Software Development Kit for Multicore Acceleration
Version 3.0

IBM

SIMD Math Library Specifications for CBE Version 2.1

SC33-8335-00

Software Development Kit for Multicore Acceleration
Version 3.0

SIMD Math Library Specifications for CBE Version 2.1

SC33-8335-00

Note

Before using this information and the product it supports, read the information in “Notices” on page 319.

First Edition

This edition applies to the SIMD Library Version 2.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. SIMD Math Library man pages

The SIMD Math Library contains a set of functions which extend the common mathematical functions to operate on vectors.

This book contains the SIMD Math Library man pages in printable form, including graphical images of the equations which cannot be rendered in man page format.

Overview

The traditional mathematical functions specified by standards such as ISO/IEC 9899:1999 (more commonly known as the ″C99 standard″) are defined in terms of scalar instructions and do not take advantage of the powerful Single Instruction, Multiple Data (SIMD) vector instructions provided by both the PPU and SPU instruction sets of the Cell BE architecture.

The SIMD Math library provides short vector versions of a subset of the traditional mathematical functions. (See the Mathematical Acceleration Subsystem (MASS) library for long vector versions.) These vector versions conform as closely as possible to the specifications set out by the scalar standards. However, fundamental differences between scalar architectures and the Cell BE architecture require some deviations, including the handling of rounding, error conditions, floating-point exceptions and special operands such as `NaN` and infinities.

The SIMD Math library is provided in the SDK as both a linkable library archive and as a set of inline function headers. The function names are differentiated from their scalar counterparts by appending a vector type suffix to the standard scalar function name. For example, the SIMD version of `fabs()` which acts on a vector float is called `fabsf4()`, and the version which acts on a vector double is called `fabsd2()`. Inline versions of functions are prefixed with an underscore character ′_′, so for example the inline version of `fabsf4()` is called `_fabsf4()`.

Both the linkable and inline versions require the inclusion of the primary header file `simdmath.h` and must be linked with the `libsimdmath.a` library. In addition, the inline versions require inclusion of a distinct header file for each function used. For example, to use the inline function `_fabsf4()` the `fabsf4.h` header file must be included in addition to `simdmath.h`. Some classification functions also require definitions from the `math.h` header file.

The linkable library archive is more convenient to code as it only requires the inclusion of a single header file, but it produces slower, larger binaries due to the branching instructions necessary for function calls, and also due to limitations of the linker. The inline functions require extra header files to be included for each math function used, but produce faster and smaller (unless inlined multiple times) binaries, because the compiler is able to reduce branching and often achieves better dual-issue rates and optimization.

For the PPU the SIMD Math library header file `simdmath.h` is located in the `/usr/include` directory, with the inline headers located in the `/usr/include/simdmath` directory and the library `libsimdmath.a` located in the `/usr/lib` directory.

For the SPU the header file is located in the `/usr/spu/include` directory, with inline headers in the `/usr/spu/include/simdmath` directory and the library located in the `/usr/spu/lib` directory.

Organisation

The SIMD Math functions are grouped into sections as follows:

1. Absolute value and sign functions
   (Remove or extract the signs from values.)
2. Classification and comparison functions
   (Return boolean values from comparison or classification of elements.)
3. Divide, multiply, modulus, remainder and reciprocal functions
   (Standard arithmetic operations.)
4. Exponentiation, Root, and Logarithmic functions
   (Functions related to exponentiation or the inverse.)
5. Gamma and Error functions
   (Probability functions.)
6. Minimum and Maximum functions
   (Return the larger, smaller or absolute difference between elements.)
7. Rounding and next functions
   (Convert floating point values to integers.)
8. Trigonometric functions
   (sin, cos, tan and their inverses.)
9. Hyperbolic functions
   (sinh, cosh, tanh and their inverses.)

Chapter 2. Absolute value and sign functions

Functions included:

absi4

NAME

absi4 - return the absolute values of integer values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed int absi4(vector signed int *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath/absi4.h>
vector signed int _absi4(vector signed int *x*);

Parameters:

x	input vector

DESCRIPTION

The **absi4** function returns a vector containing the absolute values of the elements of the input vector.

RETURN VALUE

The **absi4** function returns a signed int vector in which each element is defined as the absolute value of the corresponding element of the input vector.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine™ Architecture.

NOTES

Basis:

ISO9899 (C99) **abs** function

SEE ALSO

abs(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

fabsf4

NAME

fabsf4 - return the absolute values of floating values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fabsf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fabsf4.h>
vector float _fabsf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **fabsf4** function returns a vector containing the absolute values of the elements of the input vector.

RETURN VALUE

The function **fabsf4** returns a float vector in which each element is defined as the absolute value of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis:

ISO9899 (C99) **fabs** function

SEE ALSO

abs(3), absi4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

fabsd2

NAME

fabsd2 - return the absolute values of double values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fabsd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fabsd2.h>
vector double _fabsd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **fabsd2** function returns a vector containing the absolute values of the elements of the input vector.

RETURN VALUE

The function **fabsd2** returns a double vector in which each element is defined as the absolute value of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis:

JSRE, ISO9899 (C99) **fabs** function

SEE ALSO

abs(3), absi4(3), fabsf4(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

llabsi2

NAME

llabsi2 - return the absolute values of long long values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed long long llabsi2(vector signed long long *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <llabsi.h>
vector signed long long _llabsi2(vector signed long long *x***);**

Parameters

x	input vector

DESCRIPTION

The **llabsi2** function returns a vector containing the absolute values of the elements of the input vector.

RETURN VALUE

The **llabsi2** function returns a long long vector in which each element is defined as the absolute value of the corresponding element of the input vector.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis:

ISO9899 (C99) **abs** function

SEE ALSO

abs(3), absi4(3), fabsf4(3), fabsd2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

signbitf4

NAME

signbitf4 - return indicators of the signs of floating values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int signbitf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <signbitf4.h>
vector unsigned int _signbitf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **signbitf4** function returns a vector in which elements contain all ones or zeros, depending on the sign of the corresponding input vector element.

Note that the **signbitf4** function is not logically equivalent to (x < 0.0). IEEE 754 floating point rules include a signed zero, so if the input value is -0.0 **signbitf4** will return non-zero even though the naïve implementation will not.

RETURN VALUE

The function **signbitf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the sign bit is set for the corresponding element of *x*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **signbit** macros.

SEE ALSO

signbit(3), signbitd2(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

signbitd2

NAME

signbitd2 - return indicators of the signs of double values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long signbitd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <signbitd2.h>
vector unsigned long long _signbitd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **signbitd2** function returns a vector in which elements contain all ones or zeros, depending on the sign of the corresponding input vector element.

Note that the **signbitd2** function is not logically equivalent to (x < 0.0). IEEE 754 floating point rules include a signed zero, so if the input value is -0.0 **signbitd2** will return non-zero even though the naïve implementation will not.

RETURN VALUE

The function **signbitd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the sign bit is set for the corresponding element of *x*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **signbit** macros.

SEE ALSO

signbit(3), signbitf4(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), copysign(3), copysignf4(3), copysignd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

copysignf4

NAME

copysignf4 - copy floating element signs from one vector to another

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float copysignf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <copysignf4.h>
vector float _copysignf4(vector float *x*, vector float *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **copysignf4** function returns a copy of the vector *x* with the sign bits replaced by those from *y*.

RETURN VALUE

The function **copysignf4** returns a float vector in which each element is defined as the magnitude of the corresponding element of *x* with the sign of the corresponding element of *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **copysign** functions.

SEE ALSO

copysign(3), copysignd2(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

copysignd2

NAME

copysignd2 - copy double element signs from one vector to another

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double copysignd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <copysignd2.h>
vector double _copysignd2(vector double *x*, vector double *y*);

Parameters

x,*y*	input vectors

DESCRIPTION

The **copysignd2** function returns a copy of the vector *x* with the sign bits replaced by those from *y*.

RETURN VALUE

The function **copysignd2** returns a double vector in which each element is defined as the magnitude of the corresponding element of *x* with the sign of the corresponding element of *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **copysign** functions.

SEE ALSO

copysign(3), copysignf4(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), negate(3), negatef4(3), negated2(3), negatei4(3), negatell2(3)

negatef4

NAME

negatef4 - invert the signs of float values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float negatef4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <negatef4.h>
vector float _negatef4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **negatef4** function returns a vector of the corresponding elements of *x* in which each element has its sign negated.

RETURN VALUE

The function **negatef4** returns a float vector in which each element is defined as the negation of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **negate** functions.

SEE ALSO

negate(3), negated2(3), negatei4(3), negatell2(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3)

negated2

NAME

negated2 - invert the signs of double values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double negated2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <negated2.h>
vector double _negated2(vector double *x*);

Parameter	
x	input vector

DESCRIPTION

The **negated2** function returns a vector of the corresponding elements of *x* in which each element has its sign inverted.

RETURN VALUE

The function **negated2** returns a double vector in which each element is defined as the negation of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **negate** functions.

SEE ALSO

negate(3), negatef4(3), negatei4(3), negatell2(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3)

negatei4

NAME

negatei4 - invert the signs of integer values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed int negatei4(vector signed int *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <negatei4.h>
vector signed int _negatei4(vector signed int *x*);

Parameters	
x	input vector

DESCRIPTION

The **negatei4** function returns a vector of the corresponding elements of *x* in which each element has its sign inverted.

RETURN VALUE

The function **negatei4** returns a signed int vector in which each element is defined as the negation of the corresponding element of *x*.

If an element cannot be represented the corresponding result is undefined; no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **negate** functions.

SEE ALSO

negate(3), negatef4(3), negated2(3), negatell2(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3)

negatell2

NAME

negatell2 - invert the signs of long long values

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed long long negatell2(vector signed long long *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <negatell2.h>
vector signed long long _negatell2(vector signed long long *x*);

Parameters	
x	input vector

DESCRIPTION

The **negatell2** function returns a vector of the corresponding elements of *x* in which each element has its sign inverted.

RETURN VALUE

The function **negatell2** returns a signed long long vector in which each element is defined as the negation of the corresponding element of *x*.

If an element cannot be represented the corresponding result is undefined, no error is reported.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **negate** functions.

SEE ALSO

negate(3), negatef4(3), negated2(3), negatei4(3), abs(3), absi4(3), fabsf4(3), fabsd2(3), llabsi2(3), signbit(3), signbitf4(3), signbitd2(3), copysign(3), copysignf4(3), copysignd2(3)

Chapter 3. Classification and comparison functions

Functions included:

fpclassifyf4

NAME

fpclassifyf4 - return special values of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
#include <math.h>
vector signed int fpclassifyf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fpclassifyf4.h>
vector signed int _fpclassifyf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **fpclassifyf4** function returns a vector in which each element contains the floating point classification for the corresponding element of *x*.

RETURN VALUE

The function **fpclassifyf4** returns a signed int vector in which each element is defined as:

FP_NAN	if the element of *x* is not a number (PPU only).
FP_INFINITE	if the element of *x* is infinite (PPU only).
FP_SUBNORMAL	if the element of *x* is subnormal.
FP_ZERO	if the element of *x* is zero.
FP_NORMAL	otherwise.

These classifications are defined in `math.h`.

ENVIRONMENT

PPU and SPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fpclassify** macro.

SEE ALSO

classify(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

fpclassifyd2

NAME

fpclassifyd2 - return special values of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
#include <math.h>
vector signed long long fpclassifyd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fpclassifyd2.h>
vector signed long long _fpclassifyd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **fpclassifyd2** function returns a vector in which each element contains the floating point classification for the corresponding element of *x*.

RETURN VALUE

The function **fpclassifyd2** returns a signed long long vector in which each element is defined as:

FP_NAN	if the element of *x* is not a number.
FP_INFINITE	if the element of *x* is infinite.
FP_SUBNORMAL	if the element of *x* is subnormal.
FP_ZERO	if the element of *x* is zero.
FP_NORMAL	otherwise.

These classifications are defined in `math.h`.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fpclassify** macro.

SEE ALSO

classify(3), fpclassifyf4(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3) , islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3) , isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isequalf4

NAME

isequalf4 - verify if float elements are equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isequalf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isequalf4.h>
vector unsigned int _isequalf4(vector float *x*, vector float *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **isequalf4** function returns a vector in which each element indicates if the corresponding elements of *x* and *y* are equal. This function correctly compares subnormal numbers.

Special Cases:

NaNs always compare as unequal.

zeros compare as equal regardless of sign.

infinities compare as equal if they have the same sign.

RETURN VALUE

The function **isequalf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the elements of *x* and *y* are equal.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

SEE ALSO

isequal(3), isequald2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isequald2

NAME

isequald2 - verify if double elements are equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isequald2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isequald2.h>
vector unsigned long long _isequald2(vector double *x*, vector double *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **isequald2** function returns a vector in which each element indicates if the corresponding elements of *x* and *y* are equal. These functions correctly compare subnormal numbers.

Special Cases:

NaNs always compare as unequal.

zeros compare as equal regardless of sign.

infinities compare as equal if they have the same sign.

RETURN VALUE

The function **isequald2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the elements of *x* and *y* are equal.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

SEE ALSO

isequal(3), isequalf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isgreaterf4

NAME

isgreaterf4 - verify if float elements are greater

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isgreaterf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isgreaterf4.h>
vector unsigned int _isgreaterf4(vector float *x*, vector float *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **isgreaterf4** function returns a vector in which each element indicates if the corresponding element of *x* is greater than the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

if either element is NaN, the comparison is false.

RETURN VALUE

The function **isgreaterf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is greater than the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isgreater** macros.

SEE ALSO

isgreater(3), isgreaterd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isgreaterd2

NAME

isgreaterd2 - verify if double elements are greater

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isgreaterd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isgreaterd2.h>
vector unsigned long long _isgreaterd2(vector double *x*, vector double *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **isgreaterd2** function returns a vector in which each element indicates if the corresponding element of *x* is greater than the corresponding element of *y*. These functions correctly compare subnormal values.

Special cases:

if either element is NaN, the comparison is false.

RETURN VALUE

The function **isgreaterd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is greater than the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isgreater** macros.

SEE ALSO

isgreater(3), isgreaterf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isgreaterequalf4

NAME

isgreaterequalf4 - verify if float elements are greater or equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isgreaterequalf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isgreaterequalf4.h>
vector unsigned int _isgreaterequalf4(vector float *x*, vector float *y*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **isgreaterequalf4** function returns a vector in which each element indicates if the corresponding element of *x* is greater than or equal to the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

If either element is **NaN** the comparison is false.

If both elements are infinite with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **isgreaterequalf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is greater than or equal to the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isgreaterequal** macros.

SEE ALSO

isgreaterequal(3), isgreaterequald2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isgreaterequald2

NAME

isgreaterequald2 - verify if double elements are greater or equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isgreaterequald2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isgreaterequald2.h>
vector unsigned long long _isgreaterequald2(vector double *x*, vector double *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **isgreaterequald2** function returns a vector in which each element indicates if the corresponding element of *x* is greater than or equal to the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

If either element is **NaN** the comparison is false.

If both elements are infinite with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **isgreaterequald2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is greater than or equal to the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isgreaterequal** macros.

SEE ALSO

isgreaterequal(3), isgreaterequalf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessf4

NAME

islessf4 - verify if float elements are less

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int islessf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessf4.h>
vector unsigned int _islessf4(vector float *x*, vector float *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **islessf4** function returns a vector in which each element indicates if the corresponding element of *x* is less than the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

If either element is **NaN**, the comparison is false.

RETURN VALUE

The function **islessf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is less than the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isless** macros.

SEE ALSO

isless(3), islessd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessd2

NAME

islessd2 - verify if double elements are less

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long islessd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessd2.h>
vector unsigned long long _islessd2(vector double *x*, vector double *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **islessd2** function returns a vector in which each element indicates if the corresponding element of *x* is less than the corresponding element of *y*. These functions correctly compare subnormal values.

Special cases:

If either element is **NaN**, the comparison is false.

RETURN VALUE

The function **islessd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is less than the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isless** macros.

SEE ALSO

isless(3), islessf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessequalf4

NAME

islessequalf4 - verify if float elements are less or equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int islessequalf4(vector float *x*, **vector float** *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessequalf4.h>
vector unsigned int _islessequalf4(vector float *x*, **vector float** *y*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **islessequalf4** function returns a vector in which each element indicates if the corresponding element of *x* is less than or equal to the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

If either element is **NaN** the comparison is false.

If both elements are infinite with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **islessequalf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is less than or equal to the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **islessequal** macros.

SEE ALSO

islessequal(3), islessequald2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessequald2

NAME

islessequald2 - verify if double elements are less or equal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long islessequald2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessequald2.h>
vector unsigned long long _islessequald2(vector double *x*, vector double *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **islessequald2** function returns a vector in which each element indicates if the corresponding element of *x* is less than or equal to the corresponding element of *y*. This function correctly compares subnormal values.

Special cases:

If either element is **NaN** the comparison is false.

If both elements are infinite with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **islessequald2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is less than or equal to the corresponding element of *y*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **islessequal** macros.

SEE ALSO

islessequal(3), islessequalf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessgreaterf4

NAME

islessgreaterf4 - verify if float elements are less or greater

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int islessgreaterf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessgreaterf4.h>
vector unsigned int _islessgreaterf4(vector float *x*, vector float *y*);

Parameters

x	input vector
y	input vector

DESCRIPTION

The **islessgreaterf4** function returns a vector in which each element indicates if the corresponding element of *x* is less than or greater than the corresponding element of *y*. This function correctly compares subnormal numbers.

Special Cases:

If either element is **NaN** the elements are considered unequal.

If both elements are infinity with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **islessgreaterf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is less than or greater than the element of *y*.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **islessgreater** macros.

SEE ALSO

islessgreater(3), islessgreaterd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

islessgreaterd2

NAME

islessgreaterd2 - verify if double elements are less or greater

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long islessgreaterd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <islessgreaterd2.h>
vector unsigned long long _islessgreaterd2(vector double *x*, vector double *y*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **islessgreaterd2** function returns a vector in which each element indicates if the corresponding element of *x* is less than or greater than the corresponding element of *y*. This function correctly compares subnormal numbers.

Special Cases:

If either element is **NaN** the elements are considered unequal.

If both elements are infinity with the same sign the elements are considered equal.

The values +0 and -0 are considered equal.

RETURN VALUE

The function **islessgreaterd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is less than or greater than the element of *y*.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **islessgreater** macros.

SEE ALSO

islessgreater(3), islessgreaterf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

is0denormf4

NAME

is0denormf4 - verify if float elements are zero or subnormal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int is0denormf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <is0denormf4.h>
vector unsigned int _is0denormf4(vector float *x*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **is0denormf4** function returns a vector in which each element indicates if the corresponding element of *x* is in the set containing denormalized (subnormal) values, +0, and -0.

RETURN VALUE

The function **is0denormf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is either a denormalized value or 0.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

SEE ALSO

is0denorm(3), is0denormd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

is0denormd2

NAME

is0denormd2 - verify if double elements are zero or subnormal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long is0denormd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <is0denormd2.h>
vector unsigned long long _is0denormd2(vector double *x*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **is0denormd2** function returns a vector in which each element indicates if the corresponding element of *x* is in the set containing denormalized (subnormal) values, +0, and -0.

RETURN VALUE

The function **is0denormd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is either a denormalized value or 0.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

SEE ALSO

is0denorm(3), is0denormf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isfinitef4

NAME

isfinitef4 - verify if float elements are finite

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isfinitef4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isfinitef4.h>
vector unsigned int _isfinitef4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **isfinitef4** function returns a vector in which each element indicates if the corresponding element of *x* is finite. Finite elements include 0, subnormals and normals. Infinite elements are **Inf** and **NaN**.

RETURN VALUE

On the SPU single-precision **Inf** and **NaN** values are not representable. Therefore the function **isfinitef4** returns **UINT_MAX** for all input.

On the PPU the function **isfinitef4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is finite.
0	if the element of *x* is **Inf** or **NaN**.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) isfinite macros.

SEE ALSO

isfinite(3), isfinited2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isfinited2

NAME

isfinited2 - verify if double elements are finite

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isfinited2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isfinited2.h>
vector unsigned long long _isfinited2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **isfinited2** function returns a vector in which each element indicates if the corresponding element of *x* is finite. Finite elements include 0, subnormals and normals. Infinite elements are **Inf** and **NaN**.

RETURN VALUE

The function **isfinited2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is finite.
0	if the element of *x* is **Inf** or **NaN**.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) isfinite macros.

SEE ALSO

isfinite(3), isfinitef4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isinff4

NAME

isinff4 - verify if float elements are infinite

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isinff4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isinff4.h>
vector unsigned int _isinff4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **isinff4** function returns a vector in which each element indicates if the corresponding element of *x* is an infinity (positive or negative).

RETURN VALUE

On the SPU single-precision **Inf** values are not representable. Therefore the function **isinff4** returns **0** for all input.

On the PPU the function **isinff4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is either positive or negative infinity.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isinf** macros.

SEE ALSO

isinf(3), isinfd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isinfd2

NAME

isinfd2 - verify if double elements are infinite

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isinfd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isinfd2.h>
vector unsigned long long _isinfd2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **isinfd2** function returns a vector in which each element indicates if the corresponding element of *x* is an infinity (positive or negative).

RETURN VALUE

The function **isinfd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is either positive or negative infinity.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isinf** macros.

SEE ALSO

isinf(3), isinff4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isnanf4

NAME

isnanf4 - verify if float elements are not numbers

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isnanf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isnanf4.h>
vector unsigned int _isnanf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **isnanf4** function returns a vector in which each element indicates if the corresponding element of *x* is **NaN**.

RETURN VALUE

On the SPU single-precision **NaN** values are not representable. Therefore the function **isnanf4** returns **0** for all input.

On the PPU the function **isnanf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is **NaN**.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isnan** macros.

SEE ALSO

isnan(3), isnand2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isnand2

NAME

isnand2 - verify if double elements are not numbers

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isnand2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isnand2.h>
vector unsigned long long _isnand2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **isnand2** function returns a vector in which each element indicates if the corresponding element of *x* is **NaN**.

RETURN VALUE

The function **isnand2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is **NaN**.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isnan** macros.

SEE ALSO

isnan(3), isnanf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnormal(3), isnormalf4(3), isnormald2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isnormalf4

NAME

isnormalf4 - verify if float elements are normal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isnormalf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isnormalf4.h>
vector unsigned int _isnormalf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **isnormalf4** function returns a vector in which each element indicates if the corresponding element of *x* is normal (not subnormal, **Inf** or **NaN**).

RETURN VALUE

On the SPU single-precision **Inf** and **NaN** values are not representable. Therefore the function **isnormalf4** returns **UINT_MAX** for all input.

On the PPU the function **isnormalf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of *x* is normal.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isnormal** macro.

SEE ALSO

isnormal(3), isnormald2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isnormald2

NAME

isnormald2 - verify if double elements are normal

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isnormald2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isnormald2.h>
vector unsigned long long _isnormald2(vector double *x*);

Parameter	
x	input vector

DESCRIPTION

The **isnormald2** function returns a vector in which each element indicates if the corresponding element of *x* is normal (not subnormal, **Inf** or **NaN**).

RETURN VALUE

The function **isnormald2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of *x* is normal.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isnormal** macro.

SEE ALSO

isnormal(3), isnormalf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isunordered(3), isunorderedf4(3), isunorderedd2(3)

isunorderedf4

NAME

isunorderedf4 - verify if float elements are unordered

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned int isunorderedf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isunorderedf4.h>
vector unsigned int _isunorderedf4(vector float *x*, vector float *y*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **isunorderedf4** function returns a vector in which each element indicates if the corresponding element of either *x* or *y* is unordered (**NaN**).

RETURN VALUE

On the SPU single-precision **NaN** values are not representable. Therefore the function **isunorderedf4** returns **0** for all input.

On the PPU the function **isunorderedf4** returns an unsigned int vector in which each element is defined as:

UINT_MAX	if the element of either *x* or *y* is **NaN**.
0	otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isunordered** macros.

SEE ALSO

isunordered(3), isunorderedd2(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3)

isunorderedd2

NAME

isunorderedd2 -verify if double elements are unordered

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector unsigned long long isunorderedd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <isunorderedd2.h>
vector unsigned long long _isunorderedd2(vector double *x*, vector double *y*);

Parameters	
x	input vector
y	input vector

DESCRIPTION

The **isunorderedd2** function returns a vector in which each element indicates if the corresponding element of either *x* or *y* is unordered (**NaN**).

RETURN VALUE

The function **isunorderedd2** returns an unsigned long long vector in which each element is defined as:

ULLONG_MAX	if the element of either *x* or *y* is **NaN**.
0	otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **isunordered** macros.

SEE ALSO

isunordered(3), isunorderedf4(3), classify(3), fpclassifyf4(3), fpclassifyd2(3), isequal(3), isequalf4(3), isequald2(3), isgreater(3), isgreaterf4(3), isgreaterd2(3), isgreaterequal(3), isgreaterequalf4(3), isgreaterequald2(3), isless(3), islessf4(3), islessd2(3), islessequal(3), islessequalf4(3), islessequald2(3), islessgreater(3), islessgreaterf4(3), islessgreaterd2(3), is0denorm(3), is0denormf4(3), is0denormd2(3), isfinite(3), isfinitef4(3), isfinited2(3), isinf(3), isinff4(3), isinfd2(3), isnan(3), isnanf4(3), isnand2(3), isnormal(3), isnormalf4(3), isnormald2(3)

Chapter 4. Divide, multiply, modulus, remainder and reciprocal functions

Functions included:

divf4

NAME

divf4 - return quotients of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float divf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <divf4.h>
vector float _divf4(vector float *x*, vector float *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

This function divides each element of *x* by the corresponding element of *y* and returns a vector of the quotients.

Special Cases:
- If either input is **NaN** the result is **NaN**.
- For **Inf**/**Inf** or **0**/**0** the result is **NaN**.
- For finite/**0** the result is **Inf** with sign = sign(x)/sign(y).
- For finite/**Inf** the result is **0** with sign = sign(x)/sign(y).
- On the SPU division by 0 results in a return of **HUGE_VALF** with sign = sign(x)/sign(y).

RETURN VALUE

The function **divf4** returns a vector containing the quotients produced by dividing each element of *x* by the corresponding element of *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **div** function (**divi4**).

SEE ALSO

div(3), divf4_fast(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

divf4_fast

NAME

divf4_fast - fast return quotients of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float divf4_fast(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <divf4_fast.h>
vector float _divf4_fast(vector float *x*, vector float *y*);

Parameters

x,*y*	input vectors

DESCRIPTION

This functions divides each element of *x* by the corresponding element of *y* and returns a vector of the quotients.

Special Cases:

- If either input is **NaN** the result is **NaN**.
- For **Inf**/**Inf** or **0**/**0** the result is **NaN**.
- For finite/**0** the result is **Inf** with sign = sign(x)/sign(y).
- For finite/**Inf** the result is **0** with sign = sign(x)/sign(y).
- On the SPU division by 0 results in a return of **HUGE_VALF** with sign = sign(x)/sign(y).

RETURN VALUE

The function**divf4_fast** returns a vector containing the quotients produced by dividing each element of *x* by the corresponding element of *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **div** function (**divi4**).

SEE ALSO

div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

divd2

NAME

divd2 - return quotients of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double divd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <divd2.h>
vector double _divd2(vector double *x*, vector double *y*);

Parameters

x,*y*	input vectors

DESCRIPTION

The **divd2** function divides each element of *x* by the corresponding element of *y* and return a vector of the quotients.

Special Cases:
- If either input is **NaN** the result is **NaN**.
- For **Inf**/**Inf** or **0**/**0** the result is **NaN**.
- For finite/**Inf** the result is **0** with sign = sign(x)/sign(y).
- ″For finite/0, the result is **Inf** with sign = sign(*x*)/sign(*y*).″

RETURN VALUE

The function **divd2** returns a vector containing the quotients produced by dividing each element of *x* by the corresponding element of *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **div** function

SEE ALSO

div(3), divf4(3), divf4_fast(3), divi4(3), divu4(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

divi4

NAME

divi4 - return quotients of integer elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
divi4_t divi4(vector signed int *x*, vector signed int *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <divi4.h>
divi4_t _divi4(vector signed int *x*, vector signed int *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **divi4** function divides each element of *x* by the corresponding element of *y* and returns a vector of remainders in a structure (if the quotients can be represented).

Special Cases:
- Division by zero (positive or negative) produces positive zero, without generating an error.
- Negative zero divided by 1 produces zero.

RETURN VALUE

The function **divi4** returns the quotient and remainder in the following structures:

```
typedef struct divi4_t {
vector signed int quot;
vector signed int rem;
} divi4_t;

typedef struct divu4_t {
vector unsigned int quot;
vector unsigned int rem;
} divu4_t;
```

- Each element in the structure member *quot* is the algebraic quotient truncated towards 0.

- Each element in the structure member *rem* is the corresponding remainder, such that $x = quot*y+rem$

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **div** function (**divi4**).

SEE ALSO

divi4_t(3), div(3), divf4(3), divf4_fast(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

lldivi2

NAME

lldivi2 - return quotients of long long elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(lldivi2_t) lldivi2(vector signed long long *x*, **vector signed long long** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <lldivi2.h>
(lldivi2_t) _lldivi2(vector signed long long *x*, **vector signed long long** *y***);**

Parameters

x,y	input vectors

DESCRIPTION

The **lldivi2** function divides each element of *x* by the corresponding element of *y* and returns the quotients in a structure of type `lldivi2_t()`, which contains a vector of quotients *quot* and a vector of remainders *rem*.

Each element of the vector in the structure member *quot* is the algebraic quotient truncated towards zero. Each element of the vector in the structure member *rem* is the corresponding remainder, such that for each element *x* == *quot* * *y* + *rem*. If an element of *y* is zero, then the corresponding element of the resulting quotient is zero.

RETURN VALUE

The function **lldivi2** returns a structure containing vectors of quotients and remainders produced by dividing each element of *x* by the corresponding element of *y*. If an element of *y* is zero then the corresponding elements of the result are zero.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **lldiv** function

SEE ALSO

lldivi2_t(3), div(3), divf4(3), divf4_fast(3), divi4(3), divu4(3), divd2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

divu4

NAME

divu4 - return quotients of unsigned integer elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
divu4_t divu4(vector unsigned int *x*, vector unsigned int *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <divu4.h>
divu4_t _divu4(vector unsigned int *x*, vector unsigned int *y*);

Parameters	
x,y	input vectors

DESCRIPTION

The **divu4** function divides each element of *x* by the corresponding element of *y* and returns a vector of remainders in a structure (if the quotients can be represented).

Special Cases:
- Division by zero (positive or negative) produces positive zero, without generating an error.
- Negative zero divided by 1 produces zero.

RETURN VALUE

The function **divu4** returns the quotient and remainder in the following structures:

```
typedef struct divi4_t {
vector signed int quot;
vector signed int rem;
} divi4_t;

typedef struct divu4_t {
vector unsigned int quot;
vector unsigned int rem;
} divu4_t;
```

- Each element in the structure member *quot* is the algebraic quotient truncated towards 0.

- Each element in the structure member *rem* is the corresponding remainder, such that $x = quot*y+rem$

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **div** function (**divi4**).

SEE ALSO

divu4_t(3), div(3), divf4(3), divf4_fast(3), divi4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

lldivu2

NAME

lldivu2 - return quotients of unsigned long long elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(lldivu2_t) lldivu2(vector unsigned long long *x*, vector unsigned long long *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <_lldivu2.h>
(lldivu2_t) _lldivu2(vector unsigned long long *x*, vector unsigned long long *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **lldivu2** function divides each element of *x* by the corresponding element of *y* and returns the quotients in a structure of type `lldivu2_t()`, which contains a vector of quotients *quot* and a vector of remainders *rem*.

Each element of the vector in the structure member *quot* is the algebraic quotient truncated towards zero. Each element of the vector in the structure member *rem* is the corresponding remainder, such that for each element *x* == *quot* * *y* + *rem*. If an element of *y* is zero, then the corresponding element of the resulting quotient is zero.

RETURN VALUE

The function **lldivu2** returns a structure containing vectors of quotients and remainders produced by dividing each element of *x* by the corresponding element of *y*. If an element of *y* is zero then the corresponding elements of the result are zero.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **lldiv** function

SEE ALSO

lldivu2_t(3), div(3), divf4(3), divf4_fast(3), divi4(3), divu4(3), divd2(3), lldivi2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

fmaf4

NAME

fmaf4 - multiply and add elements of three float vectors

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fmaf4(vector float *x***, vector float** *y***, vector float** *z***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmaf4.h>
vector float _fmaf4(vector float *x***, vector float** *y***, vector float** *z***);**

Parameters	
x,*y*,*z*	input vectors

DESCRIPTION

The **fmaf4** function computes (x * y) + z.

RETURN VALUE

The function **fmaf4** returns a float vector in which each element is defined as (x*y)+z rounded as one ternary operation for each of the corresponding elements of x, y, and z.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fma** functions.

SEE ALSO

fma(3), fmad2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

fmad2

NAME

fmad2 - multiply and add elements of three double vectors

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fmad2(vector double *x*, **vector double** *y*, **vector double** *z***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmad2.h>
vector double _fmad2(vector double *x*, **vector double** *y*, **vector double** *z***);**

Parameters	
x,*y*,*z*	input vectors

DESCRIPTION

The **fmad2** function computes (*x* * *y*) + *z*.

RETURN VALUE

The function **fmad2** returns a double vector in which each element is defined as (*x***y*)+*z* rounded as one ternary operation for each of the corresponding elements of *x*, *y*, and *z*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fma** functions.

SEE ALSO

fma(3), fmaf4(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

modff4

NAME

modff4 - return signed integer and fraction values from float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float modff4(vector float *x*, **vector float** **pint***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <modff4.h>
vector float _modff4(vector float *x*, **vector float** **pint***);**

Parameters

x	input vector
**pint*	pointer to output vector of integers

DESCRIPTION

The **modff4** function determines an integer *i* plus a fraction *frac* that represent the value of each element of *x*. It returns a vector of the values *frac* and stores a vector of the integers *i* in **pint* for each corresponding element of *x*, such that:

- $x = frac + i$,
- $|frac|$ is in the interval [0,1), and
- both *frac* and *i* have the same sign as the element of *x*.

RETURN VALUE

The function **modff4** returns a float vector such that:

- the signed fractional portion of the corresponding element of *x* is returned, and
- the integral portion of each corresponding element of *x* is stored in the vector pointed to by *pint*.

If an element of *y* is zero the corresponding element of the result is undefined.

ENVIRONMENT

PPU and SPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **modf** functions.

SEE ALSO

modf(3), modfd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

modfd2

NAME

modfd2 - return signed integer and fraction values from double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double modfd2(vector double *x*, vector double **pint*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <modfd2.h>
vector double _modfd2(vector double *x*, vector double **pint*);

Parameters	
x	input vector
**pint*	pointer to output vector of integers

DESCRIPTION

The **modfd2** function determines an integer *i* plus a fraction *frac* that represent the value of each element of *x*. It returns a vector of the values *frac* and stores a vector of the integers *i* in **pint* for each corresponding element of *x*, such that:

- $x = frac + i$,
- $|frac|$ is in the interval [0,1), and
- both *frac* and *i* have the same sign as the element of *x*.

RETURN VALUE

The function **modfd2** returns a double vector such that:

- the signed fractional portion of the corresponding element of *x* is returned, and
- the integral portion of each corresponding element of *x* is stored in the vector pointed to by *pint*.

If an element of *y* is zero the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **modf** functions.

SEE ALSO

modf(3), modff4(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

fmodf4

NAME

fmodf4 - return remainders from division of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fmodf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmodf4.h>
vector float _fmodf4(vector float *x*, vector float *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **fmodf4** function computes the remainders of dividing *x* by *y*. The return values are *x* - *n***y*, where *n* are the quotients of *x*/*y*, rounded towards zero.

On the SPU, there are two implementations available:

- **fmodf4** provides computation on all IEEE floating point values (excluding floating overflow or underflow).
- **fmodf4_fast** provides computation on all floating-point *x*/*y* values in the 32-bit signed integer range. Values outside this range get clamped.

RETURN VALUE

The function **fmodf4** returns float vectors in which each element is defined as the remainder of *x*/*y* for the corresponding elements of *x* and *y*.

ENVIRONMENT

fmodf4: SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmod** functions.

SEE ALSO

fmod(3), fmodf4_fast(3), fmodd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

fmodf4_fast

NAME

fmodf4_fast - return approximate remainders from division of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fmodf4_fast(vector float *x***, vector float** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmodf4.h>
vector float _fmodf4_fast(vector float *x***, vector float** *y***);**

Parameters	
x,*y*	input vectors

DESCRIPTION

The **fmodf4** function computes the remainders of dividing *x* by *y*. The return values are *x* - *n***y*, where *n* are the quotients of *x*/*y*, rounded towards zero.

On the SPU, there are two implementations available:
- **fmodf4** provides computation on all IEEE floating point values (excluding floating overflow or underflow).
- **fmodf4_fast** provides computation on all floating-point *x*/*y* values in the 32-bit signed integer range. Values outside this range get clamped.

RETURN VALUE

The functions **fmodf4** and **fmodf4_fast** return float vectors in which each element is defined as the remainder of *x*/*y* for the corresponding elements of *x* and *y*.

ENVIRONMENT

fmodf4_fast: SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmod** functions.

SEE ALSO

fmod(3), fmodf4(3), fmodd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

fmodd2

NAME

fmodd2 - return remainder from division of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fmodd2(vector double *x*, **vector double** *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmodd2.h>
vector double _fmodd2(vector double *x*, **vector double** *y*);

Parameters

x,y	input vectors

DESCRIPTION

The **fmodd2** function computes the remainders of dividing *x* by *y*. The return values are *x* - *n***y*, where *n* are the quotients of *x*/*y*, rounded towards zero.

RETURN VALUE

The function **fmodd2** returns a double vector in which each element is defined as the remainder of *x*/*y* for the corresponding elements of *x* and *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmod** functions.

SEE ALSO

fmod(3), fmodf4(3), fmodf4_fast(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

remainderf4

NAME

remainderf4 - return remainders from division of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float remainderf4(vector float *x*, **vector float** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <remainderf4.h>
vector float _remainderf4(vector float *x*, **vector float** *y***);**

Parameters

x,y	input vectors

DESCRIPTION

The **remainderf4** function computes the remainder x REM y required by IEC 60559: *When y ≠ 0 the remainder r = x REM y is defined regardless of the rounding mode by the mathematical relation r = x - ny, where n is the integer nearest the exact value of x/y; whenever |n - x /y| = 1/2 then n is even. Thus the remainder is always exact. If r = 0 its sign shall be that of x.*

RETURN VALUE

The function **remainderf4** returns a float vector in which each element is defined as the exact remainder of *x*/*y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

IEC 60559

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **remainder** functions, IEC 60559

SEE ALSO

remainder(3), remainderd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

remainderd2

NAME

remainderd2 - return remainders from division of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double remainderd2(vector double *x*, **vector double** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <remainderd2.h>
vector double _remainderd2(vector double *x*, **vector double** *y***);**

Parameters

x,y	input vectors

DESCRIPTION

The **remainderd2** function computes the remainder *x* **REM** *y* required by IEC 60559: *When y ≠ 0 the remainder r = x REM y is defined regardless of the rounding mode by the mathematical relation r = x - ny, where n is the integer nearest the exact value of x/y; whenever |n - x /y| = 1/2 then n is even. Thus the remainder is always exact. If r = 0 its sign shall be that of x.*

RETURN VALUE

The function **remainderd2** returns a double vector in which each element is defined as the exact remainder of x/y.

ENVIRONMENT

SPU only

CONFORMING TO

IEC 60559

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **remainder** functions, IEC 60559

SEE ALSO

remainder(3), remainderf4(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

remquof4

NAME

remquof4 - return remainders and quotients from division of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float remquof4(vector float *x***, vector float** *y***, vector signed int** **pquo***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <remquof4.h>
vector float _remquof4(vector float *x***, vector float** *y***, vector signed int** **pquo***);**

Parameters	
x,y	input vectors
**pquo*	pointer to quotient vector

DESCRIPTION

The **remquof4** function returns the same vector as the corresponding **remainderf4** function. In addition it places into **pquo* a vector of values of which the sign of each is the sign of x/y, and the magnitude of each is the congruent modulo 2^n to the magnitude of the integral quotient of the corresponding element of x/y (where n is an implementation-defined integer greater than or equal to 3).

RETURN VALUE

The function **remquof4** returns a float vector in which each element is defined as the remainder of x/y.

The integral quotient of the corresponding element of x/y mod n is placed in the corresponding element of the vector pointed to by **pquo*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

IEC60559

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **remquo** functions, IEC 60559

SEE ALSO

remquo(3), remquod2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

remquod2

NAME

remquod2 - return remainders and quotients from division of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double remquod2(vector double *x*, **vector double** *y*, **vector signed int** **pquo***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <remquod2.h>
vector double _remquod2(vector double *x*, **vector double** *y*, **vector signed int** **pquo***);**

Parameters

x,y	input vectors
**pquo*	pointer to quotient vector

DESCRIPTION

The **remquod2** function returns the same vector as the corresponding **remainderd2** function. In addition it places into **pquo* a vector of values of which the sign of each is the sign of x/y, and the magnitude of each is the congruent modulo 2^n to the magnitude of the integral quotient of the corresponding element of x/y (where n is an implementation-defined integer greater than or equal to 3).

RETURN VALUE

The function **remquod2** returns a double vector in which each element is defined as the exact remainder of x/y.

The integral quotient of the corresponding element of x/y mod n is placed in the corresponding element of the vector pointed to by **pquo*. The first quotient is placed in slots 0 and 1. The second quotient is placed in slots 2 and 3.

ENVIRONMENT

SPU only

CONFORMING TO

IEC60559

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **remquo** functions, IEC 60559

SEE ALSO

remquo(3), remquof4(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), recip(3), recipf4(3), recipd2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

recipf4

NAME

recipf4 - return reciprocals of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float recipf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <recipf4.h>
vector float _recipf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **recipf4** function returns a vector of the reciprocals of the corresponding elements of *x*. The **recipf4_fast** function provides a faster but less accurate version of **recipf4**.

RETURN VALUE

The function **recipf4** returns a float vector in which each element is defined as:

- the reciprocals of the corresponding element of *x*.
- When an element of *x* is **Inf** the result is **0** with the sign of *x*.
- When an element of *x* is **0**:
  - on the PPU the result is **Inf** with the sign of *x*,
  - on the SPU the result is **HUGE_VAL** with the sign of *x*.
- When an element of *x* is **NaN** the result is **NaN**.

ENVIRONMENT

recipf4: SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **recip** functions.

SEE ALSO

recip(3), recipf4_fast(3), recipd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

recipf4_fast

NAME

recipf4_fast - return approximate reciprocals of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float recipf4_fast(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <recipf4.h>
vector float _recipf4_fast(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **recipf4_fast** function returns a vector of the reciprocals of the corresponding elements of *x*. The **recipf4** function provides a slower but more accurate version of **recipf4_fast**.

RETURN VALUE

The function **recipf4_fast** returns a float vector in which each element is defined as:

- the reciprocals of the corresponding element of *x*.
- When an element of *x* is **Inf** the result is **0** with the sign of *x*.
- When an element of *x* is **0**:
  - on the PPU the result is **Inf** with the sign of *x*,
  - on the SPU the result is **HUGE_VAL** with the sign of *x*.
- When an element of *x* is **NaN** the result is **NaN**.

ENVIRONMENT

recipf4_fast: SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **recip** functions.

SEE ALSO

recip(3), recipd2(3), recipd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

recipd2

NAME

recipd2 - return reciprocals of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double recipd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <recipd2.h>
vector double _recipd2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **recipd2** function returns a vector of the reciprocals of the corresponding elements of *x*.

RETURN VALUE

The function **recipd2** returns a float vector in which each element is defined as:
- the reciprocals of the corresponding element of *x*.
- When an element of *x* is **Inf** the result is **0** with the sign of *x*.
- When an element of *x* is **0** the result is **Inf** with the sign of *x*.
- When an element of *x* is **NaN** the result is **NaN**.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **recip** functions.

SEE ALSO

recip(3), recipf4(3), recipf4_fast(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), rsqrt(3), rsqrtf4(3), rsqrtd2(3)

rsqrtf4

NAME

rsqrtf4 - return the reciprocals of the square roots of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float rsqrtf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <rsqrtf4.h>
vector float _rsqrtf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **rsqrtf4** function returns a vector of the reciprocals of the square roots of the corresponding elements of *x*.

RETURN VALUE

The function **rsqrtf4** returns a float vector in which each element is defined as:

- the reciprocal of the square root of the corresponding element of *x*.
- When an element of *x* is less than **0**:
  - on the PPU the result is**NaN**,
  - on the SPU the result is undefined.
- When an element of *x* is **+Inf** the result is **+0**.
- When an element of *x* is **0** the result is **Inf** with the sign of the corresponding element of *x*.
- When an element of *x* is **NaN** the result is **NaN**.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

SEE ALSO

rsqrt(3), rsqrtd2(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3)

rsqrtd2

NAME

rsqrtd2 - return the reciprocals of the square roots of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double rsqrtd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <rsqrtd2.h>
vector double _rsqrtd2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **rsqrtd2** function returns a vector of the reciprocals of the square roots of the corresponding elements of *x*.

RETURN VALUE

The function **rsqrtd2** returns a float vector in which each element is defined as:
- the reciprocal of the square root of the corresponding element of *x*.
- When an element of *x* is less than **0** the result is**NaN**.
- When an element of *x* is +**Inf** the result is **+0**.
- When an element of *x* is **0** the result is **Inf** with the sign of the corresponding element of *x*.
- When an element of *x* is **NaN** the result is **NaN**.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

SEE ALSO

rsqrt(3), rsqrtf4(3), div(3), divf4(3), divi4(3), divu4(3), divd2(3), lldivi2(3), lldivu2(3), fma(3), fmaf4(3), fmad2(3), modf(3), modff4(3), modfd2(3), fmod(3), fmodf4(3), fmodd2(3), remainder(3), remainderf4(3), remainderd2(3), remquo(3), remquof4(3), remquod2(3), recip(3), recipf4(3), recipd2(3)

Chapter 5. Exponentiation, root, and logarithmic Functions

Functions included:

expf4

NAME

expf4 - return e exponentiated by float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float expf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <expf4.h>
vector float _expf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **expf4** function returns a vector of the exponential e^x for each element in *x*.

RETURN VALUE

The function **expf4** returns a float vector in which each element is defined as:
- ***e*** raised to the power of the corresponding element of *x*.
- On the SPU single-precision element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp** function

SEE ALSO

exp(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

expd2

NAME

expd2 - return e exponentiated by double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double expd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <expd2.h>
vector double _expd2(vector double *x***);**

Parameters	
x	input vector

DESCRIPTION

The **expd2** function returns a vector of the exponential e^x for each element in *x*.

RETURN VALUE

The function **expd2** returns a double vector in which each element is defined as:
- ***e*** raised to the power of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp** function

SEE ALSO

exp(3), expf4(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

exp2f4

NAME

exp2f4 - return 2 exponentiated by float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float exp2f4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <exp2f4.h>
vector float exp2f4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **exp2f4** function returns a vector of the exponential 2^x for each element in *x*.

RETURN VALUE

The function **exp2f4** returns a float vector in which each element is defined as:

- 2 raised to the power of the corresponding element of *x*.
- On the SPU single-precision element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp2** functions

SEE ALSO

exp2(3), exp2d2(3), exp(3), expf4(3), expd2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

exp2d2

NAME

exp2d2 - return 2 exponentiated by double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double exp2d2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <exp2d2.h>
vector double _exp2d2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **exp2d2** function returns a vector of the exponential 2^x for each element in *x*.

RETURN VALUE

The function **exp2d2** returns a double vector in which each element is defined as:
- 2 raised to the power of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp2** functions

SEE ALSO

exp2(3), exp2f4(3), exp(3), expf4(3), expd2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

expm1f4

NAME

expm1f4 - return one less than e exponentiated by float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float expm1f4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <expm1f4.h>
vector float _expm1f4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **expm1f4** function returns a vector of the exponential minus one $e^x - 1$ for each element in *x*.

The purpose of this function is to return mathematically accurate values, even when an element is close to 0 (zero) so the exponent is close to 1 (one) leading to floating-point cancellation errors.

RETURN VALUE

The function **expm1f4** returns a float vector in which each element is defined as:

- one less than ***e*** raised to the power of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp** function

SEE ALSO

expm1(3), expm1d2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

expm1d2

NAME

expm1d2 - return one less than e exponentiated by double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double expm1f4(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <expm1f4.h>
vector double _expm1f4(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **expm1d2** function returns a vector of the exponential minus one $e^x - 1$ for each element in *x*.

The purpose of this function is to return mathematically accurate values, even when an element is close to 0 (zero) so the exponent is close to 1 (one) leading to floating-point cancellation errors.

RETURN VALUE

The function **expm1d2** returns a double vector in which each element is defined as:

- one less than ***e*** raised to the power of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **exp** function

SEE ALSO

expm1(3), expm1f4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

frexpf4

NAME

frexpf4 - return fractions and exponents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float frexpf4(vector float *x*, **vector signed int** **pexp*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <frexpf4.h>
vector float _frexpf4(vector float *x*, **vector signed int** **pexp*);

Parameters	
x	input vector
*pexp	pointer to output vector

DESCRIPTION

The **frexpf4** function is used to split the values of the elements in *x* into a normalized fraction and an exponent. **frexpf4** returns a vector of fractions and a vector of exponent integers in **pexp*.

Each fraction element *frac*, and each exponent integer element *exp*, represent the value of the corresponding element *x*, such that:

- Every element of $|frac|$ is in the interval $\left[\frac{1}{2}, 1\right)$ or is 0.
- $x = frac \times 2^{\exp}$
- If an element of *x* is 0 the corresponding element of **pexp* is also 0.
- If an element of *x* is **NaN** the corresponding element of the result is **NaN** and the corresponding element of **pexp* is undefined.
- If an element of *x* is **Inf** the corresponding element of the result is **Inf** and the corresponding element of **pexp* is undefined.

RETURN VALUE

The function **frexpf4** returns:

- a **float** vector in which each element is defined as the normalized fraction of the corresponding element of *x*, and
- a signed **int*** vector in which each element is defined as the exponent such that 2 raised to this value and multiplied by the normalized fraction is equal to *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **frexp** functions.

SEE ALSO

frexp(3), frexpd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

frexpd2

NAME

frexpd2 - return fractions and exponents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double frexpd2(vector double *x***, vector signed long long** **pexp***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <frexpd2.h>
vector double _frexpd2(vector double *x***, vector signed long long** **pexp***);**

Parameters

x	input vector
*pexp	pointer to output vector

DESCRIPTION

The **frexpd2** function is used to split the values of the elements in *x* into a normalized fraction and an exponent. **frexpd2** returns a vector of fractions and a vector of exponent integers in **pexp*.

Each fraction element *frac*, and each exponent integer element *exp*, represent the value of the corresponding element *x*, such that:

- Every element of $|frac|$ is in the interval $\left[\frac{1}{2},1\right)$ or is 0.
- $x = frac \times 2^{\exp}$
- If an element of *x* is 0 the corresponding element of **pexp* is also 0.

RETURN VALUE

The function **frexpd2** returns:

- a **double** vector in which each element is defined as the normalized fraction of the corresponding element of *x*, and
- a **signed long long*** vector in which each element is defined as the exponent such that 2 raised to this value and multiplied by the normalized fraction is equal to *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **frexp** functions.

SEE ALSO

frexp(3), frexpf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

ldexpf4

NAME

ldexpf4 - return float elements multiplied by an integral power of 2

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float ldexpf4(vector float *x*, **vector signed int** *exp***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ldexpf4.h>
vector float _ldexpf4(vector float *x*, **vector signed int** *exp***);**

Parameters

x	vector of fractional components
exp	vector of exponential components

DESCRIPTION

The **ldexpf4** function returns a vector of $x \times 2^{\text{exp}}$ for the corresponding elements of *x* and *exp*.

RETURN VALUE

The function **ldexpf4** returns a float vector in which each element is defined as:

- $x \times 2^{\text{exp}}$ for the corresponding elements of *x* and *exp*.
- For large elements of *exp* (overflow), the element in the result saturates to **HUGE_VALF** with an appropriate sign.
- For small elements of *exp* (underflow), the corresponding result element is **0**.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ldexp** functions.

SEE ALSO

ldexp(3), ldexpd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

ldexpd2

NAME

ldexpd2 - return double elements multiplied by an integral power of 2

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double ldexpd2(vector double *x*, **vector signed int** *exp***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ldexpd2.h>
vector double _ldexpd2(vector double *x*, **vector signed int** *exp***);**

Parameters

x	vector of fractional components
exp	vector of exponential components

DESCRIPTION

The **ldexpd2** function returns a vector of $x \times 2^{\exp}$ for the corresponding elements of *x* and *exp*.

RETURN VALUE

The function **ldexpd2** returns a double vector in which each element is defined as:

- $x \times 2^{\exp}$ for the corresponding elements of *x* and *exp*.
- For large elements of *exp* (overflow), the element in the result saturates to **HUGE_VALF** with an appropriate sign.
- For small elements of *exp* (underflow), the corresponding result element is **0**.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ldexp** functions.

SEE ALSO

ldexp(3), ldexpf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

powf4

NAME

powf4 - return float elements exponentiated by float elements

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector float powf4(vector float x, vector float y);
Link with -lsimdmath
```

Inline call syntax:

```
#include <simdmath.h>
#include <powf4.h>
vector float _powf4(vector float x, vector float y);
```

Parameters

x	Vector containing base values.
y	Vector containing exponents to be applied to the base values

DESCRIPTION

The **powf4** function returns a vector of x^y for corresponding elements of *x* and *y*.

RETURN VALUE

The function **powf4** returns a float vector in which each element is defined as the corresponding element of *x* raised to the power of the corresponding element of *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **power** functions.

SEE ALSO

pow(3), powd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

powd2

NAME

powd2 - return double elements exponentiated by double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double powd2(vector double *x***, vector double** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <powd2.h>
vector double _powd2(vector double *x***, vector double** *y***);**

Parameters	
x	Vector containing base values.
y	Vector containing exponents to be applied to the base values

DESCRIPTION

The **powd2** function returns a vector of x^y for corresponding elements of *x* and *y*.

RETURN VALUE

The function **powd2** returns a double vector in which each element is defined as the corresponding element of *x* raised to the power of the corresponding element of *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **power** functions.

SEE ALSO

pow(3), powf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

hypotf4

NAME

hypotf4 - return hypotenuse lengths for float catheti

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector float hypotf4(vector float x, vector float y);
```

Link with -lsimdmath

Inline call syntax:

```
#include <simdmath.h>
#include <hypotf4.h>
vector float _hypotf4(vector float x, vector float y);
```

Parameters

x, *y*	Vectors containing the lengths of catheti (sides) from which the hypotenuses are to be calculated.

DESCRIPTION

The **hypotf4** function returns a vector of $\sqrt{x^2 + y^2}$ for corresponding elements of *x* and *y*.

RETURN VALUE

The function **hypotf4** returns a float vector in which each element is defined as the square root of the sum of the squares of the corresponding elements of *x* and *y*, without undue overflow or underflow.

ENVIRONMENT

PPU and SPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **hypot** functions.

SEE ALSO

hypot(3), hypotd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

hypotd2

NAME

hypotd2 - return hypotenuse lengths for double catheti

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector double hypotd2(vector double x, vector double y);
```
Link with -lsimdmath

Inline call syntax:

```
#include <simdmath.h>
#include <hypotd2.h>
vector double _hypotd2(vector double x, vector double y);
```

Parameters

x, *y*	Vectors containing the lengths of catheti (sides) from which the hypotenuses are to be calculated.

DESCRIPTION

The **hypotd2** function returns a vector of $\sqrt{x^2 + y^2}$ for corresponding elements of *x* and *y*, without undue overflow or underflow.

RETURN VALUE

The function **hypotd2** returns a double vector in which each element is defined as the square root of the sum of the squares of the corresponding elements of *x* and *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **hypot** functions.

SEE ALSO

hypot(3), hypotf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

sqrtf4

NAME

sqrtf4 - return accurate square root of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float sqrtf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sqrtf4.h>
vector float _sqrtf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **sqrtf4** function computes the square roots of the elements of the input vectors.

On the SPU this is a fully compliant IEEE implementation guaranteeing the correct truncated result for all valid inputs.

Note: The PPU implementation does not produce IEEE accuracy.

RETURN VALUE

The function **sqrtf4** returns a float vector in which each element is defined as $\sqrt{x}$.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sqrt** functions.

SEE ALSO

sqrt(3), sqrtf4_fast(3), sqrtd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

sqrtf4_fast

NAME

sqrtf4_fast - return approximate square root of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float sqrtf4_fast(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sqrtf4.h>
vector float _sqrtf4_fast(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **sqrtf4_fast** function computes the square roots of the elements of the input vectors.

The values returned are up to 3 ULP (units of least position) off over the input range [1.0,3.99999...]. This is the default implementation and has a histogram of error of:

ULP Error	Count
-3	0
-2	68
0	5985155
1	8611186
2	1752588
3	43324

RETURN VALUE

The function **sqrtf4_fast** returns a float vector in which each element is defined as $\sqrt{x}$.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sqrt** functions.

SEE ALSO

sqrt(3), sqrtf4(3), sqrtd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

sqrtd2

NAME

sqrtd2 - return square root of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double sqrtd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sqrtd2.h>
vector double _sqrtd2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **sqrtd2** function computes the square roots of the elements of the input vectors.

RETURN VALUE

The function **sqrtd2** returns a double vector in which each element is defined as:
$\sqrt{x}$.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sqrt** functions.

SEE ALSO

sqrt(3), sqrtf4(3), exp(3), sqrtf4_fast(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

cbrtf4

NAME

cbrtf4 - return the cube roots of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float cbrtf4(vector float *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <cbrtf4.h>
vector float _cbrtf4(vector float *x***);**

Parameters	
x	input vector

DESCRIPTION

The **cbrtf4** function computes the real cube root of each element in the input vectors.

RETURN VALUE

The function **cbrtf4** returns a float vector in which each element is defined as $\sqrt[3]{x}$

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cbrt** functions.

SEE ALSO

cbrt(3), cbrtd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

cbrtd2

NAME

cbrtd2 - return the cube roots of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double cbrtd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <cbrtd2.h>
vector double _cbrtd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **cbrtd2** function computes the real cube root of each element in their input vectors.

RETURN VALUE

The function **cbrtd2** returns a double vector in which each element is defined as $\sqrt[3]{x}$

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cbrt** functions.

SEE ALSO

cbrt(3), cbrtf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), log(3), logf4(3), logd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

logf4

NAME

logf4 - return base-e (natural) logarithms of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float logf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <logf4.h>
vector float _logf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **logf4** function returns the natural logarithms for each *x*.

RETURN VALUE

The function **logf4** returns a float vector in which each element is defined as:

- the natural logarithm for the corresponding element of *x* if the element is not **0**, or
- **-HUGE_VALF** if the value of the corresponding element of *x* is **0**.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log** functions.

SEE ALSO

log(3), logd2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

logd2

NAME

logd2 - return base-e (natural) logarithms of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double logd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <logd2.h>
vector double _logd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **logd2** function returns the natural logarithms for each *x*.

RETURN VALUE

The function **logd2** returns a double vector in which each element is defined as:
- the natural logarithm for the corresponding element of *x*.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log** functions.

SEE ALSO

log(3), logf4(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log10(3), log2f4(3), log2d2(3), log1p(3), log10f4(3), log10d2(3), logb(3), log1pf4(3), log1pd2(3), ilogb(3), logbf4(3), scalbn(3), ilogbf4(3), ilogbd2(3)

log2f4

NAME

log2f4 - return base-2 logarithms of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float log2f4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <log2f4.h>
vector float _log2f4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **log2f4** function returns a vector of the base 2 logarithms of the corresponding elements of *x*.

RETURN VALUE

The**log2f4** function returns a float vector in which each element is defined as:

- the base 2 logarithm for the corresponding element of *x* if the element is not **0**, or
- **-HUGE_VALF** if the value of the corresponding element of *x* is **0**.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log2** functions.

SEE ALSO

log2(3), log2d2(3), exp(3), expf4(3), expd2(3), exp2(3), exp2f4(3), exp2d2(3), expm1(3), expm1f4(3), expm1d2(3), frexp(3), frexpf4(3), frexpd2(3), ldexp(3), ldexpf4(3), ldexpd2(3), pow(3), powf4(3), powd2(3), hypot(3), hypotd2(3), hypotf4(3), sqrt(3), sqrtf4(3), sqrtd2(3), cbrt(3), cbrtf4(3), cbrtd2(3), log(3), logf4(3), logd2(3), log10(3), log10f4(3), log10d2(3), log1p(3), log1pf4(3), ilog1pd2(3), logb(3), logbf4(3), ilogb(3), ilogbf4(3), ilogbd2(3)

log2d2

NAME

log2d2 - return base-2 logarithms of double elements

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector double log2d2(vector double x);
Link with -lsimdmath
```

Inline call syntax:

```
#include <simdmath.h>
#include <log2d2.h>
vector double _log2d2(vector double x);
```

Parameters	
x	input vector

DESCRIPTION

The **log2d2** function returns a vector of the base 2 logarithms of the corresponding elements of *x*.

RETURN VALUE

The**log2d2** function returns a double vector in which each element is defined as:

- the base 2 logarithm for the corresponding element of *x*.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log2** functions.

SEE ALSO

log2(3), log2f4(3), exp2(3), exp2d2(3), exp2f4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), logb(3), logbf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

log10f4

NAME

log10f4 - return base-10 logarithms of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float log10f4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <log10f4.h>
vector float _log10f4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **log10f4** function returns the base 10 logarithms for each *x*.

RETURN VALUE

The function **log10f4** returns a float vector in which each element is defined as:

- the base 10 logarithm for the corresponding element of *x* if the element is not **0**, or
- **-HUGE_VALF** if the value of the corresponding element of *x* is **0**.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log10** functions.

SEE ALSO

log10(3), log10d2(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log1p(3), log1pd2(3), log1pf4(3), logb(3), logbf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

log10d2

NAME

log10d2 - return base-10 logarithms of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double log10d2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <log10d2.h>
vector double _log10d2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **log10d2** function returns the base 10 logarithms for each *x*.

RETURN VALUE

The function **log10d2** returns a double vector in which each element is defined as:
- the base 10 logarithm for the corresponding element of *x*.
- If an element of *x* is negative, the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log10** functions.

SEE ALSO

log10(3), log10f4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log1p(3), log1pd2(3), log1pf4(3), logb(3), logbf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

log1pf4

NAME

log1pf4 - return the base-e (natural) logarithms of one more than float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float log1pf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <log1pf4.h>
vector float _log1pf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **log1pf4** function returns a vector of the natural logarithms of (1+*x*) for the corresponding element of *x*.

This function returns mathematically accurate values even when the corresponding element of *x* is near 0 because it uses a different algorithm from the **log** function in the open interval (-0.5, 0.5). Outside this range the function defaults to the standard log routine.

RETURN VALUE

The function **log1pf4** returns a float vector in which each element is defined as:

- the natural logarithm of the corresponding element of (1+*x*), if the element is not **0**, or
- **-HUGE_VALF** if the value of the corresponding element of *x* is **0**.
- If an element of *x* is less than 1, the corresponding element of the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log1p** functions.

SEE ALSO

log1p(3), log1pd2(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), logb(3), logbf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

log1pd2

NAME

log1pd2 - return the base-e (natural) logarithms of one more than double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double log1pd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <log1pd2.h>
vector double _log1pd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **log1pd2** function returns a vector of the natural logarithms of (1+*x*) for the corresponding element of *x*.

This function returns mathematically accurate values even when the corresponding element of *x* is near 0 because they use a different algorithm from the **log** function in the open interval (-0.5, 0.5). Outside this range the function defaults to the standard log routine.

RETURN VALUE

The function **log1pd2** returns a double vector in which each element is defined as:

- the natural logarithm for the corresponding element of (1+*x*).
- If an element of *x* is less than 1, the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **log1p** functions.

SEE ALSO

log1p(3), log1pf4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), logb(3), logbf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

logbf4

NAME

logbf4 - return exponents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float logbf4(vector float *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <logbf4.h>
vector float _logbf4(vector float *x***);**

Parameters

x	input vector

DESCRIPTION

The **logbf4** function determines an integer exponent *exp* and a fraction *frac* that represent the value of a finite element of *x*.

RETURN VALUE

The function **logbf4** returns a float vector in which each element is defined as the exponent of the corresponding element of *x* expressed as a floating-point value, such that:

- $x = frac \times \exp^{FLT\_RADIX}$
- $|frac|$ is in the interval [1, FLT_RADIX)
- If an element of *x* is negative, the corresponding element of the result is undefined.

For the **logbf4** function on the SPU:
- if an element of *x* is 0 the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **logb** functions.

SEE ALSO

logb(3), logbd2(3), frexp(3), frexpd2(3), frexpf4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

logbd2

NAME

logbd2 - return exponents of double elements

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector double logbd2(vector double x);
```
Link with -lsimdmath

Inline call syntax:

```
#include <simdmath.h>
#include <logbd2.h>
vector double _logbd2(vector double x);
```

Parameters

x	input vector

DESCRIPTION

The **logbd2** function determines an integer exponent *exp* and a fraction *frac* that represent the value of a finite element of *x*.

RETURN VALUE

The function **logbd2** returns a double vector in which each element is defined as the exponent of the corresponding element of *x* expressed as a floating-point value, such that:

- $x = frac \times \exp^{FLT\_RADIX}$
- $|frac|$ is in the interval [1, FLT_RADIX)
- If an element of *x* is negative, the corresponding element of the result is undefined.

For the **logbd2** function on the SPU:
- if an element of *x* is 0 the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **logb** functions.

SEE ALSO

logb(3), logbf4(3), frexp(3), frexpd2(3), frexpf4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

ilogbf4

NAME

ilogbf4 - return integer exponents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
#include <math.h>
vector signed int ilogbf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ilogbf4.h>
vector signed int _ilogbf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **ilogbf4** function extracts the exponents of the input vector as signed integers.

Because the SPU treats single-precision **Inf** and **NaN** codes as regular floating point numbers, **ilogbf4** returns a result of **128** for these values. However, **FP_ILOGBNAN** is set to **INT_MAX** for compatibility with the double function **ilogbd2**.

RETURN VALUE

The function **ilogbf4** returns a signed int vector in which each element is defined as:

- the macro **FP_ILOGBNAN** if the corresponding element of *x* is not a number (**NaN**),
- the macro **FP_ILOGB0** if the corresponding element of x is equal to **0** or **Inf**, or
- the value of **(int)logb(*x*)** for the corresponding element of *x* otherwise.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ilogb** functions.

SEE ALSO

ilogb(3), ilogbd2(3), logb(3), logbf4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

ilogbd2

NAME

ilogbd2 - return integer exponents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
#include <math.h>
vector signed long long ilogbd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ilogbd2.h>
vector signed long long _ilogbd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **ilogbd2** function extracts the exponents of the input vector as signed integers.

RETURN VALUE

The function **ilogbd2** returns a signed long long vector in which each element is defined as:

- the macro **FP_ILOGBNAN** if the corresponding element of *x* is not a number (**NaN**),
- the macro **FP_ILOGB0** if the corresponding element of x is equal to **0** or **Inf**, or
- the value of **(long long)logb(*x*)** for the corresponding element of *x* otherwise.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ilogb** functions.

SEE ALSO

ilogb(3), ilogbf4(3), logb(3), logbf4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log2(3), log2d2(3), log2f4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), scalbn(3), scalbnf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

scalbnf4

NAME

scalbnf4 - return float elements multiplied by integral power of 2

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float scalbnf4(vector float *x*, vector signed int *n*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <scalbnf4.h>
vector float _scalbnf4(vector float *x*, vector signed int *n*);

Parameters	
x	input vector
n	scale factor

DESCRIPTION

The **scalbnf4** function returns a vector containing each element of *x* multiplied by 2^n computed efficiently. This function is computed without the assistance of any floating point operations and as such does not set any floating point exceptions.

RETURN VALUE

The function **scalbnf4** returns a float vector in which each element is defined as:

- the corresponding element of x multiplied by 2^n .
- If the exponent is 0 then either *x* is 0 or *x* is a subnormal, and the result will be returned as **0**.
- If the result underflows it will be returned as **0**.
- If the result overflows it will be returned as **FLT_MAX**.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **scalbn** functions.

SEE ALSO

scalbn(3), log2(3), log2d2(3), log2f4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), logb(3), logbf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

scalbllnd2

NAME

scalbllnd2 - return long long elements multiplied by integral power of 2

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector long long scalbllnd2(vector long long *x*, **vector signed int** *n***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <scalbllnd2.h>
vector long long _scalbllnd2(vector long long *x*, **vector signed int** *n***);**

Parameters

x	input vector
n	scale factor

DESCRIPTION

The **scalbllnd2** function returns a vector containing each element of *x* multiplied by 2^n computed efficiently. This function is computed without the assistance of any floating point operations and as such does not set any floating point exceptions.

RETURN VALUE

The function **scalbllnd2** returns a long long vector in which each element is defined as:

- the corresponding element of x multiplied by 2^n .
- If the exponent is 0 then either *x* is 0 or *x* is a subnormal, and the result will be returned as **0**.
- If the result underflows it will be returned as **0**.
- If the result overflows it will be returned as **FLT_MAX**.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **scalblln** functions.

SEE ALSO

scalbn(3), log2(3), log2d2(3), log2f4(3), cbrt(3), cbrtd2(3), cbrtf4(3), exp(3), expd2(3), expf4(3), exp2(3), exp2d2(3), exp2f4(3), frexp(3), frexpd2(3), frexpf4(3), hypot(3), hypotd2(3), ilogb(3), ilogbd2(3), ilogbf4(3), ldexp(3), ldexpd2(3), ldexpf4(3), log(3), logd2(3), logf4(3), log10(3), log10d2(3), log10f4(3), log1p(3), log1pd2(3), log1pf4(3), logb(3), logbf4(3), sqrt(3), sqrtd2(3), sqrtf4(3)

Chapter 6. Gamma and error functions

Functions included:

lgammaf4

NAME

lgammaf4 - return base-e (natural) logarithms of gamma functions of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float lgammaf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <lgammaf4.h>
vector float _lgammaf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **lgammaf4** function returns a float vector that contains the natural logarithms of the absolute values of the results of the gamma function.

RETURN VALUE

The function **lgammaf4** returns a float vector in which each element is defined as the natural logarithm of the absolute value of the result of the gamma function on the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **lgamma** functions.

SEE ALSO

lgamma(3), lgammad2(3), erf(3), erff4(3), erfd2(3), erfc(3), erfcf4(3), erfcd2(3)

lgammad2

NAME

lgammad2 - return base-e (natural) logarithms of gamma functions of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double lgammad2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <lgammad2.h>
vector double _lgammad2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **lgammad2** function returns a double vector that contains the natural logarithms of the absolute values of the results of the gamma function.

RETURN VALUE

The function **lgammad2** returns a double vector in which each element is defined as the natural logarithm of the absolute value of the result of the gamma function on the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **lgamma** functions.

SEE ALSO

lgamma(3), lgammaf4(3), tgamma(3), tgammad2(3), tgammaf4(3), erf(3), erfd2(3), erff4(3), erfc(3), erfcd2(3), erfcf4(3)

tgammaf4

NAME

tgammaf4 - return the gamma functions of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float tgammaf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tgammaf4.h>
vector float _tgammaf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **tgammaf4** function returns a float vector that contains the results of the gamma function.

RETURN VALUE

The function **tgammaf4** returns a float vector in which each element is defined as the result of the gamma function applied to the corresponding element of *x*.

If an element of *x* is a negative integer the corresponding element of the result is undefined; no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tgamma** functions.

SEE ALSO

tgamma(3), tgammad2(3), lgamma(3), lgammad2(3), lgammaf4(3), erf(3), erff4(3), erfd2(3), erfc(3), erfcf4(3), erfcd2(3)

tgammad2

NAME

tgammad2 - return the gamma functions of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float tgammad2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tgammad2.h>
vector float _tgammad2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **tgammad2** function returns a double vector that contains the results of the gamma function.

RETURN VALUE

The function **tgammad2** returns a double vector in which each element is defined as the result of the gamma function applied to the corresponding element of *x*.

If an element of *x* is a negative integer the corresponding element of the result is undefined; no error is reported.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tgamma** functions.

SEE ALSO

tgamma(3), tgammaf4(3), lgamma(3), lgammad2(3), lgammaf4(3), erf(3), erff4(3), erfd2(3), erfc(3), erfcf4(3), erfcd2(3)

erff4

NAME

erff4 - return the error functions of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float erff4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <erff4.h>
vector float _erff4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **erff4** function returns a vector of the error functions of the corresponding elements of *x*.

RETURN VALUE

The **erff4** function returns a float vector containing the error functions of the corresponding elements of *x*.

ENVIRONMENT

PPU and SPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **erf** functions.

SEE ALSO

erf(3), erfd2(3), erfc(3), erfcf4(3), erfcd2(3), lgamma(3), lgammaf4(3), lgammad2(3)

erfd2

NAME

erfd2 - return the error functions of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double erfd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <erfd2.h>
vector double _erfd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **erfd2** function returns a vector of the error functions of the corresponding elements of *x*.

RETURN VALUE

The **erfd2** function returns a double vector containing the error functions of the corresponding elements of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **erf** functions.

SEE ALSO

erf(3), erff4(3), erfc(3), erfcf4(3), erfcd2(3), lgamma(3), lgammaf4(3), lgammad2(3)

erfcf4

NAME

erfcf4 - return the complementary error functions of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float erfcf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <erfcf4.h>
vector float _erfcf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **erfcf4** function returns a vector of complementary error functions.

RETURN VALUE

The **erfcf4** function returns a float vector of the complementary error functions of the corresponding elements of *x*.

ENVIRONMENT

PPU and SPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **erfc** functions.

SEE ALSO

erfc(3), erfcd2(3), erf(3), erff4(3), erfd2(3), lgamma(3), lgammaf4(3), lgammad2(3)

erfcd2

NAME

erfcd2 - return the complementary error functions of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double erfcd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <erfcd2.h>
vector double _erfcd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **erfcd2** function returns a vector of complementary error functions.

RETURN VALUE

The **erfcd2** function returns a double vector of the complementary error functions of the corresponding elements of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **erfc** functions.

SEE ALSO

erfc(3), erfcf4(3), erf(3), erff4(3), erfd2(3), lgamma(3), lgammaf4(3), lgammad2(3)

Chapter 7. Maximum, minimum and difference functions

Functions included:

fmaxf4

NAME

fmaxf4 - return larger values of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fmaxf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmaxf4.h>
vector float _fmaxf4(vector float *x*, vector float *y*);

Parameters	
x.*y*	input vectors

DESCRIPTION

The **fmaxf4** function returns a vector containing the larger (more positive) elements of *x* and *y*.

RETURN VALUE

The function **fmaxf4** returns a float vector in which each element is defined as:

- the larger (more positive) of the corresponding elements of *x* and *y*.
- If one element is **NaN** and the other is numeric, the numeric value is returned.
- If both elements are **NaN**, **NaN** is returned.

On the SPU single-precision subnormal values are not coerced to zero by this function. Instead, it compares them as normal values even though the floating-point instructions of the SPU do not.

In double precision subnormals[1] equate to zero and so compare as equal. This means that the value returned may be either one of the subnormals, thereby making the following possibly true for two subnormal inputs:

fmaxf4(*a*, *b*) != fmaxf4(*b*, *a*)

1. subnormality: a) the transitive closure of normality; b) floating-point numbers too small to be expressed in normal form.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmax** functions.

SEE ALSO

fmax(3), fmaxd2(3), fmin(3), fminf4(3), fmind2(3), fdim(3), fdimf4(3), fdimd2(3)

fmaxd2

NAME

fmaxd2 - return larger values of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fmaxd2(vector double *x*, **vector double** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmaxd2.h>
vector double _fmaxd2(vector double *x*, **vector double** *y***);**

Parameters	
x.y	input vectors

DESCRIPTION

The **fmaxd2** functions returns a vector containing the larger (more positive) elements of *x* and *y*.

RETURN VALUE

The function **fmaxd2** returns a double vector in which each element is defined as:
- the larger (more positive) of the corresponding elements of *x* and *y*.
- If one element is **NaN** and the other is numeric, the numeric value is returned.
- If both elements are **NaN**, **NaN** is returned.

In double precision subnormals[2] equate to zero and so compare as equal. This means that the value returned may be either one of the subnormals, thereby making the following possibly true for two subnormal inputs:

fmaxd2(*a*, *b***) != fmaxd2(***b*, *a***)**

ENVIRONMENT

SPU only

2. subnormality: a) the transitive closure of normality; b) floating-point numbers too small to be expressed in normal form.

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmax** functions.

SEE ALSO

fmax(3), fmaxf4(3), fmin(3), fminf4(3), fmind2(3), fdim(3), fdimf4(3), fdimd2(3)

fminf4

NAME

fminf4 - return smaller values of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fminf4(vector float *x*, **vector float** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fminf4.h>
vector float _fminf4(vector float *x*, **vector float** *y***);**

Parameters	
x.*y*	input vectors

DESCRIPTION

The **fminf4** functions return a vector containing the smaller (less positive) elements of *x* and *y*.

RETURN VALUE

The function **fminf4** returns a float vector in which each element is defined as:

- the smaller (less positive) of the corresponding elements of *x* and *y*.
- If one element is **NaN** and the other is numeric, the numeric value is returned.
- If both elements are **NaN**, **NaN** is returned.

On the SPU single-precision subnormal values are not coerced to zero by this function. Instead, it compares them as normal values even though the floating-point instructions of the SPU do not.

In double precision subnormals[3] equate to zero and so compare as equal. This means that the value returned may be either one of the subnormals, thereby making the following possibly true for two subnormal inputs:

fminf4(*a***,** *b***) != fminf4(***b***,** *a***)**

3. subnormality: a) the transitive closure of normality; b) floating-point numbers too small to be expressed in normal form.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmin** functions.

SEE ALSO

fmin(3), fmind2(3), fmax(3), fmaxf4(3), fmaxd2(3), fdim(3), fdimf4(3), fdimd2(3)

fmind2

NAME

fmind2 - return smaller values of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fmind2(vector double *x*, **vector double** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fmind2.h>
vector double _fmind2(vector double *x*, **vector double** *y***);**

Parameters

x.y	input vectors

DESCRIPTION

The **fmind2** function returns a vector containing the smaller (less positive) elements of *x* and *y*.

RETURN VALUE

The function **fmind2** returns a double vector in which each element is defined as:
- the smaller (less positive) of the corresponding elements of *x* and *y*.
- If one element is **NaN** and the other is numeric, the numeric value is returned.
- If both elements are **NaN**, **NaN** is returned.

In double precision subnormals[4] equate to zero and so compare as equal. This means that the value returned may be either one of the subnormals, thereby making the following possibly true for two subnormal inputs:

fmind2(*a*, *b***) != fmind2(***b*, *a***)**

ENVIRONMENT

SPU only

4. subnormality: a) the transitive closure of normality; b) floating-point numbers too small to be expressed in normal form.

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fmin** functions.

SEE ALSO

fmin(3), fminf4(3), fmax(3), fmaxf4(3), fmaxd2(3), fdim(3), fdimf4(3), fdimd2(3)

fdimf4

NAME

fdimf4 - return the positive differences between float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float fdimf4(vector float *x*, vector float *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fdimf4.h>
vector float _fdimf4(vector float *x*, vector float *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **fdimf4** function returns a vector of the positive differences between the elements of the input vectors.

RETURN VALUE

The function **fdimf4** returns a float vector in which each element is defined as the larger of (*x* - *y*) and zero, for corresponding elements of *x* and *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fdim** functions.

SEE ALSO

fdim(3), fdimd2(3), fmax(3), fmaxf4(3), fmaxd2(3), fmin(3), fminf4(3), fmind2(3)

fdimd2

NAME

fdimd2 - return the positive differences between double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double fdimd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <fdimd2.h>
vector double _fdimd2(vector double *x*, vector double *y*);

Parameters	
x,*y*	input vectors

DESCRIPTION

The **fdimd2** function returns a vector of the positive differences between the elements of the input vectors.

RETURN VALUE

The function **fdimd2** returns a double vector in which each element is defined as the larger of (*x* - *y*) and zero, for corresponding elements of *x* and *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **fdim** functions.

SEE ALSO

fdim(3), fdimf4(3), fmax(3), fmaxf4(3), fmaxd2(3), fmin(3), fminf4(3), fmind2(3)

Chapter 8. Rounding and next functions

Functions included:

ceilf4

NAME

ceilf4 - return accurate ceilings of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float ceilf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ceilf4.h>
vector float _ceilf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

These functions round the elements of the input vector upwards to the next integer value.

They provide ceiling computation for the entire input range of IEEE floating point numbers. The ceiling of **NaN** values remain **NaN**, and the ceiling of subnormal values become zero.

On the SPU **ceilf4_fast** provides a limited range form which computes the ceiling of all floating-point values in the 32-bit signed integer range. Values outside this range get clamped to either **0** or **MAX_INT**. This mode is faster to compute, but has less range.

RETURN VALUE

The **ceilf4** function returns a float vector in which each element is defined as the smallest integer value not less than *x*.

ENVIRONMENT

Full range: SPU and PPU

Integer range: SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ceil** functions.

SEE ALSO

ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

ceilf4_fast

NAME

ceilf4_fast - return approximate ceilings of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float ceilf4_fast(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ceilf4.h>
vector float _ceilf4_fast(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

These functions round the elements of the input vector upwards to the next integer value.

They provide ceiling computation for the entire input range of IEEE floating point numbers. The ceiling of **NaN** values remain **NaN**, and the ceiling of subnormal values become zero.

On the SPU **ceilf4_fast** provides a limited range form which computes the ceiling of all floating-point values in the 32-bit signed integer range. Values outside this range get clamped to either **0** or **MAX_INT**. This mode is faster to compute, but has less range.

RETURN VALUE

The **ceilf4_fast** function returns a float vector in which each element is defined as the smallest integer value not less than *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ceil** functions.

SEE ALSO

ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

ceild2

NAME

ceild2 - return ceilings of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double ceild2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <ceild2.h>
vector double _ceild2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **ceild2** function rounds the elements of the input vector upwards to the next integer value.

This function provides ceiling computation for the entire input range of IEEE floating point numbers. The ceiling of **NaN** values remain **NaN**, and the ceiling of subnormal values become zero.

RETURN VALUE

The function **ceild2** returns a double vector in which each element is defined as the smallest integer value not less than *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **ceil** functions.

SEE ALSO

ceil(3), ceilf4(3), ceilf4(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

floorf4

NAME

floorf4 - return accurate floors of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float floorf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <floorf4.h>
vector float floorf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

These functions round the elements of the input vector downwards to the next integer value.

They provide floor computation for the entire input range of IEEE floating point numbers. The floor of **NaN** values remain **NaN**, and the floor of subnormal values become zero.

On the SPU **floorf4_fast** provides a limited range form which computes the floor of all floating-point values in the 32-bit signed integer range. Values outside this range get clamped to either **0** or **MAX_INT**. This mode is faster to compute, but has less range.

RETURN VALUE

The function **floorf4** returns a float vector in which each element is defined as the largest integer value not greater than *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **floor** functions.

SEE ALSO

floor(3), floorf4(3), floord2(3), ceil(3), ceilf4(3), ceild2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

floorf4_fast

NAME

floorf4_fast - return approximate floors of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float floorf4_fast(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <floorf4.h>
vector float _floorf4_fast(vector float *x*);

Parameters

x	input vector

DESCRIPTION

These functions round the elements of the input vector downwards to the next integer value.

They provide floor computation for the entire input range of IEEE floating point numbers. The floor of **NaN** values remain **NaN**, and the floor of subnormal values become zero.

On the SPU **floorf4_fast** provides a limited range form which computes the floor of all floating-point values in the 32-bit signed integer range. Values outside this range get clamped to either **0** or **MAX_INT**. This mode is faster to compute, but has less range.

RETURN VALUE

The function **floorf4_fast** returns a float vector in which each element is defined as the largest integer value not greater than *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **floor** functions.

SEE ALSO

floor(3), floorf4(3), floord2(3), ceil(3), ceilf4(3), ceild2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

floord2

NAME

floord2 - return floors of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double floord2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <floord2.h>
vector double _floord2(vector double *x***);**

Parameters	
x	input vector

DESCRIPTION

The **floord2** function rounds the elements of the input vector downwards to the next integer value.

This function provides floor computation for the entire input range of IEEE floating point numbers. The floor of **NaN** values remain **NaN**, and the floor of subnormal values become zero.

RETURN VALUE

The function **floord2** returns a double vector in which each element is defined as the largest integer value not greater than *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **floor** functions.

SEE ALSO

floor(3), floorf4(3), floorf4(3), ceil(3), ceilf4(3), ceild2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

nearbyintf4

NAME

nearbyintf4 - return nearest integers to float elements ignoring floating point exceptions

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float nearbyintf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <nearbyintf4.h>
vector float _nearbyintf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **nearbyintf4** function returns a vector of the corresponding elements of *x* rounded to the nearest integer, consistent with the current rounding mode but without raising an inexact floating-point exception.

Special Case:

- For the **nearbyintf4** function on the SPU the rounding mode is always towards zero.

RETURN VALUE

The function **nearbyintf4** returns a float vector in which each element is defined as the integer nearest to the corresponding element of *x* according to the current rounding mode.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **nearbyint** functions.

SEE ALSO

nearbyint(3), nearbyintd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

nearbyintd2

NAME

nearbyintd2 - return nearest integers to double elements ignoring floating point exceptions

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double nearbyintd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <nearbyintd2.h>
vector double _nearbyintd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **nearbyintd2** function returns a vector of the corresponding elements of *x* rounded to the nearest integer, consistent with the current rounding mode but without raising an inexact floating-point exception.

RETURN VALUE

The function **nearbyintd2** returns a double vector in which each element is defined as the integer nearest to the corresponding element of *x* according to the current rounding mode.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **nearbyint** functions.

SEE ALSO

nearbyint(3), nearbyintf4(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

irintf4

NAME

irintf4 - return the nearest integer values to float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(vector signed int) irintf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <irintf4.h>
(vector signed int) _irintf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **irintf4** function returns a vector containing the nearest integers to the corresponding elements of *x* consistent with the current rounding mode.

Special Cases:

- On the SPU, the rounding mode for floats is always towards zero.

RETURN VALUE

The function **irintf4** returns a vector of signed integers in which each element is defined as the nearest integer consistent with the current rounding mode for the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **irint** functions.

SEE ALSO

irint(3), llrint(3), llrintf4(3), llrintd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

llrintf4

NAME

llrintf4 - return nearest integer values to float elements consistent with rounding mode

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
llroundf4_t llrintf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <llrintf4.h>
llroundf4_t _llrintf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **llrintf4** function returns a structure of vectors of signed long long integers which are nearest to the corresponding elements of *x* consistent with the current rounding mode.

Special Cases:

- On the SPU the rounding mode is always towards 0 (zero).
- If the rounded value is outside the range of the return type the numeric result is unspecified.

RETURN VALUE

The function **llrintf4** returns a **llroundf4_t** structure containing vectors in which each element is defined as the nearest long long integer to the corresponding element of *x* consistent with the current rounding mode.

The **llroundf4_t** structure is defined:

```
typedef struct llroundf4_t {
vector signed long long vll[2];
} llroundf4_t;
```

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **llrint** functions.

SEE ALSO

llrint(3), llrintd2(3), llround(3), llroundf4(3), llroundd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), trunc(3), truncf4(3), truncd2(3)

llrintd2

NAME

llrintd2 - return nearest integer values to double elements consistent with rounding mode

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed long long llrintd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <llrintd2.h>
vector signed long long _llrintd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **llrintd2** function returns a vector of signed long long integers which are nearest to the corresponding elements of *x* consistent with the current rounding mode.

Special Cases:

- The rounding mode is always towards 0 (zero).
- If the rounded value is outside the range of the return type the numeric esult is unspecified.

RETURN VALUE

The function **llrintd2** returns a signed long long vector in which each element is defined as the nearest long long integer to the corresponding element of *x* consistent with the current rounding mode.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **llrint** functions.

SEE ALSO

llrint(3), llrintf4(3), llround(3), llroundf4(3), llroundd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), trunc(3), truncf4(3), truncd2(3)

rintf4

NAME

rintf4 - return the nearest integer values to float elements consistent with the current rounding mode

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float rintf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <rintf4.h>
vector float _rintf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **rintf4** function returns a vector which contains the corresponding elements of *x* rounded to the nearest integer consistent with the current rounding mode.

Special Case:

- On the SPU, the rounding mode is always towards zero.

RETURN VALUE

The function **rintf4** returns a float vector in which each element is defined as the integer nearest to the corresponding element of *x* according to the current rounding mode.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **rint** functions.

SEE ALSO

rint(3), rintd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

rintd2

NAME

rintd2 - return the nearest integer values to double elements consistent with the current rounding mode

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double rintd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <rintd2.h>
vector double _rintd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **rintd2** function returns a vector which contains the corresponding elements of *x* rounded to the nearest integer consistent with the current rounding mode.

RETURN VALUE

The function **rintd2** returns a double vector in which each element is defined as the integer nearest to the corresponding element of *x* according to the current rounding mode.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **rint** functions.

SEE ALSO

rint(3), rintf4(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

roundf4

NAME

roundf4 - return the nearest integer values to float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float roundf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <roundf4.h>
vector float _roundf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **roundf4** function returns a vector which contains the corresponding elements of *x* rounded to the nearest integer.

Special Cases:

- Halfway values are rounded away from 0 (zero), regardless of the current rounding direction.
- On the SPU, the rounding mode is always towards 0 (zero).

RETURN VALUE

The function **roundf4** returns a float vector in which each element is defined as the nearest integer to the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **round** functions.

SEE ALSO

round(3), roundd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

roundd2

NAME

roundd2 - return the nearest integer values to double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double roundd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <roundd2.h>
vector double _roundd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **roundd2** function returns a vector which contains the corresponding elements of *x* rounded to the nearest integer.

Special Cases:

- Halfway values are rounded away from 0 (zero), regardless of the current rounding direction
- The rounding mode is always towards 0 (zero).

RETURN VALUE

The function **roundd2** returns a double vector in which each element is defined as the nearest integer to the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **round** functions.

SEE ALSO

round(3), roundf4(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

iroundf4

NAME

iroundf4 - return the nearest integer values to float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(vector signed int) iroundf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <iroundf4.h>
(vector signed int) _iroundf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **iroundf4** function returns a vector of signed integers that contains the corresponding elements of *x* rounded to the nearest integer value, rounding halfway values away from 0 (zero), regardless of the current rounding direction.

RETURN VALUE

The function **iroundf4** returns a vector of signed integers defined as the nearest integer to the corresponding element of *x*.

If the rounded value is outside the range of the return type then the result is unspecified.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **iround** functions.

SEE ALSO

iround(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

llroundf4

NAME

llroundf4 - return nearest integer values to float elements

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
llroundf4_t llroundf4(vector float x);
Link with -lsimdmath
```

Inline call syntax:

```
#include <simdmath.h>
#include <llroundf4.h>
llroundf4_t _llroundf4(vector float x);
```

Parameters

x	input vector

DESCRIPTION

The **llroundf4** function returns a structure of vectors of signed long long integers which are nearest to the corresponding elements of *x*.

Special Cases:

- Halfway values are rounded away from 0 (zero).
- On the SPU the rounding mode is always towards 0 (zero).
- If the rounded value is outside the range of the return type the numeric result is unspecified.

RETURN VALUE

The function **llroundf4** returns a **llroundf4_t** structure containing vectors in which each element is defined as the nearest long long integer to the corresponding element of *x*.

The **llroundf4_t** structure is defined:

```
typedef struct llroundf4_t {
vector signed long long vll[2];
} llroundf4_t;
```

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **llround** functions.

SEE ALSO

llroundf4_t(3), llround(3), llroundd2(3), llrint(3), llrintf4(3), llrintd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), trunc(3), truncf4(3), truncd2(3)

llroundd2

NAME

llroundd2 - return nearest integer values to double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector signed long long llroundd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <llroundd2.h>
vector signed long long _llroundd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **llroundd2** function returns a vector of signed long long integers which are nearest to the corresponding elements of *x*.

Special Cases:
- Halfway values are rounded away from 0 (zero), regardless of the current rounding direction.
- The rounding mode is always towards 0 (zero).
- If the rounded value is outside the range of the return type the numeric result is unspecified.

RETURN VALUE

The function **llroundd2** returns a signed long long vector in which each element is defined as the nearest long long integer to the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **llround** functions.

SEE ALSO

llround(3), llroundf4(3), llrint(3), llrintf4(3), llrintd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), trunc(3), truncf4(3), truncd2(3)

truncf4

NAME

truncf4 - return nearest integers with less magnitude than float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float truncf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <truncf4.h>
vector float _truncf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **truncf4** function returns a vector of the corresponding elements of *x* rounded to the nearest integer not larger in absolute value (rounded towards 0).

RETURN VALUE

The function **truncf4** returns a float vector in which each element is defined as the nearest integer that is not larger in magnitude than the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **trunc** functions.

SEE ALSO

trunc(3), truncd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3)

truncd2

NAME

truncd2 - return nearest integers with less magnitude than double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double truncd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <truncd2.h>
vector double _truncd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **truncd2** function returns a vector of the corresponding elements of *x* rounded to the nearest integer not larger in absolute value (rounded towards 0).

RETURN VALUE

The function **truncd2** returns a double vector in which each element is defined as the nearest integer that is not larger in magnitude than the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **trunc** functions.

SEE ALSO

trunc(3), truncf4(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), nextafter(3), nextafterf4(3), nextafterd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3)

nextafterf4

NAME

nextafterf4 - return next representable values after float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float nextafterf4(vector float *x***, vector float** *y***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <nextafterf4.h>
vector float _nextafterf4(vector float *x***, vector float** *y***);**

Parameters	
x, *y*	input vectors

DESCRIPTION

The **nextafterf4** function returns a vector of the next representable value after each element of *x* in the direction of the corresponding element of *y*.

RETURN VALUE

The function **nextafterf4** returns a float vector in which each element is defined as the next representable value after the corresponding element of *x* in the direction of the corresponding element of *y*. If the element of *x* is equal to the corresponding element of *y*, the result is *y*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **nextafter** functions.

SEE ALSO

nextafter(3), nextafterd2(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

nextafterd2

NAME

nextafterd2 - return next representable values after double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double nextafterd2(vector double *x*, vector double *y*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <nextafterd2.h>
vector double _nextafterd2(vector double *x*, vector double *y*);

Parameters

x, *y*	input vectors

DESCRIPTION

The **nextafterd2** function returns a vector of the next representable value after each element of *x* in the direction of the corresponding element of *y*.

RETURN VALUE

The function **nextafterd2** returns a double vector in which each element is defined as the next representable value after the corresponding element of *x* in the direction of the corresponding element of *y*. If the element of *x* is equal to the corresponding element of *y*, the result is *y*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **nextafter** functions.

SEE ALSO

nextafter(3), nextafterf4(3), ceil(3), ceilf4(3), ceild2(3), floor(3), floorf4(3), floord2(3), nearbyint(3), nearbyintf4(3), nearbyintd2(3), irint(3), irintf4(3), llrint(3), llrintf4(3), llrintd2(3), rint(3), rintf4(3), rintd2(3), round(3), roundf4(3), roundd2(3), iround(3), iroundf4(3), llround(3), llroundf4(3), llroundd2(3), trunc(3), truncf4(3), truncd2(3)

Chapter 9. Trigonometric Functions

Functions included:

sinf4

NAME

sinf4 - return sines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float sinf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sinf4.h>
vector float _sinf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **sinf4** function returns a vector of the sines of the elements of *x*.

The result of the **sinf4** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **sinf4** returns a float vector in which each element is defined as the sine of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sin** functions.

SEE ALSO

sin(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

sind2

NAME

sind2 - return sines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double sind2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sind2.h>
vector double _sind2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **sind2** function returns a vector of the sines of the elements of *x*.

The result of the **sind2** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **sind2** returns a double vector in which each element is defined as the sine of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sin** functions.

SEE ALSO

sin(3), sinf4(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

cosf4

NAME

cosf4 - return cosines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float cosf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <cosf4.h>
vector float _cosf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **cosf4** functions returns a vector of the cosines of the elements of *x*.

The result of the **cosf4** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **cosf4** returns a float vector in which each element is defined as the cosine of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cos** functions.

SEE ALSO

cos(3), cosd2(3), sin(3), sinf4(3), sind2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

cosd2

NAME

cosd2 - return cosines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double cosd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <cosd2.h>
vector double _cosd2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **cosd2** function returns a vector of the cosines of the elements of *x*.

The result of the **cosd2** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **cosd2** returns a double vector in which each element is defined as the cosine of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cos** functions.

SEE ALSO

cos(3), cosf4(3), sin(3), sinf4(3), sind2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

tanf4

NAME

tanf4 - return tangents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float tanf4(vector float *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tanf4.h>
vector float _tanf4(vector float *x***);**

Parameters	
x	input vector

DESCRIPTION

The **tanf4** function returns a vector of the tangents of the elements of *x*.

The result of the **tanf4** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **tanf4** returns a float vector in which each element is defined as the tangent of the corresponding element of *x*.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tan** functions.

SEE ALSO

tan(3), tand2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

tand2

NAME

tand2 - return tangents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double tand2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tand2.h>
vector double _tand2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **tand2** function returns a vector of the tangents of the elements of *x*.

The result of the **tand2** function is not accurate for very large values of *x*, and no error is reported.

RETURN VALUE

The function **tand2** returns a double vector in which each element is defined as the tangent of the corresponding element of *x*.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tan** functions.

SEE ALSO

tan(3), tanf4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

sincosf4

NAME

sincosf4 - return sines and cosines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(void) sincosf4(vector float *x*, vector float **sx*, vector float **cx*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sincosf4.h>
(void) _sincosf4(vector float *x*, vector float **sx*, vector float **cx*);

Input parameter	
x	input vector

Return parameters	
**sx*	pointer to a vector of sines
**cx*	pointer to a vector of cosines

DESCRIPTION

The **sincosf4** function returns two vectors containing the sines and cosines of the elements of *x*.

RETURN VALUE

The function **sincosf4** returns two float vectors in which each element is defined as **sin**(*x*) and **cos**(*x*).

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sin** and **cos** functions.

SEE ALSO

sincos(3), sincosd2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

sincosd2

NAME

sincosd2 - return sines and cosines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
(void) sincosd2(vector double *x*, **vector double** **sx*, **vector double** **cx*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sincosd2.h>
(void) _sincosd2(vector double *x*, **vector double** **sx*, **vector double** **cx*);

Input parameter	
x	input vector

Return parameters	
**sx*	pointer to a vector of sines
**cx*	pointer to a vector of cosines

DESCRIPTION

The **sincosd2** function returns two vectors containing the sines and cosines of the elements of *x*.

RETURN VALUE

The function **sincosd2** returns two double vectors in which each element is defined as **sin**(*x*) and **cos**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sin** and **cos** functions.

SEE ALSO

sincos(3), sincosf4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

asinf4

NAME

asinf4 - return arc sines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float asinf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <asinf4.h>
vector float _asinf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **asinf4** function returns a vector of the arc sines of the elements of *x*. Inputs must be within the interval [-1,+1].

RETURN VALUE

The function **asinf4** returns a float vector in which each element is defined as:

- the arc sine of the corresponding element of *x*, if the element of x is within the interval [-1,+1],
- undefined otherwise.

Each element in the return vector is expressed in radians.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

JSRE, ISO9899 (C99) **asin** functions.

SEE ALSO

asin(3), asind2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

asind2

NAME

asind2 - return arc sines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double asind2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <asind2.h>
vector double _asind2(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **asind2** function returns a vector of the arc sines of the elements of *x*. Inputs must be within the interval [-1,+1].

RETURN VALUE

The function **asind2** returns a double vector in which each element is defined as:

- the arc sine of the corresponding element of *x*, if the element of x is within the interval [-1,+1],
- undefined otherwise.

Each element in the return vector is expressed in radians.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

JSRE, ISO9899 (C99) **asin** functions.

SEE ALSO

asin(3), asinf4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

acosf4

NAME

acosf4 - return arc cosines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float acosf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <acosf4.h>
vector float _acosf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **acosf4** function returns a vector of the arc cosines of the elements of *x*. Inputs must be within the interval [-1,+1].

RETURN VALUE

The function **acosf4** returns a float vector in which each element is defined as:

- the arc cosine of the corresponding element of *x*, if the element of x is within the interval [-1,+1],
- undefined otherwise.

Each element in the return vector is expressed in radians.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **acos** functions.

SEE ALSO

acos(3), acosd2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

acosd2

NAME

acosd2 - return arc cosines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double acosd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <acosd2.h>
vector double _acosd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **acosd2** function returns a vector of the arc cosines of the elements of *x*. Inputs must be within the interval [-1,+1].

RETURN VALUE

The function **acosd2** returns a double vector in which each element is defined as:

- the arc cosine of the corresponding element of *x*, if the element of x is within the interval [-1,+1],
- undefined otherwise.

Each element in the return vector is expressed in radians.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **acos** functions.

SEE ALSO

acos(3), acosf4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), atan(3), atanf4(3), atand2(3), atan2(3), atan2f4(3), atan2d2(3)

atanf4

NAME

atanf4 - return arc tangents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float atanf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <atanf4.h>
vector float _atanf4(vector float *x*);

Parameters

x	input vector

DESCRIPTION

The **atanf4** function returns a vector of the arc tangents of the elements of *x*. Each element in the return vector is expressed in radians.

RETURN VALUE

The function **atanf4** returns a float vector in which each element is defined as the arc tangent of the corresponding element of *x*. Each element in the return vector is expressed in radians. Return values will be within the interval $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$.

If the corresponding elements of *x* and *y* are zero then the corresponding element of the result is undefined.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atan** functions.

SEE ALSO

atan(3), atand2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan2(3), atan2f4(3), atan2d2(3)

atand2

NAME

atand2 - return arc tangents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double atand2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <atand2.h>
vector double _atand2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **atand2** function returns a vector of the arc tangents of the elements of *x*.

RETURN VALUE

The function **atand2** returns a double vector in which each element is defined as the arc tangent of the corresponding element of *x*. Each element in the return vector is expressed in radians. Return values will be within the interval $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$.

If the corresponding elements of *x* and *y* are zero then the corresponding element of the result is undefined.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atan** functions.

SEE ALSO

atan(3), atanf4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan2(3), atan2f4(3), atan2d2(3)

atan2f4

NAME

atan2f4 - return arc tangents of division of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float atan2f4(vector float *y*, vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <atan2f4.h>
vector float _atan2f4(vector float *y*, vector float *x*);

Parameters	
y,*x*	input vectors

DESCRIPTION

The **atan2f4** function calculates the arc tangents of each of the elements in *y* and *x*. This function is similar to computing **atan(***y*/*x***)**; however the sign of each of the elements is used to determine the quadrant of the result.

RETURN VALUE

The function **atan2f4** returns a float vector in which each element is defined as the arc tangent of y/x using the signs of y and x to determine the quadrant of the result.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atan2** functions.

SEE ALSO

atan2(3), atan2d2(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3)

atan2d2

NAME

atan2d2 - return arc tangents of division of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double atan2d2(vector double *y*, vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <atan2d2.h>
vector double _atan2d2(vector double *y*, vector double *x*);

Parameters

y,x	input vectors

DESCRIPTION

The **atan2d2** function calculates the arc tangents of each of the elements in *y* and *x*. This function is similar to computing **atan(***y***/***x***)**; however the sign of each of the elements is used to determine the quadrant of the result.

RETURN VALUE

The function **atan2d2** returns a double vector in which each element is defined as the arc tangent of *y*/*x* using the signs of *y* and *x* to determine the quadrant of the result.

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atan2** functions.

SEE ALSO

atan2(3), atan2f4(3), sin(3), sinf4(3), sind2(3), cos(3), cosf4(3), cosd2(3), sincos(3), sincosf4(3), sincosd2(3), tan(3), tanf4(3), tand2(3), asin(3), asinf4(3), asind2(3), acos(3), acosf4(3), acosd2(3), atan(3), atanf4(3), atand2(3)

Chapter 10. Hyperbolic Functions

Functions included:

sinhf4

NAME

sinhf4 - return hyperbolic sines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float sinhf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sinhf4.h>
vector float _sinhf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **sinhf4** function returns the hyperbolic sines of the elements of *x*.

RETURN VALUE

The function **sinhf4** returns a float vector in which each element is defined as **sinh**(*x*).

On the SPU element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sinh** functions.

SEE ALSO

sinh(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

sinhd2

NAME

sinhd2 - return hyperbolic sines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double sinhd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <sinhd2.h>
vector double _sinhd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **sinhd2** function returns the hyperbolic sines of the elements of *x*.

RETURN VALUE

The function **sinhd2** returns a double vector in which each element is defined as **sinh**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **sinh** functions.

SEE ALSO

sinh(3), sinhf4(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

coshf4

NAME

coshf4 - return hyperbolic cosines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float coshf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <coshf4.h>
vector float _coshf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **coshf4** function returns the hyperbolic cosines of the elements of *x*.

RETURN VALUE

The function **coshf4** returns a float vector in which each element is defined as **cosh**(*x*).

On the SPU element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cosh** functions.

SEE ALSO

cosh(3), coshd2(3), sinh(3), sinhf4(3), sinhd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

coshd2

NAME

coshd2 - return hyperbolic cosines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double coshd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <coshd2.h>
vector double _coshd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **coshd2** function returns the hyperbolic cosines of the elements of *x*.

RETURN VALUE

The function **coshd2** returns a double vector in which each element is defined as **cosh**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **cosh** functions.

SEE ALSO

cosh(3), coshf4(3), sinh(3), sinhf4(3), sinhd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

tanhf4

NAME

tanhf4 - return hyperbolic tangents of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float tanhf4(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tanhf4.h>
vector float _tanhf4(vector double *x*);

Parameters	
x	input vector

DESCRIPTION

The **tanhf4** function returns the hyperbolic tangents of the elements of *x*.

RETURN VALUE

The function **tanhf4** returns a float vector in which each element is defined as **tanh**(*x*).

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tanh** functions.

SEE ALSO

tanh(3), tanhd2(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

tanhd2

NAME

tanhd2 - return hyperbolic tangents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double tanhd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <tanhd2.h>
vector double _tanhd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **tanhd2** function returns the hyperbolic tangents of the elements of *x*.

RETURN VALUE

The function **tanhd2** returns a double vector in which each element is defined as **tanh**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **tanh** functions.

SEE ALSO

tanh(3), tanhf4(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

asinhf4

NAME

asinhf4 - return inverse hyperbolic sines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float asinhf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <asinf4.h>
vector float _asinhf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **asinhf4** function returns the inverse hyperbolic sines of the elements of *x*.

RETURN VALUE

The function **asinhf4** returns a float vector in which each element is defined as **asinh**(*x*).

On the SPU element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **asinh** functions.

SEE ALSO

asinh(3), asinhd2(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

asinhd2

NAME

asinhd2 - return inverse hyperbolic sines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double asinhd2(vector double *x***);**
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <asinhd2.h>
vector double _asinhd2(vector double *x***);**

Parameters

x	input vector

DESCRIPTION

The **asinhd2** function returns the inverse hyperbolic sines of the elements of *x*.

RETURN VALUE

The function **asinhd2** returns a double vector in which each element is defined as **asinh**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **asinh** functions.

SEE ALSO

asinh(3), asinhf4(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), acosh(3), acoshf4(3), acoshd2(3), atanh(3), atanhf4(3), atanhd2(3)

acoshf4

NAME

acoshf4 - return inverse hyperbolic cosines of float elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector float acoshf4(vector float *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <acoshf4.h>
vector float _acoshf4(vector float *x*);

Parameters	
x	input vector

DESCRIPTION

The **acoshf4** function returns the inverse hyperbolic cosines of the elements of *x*.

RETURN VALUE

The function **acoshf4** returns a float vector in which each element is defined as **acosh**(*x*).

On the SPU element values of the result that are greater than **HUGE_VALF** are returned as **HUGE_VALF** and no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **acosh** functions.

SEE ALSO

acosh(3), acoshd2(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), atanh(3), atanhf4(3), atanhd2(3)

acoshd2

NAME

acoshd2 - return inverse hyperbolic cosines of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double acoshd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <acoshd2.h>
vector double _acoshd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **acoshd2** function returns the inverse hyperbolic cosines of the elements of *x*.

RETURN VALUE

The function **acoshd2** returns a double vector in which each element is defined as **acosh**(*x*).

ENVIRONMENT

SPU only

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **acosh** functions.

SEE ALSO

acosh(3), acoshf4(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), atanh(3), atanhf4(3), atanhd2(3)

atanhf4

NAME

atanhf4 - return inverse hyperbolic tangents of float elements

SYNOPSIS

Procedure call syntax:

```
#include <simdmath.h>
vector float atanhf4(vector float x);
Link with -lsimdmath
```

Inline call syntax:

```
#include <simdmath.h>
#include <atanhf4.h>
vector float _atanhf4(vector float x);
```

Parameters

x	input vector

DESCRIPTION

The **atanhf4** function returns the inverse hyperbolic tangents of the elements of *x*.

RETURN VALUE

The function **atanhf4** returns a float vector in which each element is defined as **atanh**(*x*).

On the SPU, if the absolute value of *xi* is equal to 1, the corresponding element of the result is returned as **HUGE_VALF**, and if *xi* is equal to -1, the corresponding element of the result is returned as **-HUGE_VALF**. In either case, no error is reported.

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atanh** functions.

SEE ALSO

atanh(3), atanhd2(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3)

atanhd2

NAME

atanhd2 - return inverse hyperbolic tangents of double elements

SYNOPSIS

Procedure call syntax:

#include <simdmath.h>
vector double atanhd2(vector double *x*);
Link with -lsimdmath

Inline call syntax:

#include <simdmath.h>
#include <atanhd2.h>
vector double _atanhd2(vector double *x*);

Parameters

x	input vector

DESCRIPTION

The **atanhd2** function returns the inverse hyperbolic tangents of the elements of *x*.

RETURN VALUE

The function **atanhd2** returns a float vector in which each element is defined as **atanh**(*x*).

ENVIRONMENT

SPU and PPU

CONFORMING TO

SIMD Math library specification for the Cell Broadband Engine Architecture.

NOTES

Basis

ISO9899 (C99) **atanh** functions.

SEE ALSO

atanh(3), atanhf4(3), sinh(3), sinhf4(3), sinhd2(3), cosh(3), coshf4(3), coshd2(3), tanh(3), tanhf4(3), tanhd2(3), asinh(3), asinhf4(3), asinhd2(3), acosh(3), acoshf4(3), acoshd2(3)

Chapter 11. Type definitions

divi4_t

NAME

divi4_t - remainder/quotient struct for vector signed int

SYNOPSIS

```
typedef struct divi4_s {
        vector signed int quot;
        vector signed int rem;
} divi4_t;
```

ENVIRONMENT

SPU and PPU

SEE ALSO

divi4(3), divi(3)

divu4_t

NAME

divu4_t - remainder/quotient struct for vector unsigned int

SYNOPSIS

```
typedef struct divu4_s {
    vector unsigned int quot;
    vector unsigned int rem;
} divu4_t;
```

ENVIRONMENT

SPU and PPU

SEE ALSO

divu4(3), div(3)

lldivi2_t

NAME

lldivi2_t - remainder/quotient struct for vector unsigned long long

SYNOPSIS

```
typedef struct lldivi2_s {
    vector unsigned long long quot;
    vector unsigned long long rem;
} lldivi2_t;
```

ENVIRONMENT

SPU only

SEE ALSO

lldivi2(3), lldiv(3)

lldivu2_t

NAME

lldivu2_t - remainder/quotient struct for vector unsigned long long

SYNOPSIS

```
typedef struct lldivu2_s {
    vector signed long long quot;
    vector signed long long rem;
} lldivu2_t;
```

ENVIRONMENT

SPU only

SEE ALSO

lldivu2(3), lldiv(3)

llroundf4_t

NAME

llroundf4_t - struct for vector signed long long

SYNOPSIS

```
typedef struct llroundf4_s {
        vector signed long long vll[2];
} llroundf4_t;
```

ENVIRONMENT

SPU only

SEE ALSO

llroundf4(3), llround(3)

Related documentation

This topic helps you find related information.

Document location

Links to documentation for the SDK are provided on the developerWorks® Web site located at:

`http://www-128.ibm.com/developerworks/power/cell/`

Click on the **Docs** tab.

The following documents are available, organized by category:

Architecture

- *Cell Broadband Engine Architecture*
- *Cell Broadband Engine Registers*
- *SPU Instruction Set Architecture*

Standards

- *C/C++ Language Extensions for Cell Broadband Engine Architecture*
- *SPU Assembly Language Specification*
- *SPU Application Binary Interface Specification*
- *SIMD Math Library Specification for Cell Broadband Engine Architecture*
- *Cell Broadband Engine Linux® Reference Implementation Application Binary Interface Specification*

Programming

- *Cell Broadband Engine Programming Handbook*
- *Programming Tutorial*
- *SDK for Multicore Acceleration Version 3.0 Programmer's Guide*

Library

- *SPE Runtime Management library*
- *SPE Runtime Management library Version 1.2 to Version 2.0 Migration Guide*
- *Accelerated Library Framework for Cell Programmer's Guide and API Reference*
- *Accelerated Library Framework for Hybrid-x86 Programmer's Guide and API Reference*
- *Data Communication and Synchronization for Cell Programmer's Guide and API Reference*
- *Data Communication and Synchronization for Hybrid-x86 Programmer's Guide and API Reference*
- *SIMD Math Library Specification*
- *Monte Carlo Library API Reference Manual (Prototype)*

Installation

- *SDK for Multicore Acceleration Version 3.0 Installation Guide*

IBM® XL C/C++ Compiler and IBM XL Fortran Compiler

Detail about documentation for the compilers is available on the developerWorks Web site.

Draft comment
Should we name the documentation here? What is it?

IBM Full-System Simulator and debugging documentation

Detail about documentation for the simulator and debugging tools is available on the developerWorks Web site.

Draft comment
Should we name the documentation here? What is it?

PowerPC® Base

- *PowerPC Architecture™ Book, Version 2.02*
  - *Book I: PowerPC User Instruction Set Architecture*
  - *Book II: PowerPC Virtual Environment Architecture*
  - *Book III: PowerPC Operating Environment Architecture*
- *PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual Version 2.07c*

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Edition notices

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

developerWorks
IBM
PowerPC
PowerPC Architecture

Cell Broadband Engine and Cell BE are trademarks of Sony Computer Entertainment, Inc., in the United States, other countries, or both and is used under license therefrom

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

S

T

Readers' Comments — We'd Like to Hear from You

Software Development Kit for Multicore Acceleration
Version 3.0
SIMD Math Library Specifications for CBE
Version 2.1

Publication No. SC33-8335-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +49-731-16-3456
- Send your comments via e-mail to: eservdoc@de.ibm.com
- Send a note from the web page:

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC33-8335-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicherstrasse 220
D-71032 Boeblingen
Germany

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in USA

SC33-8335-00