Compiler Options Affecting Floating-Point Arithmetic

Jeff Arnold

6 May 2014

Agenda

- Value Safety
- Subnormals
- Math Libraries
- Exercises

A Note on Compiler Options

- There are many compiler options which affect floating-point results
- Not all of them are obvious
- Some of them are enabled/disabled by other options
  - `-On`
  - `-march` and others which specify platform characteristics
- Options differ among compilers

Value Safety

"Value Safety" refers to transformations which, while algebraically valid, may affect floating-point results.

"Value Safety" requires that optimizations which could conceivably change the result of any floating-point calculation as specified by the programming language used in any possible way are disallowed.

- Changes to underflow or overflow behavior
- Effects of an operand which is not a normal floating-point number. E.g., $\pm\infty$ or a NaN

Value Safety

In "safe" mode, the compiler may not make changes such as

$(x + y) + z \Leftrightarrow x + (y + z)$	Reassociations are not value-safe
$x * (y + z) \Leftrightarrow x * y + x * z$	Distributions are not value-safe
$x/x \Leftrightarrow 1.0$	x may be 0, ∞ or a NaN
$x + 0 \Leftrightarrow x$	x may be -0 or a NaN
$x * 0 \Leftrightarrow 0$	x may be -0, ∞ or a NaN
$\ldots$	

Optimizations Affecting Value Safety

- Expression rearrangements
- Flush-to-zero
- Approximate division and square root
- Math library accuracy

Expression Rearrangements

These rearrangements are not value-safe:

- $(a \oplus b) \oplus c \Rightarrow a \oplus (b \oplus c)$
- $a \otimes (b \oplus c) \Rightarrow (a \otimes b) \oplus (a \oplus c)$

To disallow these changes:

gcc Don't use `-ffast-math`

icc Use `-fp-model precise`

- Recall that options such as `-On` are "aggregated" or "composite" options
  - they enable/disable many other options
  - their composition may change with new compiler releases

Disallowing rearrangements may affect performance

Subnormal Numbers and Flush-To-Zero

- Subnormal numbers extend the range of floating-point numbers but with reduced precision and reduced performance
- If you do not require subnormals, disable their generation
- "Flush-To-Zero" means "Replace all generated subnormals with 0"
  - Note that this may affect tests for == 0.0 and != 0.0
- If using SSE or AVX, this replacement is fast since it is done by the hardware

Subnormal Numbers and Flush-To-Zero

gcc `-ffast-math` enables flush-to-zero

gcc But `-O3 -ffast-math` disables flush-to-zero

icc Done by default at -O1 or higher

icc Use of `-no-ftz` or `fp-model precise` to will prevent this

icc Use `-fp-model precise -ftz` to get both "precise" behavior and subnormals

- Options must be applied to the program unit containing `main` as well

Reductions

- Summation is an example of a reduction
- Parallel implementations of reductions are inherently value-unsafe
  - the parallel implementation can be through vectorization or multi-threading or both

  icc use of `-fp-model precise` disables vectorization and parallelization via threading
  - there are OpenMP and TBB options to make reductions "reproducible"
    - For OpenMP `KMP_DETERMINSTIC_REDUCTION=yes`

The Hardware Floating-Point Environment

The hardware floating-point environment is controled by the FPU control word

- Rounding mode
- Status flags
- Exception mask
- Control of subnormals

If you change anything affecting the default state of the FPU, you must tell the compiler

- Use `#pragma STDC FENV_ACCESS ON`

icc Use `-fp-model strict`

`#pragma STDC FENV_ACCESS ON` is required if flags are accessed

Precise Exceptions

"Precise Exceptions": floating-point exceptions are reported exactly when they occur

To enable precision exceptions

- Use `#pragma float_control(except, on)`

icc Use `-fp-model strict` or `-fp-model except`

Enabling precise exceptions disables speculative execution of floating-point instructions

Math Library Features – icc

A variety of options to control precision and consistency of results

- `-fimf-precision=<high|medium|low>[:funclist]`
- `-fimf-arch-consistency=<true|false>[:funclist]`
- And several more options
  - `-fimf-absolute-error=<value>[:funclist]`
  - `-fimf-accuracy-bits=<value>[:funclist]`
  - `...`

Questions

Exercises

- Reassociation

  Experiment with `x=a+b+c` or `a+(b+c)` or `(a+b)+c`

  Also the replacement $x/y \Rightarrow x * (1.0/y)$
- Abrupt Underflow

  Experiment with subnormals