LLVM Numerics Improvements

Michael C. Berg, Apple

LLVM Developers' Meeting, Brussels, Belgium, April 2019

Agenda

Handling Numerics via Flags

Current LLVM Numerics Models

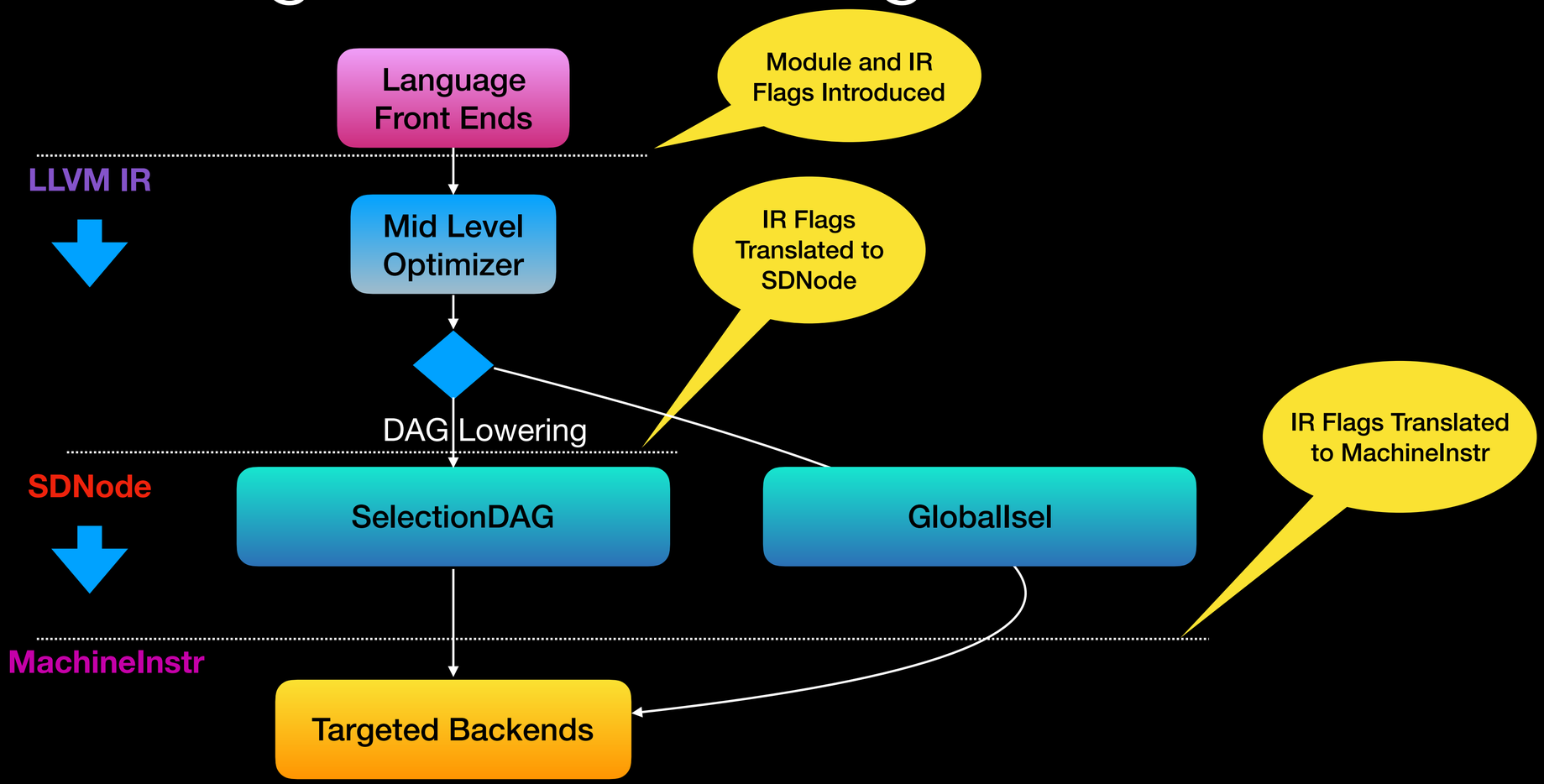
How Unsafe Changes Behavior

Mixed Mode

Flag Guided Optimizations

Conclusions

Handling Numerics via Flags



Agenda

Handling Numerics via Flags

Current LLVM Numerics Models

How Unsafe Changes Behavior

Mixed Mode

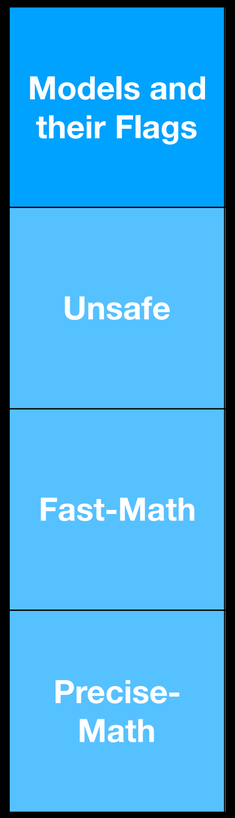
Flag Guided Optimizations

Conclusions

Current LLVM Numerics Models

- Unsafe : module-wide scope overrides Fast Math Flags (FMF).
- Fast-Math: IR scope, FMFs all set.
- Precise-Math: IR scope, FMFs all unset, IEEE–754.

Current LLVM Numerics Models



Current LLVM Numerics Models

Models and their Flags	Nsz
Unsafe	Overrides
Fast-Math	√
Precise-Math	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan
Unsafe	Overrides	Overrides
Fast-Math	√	√
Precise-Math	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan	Ninf
Unsafe	Overrides	Overrides	Overrides
Fast-Math	√	√	√
Precise-Math	X	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Ninf: Allow optimizations to assume the arguments and result are not +/-Inf.

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan	Ninf	Arcp
Unsafe	Overrides	Overrides	Overrides	Overrides
Fast-Math	√	√	√	√
Precise-Math	X	X	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Ninf: Allow optimizations to assume the arguments and result are not +/-Inf.

Arcp: Allow optimizations to use reciprocal operations with approximate expressions.

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan	Ninf	Arcp	Contract
Unsafe	Overrides	Overrides	Overrides	Overrides	Overrides
Fast-Math	√	√	√	√	√
Precise-Math	X	X	X	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Ninf: Allow optimizations to assume the arguments and result are not +/-Inf.

Arcp: Allow optimizations to use reciprocal operations with approximate expressions.

Contract: Allow floating-point contraction (e.g. fusing a multiply add/sub).

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc
Unsafe	Overrides	Overrides	Overrides	Overrides	Overrides	Overrides
Fast-Math	√	√	√	√	√	√
Precise-Math	X	X	X	X	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Ninf: Allow optimizations to assume the arguments and result are not +/-Inf.

Arcp: Allow optimizations to use reciprocal operations with approximate expressions.

Contract: Allow floating-point contraction (e.g. fusing a multiply add/sub).

Reassoc: Allow reassociation transformations on floating-point instructions.

Current LLVM Numerics Models

Models and their Flags	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc	Afn
Unsafe	Overrides	Overrides	Overrides	Overrides	Overrides	Overrides	Overrides
Fast-Math	√	√	√	√	√	√	√
Precise-Math	X	X	X	X	X	X	X

Nsz: Allow optimizations to treat the sign of a zero argument or result as insignificant.

Nnan: Allow optimizations to assume the arguments and result are not NaN.

Ninf: Allow optimizations to assume the arguments and result are not +/-Inf.

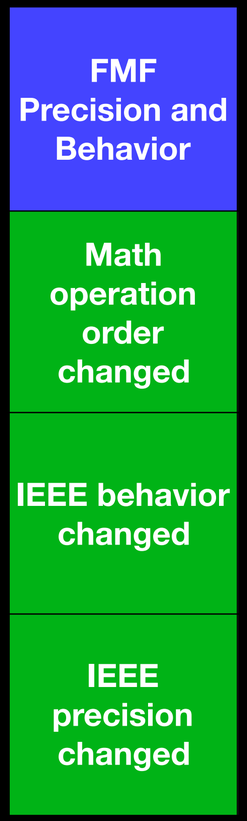
Arcp: Allow optimizations to use reciprocal operations with approximate expressions.

Contract: Allow floating-point contraction (e.g. fusing a multiply add/sub).

Reassoc: Allow reassociation transformations on floating-point instructions.

Afn: Allow substitution of approximate calculations for functions (sin, log, cos, etc).

Current LLVM Numerics Models



Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz
Math operation order changed	√
IEEE behavior changed	√
IEEE precision changed	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan
Math operation order changed	√	√
IEEE behavior changed	√	√
IEEE precision changed	√	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf
Math operation order changed	√	√	X
IEEE behavior changed	√	√	√
IEEE precision changed	√	√	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp
Math operation order changed	√	√	X	NA
IEEE behavior changed	√	√	√	√
IEEE precision changed	√	√	√	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp	Contract
Math operation order changed	√	√	X	NA	√
IEEE behavior changed	√	√	√	√	√
IEEE precision changed	√	√	√	√	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc
Math operation order changed	√	√	X	NA	√	√
IEEE behavior changed	√	√	√	√	√	√
IEEE precision changed	√	√	√	√	√	√

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc
Math operation order changed	√	√	X	NA	√	√
IEEE behavior changed	√	√	√	√	√	√
IEEE precision changed	√	√	√	√	√	√

Changing order of operations may cause rounding differences, NaN and Inf instances may materialize in new ways or even disappear, generalizing the intended values expected in user code.

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc	Afn
Math operation order changed	√	√	X	NA	√	√	NA
IEEE behavior changed	√	√	√	√	√	√	√
IEEE precision changed	√	√	√	√	√	√	√

Changing order of operations may cause rounding differences, NaN and Inf instances may materialize in new ways or even disappear, generalizing the intended values expected in user code.

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

FMF Precision and Behavior	Nsz	Nnan	Ninf	Arcp	Contract	Reassoc	Afn	Fast
Math operation order changed	√	√	X	NA	√	√	NA	√
IEEE behavior changed	√	√	√	√	√	√	√	√
IEEE precision changed	√	√	√	√	√	√	√	√

Changing order of operations may cause rounding differences, NaN and Inf instances may materialize in new ways or even disappear, generalizing the intended values expected in user code.

Notes: The above FMF on IR maps to the same optimizations as Unsafe

Current LLVM Numerics Models

Current LLVM Numerics Models

Model Attributes
Fine Grain Control
IR annotated with flags
NaNs and Infs Preserved
Best Performance and Size
IEEE Compliant

Current LLVM Numerics Models

Model Attributes	Unsafe
Fine Grain Control	X
IR annotated with flags	NA
NaNs and Infs Preserved	X
Best Performance and Size	√
IEEE Compliant	X

Current LLVM Numerics Models

Model Attributes	Unsafe	Fast-math
Fine Grain Control	X	√
IR annotated with flags	NA	√
NaNs and Infs Preserved	X	X
Best Performance and Size	√	√
IEEE Compliant	X	X

Current LLVM Numerics Models

Model Attributes	Unsafe	Fast-math	Precise-math
Fine Grain Control	X	√	√
IR annotated with flags	NA	√	None or arcp
NaNs and Infs Preserved	X	X	√
Best Performance and Size	√	√	X
IEEE Compliant	X	X	√

Current LLVM Numerics Models

Model Attributes	Unsafe	Fast-math	Precise-math	Unsafe with Precise-math
Fine Grain Control	X	√	√	X
IR annotated with flags	NA	√	None or arcp	NA
NaNs and Infs Preserved	X	X	√	X
Best Performance and Size	√	√	X	X
IEEE Compliant	X	X	√	X

Agenda

Handling Numerics via Flags

Current LLVM Numerics Models

How Unsafe Changes Behavior

Mixed Mode

Flag Guided Optimizations

Conclusions

How Unsafe Changes Behavior

- Code emitted as precise can be modified by Unsafe.
- Math functions like acos, cos, sin, asin, etc created by another model can have modified behavior and precision.
- Reassociation globally/locally removes constraints.

Agenda

Handling Numerics via Flags

Current LLVM Numerics Models

How Unsafe Changes Behavior

Mixed Mode

Flag Guided Optimizations

Conclusions

Mixed Mode

- Interleave IR with mixture of flags at some granularity (lib, function, expression).
- Incompatible with Unsafe
- Fast-Math, Precise-Math and other models can coexist.
- Fine granularity of optimization control
- No loss of generality from expressed model
- More design options to manage optimizations

Mixed Mode

Model Attributes	Unsafe	Fast-math	Precise-math	Mixed Mode	Unsafe with Precise-math
Fine Grain Control	X	√	√	√	X
IR annotated with flags	NA	√	None or arcp	In context	NA
NaNs and Infs Preserved	X	X	√	In context	X
Best Performance and Size	√	√	X	In context	X
IEEE Compliant	X	X	√	In context	X

Mixed Mode

Model Attributes	Unsafe	Fast-math	Precise-math	Mixed Mode	Unsafe with Precise-math
Fine Grain Control	X	√	√	√	X
IR annotated with flags	NA	√	None or arcp	In context	NA
NaNs and Infs Preserved	X	X	√	In context	X
Best Performance and Size	√	√	X	In context	X
IEEE Compliant	X	X	√	In context	X

Mixed Mode is available in LLVM 8.0

Agenda

Handling Numerics via Flags

Current Models in LLVM

How Unsafe Changes Behavior

Mixed Mode

Flag Guided Optimizations

Conclusions

Fadd Combine

fadd nsz reassoc (fadd x,c1), c2 -> fadd nsz reassoc x, c1 + c2

For the following f32 input:

```
  %x ~= 3.4028234664E+38 (largest positive number in f32)
  c1 = 1.0, c2 = -1.0
```

We convert this IR:

```
  %t1 = fadd float %x,  0x3FF0000000000000                ; t1 = x + 1.0
  %t2 = fadd nsz reassoc float %t1,  0xBFF0000000000000   ; t2 = t1 + -1.0
```

To this with Unsafe or IR flags:

```
  %t3 = fadd nsz reassoc %x, 0.0

  The result of %t3 is %x
```

Whereas the precise version yields:

```
  %t1 results in Infinity, which propagates to %t2
```

Fmul Combine

fmul reassoc (fmul x, c1), c2 -> fmul reassoc x, c1 * c2

For the following f32 input:

```
%x = 10, %t1 = 0.3
0x36A0000000000000 ~= 1.4012984643E-45 (smallest positive number)
```

We convert this IR:

```
%t1 = fdiv float 3.0, 10.0
%t2 = fmul reassoc float %x, 0x36A0000000000000  ; t2 = x * 1.4012984643E-45
%t3 = fmul reassoc float %t2, %t1                ; t3 = t2 * 0.3
```

To this with Unsafe or IR flags:

```
%t4 = fmul reassoc float %t2, 0
1.4012984643E-45 * 0.3, which is correctly rounded to zero.
```

Whereas the precise version yields:

```
1.4012984643E-44 * 0.3, which is non zero.
```

Fdiv Code Generation

This IR:

```
  %div = fdiv arcp half %x, 10.0
  %z = fpext half %div to float
```

Produces (Unsafe/Fast) x86_64 with avx:

```
.LCPI4_0:
        .long   1036828672              # float 0.0999755859
…
  vmulss  .LCPI4_0(%rip), %xmm0, %xmm0  # z = x * 0.0999755859
```

This IR:

```
  %div = fdiv half %x, 10.0
  %z = fpext half %div to float
```

Produces (Precise) x86_64 with avx:

```
.LCPI4_0:
        .long   1092616192              # float 10
…
  vdivss  .LCPI4_0(%rip), %xmm0, %xmm0  # z = x / 10
```

Agenda

Handling Numerics via Flags

Current Models in LLVM

How Unsafe Changes Behavior

Mixed Mode

Flag Guided Optimizations

Conclusions

Conclusions

- Emit flags on IR and exclude Unsafe to get desired model behavior.
- Mixed mode facilitates fine grained control, while promoting versatility in implementing optimizations.
- Compiler implementers can use the current infrastructure to implement Mixed mode today for their targets.

Future Work Ideas

- FMF function specialization along a call edge
- Inlining with FMF applied from caller instance of call
- Pragma controls
- Per function controls for replacing math lib calls
- New Math Models, new FMF and combinatorics

Note: See llvm-dev EuroLLVM Numerics issues email thread for continuing discussion

Questions?