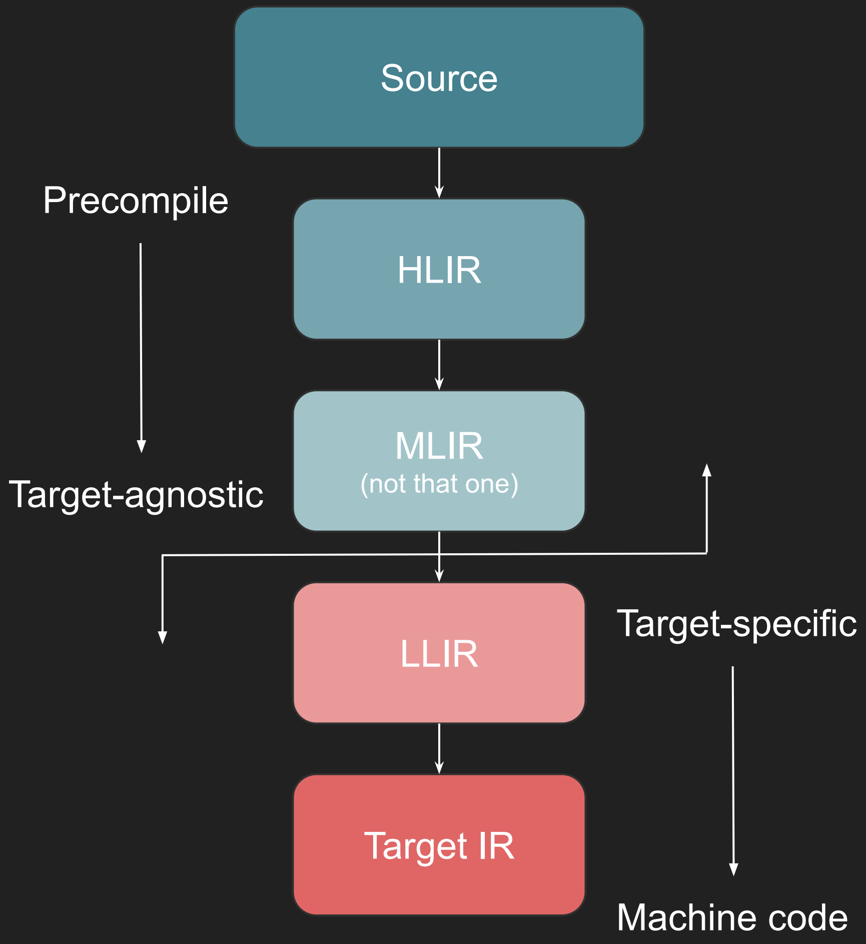
Target-Independent Integer Arithmetic

jeff@modular.com

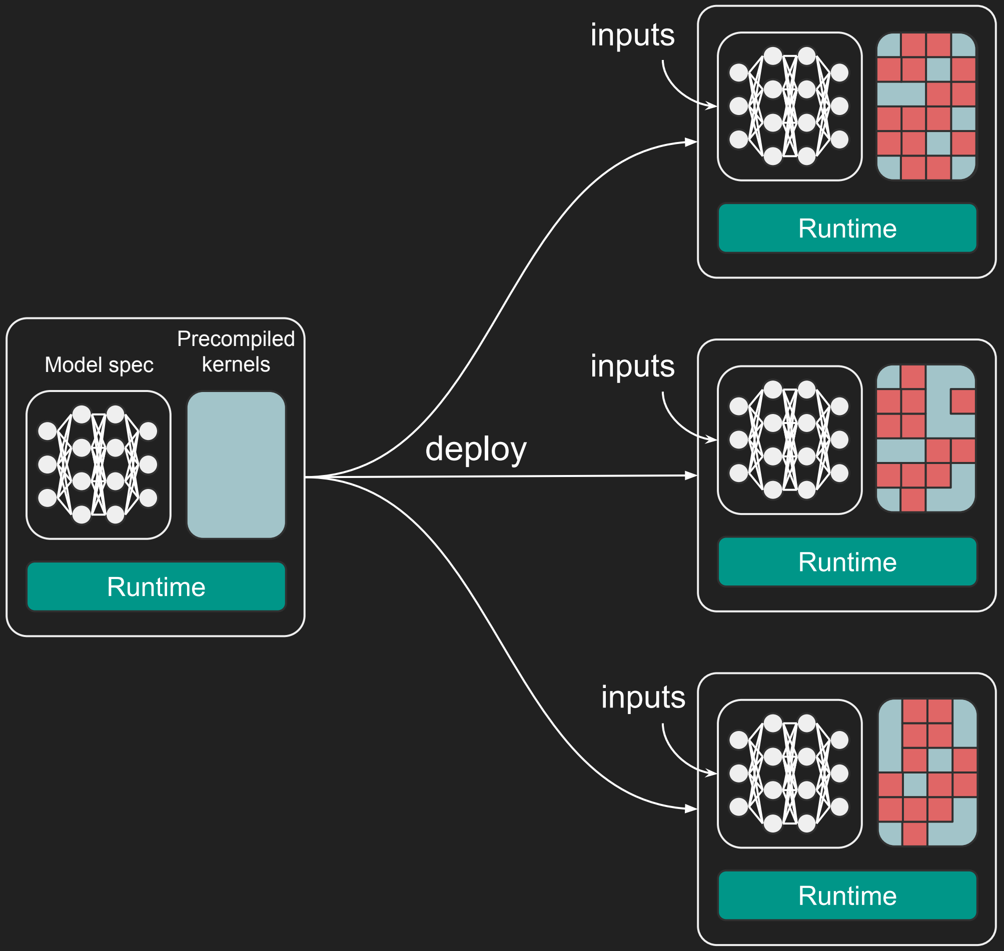
Motivation

- Integer types with target-specific width
  - E.g. C integer types: int, short, long, long long, size_t, intptr_t
- Easy, just always attach a target to the IR
  - The only portable version of the code will be the source!

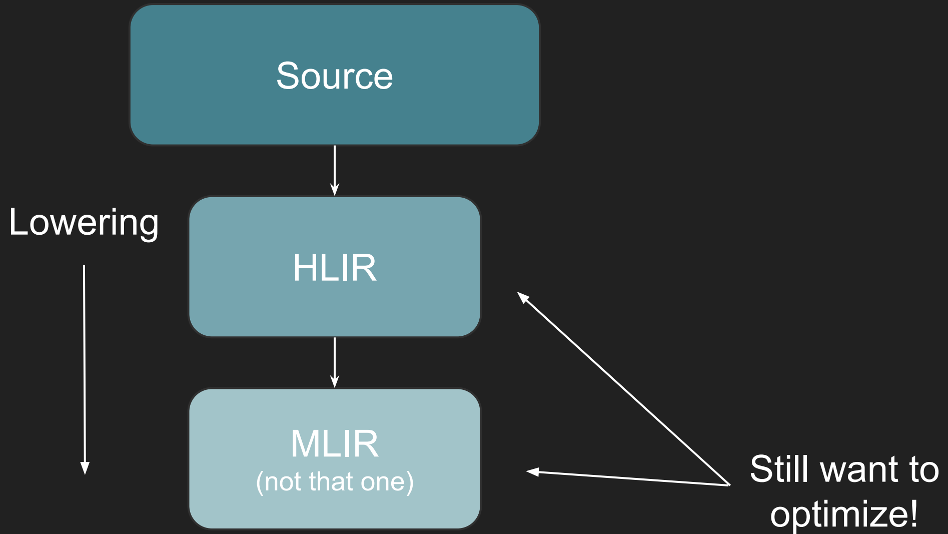
Motivation


Source
Precompile
HLIR
Target-agnostic
MLIR
(not that one)
LLIR
Target-specific
Target IR
Machine code

Motivation


Model spec
Precompiled kernels
Runtime
deploy
inputs
Runtime
inputs
Runtime
inputs
Runtime

Motivation


Source
Lowering
HLIR
MLIR
(not that one)
Still want to
optimize!

Use the maximum possible width?

- Not all types have maximum widths, e.g. the C “at least N bytes” types
- Pick a reasonable maximum or use arbitrary-precision?
- Truncate results to actual width when target is known

Use the maximum possible width?

- Not all types have maximum widths, e.g. the C “at least N bytes” types
- Pick a reasonable maximum or use arbitrary-precision?
- Truncate results to actual width when target is known

Use 64-bit for `int`

$$trunc_{16}(40000 * 2/10) = 8000$$
$$(40000_{16} * 2/10) = 1446$$

Formulation

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

Formulation

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

$$b(f(x, y), i) = g(b(x, j), b(y, j) ...), j \leq i$$

Formulation

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

$$b(f(x, y), i) = g(b(x, j), b(y, j)\ldots), j \leq i$$

✅ add
✅ sub (2's compl)
✅ mul
✅ left shift
✅ and, or, xor

❌ div
❌ floor
❌ min, max
❌ right shift
❌ cmp

Formulation

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

Formulation

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

Just plug it in and check

✅ $trunc_8(2070/8) = 2 == trunc_8(2070)/trunc_8(8)$

MLIR `index` Dialect

- Implements operations on the builtin MLIR `index` type
  - With the appropriate folding logic
- Implements the 🔥 Int type
- PSA: Don’t use `arith` dialect for index types 😤

Integer range analysis

- Almost the same as folding

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

$$trunc_8([180, 200] * [1, 2]) = [180, 144]$$

credit: Krzysztof Drewniak

Integer range analysis

- Almost the same as folding

$$trunc_b(f(x, y)) == f(trunc_b(x), trunc_b(y))$$

$$trunc_8([180, 200] * [1, 2]) = [180, 144]$$

- When not satisfied, **take the union of ranges** computed at the minimum and maximum widths

$$f(trunc_b(x), trunc_b(y)) \cup f(x, y)$$

credit: Krzysztof Drewniak

Thanks!

🥴