Intel Math Libraries Testing and Validation Methodologies

Evgeny Gvozdev, Intel® Corporation,
evgeny.gvozdev@intel.com

International Conference Dedicated to the 100th Anniversary of the Birthday of Sergei L. Sobolev
Novosibirsk
October 8, 2008

Copyright © 2008, Intel Corporation. All rights reserved
Content

• A little bit history
• Requirements
• Design Decisions
• Results
• Summary

Intel, Intel logo, Intel. Leap ahead., Intel. Leap ahead. logo, Itanium are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others
A Little Bit History

• Story started about 10 years ago
• Intel® C Compiler
• LIBM: fast and accurate elementary math functions
• Test suites:
  – Internally used
  – Elefunt* (Elementary Function Tests)[3]
  – UCB* (Univ. California Berkeley Tests)
    http://www.netlib.org/fp/ucbtest.tgz
A Little Bit History

• Subset of functions, precisions
• Limited functionality
• Hard to extend
• Mostly random data rather than carefully selected
• Very limited special argument tests
• Black-box. Not-implementation/algorithm specific

There was no comprehensive suite for testing and verification of Elementary Math functions
Requirements

• Tests
  – Accuracy
  – Performance
  – Standards compliance
  – Function properties
  – Identities

• Real and complex
  – Float
  – Double
  – Long double (80 bits)
  – Quad (128 bits)
Requirements

- Portable (OS, compiler, processor, endian)
- Usable for testing and verification
- Expandable for testing new functions
- Operates in different FP environments
- Tunable interface, workload, output verbosity
- Flexible, extendable input data

*Need a powerful, portable, extendable, tunable Tool*
Design Decisions.  
White/black box, data driven, extendable

• Investigated each function to test
• Possible implementation algorithms and their hard paths
• Possible different algorithm paths for performance testing
• Binary arithmetic specific cases
• ~140 MB of data
• Thousands fixedvector arguments per function
• Tens/hundreds of thousand run-time generated
• Ability to set up custom data
Design Decisions.
Data driven, pre-calculated

• Initial set
  – Random, Min/Max, Denorms, Integers, ...

• Hard to round cases (Table Maker's Dilemma)[5]

• Other hard to process data
  – close to n*Pi
  – overflow/underflow thresholds
  – common Table implementation methods boundaries

• Standards conformance data

• All worst data from many various LIBMs testing collected
Design Decisions.
Data driven, run-time generated, tunable

• Random Intervals
• Neighborhood of specific points
• Test specified interval exhaustively
• Table-lookup (exhaustive n most significant bits of mantissa)
• Direct manipulations of bits in specific mantissa range
• N perturbations, K bits changed in mantissa bits range B1:B0
• N = 3, K = 2, B1 = 5, B0 = 2
• S EXP ...MMM0101MM
• S EXP ...MMM1010MM
• S EXP ...MMM0110MM
Design Decisions.

convenient, developers/managers usable design

• Command line interface with GUI on top
• Text input, text and .csv output, easy to pre/post process
  – 1.0, 3ff00000, 0x1p0
• Tunable configuration
  – Suite common config file - *function specific script* - run option
• Test specific argument and interval from command line
• Exclude specific intervals from testing
• Testing arbitrary (up to ~300 bits) precision “kernels”
• On the fly performance comparison of 2 libraries
Design Decisions.
Intel® Math Library Test Suite GUI

Function COS
Test Intervals
Total Args 727701
Tested 61700
0x11b2b548d7c50p-5
Rem.Time 0:11:09
Test/Sec 995.161

Errors
Reference 0
Flags 0
Monotony 0 (1265K)
Symmetry 0 (6198K)
MUE 0.500023097

Flags to Check
I U O Z V D

Rounding Mode(s)
H Z U D

Intel Math Libraries Testing and Validation Methodologies
Design Decisions.
GUI Errors file

<table>
<thead>
<tr>
<th>Function J0. Test Quick. Round To Nearest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR (ULPs) on Value 1193 (1193 in Data Set FixedVector)</td>
</tr>
<tr>
<td>Input Argument: 0x14077a7ebe19aap5</td>
</tr>
<tr>
<td>Computed Result: 0x1844436fffd1b9cp-30</td>
</tr>
<tr>
<td>Expected Result: 0x1844436fffd1b3cp-30</td>
</tr>
<tr>
<td>Precise Result: 0x1844436fffd1b3b84cfb6p-30</td>
</tr>
<tr>
<td>-- Ulp Error --: 96.4812055715465110</td>
</tr>
<tr>
<td>-- Result is not rounded correctly</td>
</tr>
<tr>
<td>Computed Flags: Iuozvd</td>
</tr>
<tr>
<td>Computed Errno: Unchanged</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function J0. Test Quick. Round To Nearest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR (ULPs) on Value 1197 (1197 in Data Set FixedVector)</td>
</tr>
<tr>
<td>Input Argument: 0x1599992c65d0d8ep5</td>
</tr>
<tr>
<td>Computed Result: 0x1b55d57acf830fp-51</td>
</tr>
<tr>
<td>Expected Result: 0x1b55d58656415bp-51</td>
</tr>
<tr>
<td>Precise Result: 0x1b55d58656415b75b85bp-51</td>
</tr>
<tr>
<td>-- Ulp Error --: -1.9337991645984432e+008</td>
</tr>
<tr>
<td>-- Result is not rounded correctly</td>
</tr>
<tr>
<td>Computed Flags: Iuozvd</td>
</tr>
<tr>
<td>Computed Errno: Unchanged</td>
</tr>
</tbody>
</table>
## Design Decisions.

Text Summary file

<table>
<thead>
<tr>
<th>Function</th>
<th>PASS/FAIL</th>
<th>Max Ulp error</th>
<th>Error Counts</th>
<th>Rounding</th>
<th>Arguments Spec/Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIN</td>
<td>PASS</td>
<td>0.515082</td>
<td>0 P P P ***** * *</td>
<td>Near</td>
<td>42/31102</td>
</tr>
<tr>
<td>COS</td>
<td>PASS</td>
<td>0.518672</td>
<td>0 P P P ***** * *</td>
<td>Near</td>
<td>40/30380</td>
</tr>
<tr>
<td>TAN</td>
<td>PASS</td>
<td>0.541852</td>
<td>0 P P P ***** * *</td>
<td>Near</td>
<td>42/25031</td>
</tr>
<tr>
<td>ASIN</td>
<td>FAIL</td>
<td>0.535745</td>
<td>0 P 99 N***** * *</td>
<td>Near</td>
<td>42/27973</td>
</tr>
<tr>
<td>ACOS</td>
<td>FAIL</td>
<td>0.531348</td>
<td>0 P 76 B****N * *</td>
<td>Near</td>
<td>41/29519</td>
</tr>
<tr>
<td>ACOS</td>
<td>FAIL</td>
<td>0.531348</td>
<td>0 P 40 B****N * *</td>
<td>Near</td>
<td>41/1300</td>
</tr>
<tr>
<td>ACOS</td>
<td>FAIL</td>
<td>0.999864</td>
<td>0 P 41 S****N * B</td>
<td>+Inf</td>
<td>41/1300</td>
</tr>
<tr>
<td>ACOS</td>
<td>FAIL</td>
<td>1.000545</td>
<td>0 P 40 N*****N * *</td>
<td>-Inf</td>
<td>41/1300</td>
</tr>
<tr>
<td>ACOS</td>
<td>FAIL</td>
<td>1.002306</td>
<td>0 P 40 N*****N * *</td>
<td>Zero</td>
<td>41/1300</td>
</tr>
</tbody>
</table>

*White/black box, data driven, convenient, developers/managers usable design*
Results.

- Linux*/Microsoft* Windows*/HP-UX*/Sun* Solaris* OS
- Intel® C Compiler GNU*/Microsoft*/HP-UX* compilers
- IA32/Intel® 64/Itanium®/Sun* SPARC* architectures
- Intel XScale® microarchitecture
- Intel, GLIBC, Microsoft*, HP-UX*, Correctly rounded ... LIBMs
- VML Vector Math Library. Intel® MKL
- SVML Short Vector Math Library. Intel® C Compiler
- Intel® C Compiler validation
- Intel hardware validation
Results.
Intel, GLIBC, Microsoft* LIBMs accuracy

Math functions Presence and Accuracy

- Intel
- Microsoft
- GLIBC 2.3.4
- GLIBC June 2008
Results.
Close look-up at 0.5 ULP area

Accuracy near 0.5 ULP

Error, ULPs

Functions

Intel
Microsoft
GLIBC 2.3.4
GLIBC June 2008
Results.
Most common functions comparison

Accuracy data

Error, ULPs

- Intel
- Microsoft
- GLIBC June 2008

Functions

sin, asin, tan, atan, exp, log, sinh, asinh, tanh, atanh, pow, sqrt
Results. GUI ULP chart Intel atan(). Random data.
Results. GUI ULP chart Intel atan(). Suite data.
Results. GUI ULP chart Intel asindl().
Results. GUI ULP chart Microsoft* exp().
Results. GUI ULP chart Microsoft* sinh().
Results. Auto performance charts Intel cos().
Results. Auto performance charts Intel acos().

acos() function

Cycles

Domain

-1.87E+307, -1.00E+00, -9.85E-01, -8.48E-01, -8.95E-04, 0.00E-00, 3.16E-15, 8.34E-02, 8.66E-01, 1.00E+00, 1.12E+307, 1.87E+307
Results.

Correctly Rounded (CR) LIBMs

- **IBM* Accurate Portable MathLib*. 1999, 2002. Round to nearest; No errno support; No exception flags settings sin cos tan asin acos atan exp log pow sqrt remainder atan2

- A library of fast math subroutines for Java. Abraham Ziv, Moshe Olshansky

- **Sun* libmcr* 0.9 Beta; December 2004. All round modes; No errno support; Set exception flags exp log pow sin cos tan atan

- A reference correctly-rounded library of basic double-precision transcendental elementary functions.

- **ENS-Lyon crlibm* 0.8beta, 1.0beta1; January 2005, February 2007. All round modes; No errno support; Tries to set Overflow/Underflow sin cos tan asin atan sinh cosh exp log log2 log10. LGPL.

- A mathematical library (libm) with proven, IEEE-754 compliant, correct rounding in the four rounding modes, and performances comparable to standard libms.
Results.
CR LIBMs. Issues Found

- IBM* Accurate Portable MathLib*: pow, cot, sin, tan, remainder, atan2
- Sun* libmcr*: sin, tan, atan, exp, log, pow
- ENS-Lyon crlibm*: sin, cos, tan, atan, cot, sinh, log, log2, log10

- Architectures difference
- Coding errors
- C99 Special cases (Infs, NaN, Max/Min)
- Denormal operands
- Binade boundaries
Results.
Acknowledgements

- **IBM* Accurate Portable MathLib* readme.**
  “Our thanks are given to the IMLTS (Intel Math Library Test Suite) group, and particularly to Shane Story and Eugeny Gvozdev, for testing the library several times and discovering numerous bugs which otherwise could have remained unnoticed.”

- **ENS-Lyon crlibm* .pdf doc.**
  “Many thanks to...The Intel Nizhniy-Novgorod Lab, especially Andrey Naraikin, Sergei Maidanov and Evgeny Gvozdev;”
Designers and Implementers

• Yuri Akutin
• Alexey Ershov
• Evgeny Gvozdev
• Svetlana Gvozdeva
• Vladimir Lunev
• Elena Luneva
• Victor Shumilin
• Shane Story

*Most powerful tool in industry, proven on many LIBMs including designed as correctly rounded.*
Summary

There was no comprehensive suite for testing and verification of Elementary Math functions

Need a powerful, portable, extendable, tunable Tool

White/black box, data driven, convenient, developers/managers usable design

Most powerful tool in industry, proven on many LIBMs including designed as correctly rounded.
References

Support Material

• Specific bugs found in correctly-rounded LIBMs
IBM* Accurate Portable MathLib*

Function COT. Test Quick. Round To Nearest.
Input Argument: 7be81ae0dffa3b33 --- 7.34096043773508872e+288
Computed Result: 432c0a894108ce00 --- 3.94644198289587200e+15
Expected Result: 432c0a894108cd05 --- 3.94644198289574650e+15
Precise Result: 432c0a894108cd049f2d86 --- 3.94644198289574630894e+0015

Function COT. Test Quick. Round To Nearest.
Input Argument: 41339c6fd67805a7 --- 1.28523183776889159e+06
Computed Result: c3340d0d167bc6c0 --- -5.64384939715961600e+15
Expected Result: c3340d0d167bccd6 --- -5.64384939716117400e+15
Precise Result: c3340d0d167bccd6443e1a --- -5.643849397161174266572e+0015
IBM* Accurate Portable MathLib*

Function SIN. Test Quick. Round To Nearest.
Input Argument: 3fd02ae7238a0000       --- 2.52618584352603650e-001
Computed Result: 3fcffe0b1764ca4d       --- 2.49940287039481040e-001
Expected Result: 3fcffe0b1764ca4c       --- 2.49940287039481010e-001
Precise Result: 3fcffe0b1764ca4c7d2329 --- 2.49940287039481026343739e-0001

Function REMAINDER. Test Quick. Round To Nearest.
Input Argument: bff0000000000000       --- -1.00000000000000000e+000
bff00068db8bac71       --- -1.00010000000000000e+000
Computed Result: 3f1a36e2eb1c0000       --- 9.9999999997669420e-005
Expected Result: 3f1a36e2eb1c4000       --- 9.9999999998889870e-005
Precise Result: 3f1a36e2eb1c4000000000 --- 9.9999999998898865875957e-0005
Function REMAINDER. Test Quick. Round To Nearest.

Input Argument: bfebbbbbbbbbbbbbc  ---  -8.6666666666666700e-001
                   bfd556f8c384071c  ---  -3.3343333333333470e-001

Computed Result: 3fc11ae5a6293bb0  ---  1.3363333333333490e-001
Expected Result: 3fc11ae5a6293bb8  ---  1.3363333333333710e-001
Precise Result: 3fc11ae5a6293bb8000000  ---  1.3363333333333714776359e-001

Function REMAINDER. Test Intervals. Round To Nearest.

Input Argument: bf50000000000000  ---  -9.76562500000000000e-004
                   bf2999999999999f  ---  -1.9531250000000150e-004

Computed Result: 3c30000000000000  ---  8.67361737988403550e-019
Expected Result: 3c2b000000000000  ---  7.31836466427715490e-019
Precise Result: 3c2b000000000000000000  ---  7.31836466427715492955031e-0019
**IBM* Accurate Portable MathLib***

**Function REMAINDER. Test Intervals. Round To Nearest.**

Input Argument: ffefffffffffffffff  ---  -1.79769313486231571e+308
7fefffffffffffffff  ---  1.79769313486231571e+308

Computed Result: 0000000000000000  ---  Zero
Expected Result: 8000000000000000  ---  -Zero
Precise Result: 8000000000000000000000  ---  -Zero

**Function TAN. Test Special. Round To Nearest.**

Input Argument: 7ff0000000000001  ---  SNaN

Computed Result: 0000000000000000  ---  Zero
Expected Result: 7ff8000000000001  ---  QNaN

**Function TAN. Test Special. Round To Nearest.**

Input Argument: 7ff8000000000001  ---  QNaN

Computed Result: 0000000000000000  ---  Zero
Expected Result: 7ff8000000000001  ---  QNaN
IBM* Accurate Portable MathLib*

Function TAN. Test Special. Round To Nearest.
Input Argument: 7ff0000000000000 --- Infinity
Computed Result: 0000000000000000 --- Zero
Expected Result: fff8000000000000 --- QNaN_Indefinite

Function ATAN2. Test Quick. Round To Nearest.
Input Argument: 7feffffffffffffff --- 1.79769313486231570e+308
7feffffffffffffff --- 1.79769313486231570e+308
Computed Result: fff8000000000000 --- QNaN_Indefinite
Expected Result: 3fe921fb54442d18 --- 7.85398163397448280e-001
Precise Result: 3fe921fb54442d18469898 --- 7.853981633974483096156e-001
IBM* Accurate Portable MathLib*

Function POW. Test Quick. Round To Nearest.
Input Argument: bf5000000000000000000000  --- -9.7656250000000000e-004
000000000000000000000000  --- Zero
Computed Result: 7ff8000000000000  --- QNaN
Expected Result: 3ff0000000000000000000000  --- 1.000000000000000000e+000
Precise Result: 3ff0000000000000000000000  --- 1.000000000000000000e+000

Function POW. Test Quick. Round To Nearest.
Input Argument: 7fe0000001fffffff  --- 8.988465741280865500e+307
bff0000000000000000000000  --- -1.000000000000000000e+000
Computed Result: 007fffffff000000  --- 1.112536920964554600e-308
Expected Result: 007fffffff000001  --- 1.112536920964555100e-308
Precise Result: 007fffffff0000009fffffff  --- 1.112536920964554941878050e-0308
IBM* Accurate Portable MathLib*

Function POW. Test Special. Round To Nearest.
Input Arguments: 8000000000000000 --- Zero
                  bfeffffffffffff --- -9.99999999999999890e-001
Computed Result: 7ff8000000000000 --- QNaN
Expected Result: 7ff0000000000000 --- Infinity

Function POW. Test Special. Round To Nearest.
Input Arguments: bff0000000000000 --- -1.00000000000000000e+000
                  7ff0000000000000 --- Infinity
Computed Result: 7ff8000000000000 --- QNaN
Expected Result: 3ff0000000000000 --- 1.00000000000000000e+000

Function POW. Test Special. Round To Nearest.
Input Arguments: 3ff0000000000000 --- 1.00000000000000000e+000
                  fff8000000000001 --- QNaN
Computed Result: 7ff8000000000000 --- QNaN
Expected Result: 3ff0000000000000 --- 1.00000000000000000e+000
Function SIN. Test Quick. Round To Nearest.
Input Argument: 000fffffffffffffffff  ---  2.22507385850720089e-308
Computed Result: 000fffffffffffffffffe  ---  2.22507385850720039e-308
Expected Result: 000fffffffffffffffff  ---  2.22507385850720089e-308
Precise Result: 000fffffffffffffffffefffffffff  ---  2.225073858507200889024e-308

Function TAN. Test Quick. Round To Nearest.
Input Argument: 000fffffffffffffffff  ---  2.22507385850720089e-308
Computed Result: 0010000000000000  ---  2.22507385850720138e-308
Expected Result: 000fffffffffffffffff  ---  2.22507385850720089e-308
Precise Result: 000fffffffffffffffff000000  ---  2.225073858507200889024e-308
## Sun* libmcr*

### Function ATAN. Test Quick. Round To Nearest.

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>000fffffffffffffff</td>
<td>000ffffffffffffe</td>
<td>000fffffffffffffff</td>
<td>000fffffffffffffffeffffffffff</td>
</tr>
</tbody>
</table>

### Function EXP. Test Quick. Round To Nearest.

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>bc900000000000001</td>
<td>3ff0000000000000</td>
<td>3fefffffffffffffff</td>
<td>3feffffffffffffff7fffffff</td>
</tr>
</tbody>
</table>
Sun* libmcr*

Function LOG. Test Special. Round To Nearest.
Input Argument: fff0000000000000 --- -Infinity
Computed Result: fff0000000000000 --- -Infinity
Expected Result: fff8000000000000 --- QNaN_Indefinite

Function POW. Test Quick. Round To Nearest.
Input Argument: bfe0000000022000 --- -5.00000000015461410e-01
c08e2800000000000 --- -9.65000000000000000e+02
Computed Result: bfefffffffeffac004 --- -9.99999970159479279e-01
Expected Result: fc3fffffffefac004 --- -3.11850039059032140e+290
Precise Result: fc3fffffffefac00403b0c6 --- -3.1185003905903214097063e+0290
Sun* libmcr*

Function POW. Test Quick. Round To Nearest.
Input Argument: bfe00000044000000 --- -5.00000126659870148e-01
c0861000000000000 --- -7.060000000000000000000e+02
Computed Result: 3fefe8f898fad7 --- 9.99821172277587489e-01
Expected Result: 6c0ffe8f898fad7 --- 3.36588495579233237e+212
Precise Result: 6c0ffe8f898fad6805ef5 --- 3.3658849557923321850473e+0212

Function POW. Test Quick. Round To Nearest.
Input Argument: bfefffffffffffffff --- -9.99999999999999889e-01
bff00000000000000 --- -1.000000000000000000000e+00
Computed Result: bff00000000000000 --- -1.000000000000000000000e+00
Expected Result: bff00000000000001 --- -1.00000000000000022e+00
Precise Result: bff0000000000000008000000 --- -1.0000000000000001110223e+0000
**Sun* libmcr*\**

Function POW. Test Cmdline. Round To Nearest.

Input Argument: 4050000000000013 --- 6.400000000000002700e+01
3fe0000000000000 --- 5.00000000000000000e-01

Computed Result: 4020000000000009a --- 8.000000000000001776e+00
Expected Result: 40200000000000099 --- 8.000000000000001599e+00
Precise Result: 40200000000000097ffffff --- 8.00000000000000168753899e+00

Function POW. Test Intervals. Round To Nearest.

Input Argument: 3fe0000000000000 --- 5.00000000000000000e-01
3ffffffffffffffb --- 1.99999999999999889e+00

Computed Result: 3ff0000000000000 --- 1.00000000000000000e+00
Expected Result: 3fd000000000000003 --- 2.50000000000000167e-01
Precise Result: 3fd000000000000003773a77 --- 2.5000000000000019238699e-001
## Sun* libmcr* Exception flags

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIN</td>
<td>FF**** Near</td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td>FF**** +Inf</td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td>FF**** -Inf</td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td>F***** Zero</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>***** Near</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>***** +Inf</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>***** -Inf</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>***** Zero</td>
<td></td>
</tr>
<tr>
<td>TAN</td>
<td>***** Near</td>
<td></td>
</tr>
<tr>
<td>TAN</td>
<td>***** +Inf</td>
<td></td>
</tr>
<tr>
<td>TAN</td>
<td>***** -Inf</td>
<td></td>
</tr>
<tr>
<td>TAN</td>
<td>***** Zero</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATAN</td>
<td>FF**** Near</td>
<td></td>
</tr>
<tr>
<td>ATAN</td>
<td>FF**** +Inf</td>
<td></td>
</tr>
<tr>
<td>ATAN</td>
<td>FF**** -Inf</td>
<td></td>
</tr>
<tr>
<td>ATAN</td>
<td>F***** Zero</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td><em>F</em>*** Near</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td><em>F</em>*** +Inf</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td><em>F</em>*** -Inf</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td><em>F</em>*** Zero</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>***<em>F</em> Near</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>***<em>F</em> +Inf</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>***<em>F</em> -Inf</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>***<em>F</em> Zero</td>
<td></td>
</tr>
<tr>
<td>POW</td>
<td><em>F</em>FF* Near</td>
<td></td>
</tr>
<tr>
<td>POW</td>
<td><em>F</em>FFF +Inf</td>
<td></td>
</tr>
<tr>
<td>POW</td>
<td><em>F</em>FFF -Inf</td>
<td></td>
</tr>
<tr>
<td>POW</td>
<td><em>F</em>FFF Zero</td>
<td></td>
</tr>
</tbody>
</table>
## ENS-Lyon crlibm*

**sin, cos, tan, atan, round to nearest**

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7fe0000000000000</td>
<td>0000000000000000</td>
<td>3fe205248cbdb760</td>
<td>3fe205248cbdb75fe5a5cc</td>
</tr>
<tr>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>8.98846567431157954e+307</td>
<td>5.63127779850884025e-01</td>
<td>5.6312777985088401345294e-0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7ff0000000000001</td>
<td>0000000000000000</td>
<td>7ff8000000000001</td>
<td>7ff8000000000001</td>
</tr>
<tr>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>SNaN</td>
<td>QNaN</td>
<td>QNaN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7fffffffffffffff</td>
<td>0000000000000000</td>
<td>7fffffffffffffff</td>
<td>7fffffffffffffff</td>
</tr>
<tr>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>QNaN</td>
<td>QNaN</td>
<td>QNaN</td>
</tr>
</tbody>
</table>
ENS-Lyon crlibm*

sin, cos, tan; round to nearest

Input Argument: 7ff0000000000000       ---   Infinity
Computed Result: 0000000000000000       ---   Zero
Expected Result: fff8000000000000       ---   QNaN_Indefinite

Input Argument: fff0000000000000       ---   -Infinity
Computed Result: 0000000000000000       ---   Zero
Expected Result: fff8000000000000       ---   QNaN_Indefinite
**ENS-Lyon crlibm***

Function ATAN. Test Template. Round To Nearest.

Input Argument:  43b8000000000000  ---  1.72938225691027046e+18
Computed Result:  7ff8000000000000  ---  QNaN
Expected Result:  3ff921fb54442d18  ---  1.57079632679489656e+00
Precise Result:  3ff921fb54442d185edee  ---  1.570796326794896618653e+0000

Function SINH. Test Quick. Round To Nearest.

Input Argument:  8000000000000000  ---  -Zero
Computed Result:  0000000000000000  ---  Zero
Expected Result:  8000000000000000  ---  -Zero
Precise Result:  8000000000000000000000  ---  -Zero
ENS-Lyon crlibm*

**log2, log10; round to nearest**

**Input Argument:** 00000000000000000000000000000000

**Computed Result:** 7ff0000000000000000000000000000

**Expected Result:** fff0000000000000000000000000000

--- Zero

--- Infinity

--- -Infinity

**Input Argument:** 80000000000000000000000000000000

**Computed Result:** 7ff0000000000000000000000000000

**Expected Result:** fff0000000000000000000000000000

--- -Zero

--- Infinity

--- -Infinity

**Function COT. Test Quick. Round To Nearest.**

**Input Argument:** 3ffe417c1bda4617

**Computed Result:** bfd47e031e5863fb

**Expected Result:** bfd538f5dc20b48a

**Precise Result:** bfd538f5dc20b48a613329

--- 1.89098749999937410e+00

--- 3.209173204477486e-01

--- 3.316015863118273e-01

--- 3.316015863118275550917e-001
## ENS-Lyon crlibm*

Function SIN. Test Template. Round To Zero.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010000000000001</td>
<td>0010000000000001</td>
<td>00100000000000000</td>
<td>0010000000000000000ffffff</td>
</tr>
</tbody>
</table>

Input Argument: 0010000000000001 --- 2.22507385850720188e-308  
Computed Result: 0010000000000001 --- 2.22507385850720188e-308  
Expected Result: 00100000000000000 --- 2.22507385850720138e-308  
Precise Result: 0010000000000000000ffffff --- 2.2250738585072018771558e-0308

Function SIN. Test Template. Round To -Infinity.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Computed Result</th>
<th>Expected Result</th>
<th>Precise Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010010000000000</td>
<td>0010010000000000</td>
<td>00100100000000000</td>
<td>001001000000000000000ffffff</td>
</tr>
</tbody>
</table>

Input Argument: 0010010000000000 --- 2.22561708942968849e-308  
Computed Result: 00100100000000000 --- 2.22561708942968849e-308  
Expected Result: 00100100000000000 --- 2.22561708942968800e-308  
Precise Result: 001001000000000000000ffffff --- 2.2256170894296884928029e-0308
ENS-Lyon crlibm*

Function EXP. Test Quick. Round To Zero.
Input Argument: 3fe00000000000000 --- 5.0000000000000000e-01
Computed Result: 0000000000000000 --- Zero
Expected Result: 3ffa61298e1e069b --- 1.6487212707001279e+00
Precise Result: 3ffa61298e1e069bc972df --- 1.6487212707001281468486e+0000

Function EXP. Test Intervals. Round To -Infinity.
Input Argument: bfd4bfec17d14ef8 --- -3.24214003810296969e-01
Computed Result: 7fdfffffffffffffff --- 8.98846567431157854e+307
Expected Result: 3fe72399222627910 --- 7.23095480632567345e-01
Precise Result: 3fe723992226791040c657 --- 7.2309548063256737260601e-0001
ENS-Lyon crlibm*

Function SIN. Test Quick. Round To -Infinity.
Input Argument: 3ff921fb54c42d19  --- 1.57079632865754193e+00
Computed Result: 3fefexffffffffffe  --- 9.9999999999999778e-01
Expected Result: 3fefexfffffffffff  --- 9.9999999999999889e-01
Precise Result: 3fefefffffffffffffffbefff  --- 9.99999999999999826527e-0001

Function COS. Test Intervals. Round To -Infinity.
Input Argument: 401921fb54c42d18  --- 6.28318531463016683e+00
Computed Result: 3fefexffffffffffe  --- 9.9999999999999778e-01
Expected Result: 3fefexfffffffffff  --- 9.9999999999999889e-01
Precise Result: 3fefefffffffffffffffcc00000  --- 9.99999999999997224442e-0001

Function ATAN. Test Intervals. Round To -Infinity.
Input Argument: 3e40000000000000  --- 7.45058059692382812e-09
Computed Result: 3e3fffffffffffffffbe  --- 7.45058059692382647e-09
Expected Result: 3e3fffffffffffffff  --- 7.45058059692382730e-09
Precise Result: 3e3fffffffffffffffdd55555  --- 7.4505805969238279871365e-0009
ENS-Lyon crlibm*

Function COSH. Test Intervals. Round To -Infinity.
Input Argument: 40865294a5294a53 --- 7.14322580645161338e+02
Computed Result: 7ff0000000000000 --- Infinity
Expected Result: 7feffffffffffffffffff --- 1.79769313486231571e+308
Precise Result: 7ff76b6b13dc6a52dca893 --- 8.420251771759003418637e+0309

Function LOG. Test Quick. Round To +Infinity.
Input Argument: 3ff0000000000000 --- 1.00000000000000000e+00
Computed Result: 0000000000000001 --- 4.94065645841246544e-324
Expected Result: 0000000000000000 --- Zero
Precise Result: 000000000000000000000000 --- Zero
# ENS-Lyon crlibm* Exception flags

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIN</td>
<td>FF**F*</td>
<td>Near</td>
</tr>
<tr>
<td>ASIN</td>
<td>FF****</td>
<td>Near</td>
</tr>
<tr>
<td>SIN</td>
<td>FF**F*</td>
<td>+Inf</td>
</tr>
<tr>
<td>ASIN</td>
<td>FF****</td>
<td>+Inf</td>
</tr>
<tr>
<td>SIN</td>
<td>FF**F*</td>
<td>-Inf</td>
</tr>
<tr>
<td>ASIN</td>
<td>FF****</td>
<td>-Inf</td>
</tr>
<tr>
<td>SIN</td>
<td>FF**F*</td>
<td>Zero</td>
</tr>
<tr>
<td>ASIN</td>
<td>FF****</td>
<td>Zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>F***FF</td>
<td>Near</td>
</tr>
<tr>
<td>ATAN</td>
<td>FF****</td>
<td>Near</td>
</tr>
<tr>
<td>COS</td>
<td>F***FF</td>
<td>+Inf</td>
</tr>
<tr>
<td>ATAN</td>
<td>FF****</td>
<td>+Inf</td>
</tr>
<tr>
<td>COS</td>
<td>F***FF</td>
<td>-Inf</td>
</tr>
<tr>
<td>ATAN</td>
<td>FF****</td>
<td>-Inf</td>
</tr>
<tr>
<td>COS</td>
<td>F**<em>F</em></td>
<td>Zero</td>
</tr>
<tr>
<td>ATAN</td>
<td>FF****</td>
<td>Zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAN</td>
<td>FF**F*</td>
<td>Near</td>
</tr>
<tr>
<td>EXP</td>
<td>FFF**F</td>
<td>Near</td>
</tr>
<tr>
<td>TAN</td>
<td>FF**F*</td>
<td>+Inf</td>
</tr>
<tr>
<td>EXP</td>
<td>FFF**F</td>
<td>+Inf</td>
</tr>
<tr>
<td>TAN</td>
<td>FF**F*</td>
<td>-Inf</td>
</tr>
<tr>
<td>EXP</td>
<td>FFF**F</td>
<td>-Inf</td>
</tr>
<tr>
<td>TAN</td>
<td>FF**F*</td>
<td>Zero</td>
</tr>
<tr>
<td>EXP</td>
<td>FFF**F</td>
<td>Zero</td>
</tr>
</tbody>
</table>
## ENS-Lyon crlibm* Exception flags

<table>
<thead>
<tr>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
<th>Function</th>
<th>Flags</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINH</td>
<td>F<strong>F</strong></td>
<td>Near</td>
<td>SINH</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
</tr>
<tr>
<td>SINH</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
<td>LOG2</td>
<td>F<strong>F</strong></td>
<td>Near</td>
</tr>
<tr>
<td>SINH</td>
<td>F<strong>F</strong></td>
<td>-Inf</td>
<td>LOG2</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
</tr>
<tr>
<td>SINH</td>
<td>F<strong>F</strong></td>
<td>Zero</td>
<td>LOG2</td>
<td>F<strong>F</strong></td>
<td>-Inf</td>
</tr>
<tr>
<td>COSH</td>
<td>F<strong>F</strong></td>
<td>Near</td>
<td>LOG10</td>
<td>F<strong>F</strong></td>
<td>Near</td>
</tr>
<tr>
<td>COSH</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
<td>LOG10</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
</tr>
<tr>
<td>COSH</td>
<td>F<strong>F</strong></td>
<td>-Inf</td>
<td>LOG10</td>
<td>F<strong>F</strong></td>
<td>-Inf</td>
</tr>
<tr>
<td>COSH</td>
<td>F<strong>F</strong></td>
<td>Zero</td>
<td>LOG10</td>
<td>F<strong>F</strong></td>
<td>Zero</td>
</tr>
<tr>
<td>LOG</td>
<td>*<strong>F</strong></td>
<td>Near</td>
<td>LOG</td>
<td>F<strong>F</strong></td>
<td>+Inf</td>
</tr>
<tr>
<td>LOG</td>
<td>*<strong>F</strong></td>
<td>+Inf</td>
<td>LOG</td>
<td>F<strong>F</strong></td>
<td>-Inf</td>
</tr>
<tr>
<td>LOG</td>
<td>*<strong>F</strong></td>
<td>-Inf</td>
<td>LOG</td>
<td>F<strong>F</strong></td>
<td>Zero</td>
</tr>
<tr>
<td>LOG</td>
<td>*<strong>F</strong></td>
<td>Zero</td>
<td>LOG</td>
<td>F<strong>F</strong></td>
<td>Zero</td>
</tr>
</tbody>
</table>
IBM* Accurate Portable MathLib*

- Architectures difference. First many functions failed because developer implied FP ops performed in double precision (uses appropriate FP constants). Intel processor default mode is 80 bit for Linux. Using double precision resolved problem.

- pow: for $x^{-1}$ user returns $1/x$; Compiler uses fdiv hardware instruction, double roundings occurs for Intel processor precision control set to 53 bits. It works fine for full 80 bit precision control set.

- Developer bugs. sin: error slightly more than 0.5 ULP 
cot, remainder: Big ULP error 
cot: “for some unknown reason I thought that an absolute error of $1/x$ equals to that of $x$...”

- C99 Special cases: atan2, remainder, pow, tan: processing Infs, NaNs, Max/Min FP numbers.
Sun* libmcr*

• Architecture difference again – 80 vs. 64 bits internal calculations.

  “Arguments that result in correct results on SPARC, but fail for i386 machines. These are the result of i386 machines using double-extended to do FP arithmetic. SPARC uses only 64-bit floating point registers. That being the case, several failures on i386 are the result of constants that are meant for 64-bit FP registers. The exp() and pow() failures are isolated to the i386 and I believe have to do with double rounding issues. I'm looking for solutions for these … as I type.”
Sun* libmcr*, ENS-Lyon crlibm*

- Mostly developer’s bugs.
  - Binade boundaries – change ULP value.
  - C99 Special cases.
  - Work with denormal numbers.

- “Errors replicated on SPARC, our initial development machine. These were coding errors that left holes in processing.”

- “We have a stupid bug there: for directed rounding modes I compute the ulp of a number and then add or subtract it, and of course for 1 (and other numbers that lie exactly at the boundary of a binade) I subtract a quantity that is twice the ulp of the previous binade. The solution is to add/subtract 1 to the binary integer representing the double, which always work except for zero and infty.”

- “Most errors related to special cases were stupid ones (reasoning only on positive values, < instead of <= , this kind of mistakes).”