The
Connection Machine
System

Paris Reference Manual Supplement

Version 5.1
June 1989

Thinking Machines Corporation
Cambridge, Massachusetts

First printing, June 1989

The information in this document is subject to change without notice and should not be construed as a commitment by Thinking Machines Corporation. Thinking Machines Corporation reserves the right to make changes to any products described herein to improve functioning or design. Although the information in this document has been reviewed and is believed to be reliable, Thinking Machines Corporation does not assume responsibility or liability for any errors that may appear in this document. Thinking Machines Corporation does not assume any liability arising from the application or use of any information or product described herein.

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

Contents

Supplement Dictionary

Overview of 5.1 Instructions

About This Supplement

This supplement includes two sections, a conceptual overview and a dictionary. The first section introduces the new Paris instructions for Version 5.1. It is organized by groups of functionally related features. The second section contains reference documentation for the 5.1 features, organized alphabetically. In a future printing, this supplement will be merged with the *Paris Reference Manual.*

Features New with Paris 5.1

The following categories of features are new with Paris, Version 5.1.

Complex Numbers

Most Paris operations previously available for integers and floating-point numbers now are supported for complex operands. Complex numbers are defined to have real and imaginary parts, each represented as a floating-point number.

More formally, a complex floating-point data item is specified by three parameters exactly like those for a floating-point data item: a bit address a, a significand length s, and an exponent length e. The data item consists of two consecutive floating-point data items, with the real part at address a and the imaginary part at address $a+s+e+1$. The total number of bits in the representation is $2(s+e+1)$, and the data item occupies the bits with addresses a through $a+2(s+e)+1$, inclusive.

The prefix **c-** designates instructions that take one or more complex operands.

CM_complex_t is the type of a complex immediate operand in C.

Complex Unary Arithmetic Operations

Paris Version 5.1 includes the following unary operations on complex operands:

```
CM:f-c-abs-2-1L
CM:c-abs-{1, 2}-1L
CM:c-negate-{1, 2}-1L
CM:c-exp-{1, 2}-1L
CM:c-ln-{1, 2}-1L
CM:c-sqrt-{1, 2}-1L
CM:c-c-signum-{1, 2}-1L
CM:c-conjugate-{1, 2}-1L
CM:c-reciprocal-{1, 2}-1L
CM:c-f-cis-2-1L
CM:f-c-phase-2-1L
```

A full set of transcendental and trigonometric functions on complex numbers are also now available.

```
CM:c-{a,-}sin-{1, 2}-1L
CM:c-{a,-}cos-{1, 2}-1L
CM:c-{a,-}tan-{1, 2}-1L
CM:c-{a,-}sinh-{1, 2}-1L
CM:c-{a,-}cosh-{1, 2}-1L
CM:c-{a,-}tanh-{1, 2}-1L
```

Complex Binary Arithmetic Operations

Paris Version 5.1 includes complex versions for most binary operations. Basic addition, subtraction, multiplication, and division are provided.

```
CM:c-add{-, -constant, -always, -const-always}-{2,3}-1L
CM:c-subtract{-, -constant, -always, -const-always}-{2,3}-1L
CM:c-multiply{-, -constant, -always, -const-always}-{2,3}-1L
CM:c-divide{-, -constant, -always, -const-always}-{2,3}-1L
```

A complete set of complex exponentiation operations is also included.

```
CM:c-{c, f, s, u}-power{-, -constant}-{2, 3}-1L
```

Special cases of complex reverse subtraction and reverse division are supported.

```
CM:c-divinto{-, -always}-2-1L
CM:c-divinto-{constant, const-always}-{2, 3}1L
CM:c-subfrom{-, -always}-2-1L
CM:c-subfrom-{constant, const-always}-{2, 3}1L
```

Complex Arithmetic Comparisons

The two essential comparison operations are available for complex numbers: equal and not-equal.

```
CM:c-eq{-, -constant, -zero}-1L
CM:c-ne{-, -constant, -zero}-1L
```

Complex Move and Read/Write Processor

Copying complex data between CM fields is supported by these instructions:

```
CM:c-move-2L
CM:c-move{-, -constant, -always, zero}-1L
CM:c-move-{const,  zero}-{ always}-1L
```

Transferring data between the CM and the front end is accomplished with the following instructions:

```
CM:c-read-from-processor-1L
CM:c-write-to-processor-1L
```

Complex NEWS Communication

General communication and communication with computation is provided for complex numbers by the following instructions:

```
CM:multispread-c-add-1L
CM:reduce-with-c-add-1L
CM:scan-with-c-add-1L
CM:send-with-c-add-1L
CM:spread-with-c-add-1L
```

Global reduction is performed on complex numbers by the following instruction:

```
CM:global-c-add-1L
```

Field Aliasing

A field alias is a field-id that references a field already referenced by at least one other field-id. By using field aliases, it is possible to reference the same CM memory field from within different VP sets.

These are the operations that create, destroy, and manipulate field aliases:

```
CM:change-field-alias
CM:is-field-an-alias
CM:make-field-alias
CM:remove-field-alias
CM:set-field-alias-vp-set
```

Power of Two NEWS

A new instruction, with both conditional and unconditional versions, performs near-neighbor communication between processors that are separated by a particular distance. That distance must be a power of two, measured in intervening processors and inclusive of the source processor. Instructions of the following form support power of two NEWS communication:

```
CM:get-from-power-two{-, -always}-1L
```

Floating-Point Conversion

It is now easy to convert floating-point numbers between the IEEE format used in the Connection Machine system and VAX floating-point format. The following new instructions provide this capability:

CM:f-ieee-to-vax-1L
CM:f-vax-to-ieee-1L

NEWS With Floating-Point Combiners

Paris Version 5.1 introduces instructions that calculate a special form of binary addition, subtraction, and multiplication in which one operand is retrieved from a NEWS neighbor of the destination field.

CM:f-news-add{-, -always}-{2, 3}-1L
CM:f-news-add-const{-, -a}-3-1L
CM:f-news-add{-, -const}-mult-4-1L

CM:f-news-sub{-, -always}-{2, 3}-1L
CM:f-news-sub-const{-, -a}-3-1L
CM:f-news-sub{-, -const}-mult-4-1L

CM:f-news-mult{-, -always}-{2, 3}-1L
CM:f-news-mult-const{-, -a}-4-1L
CM:f-news-mult{-, -const}-add-4-1L
CM:f-news-mult{-, -const}-sub-4-1L

Floating-Point Multiplication and Reverse Subtraction Combined

A new set of instructions combine floating-point multiplication with reverse subtractions in a variety of ways.

CM:f-mult-subf{-, -const}-1L
CM:f-mult-const-subf{-, -const}-1L
CM:f-subf-const-mult{-, -const}-1L

Floating-Point Modulo Division and Rounding

Floating-point modulo division and rounding instructions are added to Paris with Version 5.1.

```
CM:f-mod{~, -constant}-{2, 3}-1L
CM:f-f-round-{1, 2}-1L
```

Floating-Point Exponentials and Logarithms

```
CM:f-exp2-{1, 2}-1L
CM:f-log2-{1, 2}-1L
CM:f-log10-{1, 2}-1L
```

Integer Exponentiation

In Version 5.0, integer exponentiation was supported only by instructions of this form:

```
CM:s-s-power{~, -constant}-{2, 3} -1L
```

In Version 5.1, Paris supports a complete suite of integer exponentiation instructions:

```
CM:{s, u}-{s, u}-power-3-3L
CM:{s, u}-{s, u}-power-constant-2-1L
CM:{s, u}-{s, u}-power-constant-3-{1, 2}L
```

Moves Across VP Sets

Now it is possible to move data between VP sets. A new instruction allows copying all or a portion of one multi-dimensional block of data from the current VP set into a similarly shaped region in another VP set. Moves across VP sets is supported by the following instruction:

```
CM:cross-vp-move-1L
```

Heap Compression

After turning automatic heap compression off, programmers can control heap compression explicitly with this new operation:

CM:compress-heap

Interned Geometries and Vp Sets

Paris 5.1 supports a special class of geometry and VP set objects: *interned* objects. Interned objects are created with instructions whose names begin with **CM:intern**. Unlike a geometry or VP set created by one of the operations with **CM:create** in its name, an interned geometry or VP set may be be accessed simply by describing it—the id need not be known. This interning facility is especially useful to compiler writers. Also, interning can render application programs more readable by allowing data created within the same VP set to use different names for the VP set id.

The instructions that return interned geometries and VP sets are:

CM:intern-geometry
CM:intern-detailed-geometry
CM:intern-identical-vp-set

Notice that a geometry or VP set may either be interned or uninterned and remains one or the other throughout the duration of its existence. For instance, a geometry created with **CM:create-geometry** may not subsequently be interned with a call to **CM:intern-geometry**; the **CM:intern-** and **CM:create-** instructions result in substantively different kinds of objects.

Supplement Dictionary

F-C-ABS

The absolute value of the source field is returned in the destination field.

Formats CM:f-c-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 the *dest* field in this format is $s + e + 1$. The total length of the *source* field in this format is $2(s + e + 1)$.

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

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
    $dest[k] \leftarrow \sqrt{(source[k].real)^2 + (source[k].imag)^2}$
    if ⟨overflow occurred in processor k⟩ then overflow-flag[k] ← 1
```

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

C-ACOS

Computes, in each selected processor, the arc cosine of the complex source field and stores it in the complex destination field.

Formats CM:c-acos-1-1L *dest/source*, *s*, *e*
CM:c-acos-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \cos^{-1} \textit{source}[k]$
    if $\langle$overflow occurred in processor $k\rangle$ then $\textit{overflow-flag}[k] \leftarrow 1$
```

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

The following definition of arc cosine determines the range and branch cuts for a complex number *z*.

$$-i \log \left(z + i\sqrt{1 - z^2}\right)$$

C-ACOSH

Computes, in each selected processor, the arc hyperbolic cosine of the complex source field and stores it in the complex destination field.

Formats CM:c-acosh-1-1L *dest/source*, *s*, *e*

CM:c-acosh-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \cosh^{-1} \textit{source}[k]$
    if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
```

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

The following definition of inverse hyperbolic cosine determines the range and branch cuts of a complex number *z*.

$$\log\left(z + (z+1)\sqrt{\frac{(z-1)}{(z+1)}}\right)$$

C-ADD

The sum of two complex source values is placed in the destination field.

Formats

CM:c-add-2-1L	*dest/source1*, *source2*, *s*, *e*
CM:c-add-always-2-1L	*dest/source1*, *source2*, *s*, *e*
CM:c-add-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-add-always-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-add-constant-2-1L	*dest/source1*, *source2-value*, *s*, *e*
CM:c-add-const-always-2-1L	*dest/source1*, *source2-value*, *s*, *e*
CM:c-add-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
CM:c-add-const-always-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*

Operands

dest The complex destination field.

source1 The complex first source field.

source2 The complex second source field.

source2-value A complex 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 $2(s + e + 1)$.

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 complex fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

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

```
if (always or $context\text{-}flag[k] = 1$) then
    $dest[k] \leftarrow source1[k] + source2[k]$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

Two operands, *source1* and *source2*, are added as complex numbers. The result is stored into memory. 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 constant operand *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-ASIN

Calculates the arc sine of the complex source field values and stores the result in the complex destination field.

Formats CM:c-asin-1-1L *dest/source*, *s*, *e*
CM:c-asin-2-1L *dest*, *source*, *s*, *e*

Operands

dest The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \sin^{-1} \textit{source}[k]$
    if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
```

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

The following definition of arc sine determines the range and branch cuts of a complex number z.

$$-i \log \left(i \times z + \sqrt{1 - z^2} \right)$$

C-ASINH

Calculates the arc hyperbolic sine of the complex source field values and stores the result in the complex destination field.

Formats CM:c-asinh-1-1L *dest/source*, *s*, *e*
CM:c-asinh-2-1L *dest*, *source*, *s*, *e*

Operands	*dest*	The complex destination field.
	source	The complex 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 $2(s+e+1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \sinh^{-1} source[k]$
```

The arc hyperbolic sine of the value of the *source* field is stored into the *dest* field.

The following definition of the inverse hyperbolic sine determines the range and branch cuts for a complex number *z*.

$$\log\left(z + \sqrt{1+z^2}\right)$$

C-ATAN

Calculates the arc tangent of the complx source field values and stores the result in the complex destination field.

Formats CM:c-atan-1-1L *dest*/*source*, *s*, *e*

CM:c-atan-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s+e+1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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] ← tan^{-1} source[k]
```

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

The following definition for arc tangent determines the range and branch cuts for a complex number z.

$$-i\log\left((1+i\times z)\times\sqrt{\frac{1}{(1+z^2)}}\right)$$

C-ATANH

Calculates the arc hyperbolic tangent of the complex source field values and stores the result in the complex destination field.

Formats

CM:c-atanh-1-1L *dest/source*, *s*, *e*
CM:c-atanh-2-1L *dest*, *source*, *s*, *e*

Operands

dest	The complex destination field.
source	The complex 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 $2(s+e+1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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] ← tanh⁻¹ source[k]
```

The arc hyperbolic tangent of the value of the *source* field is stored into the *dest* field.

The following definition of the arc hyperbolic tangent determines the range and branch cuts for a complex number *z*.

$$\log\left((1+z)\sqrt{1-\frac{1}{z^2}}\right)$$

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 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 *vp-set*. (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 context-flag[k] = 1 then
      dest[k].real ← cos source[k]
      dest[k].imag ← sin 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.

COMPRESS-HEAP

Invokes the heap compression mechanism on demand.

Formats CM:compress-heap

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

Heap compression removes heap memory fragmentation.

By default, the configuration variable CM:*heap-comression-enabled* is T (true), causing automatic heap compression whenever the stack and heap try to grow into each other. Therefore, under normal circumstances it not necessary to use the CM:compress-heap instruction.

Automatic heap compression can, however, make performance calculations unpredictable. To ensure deterministic performance, set CM:*heap-comression-enabled* to NIL (false, 0), arrange data structures to avoid fragmentation where possible, and explicitly invoke CM:compress-heap as necessary.

The variable CM:*heap-compression-messages-enabled* determines whether a message is issued when heap compression occurs. By default, this value is T (true, 1) and heap compression messages are issued. If this variable is NIL (false, 0), heap compression occurs without report.

C-CONJUGATE

The conjugate of the complex *source* field is placed in the complex *dest* field.

Formats CM:c-conjugate-1-1L *dest/source*, *s*, *e*
CM:c-conjugate-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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

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

Given a complex number C the conjugate C' consists of a real part equal to the real part of C and an imaginary part equal to the negation of the imaginary part of C. The conjugate of the complex *source* field is placed in the *dest* field.

C-COS

Calculates the cosine of the complex source field and stores the result in the complex destination field.

Formats

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

Operands

dest The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap

The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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[$k$] $\leftarrow$ cos source[$k$]
        if $\langle$overflow occurred in processor $k\rangle$ then overflow-flag[$k$] $\leftarrow$ 1
```

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

C-COSH

Calculates, in each selected processor, the hyperbolic cosine of the complex source field value and stores it in the complex destination field.

Formats CM:c-cosh-1-1L *dest/source*, *s*, *e*
CM:c-cosh-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s+e+1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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] ← cosh source[k]
      if ⟨overflow occurred in processor k⟩ then overflow-flag[k] ← 1
```

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

CROSS-VP-MOVE

Places a copy of all or a portion of the source field, taken from the current VP set, into the destination field, in another VP set. Specified axes and coordinates of the source VP set are mapped to specified axes and coordinates of the destination VP set and data is copied according to this mapping.

Formats CM:cross-vp-move-1L *dest*, *source*, *axis-mapping*, *source-axis-coords*, *dest-axis-coords*, *len*

Operands *dest* The dest field. This is in the destination VP set.

source The source field. This is in the current VP set.

axis-mapping A front-end vector of unsigned integer values, optionally including the null value CM:*no-axis*. The length of this vector is equal to the number of axes in the current VP set.

source-axis-coords A front-end vector of unsigned integer values, optionally including the null value CM:*no-axis*. The length of this vector is equal to the number of axes in the current VP set.

dest-axis-coords A front-end vector of unsigned integer values, optionally including the null value CM:*no-axis*. The length of this vector is equal to the number of axes in the *dest* VP set.

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

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

Context The non-always operations are conditional.

The always operations are unconditional.

In each participating processor, *len* bits are copied from the *source* field into the specified *dest* field, which may be in another processor.

The three arguments *axis-mapping*, *source-axis-coords*, and *dest-axis-coords* specify the size, shape, and orientation of the source data and of its destination. These are signed integer vectors. The length of the first two is equal to the rank (number of dimensions) of the current VP set (which is also the *source* VP set). The length of the third is equal to the rank of the destination VP set.

First, *axis-mapping* specifies, by position and value, a mapping between the axes of the source VP set geometry and the axes of the destination VP set geometry. Thus, source axis A maps to destination axis *axis-mapping*$[A]$. Any mapped axes must be of equal length.

Wherever *axis-mapping* contains the value CM:*no-axis*, only one element along the corresponding source axis is copied to the destination geometry. In this case, the cross-vp mapping is determined by the next two arguments.

The *source-axis-coords* vector specifies a coordinate point along each axis not mapped in the *axis-mapping* vector. The *source-axis-coords* vector must contain the null value CM:*no-axis* wherever the *axis-mapping* vector does not. Conversely, wherever *axis-mapping* contains the null value, *source-axis-coords* must contain an integer.

Each integer in the *source-axis-coords* vector specifies a coordinate along the corresponding source VP set axis. For example, if *source-axis-coords*$[A] = B$, only data of coordinate B along axis A of the source geometry will be copied to the destination geometry.

The *dest-axis-coords* vector specifies a coordinate point along each axis not mapped in the *axis-mapping* vector. Destination VP set axes are mapped in the *axis-mapping* vector by value. Thus, if *axis-mapping*$[A] = B$, then *dest-axis-coords*$[B]$ must be CM:*no-axis*; the remaining *dest-axis-coords* elements must be integers.

Each integer in the *dest-axis-coords* vector specifies a coordinate point along the corresponding destination VP set axis. For example, if *dest-axis-coords*$[A] = B$, only coordinate B along axis A of the destination geometry will receive data from the source geometry.

C-DIVIDE

The quotient of two complex source values is placed in the destination field.

Formats

CM:c-divide-2-1L	*dest*/*source1*, *source2*, *s*, *e*
CM:c-divide-always-2-1L	*dest*/*source1*, *source2*, *s*, *e*
CM:c-divide-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-divide-always-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-divide-constant-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
CM:c-divide-const-always-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
CM:c-divide-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
CM:c-divide-const-always-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
CM:c-divinto-2-1L	*dest*/*source2*, *source1*, *s*, *e*
CM:c-divinto-always-2-1L	*dest*/*source2*, *source1*, *s*, *e*
CM:c-divinto-constant-2-1L	*dest*/*source2*, *source1-value*, *s*, *e*
CM:c-divinto-const-always-2-1L	*dest*/*source2*, *source1-value*, *s*, *e*
CM:c-divinto-constant-3-1L	*dest*, *source2*, *source1-value*, *s*, *e*
CM:c-divinto-const-always-3-1L	*dest*, *source2*, *source1-value*, *s*, *e*

Operands

dest The complex destination field. This is the quotient.

source1 The complex first source field. This is the dividend.

source2 The complex second source field. This is the divisor.

source1-value A complex immediate operand to be used as the first source.

source2-value A complex 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 $2(s + e + 1)$.

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 complex 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 unaffected.

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 (always or *context-flag*$[k] = 1$) then
    *dest*$[k] \leftarrow$ *source1*$[k]$/*source2*$[k]$
    if *source2*$[k] = 0$ then *test-flag*$[k] \leftarrow 1$
    if ⟨overflow occurred in processor $k$⟩ then *overflow-flag*$[k] \leftarrow 1$
```

The *source1* operand is divided by the *source2* operand, treating both as complex numbers. The result is stored into memory. 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 constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-EQ

Compares two complex source values. The *test-flag* is set if they are equal, and otherwise it is cleared.

Formats

CM:c-eq-1L	*source1*, *source2*, *s*, *e*
CM:c-eq-constant-1L	*source1*, *source2-value*, *s*, *e*
CM:c-eq-zero-1L	*source1*, *s*, *e*

Operands

source1 The complex first source field.

source2 The complex second source field.

source2-value A complex immediate operand to be used as the second source. For CM:c-eq-zero-1L, this implicitly has the value zero.

s, *e* The significand and exponent lengths for the *source1* and *source2* fields. The total length of an operand in this format is $2(s+e+1)$.

Overlap The fields *source1* and *source2* may overlap in any manner.

Flags *test-flag* is set if *source1* is equal to *source2*; otherwise it is cleared.

Context This operation is conditional. The 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 $context\text{-}flag[k] = 1$ then
    if $source1[k] = source2[k]$
      $test\text{-}flag[k] \leftarrow 1$
    else
      $test\text{-}flag[k] \leftarrow 0$
```

Two operands are compared as complex numbers. The first operand is a memory field; the second is a memory field or an immediate value. The *test-flag* is set if the first operand is equal to the second operand, and is cleared otherwise. Note that comparisons ignore the sign of zero; $+0$ and -0 are considered to be equal.

The constant operand *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-EXP

The exponent of the complex source field is stored in the complex destination field.

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

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $context\text{-}flag[k] = 1$ then
        $dest[k] \leftarrow \exp source[k]$
        if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

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

F-EXP

Calculates, in each selected processor, the exponential function 2^s, where s is the floating-point source field, and stores the result in the floating-point destination field.

Formats CM:f-exp2-1-1L *dest/source*, *s*, *e*
CM:f-exp2-2-1L *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 $\textit{context-flag}[k] = 1$ then
    if $\textit{source}[k] = +\infty$ then
      $\textit{dest}[k] \leftarrow +\infty$
    else if $\textit{source}[k] = -\infty$ then
      $\textit{dest}[k] \leftarrow +0$
    else
      $\textit{dest}[k] \leftarrow 2^{\textit{source}[k]}$
    if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
```

Call the value of the *source* field s; the value 2^s is stored into the *dest* field.

GET-FROM-POWER-TWO

Each processor gets a message from a processor that is a specified distance away in the NEWS grid. The distance must be a power of two.

Formats

CM:get-from-power-two-1L	*dest*, *source*, *axis*, *log-2-distance*, *direction*, *len*
CM:get-from-power-two-always-1L	*dest*, *source*, *axis*, *log-2-distance*, *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.

log-2-distance An unsigned integer immediate operand to be used as the base 2 logarithm of *distance*, where *distance* must be a power of 2.

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

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*.

Note that in the conditional case data storage depends only on the *context-flag* of the processor receiving the data, not on the *context-flag* of the processor from which the data is obtained.

Definition

```
For every virtual processor k in the current-vp-set do
  if (always or context-flag[k] = 1) then
    let g = geometry(current-vp-set)
    dest[k] ← source[news-relative(g, k, axis, direction, log-2-distance)]
```

where *news-relative* is defined in the NEWS Communication section of the Instruction Set Overview chapter.

The *dest* field in each processor receives the contents of the *source* field of that processor's relative along the NEWS axis specified by *axis*, in the direction specified by *direction*, and at the distance specified by *log-2-distance*.

The immediate operand *log-2-distance*, is $\log_2$ *distance*, where *distance* is the distance, along axis *axis*, between each destination processor and the source processor from which it retrieves data. In terms of this operand, *distance* is $2^{\textit{log-2-distance}}$.

If *direction* is :upward then each processor retrieves data from a relative whose NEWS coordinate is (*coordinate* + *distance* mod *axis-length*). For most processors, this means getting from a processor whose coordinate is greater. The GET wraps around however; the processor whose coordinate is greatest retrieves data from the processor whose coordinate is (0 + *distance*).

If *direction* is :downward then each processor retrieves data from a relative whose NEWS coordinate is (*coordinate* − *distance* mod *axis-length*). For most processors, this means getting from a processor whose coordinate is less. The GET wraps around however; the processor whose coordinate is zero retrieves data from the processor whose coordinate is (*max-coordinate*(*axis*) − *distance*).

GLOBAL-C-ADD

The sum of the values in the complex source field is returned to the front end as a complex number.

Formats result ← CM:global-c-add-1L *source*, *s*, *e*

Operands *source* The complex source field.

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

Result A complex number, the sum of the *source* field.

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

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

Definition

```
Let $P = \{ m \mid 0 \le m <$ CM:*user-send-address-limit* $\}$
Let $S = \{ m \mid m \in P \wedge context\text{-}flag[m] = 1 \}$
If $|S| = 0$ then
  return $+0$ to front end
else
  return $\left( \sum_{m \in S} source[m] \right)$ to front end
```

The CM:global-c-add-1L operation sums the *source* field values from all selected processors, treated as complex numbers. The sum is sent to the front-end computer as a complex number and returned as the result of the operation. If there are no selected processors, then the value +0 is returned.

F-IEEE-TO-VAX

Converts the floating-point source field values from IEEE floating-point format to VAX floating-point format and stores the result in the destination field.

Formats CM:f-ieee-to-vax-1L *vax-dest*, *ieee-source*, *len*

Operands

vax-dest The floating-point destination field.

ieee-source The floating-point source field.

len The length of the *vax-dest* and *ieee-source* fields. The value of *len* must be either 32 or 64.

Overlap The fields *vax-dest* and *ieee-source* may overlap in any manner.

Flags *overflow-flag* is set if the ieee-source cannot be represented in the destination field; otherwise it is cleared. If *ieee-source* represents ∞ or NaN, then *vax-dest* is set to the "undefined variable" value in VAX format and the *overflow-flag* is cleared. If *ieee-source* represents −0.0, it is converted to VAX 0.0 and the *overflow-flag* is cleared.

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

The Connection Machine operates internally on floating point data in IEEE format whereas the VAX uses a VAX floating-point format. In each active processor, this function converts a floating-point field in standard IEEE format to a field in VAX format.

The value of *len* specifies the precision of *vax-dest*. If *len* is specified as 32, then VAX 'F' format is used. If *len* is specified as 64, then VAX 'D' format is used.

VAX and IEEE floating-point formats are incompatible, so there are a number of potential inaccuracies in the translation. In general, if the conversion is accurate then the overflow flag is cleared; if inaccurate, then the overflow flag is set. See the flags description above.

This instruction is useful for rapidly converting floating-point data to VAX format, even if a VAX front end is not being used. For example, if data is to be transferred from a file in the CM file system to a VAX, CM:f-ieee-to-vax-1L should be called before writing the data file.

All Paris CM to front end data transfer functions automatically convert the data to the appropriate front-end format so it is not necessary to call CM:ieee-to-vax before calling, for instance, one of the read-from-news-array instructions.

To convert data back to IEEE floating-point format, see the definition of CM:f-vax-to-ieee-1L.

INTERN-DETAILED-GEOMETRY

Returns an interned geometry given detailed information about how the grid is laid out.

Formats result ← CM:intern-detailed-geometry *axis-descriptor-array*, *[rank]*

Operands *axis-descriptor-array* A front-end vector of descriptors for the grid axes. In the C interface, the elements of the *axis-descriptor-array* must be of type CM_axis_descriptor_t, that is, they must be pointers to structures of type CM_axis_descriptor.

In the Lisp interface, the *axis-descriptor-array* may be either a list of descriptors or an array of descriptors.

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

Result A geometry-id, identifying the existing or newly created interned geometry.

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

By using interned geometries, modules that require identical geometries can use identical geometries – without having to keep track of the geometry-id's.

CM:intern-detailed-geometry takes an array of descriptors. Each descriptor describes one NEWS axis in some detail. Most of the components are unsigned integers, but the value of the *ordering* component must be either :news-order or :send-order. The CM:create-detailed-geometry dictionary entry defines the type of the ordering component and of the descriptor for each language interface.

CM:intern-detailed-geometry is identical to CM:create-detailed-geometry with this exception: it returns an *interned* geometry-id. An interned geometry-id is a geometry-id returned by CM:intern-detailed-geometry or by CM:intern-geometry; a geometry-id returned by CM:create-detailed-geometry or by CM:create-geometry may *not* be interned.

CM:create-detailed-geometry returns a unique, uninterned geometry-id each time it is called. In contrast, CM:intern-detailed-geometry returns an existing interned geometry-id if it can. If there is an interned geometry with an axis descriptor array that matches the supplied *axis-descriptor-array*, it is returned. Otherwise, CM:intern-detailed-geometry returns a new interned geometry-id. The returned geometry-id may be used to create a VP set or to respecify the geometry of an existing VP set.

Once the interned geometry has been created, the user may destroy the array created to provide the dimension information. All necessary information is copied from this array when the geometry is created.

INTERN-GEOMETRY

Returns an interned geometry given grid axis lengths.

Formats result ← CM:intern-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 in between 1 and CM:*max-geometry-rank*, inclusive. This argument is not provided when calling Paris from Lisp.

Result A geometry-id, identifying the existing or newly created interned geometry.

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

By using interned geometries, codes that require identical geometries can use identical geometries – without having to keep track of the geometry-id's.

CM:intern-geometry is identical to CM:create-geometry with this exception: it returns an *interned* geometry-id. An interned geometry-id is a geometry-id returned by CM:intern-geometry or by CM:intern-detailed-geometry; a geometry-id returned by CM:create-geometry or by CM:create-detailed-geometry may *not* be interned.

CM:create-geometry returns a unique, uninterned geometry-id each time it is called. In contrast, CM:intern-geometry returns an existing interned geometry-id if it can. If there is a geometry, created by CM:intern-geometry and with dimensions that match those specified in *dimension-array*, it is returned. Otherwise, CM:intern-geometry returns a new interned geometry-id. The returned geometry-id may be used to create a VP set or to respecify the geometry of an existing VP set.

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.

The geometry is laid out so as to optimize performance under the assumption that the axes are used equally frequently for NEWS communication. The operations CM:create-detailed-geometry or CM:intern-detailed-geometry may be used instead to more precisely control layout for performance tuning.

Once the interned 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 when the geometry is created.

INTERN-IDENTICAL-VP-SET

Returns an interned VP set, within which fields may be allocated.

Formats result ← CM:intern-identical-vp-set *geometry-id*

Operands *geometry-id* A geometry-id.

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

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

This operation returns a vp-set-id for an *interned* VP set. An interned VP set is a VP set referenced by a vp-set-id returned by CM:intern-identical-vp-set. VP set interning allows different modules to reference identical VP sets and reduces VP set memory management overhead.

CM:intern-identical-vp-set returns an existing, interned vp-set-id if there is an existing, interned VP set whose geometry is identical to the geometry specified by *geometry-id.* Otherwise, CM:intern-identical-vp-set returns a new, interned vp-set-id.

Once a VP set has been created as interned, it may never be uninterned. Similarly, an uninterned VP set (created for instance with CM:create-vp-set) may never become interned.

An interned VP set may be used in the same ways as an uninterned VP set. For instance, it may be given to other Paris operations in order to create memory fields in which data may be stored. It may also be deallocated with CM:deallocate-vp-set.

IS-FIELD-AN-ALIAS

Returns true if the specified field-id is an alias field-id, false otherwise.

Formats result ← CM:is-field-an-alias *field-id*

Operands *field-id* A field-id.

Result True if *field-id* is an alias field-id, and false otherwise.

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

This operation allows a program to determine whether a given field-id is an alias field-id created with CM:make-field-alias as opposed to a regular field-id created with a field allocation instruction such as CM:allocate-stack-field.

C-LN

The natural logarithm of the complex source field values is placed in the complex destination field.

Formats CM:c-ln-1-1L *dest/source*, *s*, *e*
CM:c-ln-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

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

Flags *test-flag* is set if the *source* is zero; 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \ln source[k]$
```

The value $\ln s$ is stored into the *dest* field, where s is the value of the *source* field. This is the natural logarithm to the base $e \approx 2.718281828\ldots$.

F-LOG2

The base two logarithm of the floating-point source field is placed in the floating-point destination field.

Formats CM:f-log-1-1L *dest/source*, *s*, *e*
CM:f-log-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 *test-flag* is set if the *source* is zero; 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \log_2 source[k]$
```

The value $\log_2 s$ is stored into the *dest* field, where s is the value of the *source* field. This is the logarithm to the base two of the floating-point source field.

F-LOG10

The base ten logarithm of the floating-point source field is placed in the floating-point destination field.

Formats CM:f-log10-1-1L *dest/source*, *s*, *e*
CM:f-log10-2-1L *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 *test-flag* is set if the *source* is zero; 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \log_{10} source[k]$
```

The value $\log_{10} s$ is stored into the *dest* field, where s is the value of the *source* field. This is the logarithm to the base ten of the floating-point source field.

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 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 original 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, which is the number of bits that the field occupies in each physical processor. The physical length is equal to the field length (the number of bits the field occupies in each virtual processor) multiplied by the VP ratio of the current VP set.

It is possible for the physical length of an alias field to be different from the physical 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 $context\text{-}flag[k] = 1$ then
    if $source2[k] = 0$ then
      $dest[k] \leftarrow \langle\text{unpredictable}\rangle$
      $test\text{-}flag[k] \leftarrow 1$
    else
      $dest[k] \leftarrow source1[k] - source2[k] \times \left\lfloor \frac{source1[k]}{source2[k]} \right\rfloor$
      $test\text{-}flag[k] \leftarrow 0$
    if $\langle\text{overflow occurred in processor } k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$
```

The residue resulting from the reduction of the floating-point *source1* operand divided by the *source2* operand is stored in the *dest* field. 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 constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-MOVE

Copies a complex source value into the destination field.

Formats

CM:c-move-2L	*dest*, *source*, *ds*, *de*, *ss*, *se*
CM:c-move-1L	*dest*, *source*, *s*, *e*
CM:c-move-always-1L	*dest*, *source*, *s*, *e*
CM:c-move-constant-1L	*dest*, *source-value*, *s*, *e*
CM:c-move-const-always-1L	*dest*, *source-value*, *s*, *e*
CM:c-move-zero-1L	*dest*, *s*, *e*
CM:c-move-zero-always-1L	*dest*, *s*, *e*

Operands

dest The complex destination field.

source The complex source field.

source-value The complex source field. For CM:c-move-zero-1L and CM:c-move-zero-always-1L, this implicitly has the value zero.

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

ds, *de* For CM:c-move-2L, the significand and exponent lengths for the *dest* field. The total length of an operand in this format is $2(ds + de + 1)$.

ss, *se* For CM:c-move-2L, the significand and exponent lengths for the *source* field. The total length of an operand in this format is $2(ss + se + 1)$.

Overlap

The fields *dest* and *source* may overlap in any manner.

Flags

overflow-flag is set if the result cannot be represented in the destination field; otherwise it is cleared. This can occur only for CM:c-move-2L.

Context

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

The always operations are unconditional. The destination and flag 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 $context\text{-}flag[k] = 1$) then
    $dest[k] \leftarrow source[k]$
    if $\langle$overflow occurred in processor $k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$
    else $overflow\text{-}flag[k] \leftarrow 0$
  as appropriate.
```

The *source* field or value is copied into the *dest* field.

However, overlapping fields are not handled carefully and should be avoided.

F-MULT-SUBF

Calculates a value $b - xa$ and places it in the destination.

Formats

CM:f-mult-subf-1L	*dest*, *source1*, *source2*, *source3*, *s*, *e*
CM:f-mult-const-subf-1L	*dest*, *source1*, *source2-value*, *source3*, *s*, *e*
CM:f-mult-subf-const-1L	*dest*, *source1*, *source2*, *source3-value*, *s*, *e*
CM:f-mult-const-subf-const-1L	*dest*, *source1*, *source2-value*, *source3-value*, *s*, *e*

Operands

dest The floating-point destination field.

source1 The floating-point first source field.

source2 The floating-point second source (multiplier) field.

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

source3 The floating-point third source (minuend) field.

source3-value A floating-point immediate operand to be used as the third source (minuend).

s, *e* The significand and exponent lengths for the *dest*, *source1*, *source2*, and *source3* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *source1*, *source2*, and *source3* 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 *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 $\textit{context-flag}[k] = 1$ then
        $\textit{dest}[k] \leftarrow \textit{source3}[k] - (\textit{source1}[k] \times \textit{source2}[k])$
        if $\langle$overflow occurred in processor $k\rangle$ then $\textit{overflow-flag}[k] \leftarrow 1$
```

Two operands *source1* and *source2* are multiplied as floating-point numbers and the product is subtracted from a third operand, *source3*. The result is stored into memory. The various operand formats allow operands to be either memory fields or constants.

The constant operands *source2-value* and *source3-value* should be double-precision front-end values (in Lisp, automatic coercion is performed if necessary). The constants are then converted, in effect, to the format specified by *s* and *e* before the operation is performed.

A call to CM:f-mult-subf-1L is equivalent to the sequence

```
CM:f-multiply-3-1L   temp, source1, source2, s, e
CM:f-subtract-3-1L   dest, source3, temp, s, e
```

but may be faster.

C-MULTIPLY

The product of two complex source values is placed in the destination field.

Formats	CM:c-multiply-2-1L	*dest*/*source1*, *source2*, *s*, *e*
	CM:c-multiply-always-2-1L	*dest*/*source1*, *source2*, *s*, *e*
	CM:c-multiply-3-1L	*dest*, *source1*, *source2*, *s*, *e*
	CM:c-multiply-always-3-1L	*dest*, *source1*, *source2*, *s*, *e*
	CM:c-multiply-constant-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
	CM:c-multiply-const-always-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
	CM:c-multiply-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
	CM:c-multiply-const-always-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*

Operands *dest* The complex destination field.

source1 The complex first source field.

source2 The complex second source field.

source2-value A complex 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 $2(s + e + 1)$.

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 complex fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow source1[k] \times source2[k]$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

Two operands, *source1* and *source2*, are multiplied as complex numbers. The result is stored into memory. 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 constant operand *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

MULTISPREAD-C-ADD

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

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

Operands *dest* The complex destination field.

source The complex 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 $2(s+e+1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    let $r = rank(g)$
    let $axis\text{-}set = \{ m \mid 0 \le m < r \land (axis\text{-}mask\langle m \rangle = 1) \}$
    let $C_k = \{ m \mid m \in hyperplane(g, k, axis\text{-}set) \land context\text{-}flag[m] = 1 \}$
    $dest[k] \leftarrow \left( \sum_{m \in C_k} source[m] \right)$
where hyperplane is as defined on page 34 of the Paris Reference Manual.
```

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

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

```
for all integers $j$, $0 \le j < rank(geometry(current\text{-}vp\text{-}set))$, in any sequential order, do
  if $axis\text{-}mask\langle j \rangle = 1$ then
    CM:spread-with-c-add-1L  dest, source, j, s, e
```

but may be faster.

C-NE

Compares two complex source values. The *test-flag* is set if they are not equal; otherwise it is cleared.

Formats	CM:c-ne-1L	*source1*, *source2*, *s*, *e*
	CM:c-ne-constant-1L	*source1*, *source2-value*, *s*, *e*
	CM:c-ne-zero-1L	*source1*, *s*, *e*

Operands *source1* The complex first source field.

source2 The complex second source field.

source2-value A complex immediate operand to be used as the second source. For CM:c-ne-zero-1L, this implicitly has the value zero.

s, *e* The significand and exponent lengths for the *source1* and *source2* fields. The total length of an operand in this format is $2(s+e+1)$.

Overlap The fields *source1* and *source2* may overlap in any manner.

Flags *test-flag* is set if *source1* is not equal to *source2*; otherwise it is cleared.

Context This operation is conditional. The 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 source1[k] ≠ source2[k]
      test-flag[k] ← 1
    else
      test-flag[k] ← 0
```

Two operands are compared as complex numbers. The first operand is a memory field; the second is a memory field or an immediate value. The *test-flag* is set if the first operand is not equal to the second operand, and is cleared otherwise. Note that comparisons ignore the sign of zero; $+0$ and -0 are considered to be equal.

The constant operand *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-NEGATE

Copies a complex number with both signs inverted.

Formats CM:c-negate-1-1L *dest/source*, *s*, *e*
CM:c-negate-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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
      $\textit{dest}[k].\textit{real} \leftarrow -\textit{source}[k].\textit{real}$
      $\textit{dest}[k].\textit{imag} \leftarrow -\textit{source}[k].\textit{imag}$
```

A copy of the *source* operand, with both sign bits inverted, is placed in the *dest* operand.

F-NEWS-ADD

The sum of two floating-point source values (one from a NEWS neighbor) is placed in the destination field.

Formats

CM:f-news-add-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-add-always-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-add-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-add-always-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-add-const-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*
CM:f-news-add-const-a-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*

Operands

dest The floating-point destination field.

source The floating-point source field.

source1 The floating-point first source field.

source2 The floating-point second source field.

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

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

direction Either :upward or :downward.

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

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

Context

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

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

Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

```
For every virtual processor k in the current-vp-set do
   if $context\text{-}flag[k] = 1$ then
      let $g = geometry(current\text{-}vp\text{-}set)$
      $dest[k] \leftarrow source1[k] + source2[news\text{-}neighbor(g, k, axis, direction)]$
      if ⟨overflow occurred in processor k⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

where *news-neighbor* is is defined in the NEWS Communication section of the Instruction Set Overview Chapter.

Two source operands are added as floating-point numbers and the result is stored in *dest*. The various operand formats allow source operands to be either memory fields or constants. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

The instructions with two operands take *source* from a NEWS neighbor, sum it with *dest* and store the result back in *dest*.

For the instructions CM:f-news-add-3-1L and CM:f-news-add-always-3-1L, *source2* is taken from a NEWS neighbor.

The instructions CM:f-news-add-const-3-1L and CM:f-news-add-const-a-3-1L take *source1* is from a NEWS neighbor. Note that the *a* in CM:f-news-add-const-a-3-1L stands for "always."

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

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

CM:get-from-news-1L *temp*, *source2*, *axis*, *direction*, *len*
CM:f-add-3-1L *dest*, *source1*, *temp*, *s*, *e*

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-ADD-MULT

Calculates the value $(a + x)b$, where one of the operands is taken from a NEWS neighbor, and places the result in the destination.

Formats

CM:f-news-add-mult-4-1L	*dest*, *source1*, *source2*, *source3*, *axis*, *direction*, *s*, *e*
CM:f-news-add-const-mult-4-1L	*dest*, *source1*, *source2-value*, *source3*, *axis*, *directio*

Operands

dest The floating-point destination field.

source1 The floating-point first source field.

source2 The floating-point second source field.

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

source3 A floating-point immediate operand to be used as the third source.

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

direction Either :upward or :downward.

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

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. Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

```
For every virtual processor $k$ in the $current\text{-}vp\text{-}set$ do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow (source1 + source2[news\text{-}neighbor(g, k, axis, direction)]) \times source3[k$
    if $\langle$overflow occurred in processor $k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$
```

NEWS-ADD-MULT

The sum of two source operands is multiplied by the value of a third source operand. The result is stored in *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. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

The CM:f-news-add-mult-4-1L instruction takes *source2* from a NEWS neighbor. For the CM:f-news-add-const-mult-4-1L instruction, *source2* is a constant and *source3* is taken from a NEWS neighbor.

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-add-mult is equivalent to the sequence

```
CM:get-from-news-1L   temp, source2, axis, direction, len
CM:f-add-mult-1L   souce1, temp, source3, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-MULT

The product of two floating-point source values (one from a NEWS neighbor) is placed in the destination field.

Formats

CM:f-news-mult-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-always-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-always-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-const-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-const-a-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*

Operands

dest The floating-point destination field.

source1 The floating-point first source field.

source2 The floating-point second source field.

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

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

direction Either :upward or :downward.

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

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

Context

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

The always operations are unconditional. The destination and flag may be altered regardless of the value of the *context-flag*. Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

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

```
if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow source1[k] \times source2[news\text{-}neighbor(g, k, axis, direction)]$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

Two source operands are multiplied as floating-point numbers. The result is stored in *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. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

The instructions with two operands take *source* from a NEWS neighbor, multiply it with *dest*, and store the result back in *dest*.

For the instructions CM:f-news-mult-3-1L and CM:f-news-mult-always-3-1L, *source2* is taken from a NEWS neighbor.

For the instructions CM:f-news-mult-const-3-1L and CM:f-news-mult-const-a-3-1L, *source1* is taken from a NEWS neighbor. Note that the *a* in CM:f-news-mul-const-always-3-1L stands for "always." This is necessary to meet the 31 character limit on instruction names.

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-mult-3-1L is equivalent to the sequence

```
CM:get-from-news-1L  temp, source2, axis, direction, len
CM:f-multiply-3-1L   dest, source1, temp, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-MULT-ADD

The product of two floating-point source values (one from a NEWS neighbor) is added to yet another floating-point source value; the result is placed in the destination field.

Formats

CM:f-news-mult-add-4-1L *dest*, *source1*, *source2*, *source3*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-const-add-4-1L *dest*, *source1*, *source2-value*, *source3*, *axis*, *directio*

Operands

dest The floating-point destination field.

source1 The floating-point multiplicand (from news neighbor) field.

source2 The floating-point multiplier field.

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

source3 The floating-point augend field.

source3-value A floating-point immediate operand to be used as the augend.

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

direction Either :upward or :downward.

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*, *source2*, and *source3* 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

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.

Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

```
For every virtual processor $k$ in the current-vp-set do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow source1[k] \times source2[news\text{-}neighbor(g, k, axis, direction)] + source3[k$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

Two operands are multiplied as floating-point numbers; to the product is added a third operand. The result is stored into memory. The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

For CM:f-news-mult-add-4-1L, *source2* is taken from a NEWS neighbor.

For CM:f-news-mult-const-add-4-1L, *source2* is a constant and *source3* is taken from a NEWS neighbor.

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* or *source3-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-mult-add-4-1L is equivalent to the sequence

```
CM:get-from-news-1L  temp, source2, axis, direction, len
CM:f-multiply-3-1L  temp, source1, temp, s, e
CM:f-add-3-1L  dest, temp, source3, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-MULT-SUB

From the product of two floating-point source values (one from a NEWS neighbor) is subtracted yet another floating-point source value; the result is placed in the destination field.

Formats

CM:f-news-mult-sub-4-1L	*dest*, *source1*, *source2*, *source3*, *axis*, *direction*, *s*, *e*
CM:f-news-mult-const-sub-4-1L	*dest*, *source1*, *source2-value*, *source3*, *axis*, *direction*, *s*, *e*

Operands

dest The floating-point destination field.

source1 The floating-point multiplicand field.

source2 The floating-point multiplier field.

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

source3 The floating-point subtrahend field.

source3-value A floating-point immediate operand to be used as the subtrahend.

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

direction Either :upward or :downward.

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*, *source2*, and *source3* 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 *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.

Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

```
For every virtual processor $k$ in the $current\text{-}vp\text{-}set$ do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow source1[k] \times source2[news\text{-}neighbor(g, k, axis, direction)] - source3[k$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

Two operands, *source1* and *source2*, are multiplied as floating-point numbers; from the product is subtracted a third operand, *source3*. The result is stored into memory. The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

For CM:f-news-mult-sub-4-1L, *source2* is taken from a NEWS neighbor.

For and CM:f-news-mult-const-sub-4-1L, *source2* is a constant and *source3* is taken from a NEWS neighbor.

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* or *source3-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-mult-sub-4-1L is equivalent to the sequence

```
CM:get-from-news-1L  temp, source2, axis, direction, len
CM:f-multiply-3-1L  temp, source1, temp, s, e
CM:f-subtract-3-1L  dest, temp, source3, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-SUB

The difference of two floating-point source values (one from a NEWS neighbor) is placed in the destination field.

Formats

CM:f-news-sub-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-always-2-1L	*dest*, *source*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-always-3-1L	*dest*, *source1*, *source2*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-const-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-const-a-3-1L	*dest*, *source1*, *source2-value*, *axis*, *direction*, *s*, *e*

Operands

dest The floating-point destination field. This is the difference, the result of the subtraction operation.

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

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

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

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

direction Either :upward or :downward.

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

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

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

Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

NEWS-SUB

Definition

```
For every virtual processor $k$ in the $current\text{-}vp\text{-}set$ do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow source1[k] - source2[news\text{-}neighbor(g, k, axis, direction)]$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

The operands are treated as as floating-point numbers and one is subtracted from another. The result is stored into the 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. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

The instructions with two operands take *source* from a NEWS neighbor, subtract it from *dest*, and store the result stored back in *dest*.

For the instructions CM:f-news-sub-3-1L and CM:f-news-sub-always-3-1L, *source2* is obtained from a NEWS neighbor.

For the instructions CM:f-news-sub-const-3-1L and CM:f-news-sub-const-a-3-1L, *source2* is a constant and *source1* is obtained from a NEWS neighbor. Note that the *a* in CM:f-news-sub-const-a-3-1L stands for "always."

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-sub-3-1L is equivalent to the sequence

```
CM:get-from-news-1L   temp, source2, axis, direction, len
CM:f-subtract-3-1L   dest, source1, temp, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-NEWS-SUB-MULT

Calculates the value $(a - x)b$, when one of the operands is taken from a NEWS neighbor, and places the result in the destination.

Formats

CM:f-news-sub-mult-4-1L *dest*, *source1*, *source2*, *source3*, *axis*, *direction*, *s*, *e*
CM:f-news-sub-const-mult-4-1L *dest*, *source1*, *source2-value*, *source3*, *axis*, *directior*

Operands

dest The floating-point destination field.

source1 The floating-point first source field.

source2 The floating-point second source field.

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

source3 The floating-point third source field.

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

direction Either :upward or :downward.

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

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.

Note that in the conditional cases the storing of data depends only on the *context-flag* of the processor receiving the data.

Definition

```
For every virtual processor $k$ in the $current\text{-}vp\text{-}set$ do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    $dest[k] \leftarrow (source1 - source2[news\text{-}neighbor(g, k, axis, direction)]) \times source3[k]$
    if $\langle$overflow occurred in processor $k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$
```

NEWS-SUB-MULT

The difference of two operands is multiplied by the value of a third operand. The result is stored into memory. The various operand formats allow operands to be either memory fields or constants; in some cases the destination field initially contains one source operand. Each instruction takes one source field from a NEWS neighbor; the default is *source2*.

The CM:f-news-sub-mult-4-1L instruction takes *source2* from a NEWS neighbor. For the CM:f-news-sub-const-mult-4-1L instruction, *source2* is a constant and *source3* is taken from a NEWS neighbor.

If *direction* is :upward then each processor retrieves data from the neighbor whose NEWS coordinate is one greater along *axis*, with the processor whose coordinate is greatest retrieving data from the processor whose coordinate is zero.

If *direction* is :downward then each processor retrieves data from the neighbor whose NEWS coordinate is one less along *axis*, with the processor whose coordinate is zero retrieving data from the processor whose coordinate is greatest.

The constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

A call to CM:f-news-sub-mult-4-1L is equivalent to the sequence

```
CM:get-from-news-1L   temp, source2, axis, direction, len
CM:f-sub-mult-1L   dest, source1, temp source3, s, e
```

but is faster at high VP ratios and requires little temporary memory.

F-C-PHASE

Calculates the phase of the complex source field and puts the result in the floating-point destination field.

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

Operands *dest* The floating-point destination field.

source The complex 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 $s + e + 1$. The total length of the *source* field in this format is $2(s + e + 1)$.

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

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] ← atan2(source[k].imag, source[k].real)
        if ⟨overflow occurred in processor k⟩ then overflow-flag[k] ← 1
```

The phase of a number is the angle part of its polar representation as a complex number.

C-C-POWER

Raises a complex number to a complex power.

Formats

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

Operands

dest The complex destination field.

source1 The complex first source field.

source2 The complex second source field.

source2-value A complex 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 $2(s + e + 1)$.

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 complex fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if floating-point overflow occurs; otherwise it is unaffected. *test-flag* is set if zero is raised to a non-positive power; 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
    $dest[k] \leftarrow source1[k]^{source2[k]}$
    if $source1[k] = 0.0$ and $source2[k].real \leq 0.0$
    and $source2[k].imag = 0.0$ then
      $test\text{-}flag[k] \leftarrow 1$
    else $test\text{-}flag[k] \leftarrow 0$
    if ⟨overflow occurred in processor k⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

The *source1* field (the base) is raised to the power *source2* (the exponent), using exp and ln operations.

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 constant operand *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-F-POWER

Raises a complex number to a floating-point power.

Formats

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

Operands

dest The complex destination field.

source1 The complex first source field.

source2 The floating-point second source field.

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

s, *e* The significand and exponent lengths for the *dest and source1* and *source2* fields. The total length of the *dest and source1* field in this format is $2(s + e + 1)$. The total length of the *source2* field 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 complex fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

Flags *overflow-flag* is set if floating-point overflow occurs; otherwise it is unaffected. *test-flag* is set if zero is raised to a non-positive power; 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 $\textit{current-vp-set}$ do
  if $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \textit{source1}[k]^{\textit{source2}[k]}$
    if $\textit{source1}[k] = 0.0$ and $\textit{source2}[k].\textit{real} \leq 0.0$
    and $\textit{source2}[k].\textit{imag} = 0.0$ then
      $\textit{test-flag}[k] \leftarrow 1$
    else $\textit{test-flag}[k] \leftarrow 0$
    if $\langle$overflow occurred in processor $k\rangle$ then $\textit{overflow-flag}[k] \leftarrow 1$
```

The *source1* field (the base) is raised to the power *source2* (the exponent), using exp and ln operations.

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 constant operand *source2-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-S-POWER

Raises a complex number to a signed integer power.

Formats

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

Operands

dest The complex destination field.

source1 The complex base field.

source2 The signed integer exponent field.

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 $2(s+e+1)$.

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 complex 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. *test-flag* is set if zero is raised to a negative power; 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
    $dest[k] \leftarrow source1[k]^{source2[k]}$
    if $source1[k] = 0.0$ and $source2[k] < 0$ then
      $test\text{-}flag[k] \leftarrow 1$
    else $test\text{-}flag[k] \leftarrow 0$
    if ⟨overflow occurred in processor k⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

The *source1* field (the base) is raised to the power *source2* (the exponent), using repeated multiplications.

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.

C-U-POWER

Raises a complex number to an unsigned integer power.

Formats

CM:c-u-power-3-2L	*dest*, *source1*, *source2*, *slen2*, *s*, *e*
CM:c-u-power-2-2L	*dest*/*source1*, *source2*, *slen2*, *s*, *e*
CM:c-u-power-constant-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
CM:c-u-power-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*

Operands

dest The complex destination field.

source1 The complex base field.

source2 The unsigned integer exponent field.

source2-value An unsigned 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 $2(s+e+1)$.

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. 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 complex 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 $\mathit{context\text{-}flag}[k] = 1$ then
    $\mathit{desk}[k] \leftarrow \mathit{source1}[k]^{\mathit{source2}[k]}$
    if ⟨overflow occurred in processor $k$⟩ then $\mathit{overflow\text{-}flag}[k] \leftarrow 1$
```

The *source1* field (the base) is raised to the power *source2* (the exponent), using repeated multiplications.

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-U-POWER

Raises a signed integer to a unsigned integer power.

Formats

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

Operands

dest The signed integer destination field.

source1 The signed integer base field.

source2 The unsigned integer exponent field.

source2-value An unsigned 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-u-power-3-3L and CM:s-u-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*.

slen1 For CM:s-u-power-3-3L and CM:s-u-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*.

slen2 For CM:s-u-power-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. However, *source1* must be either disjoint from or identical to the *dest* field while *source2* must be disjoint from 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 $context\text{-}flag[k] = 1$ then
   if $source2[k] = 0$ then
      $dest[k] \leftarrow 1$
   else
   $dest[k] \leftarrow (source1[k])^{source2[k]}$
   if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
   else $overflow\text{-}flag[k] \leftarrow 0$
```

The *source1* field (the base) is raised to the power *source2* (the exponent). 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 *source2-value* should be an unsigned 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-S-POWER

Raises a unsigned integer to a signed integer power.

Formats

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

Operands

dest The unsigned integer destination field.

source1 The unsigned 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:u-s-power-3-3L and CM:u-s-power-constant-3-2L, the length of the *dest* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen1 For CM:u-s-power-3-3L and CM:u-s-power-constant-3-2L, the length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen2 For CM:u-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. However, *source1* must be either disjoint from or identical to the *dest* field while *source2* must be disjoint from 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.

test-flag is set if zero is raised to a negative power; 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
    $\textit{test-flag}[k] \leftarrow 0$
    if $\textit{source1}[k] = 0$ then
      $\textit{test-flag}[k] \leftarrow 1$
    if $\textit{source2}[k] < 0$ then
      $\textit{dest}[k] \leftarrow \left\lfloor 1 \div \textit{source1}[k]^{|\textit{source2}[k]|} \right\rfloor$
    else if $\textit{source2}[k] = 0$ then
      $\textit{dest}[k] \leftarrow 1$
    else
    $\textit{dest}[k] \leftarrow (\textit{source1}[k])^{\textit{source2}[k]}$
    if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
    else $\textit{overflow-flag}[k] \leftarrow 0$
```

The *source1* field (the base) is raised to the power *source2* (the exponent). If the exponent is negative, the result is the truncation of the reciprocal of *source1* raised to the absolute value of *source2*. 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* and *test-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. If, in any particular processor, an attempt is made to raise zero to a negative power, the test flag in that processor is set.

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-U-POWER

Raises an unsigned integer to an unsigned integer power.

Formats

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

Operands

dest The unsigned integer destination field.

source1 The unsigned integer base field.

source2 The unsigned integer exponent field.

source2-value An unsigned integer immediate operand to be used as the second source.

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

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

slen1 For CM:u-u-power-3-3L and CM:u-u-power-constant-3-2L, the length of the *source1* field. This must be non-negative and no greater than CM:*maximum-integer-length*.

slen2 For CM:u-u-power-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 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 $\textit{current-vp-set}$ do
   if $\textit{context-flag}[k] = 1$ then
      if $\textit{source2}[k] = 0$ then
         $\textit{dest}[k] \leftarrow 1$
      else
      $\textit{dest}[k] \leftarrow (\textit{source1}[k])^{\textit{source2}[k]}$
      if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
      else $\textit{overflow-flag}[k] \leftarrow 0$
```

The *source1* field (the base) is raised to the power *source2* (the exponent). 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 *source2-value* should be an unsigned 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.*

C-READ-FROM-PROCESSOR

Reads the source field of a single specified processor as a complex number and returns it to the front end.

Formats result ← CM:c-read-from-processor-1L *send-address-value*, *source*, *len*

Operands *send-address-value* An immediate operand, the send address of a single particular processor.

source The complex source field.

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

Result A complex number, the contents of the *source* field in the specified virtual processor.

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

Definition Return *source*[*send-address-value*] to front end

The *source* field of the processor whose send address is the immediate operand *send-address-value* is read and returned as a floating-point number to the front end.

C-RECIPROCAL

Calculates the reciprocal of a complex number.

Formats CM:c-reciprocal-1-1L *dest/source*, *s*, *e*
CM:c-reciprocal-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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.

test-flag is set if divistion by zero 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 $\textit{context-flag}[k] = 1$ then
        $\textit{dest}[k] \leftarrow \frac{1}{\textit{source}[k]}$
```

A reciprocal of the complex *source* field is place in the complex *dest* field.

REDUCE-WITH-C-ADD

Within each scan class one particular processor (if it is selected) receives the sum of the complex source fields from all the selected processors in that scan class.

Formats CM:reduce-with-c-add-1L *dest*, *source*, *axis*, *s*, *e*, *to-coordinate*

Operands

dest The complex destination field.

source The complex source field.

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

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

to-coordinate An unsigned integer immediate operand to be used as the NEWS coordinate along *axis* indicating which element of the scan class, if any, is to receive the result.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{current-vp-set}$ do
    if $\textit{context-flag}[k] = 1$ then
        let $g = \textit{geometry}(\textit{current-vp-set})$
        let $C_k = \textit{scan-subclass}(g, k, \textit{axis})$
        if $\textit{extract-news-coordinate}(g, \textit{axis}, k) = \textit{to-coordinate}$ then
            $\textit{dest}[k] \leftarrow \left( \sum_{m \in C_k} \textit{source}[m] \right)$
```

where *scan-subclass* is as defined on page 36 of the *Paris Reference Manual*.

See section 5.16 beginning on page 34 for a general description of reduce operations. The CM:reduce-with-c-add operation combines *source* fields by performing complex addition.

The operation CM:reduce-with-c-add-1L differs from CM:spread-with-c-add-1L only in that the result is stored in (at most) one processor of the scan class rather than in all selected processors of the scan class.

REMOVE-FIELD-ALIAS

Removes the specified alias field-id from the field to which it refers, leaving the field intact.

Formats CM:remove-field-alias *alias-id*

Operands *alias-id* An alias field-id. This must be an alias field-id returned by CM:make-field-alias.

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

Removing an alias field-id does not affect the memory field to which it refers.

F-F-ROUND

Rounds each source field value to the nearest integer value and stores the result as a floating-point number in the destination field.

Formats

```
CM:f-f-round-1-1L   dest/source, s, e
CM:f-f-round-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] ← sign(source) × round(source[k])
```

The *source* field, treated as a floating-point number, is rounded to the nearest intege and the result is stored in the *dest* field as a floating-point number.

If the *source* field value is exactly midway between two integers, then it is rounded to the even integer.

SCAN-WITH-C-ADD

The destination field in every selected processor receives the sum of the complex source fields from processors below or above it in some ordering of the processors.

Formats CM:scan-with-c-add-1L *dest*, *source*, *axis*, *s*, *e*, *direction*, *inclusion*, *smode*, *sbit*

Operands

dest The complex destination field.

source The complex source field.

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

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

direction Either :upward or :downward.

inclusion Either :exclusive or :inclusive.

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

sbit The segment bit or start bit (a one-bit 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 $current\text{-}vp\text{-}set$ do
  if $context\text{-}flag[k] = 1$ then
    let $g = geometry(current\text{-}vp\text{-}set)$
    let $S_k = scan\text{-}subset(g, k, axis, direction, inclusion, smode, sbit)$
    if $|S_k| = 0$ then
      $dest[k] \leftarrow 0$
    else
      $dest[k] \leftarrow \left(\sum_{m \in S_k} source[m]\right)$
```

where *scan-subset* is as defined on page 36 of the *Paris Reference Manual.*

See the section beginning on 34 for a general description of scan operations and the effect of the *axis*, *direction*, *inclusion*, *smode*, and *sbit* operands.

The CM:scan-with-c-add operation combines *source* fields by performing complex addition. If the scan subset for a selected processor is empty, then the complex value +0.0 is stored in the *dest* field for that processor. Note that this can occur only when the *inclusion* argument is :exclusive.

SEND-WITH-C-ADD

Sends a message from every selected processor to a destination processor. Each selected processor may specify any processor as the destination, including itself. A destination processor may receive messages even if 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 complex addition.

Formats CM:send-with-c-add-1L *dest*, *send-address*, *source*, *s*, *e*, *notify*

Operands *dest* The complex destination field.

send-address The field containing a send-address that indicates which processor is to receive the message.

source The complex 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 $2(s + e + 1)$.

notify The notification bit (a one-bit field).

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 all processors regardless of the value of the *context-flag*.

Definition

```
Let $P = \{ m \mid 0 \le m \le \text{CM:*user-send-address-limit*} \}$
For every virtual processor $k$ in $\textit{vp-set}(\textit{dest})$ do
  let $S_k = \{ m \mid m \in P \wedge \textit{context-flag}[m] = 1 \wedge \textit{send-address}[m] = k \}$
  if $|S_k| = 0$ then
    if $\textit{notify}[k] \not\equiv \text{CM:*no-field*}$ then $\textit{notify}[k] \leftarrow 0$
  else
    if $\textit{notify}[k] \not\equiv \text{CM:*no-field*}$ then $\textit{notify}[k] \leftarrow 1$
    $\textit{dest}[k] \leftarrow \textit{dest}[k] + \left( \sum_{m \in S_k} \textit{source}[m] \right)$
```

For every selected processor p_s, a message *length* bits long is sent from that processor to the processor p_d whose absolute 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 the *dest* field within processor p_d.

The CM:send-with-c-add operation adds incoming messages to the *dest* field, treating all quantities as complex numbers. To receive the sum of only the messages, the destination area should initially be set to zero in all processors that might receive a message.

SET-FIELD-ALIAS-VP-SET

Sets the VP set of the specified alias field-id to the specified VP set.

Formats CM:set-field-alias-vp-set *alias-id*, *vp-set*

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

vp-set A vp-set-id. This need not be the current VP set.

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

This function sets the VP set of *alias-field* to *vp-set*.

An error is signaled if the physical length of the aliased field is not exactly divisible by the VP ratio of *vp-set*. (See the definitions of CM:make-field-alias for more information about the physical length of an aliased field.)

C-C-SIGNUM

The signum of the complex source field is stored in the complex destination field.

Formats

CM:c-c-signum-1-1L *dest/source*, *s*, *e*
CM:c-c-signum-2-1L *dest*, *source*, *s*, *e*

Operands

dest The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap

The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{current-vp-set}$ do
  if $\textit{context-flag}[k] = 1$ then
    $\textit{dest}[k] \leftarrow \textit{signum}(\textit{source}[k])$
```

The signum of a complex number is a complex number of the same phase but with unit magnitude, unless the numer is a complex zero, in which case the result is a complex zero.

C-SIN

The sine of the complex source field is placed in the complex destination field.

Formats CM:c-sin-1-1L *dest*/*source*, *s*, *e*
CM:c-sin-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \sin source[k]$
    if $\langle$overflow occurred in processor $k\rangle$ then $overflow\text{-}flag[k] \leftarrow 1$
```

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

C-SINH

The hyperbolic sine of the complex source field is placed in the complex destination field.

Formats

CM:c-sinh-1-1L	*dest/source*, *s*, *e*
CM:c-sinh-2-1L	*dest*, *source*, *s*, *e*

Operands

dest	The complex destination field.
source	The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\mathit{current\text{-}vp\text{-}set}$ do
  if $\mathit{context\text{-}flag}[k] = 1$ then
    $\mathit{dest}[k] \leftarrow \sinh \mathit{source}[k]$
```

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

SPREAD-WITH-C-ADD

The destination field in every selected processor receives the sum of the complex source fields from processors below or above it in some ordering of the processors.

Formats CM:spread-with-c-add-1L *dest*, *source*, *axis*, *s*, *e*

Operands	*dest*	The complex destination field.
	source	The complex source field.
	axis	An unsigned integer immediate operand to be used as the the number of a NEWS axis.
	s, *e*	The significand and exponent lengths for the *dest* and *source* fields. The total length of an operand in this format is $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\mathit{current\text{-}vp\text{-}set}$ do
   if $\mathit{context\text{-}flag}[k] = 1$ then
      let $C_k = \mathit{scan\text{-}subclass}(k, \{\, \mathit{axis} \,\})$
      $\mathit{dest}[k] \leftarrow \left( \sum_{m \in C_k} \mathit{source}[m] \right)$
where scan-subclass is as defined on page 36 of the Paris Reference Manual.
```

See the section beginning on page 36 for a general description of spread operations. The CM:spread-with-c-add operation combines *source* fields by performing complex addition.

A call to CM:spread-with-c-add-1L is equivalent to the sequence

CM:scan-with-c-add-1L *dest*, *source*, *axis*, *s*, *e*, :upward, :inclusive, :none, *dont-care*
CM:scan-with-copy-1L *dest*, *source*, *axis*, $2 \times (s + e + 1)$, :downward, :inclusive, :none, *dont-care*

but may be faster.

C-SQRT

Calculates the square root of the complex source field and places it in the complex destination field.

Formats CM:c-sqrt-1-1L *dest/source*, *s*, *e*
CM:c-sqrt-2-1L *dest*, *source*, *s*, *e*

Operands *dest* The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $context\text{-}flag[k] = 1$ then
    $dest[k] \leftarrow \sqrt{source}$
```

In each selected processor, the square root of the *source* field value is placed in the *dest* field.

F-SUBF-CONST-MULT

Calculates a value $(b - a)x$ and places it in the destination.

Formats

CM:f-subf-const-mult-1L *dest*, *source1*, *source2-value*, *source3*, *s*, *e*
CM:f-subf-const-mult-const-1L *dest*, *source1*, *source2-value*, *source3-value*, *s*, *e*

Operands

dest The floating-point destination field.

source1 The floating-point first source (subtrahend) field.

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

source3 The floating-point third source (multiplier) field.

source3-value A floating-point immediate operand to be used as the third source (multiplier).

s, *e* The significand and exponent lengths for the *dest*, *source1*, *source2*, and *source3* fields. The total length of an operand in this format is $s + e + 1$.

Overlap The fields *source1* and *source3* 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 *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 $context\text{-}flag[k] = 1$ then
        $dest[k] \leftarrow (source2\text{-}value[k] - source1[k]) \times source3[k]$
    if ⟨overflow occurred in processor $k$⟩ then $overflow\text{-}flag[k] \leftarrow 1$
```

The operand *source1* is subtracted from *source2-value*, treating them as floating-point numbers, and then the difference is multiplied by a third operand *source3*. The result is stored into memory. The various operand formats allow operands to be either memory fields or constants.

The constant operands *source2-value* and *source3-value* should be double-precision front-end values (in Lisp, automatic coercion is performed if necessary). The constants are then converted, in effect, to the format specified by *s* and *e* before the operation is performed.

A call to CM:f-subf-const-mult-1L is equivalent to the sequence

CM:f-subfrom-constant-3-1L *dest*, *source1*, *source2-value*, *s*, *e*
CM:f-multiply-3-1L *dest*, *dest*, *source3*, *s*, *e*

but may be faster.

C-SUBTRACT

The difference of two complex source values is placed in the destination field.

Formats

CM:c-subtract-2-1L	*dest*/*source1*, *source2*, *s*, *e*
CM:c-subtract-always-2-1L	*dest*/*source1*, *source2*, *s*, *e*
CM:c-subtract-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-subtract-always-3-1L	*dest*, *source1*, *source2*, *s*, *e*
CM:c-subtract-constant-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
CM:c-subtract-const-always-2-1L	*dest*/*source1*, *source2-value*, *s*, *e*
CM:c-subtract-constant-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
CM:c-subtract-const-always-3-1L	*dest*, *source1*, *source2-value*, *s*, *e*
CM:c-subfrom-2-1L	*dest*/*source2*, *source1*, *s*, *e*
CM:c-subfrom-always-2-1L	*dest*/*source2*, *source1*, *s*, *e*
CM:c-subfrom-constant-2-1L	*dest*/*source2*, *source1-value*, *s*, *e*
CM:c-subfrom-const-always-2-1L	*dest*/*source2*, *source1-value*, *s*, *e*
CM:c-subfrom-constant-3-1L	*dest*, *source2*, *source1-value*, *s*, *e*
CM:c-subfrom-const-always-3-1L	*dest*, *source2*, *source1-value*, *s*, *e*

Operands

dest The complex destination field. This is the difference, the result of the subtraction operation.

source1 The complex first source field. This is the minuend.

source2 The complex second source field. This is the subtrahend.

source1-value A complex immediate operand to be used as the first source.

source2-value A complex 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 $2(s + e + 1)$.

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 complex fields are identical if they have the same address and the same format. It is permissible for all the fields to be identical.

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*$[k] \leftarrow$ *source1*$[k] -$ *source2*$[k]$
        if ⟨overflow occurred in processor $k$⟩ then *overflow-flag*$[k] \leftarrow 1$
```

The operand *source2* is subtracted from *source1*, treated as as complex numbers. The result is stored into the 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 "subfrom" operations allow for the destination to be subtracted from the other operand, or for a memory field to be subtracted from an immediate value.

The constant operand *source1-value* or *source2-value* should be a double-precision complex front-end value (in Lisp, automatic coercion is performed if necessary). Before the operation is performed, the constant is converted, in effect, to the format specified by *s* and *e*.

C-TAN

Calculates the complex tangent of the source field values and stores the result in the complex destination field.

Formats CM:c-tan-1-1L *dest/source*, *s*, *e*
CM:c-tan-2-1L *dest*, *source*, *s*, *e*

Operands

dest The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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 $\textit{context-flag}[k] = 1$ then
      $\textit{dest}[k] \leftarrow \tan \textit{source}[k]$
      if ⟨overflow occurred in processor $k$⟩ then $\textit{overflow-flag}[k] \leftarrow 1$
```

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

C-TANH

Calculates the complex hyperbolic tangent of the source field values and stores the result in the complex destination field.

Formats

```
CM:c-tanh-1-1L  dest/source, s, e
CM:c-tanh-2-1L  dest, source, s, e
```

Operands

dest The complex destination field.

source The complex 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 $2(s + e + 1)$.

Overlap The *source* field must be either disjoint from or identical to the *dest* field. Two complex 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] ← tanh source
```

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

F-VAX-TO-IEEE

Converts the floating-point source field values from VAX floating-point format to IEEE floating-point format and stores the result in the destination field.

Formats CM:f-vax-to-ieee-1L *ieee-dest*, *vax-source*, *len*

Operands *ieee-dest* The floating-point destination field.

vax-source The floating-point source field.

len The length of the *vax-source* and *ieee-dest* fields. The value of *len* must be either 32 or 64.

Overlap The fields *ieee-dest* and *vax-source* may overlap in any manner.

Flags *overflow-flag* is set if the vax-source cannot be represented in the destination field; otherwise it is cleared. If *vax-source* is the VAX "undefined variable", the IEEE destination is set to NaN(all 1's) and the *overflow-flag* is cleared. VAX double precision format uses three more mantissa bits than the IEEE double precision format uses. These bits are simply dropped during the conversion. The *overflow-flag* is always cleared for double-precision conversion.

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

The CM operates internally on floating point data in IEEE format whereas the VAX uses a VAX floating-point format. In each active processor, this function converts a floating-point field in VAX format to a field in standard IEEE format.

The value of *len* specifies the precision of *vax-source*. If *len* is specified as 32, then VAX 'F' format is used. If *len* is specified as 64, then VAX 'D' format is used.

VAX and IEEE floating-point formats are incompatible, so there are a number of potential inaccuracies in the translation. These are described in the flags description above.

This instruction is useful for rapidly converting floating-point data from VAX to IEEE format. For example, if data is transferred from a VAX to a file in the CM file system, CM:f-vax-to-ieee-1L should be called after reading the data file.

All Paris front end to CM data transfer functions automatically convert the data from the front-end format appropriately so it is not necessary to call CM:vax-to-ieee before calling, for instance, one of the write-to-news-array instructions.

To convert data back to VAX floating-point format, see the definition of CM:f-ieee-to-vax-1L.

C-WRITE-TO-PROCESSOR

Stores an immediate complex number operand value into the destination field of a single specified processor.

Formats CM:c-write-to-processor-1L *send-address-value*, *dest*, *source-value*, *len*

Operands *send-address-value* An immediate operand, the send address of a single particular processor.

dest The complex destination field.

source-value A complex immediate operand to be used as the source.

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

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

Definition $dest[send\text{-}address\text{-}value] \leftarrow source\text{-}value$

The specified *source-value*, a complex number, is stored into the *dest* field of the processor whose send address is the immediate operand *send-address-value*.

The constant operand *source-value* should be a double-precision front-end value (in Lisp, automatic coercion is performed if necessary).

U-F-TRUNCATE

Rounds each source field value to the largest integer not greater than that value and [illegible] 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 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
        [illegible]
```

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.