
      let $r = \textit{geometry-total-vp-ratio}(\textit{geometry}(\textit{current-vp-set}))$
      let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
      let $i = \textit{index}[k]$
      for all j such that $0 \le j < \textit{dlen}$ do
        $\textit{dest}[k]\langle j \rangle \leftarrow \textit{array}[k - m \times r + (j \bmod 32) \times r]\langle 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor) \rangle$
    else
      $\langle$error$\rangle$

This is a simple form of array reference for parallel arrays whose elements are stored across the memory of individual processors. To each processor belongs an array of extent *index-limit* with elements of length *dlen*.

The *array* element indexed by each active processor is copied into the *dest* field of that processor. Different processors may reference different elements of their arrays. For this reason, this form of array referencing is known as *indirect addressing*.

Each processor has an array index stored in the field *index*. This is used to index into an area of CM memory, *array*, whose allocated length in bits should be at least

$$\left(\mathit{index\text{-}limit} \times \left\lceil \frac{\mathit{dlen}}{32} \right\rceil\right) \times 32$$

The argument *index-limit* is one greater than the largest allowed value of the index. It is an error for any *index* value to equal or exceed this limit.

A field of length *dlen*, and starting at address $\mathit{array} + i \times 32$, where i is the the unsigned number stored at *index*, is copied to *dest* in all selected processors. Even this is not quite accurate, because the array data is not organized in the same manner as for CM:aref. Instead, it is organized in a peculiar way for fast per-processor access. Parallel arrays stored in this format are termed *slicewise parallel arrays*.

Slicewise parallel array data is arranged with successive bits stored in successive processors within groups of 32 virtual processors. Thus, slicewise array data belonging to one processor is spread over the memories of the 32 processors in its group and the memory of each processor holds data belonging to all 32 processors.

A region of memory set aside for a slicewise array of the format required by CM:aref32 should be accessed only through the operations CM:aset32 and CM:aref32, related operations such as CM:get-aref32 and CM:send-aset32-overwrite, or operations that copy the array as a whole from all processors (such as I/O operations). It is also possible to operate on this memory in blocks of 32-bit square matrices with the CM:transpose32 instruction.

AREF32-SHARED

Fetches an array element specified by a per-processor index and copies it to a fixed destination. The source array is stored in a special format that allows fast access, and is accessed in such a way that all the virtual processors within a group of 32 physical processors share the same array.

Formats

CM:aref32-shared-2L	*dest, array, index, dlen, index-len, index-limit*
CM:aref32-shared-always-2L	*dest, array, index, dlen, index-len, index-limit*

Operands

dest — The destination field.

array — The source array field. This must be a contiguous region in CM memory. It need not be in the current VP set.

index — The unsigned integer index field. This is used as the per-processor index into *array*.

dlen — The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This is normally taken as the array element length and must be a multiple of 32. As a special case, *dlen* may be 8 or 16 and, if so, access into both the source and the destination fields is offset appropriately.

index-len — The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit — An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of *array*.

Overlap — The fields *array* and *index* may overlap in any manner. However, the *array* and *index* fields must not overlap the *dest* field.

Context — The non-always operations are conditional. The destination may be altered only in processors whose *context-flag* is 1.

The always operations are unconditional. The destination may be altered regardless of the value of the *context-flag*.

Definition For every virtual processor k in the *current-vp-set* do
  if (always or $\textit{context-flag}[k] = 1$) then
    if $\textit{index}[k] < \textit{index-limit}$ then
      for all j such that $0 \le j < \textit{dlen}$ do
        $\textit{dest}[k]\langle j \rangle \leftarrow$

$$array\left[32\left\lfloor\frac{k}{32r}\right\rfloor + (j \bmod 32)\right]\left\langle index\text{-}limit\left\lfloor\frac{j}{32}\right\rfloor + index[k]\right\rangle$$

else

⟨error⟩

where r is the VP ratio, and where j is the bit position in each field.

This is a simple form of array reference for arrays whose elements are stored across the memory of individual processors and accessed in such a way that many processors appear to share a single array of extent *index-limit* with elements of length *dlen*.

The shared array element (or a portion of it) indexed is copied into *dest* in all (selected) processors. Different processors may access different elements of the shared array. For this reason, this form of array referencing is known as *indirect addressing.*

Each processor has an array index stored in the field *index*. This is used to index into *array*. The argument *index-limit* is one greater than the largest allowed value of the index. It is an error for any *index* value to equal or exceed this limit.

The data within the source array area is not organized in the same manner as for CM:aref; instead, it is organized in a peculiar way for fast per-processor access. Shared arrays stored in this format are termed *slicewise shared arrays.*

Slicewise shared array data is arranged with successive bits stored in successive processors, within groups of 32 physical processors. Each 32-bit word of each element is stored separately in processor memories, as follows: The low-order 32 bits of all elements are grouped together across processor memories in a field of length 32 × *index-limit* bits. Similarly, the next 32 bits of all elements are grouped together, and so on, up to the high-order bits of all array elements. This data format allows fast hardware-supported access to the individual elements of a shared array.

A region of memory set aside for an array of the format required by CM:aref32-shared must be contiguous in memory. It must therefore be allocated all at once, at a VP ratio of 1, with a single call to CM:allocate-stack-field or to CM:allocate-heap-field. Alternatively, from Lisp, the memory may be allocated within a with-stack-field form at a VP ratio of 1.

The area of CM memory occupied by *array* should be allocated at a VP ratio of 1 as a field whose length in bits is exactly

$$index\text{-}limit \times dlen$$

Shared array memory should be accessed only with the operations CM:aref32-shared and CM:aset32-shared, or with operations that copy the array as a whole from all processors (such as I/O operations). Data in such a region of memory may, however, be reoriented with the CM:transpose32 instruction.

As a special case, if the *dlen* argument is specified as 8 or 16, then each processor accesses one byte or one half-word of a 32-bit element. The *index-limit* argument must be specified as the extent of the array when considered to contain 32-bit elements. Nonetheless, valid *index* values are integers 0 through 2 or 4 times this *index-limit*. The *index* argument may be thought of as consisting of two fields, one that indexes a 32-bit array element and one that indexes an 8- or 16-bit offset into that element. To index bytes, the low 2 bits of *index* specify the offset. To index half-words, the low 1 bit of *index* specifies the offset.

ASET

Stores into an array element specified by a per-processor index a value copied from a fixed source field.

Formats CM:aset-2L *source*, *array*, *index*, *slen*, *index-len*, *index-limit*, *element-len*

Operands

source The source field.

array The destination array field.

index The unsigned integer index into the array field.

slen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*.

element-len An unsigned integer immediate operand to be used as the length of an array element.

Overlap The fields *source* and *index* may overlap in any manner. However, the *source* and *index* fields must not overlap the *array* field.

Flags *test-flag* is set if the value in the *index* field is less than the *index-limit*; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    if $\textit{index}[k] < \textit{index-limit}$ then
      let $p = \textit{index}[k] \times \textit{element-len}$
      $\textit{array}[k]\langle p : p + \textit{slen} - 1\rangle \leftarrow \textit{source}[k]$
      $\textit{test-flag}[k] \leftarrow 1$
    else
      $\textit{test-flag}[k] \leftarrow 0$

This is a simple form of array modification, for arrays stored in the memory of individual processors. Each processor has an array index stored in the field *index*. This is used to index into an *array*, whose length in bits should be *index-limit* $\times$ *element-len*. The *source* field is copied into the element indexed (or a portion of it) in all selected processors. Thus different processors may modify different elements of their arrays.

More precisely, the *source* field is copied to a field of length *slen* and starting at address *array* $+ i \times$ *element-len*, where i is the unsigned number stored at *index*, in all selected processors.

The argument *index-limit* is one greater than the largest allowed value of the index. Those processors that have index values greater than or equal to *index-limit* do not alter the value of the destination field; they also clear *test-flag*. All processors in which the index field is less than *index-limit* set *test-flag*. The argument *element-len* is the length of individual elements of the array. Usually this will be the same as *dest-length*, but for certain applications it is worthwhile for it to differ. For example, within an array of 128-bit records one may store into just one 16-bit component of an indexed record by letting *slen* be 32, letting *element-len* be 128, and by offsetting the *array* address by the offset within each record of the 16-bit quantity to be modified. As another example, to modify a 4-character substring of a string of 8-bit characters, one may let *slen* be 32 and *element-len* be 8.

ASET32

Copies data from a fixed source to the destination array elements specified by a per-processor index. The destination array is stored in a special format that allows fast access.

Formats

CM:aset32-2L *source*, *array*, *index*, *slen*, *index-len*, *index-limit*

Operands

source	The source field.
array	The destination array field.
index	The unsigned integer index field. This is used as the per-processor index into *array*.
slen	The length of the *source* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This is taken as the *array* element length and must be a multiple of 32.
index-len	The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.
index-limit	An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the *array* extent.

Overlap The fields *source* and *index* may overlap in any manner. However, the *source* and *index* fields must not overlap the *array* field.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition

For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    if *index*$[k] <$ *index-limit* then
      let $r =$ *geometry-total-vp-ratio*(*geometry*(*current-vp-set*))
      let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
      let $i =$ *index*$[k]$
      for all j such that $0 \le j <$ *slen* do
        $array[k - m \times r + (j \bmod 32) \times r]\langle 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor)\rangle \leftarrow source[k]\langle j\rangle$
    else
      $\langle$error$\rangle$

This is a simple form of array modification for parallel arrays whose elements are stored across the memory of individual processors. To each processor belongs an array of extent *index-limit* with elements of length *slen*.

The *source* field value for each active processor is copied into the indexed array element belonging to that processor. Thus different processors may modify different elements of their arrays. For this reason, this form of array access is known as *indirect addressing*.

Each processor has an array index stored in the field *index*. This is used to index into an area of CM memory, *array*, whose allocated length in bits should be at least

$$\left(\mathit{index\text{-}limit} \times \left\lceil \frac{\mathit{slen}}{32} \right\rceil\right) \times 32$$

The argument *index-limit* is one greater than the largest allowed value of the index. It is an error for any *index* value to equal or exceed this limit.

In all selected processors, the *source* field is copied to a field of length *slen* and starting at address $\mathit{array} + i \times 32$, where i is the the unsigned number stored at *index*. Even this is not quite accurate, because the data within the destination *array* area is not organized in the same manner as for CM:aset. Instead, it is organized in a peculiar way for fast per-processor access. Parallel arrays stored in this format are termed *slicewise parallel arrays*.

Slicewise parallel array data is arranged with successive bits stored in successive processors within groups of 32 virtual processors. Thus, slicewise array data belonging to one processor is spread over the memories of the 32 processors in its group and the memory of each processor holds data belonging to all 32 processors.

A region of memory set aside for a slicewise array of the format required by CM:aset32 should be accessed only through the operations CM:aref32 and CM:aset32, related operations such as CM:send-aset32-overwrite and CM:get-aref32, or operations that copy the array as a whole from all processors (such as I/O operations). It is also possible to operate on this memory in blocks of 32-bit square matrices with the CM:transpose32 instruction.

ASET32-SHARED

Copies data from a fixed source to the destination array elements specified by a per-processor index. The array is stored in a special format that allows fast access, and is accessed in such a way that all the virtual processors within a group of 32 physical processors share the same array.

Formats CM:aset32-shared-2L *source*, *array*, *index*, *slen*, *index-len*, *index-limit*

Operands

source — The source field.

array — The destination array field. This must be contiguous region in CM memory. It need not be in the current VP set.

index — The unsigned integer index field. This is used as the per-processor index into the *array*.

slen — The length of the *source* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This must be a multiple of 32 and is taken as the array element length.

index-len — The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit — An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of *array*.

Overlap The fields *source* and *index* may overlap in any manner. However, the *source* and *index* fields must not overlap the *array* field.

Context This operation is conditional, but whether data is copied depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is stored into the the field indicated by *array* regardless of the *context-flag* of the receiving processor.

Definition For every virtual processor k in the *current-vp-set* do

if *context-flag*$[k] = 1$ then
  if *index*$[k] <$ *index-limit* then
    for all j such that $0 \le j < dlen$ do
      $array\left[32\left\lfloor \frac{k}{32r} \right\rfloor + (j \bmod 32)\right]\left\langle index\text{-}limit \left\lfloor \frac{j}{32} \right\rfloor + index[k]\right\rangle \leftarrow source[k]\langle j\rangle$
  else
    ⟨error⟩

where r is the VP ratio, and where j is the bit position in each field.

> For any two active virtual processors, k and k', if $index[k] = index[k']$, then either $source[k]$ or $source[k']$ is stored in *dest*, depending upon the implementation.

This is a simple form of array modification for arrays whose elements are stored across the memory of individual processors and accessed in such a way that many processors appear to share a single array of extent *index-limit* with elements of length *slen*.

The *source* field in each selected processor is copied into the array element (or a portion of it) indexed. Different processors may modify different elements of the shared array. For this reason, this form of array referencing is known as *indirect addressing*. If several processors sharing the same array attempt to modify the same element in a single CM:aset32-shared operation, then one of the values is stored and the rest are discarded.

Each processor has an array index stored in the field *index*. This is used to index into *array*. The argument *index-limit* is one greater than the largest allowed value of the index. It is an error for any *index* value to equal or exceed this limit.

The data within the destination array area is not organized in the same manner as for CM:aset; instead, it is organized in a peculiar way for fast per-processor access. Shared arrays stored in this format are termed *slicewise shared arrays*.

Slicewise shared array data is arranged with successive bits stored in successive processors, within groups of 32 physical processors. Each 32-bit word of each element is stored separately in processor memories, as follows: The low-order 32 bits of all elements are grouped together across processor memories in a field of length $32 \times$ *index-limit* bits. Similarly, the next 32 bits of all elements are grouped together, and so on, up to the high-order bits of all array elements. This data format allows fast hardware-supported access to the individual elements of a shared array.

A region of memory set aside for an array of the format required by CM:aset32-shared must be contiguous in memory. It must therefore be allocated all at once, at a VP ratio of 1, with a single call to CM:allocate-stack-field or to CM:allocate-heap-field. Alternatively, from Lisp, the memory may be allocated within a with-stack-field form at a VP ratio of 1.

An area of CM memory occupied by *array* should be allocated at a VP ratio of 1 as a field whose length in bits is exactly

$$\textit{index-limit} \times \textit{dlen}$$

Shared array memory should be accessed only with the operations CM:aref32-shared and CM:aset32-shared, or with operations that copy the array as a whole from all processors (such as I/O operations). Data in such a region of memory may, however, be reoriented with the CM:transpose32 instruction.

AVAILABLE-MEMORY

Determines the number of bits of memory, per virtual processor, that remain available for allocation on either the heap or the stack.

Formats result ← CM:available-memory

Result An unsigned integer, the number of bits available.

Context This operation is unconditional. It does not depend on the *context-flag*.

The number of bits available for allocation by either CM:allocate-heap-field or CM:allocate-stack-field is returned to the front end as an integer. The return value represents the number of bits available for each virtual processor in the current VP set.

F-F-CEILING

Determines the smallest integral value that is not less than the floating-point source field value in each selected processor and stores it in the floating-point destination field.

Formats CM:f-f-ceiling-1-1L *dest*/*source*, *s*, *e*
CM:f-f-ceiling-2-1L *dest*, *source*, *s*, *e*

Operands

dest	The floating-point destination field.
source	The floating-point source field.
s, *e*	The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if *context-flag*$[k] = 1$ then
$dest[k] \leftarrow \lceil source[k] \rceil$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of $+\infty$, which is stored into the *dest* field as a floating-point-number.

Note that overflow cannot occur.

S-CEILING

The ceiling of the quotient of two signed integer source values is placed in the destination field. Overflow is also computed.

Formats CM:s-ceiling-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The signed integer quotient field.

source1 The signed integer dividend field.

source2 The signed integer divisor field.

dlen For CM:s-ceiling-3-3L, the length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen1 For CM:s-ceiling-3-3L, the length of the *source1* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen2 For CM:s-ceiling-3-3L, the length of the *source2* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the quotient cannot be represented in the destination field; otherwise it is cleared.

test-flag is set if the divisor is zero; otherwise it is cleared.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \left\lceil \dfrac{source1[k]}{source2[k]} \right\rceil$
    if ⟨overflow occurred in processor k⟩ then $overflow\text{-}flag[k] \leftarrow 1$
    else $overflow\text{-}flag[k] \leftarrow 0$
    if $source2[k] = 0$ then
      $test[k] \leftarrow 1$
    else $test[k] \leftarrow 0$

CEILING

The signed integer *source1* operand is divided by the signed integer *source2* operand. The ceiling of the mathematical quotient is stored into the signed integer memory field *dest.*

The *overflow-flag* and *test-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

S-F-CEILING

The floating-point source field values are converted to signed integer values and stored in the destination field.

Formats CM:s-f-ceiling-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands	*dest*	The signed integer destination field.
	source	The floating-point source field.
	len	The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.
	s, *e*	The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap — The fields *dest* and *source* must not overlap in any manner.

Flags — *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context — This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    dest$[k] \leftarrow \lceil$*source*$[k]\rceil$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of $+\infty$. The result is stored into the *dest* field as a signed integer.

U-CEILING

The ceiling of the quotient of two unsigned integer source values is placed in the destination field. Overflow is also computed.

Formats CM:u-ceiling-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The unsigned integer quotient field.

source1 The unsigned integer dividend field.

source2 The unsigned integer divisor field.

dlen For CM:u-ceiling-3-3L, the length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen1 For CM:u-ceiling-3-3L, the length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen2 For CM:u-ceiling-3-3L, the length of the *source2* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the quotient cannot be represented in the destination field; otherwise it is cleared.

test-flag is set if the divisor is zero; otherwise it is cleared.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

if $\textit{context-flag}[k] = 1$ then

$\textit{dest}[k] \leftarrow \left\lceil \dfrac{\textit{source1}[k]}{\textit{source2}[k]} \right\rceil$

if ⟨overflow occurred in processor k⟩ then $\textit{overflow-flag}[k] \leftarrow 1$

else $\textit{overflow-flag}[k] \leftarrow 0$

if $\textit{source2}[k] = 0$ then

$\textit{test}[k] \leftarrow 1$

else $\textit{test}[k] \leftarrow 0$

The unsigned integer *source1* operand is divided by the unsigned integer *source2* operand. The ceiling of the mathematical quotient is stored into the unsigned integer memory field *dest*.

The *overflow-flag* and *test-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

U-F-CEILING

The floating-point source field values are converted to unsigned integer values and stored in the destination field.

Formats CM:u-f-ceiling-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest The unsigned integer destination field.

source The floating-point source field.

len The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *dest* and *source* must not overlap in any manner.

Flags *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    dest $\leftarrow \lceil source \rceil$
    if $\langle$overflow occurred in processor $k\rangle$ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of $+\infty$, which is stored into the *dest* field as an unsigned integer.

CLEAR-ALL-FLAGS

Clears all flags (but not the context bit).

Formats CM:clear-all-flags
CM:clear-all-flags-always

Context The non-always operations are conditional.
The always operations are unconditional.

Definition For every virtual processor k in the *current-vp-set* do
  if (always or *context-flag*$[k] = 1$) then
    test-flag$[k] \leftarrow 0$
    overflow-flag$[k] \leftarrow 0$

Within each processor, all flags for that processor are cleared (but not the context bit).

most recent such operation was CM:cold-boot, then the same virtual processor configuration set up then will be used this time. If the most recent such operation was CM:attach, then the number of virtual processors will be equal to the number of physical processors, and the virtual NEWS grid will have the same shape as the physical NEWS grid.

Bootstrapping a Connection Machine system includes the following actions:

- Evaluating all initialization forms stored in the variable CM:*before-cold-boot-initializations*. This is done before anything else.
- Loading microcode into the Connection Machine microcontroller and initiating microcontroller execution.
- Clearing and initializing the memory of allocated Connection Machine processors.
- Initializing all of the global configuration variables described in section 3.6.
- Initializing the pseudo-random number generator by effectively invoking the operation CM:initialize-random-number-generator with no seed.
- Initializing the system lights-display mode by effectively invoking the operation CM:set-system-leds-mode with an argument of t.
- Evaluating all initialization forms stored in the variable CM:*after-cold-boot-initializations*. This is done after everything else.

If the cold-booting operation fails, then an error is signalled. If it succeeds, then three values are returned: the number of virtual processors, the number of physical processors, and the number of bits available for the user in each virtual processor. (These are exactly the values of the configuration variables CM:*user-cube-address-limit*, CM:*physical-cube-address-limit*, and CM:*user-memory-address-limit*.

In the C/Paris and Fortran/Paris interfaces, the cold-booting operation is performed by a user command cmcoldboot at shell level. See the *Front End Subsystems* manual.

F-COMPARE

Compares two floating-point source values and stores into the signed integer destination field the result `-1`, `0`, or `1` depending on whether the first source value is less than, equal to, or greater than the second source value.

Formats CM:f-compare-3-2L *dest*, *source1*, *source2*, *dlen*, *s*, *e*

Operands

dest The signed integer destination field.

source1 The floating-point first source field.

source2 The floating-point second source field.

dlen The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source1* and *source2* fields. The total length of an operand in this format is $s + e + 1$.

Overlap

The fields *dest* and *source1* must not overlap in any manner. The fields *dest* and *source2* must not overlap in any manner. The fields *source1* and *source2* may overlap in any manner.

Context

This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

```
if context-flag[k] = 1 then
   if source1[k] < source2[k] then
      dest[k] ← −1
   else if source1[k] > source2[k] then
      dest[k] ← 1
   else
      dest[k] ← 0
```

Two operands are compared as floating-point numbers. The destination receives the signed integer value `-1`, `0`, or `1` depending on whether the first source value is less than, equal to, or greater than the second source value.

S-COMPARE

Compares two signed integer source values and stores into the signed integer destination field the result -1, 0, or 1 depending on whether the first source value is less than, equal to, or greater than the second source value.

Formats CM:s-compare-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The signed integer destination field.

source1 The signed integer first source field.

source2 The signed integer second source field.

dlen The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen1 The length of the *source1* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen2 The length of the *source2* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The fields *dest* and *source1* must not overlap in any manner. The fields *dest* and *source2* must not overlap in any manner. The fields *source1* and *source2* may overlap in any manner.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $context\text{-}flag[k] = 1$ then
    if $source1[k] < source2[k]$ then
      $dest[k] \leftarrow -1$
    else if $source1[k] > source2[k]$ then
      $dest[k] \leftarrow 1$
    else
      $dest[k] \leftarrow 0$

Two operands are compared as signed integers. The destination receives the value -1, 0, or 1 depending on whether the first source value is less than, equal to, or greater than the second source value.

U-COMPARE

Compares two unsigned integer source values and stores into the signed integer destination field the result -1, 0, or 1 depending on whether the first source value is less than, equal to, or greater than the second source value.

Formats CM:u-compare-3-3L *dest, source1, source2, dlen, slen1, slen2*

Operands

dest	The signed integer destination field.
source1	The unsigned integer first source field.
source2	The unsigned integer second source field.
dlen	The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.
slen1	The length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.
slen2	The length of the *source2* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The fields *dest* and *source1* must not overlap in any manner. The fields *dest* and *source2* must not overlap in any manner. The fields *source1* and *source2* may overlap in any manner.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    if $\textit{source1}[k] < \textit{source2}[k]$ then
      $\textit{dest}[k] \leftarrow -1$
    else if $\textit{source1}[k] > \textit{source2}[k]$ then
      $\textit{dest}[k] \leftarrow 1$
    else
      $\textit{dest}[k] \leftarrow 0$

Two operands are compared as unsigned integers. The destination receives the signed integer value -1, 0, or 1 depending on whether the first source value is less than, equal to, or greater than the second source value.

F-COS

Calculates, in each selected processor, the cosine of the floating-point source field value and stores it in the floating-point destination field.

Formats CM:f-cos-1-1L *dest/source*, *s*, *e*
CM:f-cos-2-1L *dest*, *source*, *s*, *e*

Operands

dest The floating-point destination field.

source The floating-point source field.

s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $\textit{context-flag}[k] = 1$ then
$\textit{dest}[k] \leftarrow \cos \textit{source}[k]$

The cosine of the value of the *source* field is stored into the *dest* field.

```
typedef enum {CM_news_order, CM_send_order } CM_axis_order_t;

typedef struct CM_axis_descriptor {
  unsigned length;
  unsigned weight;
  CM_axis_order_t ordering;
  unsigned char on_chip_bits;
  unsigned char off_chip_bits;
} * CM_axis_descriptor_t;
```

Actually, this structure has other components as well. C code should use the definition of CM_axis_descriptor from the cmtypes.h include file.

The Fortran/Paris interface defines CM_axis_descriptor as an array:

```
INTEGER RANK,DESCRIPTOR_ARRAY(7,RANK)
```

The elements of each Fortran axis descriptor are defined such that:

$DESCRIPTOR\_ARRAY(1, I)$ is the length of axis I
$DESCRIPTOR\_ARRAY(2, I)$ is the weight of axis I
$DESCRIPTOR\_ARRAY(3, I)$ is the ordering of axis I
$DESCRIPTOR\_ARRAY(4, I)$ is the on-chip bits of axis I
$DESCRIPTOR\_ARRAY(6, I)$ is the off-chip bits of axis I

Thus CM:axis-descriptor-array is, in Fortran, an array of axis descriptor arrays.

The Lisp definitions of the type of the ordering component and of the axis descriptor are shown below.

```
(deftype cm:axis-order () '(member :news-order :send-order))

(defstruct CM:axis-descriptor
  (length 0) (weight 0) (ordering :news-order)
  (on-chip-bits 0) (off-chip-bits 0))
```

The *axis-descriptor-array* operand must be created by first making one axis descriptor for each axis and then using these to assign values to the array elements. An example in C is given below. Notice that *axis1* and *axis2* are *pointers* to axis descriptor structures and that the descriptor structures are zeroed before any values are assigned.

```
CM_geometry_id_t  my_geometry;
CM_axis_descriptor_t  my_geometry_axes[2];
CM_axis_descriptor_t  axis1, axis2;
```

```
axis1 = (cm_axis_descriptor_t)malloc(sizeof(struct CM_axis_descriptor));
axis2 = (cm_axis_descriptor_t)malloc(sizeof(struct CM_axis_descriptor));
bzero(axis1, sizeof(struct CM_axis_descriptor));
bzero(axis2, sizeof(struct CM_axis_descriptor));
axis1->length = 128;
axis2->length = 256;
axis1->weight = 5;
axis2->weight = 10;
axis1->ordering = CM_news_order;
axis2->ordering = CM_news_order;

my_geometry_axes[0] = axis1;
my_geometry_axes[1] = axis2;
my_geometry = CM_create_detailed_geometry(my_geometry_axes, 2);
```

The following example specifies the same axes, descriptor array, and geometry in Lisp. Notice that the constructor CM:make-axis-descriptor is used.

```
(setq my-geometry-axes make-array(2))
(setq axis1
 (CM:make-axis-descriptor :length 128 :weight 5
  :ordering :news-order))
(setq axis2
 (CM:make-axis-descriptor :length 256 :weight 10
  :ordering :news-order)))
(setf (aref my-geometry-axes 0) axis1)
(setf (aref my-geometry-axis 1) axis2)
(setq my-geometry (CM:make-detailed-geometry my-geometry-axes 2)
```

Once the geometry has been created, the user may destroy the descriptors and the array used to provide axis information. All necessary information is copied out of these structures as the geometry is created.

The "length" component of an axis descriptor specifies the length of the axis; it must be a power of two.

The "weight" component of the axis descriptors specifies the relative frequency of interprocessor communication along different axes. For instance, in the above example it is assumed that communication occurs about half as often along *axis1*, which is given a weight of 5, as along *axis2*, which is given a weight of 10. Only the relative values of the weight components matter. The same communication traffic could be specified with weights of 1 and 2, or of 3 and 6. If all weights are 1, it is assumed that all axes are used equally frequently.

Given a set of weight components, Paris lays out the hypercube grid for optimal performance. Virtual processors are mapped onto the physical hypercube in a pattern that exploits the fact that communication is especially rapid among virtual processors within the same physical processor and among virtual processors within the same physical chip.

The "ordering" component of an axis descriptor specifies how NEWS coordinates are mapped onto physical processors for that axis. The value :news-order specifies the usual embedding of the grid into the hypercube such that processors with adjacent NEWS coordinates are in fact neighbors within the hypercube. The value :send-order specifies that, if processor A has a smaller NEWS coordinate than processor B, then A also has a smaller send-address than B. This ordering is rarely used. However, :send-order ordering *is* useful for specific applications such as FFT. The value :framebuffer-order is provided solely for creating VP sets that are used as image buffers (for details, see chapter 1 of the *Generic Display Interface Reference Manual*).

If the "weight" components are all 1, then the mapping of virtual to physical processors can be specified with the "on-chip-bits" and "off-chip-bits" components of the axis descriptors. This is not recommended. To tune performance for communication, use the weight component.

CREATE-GEOMETRY

Creates a new geometry given the grid axis lengths. See also CM:intern-geometry.

Formats result ← CM:create-geometry *dimension-array*, *[rank]*

Operands *dimension-array* A front-end vector of unsigned integer lengths of the grid axes. In the Lisp interface, this may be a list of dimension lengths instead of an array of dimension lengths, at the user's option.

rank An unsigned integer, the rank (number of dimensions) of the *dimension-array*. This must be inbetween 1 and CM:*max-geometry-rank*, inclusive. This argument is not provided when calling Paris from Lisp.

Result A geometry-id, identifying the newly created geometry.

Context This operation is unconditional. It does not depend on the *context-flag*.

The *dimension-array* must be a one-dimensional array of nonnegative integers; each must be a power of two. The product of all these integers must be a multiple of the number of physical processors attached for use by this process.

This operation returns a geometry-id for a newly created geometry whose dimensions are specified by the *dimension-array*. The length of axis j of the resulting geometry will be equal to *dimension-array*$[j]$. Such a geometry-id may then be used to create a VP set, or to respecify the geometry of an existing VP set.

The geometry will be laid out so as to optimize performance under the assumption that the axes are used equally frequently for NEWS communication. The operation CM:create-detailed-geometry may be used instead to get more precise control over layout for performance tuning.

Once the geometry has been created, the user may destroy the array used to provide the dimension information. All necessary information is copied out of this array as the geometry is created.

DEPOSIT-NEWS-COORDINATE

Modifies a send address to reflect a specific NEWS coordinate.

Formats

CM:deposit-news-coordinate-1L *geometry*, *dest*/*send-address*, *axis*, *coordinate*, *slen*

CM:deposit-news-constant-1L *geometry*, *dest*/*send-address*, *axis*, *coordinate-value*, *slen*

Operands

geometry A geometry-id. This geometry determines the NEWS dimensions to be used.

dest The unsigned integer destination field. (In the instruction formats currently provided, the *dest* field is always the same as the *send-address* source field. The length of this field is implicitly the same as *geometry-send-address-length*(*geometry*).)

send-address The unsigned integer send-address field.

axis An unsigned integer immediate operand to be used as the number of a NEWS axis.

coordinate The unsigned integer NEWS coordinate field. field. This specifies the position along the corrsponding axis of the processor whose send address is to be calculated.

coordinate-value An unsigned integer immediate operand to be used as the NEWS coordinate along the specified axis.

slen The length of the *coordinate* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap For CM:deposit-news-coordinate-1L, the *coordinate* field must not overlap the *dest* field.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if *context-flag*$[k] = 1$ then
 dest$[k] \leftarrow$ *deposit-news-coordinate*(*geometry*, *send-address*, *axis*, *coordinate*)
where *deposit-news-coordinate* is as defined on page 33.

This function calculates, within each selected processor, the send-address of a processor that has a specified coordinate along a specified NEWS axis, with all other coordinates equal to those for the processor identified by *send-address*.

FE-DEPOSIT-NEWS-COORDINATE

Calculates on the front end the modification of a send address to reflect a specific NEWS coordinate.

Formats result ← CM:fe-deposit-news-coordinate *geometry*, *send-address*, *axis*, *coordinate*

Operands

geometry A geometry-id. This geometry determines the NEWS dimensions to be used.

send-address An unsigned integer immediate operand to be used as the send address of some processor.

axis An unsigned integer immediate operand to be used as the number of a NEWS axis.

coordinate An unsigned integer immediate operand to be used as the NEWS coordinate along the specified axis.

Result An unsigned integer, the send address of the processor whose coordinate along the specified axis is *coordinate* and whose coordinate along all other axes equals those of *send-address*.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Return *deposit-news-coordinate*(*geometry*, *send-address*, *axis*, *coordinate*) where *deposit-news-coordinate* is as defined on page 33.

This function calculates, entirely on the front end, the send-address of a processor that has a specified coordinate along a specified NEWS axis, with all other coordinates equal to those for the processor identified by *send-address*.

F-EXP

Calculates, in each selected processor, the exponential function e^x of the floating-point source field and stores it in the floating-point destination field.

Formats CM:f-exp-1-1L *dest/source*, *s*, *e*
CM:f-exp-2-1L *dest*, *source*, *s*, *e*

Operands
dest The floating-point destination field.

source The floating-point source field.

s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Flags *overflow-flag* is set if floating-point overflow occurs; otherwise it is unaffected.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    if *source*$[k] = +\infty$ then
      dest$[k] \leftarrow +\infty$
    else if *source*$[k] = -\infty$ then
      dest$[k] \leftarrow +0$
    else
      dest$[k] \leftarrow$ exp *source*$[k]$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$

Call the value of the *source* field s; the value e^s is stored into the *dest* field, where $e \approx 2.718281828\ldots$ is the base of the natural logarithms.

EXTRACT-MULTI-COORDINATE

Determines the NEWS multi-coordinate of a processor specified by send-address.

Formats CM:extract-multi-coordinate-1L *geometry*, *dest*, *axis-mask*, *send-address*, *dlen*

Operands

geometry A geometry-id. This geometry determines the NEWS dimensions to be used.

dest The unsigned integer destination field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

send-address The send-address field. For each processor, this identifies the send-address of some other processor.

dlen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    let $\textit{axis-set} = \{\, m \mid 0 \leq m < r \wedge (\textit{axis-mask}\langle m \rangle = 1)\,\}$
    $\textit{dest}[k] \leftarrow \textit{extract-multi-coordinate}(\textit{geometry}, \textit{axis-set}, \textit{send-address})$

where *extract-multi-coordinate* is as defined on page 34.

This function calculates, within each selected processor, the NEWS multi-coordinate of a processor along specified NEWS axes. The axes are indicated by the *axis-mask* argument; the processor is identified by its send-address.

FE-EXTRACT-MULTI-COORDINATE

Calculates, on the front end, the NEWS multi-coordinate of a processor specified by send-address.

Formats result ← CM:fe-extract-multi-coordinate *geometry, axis-mask, send-address*

Operands

geometry A geometry-id. This geometry determines the NEWS dimensions to be used.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

send-address An unsigned integer immediate operand to be used as the send address of some processor.

Result An unsigned integer, the NEWS multi-coordinate of the specified processor along the specified axes.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Let $axis\text{-}set = \{ m \mid 0 \leq m < r \wedge (axis\text{-}mask\langle m \rangle = 1) \}$
Return $extract\text{-}multi\text{-}coordinate(geometry, axis\text{-}set, send\text{-}address)$
where *extract-multi-coordinate* is as defined on page 34.

This function calculates, entirely on the front end, the NEWS multi-coordinate of a processor along specified NEWS axes. The axes are indicated by the *axis-mask* argument; the processor is identified by its send-address.

EXTRACT-NEWS-COORDINATE

Determines the NEWS coordinate of a processor specified by send-address.

Formats CM:extract-news-coordinate-1L *geometry*, *dest*, *axis*, *send-address*, *dlen*

Operands

geometry A geometry-id. This geometry determines the NEWS dimensions to be used.

dest The unsigned integer destination field.

axis An unsigned integer immediate operand to be used as the number of a NEWS axis.

send-address The send-address field. For each processor, this identifies the send-address of some other processor.

dlen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \textit{extract-news-coordinate}(\textit{geometry}, \textit{axis}, \textit{send-address})$

where *extract-news-coordinate* is as defined on page 33.

This function calculates, within each selected processor, the NEWS coordinate of a processor along a specified NEWS axis. The axis is indicated by the *axis* argument; the processor is identified by its send-address.

S-FLOOR

The floor of the quotient of two signed integer source values is placed in the destination field. Overflow is also computed.

Formats CM:s-floor-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The signed integer quotient field.

source1 The signed integer dividend field.

source2 The signed integer divisor field.

dlen For CM:s-floor-3-3L, the length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen1 For CM:s-floor-3-3L, the length of the *source1* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen2 For CM:s-floor-3-3L, the length of the *source2* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the quotient cannot be represented in the destination field; otherwise it is cleared.

test-flag is set if the divisor is zero; otherwise it is cleared.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    $dest[k] \leftarrow \left\lfloor \frac{source1[k]}{source2[k]} \right\rfloor$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$
    else *overflow-flag*$[k] \leftarrow 0$
    if $source2[k] = 0$ then
      $test[k] \leftarrow 1$
    else $test[k] \leftarrow 0$

FLOOR

The signed integer *source1* operand is divided by the signed integer *source2* operand. The floor of the mathematical quotient is stored into the signed integer memory field *dest*.

The *overflow-flag* and *test-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

S-F-FLOOR

Calculates, in each selected processsor, the largest integer that is not greater than a specified floating-point value and stores the result as a signed integer field.

Formats CM:s-f-floor-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest The signed integer destination field.

source The floating-point source field.

len The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *dest* and *source* must not overlap in any manner.

Flags *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    dest$[k] \leftarrow \lfloor$*source*$[k]\rfloor$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$
    else *overflow-flag*$[k] \leftarrow 0$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of $-\infty$, which is stored into the *dest* field as a signed integer.

U-FLOOR

The floor of the quotient of two unsigned integer source values is placed in the destination field. Overflow is also computed.

Formats CM:u-floor-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The unsigned integer quotient field.

source1 The unsigned integer dividend field.

source2 The unsigned integer divisor field.

dlen For CM:s-floor-3-3L, the length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen1 For CM:s-floor-3-3L, the length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen2 For CM:s-floor-3-3L, the length of the *source2* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the quotient cannot be represented in the destination field; otherwise it is cleared.

test-flag is set if the divisor is zero; otherwise it is cleared.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    $dest[k] \leftarrow \left\lfloor \frac{source1[k]}{source2[k]} \right\rfloor$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$
    else *overflow-flag*$[k] \leftarrow 0$
    if $source2[k] = 0$ then
      $test[k] \leftarrow 1$
    else $test[k] \leftarrow 0$

FLOOR

The unsigned integer *source1* operand is divided by the unsigned integer *source2* operand. The floor of the mathematical quotient is stored into the unsigned integer memory field *dest.*

The *overflow-flag* and *test-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

U-F-FLOOR

Converts floating-point source field values into unsigned integers by rounding towards $-\infty$.

Formats CM:u-f-floor-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest The unsigned integer destination field.

source The floating-point source field.

len The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *dest* and *source* must not overlap in any manner.

Flags *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    dest $\leftarrow \lfloor source \rfloor$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of $-\infty$. The result is stored into the *dest* field as an unsigned integer.

FE-FROM-GRAY-CODE

Calculates, on the front end, the Gray code representation of a specified integer.

Formats result $\leftarrow$ CM:fe-from-gray-code *code*

Operands *code* An unsigned integer immediate operand to be used as the Gray encoding, represented as a nonnegative integer.

Result An unsigned integer, the nonnegative integer represented by *code*.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Let $n = \textit{integer-length}(\textit{code})$

Return $\bigoplus_{j=0}^{n-1} \left\lfloor \frac{\textit{code}}{2^j} \right\rfloor$

This function calculates, entirely on the front end, the integer represented by a bit-string encoding *code* in a particular reflected binary Gray code.

Note that the binary value 0 is always equivalent to a Gray code string that is all 0-bits.

GEOMETRY-SEND-ADDRESS-LENGTH

Returns the number of bits needed to represent a send-address.

Formats result ← CM:geometry-send-address-length *geometry-id*

Operands *geometry-id* A geometry-id.

Result An unsigned integer, the number of bits required to represent a send-address for a processor in the specified geometry.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Let $n = rank(geometry\text{-}id)$

Return $\sum_{j=0}^{n-1} integer\text{-}length(axis\text{-}descriptors(geometry\text{-}id)[j].length - 1)$

This operation returns the number of bits required to represent a send-address for a virtual processor in any VP set whose geometry is the one specified by the *geometry-id*. This will be equal to the sum of the numbers of bits needed to represent NEWS coordinates for all the axes.

GEOMETRY-SERIAL-NUMBER

Assigns a unique number to the specified geometry.

Formats result ← CM:geometry-serial-number *geometry-id*

Operands *geometry-id* A geometry-id. This geometry-id must be obtained by calling CM:create-geometry or CM:create-detailed-geometry.

Result The serial number that uniquely identifies the geometry.

Context This operation is unconditional. It does not depend on the *context-flag*.

A unique number, the serial number, is assigned to the specified geometry. This facilitates geometry-based caching; geometry serial numbers are useful as hash table keys.

Note that geometry-id's are not unique identifiers. After a geometry is deallocated, its id may be reused for another geometry. In contrast, geometry serial numbers are guaranteed to be unique.

GEOMETRY-TOTAL-PROCESSORS

Returns the number of virtual processors for a geometry.

Formats result $\leftarrow$ CM:geometry-total-processors *geometry-id*

Operands *geometry-id* A geometry-id.

Result An unsigned integer, the total number of processors in the specified geometry.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Let $n = \mathit{rank}(\mathit{geometry\text{-}id})$

Return $\prod_{j=0}^{n-1} \mathit{axis\text{-}descriptors}(\mathit{geometry\text{-}id})[j].\mathit{length}$

This operation returns the total number of virtual processors in any VP set whose geometry is the one specified by the *geometry-id*. This will be equal to the product of the lengths of all the axes.

GEOMETRY-TOTAL-VP-RATIO

Returns the total VP ratio for a specified geometry.

Formats result ← CM:geometry-total-vp-ratio *geometry-id*

Operands *geometry-id* A geometry-id.

Result An unsigned integer, the number of virtual processors represented within each physical processor for the specified geometry.

Context This operation is unconditional. It does not depend on the *context-flag*.

Definition Let $n = rank(geometry\text{-}id)$

Return $\prod_{j=0}^{n-1} axis\text{-}descriptor(geometry\text{-}id)[j].vp\text{-}ratio$

This operation returns the total VP ratio for a specified geometry. This is equal to the total number of virtual processors for the geometry, divided by the total number of physical processors.

GET

Each selected processor gets a message from a specified source processor, possibly itself. A source processor may supply messages even if it is not selected. Messages are all retrieved from the same memory address within each source processor, and all the source processors may be in a VP set different from the VP set of the destination processors.

Formats CM:get-1L *dest*, *send-address*, *source*, *len*

Operands

dest The destination field.

send-address The send-address field. For each processor, this indicates from which processor a message is retrieved.

source The source field.

len The length of the *dest* and *source* fields.

Overlap The *send-address* and *source* may overlap in any manner. The *dest* field may overlap with the *send-address* or *source* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *source* only if at most one of them will be used within each processor.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $context\text{-}flag[k] = 1$ then
$dest[k] \leftarrow source[send\text{-}address[k]]$

For every selected processor p_d, a message *length* bits long is sent to p_d from the processor p_s whose send-address is in the field *send-address* in the memory of processor p_d. The message is taken from the *source* field within processor p_s and is stored into the field at location *dest* within processor p_d. Although the *send-address* operand is a field in the VP set of the destination processors, its value must specify a valid send address for *source*, which may belong to a different VP set.

Note that more than one selected processor may request data from the same source processor p_s, in which case the same data is sent to each of the requesting processors.

GET-AREF32

Each selected processor gets a message from a specified array field within any specified source processor (possibly itself). A source processor may supply messages even if it is not selected. Messages are all retrieved from the same memory address within each source processor.

Formats CM:get-aref32-2L *dest*, *send-address*, *array*, *index*, *dlen*, *index-len*, *index-limit*

Operands

dest The destination field.

send-address The send-address field. For each processor, this indicates from which processor a message is retrieved.

array The source array field. This must be stored in the special format required by CM:aref32.

index The unsigned integer index into the array field. This is used as a per-processor index into *array*. It specifies portions of the *array* memory area in increments of *dlen*.

dlen The length of the *dest* field.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of *array*.

Overlap The *send-address* and *array* may overlap in any manner. The *dest* field may overlap with the *send-address* or *array* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *array* only if at most one of them will be used within each processor.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    if $\textit{index}[k] < \textit{index-limit}$ then
      let $r = \textit{geometry-total-vp-ratio}(\textit{geometry}(\textit{current-vp-set}))$
      let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
      let $i = \textit{index}[k]$

for all j such that $0 \leq j < dlen$ do
  let $q = send\text{-}address[k] - m \times r + (j \bmod 32) \times r$
  let $b = i + \left\lfloor \frac{j}{32} \right\rfloor$
  $dest[k]\langle j\rangle \leftarrow array[q]\langle b\rangle$
else
  $\langle$error$\rangle$

For every selected processor p_d, a message *length* bits long is sent to p_d from the processor p_s whose send-address is in the field *send-address* in the memory of processor p_d. The message is taken from the *array* field within processor p_s as if by the operation aref32 and is stored into the field at location *dest* within processor p_d.

Note that more than one selected processor may request data from the same source processor p_s, possibly from different locations within the *array*. Note also that in each case the array element to be sent from processor p_s to processor p_d is determined by the value of *index* within p_d, not the value within p_s.

GLOBAL-U-MAX

One unsigned integer is examined in every selected processor, and the largest of all these integers is returned to the front end as an unsigned integer.

Formats result ← CM:global-u-max-1L *source*, *len*

Operands *source* The unsigned integer source field.

len The length of the *source* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Result An unsigned integer, the largest of the *source* fields.

Overlap There are no constraints, because overlap is not possible.

Flags *test-flag* is set if the value in a particular processor equals the maximum; otherwise it is cleared.

Context This operation is conditional. The result returned depends only upon processors whose *context-flag* is 1.

Definition Let $S = \{ m \mid m \in \textit{current-vp-set} \land \textit{context-flag}[m] = 1 \}$
If $|S| = 0$ then
  return $2^{len} - 1$ to front end
else
  let $R = \left(\max_{m \in S} \textit{source}[m] \right)$
  For every virtual processor k in the *current-vp-set* do
    if $\textit{context-flag}[k] = 1$ then
      if $\textit{source}[k] = R$ then
        $\textit{test-flag}[k] \leftarrow 1$
      else
        $\textit{test-flag}[k] \leftarrow 0$
  return R to front end

The CM:global-u-max operation returns the largest of the unsigned-integer *source* fields of all selected processors. This largest value is sent to the front-end computer as an unsigned integer and returned as the result of the operation. In addition, the *test-flag* is set in every selected processor whose field is equal to the finally computed value, and is cleared in all other selected processors. If there are no selected processors, then the value 0 is returned.

GLOBAL-U-MAX-S-INTLEN

One signed integer is examined in every selected processor, and the largest *length* of all these integers is returned to the front end as an unsigned integer.

Formats result $\leftarrow$ CM:global-u-max-s-intlen-1L *source*, *len*

Operands *source* The signed integer source field.

len The length of the *source* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Result An unsigned integer, the length of the *source* field value of greatest length.

Overlap There are no constraints, because overlap is not possible.

Flags *test-flag* is set if the value in a particular processor has a length equal to the maximum; otherwise it is cleared.

Context This operation is conditional. The result returned depends only upon processors whose *context-flag* is 1.

Definition Let $S = \{ m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \}$
If $|S| = 0$ then
  return -2^{len-1} to front end
else
  let $R = \left(\max_{m \in S} \left\lceil \log_2 \left(\frac{1}{2} + \left| \frac{1}{2} + \textit{source}[m] \right| \right) \right\rceil \right)$
  For every virtual processor k in the *current-vp-set* do
    if $\textit{context-flag}[k] = 1$ then
      if $\textit{source}[k] = R$ then
        $\textit{test-flag}[k] \leftarrow 1$
      else
        $\textit{test-flag}[k] \leftarrow 0$
  return R to front end

The CM:global-u-max-s-intlen operation computes the integer-length of each signed integer *source* value. The largest length is sent to the front-end computer as an unsigned integer and returned as the result of the operation. In addition, the *test-flag* is set in every selected processor whose field is equal to the finally computed value, and is cleared in all other selected processors. If there are no selected processors, then the value 0 is returned.

A call to CM:global-u-max-s-intlen-1L is equivalent to the sequence

CM:u-integer-length-1L *temp*, *source*, *len*, *len*
CM:global-u-max-1L *temp*, *len*

but may be faster.

GLOBAL-F-MIN

One floating-point number is examined in every selected processor, and the smallest of all these integers (that is, the one closest to $-\infty$) is returned to the front end as a floating-point number.

Formats result $\leftarrow$ CM:global-f-min-1L *source*, *s*, *e*

Operands

source The floating-point source field.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Result A floating-point number, the smallest of the *source* fields.

Overlap There are no constraints, because overlap is not possible.

Flags *test-flag* is set if the value in a particular processor equals the minimum; otherwise it is cleared.

Context This operation is conditional. The result returned depends only upon processors whose *context-flag* is 1.

Definition

Let $S = \{\, m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \,\}$
If $|S| = 0$ then
  return $+\infty$ to front end
else
  let $R = \left(\min_{m \in S} \textit{source}[m]\right)$
  For every virtual processor k in the *current-vp-set* do
    if $\textit{context-flag}[k] = 1$ then
      if $\textit{source}[k] = R$ then
        $\textit{test-flag}[k] \leftarrow 1$
      else
        $\textit{test-flag}[k] \leftarrow 0$
  return R to front end

The CM:global-f-min operation returns the smallest (that is, closest to $-\infty$) of the floating-point *source* fields of all selected processors. This smallest value is sent to the front-end computer as a floating-point number and returned as the result of the operation. In addition, the *test-flag* is set in every selected processor whose field is equal to the finally computed value, and is cleared in all other selected processors. If there are no selected processors, then the value $+\infty$ is returned.

GLOBAL-S-MIN

One signed integer is examined in every selected processor, and the smallest of all these integers (that is, the one closest to $-\infty$) is returned to the front end as a signed integer.

Formats result $\leftarrow$ CM:global-s-min-1L *source*, *len*

Operands *source* The signed integer source field.

len The length of the *source* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Result A signed integer, the smallest of the *source* fields.

Overlap There are no constraints, because overlap is not possible.

Flags *test-flag* is set if the value in a particular processor equals the minimum; otherwise it is cleared.

Context This operation is conditional. The result returned depends only upon processors whose *context-flag* is 1.

Definition
Let $S = \{ m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \}$
If $|S| = 0$ then
  return -2^{len-1} to front end
else
  let $R = \left(\min_{m \in S} \textit{source}[m] \right)$ to front end
  For every virtual processor k in the *current-vp-set* do
    if $\textit{context-flag}[k] = 1$ then
      if $\textit{source}[k] = R$ then
        $\textit{test-flag}[k] \leftarrow 1$
      else
        $\textit{test-flag}[k] \leftarrow 0$
  return R to front end

The CM:global-s-min operation returns the smallest (that is, closest to $-\infty$) of the signed-integer *source* fields of all selected processors. This smallest value is sent to the front-end computer as a signed integer and returned as the result of the operation. In addition, the *test-flag* is set in every selected processor whose field is equal to the finally computed value, and is cleared in all other selected processors. If there are no selected processors, then the value $2^{len-1} - 1$ is returned.

GLOBAL-U-MIN

One unsigned integer is examined in every selected processor, and the smallest of all these integers is returned to the front end as an unsigned integer.

Formats result $\leftarrow$ CM:global-u-min-1L *source*, *len*

Operands *source* The unsigned integer source field.

len The length of the *source* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Result An unsigned integer, the smallest of the *source* fields.

Overlap There are no constraints, because overlap is not possible.

Flags *test-flag* is set if the value in a particular processor equals the minimum; otherwise it is cleared.

Context This operation is conditional. The result returned depends only upon processors whose *context-flag* is 1.

Definition Let $S = \{ m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \}$
If $|S| = 0$ then
  return 0 to front end
else
  let $R = \left(\min_{m \in S} \textit{source}[m] \right)$
  For every virtual processor k in the *current-vp-set* do
    if $\textit{context-flag}[k] = 1$ then
      if $\textit{source}[k] = R$ then
        $\textit{test-flag}[k] \leftarrow 1$
      else
        $\textit{test-flag}[k] \leftarrow 0$
  return R to front end

The CM:global-u-min operation returns the smallest (that is, closest to $-\infty$) of the unsigned-integer *source* fields of all selected processors. This smallest value is sent to the front-end computer as an unsigned integer and returned as the result of the operation. In addition, the *test-flag* is set in every selected processor whose field is equal to the finally computed value, and is cleared in all other selected processors. If there are no selected processors, then the value $2^{len} - 1$ is returned.

LOAD-CONTEXT

Unconditionally reads a bit from memory and loads it into the context bit.

Formats CM:load-context *source*

Operands *source* The source bit (a one-bit field).

Context This operation is unconditional.

Definition For every virtual processor k in the *current-vp-set* do
$context\text{-}flag[k] \leftarrow source[k]$

Within each processor, a bit is read from memory and unconditionally loaded into the context bit for that processor.

LOAD-flag

Reads a bit from memory and loads it into a flag.

Formats

CM:load-test	*source*
CM:load-test-always	*source*
CM:load-overflow	*source*
CM:load-overflow-always	*source*

Operands *source* The source bit (a one-bit field).

Context The non-always operations are conditional.

The always operations are unconditional.

Definition For every virtual processor k in the *current-vp-set* do
if $\mathit{context\text{-}flag}[k] = 1$ then
$\mathit{flag}[k] \leftarrow \mathit{source}[k]$

where *flag* is *test-flag* or *overflow-flag*, as appropriate.

Within each processor, a bit is read from memory and loaded into the indicated flag for that processor.

MULTISPREAD-F-ADD

The destination field in every selected processor receives the sum of the floating-point source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-f-add-1L *dest*, *source*, *axis-mask*, *s*, *e*

Operands

dest	The floating-point destination field.
source	The floating-point source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
s, *e*	The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $\textit{context-flag}[k] = 1$ then
let $g = \textit{geometry}(\textit{current-vp-set})$
let $r = \textit{rank}(g)$
let $\textit{axis-set} = \{\, m \mid 0 \le m < r \wedge (\textit{axis-mask}\langle m \rangle = 1) \,\}$
let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1 \,\}$

$$\textit{dest}[k] \leftarrow \left(\sum_{m \in C_k} \textit{source}[m] \right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-f-add operation combines *source* fields by performing floating-point addition.

A call to CM:multispread-f-add-1L is equivalent to the sequence

CM:f-move-zero-always-1L *temp*, *s*, *e*
CM:f-move-1L *temp*, *source*, *s*, *e*
CM:store-context *ctemp*
CM:set-context

for all integers j, $0 \leq j < rank(geometry(current\text{-}vp\text{-}set))$, in any sequential order, do
  if $axis\text{-}mask\langle j\rangle = 1$ then
    CM:spread-with-f-add-1L *temp*, *temp*, j, *s*, *e*
CM:load-context *ctemp*
CM:f-move-1L *dest*, *temp*, *s*, *e*

but may be faster.

MULTISPREAD-S-ADD

The destination field in every selected processor receives the sum of the signed integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-s-add-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest	The signed integer destination field.
source	The signed integer source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
len	The length of the *dest* and *source* fields. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

if $\textit{context-flag}[k] = 1$ then
let $g = \textit{geometry}(\textit{current-vp-set})$
let $r = \textit{rank}(g)$
let $\textit{axis-set} = \{\, m \mid 0 \le m < r \wedge (\textit{axis-mask}\langle m\rangle = 1)\,\}$
let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1\,\}$

$$\textit{dest}[k] \leftarrow \left(\sum_{m \in C_k} \textit{source}[m]\right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-s-add operation combines *source* fields by performing signed integer addition.

MULTISPREAD-U-ADD

The destination field in every selected processor receives the sum of the unsigned integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-u-add-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest	The unsigned integer destination field.
source	The unsigned integer source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
len	The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $context\text{-}flag[k] = 1$ then
let $g = geometry(current\text{-}vp\text{-}set)$
let $r = rank(g)$
let $axis\text{-}set = \{\, m \mid 0 \le m < r \wedge (axis\text{-}mask\langle m\rangle = 1)\,\}$
let $C_k = \{\, m \mid m \in hyperplane(g, k, axis\text{-}set) \wedge context\text{-}flag[m] = 1\,\}$

$$dest[k] \leftarrow \left(\sum_{m \in C_k} source[m]\right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-u-add operation combines *source* fields by performing unsigned integer addition.

MULTISPREAD-COPY

The destination field in every selected processor receives a copy of the source value from a particular value within its scan subclass.

Formats CM:multispread-copy-1L *dest*, *source*, *axis-mask*, *len*, *multi-coordinate*

Operands

dest The unsigned integer destination field.

source The unsigned integer source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

multi-coordinate An unsigned integer, the multi-coordinate indicating which element of each hyperplane is to be replicated throughout that hyperplane.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if $\textit{context-flag}[k] = 1$ then
  let $g = \textit{geometry}(\textit{current-vp-set})$
  let $r = \textit{rank}(g)$
  let $\textit{axis-set} = \{\, m \mid 0 \le m < r \wedge (\textit{axis-mask}\langle m \rangle = 1)\,\}$
  let $c = \textit{deposit-multi-coordinate}(g, k, \textit{axis-set}, \textit{multi-coordinate})$
  $\textit{dest}[k] \leftarrow \textit{source}[c]$

where *deposit-multi-coordinate* is as defined on page 34.

See section 5.16 on page 34 for a general description of multispread operations.

MULTISPREAD-LOGAND

The destination field in every selected processor receives the bitwise logical AND of the source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-logand-1L *dest*, *source*, *axis-mask*, *len*

Operands *dest* The destination field.

source The source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two bit fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $\mathit{context\text{-}flag}[k] = 1$ then
let $g = \mathit{geometry}(\mathit{current\text{-}vp\text{-}set})$
let $r = \mathit{rank}(g)$
let $\mathit{axis\text{-}set} = \{\, m \mid 0 \le m < r \wedge (\mathit{axis\text{-}mask}\langle m\rangle = 1)\,\}$
let $C_k = \{\, m \mid m \in \mathit{hyperplane}(g, k, \mathit{axis\text{-}set}) \wedge \mathit{context\text{-}flag}[m] = 1\,\}$

$$\mathit{dest}[k] \leftarrow \left(\bigwedge_{m \in C_k} \mathit{source}[m] \right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-logand operation combines *source* fields by performing bitwise logical AND operations.

MULTISPREAD-LOGIOR

The destination field in every selected processor receives the bitwise logical inclusive OR of the source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-logior-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest The destination field.

source The source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two bit fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

if $\textit{context-flag}[k] = 1$ then

let $g = \textit{geometry}(\textit{current-vp-set})$

let $r = \textit{rank}(g)$

let $\textit{axis-set} = \{\, m \mid 0 \leq m < r \wedge (\textit{axis-mask}\langle m \rangle = 1) \,\}$

let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1 \,\}$

$$\textit{dest}[k] \leftarrow \left(\bigvee_{m \in C_k} \textit{source}[m] \right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-logior operation combines *source* fields by performing bitwise logical inclusive OR operations.

MULTISPREAD-LOGXOR

The destination field in every selected processor receives the bitwise logical exclusive OR of the source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-logxor-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest The destination field.

source The source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two bit fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if $\mathit{context\text{-}flag}[k] = 1$ then
  let $g = \mathit{geometry}(\mathit{current\text{-}vp\text{-}set})$
  let $r = \mathit{rank}(g)$
  let $\mathit{axis\text{-}set} = \{\, m \mid 0 \le m < r \wedge (\mathit{axis\text{-}mask}\langle m \rangle = 1)\,\}$
  let $C_k = \{\, m \mid m \in \mathit{hyperplane}(g, k, \mathit{axis\text{-}set}) \wedge \mathit{context\text{-}flag}[m] = 1 \,\}$

$$\mathit{dest}[k] \leftarrow \left(\bigoplus_{m \in C_k} \mathit{source}[m] \right)$$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-logxor operation combines *source* fields by performing bitwise logical exclusive OR operations.

MULTISPREAD-F-MAX

The destination field in every selected processor receives the largest of the floating-point source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-f-max-1L *dest*, *source*, *axis-mask*, *s*, *e*

Operands

dest The floating-point destination field.

source The floating-point source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\mathit{context\text{-}flag}[k] = 1$ then
    let $g = \mathit{geometry}(\mathit{current\text{-}vp\text{-}set})$
    let $r = \mathit{rank}(g)$
    let $\mathit{axis\text{-}set} = \{\, m \mid 0 \leq m < r \wedge (\mathit{axis\text{-}mask}\langle m \rangle = 1)\,\}$
    let $C_k = \{\, m \mid m \in \mathit{hyperplane}(g, k, \mathit{axis\text{-}set}) \wedge \mathit{context\text{-}flag}[m] = 1\,\}$
    $\mathit{dest}[k] \leftarrow \left(\max_{m \in C_k} \mathit{source}[m]\right)$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-f-max operation combines *source* fields by performing a floating-point maximum operation.

MULTISPREAD-S-MAX

The destination field in every selected processor receives the largest of the signed integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-s-max-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest	The signed integer destination field.
source	The signed integer source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
len	The length of the *dest* and *source* fields. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap: The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context: This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

if $\mathit{context\text{-}flag}[k] = 1$ then

let $g = \mathit{geometry}(\mathit{current\text{-}vp\text{-}set})$

let $r = \mathit{rank}(g)$

let $\mathit{axis\text{-}set} = \{\, m \mid 0 \leq m < r \wedge (\mathit{axis\text{-}mask}\langle m \rangle = 1)\,\}$

let $C_k = \{\, m \mid m \in \mathit{hyperplane}(g, k, \mathit{axis\text{-}set}) \wedge \mathit{context\text{-}flag}[m] = 1\,\}$

$\mathit{dest}[k] \leftarrow \left(\max_{m \in C_k} \mathit{source}[m]\right)$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-s-max operation combines *source* fields by performing a signed integer maximum operation.

MULTISPREAD-U-MAX

The destination field in every selected processor receives the largest of the unsigned integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-u-max-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest The unsigned integer destination field.

source The unsigned integer source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    let $g = \textit{geometry}(\textit{current-vp-set})$
    let $r = \textit{rank}(g)$
    let $\textit{axis-set} = \{\, m \mid 0 \leq m < r \wedge (\textit{axis-mask}\langle m\rangle = 1)\,\}$
    let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1 \,\}$
    $\textit{dest}[k] \leftarrow \left(\max\limits_{m \in C_k} \textit{source}[m]\right)$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-u-max operation combines *source* fields by performing an unsigned integer maximum operation.

MULTISPREAD-F-MIN

The destination field in every selected processor receives the smallest of the floating-point source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-f-min-1L *dest*, *source*, *axis-mask*, *s*, *e*

Operands

dest	The floating-point destination field.
source	The floating-point source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
s, *e*	The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    let $g = \textit{geometry}(\textit{current-vp-set})$
    let $r = \textit{rank}(g)$
    let $\textit{axis-set} = \{\, m \mid 0 \le m < r \wedge (\textit{axis-mask}\langle m \rangle = 1)\,\}$
    let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1\,\}$
    $$\textit{dest}[k] \leftarrow \left(\min_{m \in C_k} \textit{source}[m]\right)$$
where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-f-min operation combines *source* fields by performing a floating-point minimum operation.

MULTISPREAD-S-MIN

The destination field in every selected processor receives the smallest of the signed integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-s-min-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest	The signed integer destination field.
source	The signed integer source field.
axis-mask	An unsigned integer, the mask indicating a set of NEWS axes.
len	The length of the *dest* and *source* fields. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    let $g = \textit{geometry}(\textit{current-vp-set})$
    let $r = \textit{rank}(g)$
    let $\textit{axis-set} = \{\, m \mid 0 \le m < r \wedge (\textit{axis-mask}\langle m \rangle = 1) \,\}$
    let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1 \,\}$
    $\textit{dest}[k] \leftarrow \left(\min\limits_{m \in C_k} \textit{source}[m] \right)$

where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-s-min operation combines *source* fields by performing a signed integer minimum operation.

MULTISPREAD-U-MIN

The destination field in every selected processor receives the smallest of the unsigned integer source fields from all processors in the same hyperplane through the NEWS grid.

Formats CM:multispread-u-min-1L *dest*, *source*, *axis-mask*, *len*

Operands

dest The unsigned integer destination field.

source The unsigned integer source field.

axis-mask An unsigned integer, the mask indicating a set of NEWS axes.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if $\textit{context-flag}[k] = 1$ then
let $g = \textit{geometry}(\textit{current-vp-set})$
let $r = \textit{rank}(g)$
let $\textit{axis-set} = \{\, m \mid 0 \leq m < r \wedge (\textit{axis-mask}\langle m \rangle = 1)\,\}$
let $C_k = \{\, m \mid m \in \textit{hyperplane}(g, k, \textit{axis-set}) \wedge \textit{context-flag}[m] = 1 \,\}$
$$\textit{dest}[k] \leftarrow \left(\min_{m \in C_k} \textit{source}[m] \right)$$
where *hyperplane* is as defined on page 36.

See section 5.16 on page 34 for a general description of multispread operations. The CM:multispread-u-min operation combines *source* fields by performing an unsigned integer minimum operation.

MY-NEWS-COORDINATE

Stores the NEWS coordinate of each selected processor along a specified NEWS axis into a destination field within that processor.

Formats CM:my-news-coordinate-1L *dest*, *axis*, *dlen*

Operands	*dest*	The unsigned integer destination field.
	axis	An unsigned integer immediate operand to be used as the number of a NEWS axis.
	dlen	The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.
Context		This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
if *context-flag*$[k] = 1$ then
let $g = geometry(current\text{-}vp\text{-}set)$
$dest[k] \leftarrow extract\text{-}news\text{-}coordinate(g, axis, k)$

where *extract-news-coordinate* is as defined on page 33.

This function calculates, within each selected processor, the NEWS coordinate of that processor along a specified NEWS axis.

MY-SEND-ADDRESS

Stores the send-address of each selected processor into a destination field in that processor.

Formats CM:my-send-address *dest*

Operands *dest* The unsigned integer destination field. This must be no less than the value returned by CM:geometry-send-address-length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow k$

This function stores into the *dest* field, within each selected processor, the send-address of that processor.

The *source1* field (the base) is raised to the power *source2* (the exponent).

The result is stored into the memory field *dest.* The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand.

S-S-POWER

Raises a signed integer to a signed integer power.

Formats

CM:s-s-power-3-3L	*dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*
CM:s-s-power-2-1L	*dest/source1*, *source2*, *len*
CM:s-s-power-3-1L	*dest*, *source1*, *source2*, *len*
CM:s-s-power-constant-2-1L	*dest/source1*, *source2-value*, *len*
CM:s-s-power-constant-3-1L	*dest*, *source1*, *source2-value*, *len*
CM:s-s-power-constant-3-2L	*dest*, *source1*, *source2-value*, *dlen*, *slen*

Operands

dest The signed integer destination field.

source1 The signed integer base field.

source2 The signed integer exponent field.

source2-value A signed integer immediate operand to be used as the second source.

len The length of the *dest*, *source1*, and *source2* fields. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

dlen For CM:s-s-power-3-3L and CM:s-s-power-constant-3-2L, the length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen For CM:s-s-power-constant-3-2L, the length of the *source1* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen1 For CM:s-s-power-3-3L, the length of the *source1* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

slen2 For CM:s-s-power-3-3L, the length of the *source2* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the result cannot be represented in the destination field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

if *context-flag*$[k] = 1$ then
    if *source2*$[k] < 0$ then
        dest$[k] \leftarrow 0$
    else if *source2*$[k] = 0$ then
        dest$[k] \leftarrow 1$
    else
    dest$[k] \leftarrow (\mathit{source1}[k])^{\mathit{source2}[k]}$
    if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$
    else *overflow-flag*$[k] \leftarrow 0$

The *source1* field (the base) is raised to the power *source2* (the exponent). If the exponent is negative, the result is always 0; if the exponent is zero, the result is always 1.

The result is stored into the memory field *dest*. The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand.

The *overflow-flag* may be altered by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

The constant operand *source1-value* or *source2-value* should be a signed integer front-end value. The operation is performed properly in all cases; the constant need not be representable in the number of bits specified by *len*.

POWER-UP

This operation resets the Nexus, causing all front-end computers to become logically detached from the Connection Machine system.

Formats CM:power-up

Context This operation is unconditional. It does not depend on the *context-flag*.

This function resets the state of the Nexus, causing all front-end computers to become logically detached from the Connection Machine system. When a Connection Machine system is first powered up or is to be completely reset for other reasons, this is the first operation to perform. Any of the front-end computers may be used to do it.

If users on other front-end computers are actively using the Connection Machine system, their computations will be disrupted. Normally all the front-end computers are connected not only through the Connection Machine Nexus but also through some sort of communications network; a front end that executes CM:power-up will attempt to send messages through this network to the other front-end computers on the same Nexus indicating that a CM:power-up operation is being performed.

F-RANK

The destination field in every selected processor receives the rank of that processor's key among all keys in the scan set for that processor.

Formats CM:f-rank-2L *dest*, *source*, *axis*, *dlen*, *s*, *e*, *direction*, *smode*, *sbit*

Operands

- *dest* — The unsigned integer destination field.
- *source* — The floating-point source field. This is the sort key.
- *axis* — An unsigned integer immediate operand to be used as the number of a NEWS axis.
- *dlen* — The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This must be no larger than the value returned by CM:geometry-coordinate-length.
- *s*, *e* — The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.
- *direction* — Either :upward or :downward.
- *smode* — Either :none, :start-bit, or :segment-bit.
- *sbit* — The segment bit or start bit (a one-bit field). If *smode* is :none then this may be CM:*no-field*.

Overlap — The fields *source* and *sbit* may overlap in any manner. However, the *source* and *sbit* fields must not overlap the *dest* field.

Context — This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do

  if $\mathit{context\text{-}flag}[k] = 1$ then

    let $g = \mathit{geometry}(\mathit{current\text{-}vp\text{-}set})$

    let $S_k = \mathit{scan\text{-}set}(g, k, \mathit{axis}, \mathit{direction}, \mathit{smode}, \mathit{sbit})$

    case *direction* of

      :upward:

        let $L_k = \{\, m \mid m \in S_k \wedge ((\mathit{source}[m] < \mathit{source}[k]) \vee (\mathit{source}[m] = \mathit{source}[k]$

      :downward:

        let $L_k = \{\, m \mid m \in S_k \wedge ((\mathit{source}[m] > \mathit{source}[k]) \vee (\mathit{source}[m] = \mathit{source}[k]$

      $\mathit{dest}[k] \leftarrow |L_k|$

where *scan-set* is as defined on page 37.

See section 5.16 on page 34 for a general description of scan sets and the effect of the *axis*, *direction*, *smode*, and *sbit* operands.

This operation determines the ordering necessary to sort the *source* fields within each scan set. It does not not actually move the data so as to sort it, but merely indicates where the data should be moved so as to sort it.

In more detail: The *dest* field in each selected processor receives, as an unsigned integer, the rank of that processor's key within the set of keys in the scan set for that processor. The smallest key has rank 0, the next smallest has rank 1, and so on; the largest key has rank $n-1$ where n is the number of processors in the scan set. This rank may be used to calculate a send address a CM:send operation may then be used to put the data into sorted order. (An advantage of decoupling the rank determination from the reordering process is that the data to be moved may be much larger than the key that determines the ordering, and indeed it may be desirable to reorder the other data but not the key itself. In this way ranking and reordering each need operate only on the relevant data.)

S-RANK

The destination field in every selected processor receives the rank of that processor's key among all keys in the scan set for that processor.

Formats CM:s-rank-2L *dest*, *source*, *axis*, *dlen*, *slen*, *direction*, *smode*, *sbit*

Operands

dest	The unsigned integer destination field.
source	The signed integer source field. This is the sort key.
axis	An unsigned integer immediate operand to be used as the number of a NEWS axis.
dlen	The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This must be no larger than the value returned by CM:geometry-coordinate-length.
slen	The length of the *source* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.
direction	Either :upward or :downward.
smode	Either :none, :start-bit, or :segment-bit.
sbit	The segment bit or start bit (a one-bit field). If *smode* is :none then this may be CM:*no-field*.

Overlap The fields *source* and *sbit* may overlap in any manner. However, the *source* and *sbit* fields must not overlap the *dest* field.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if $\textit{context-flag}[k] = 1$ then
  let $g = \textit{geometry}(\textit{current-vp-set})$
  let $S_k = \textit{scan-set}(g, k, \textit{axis}, \textit{direction}, \textit{smode}, \textit{sbit})$
  case *direction* of
   :upward:
    let $L_k = \{\, m \mid m \in S_k \wedge ((\textit{source}[m] < \textit{source}[k]) \vee (\textit{source}[m] = \textit{source}[k]$
   :downward:
    let $L_k = \{\, m \mid m \in S_k \wedge ((\textit{source}[m] > \textit{source}[k]) \vee (\textit{source}[m] = \textit{source}[k]$
   $\textit{dest}[k] \leftarrow |L_k|$

where *scan-set* is as defined on page 37.

See section 5.16 on page 34 for a general description of scan sets and the effect of the *axis*, *direction*, *smode*, and *sbit* operands.

This operation determines the ordering necessary to sort the *source* fields within each scan set. It does not not actually move the data so as to sort it, but merely indicates where the data should be moved so as to sort it.

In more detail: The *dest* field in each selected processor receives, as an unsigned integer, the rank of that processor's key within the set of keys in the scan set for that processor. The smallest key has rank 0, the next smallest has rank 1, and so on; the largest key has rank $n-1$ where n is the number of processors in the scan set. This rank may be used to calculate a send address a CM:send operation may then be used to put the data into sorted order. (An advantage of decoupling the rank determination from the reordering process is that the data to be moved may be much larger than the key that determines the ordering, and indeed it may be desirable to reorder the other data but not the key itself. In this way ranking and reordering each need operate only on the relevant data.)

U-RANK

The destination field in every selected processor receives the rank of that processor's key among all keys in the scan set for that processor.

Formats CM:u-rank-2L *dest*, *source*, *axis*, *dlen*, *slen*, *direction*, *smode*, *sbit*

Operands

dest The unsigned integer destination field.

source The unsigned integer source field. This is the sort key.

axis An unsigned integer immediate operand to be used as the number of a NEWS axis.

dlen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*. This must be no larger than the value returned by CM:geometry-coordinate-length.

slen The length of the *source* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

direction Either :upward or :downward.

smode Either :none, :start-bit, or :segment-bit.

sbit The segment bit or start bit (a one-bit field). If *smode* is :none then this may be CM:*no-field*.

Overlap The fields *source* and *sbit* may overlap in any manner. However, the *sbit* field must not overlap the *dest* field, and the field *source* must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    let $S_k = scan\text{-}set(g, k, axis, direction, smode, sbit)$
    case *direction* of
      :upward:
        let $L_k = \{\, m \mid m \in S_k \wedge ((source[m] < source[k]) \vee (source[m] = source[k]$ /
      :downward:
        let $L_k = \{\, m \mid m \in S_k \wedge ((source[m] > source[k]) \vee (source[m] = source[k]$ /
      $dest[k] \leftarrow |L_k|$

where *scan-set* is as defined on page 37.

See section 5.16 on page 34 for a general description of scan sets and the effect of the *axis*, *direction*, *smode*, and *sbit* operands.

This operation determines the ordering necessary to sort the *source* fields within each scan set. It does not not actually move the data so as to sort it, but merely indicates where the data should be moved so as to sort it.

In more detail: The *dest* field in each selected processor receives, as an unsigned integer, the rank of that processor's key within the set of keys in the scan set for that processor. The smallest key has rank 0, the next smallest has rank 1, and so on; the largest key has rank $n-1$ where n is the number of processors in the scan set. This rank may be used to calculate a send address a CM:send operation may then be used to put the data into sorted order. (An advantage of decoupling the rank determination from the reordering process is that the data to be moved may be much larger than the key that determines the ordering, and indeed it may be desirable to reorder the other data but not the key itself. In this way ranking and reordering each need operate only on the relevant data.)

S-F-ROUND

Converts floating-point source field values to signed integer values by rounding to the nearest integer.

Formats CM:s-f-round-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest The signed integer destination field.

source The floating-point source field.

len The length of the *dest* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *dest* and *source* must not overlap in any manner.

Flags *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    let $v = source[k]$
    if $v > \left\lfloor v + \frac{1}{2} \right\rfloor$ then
      $dest[k] \leftarrow \lfloor v \rfloor$
    else if $v < \left\lfloor v + \frac{1}{2} \right\rfloor$ then
      $dest[k] \leftarrow \lceil v \rceil$
    else if $even(\lfloor v \rfloor)$ then
      $dest[k] \leftarrow \lfloor v \rfloor$
    else
      $dest[k] \leftarrow \lceil v \rceil$
    if $\langle$overflow occurred in processor $k\rangle$ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer (to the nearest even integer if its value is equal to an integer plus $\frac{1}{2}$). The result is stored into the *dest* field as a signed integer.

U-ROUND

The quotient of two unsigned integer source values, rounded to the nearest integer, is placed in the destination field. Overflow is also computed.

Formats CM:s-round-3-3L *dest*, *source1*, *source2*, *dlen*, *slen1*, *slen2*

Operands

dest The unsigned integer quotient field.

source1 The unsigned integer dividend field.

source2 The unsigned integer divisor field.

dlen The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen1 The length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen2 The length of the *source2* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. Each of them, however, must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if the quotient cannot be represented in the destination field; otherwise it is cleared.

test-flag is set if the divisor is zero; otherwise it is cleared.

Context This operation is conditional. The destination and flags may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if $\textit{context-flag}[k] = 1$ then
  let $v = \dfrac{\textit{source1}[k]}{\textit{source2}[k]}$
  if $v > \left\lfloor v + \frac{1}{2} \right\rfloor$ then
   $\textit{dest}[k] \leftarrow \lfloor v \rfloor$
  else if $v < \left\lfloor v + \frac{1}{2} \right\rfloor$ then
   $\textit{dest}[k] \leftarrow \lceil v \rceil$
  else if $\textit{even}(\lfloor v \rfloor)$ then
   $\textit{dest}[k] \leftarrow \lfloor v \rfloor$

else
  $dest[k] \leftarrow \lceil v \rceil$
if $\langle$overflow occurred in processor $k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$

The unsigned integer *source1* operand is divided by the unsigned integer *source2* operand. The mathematical quotient, rounded to the nearest integer (or to whichever of two equally near neighbors is even) is stored into the unsigned integer memory field *dest*.

The *overflow-flag* and *test-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

U-F-ROUND

Converts the floating-point source field values to unsigned integer values, which are stored in the destination field.

Formats CM:u-f-round-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest The unsigned integer destination field.

source The floating-point source field.

len The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

s, *e* The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *dest* and *source* must not overlap in any manner.

Flags *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if *context-flag*$[k] = 1$ then
    if *dest* $> \lfloor source \rfloor$ then
      dest $\leftarrow \lfloor source \rfloor$
    else if *dest* $< \lfloor source \rfloor$ then
      dest $\leftarrow \lceil source \rceil$
    else if *even*$(\lfloor source \rfloor)$ then
      dest $\leftarrow \lfloor source \rfloor$
    else
      dest $\leftarrow \lceil source \rceil$
    if $\langle$overflow occurred in processor $k\rangle$ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer (to the nearest even integer if its value is equal to an integer plus $\frac{1}{2}$), which is stored into the *dest* field as an unsigned integer.

RESET-TIMER

For the C/Paris and Fortran/Paris interfaces, resets the timing facility before timing other operations.

Formats CM:reset-timer

Context This operation is unconditional. It does not depend on the *context-flag.*

The function CM:reset-timer is used in the C/Paris and Fortran/Paris interfaces to reset the facility for timing the execution of other operations on the Connection Machine system.

One should first call CM:reset-timer to clear the timing counters. Subsequently one may alternately call CM:start-timer and CM:stop-timer. The amounts of real time and run time between a start and a stop are accumulated into the counters. One may start and stop the clocks repeatedly. Every time CM:stop-timer is called, it returns a structure of type CM_timeval_t that contains time accumulated between all start/stop call pairs since the last call to CM:reset-timer.

The timing facility is provided in the Lisp/Paris interfaces through the CM:time macro.

F-S-SCALE

In each selected processor, multiplies a floating-point number by a specified power of two and stores the result into the destination.

Formats

CM:f-s-scale-2-2L	*dest/source1*, *source2*, *slen2*, *s*, *e*
CM:f-s-scale-3-2L	*dest*, *source1*, *source2*, *slen2*, *s*, *e*
CM:f-s-scale-constant-2-1L	*dest/source1*, *source2-value*, *s*, *e*
CM:f-s-scale-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*

Operands

dest The floating-point destination field.

source1 The floating-point first source field. This is the quantity to be scaled.

source2 The signed integer second source field. This is the base-2 logarithm of the scale factor.

source2-value A signed integer immediate operand to be used as the second source.

s, *e* The significand and exponent lengths for the *dest* and *source1* fields. The total length of an operand in this format is $s + e + 1$.

slen2 The length of the *source2* field. This must be no smaller than 2 but no greater than CM:*maximum-integer-length*.

Overlap The fields *source1* and *source2* may overlap in any manner. However, the *source2* field must not overlap the *dest* field, and the field *source1* must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Flags *overflow-flag* is set if floating-point overflow occurs; otherwise it is unaffected.

Context This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
    if *context-flag*$[k] = 1$ then
        $dest[k] \leftarrow \left\lfloor source1[k] \times 2^{source2[k]} \right\rfloor$
        if $\langle$overflow occurred in processor $k\rangle$ then *overflow-flag*$[k] \leftarrow 1$

The operand *source1* is scaled by the power of two specified by *source2*.

The result is stored into the memory field *dest*. The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand.

SEND-ASET32-U-ADD

Sends a message from every selected processor to a specified destination processor and stores it there, as if by aset32, in an array. Each selected processor may specify any processor as the destination, including itself. A destination processor may receive messages even if it is not selected. All incoming messages are combined with the destination array element using unsigned integer addition.

Formats CM:send-aset32-u-add-2L *array*, *send-address*, *source*, *index*, *slen*, *index-len*, *index-limit*

Operands

array The destination array field.

send-address The send-address field. For each processor, this indicates to which processor a message is sent.

source The source field.

index The unsigned integer index into the array field. This is used as a per-processor index into *array*. It specifies portions of the *array* memory area in increments of *slen*.

slen The length of the *source* field. This must be a multiple of 32.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of the destination array.

Overlap The *send-address* and *source* may overlap in any manner. The *dest* field may overlap with the *send-address* or *source* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *source* only if at most one of them will be used within each processor.

Context This operation is conditional, but whether a message is sent depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is combined with the the field indicated by *array* regardless of the *context-flag* of the receiving processor.

Definition For every virtual processor k in the *current-vp-set* do

let $S_k = \{ m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \wedge \textit{send-address}[m] = k \}$

for every processor k' in S_k do

if $index[k'] < index\text{-}limit$ then
  let $r = geometry\text{-}total\text{-}vp\text{-}ratio(geometry(current\text{-}vp\text{-}set))$
  let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
  let $i = index[k']$
  for all j such that $0 \le j < dlen$ do
    let $temp_k\langle j \rangle = array[k - m \times r + (j \bmod 32) \times r]\langle 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor) \rangle$
  let $sum_k = temp_k + source[k']$
  for all j such that $0 \le j < dlen$ do
    $array[k - m \times r + (j \bmod 32) \times r]\langle 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor) \rangle \leftarrow sum_k\langle j \rangle$
else
  ⟨error⟩

For every selected processor p_s, a message *length* bits long is sent from that processor to the processor p_d whose send address is stored at location *send-address* in the memory of processor p_s. The message is taken from the *source* field within processor p_s and is stored into an array element within processor p_d. Note that in each case the array element to be modified in processor p_d is determined by the value of *index* within p_s, not the value within p_d.

The CM:send-aset32-u-add operation combines incoming messages with unsigned integer addition. To receive the sum of only the messages, the destination *array* should first be cleared in all processors that might receive a message.

SEND-ASET32-LOGIOR

Sends a message from every selected processor to a specified destination processor and stores it there, as if by **aset32**, in an array. Each selected processor may specify any processor as the destination, including itself. A destination processor may receive messages even if it is not selected. All incoming messages are combined with the destination array element using bitwise logical inclusive OR.

Formats CM:send-aset32-logior-2L *array*, *send-address*, *source*, *index*, *slen*, *index-len*, *index-limit*

Operands

array The destination array field.

send-address The send-address field. For each processor, this indicates to which processor a message is sent.

source The source field.

index The unsigned integer index into the array field. This is used as a per-processor index into *array*. It specifies portions of the *array* memory area in increments of *slen*.

slen The length of the *source* field. This must be a multiple of 32.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of the destination array.

Overlap The *send-address* and *source* may overlap in any manner. The *dest* field may overlap with the *send-address* or *source* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *source* only if at most one of them will be used within each processor.

Context This operation is conditional, but whether a message is sent depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is combined with the the field indicated by *array* regardless of the *context-flag* of the receiving processor.

Definition For every virtual processor k in the *current-vp-set* do

let $S_k = \{ m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \wedge \textit{send-address}[m] = k \}$

for every processor k' in S_k do

if $index[k'] < index\text{-}limit$ then
  let $r = geometry\text{-}total\text{-}vp\text{-}ratio(geometry(current\text{-}vp\text{-}set))$
  let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
  let $i = index[k']$
  for all j such that $0 \le j < dlen$ do
    let $q = k - m \times r + (j \bmod 32) \times r$
    let $b = 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor)$
    $array[q]\langle b\rangle \leftarrow array[q]\langle b\rangle \vee source[k']\langle j\rangle$
else
  $\langle$error$\rangle$

For every selected processor p_s, a message *length* bits long is sent from that processor to the processor p_d whose send address is stored at location *send-address* in the memory of processor p_s. The message is taken from the *source* field within processor p_s and is stored into an array element within processor p_d. Note that in each case the array element to be modified in processor p_d is determined by the value of *index* within p_s, not the value within p_d.

The CM:send-aset32-logior operation combines incoming messages with a bitwise logical inclusive OR operation. To receive the logical inclusive OR of only the messages, the destination *array* should first be cleared in all processors that might receive a message.

SEND-ASET32-OVERWRITE

Sends a message from every selected processor to a specified destination processor and stores it there, as if by aset32, in an array. Each selected processor may specify any processor as the destination, including itself. A destination processor may receive messages even if it is not selected. If a processor receives more than one message destinated for the same array element, then one is stored in that array element and the rest are discarded.

Formats

CM:send-aset32-overwrite-2L *array*, *send-address*, *source*, *index*, *slen*, *index-len*, *index-limit*

Operands

array The destination array field.

send-address The send-address field. For each processor, this indicates to which processor a message is sent.

source The source field.

index The unsigned integer index into the array field. This is used as a per-processor index into *array*. It specifies portions of the *array* memory area in increments of *slen*.

slen The length of the *source* field. This must be a multiple of 32.

index-len The length of the *index* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

index-limit An unsigned integer immediate operand to be used as the exclusive upper bound for the *index*. This is taken as the extent of the destination array.

Overlap The *send-address* and *source* may overlap in any manner. The *dest* field may overlap with the *send-address* or *source* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *source* only if at most one of them will be used within each processor.

Context This operation is conditional, but whether a message is sent depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is combined with the the field indicated by *array* regardless of the *context-flag* of the receiving processor.

Definition

For every virtual processor k in the *current-vp-set* do

let $S_k = \{\, m \mid m \in \textit{current-vp-set} \wedge \textit{context-flag}[m] = 1 \wedge \textit{send-address}[m] = k \,\}$

let $k' = \textit{choice}(S_k)$

if *index*$[k'] <$ *index-limit* then
  let $r =$ *geometry-total-vp-ratio*(*geometry*(*current-vp-set*))
  let $m = \left\lfloor \frac{k}{r} \right\rfloor \bmod 32$
  let $i =$ *index*$[k']$
  for all j such that $0 \leq j <$ *dlen* do
    $array[k - m \times r + (j \bmod 32) \times r]\langle 32 \times (i + \left\lfloor \frac{j}{32} \right\rfloor)\rangle \leftarrow source[k']\langle j \rangle$
else
  ⟨error⟩

For every selected processor p_s, a message *length* bits long is sent from that processor to the processor p_d whose send address is stored at location *send-address* in the memory of processor p_s. The message is taken from the *source* field within processor p_s and is stored into an array element within processor p_d. Note that in each case the array element to be modified in processor p_d is determined by the value of *index* within p_s, not the value within p_d.

The CM:**send-aset32-overwrite** operation will store one of the messages sent to a particular array element, discarding all other messages as well as the original contents of that array element in the receiving processor.

SEND-TO-NEWS

Each processor sends a message to a neighboring processor along a specified NEWS axis.

Formats CM:send-to-news-1L *dest*, *source*, *axis*, *direction*, *len*
CM:send-to-news-always-1L *dest*, *source*, *axis*, *direction*, *len*

Operands

dest The destination field.

source The source field.

axis An unsigned integer immediate operand to be used as the number of a NEWS axis.

direction Either :upward or :downward.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two bit fields are identical if they have the same address and the same length.

Context This operation is conditional, but whether data is copied depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is stored into the the field indicated by *dest* regardless of the *context-flag* of the receiving processor.

Note that in the conditional case the storing of data depends only on the *context-flag* of the processor sending the data, not on the *context-flag* of the processor receiving the data.

Definition For every virtual processor k in the *current-vp-set* do
  if (always or $\textit{context-flag}[k] = 1$) then
    let $g = \textit{geometry}(\textit{current-vp-set})$
    $\textit{dest}[\textit{news-neighbor}(g, k, \textit{axis}, \textit{direction})] \leftarrow \textit{source}[k]$

The *source* field in each processor is stored into the *dest* field of that processor's neighbor along the NEWS axis specified by *axis* in the direction specified by *direction*.

If *direction* is :upward then each processor stores data into the neighbor whose NEWS coordinate is one greater, with the processor whose coordinate is greatest storing data into the processor whose coordinate is zero.

If *direction* is :downward then each processor stores data into the neighbor whose NEWS coordinate is one less, with the processor whose coordinate is zero storing data into the processor whose coordinate is greatest.

SEND-WITH-F-ADD

Sends a message from every selected processor to a specified destination processor. Each selected processor may specify any processor as the destination, including itself. A destination processor may receive messages even if it is not selected, and all the destination processors may be in a VP set different from the VP set of the source processors. Messages are all delivered to the same address within each receiving processor. All incoming messages are combined with the destination field using floating-point addition.

Formats CM:send-with-f-add-1L *dest*, *send-address*, *source*, *s*, *e*, *notify*

Operands

dest The floating-point destination field.

send-address The send-address field. For each processor, this indicates to which processor a message is sent.

source The floating-point source field.

s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

notify The notification bit (a one-bit field). This argument may be CM:*no-field* if no notification of message receipt is desired.

Overlap The *send-address* and *source* may overlap in any manner. The *dest* field may overlap with the *send-address* or *source* but, if it does, then it is forbidden to send a message to a selected processor. In other words, the *dest* may overlap with the *send-address* or *source* only if at most one of them will be used within each processor.

Context This operation is conditional, but whether a message is sent depends only on the *context-flag* of the originating processor; the message, once transmitted to the receiving processor, is combined with the *dest* field regardless of the *context-flag* of the receiving processor. The *notify* bit may be altered in any processor regardless of the value of the *context-flag*.

Definition For every virtual processor k in the *current-vp-set* do

let $S_k = \{\, m \mid m \in \textit{current-vp-set} \land \textit{context-flag}[m] = 1 \land \textit{send-address}[m] = k \,\}$

if $|S_k| = 0$ then

if $\textit{notify}[k] \not\equiv$ CM:*no-field* then $\textit{notify}[k] \leftarrow 0$

else

if $\textit{notify}[k] \not\equiv$ CM:*no-field* then $\textit{notify}[k] \leftarrow 1$

$$\textit{dest}[k] \leftarrow \textit{dest}[k] + \left(\sum_{m \in S_k} \textit{source}[m] \right)$$

STORE-flag

Conditionally stores a flag bit into memory.

Formats

CM:store-test	*dest*
CM:store-test-always	*dest*
CM:store-overflow	*dest*
CM:store-overflow-always	*dest*

Operands *dest* The destination bit (a one-bit field).

Context The non-always operations are conditional. The destination may be altered only in processors whose *context-flag* is 1.

The always operations are unconditional. The destination may be altered regardless of the value of the *context-flag*.

Definition For every virtual processor k in the *current-vp-set* do
  if $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \textit{flag}[k]$

where *flag* is *test-flag* or *overflow-flag*, as appropriate.

Within each processor, the indicated flag for that processor is stored into memory.

FE-STRUCTURE-ARRAY-FORMAT

This instruction returns an array format descriptor for a particular slot in an array of structures. A format descriptor may be passed to any array transfer instruction to specify a front-end array format, although this is not required. See also CM:fe-array-format and CM:fe-packed-array-format.

This instruction is not provided for the Lisp interface to Paris.

Formats result ← CM:fe-structure-array-format *cm-element-byte-size*, *structure-byte-size*

Operands *cm-element-byte-size* A signed integer immediate operand to be used as the number of bytes each Connection Machine element occupies in the front-end array. This must be a power of two between 1 and 16.

structure-byte-size A signed integer immediate operand to be used as the length of the front-end structure in bytes. This may be any positive integer.

Result The array format descriptor specified.

Context This is a front-end operation. It does not depend on the value of the *context-flag*.

The return value is a format descriptor for a front-end array of structures. Such a format descriptor can be passed to any of the CM array transfer instructions in order to allow transfers in either direction between CM fields and a front-end array of structures. If this is done, one CM element per selected processor is copied into, or receives data from, the specified slot across an array of structures on the front end.

Values for both *cm-element-byte-size* and *cm-structure-byte-size* may be obtained by calls to sizeof(...).

The value of *cm-element-byte-size* specifies the length of the structure slot in bytes. It also defines the unit of measure for the *fe-offset-vector* argument to the CM:read-from-news-array and CM:write-to-news-array instructions.

The value of *structure-byte-size* specifies the length of the entire stucture in bytes. It also defines the unit of measure for the argument *fe-dimension-vector* to the CM:read-from-news-array and CM:write-to-news-array instructions.

If a slot other than the first slot in the front-end structure is the destination of a CM:read-from-news-array or the source for a CM:write-to-news-array transfer instruction, then a pointer to that slot must be provided as the value of *front-end-array*. This is a bit tricky. The

U-TO-GRAY-CODE

Converts an unsigned binary integer to a bit string representing a Gray-coded integer value.

Formats CM:u-to-gray-code-1-1L *dest/source*, *len*
CM:u-to-gray-code-2-1L *dest*, *source*, *len*

Operands

dest The destination field.

source The unsigned integer source field.

len The length of the *dest* and *source* fields. This must be non-negative and no greater than CM:*maximum-integer-length*.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two integer fields are identical if they have the same address and the same length.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
  if $context\text{-}flag[k] = 1$ then
    $dest[k]\langle len-1\rangle \leftarrow source[k]\langle len-1\rangle$
    for j from $len-2$ to 0 do
      $dest[k]\langle j\rangle \leftarrow source[k]\langle j\rangle \oplus source[k]\langle j+1\rangle$

The *source* operand is an unsigned binary integer, and is converted to a bit-string value in a particular reflected binary Gray code. The position of that value in the standard Gray code sequence is the *source*.

Note that the binary value 0 is always equivalent to a Gray code string that is all 0-bits.

TRANSPOSE32

Within each cluster of 32 physical processors, for every group of 32 virtual processors in such a cluster, copies one 32-bit field to another. During this copying operation, transposes the data as a 32-by-32 bit matrix. Thus, each virtual processor receives one bit from the source value of each virtual processor in its group of 32.

Formats CM:transpose32-1-1L *dest/source*, *len*
CM:transpose32-2-1L *dest*, *source*, *len*

Operands

source The source field.

dest The destination field.

len The length of the *source* and *dest* fields. This must be non-negative and no greater than CM:*maximum-integer-length*. This must be a multiple of 32.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two bit fields are identical if they have the same address and the same length. The fields *dest* and *source* may overlap in any manner.

Context This operation is unconditional. The destination may be altered regardless of the value of the *context-flag*.

Definition For every virtual processor k in the *current-vp-set* do
if $\textit{context-flag}[k] = 1$ then
for all j such that $0 \le j < \textit{dlen}$ do

$$\textit{dest}[k]\langle j\rangle \leftarrow \textit{source}\left[32r\left\lfloor\frac{k}{32r}\right\rfloor + (k \bmod r) + r(j \bmod 32)\right]\left\langle 32\left\lfloor\frac{j}{32}\right\rfloor + \frac{k \bmod 32}{r}\right\rangle$$

where r is the value of CM:*virtual-to-physical-processor-ratio* and j is the bit position in each field.

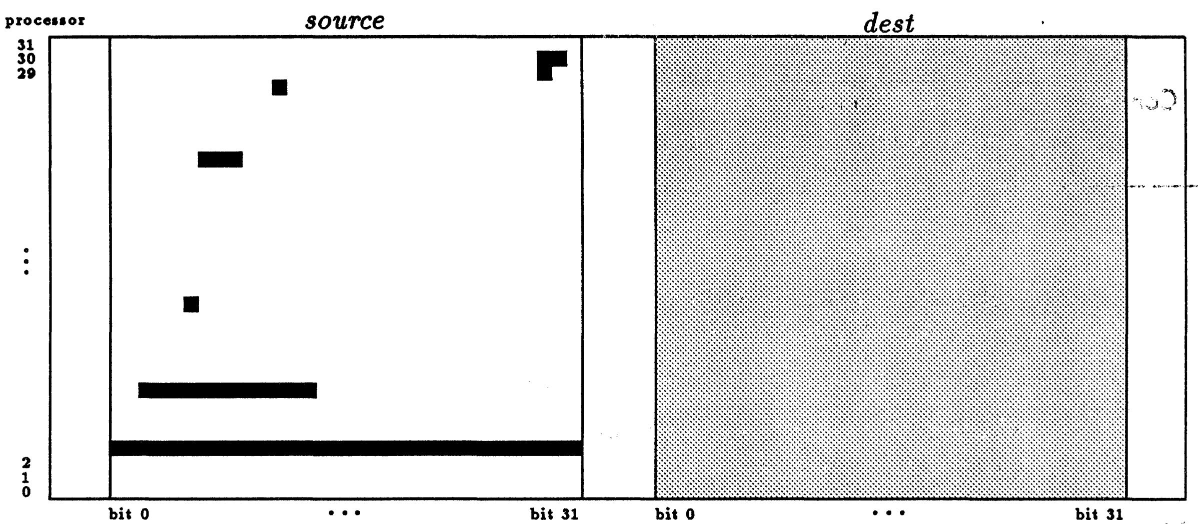
This instruction copies each 32-bit field to the corresponding 32-bit field within each virtual processor. In the course of copying the bits, it "transposes" them so that a 32-bit value lying entirely within the *source* field of one virtual processor is made to occupy a memory slice, that is, one bit in each of 32 virtual processors. The opposite is also true: the 32-bit value that ends up in the *dest* field of a virtual processor is made up of one bit from each of 32 virtual processors. Transposed data is said to be stored in a *slicewise* format.

For the purposes of this instruction, the physical processors are divided into clusters of 32. Two processors are in the same cluster if their physical processor numbers agree in all but the five least significant bits.

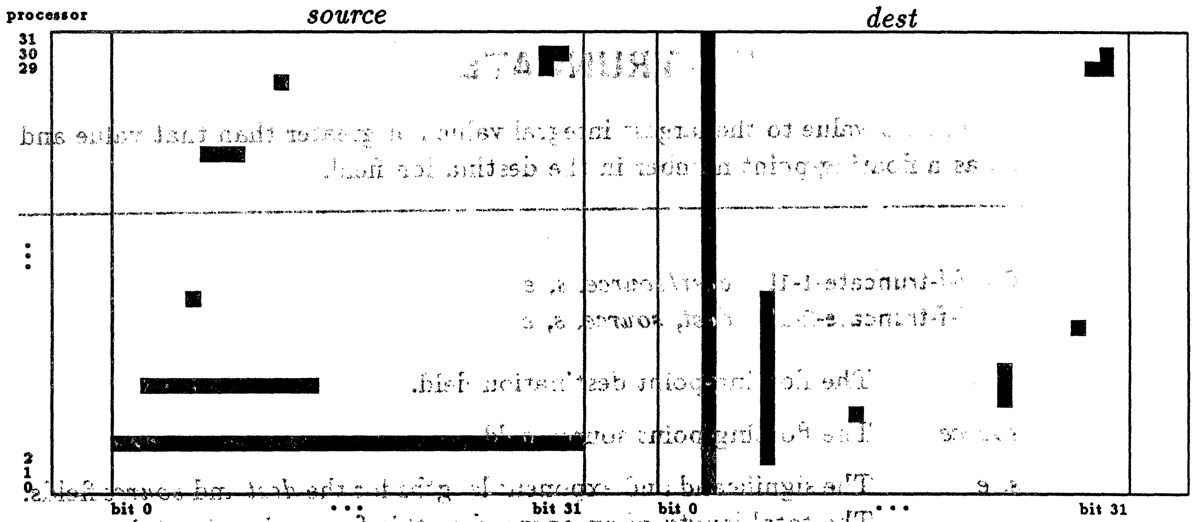
The virtual processors are similarly divided into groups of 32; a group of virtual processors consists of one virtual processor from each physical processor of a cluster, such that the virtual processors occupy the same physical memory locations within their respective physical processors. Thus, two virtual processors are in the same group if their virtual processor numbers agree in all but bit positions n through $n + 4$, where n is the number of virtual processors bits in each physical processor.

The CM: **transpose32** operation may then be understood as taking the 32 32-bit *source* values from a group of 32 virtual processors as the rows of a 32-by-32 bit matrix, and then storing the columns of this matrix into the *dest* fields of these same virtual processors.

The process may be understood pictorially. Suppose that before the operation the memory of a group of 32 virtual processors looks like this:



Then, after the CM: **transpose-data** operation, it will look like this:



Knowledge of the internal details of Connection Machine VP memory layout is required to use this instruction properly on *source* values represented in more than 32-bits.

This instruction reorients processor data into a slicewise format that permits rapid, indirect field addressing. A memory region containing transposed data may be viewed either as a single, *shared slicewise array* or as a set of *parallel slicewise arrays*. (See the CM:aref32 and CM:aref32-shared dictionary entries for a description of these data formats.) Viewed as a shared slicewise array, this is especially useful for quickly constructing lookup tables.

Transposition is reversed by applying the CM:transpose32 instruction to a field already stored in the slicewise format. To preserve the correlation between processors and data, this instruction should not be used on slicewise data that was orginally stored by providing CM:aset32 or CM:aset32-shared with an *index-limit* other than 32.

F-F-TRUNCATE

Rounds each source field value to the largest integral value not greater than that value and stores the result as a floating-point number in the destination field.

Formats CM:f-f-truncate-1-1L *dest*/*source*, *s*, *e*
CM:f-f-truncate-2-1L *dest*, *source*, *s*, *e*

Operands
dest The floating-point destination field.
source The floating-point source field.
s, *e* The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two floating-point fields are identical if they have the same address and the same format.

Context This operation is conditional. The destination may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor *k* in the *current-vp-set* do
if *context-flag*[*k*] = 1 then
$dest[k] \leftarrow sign(source) \times \lfloor |source[k]| \rfloor$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of zero, which is stored into the *dest* field as a floating-point number.

if *source2*$[k] = 0$ then
  $dest[k] \leftarrow$ ⟨unpredictable⟩
else
  $dest[k] \leftarrow \left\lfloor \frac{source1[k]}{source2[k]} \right\rfloor$
if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$
else *overflow-flag*$[k] \leftarrow 0$

The unsigned integer *source1* operand is divided by the unsigned integer *source2* operand. The floor of the mathematical quotient is stored into the unsigned integer memory field *dest*. The various operand formats allow operands to be either memory fields are constants; in some cases the destination field initially contains one source operand.

The *overflow-flag* may be affected by these operations. If overflow occurs, then the destination field will contain as many of the low-order bits of the true result as will fit.

The constant operand *source2-value* should be a signed integer front-end value. The operation is performed properly in all cases; the constant need not be representable in the number of bits specified by *len*.

U-F-TRUNCATE

Rounds each source field value to the largest integer not greater than that value and stores the result as an unsigned integer in the destination field.

Formats CM:u-f-truncate-2-2L *dest*, *source*, *dlen*, *s*, *e*

Operands

dest	The unsigned integer destination field.
source	The floating-point source field.
len	The length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.
s, *e*	The significand and exponent lengths for the *source* field. The total length of an operand in this format is $s + e + 1$.

Overlap: The fields *dest* and *source* must not overlap in any manner.

Flags: *overflow-flag* is set if the result cannot be represented in the *dest* field; otherwise it is cleared.

Context: This operation is conditional. The destination and flag may be altered only in processors whose *context-flag* is 1.

Definition For every virtual processor k in the *current-vp-set* do
 if *context-flag*$[k] = 1$ then
  $dest \leftarrow sign(source) \times \lfloor |source| \rfloor$
  if ⟨overflow occurred in processor k⟩ then *overflow-flag*$[k] \leftarrow 1$

The *source* field, treated as a floating-point number, is rounded to the nearest integer in the direction of zero, and the result is stored into the *dest* field as an unsigned integer.