AMD LibM - Release Notes - version 3.0.1

AMD LibM Contents

AMD LibM is a math library. It provides optimized implementation of a number of math functions from the C99 standard. The library is designed for use in 64-bit programs on x86-64 machines. Static and dynamic libraries are available for Linux(R) and Windows(R) operating systems.

The Linux(R) version of the library is built using GCC 4.5.0. The Windows(R) version is built using Microsoft(R) Visual Studio(R) 2010.

New Features in 3.0.1

In this release, support for the next generation AVX, XOP and FMA4 instruction set is added. The functions which are optimized for the above instruction set are as follows. They all have amd_ prefixed.

Scalar Functions:

Trigonometric

---

* cosf, cos, sinf, sin, tanf, tan, sincosf, sincos

Exponential & Logarithmic

---

* expf, exp, exp2f, exp2, exp10f, exp10, expm1f, expm1
* logf, log, log10f, log10, log2f, log2, log1pf, log1p

Power & Absolute value

---

* cbrtf, cbrt

Vector Functions:

Exponential

---

* vrs4_expf, vrs4_exp2f, vrs4_exp10f, vrs4_expm1f
* vrsa_expf, vrsa_exp2f, vrsa_exp10f, vrsa_expm1f
* vrd2_exp, vrd2_exp2, vrd2_exp10, vrd2_expm1
* vrda_exp, vrda_exp2, vrda_exp10, vrda_expm1

Logarithmic

---

* vrs4_logf, vrs4_log2f, vrs4_log10f, vrs4_log1pf
* vrd2_log, vrd2_log2, vrd2_log10, vrd2_log1p

Trigonometric

---

* vrs4_cosf, vrs4_sinf
* vrd2_cos, vrd2_sin
* vrd2_sincos
* vrs4_sincosf
* vrd2_tan, vrs4_tanf

This version also has some new Scalar and Vector functions. They are as follows:

* fma
* fmaf
* vrd2_pow
* vrd2_sincos
* vrd2_tan
* vrda_sincos
* vrs4_powxf
* vrs4_sincosf
* vrs4_tanf
* vrsa_powxf
* vrsa_sincosf

Version 3.0.1 Contents

The scalar functions listed below are present in the library. They all have an 'amd_' prefix.

Scalar Functions

Trigonometric

* cosf, cos, sinf, sin, tanf, tan, sincosf, sincos
* acosf, acos, asinf, asin, atanf, atan, atan2f, atan2

Hyperbolic

* coshf, cosh, sinhf, sinh, tanhf, tanh
* acoshf, acosh, asinhf, asinh, atanhf, atanh

Exponential & Logarithmic

* expf, exp, exp2f, exp2, exp10f, exp10, expm1f, expm1
* logf, log, log10f, log10, log2f, log2, log1pf, log1p
* logbf, logb, ilogbf, ilogb
* modff, modf, frexpf, frexp, ldexpf, ldexp
* scalbnf, scalbn, scalblnf, scalbln

Power & Absolute value

* powf, pow, cbrtf, cbrt, sqrtf, sqrt, hypotf, hypot
* fabsf, fabs, fma, fmaf

Nearest integer

* ceilf, ceil, floorf, floor, truncf, trunc
* rintf, rint, roundf, round, nearbyintf, nearbyint
* lrintf, lrint, llrintf, llrint
* lroundf, lround, llroundf, llround

Remainder

* fmodf, fmod, remainderf, remainder

Manipulation

* copysignf, copysign, nanf, nan, finitef, finite
* nextafterf, nextafter, nexttowardf, nexttoward

Maximum, Minimum & Difference

---

* fdimf, fdim, fmaxf, fmax, fminf, fmin

Vector Functions

Exponential

---

* vrs4_expf, vrs4_exp2f, vrs4_exp10f, vrs4_expm1f
* vrsa_expf, vrsa_exp2f, vrsa_exp10f, vrsa_expm1f
* vrd2_exp, vrd2_exp2, vrd2_exp10, vrd2_expm1
* vrda_exp, vrda_exp2, vrda_exp10, vrda_expm1

Logarithmic

---

* vrs4_logf, vrs4_log2f, vrs4_log10f, vrs4_log1pf
* vrsa_logf, vrsa_log2f, vrsa_log10f, vrsa_log1pf
* vrd2_log, vrd2_log2, vrd2_log10, vrd2_log1p
* vrda_log, vrda_log2, vrda_log10, vrda_log1p

Trigonometric

---

* vrs4_cosf, vrs4_sinf
* vrsa_cosf, vrsa_sinf
* vrd2_cos, vrd2_sin
* vrda_cos, vrda_sin
* vrd2_sincos,vrda_sincos
* vrs4_sincosf,vrsa_sincosf
* vrd2_tan, vrs4_tanf

Power

---

* vrs4_cbrtf, vrd2_cbrt, vrs4_powf, vrs4_powxf
* vrsa_cbrtf, vrda_cbrt, vrsa_powf, vrsa_powxf
* vrd2_pow

The declarations for all of these functions are as specified in the C99 standard. They are equivalent to the declarations present in the standard math header 'math.h'. The only difference is that the functions in AMD LibM have 'amd_' prefix in their names.

Using AMD LibM

Using AMD LibM is easy. Include 'amdlibm.h' in your C/C++ code and call the desired math function. If math.h has to be included, include math.h before amdlibm.h. Link in the appropriate version of the library in your program.

Refer to the examples directory for illustration.

The Linux libraries have a dependency on system math library. When linking AMD LibM, ensure it precedes system math library in the link order i.e., "-lamdlibm" should come before "-lm". Explicit linking of system math library is required when using gcc C compiler. With g++ compiler (for C++), this is not needed.

A simple trick can be used to replace existing math function calls in order to use AMD LibM. To replace a single function, for example, to replace all 'sin' function calls in existing code, use the following construct in your source code.

```
#undef sin
#define sin amd_sin
```

In order to replace all math functions to that of AMD LibM, define 'REPLACE_WITH_AMDLIBM' preprocessor macro.

Refer to the examples directory for illustration.

Reporting bugs

Visit the following URL:

http://developer.amd.com/support/KnowledgeBase/pages/HelpdeskTicketForm.aspx?Category=1&SubCategory=53