hsqd

Claude Heiland-Allen

2010–2015

Contents

1 cbits/safe_dd.c

```
#include <qd/fpu.h>
#include <qd/c_dd.h>

#define PRESERVE(S) do { unsigned int safe_fpu; fpu_fix_start(&safe_fpu); S; ↙
    ↳ fpu_fix_end(&safe_fpu); } while(0)

/* dd = dd OP dd */
#define SAFE(F) void safe_dd_##F(const double *a, const double *b, double *c) { ↙
    ↳ PRESERVE(c_dd_##F(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
SAFE(atan2)
```

```
#undef SAFE

/* dd = d OP dd */
#define SAFE(F) void safe_dd_##F##_d_dd(double a, const double *b, double *c) { PRESERVE(c_dd_##F##_d_dd(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* dd = dd OP d */
#define SAFE(F) void safe_dd_##F##_dd_d(const double *a, double b, double *c) { PRESERVE(c_dd_##F##_dd_d(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* dd = OP dd */
#define SAFE(F) void safe_dd_##F(const double *a, double *b) { PRESERVE(c_dd_##F(a,b)); }
SAFE(copy)
SAFE(sqrt)
SAFE(sqr)
SAFE(abs)
SAFE(nint)
SAFE(aint)
SAFE(floor)
SAFE(ceil)
SAFE(exp)
SAFE(log)
SAFE(log10)
SAFE(sin)
SAFE(cos)
SAFE(tan)
SAFE(asin)
SAFE(acos)
SAFE(atan)
SAFE(sinh)
SAFE(cosh)
SAFE(tanh)
SAFE(asinh)
SAFE(acosh)
SAFE(atanh)
SAFE(neg)
#undef SAFE

/* misc */
void safe_dd_copy_d(double a, double *b) { PRESERVE(c_dd_copy_d(a,b)); }
void safe_dd_npwr(const double *a, int b, double *c) { PRESERVE(c_dd_npwr(a,b,c)); }
void safe_dd_nroot(const double *a, int b, double *c) { PRESERVE(c_dd_nroot(a,b,c)); }
void safe_dd_sincos(const double *a, double *b, double *c) { PRESERVE(c_dd_sincos(a,b,c)); }
```

```
void safe_dd_sincosh(const double *a, double *b, double *c) { PRESERVE(c_dd_sincosh(a,b,c)); }
void safe_dd_comp(const double *a, const double *b, int *c) { PRESERVE(c_dd_comp(a,b,c)); }
void safe_dd_comp_dd_d(const double *a, double b, int *c) { PRESERVE(c_dd_comp_dd_d(a,b,c)); }
void safe_dd_comp_d_dd(double a, const double *b, int *c) { PRESERVE(c_dd_comp_d_dd(a,b,c)); }
void safe_dd_read(const char *a, double *b) { PRESERVE(c_dd_read(a,b)); }
void safe_dd_swrite(const double *a, int b, char *c, int d) { PRESERVE(c_dd_swrite(a,b,c,d)); }
void safe_dd_write(const double *a) { PRESERVE(c_dd_write(a)); }
void safe_dd_rand(double *a) { PRESERVE(c_dd_rand(a)); }
void safe_dd_pi(double *a) { PRESERVE(c_dd_pi(a)); }

#undef PRESERVE
```

2 cbits/safe_qd.c

```
#include <qd/fpu.h>
#include <qd/c_qd.h>

#define PRESERVE(S) do { unsigned int safe_fpu; fpu_fix_start(&safe_fpu); S; fpu_fix_end(&safe_fpu); } while(0)

/* qd = qd OP qd */
#define SAFE(F) void safe_qd_##F(const double *a, const double *b, double *c) { PRESERVE(c_qd_##F(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
SAFE(atan2)
#undef SAFE

/* qd = dd OP qd */
#define SAFE(F) void safe_qd_##F##_dd_qd(const double *a, const double *b, double *c) { PRESERVE(c_qd_##F##_dd_qd(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* qd = qd OP dd */
#define SAFE(F) void safe_qd_##F##_qd_dd(const double *a, const double *b, double *c) { PRESERVE(c_qd_##F##_qd_dd(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* dd = d OP qd */
#define SAFE(F) void safe_qd_##F##_d_qd(double a, const double *b, double *c) { PRESERVE(c_qd_##F##_d_qd(a,b,c)); }
SAFE(add)
```

```
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* dd = qd OP d */
#define SAFE(F) void safe_qd_##F##_qd_d(const double *a, double b, double *c) { PRESERVE(c_qd_##F##_qd_d(a,b,c)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* qd OP = qd */
#define SAFE(F) void safe_qd_self##F(const double *a, double *b) { PRESERVE(c_qd_self##F(a,b)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* qd OP = dd */
#define SAFE(F) void safe_qd_self##F##_dd(const double *a, double *b) { PRESERVE(c_qd_self##F##_dd(a,b)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* qd OP = d */
#define SAFE(F) void safe_qd_self##F##_d(double a, double *b) { PRESERVE(c_qd_self##F##_d(a,b)); }
SAFE(add)
SAFE(sub)
SAFE(mul)
SAFE(div)
#undef SAFE

/* qd = OP qd */
#define SAFE(F) void safe_qd_##F(const double *a, double *b) { PRESERVE(c_qd_##F(a,b)); }
SAFE(copy)
SAFE(sqrt)
SAFE(sqr)
SAFE(abs)
SAFE(nint)
SAFE(aint)
SAFE(floor)
SAFE(ceil)
SAFE(exp)
SAFE(log)
SAFE(log10)
SAFE(sin)
SAFE(cos)
```

```
SAFE(tan)
SAFE(asin)
SAFE(acos)
SAFE(atan)
SAFE(sinh)
SAFE(cosh)
SAFE(tanh)
SAFE(asinh)
SAFE(acosh)
SAFE(atanh)
SAFE(neg)
#undef SAFE

/* misc */
void safe_qd_copy_dd(const double *a, double *b) { PRESERVE(c_qd_copy_dd(a,b)); }
void safe_qd_copy_d(double a, double *b) { PRESERVE(c_qd_copy_d(a,b)); }
void safe_qd_npwr(const double *a, int b, double *c) { PRESERVE(c_qd_npwr(a,b,c)); }
void safe_qd_nroot(const double *a, int b, double *c) { PRESERVE(c_qd_nroot(a,b,c)); }
void safe_qd_sincos(const double *a, double *b, double *c) { PRESERVE(c_qd_sincos(a,b,c)); }
void safe_qd_sincosh(const double *a, double *b, double *c) { PRESERVE(c_qd_sincosh(a,b,c)); }
void safe_qd_comp(const double *a, const double *b, int *c) { PRESERVE(c_qd_comp(a,b,c)); }
void safe_qd_comp_qd_d(const double *a, double b, int *c) { PRESERVE(c_qd_comp_qd_d(a,b,c)); }
void safe_qd_comp_d_qd(double a, const double *b, int *c) { PRESERVE(c_qd_comp_d_qd(a,b,c)); }
void safe_qd_read(const char *a, double *b) { PRESERVE(c_qd_read(a,b)); }
void safe_qd_swrite(const double *a, int b, char *c, int d) { PRESERVE(c_qd_swrite(a,b,c,d)); }
void safe_qd_write(const double *a) { PRESERVE(c_qd_write(a)); }
void safe_qd_rand(double *a) { PRESERVE(c_qd_rand(a)); }
void safe_qd_pi(double *a) { PRESERVE(c_qd_pi(a)); }

#undef PRESERVE
```

3 .gitignore

```
dist
```

4 LICENSE

```
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.

    * Neither the name of Claude Heiland-Allen nor the names of other
      contributors may be used to endorse or promote products derived
      from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5 Numeric/QD/DoubleDouble.hs

```
{- |
Module      :  Numeric.QD.DoubleDouble
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

High-level interface to libqd for double-double numbers.
-}
{-# LANGUAGE DeriveDataTypeable #-}
module Numeric.QD.DoubleDouble
  ( DoubleDouble(DoubleDouble)
  , toDouble
  , fromDouble
  , sqr
  ) where

import Foreign (Ptr, alloca, castPtr, Storable(..), with)
import System.IO.Unsafe (unsafePerformIO)
import Foreign.C (CDouble, CInt)
import Data.Ratio ((%))
import Data.Bits (shiftL)
import Data.Typeable (Typeable(..))
import Numeric (readSigned, readFloat)
import Text.FShow.Raw (BinDecode(..), DecimalFormat(..), FormatStyle(Generic), ↙
    ↳ decimalFormat)

import Numeric.QD.DoubleDouble.Raw.Safe
  ( c_dd_add
  , c_dd_sub
  , c_dd_mul
  , c_dd_div
  , c_dd_pi
```

```
    , c_dd_exp
    , c_dd_sqrt
    , c_dd_log
    , c_dd_sin
    , c_dd_cos
    , c_dd_tan
    , c_dd_asin
    , c_dd_acos
    , c_dd_atan
    , c_dd_sinh
    , c_dd_cosh
    , c_dd_tanh
    , c_dd_asinh
    , c_dd_acosh
    , c_dd_atanh
    , c_dd_comp
    , c_dd_neg
    , c_dd_abs
    , c_dd_sqr
    )

-- | @'DoubleDouble' a b@ represents the unevaluated sum @a + b@.
data DoubleDouble = DoubleDouble {-# UNPACK #-} !CDouble {-# UNPACK #-} !CDouble
    ↳ deriving Typeable

instance BinDecode DoubleDouble where
  decode (DoubleDouble a b) =
    let repack (0,  0 ) (bm, be) = (bm, be)
        repack (am, ae) (0,  0 ) = (am, ae)
        repack (am, ae) (bm, be) = (am `shiftL` (ae - be) + bm, be)
    in  foldl1 repack $ map decodeFloat [a, b]
  showDigits _ = 35 -- FIXME somewhat arbitrary

-- FIXME check these
instance DecimalFormat DoubleDouble where
  nanTest (DoubleDouble a _b) = isNaN a
  infTest (DoubleDouble a _b) = isInfinite a
  negTest x = x < 0

-- | Extract the first component and convert to 'Double'.
toDouble :: DoubleDouble -> Double
toDouble (DoubleDouble a _) = realToFrac a

-- | Convert from 'Double' by pairing with 0.
fromDouble :: Double -> DoubleDouble
fromDouble a = DoubleDouble (realToFrac a) 0

instance Eq DoubleDouble where
  a == b = a `compare` b == EQ
  a /= b = a `compare` b /= EQ

instance Ord DoubleDouble where
  a `compare` b = unsafePerformIO $ with a $ \p -> with b $ \q -> alloca $ \r ->
      ↳ do
                  c_dd_comp (castPtr p) (castPtr q) (castPtr r)
                  i <- peek r
                  return $ i `compare` (0 :: CInt)
```

```
90
instance Show DoubleDouble where
  show = decimalFormat (Generic Nothing) Nothing

instance Read DoubleDouble where
95  readsPrec _ = readSigned readFloat

instance Num DoubleDouble where
  (+) = lift_dd_dd_dd c_dd_add
  (*) = lift_dd_dd_dd c_dd_mul
100  (-) = lift_dd_dd_dd c_dd_sub
  negate = lift_dd_dd c_dd_neg
  abs = lift_dd_dd c_dd_abs
  signum a = case a `compare` 0 of { LT -> -1 ; EQ -> 0 ; GT -> 1 }
  fromInteger i = fromRational (i % 1)
105
-- | Square a 'DoubleDouble' number.
sqr :: DoubleDouble -> DoubleDouble
sqr = lift_dd_dd c_dd_sqr

110 instance Fractional DoubleDouble where
  (/) = lift_dd_dd_dd c_dd_div
  recip b = 1 / b
  fromRational  k = let a = fromRational k
                        k' = k - toRational a
115                     b = fromRational k'
                    in  DoubleDouble a b

instance Real DoubleDouble where
  toRational (DoubleDouble a b) = toRational a + toRational b
120
instance RealFrac DoubleDouble where
  properFraction k = let (n, r) = properFraction (toRational k)
                     in  (n, fromRational r)
  truncate = truncate . toRational
125  round = round . toRational
  ceiling = ceiling . toRational
  floor = floor . toRational

instance Floating DoubleDouble where
130  pi = unsafePerformIO $ alloca $ \r -> c_dd_pi (castPtr r) >> peek r
  exp = lift_dd_dd c_dd_exp
  sqrt = lift_dd_dd c_dd_sqrt
  log = lift_dd_dd c_dd_log
  sin = lift_dd_dd c_dd_sin
135  cos = lift_dd_dd c_dd_cos
  tan = lift_dd_dd c_dd_tan
  asin = lift_dd_dd c_dd_asin
  acos = lift_dd_dd c_dd_acos
  atan = lift_dd_dd c_dd_atan
140  sinh = lift_dd_dd c_dd_sinh
  cosh = lift_dd_dd c_dd_cosh
  tanh = lift_dd_dd c_dd_tanh
  asinh = lift_dd_dd c_dd_asinh
  acosh = lift_dd_dd c_dd_acosh
145  atanh = lift_dd_dd c_dd_atanh
```

```
-- instance Enum DoubleDouble -- FIXME

instance Storable DoubleDouble where
  sizeOf _ = 2 * sizeOf (0 :: CDouble)
  alignment _ = alignment (0 :: CDouble)
  peek p = do
    let q = castPtr p
    a <- peekElemOff q 0
    b <- peekElemOff q 1
    return $ DoubleDouble a b
  poke p (DoubleDouble a b) = do
    let q = castPtr p
    pokeElemOff q 0 a
    pokeElemOff q 1 b

lift_dd_dd :: (Ptr CDouble -> Ptr CDouble -> IO ()) -> DoubleDouble -> ↙
    ↳ DoubleDouble
lift_dd_dd f a = unsafePerformIO $ with a $ \p -> alloca $ \r -> f (castPtr p) (↙
    ↳ castPtr r) >> peek r

lift_dd_dd_dd :: (Ptr CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()) -> ↙
    ↳ DoubleDouble -> DoubleDouble -> DoubleDouble
lift_dd_dd_dd f a b = unsafePerformIO $ with a $ \p -> with b $ \q -> alloca $ \↙
    ↳ r -> f (castPtr p) (castPtr q) (castPtr r) >> peek r
```

6 Numeric/QD/DoubleDouble/Raw.hs

```
{- |
Module      :  Numeric.QD.DoubleDouble.Raw
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Bindings to libqd for double-double numbers.
-}
module Numeric.QD.DoubleDouble.Raw
  ( module Numeric.QD.DoubleDouble.Raw.Safe
  ) where

import Numeric.QD.DoubleDouble.Raw.Safe
```

7 Numeric/QD/DoubleDouble/Raw/Safe.hs

```
{- |
Module      :  Numeric.QD.DoubleDouble.Raw.Safe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Safe bindings to libqd for double-double numbers.
```

```
    These bindings are to foreign wrappers around libqd, which set and
    restore the FPU flags correctly.
    -}
15  {-# LANGUAGE ForeignFunctionInterface #-}
    module Numeric.QD.DoubleDouble.Raw.Safe
      ( c_dd_add
      , c_dd_add_d_dd
      , c_dd_add_dd_d
20    , c_dd_sub
      , c_dd_sub_d_dd
      , c_dd_sub_dd_d
      , c_dd_mul
      , c_dd_mul_d_dd
25    , c_dd_mul_dd_d
      , c_dd_div
      , c_dd_div_d_dd
      , c_dd_div_dd_d
      , c_dd_copy
30    , c_dd_copy_d
      , c_dd_sqrt
      , c_dd_sqr
      , c_dd_abs
      , c_dd_npwr
35    , c_dd_nroot
      , c_dd_nint
      , c_dd_aint
      , c_dd_floor
      , c_dd_ceil
40    , c_dd_exp
      , c_dd_log
      , c_dd_log10
      , c_dd_sin
      , c_dd_cos
45    , c_dd_tan
      , c_dd_asin
      , c_dd_acos
      , c_dd_atan
      , c_dd_atan2
50    , c_dd_sinh
      , c_dd_cosh
      , c_dd_tanh
      , c_dd_asinh
      , c_dd_acosh
55    , c_dd_atanh
      , c_dd_sincos
      , c_dd_sincosh
      , c_dd_read
      , c_dd_swrite
60    , c_dd_neg
      , c_dd_comp
      , c_dd_comp_dd_d
      , c_dd_comp_d_dd
      , c_dd_pi
65    , c_dd_write
      , c_dd_rand
      ) where
```

```
import Foreign (Ptr)
import Foreign.C (CChar(..), CDouble(..), CInt(..))

foreign import ccall unsafe "safe_dd.h safe_dd_add" c_dd_add :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_add_d_dd" c_dd_add_d_dd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_add_dd_d" c_dd_add_dd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_sub" c_dd_sub :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_sub_d_dd" c_dd_sub_d_dd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_sub_dd_d" c_dd_sub_dd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_mul" c_dd_mul :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_mul_d_dd" c_dd_mul_d_dd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_mul_dd_d" c_dd_mul_dd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_div" c_dd_div :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_div_d_dd" c_dd_div_d_dd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_div_dd_d" c_dd_div_dd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_copy" c_dd_copy :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_copy_d" c_dd_copy_d :: CDouble ->↙
    ↳  Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_sqrt" c_dd_sqrt :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_sqr" c_dd_sqr :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_abs" c_dd_abs :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_npwr" c_dd_npwr :: Ptr CDouble ->↙
    ↳  CInt -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_nroot" c_dd_nroot :: Ptr CDouble ↙
    ↳ -> CInt -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_nint" c_dd_nint :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_aint" c_dd_aint :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_floor" c_dd_floor :: Ptr CDouble ↙
    ↳ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_ceil" c_dd_ceil :: Ptr CDouble ->↙
```

```
    ↪  Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_exp" c_dd_exp :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_log" c_dd_log :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_log10" c_dd_log10 :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_sin" c_dd_sin :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_cos" c_dd_cos :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_tan" c_dd_tan :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_asin" c_dd_asin :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_acos" c_dd_acos :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_atan" c_dd_atan :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_atan2" c_dd_atan2 :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_sinh" c_dd_sinh :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_cosh" c_dd_cosh :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_tanh" c_dd_tanh :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_asinh" c_dd_asinh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_acosh" c_dd_acosh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_atanh" c_dd_atanh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_sincos" c_dd_sincos :: Ptr ↙
    ↪ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_sincosh" c_dd_sincosh :: Ptr ↙
    ↪ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_read" c_dd_read :: Ptr CChar -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_swrite" c_dd_swrite :: Ptr ↙
    ↪ CDouble -> CInt -> Ptr CChar -> CInt -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_neg" c_dd_neg :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_comp" c_dd_comp :: Ptr CDouble ->↙
    ↪  Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_comp_dd_d" c_dd_comp_dd_d :: Ptr ↙
    ↪ CDouble -> CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_comp_d_dd" c_dd_comp_d_dd :: ↙
```

```
    ↳ CDouble -> Ptr CDouble -> Ptr CInt -> IO ()

foreign import ccall unsafe "safe_dd.h safe_dd_pi" c_dd_pi :: Ptr CDouble -> IO ↙
    ↳ ()

-- these two are almost certainly not referentially transparent
foreign import ccall unsafe "safe_dd.h safe_dd_write" c_dd_write :: Ptr CDouble ↙
    ↳ -> IO ()
foreign import ccall unsafe "safe_dd.h safe_dd_rand" c_dd_rand :: Ptr CDouble ->↙
    ↳ IO ()
```

8 Numeric/QD/DoubleDouble/Raw/Unsafe.hs

```
{- |
Module      :  Numeric.QD.DoubleDouble.Raw.Unsafe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Unsafe bindings to libqd for double-double numbers.

It is strongly recommended to use instead the
@'Numeric.QD.DoubleDouble.Raw.Safe'@ wrappers.
-}
{-# LANGUAGE ForeignFunctionInterface #-}
module Numeric.QD.DoubleDouble.Raw.Unsafe
  ( c_dd_add
  , c_dd_add_d_dd
  , c_dd_add_dd_d
  , c_dd_sub
  , c_dd_sub_d_dd
  , c_dd_sub_dd_d
  , c_dd_mul
  , c_dd_mul_d_dd
  , c_dd_mul_dd_d
  , c_dd_div
  , c_dd_div_d_dd
  , c_dd_div_dd_d
  , c_dd_copy
  , c_dd_copy_d
  , c_dd_sqrt
  , c_dd_sqr
  , c_dd_abs
  , c_dd_npwr
  , c_dd_nroot
  , c_dd_nint
  , c_dd_aint
  , c_dd_floor
  , c_dd_ceil
  , c_dd_exp
  , c_dd_log
  , c_dd_log10
  , c_dd_sin
  , c_dd_cos
```

```
45      , c_dd_tan
        , c_dd_asin
        , c_dd_acos
        , c_dd_atan
        , c_dd_atan2
50      , c_dd_sinh
        , c_dd_cosh
        , c_dd_tanh
        , c_dd_asinh
        , c_dd_acosh
55      , c_dd_atanh
        , c_dd_sincos
        , c_dd_sincosh
        , c_dd_read
        , c_dd_swrite
60      , c_dd_neg
        , c_dd_comp
        , c_dd_comp_dd_d
        , c_dd_comp_d_dd
        , c_dd_pi
65      , c_dd_write
        , c_dd_rand
        ) where

    import Foreign (Ptr)
70  import Foreign.C (CChar(..), CDouble(..), CInt(..))

    foreign import ccall unsafe "qd/c_dd.h c_dd_add" c_dd_add :: Ptr CDouble -> Ptr ↙
        ↳ CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_add_d_dd" c_dd_add_d_dd :: CDouble ↙
        ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_add_dd_d" c_dd_add_dd_d :: Ptr ↙
        ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
75
    foreign import ccall unsafe "qd/c_dd.h c_dd_sub" c_dd_sub :: Ptr CDouble -> Ptr ↙
        ↳ CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_sub_d_dd" c_dd_sub_d_dd :: CDouble ↙
        ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_sub_dd_d" c_dd_sub_dd_d :: Ptr ↙
        ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

80  foreign import ccall unsafe "qd/c_dd.h c_dd_mul" c_dd_mul :: Ptr CDouble -> Ptr ↙
        ↳ CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_mul_d_dd" c_dd_mul_d_dd :: CDouble ↙
        ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_mul_dd_d" c_dd_mul_dd_d :: Ptr ↙
        ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

    foreign import ccall unsafe "qd/c_dd.h c_dd_div" c_dd_div :: Ptr CDouble -> Ptr ↙
        ↳ CDouble -> Ptr CDouble -> IO ()
85  foreign import ccall unsafe "qd/c_dd.h c_dd_div_d_dd" c_dd_div_d_dd :: CDouble ↙
        ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
    foreign import ccall unsafe "qd/c_dd.h c_dd_div_dd_d" c_dd_div_dd_d :: Ptr ↙
        ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()

    foreign import ccall unsafe "qd/c_dd.h c_dd_copy" c_dd_copy :: Ptr CDouble -> ↙
        ↳ Ptr CDouble -> IO ()
```

```
foreign import ccall unsafe "qd/c_dd.h c_dd_copy_d" c_dd_copy_d :: CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_sqrt" c_dd_sqrt :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_sqr" c_dd_sqr :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_abs" c_dd_abs :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_npwr" c_dd_npwr :: Ptr CDouble -> ↙
    ↳ CInt -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_nroot" c_dd_nroot :: Ptr CDouble -> ↙
    ↳ CInt -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_nint" c_dd_nint :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_aint" c_dd_aint :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_floor" c_dd_floor :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_ceil" c_dd_ceil :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_exp" c_dd_exp :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_log" c_dd_log :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_log10" c_dd_log10 :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_sin" c_dd_sin :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_cos" c_dd_cos :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_tan" c_dd_tan :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_asin" c_dd_asin :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_acos" c_dd_acos :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_atan" c_dd_atan :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_atan2" c_dd_atan2 :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_sinh" c_dd_sinh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_cosh" c_dd_cosh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_tanh" c_dd_tanh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_asinh" c_dd_asinh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
```

```
foreign import ccall unsafe "qd/c_dd.h c_dd_acosh" c_dd_acosh :: Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_atanh" c_dd_atanh :: Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_sincos" c_dd_sincos :: Ptr CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_sincosh" c_dd_sincosh :: Ptr CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_read" c_dd_read :: Ptr CChar -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_swrite" c_dd_swrite :: Ptr CDouble -> CInt -> Ptr CChar -> CInt -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_neg" c_dd_neg :: Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_comp" c_dd_comp :: Ptr CDouble -> Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_comp_dd_d" c_dd_comp_dd_d :: Ptr CDouble -> CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_comp_d_dd" c_dd_comp_d_dd :: CDouble -> Ptr CDouble -> Ptr CInt -> IO ()

foreign import ccall unsafe "qd/c_dd.h c_dd_pi" c_dd_pi :: Ptr CDouble -> IO ()

-- these two are almost certainly not referentially transparent
foreign import ccall unsafe "qd/c_dd.h c_dd_write" c_dd_write :: Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_dd.h c_dd_rand" c_dd_rand :: Ptr CDouble -> IO ()
```

9 Numeric/QD/FPU/Raw/Unsafe.hs

```
{- |
Module      :  Numeric.QD.FPU.Raw.Unsafe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Unsafe FPU manipulation bindings.
-}
{-# LANGUAGE ForeignFunctionInterface #-}
module Numeric.QD.FPU.Raw.Unsafe
  ( fpu_fix_start
  , fpu_fix_end
  ) where

import Foreign (Ptr)
import Foreign.C (CInt)

foreign import ccall unsafe "qd/fpu.h fpu_fix_start" fpu_fix_start :: Ptr CInt -> IO ()
```

```
foreign import ccall unsafe "qd/fpu.h fpu_fix_end" fpu_fix_end :: Ptr CInt -> IO↙
    ↳ ()
```

10 Numeric/QD/FPU/Unsafe.hs

```
{- |
Module      :  Numeric.QD.FPU.Unsafe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Unsafe FPU manipulation functions.
-}
module Numeric.QD.FPU.Unsafe
  ( unsafePreservingFPU
  ) where

import Foreign.Marshal.Alloc (alloca)
import Control.Exception (bracket_)

import Numeric.QD.FPU.Raw.Unsafe (fpu_fix_start, fpu_fix_end)

-- | @'unsafePreservingFPU' f@ executes the computation @f@, ensuring
-- that the FPU control words are set to avoid problems from excess
-- precision.  See the libqd documentation for further details.
--
-- This function is unsafe in a threaded runtime as Haskell threads can
-- migrate between OS threads, moreover there is no checking for nested
-- calls - this results in race conditions.
--
-- Some steps can be taken to mitigate some of this badness; perhaps
-- using (for example) @'GHC.Conc.forkOnIO'@ might help.

unsafePreservingFPU :: IO a -> IO a
unsafePreservingFPU f = alloca $ \p -> bracket_ (fpu_fix_start p) (fpu_fix_end p↙
    ↳ ) f
```

11 Numeric/QD.hs

```
{- |
Module      :  Numeric.QD
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Higher-level wrapper over bindings to libqd.
-}
module Numeric.QD
  ( module Numeric.QD.DoubleDouble
  , module Numeric.QD.QuadDouble
```

```
15    ) where

import Numeric.QD.DoubleDouble hiding (toDouble, fromDouble, sqr)
import Numeric.QD.QuadDouble hiding (toDouble, fromDouble, toDoubleDouble, ↙
    ↳ fromDoubleDouble, sqr)
```

12 Numeric/QD/QuadDouble.hs

```
{- |
Module      : Numeric.QD.QuadDouble
Copyright   : (c) Claude Heiland-Allen 2011
License     : BSD3
5
Maintainer  : claude@mathr.co.uk
Stability   : unstable
Portability : portable

10  High-level interface to libqd for quad-double numbers.
-}
{-# LANGUAGE DeriveDataTypeable #-}
module Numeric.QD.QuadDouble
  ( QuadDouble(QuadDouble)
15  , toDouble
  , fromDouble
  , toDoubleDouble
  , fromDoubleDouble
  , sqr
20  ) where

import Foreign (Ptr, alloca, castPtr, Storable(..), with)
import System.IO.Unsafe (unsafePerformIO)
import Foreign.C (CDouble, CInt)
25  import Data.Bits (shiftL)
import Data.Ratio ((%))
import Data.Typeable (Typeable(..))
import Numeric (readSigned, readFloat)
import Text.FShow.Raw (BinDecode(..), DecimalFormat(..), FormatStyle(Generic), ↙
    ↳ decimalFormat)
30
import Numeric.QD.DoubleDouble (DoubleDouble(DoubleDouble))
import Numeric.QD.QuadDouble.Raw.Safe
  ( c_qd_add
  , c_qd_sub
35  , c_qd_mul
  , c_qd_div
  , c_qd_pi
  , c_qd_exp
  , c_qd_sqrt
40  , c_qd_log
  , c_qd_sin
  , c_qd_cos
  , c_qd_tan
  , c_qd_asin
45  , c_qd_acos
  , c_qd_atan
  , c_qd_sinh
  , c_qd_cosh
```

```
    , c_qd_tanh
    , c_qd_asinh
    , c_qd_acosh
    , c_qd_atanh
    , c_qd_comp
    , c_qd_neg
    , c_qd_abs
    , c_qd_sqr
    )

-- | @'QuadDouble' a b c d@ represents the unevaluated sum @a + b + c + d@.
data QuadDouble = QuadDouble {-# UNPACK #-} !CDouble {-# UNPACK #-} !CDouble {-# UNPACK #-} !CDouble {-# UNPACK #-} !CDouble deriving Typeable

instance BinDecode QuadDouble where
  decode (QuadDouble a b c d) =
    let repack (0,  0 ) (bm, be) = (bm, be)
        repack (am, ae) (0,  0 ) = (am, ae)
        repack (am, ae) (bm, be) = (am `shiftL` (ae - be) + bm, be)
    in  foldl1 repack $ map decodeFloat [a, b, c, d]
  showDigits _ = 70 -- FIXME somewhat arbitrary

-- FIXME check these
instance DecimalFormat QuadDouble where
  nanTest (QuadDouble a _ _ _) = isNaN a
  infTest (QuadDouble a _ _ _) = isInfinite a
  negTest x = x < 0

-- | Convert to 'Double'.
toDouble :: QuadDouble -> Double
toDouble (QuadDouble a _ _ _) = realToFrac a

-- | Convert from 'Double'.
fromDouble :: Double -> QuadDouble
fromDouble a = QuadDouble (realToFrac a) 0 0 0

-- | Convert to 'DoubleDouble'.
toDoubleDouble :: QuadDouble -> DoubleDouble
toDoubleDouble (QuadDouble a b _ _) = DoubleDouble a b

-- | Convert from 'DoubleDouble'.
fromDoubleDouble :: DoubleDouble -> QuadDouble
fromDoubleDouble (DoubleDouble a b) = QuadDouble a b 0 0

instance Eq QuadDouble where
  a == b = a `compare` b == EQ
  a /= b = a `compare` b /= EQ

instance Ord QuadDouble where
  a `compare` b = unsafePerformIO $ with a $ \p -> with b $ \q -> alloca $ \r ->
      do
                    c_qd_comp (castPtr p) (castPtr q) (castPtr r)
                    i <- peek r
                    return $ i `compare` (0 :: CInt)

instance Num QuadDouble where
  (+) = lift_qd_qd_qd c_qd_add
```

```
    (*) = lift_qd_qd_qd c_qd_mul
    (-) = lift_qd_qd_qd c_qd_sub
    negate = lift_qd_qd c_qd_neg
    abs = lift_qd_qd c_qd_abs
    signum a = case a `compare` 0 of { LT -> -1 ; EQ -> 0 ; GT -> 1 }
    fromInteger i = fromRational (i % 1)

-- | Square a 'QuadDouble' number.
sqr :: QuadDouble -> QuadDouble
sqr = lift_qd_qd c_qd_sqr

instance Fractional QuadDouble where
    (/) = lift_qd_qd_qd c_qd_div
    recip b = 1 / b
    fromRational  k = let a = fromRational k
                          ka = k  - toRational a
                          b = fromRational ka
                          kb = ka - toRational b
                          c = fromRational kb
                          kc = kb - toRational c
                          d = fromRational kc
                      in  QuadDouble a b c d

instance Real QuadDouble where
    toRational (QuadDouble a b c d) = toRational a + toRational b + toRational c +↙
        ↳ toRational d

instance RealFrac QuadDouble where
    properFraction k = let (n, r) = properFraction (toRational k)
                       in  (n, fromRational r)
    truncate = truncate . toRational
    round = round . toRational
    ceiling = ceiling . toRational
    floor = floor . toRational

instance Floating QuadDouble where
    pi = unsafePerformIO $ alloca $ \r -> c_qd_pi (castPtr r) >> peek r
    exp = lift_qd_qd c_qd_exp
    sqrt = lift_qd_qd c_qd_sqrt
    log = lift_qd_qd c_qd_log
    sin = lift_qd_qd c_qd_sin
    cos = lift_qd_qd c_qd_cos
    tan = lift_qd_qd c_qd_tan
    asin = lift_qd_qd c_qd_asin
    acos = lift_qd_qd c_qd_acos
    atan = lift_qd_qd c_qd_atan
    sinh = lift_qd_qd c_qd_sinh
    cosh = lift_qd_qd c_qd_cosh
    tanh = lift_qd_qd c_qd_tanh
    asinh = lift_qd_qd c_qd_asinh
    acosh = lift_qd_qd c_qd_acosh
    atanh = lift_qd_qd c_qd_atanh

-- instance Enum QuadDouble -- FIXME

instance Show QuadDouble where
    show = decimalFormat (Generic Nothing) Nothing
```

```
160
instance Read QuadDouble where
  readsPrec _ = readSigned readFloat

instance Storable QuadDouble where
165   sizeOf _ = 4 * sizeOf (0 :: CDouble)
  alignment _ = alignment (0 :: CDouble)
  peek p = do
    let q = castPtr p
    a <- peekElemOff q 0
170   b <- peekElemOff q 1
    c <- peekElemOff q 2
    d <- peekElemOff q 3
    return $ QuadDouble a b c d
  poke p (QuadDouble a b c d) = do
175   let q = castPtr p
    pokeElemOff q 0 a
    pokeElemOff q 1 b
    pokeElemOff q 2 c
    pokeElemOff q 3 d
180
lift_qd_qd :: (Ptr CDouble -> Ptr CDouble -> IO ()) -> QuadDouble -> QuadDouble
lift_qd_qd f a = unsafePerformIO $ with a $ \p -> alloca $ \r -> f (castPtr p) (↙
    ↪ castPtr r) >> peek r

lift_qd_qd_qd :: (Ptr CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()) -> ↙
    ↪ QuadDouble -> QuadDouble -> QuadDouble
185 lift_qd_qd_qd f a b = unsafePerformIO $ with a $ \p -> with b $ \q -> alloca $ \↙
    ↪ r -> f (castPtr p) (castPtr q) (castPtr r) >> peek r
```

13 Numeric/QD/QuadDouble/Raw.hs

```
{- |
Module      :  Numeric.QD.QuadDouble.Raw
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3
5
Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

10 Raw bindings to libqd for quad-double.
-}
module Numeric.QD.QuadDouble.Raw
  ( module Numeric.QD.QuadDouble.Raw.Safe
  ) where
15
import Numeric.QD.QuadDouble.Raw.Safe
```

14 Numeric/QD/QuadDouble/Raw/Safe.hs

```
{- |
Module      :  Numeric.QD.QuadDouble.Raw.Safe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3
5
```

```
    Maintainer  :  claude@mathr.co.uk
    Stability   :  unstable
    Portability :  portable

10  Safe bindings to libqd for quad-double numbers.

    These bindings are to foreign wrappers around libqd, which set and
    restore the FPU flags correctly.
    -}
15  {-# LANGUAGE ForeignFunctionInterface #-}
    module Numeric.QD.QuadDouble.Raw.Safe
      ( c_qd_add
      , c_qd_add_dd_qd
      , c_qd_add_qd_dd
20    , c_qd_add_d_qd
      , c_qd_add_qd_d
      , c_qd_selfadd
      , c_qd_selfadd_dd
      , c_qd_selfadd_d
25    , c_qd_sub
      , c_qd_sub_dd_qd
      , c_qd_sub_qd_dd
      , c_qd_sub_d_qd
      , c_qd_sub_qd_d
30    , c_qd_selfsub
      , c_qd_selfsub_dd
      , c_qd_selfsub_d
      , c_qd_mul
      , c_qd_mul_dd_qd
35    , c_qd_mul_qd_dd
      , c_qd_mul_d_qd
      , c_qd_mul_qd_d
      , c_qd_selfmul
      , c_qd_selfmul_dd
40    , c_qd_selfmul_d
      , c_qd_div
      , c_qd_div_dd_qd
      , c_qd_div_qd_dd
      , c_qd_div_d_qd
45    , c_qd_div_qd_d
      , c_qd_selfdiv
      , c_qd_selfdiv_dd
      , c_qd_selfdiv_d
      , c_qd_copy
50    , c_qd_copy_dd
      , c_qd_copy_d
      , c_qd_sqrt
      , c_qd_sqr
      , c_qd_abs
55    , c_qd_npwr
      , c_qd_nroot
      , c_qd_nint
      , c_qd_aint
      , c_qd_floor
60    , c_qd_ceil
      , c_qd_exp
      , c_qd_log
```

```
    , c_qd_log10
    , c_qd_sin
    , c_qd_cos
    , c_qd_tan
    , c_qd_asin
    , c_qd_acos
    , c_qd_atan
    , c_qd_atan2
    , c_qd_sinh
    , c_qd_cosh
    , c_qd_tanh
    , c_qd_asinh
    , c_qd_acosh
    , c_qd_atanh
    , c_qd_sincos
    , c_qd_sincosh
    , c_qd_read
    , c_qd_swrite
    , c_qd_neg
    , c_qd_comp
    , c_qd_comp_qd_d
    , c_qd_comp_d_qd
    , c_qd_pi
    , c_qd_write
    , c_qd_rand
    ) where

import Foreign (Ptr)
import Foreign.C (CChar(..), CDouble(..), CInt(..))

foreign import ccall unsafe "safe_qd.h safe_qd_add" c_qd_add :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_add_dd_qd" c_qd_add_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_add_qd_dd" c_qd_add_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_add_d_qd" c_qd_add_d_qd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_add_qd_d" c_qd_add_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfadd" c_qd_selfadd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfadd_dd" c_qd_selfadd_dd :: ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfadd_d" c_qd_selfadd_d :: ↙
    ↳ CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_sub" c_qd_sub :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_sub_dd_qd" c_qd_sub_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_sub_qd_dd" c_qd_sub_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_sub_d_qd" c_qd_sub_d_qd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_sub_qd_d" c_qd_sub_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
```

```
foreign import ccall unsafe "safe_qd.h safe_qd_selfsub" c_qd_selfsub :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfsub_dd" c_qd_selfsub_dd :: ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfsub_d" c_qd_selfsub_d :: ↙
    ↳ CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_mul" c_qd_mul :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_mul_dd_qd" c_qd_mul_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_mul_qd_dd" c_qd_mul_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_mul_d_qd" c_qd_mul_d_qd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_mul_qd_d" c_qd_mul_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfmul" c_qd_selfmul :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfmul_dd" c_qd_selfmul_dd :: ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfmul_d" c_qd_selfmul_d :: ↙
    ↳ CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_div" c_qd_div :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_div_dd_qd" c_qd_div_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_div_qd_dd" c_qd_div_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "safe_qd.h safe_qd_div_d_qd" c_qd_div_d_qd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_div_qd_d" c_qd_div_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfdiv" c_qd_selfdiv :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfdiv_dd" c_qd_selfdiv_dd :: ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_selfdiv_d" c_qd_selfdiv_d :: ↙
    ↳ CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_copy" c_qd_copy :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_copy_dd" c_qd_copy_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_copy_d" c_qd_copy_d :: CDouble ->↙
    ↳  Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_sqrt" c_qd_sqrt :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_sqr" c_qd_sqr :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_abs" c_qd_abs :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_npwr" c_qd_npwr :: Ptr CDouble ->↙
```

```
    ↪ CInt -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_nroot" c_qd_nroot :: Ptr CDouble ↙
    ↪ -> CInt -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_nint" c_qd_nint :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_aint" c_qd_aint :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_floor" c_qd_floor :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_ceil" c_qd_ceil :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_exp" c_qd_exp :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_log" c_qd_log :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_log10" c_qd_log10 :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_sin" c_qd_sin :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_cos" c_qd_cos :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_tan" c_qd_tan :: Ptr CDouble -> ↙
    ↪ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_asin" c_qd_asin :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_acos" c_qd_acos :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_atan" c_qd_atan :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_atan2" c_qd_atan2 :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> Ptr CDouble -> IO ();

foreign import ccall unsafe "safe_qd.h safe_qd_sinh" c_qd_sinh :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_cosh" c_qd_cosh :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_tanh" c_qd_tanh :: Ptr CDouble ->↙
    ↪ Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_asinh" c_qd_asinh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_acosh" c_qd_acosh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_atanh" c_qd_atanh :: Ptr CDouble ↙
    ↪ -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_sincos" c_qd_sincos :: Ptr ↙
    ↪ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_sincosh" c_qd_sincosh :: Ptr ↙
    ↪ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "safe_qd.h safe_qd_read" c_qd_read :: Ptr CChar -> ↙
    ↪ Ptr CDouble -> IO ()
```

```
foreign import ccall unsafe "safe_qd.h safe_qd_swrite" c_qd_swrite :: Ptr ↙
    ↳ CDouble -> CInt -> Ptr CChar -> CInt -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_neg" c_qd_neg :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_comp" c_qd_comp :: Ptr CDouble ->↙
    ↳  Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_comp_qd_d" c_qd_comp_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_comp_d_qd" c_qd_comp_d_qd :: ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_pi" c_qd_pi :: Ptr CDouble -> IO ↙
    ↳ ()

-- these two are almost certainly not referentially transparent
foreign import ccall unsafe "safe_qd.h safe_qd_write" c_qd_write :: Ptr CDouble ↙
    ↳ -> IO ()
foreign import ccall unsafe "safe_qd.h safe_qd_rand" c_qd_rand :: Ptr CDouble ->↙
    ↳  IO ()
```

15 Numeric/QD/QuadDouble/Raw/Unsafe.hs

```
{- |
Module      :  Numeric.QD.QuadDouble.Raw.Unsafe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Unsafe bindings to libqd for quad-double numbers.

It is strongly recommended to use instead the
@'Numeric.QD.QuadDouble.Raw.Safe'@ wrappers.
-}
{-# LANGUAGE ForeignFunctionInterface #-}
module Numeric.QD.QuadDouble.Raw.Unsafe
  ( c_qd_add
  , c_qd_add_dd_qd
  , c_qd_add_qd_dd
  , c_qd_add_d_qd
  , c_qd_add_qd_d
  , c_qd_selfadd
  , c_qd_selfadd_dd
  , c_qd_selfadd_d
  , c_qd_sub
  , c_qd_sub_dd_qd
  , c_qd_sub_qd_dd
  , c_qd_sub_d_qd
  , c_qd_sub_qd_d
  , c_qd_selfsub
  , c_qd_selfsub_dd
  , c_qd_selfsub_d
  , c_qd_mul
  , c_qd_mul_dd_qd
  , c_qd_mul_qd_dd
  , c_qd_mul_d_qd
```

```
      , c_qd_mul_qd_d
      , c_qd_selfmul
      , c_qd_selfmul_dd
      , c_qd_selfmul_d
      , c_qd_div
      , c_qd_div_dd_qd
      , c_qd_div_qd_dd
      , c_qd_div_d_qd
      , c_qd_div_qd_d
      , c_qd_selfdiv
      , c_qd_selfdiv_dd
      , c_qd_selfdiv_d
      , c_qd_copy
      , c_qd_copy_dd
      , c_qd_copy_d
      , c_qd_sqrt
      , c_qd_sqr
      , c_qd_abs
      , c_qd_npwr
      , c_qd_nroot
      , c_qd_nint
      , c_qd_aint
      , c_qd_floor
      , c_qd_ceil
      , c_qd_exp
      , c_qd_log
      , c_qd_log10
      , c_qd_sin
      , c_qd_cos
      , c_qd_tan
      , c_qd_asin
      , c_qd_acos
      , c_qd_atan
      , c_qd_atan2
      , c_qd_sinh
      , c_qd_cosh
      , c_qd_tanh
      , c_qd_asinh
      , c_qd_acosh
      , c_qd_atanh
      , c_qd_sincos
      , c_qd_sincosh
      , c_qd_read
      , c_qd_swrite
      , c_qd_neg
      , c_qd_comp
      , c_qd_comp_qd_d
      , c_qd_comp_d_qd
      , c_qd_pi
      , c_qd_write
      , c_qd_rand
      ) where

import Foreign (Ptr)
import Foreign.C (CChar(..), CDouble(..), CInt(..))

foreign import ccall unsafe "qd/c_qd.h c_qd_add" c_qd_add :: Ptr CDouble -> Ptr ↙
```

```
    ↳ CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_add_dd_qd" c_qd_add_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_add_qd_dd" c_qd_add_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_add_d_qd" c_qd_add_d_qd :: CDouble ↙
    ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_add_qd_d" c_qd_add_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfadd" c_qd_selfadd :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfadd_dd" c_qd_selfadd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfadd_d" c_qd_selfadd_d :: CDouble↙
    ↳  -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_sub" c_qd_sub :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_sub_dd_qd" c_qd_sub_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_sub_qd_dd" c_qd_sub_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_sub_d_qd" c_qd_sub_d_qd :: CDouble ↙
    ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_sub_qd_d" c_qd_sub_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfsub" c_qd_selfsub :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfsub_dd" c_qd_selfsub_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfsub_d" c_qd_selfsub_d :: CDouble↙
    ↳  -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_mul" c_qd_mul :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_mul_dd_qd" c_qd_mul_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_mul_qd_dd" c_qd_mul_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_mul_d_qd" c_qd_mul_d_qd :: CDouble ↙
    ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_mul_qd_d" c_qd_mul_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfmul" c_qd_selfmul :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfmul_dd" c_qd_selfmul_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfmul_d" c_qd_selfmul_d :: CDouble↙
    ↳  -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_div" c_qd_div :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_div_dd_qd" c_qd_div_dd_qd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_div_qd_dd" c_qd_div_qd_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> Ptr CDouble -> IO ();
foreign import ccall unsafe "qd/c_qd.h c_qd_div_d_qd" c_qd_div_d_qd :: CDouble ↙
```

```
    ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_div_qd_d" c_qd_div_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfdiv" c_qd_selfdiv :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfdiv_dd" c_qd_selfdiv_dd :: Ptr ↙
    ↳ CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_selfdiv_d" c_qd_selfdiv_d :: CDouble↙
    ↳  -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_copy" c_qd_copy :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_copy_dd" c_qd_copy_dd :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_copy_d" c_qd_copy_d :: CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_sqrt" c_qd_sqrt :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_sqr" c_qd_sqr :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_abs" c_qd_abs :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_npwr" c_qd_npwr :: Ptr CDouble -> ↙
    ↳ CInt -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_nroot" c_qd_nroot :: Ptr CDouble -> ↙
    ↳ CInt -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_nint" c_qd_nint :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_aint" c_qd_aint :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_floor" c_qd_floor :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_ceil" c_qd_ceil :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_exp" c_qd_exp :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_log" c_qd_log :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_log10" c_qd_log10 :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_sin" c_qd_sin :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_cos" c_qd_cos :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_tan" c_qd_tan :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_asin" c_qd_asin :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_acos" c_qd_acos :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
```

```
foreign import ccall unsafe "qd/c_qd.h c_qd_atan" c_qd_atan :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_atan2" c_qd_atan2 :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CDouble -> IO ();

foreign import ccall unsafe "qd/c_qd.h c_qd_sinh" c_qd_sinh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_cosh" c_qd_cosh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_tanh" c_qd_tanh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_asinh" c_qd_asinh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_acosh" c_qd_acosh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_atanh" c_qd_atanh :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_sincos" c_qd_sincos :: Ptr CDouble ↙
    ↳ -> Ptr CDouble -> Ptr CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_sincosh" c_qd_sincosh :: Ptr CDouble↙
    ↳  -> Ptr CDouble -> Ptr CDouble -> IO ()

foreign import ccall unsafe "qd/c_qd.h c_qd_read" c_qd_read :: Ptr CChar -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_swrite" c_qd_swrite :: Ptr CDouble ↙
    ↳ -> CInt -> Ptr CChar -> CInt -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_neg" c_qd_neg :: Ptr CDouble -> Ptr ↙
    ↳ CDouble -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_comp" c_qd_comp :: Ptr CDouble -> ↙
    ↳ Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_comp_qd_d" c_qd_comp_qd_d :: Ptr ↙
    ↳ CDouble -> CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_comp_d_qd" c_qd_comp_d_qd :: CDouble↙
    ↳  -> Ptr CDouble -> Ptr CInt -> IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_pi" c_qd_pi :: Ptr CDouble -> IO ()

-- these two are almost certainly not referentially transparent
foreign import ccall unsafe "qd/c_qd.h c_qd_write" c_qd_write :: Ptr CDouble -> ↙
    ↳ IO ()
foreign import ccall unsafe "qd/c_qd.h c_qd_rand" c_qd_rand :: Ptr CDouble -> IO↙
    ↳  ()
```

16 Numeric/QD/Raw.hs

```
{- |
Module      :  Numeric.QD.Raw
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Raw bindings to libqd.
-}
```

```
module Numeric.QD.Raw
  ( module Numeric.QD.Raw.Safe
  ) where

import Numeric.QD.Raw.Safe
```

17 Numeric/QD/Raw/Safe.hs

```
{- |
Module      :  Numeric.QD.QuadDouble.Safe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Safe bindings to libqd.

These bindings are to foreign wrappers around libqd, which set and
restore the FPU flags correctly.
-}
module Numeric.QD.Raw.Safe
  ( module Numeric.QD.DoubleDouble.Raw.Safe
  , module Numeric.QD.QuadDouble.Raw.Safe
  ) where

import Numeric.QD.DoubleDouble.Raw.Safe
import Numeric.QD.QuadDouble.Raw.Safe
```

18 Numeric/QD/Raw/Unsafe.hs

```
{- |
Module      :  Numeric.QD.Raw.Unsafe
Copyright   :  (c) Claude Heiland-Allen 2011
License     :  BSD3

Maintainer  :  claude@mathr.co.uk
Stability   :  unstable
Portability :  portable

Unsafe bindings to libqd.

It is strongly recommended to use instead the
@'Numeric.QD.Raw.Safe'@ wrappers.
-}
module Numeric.QD.Raw.Unsafe
  ( module Numeric.QD.DoubleDouble.Raw.Unsafe
  , module Numeric.QD.FPU.Raw.Unsafe
  , module Numeric.QD.QuadDouble.Raw.Unsafe
  ) where

import Numeric.QD.DoubleDouble.Raw.Unsafe
import Numeric.QD.FPU.Raw.Unsafe
import Numeric.QD.QuadDouble.Raw.Unsafe
```

19 qd.cabal

```
Name:                qd
Version:             1.0.2.1
Synopsis:            double-double and quad-double number type via libqd
Description:
    This package supports both a double-double datatype (approx. 32 decimal digits)
    and a quad-double datatype (approx. 64 decimal digits), using libqd (which is
    implemented in C++ with C and Fortran wrappers).  To compile this package you
    need libqd to be installed.
    .
    @'Numeric.QD.DoubleDouble.DoubleDouble'@ and @'Numeric.QD.QuadDouble.QuadDouble'@
    are strict tuples of @CDouble@s, with instances of:
    @'BinDecode'@, @'DecimalFormat'@, @'Eq'@, @'Floating'@, @'Fractional'@, @'Num'@,
    @'Ord'@, @'Read'@, @'Real'@, @'RealFrac'@, @'Show'@, @'Storable'@, @'Typeable'@.
    .
    Additional note: libqd depends on 64bit doubles, while some FPU architectures
    use 80bit.  When using the Unsafe modules this might cause erroneous results;
    the Safe modules (used by the instances above) set and restore the FPU flags
    in foreign code to avoid race conditions from pre-emptive Haskell threading.

License:             BSD3
License-file:        LICENSE
Author:              Claude Heiland-Allen
Maintainer:          claude@mathr.co.uk
Category:            Math
Build-type:          Simple
Cabal-version:       >=1.6

Library
  Build-depends:     base >= 4 && < 5, floatshow >= 0.2 && < 0.3
  Extra-Libraries:   qd
  Exposed-modules:   Numeric.QD
                     Numeric.QD.DoubleDouble
                     Numeric.QD.QuadDouble
                     Numeric.QD.Raw
                     Numeric.QD.Raw.Safe
                     Numeric.QD.Raw.Unsafe
                     Numeric.QD.FPU.Raw.Unsafe
                     Numeric.QD.FPU.Unsafe
                     Numeric.QD.DoubleDouble.Raw
                     Numeric.QD.DoubleDouble.Raw.Safe
                     Numeric.QD.DoubleDouble.Raw.Unsafe
                     Numeric.QD.QuadDouble.Raw
                     Numeric.QD.QuadDouble.Raw.Safe
                     Numeric.QD.QuadDouble.Raw.Unsafe
  C-sources:         cbits/safe_dd.c cbits/safe_qd.c
  GHC-options:       -Wall -fno-excess-precision
  GHC-prof-options:  -prof -auto-all -caf-all
```

```
source-repository head
  type:     git
  location: http://code.mathr.co.uk/hsqd.git

source-repository this
  type:     git
  location: http://code.mathr.co.uk/hsqd.git
  tag:      v1.0.2.1
```

20 Setup.hs

```
import Distribution.Simple
main = defaultMain
```

21 tests/DDdecodeFloat.hs

```
{-# LANGUAGE ScopedTypeVariables #-}

import Data.Bits (shiftL)
import Control.Monad (replicateM_, when)
import System.Random (randomRIO)
import Numeric.QD (DoubleDouble, unsafePreservingFPU)

rand :: forall a . RealFloat a => IO a
rand = do
  let ff = floatDigits (0 :: a)
      mMin = 1 `shiftL` (ff - 1)
      mMax = (1 `shiftL` ff) - 1
      rr = floatRange (0 :: a)
  sign <- randomRIO (-1, 1 :: Int)
  if (sign == 0)
    then return $ encodeFloat 0 0
    else do
      bits <- randomRIO (mMin, mMax)
      expo <- randomRIO (fst rr + 300, snd rr - 300)
      return $ encodeFloat (fromIntegral sign * bits) expo

main :: IO ()
main = unsafePreservingFPU $ replicateM_ 1000000 $ do
  f <- rand
  when (uncurry encodeFloat ({-# SCC "decode" #-} decodeFloat f) /= f) $ print (f :: DoubleDouble)
```

22 tests/QDdecodeFloat.hs

```
{-# LANGUAGE ScopedTypeVariables #-}

import Data.Bits (shiftL)
import Control.Monad (replicateM_, when)
import System.Random (randomRIO)
import Numeric.QD (QuadDouble, unsafePreservingFPU)

rand :: forall a . RealFloat a => IO a
rand = do
  let ff = floatDigits (0 :: a)
```

```
      mMin = 1 `shiftL` (ff - 1)
      mMax = (1 `shiftL` ff) - 1
      rr = floatRange (0 :: a)
  sign <- randomRIO (-1, 1 :: Int)
  if (sign == 0)
    then return $ encodeFloat 0 0
    else do
      bits <- randomRIO (mMin, mMax)
      expo <- randomRIO (fst rr + 300, snd rr - 300)
      return $ encodeFloat (fromIntegral sign * bits) expo

main :: IO ()
main = unsafePreservingFPU $ replicateM_ 1000000 $ do
  f <- rand
  when (uncurry encodeFloat ({-# SCC "decode" #-} decodeFloat f) /= f) $ print (f :: QuadDouble)
```

23 tests/Ray.hs

```
{-# LANGUAGE CPP #-}
import Control.Monad (forM_)
import Data.Complex (Complex((:+)), mkPolar)
import Data.Ratio ((%))
import Numeric.QD

type N = Int
type Q = Rational
type R = number
type C = Complex R

radius :: R
radius = 2 ** 24

sharpness :: N
sharpness = 4

limit :: N
limit = 64

ray :: Q -> [C]
ray angle = map fst . iterate (step angle) $ (mkPolar radius (2 * pi * fromRational angle), (0, 0))

step :: Q -> (C, (N, N)) -> (C, (N, N))
step angle (c, (k0, j0))
  | j > sharpness = step angle (c, (k0 + 1, 0))
  | otherwise = (c', (k0, j0 + 1))
  where
    k = k0 + 1
    j = j0 + 1
    m = (k - 1) * sharpness + j
    r = radius ** ((1/2) ** (fromIntegral m / fromIntegral sharpness))
    t = mkPolar (r ** (2 ** fromIntegral k0)) ((2 ** fromIntegral k0) * 2 * pi * fromRational angle)
    c' = iterate n c !! limit
    n z = z - (cc - t) / dd
     where
```

```
      (cc, dd) = ncnd k
      ncnd 1 = (z, 1)
      ncnd i = let (nc, nd) = ncnd (i - 1) in (nc * nc + z, 2 * nc * nd + 1)

main :: IO ()
main = unsafePreservingFPU $ do
  forM_ [2 .. 7] $ \ i -> do
    let d = 2 ^ i - 1
    forM_ [1 .. (d - 1)] $ \ n -> do
      putStrLn $ "# " ++ show n ++ "/" ++ show d
      let angle = n % d
          rs = ray angle
          printC (x:+y) = putStrLn (show x ++ " " ++ show y)
      mapM_ printC (take 100 rs)
      putStrLn ""
      putStrLn ""
```

24 tests/Ray.sh

```
#!/bin/bash
for number in "Float" "Double" "DoubleDouble" "QuadDouble"
do
  ghc --make -fforce-recomp -O3 -Dnumber="${number}" Ray.hs -o Ray-"${number}"
  time ./Ray-"${number}" >/dev/null
done
```

25 tests/readshow.hs

```
import Data.Ratio ((%))
import Data.IORef (newIORef, readIORef, writeIORef)
import Control.Monad (replicateM_, when)
import System.Environment (getArgs)
import System.Random (randomRIO)
import Numeric.QD (DoubleDouble, QuadDouble)

rand :: IO Rational
rand = do
  let bits = 300 :: Integer
  num <- randomRIO (1, 2^bits)
  den <- randomRIO (1, 2^bits)
  sign <- signum `fmap` randomRIO (-100, 100)
  return $ (sign * num) % den

main :: IO ()
main = do
  [n] <- map read `fmap` getArgs
  dr <- newIORef (0 :: Int)
  qr <- newIORef (0 :: Int)
  replicateM_ n $ do
    f <- rand
    let d = fromRational f :: DoubleDouble
        q = fromRational f :: QuadDouble
        ds = show d
        qs = show q
    when (show (read ds :: DoubleDouble) /= ds) $ do
      m <- readIORef dr
```

```
        writeIORef dr $! m + 1
      when (show (read qs :: QuadDouble) /= qs) $ do
        m <- readIORef qr
        writeIORef qr $! m + 1
    ds <- readIORef dr
    qs <- readIORef qr
    print ( 100 * fromIntegral ds / fromIntegral n :: Double
          , 100 * fromIntegral qs / fromIntegral n :: Double )
```