Target-Independent Integer Arithmetic
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)

Target-specific

LLIR

Target IR

Machine code
Motivation
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 \texttt{int}

\[
\text{trunc}_{16}(40000 \times 2/10) = 8000 \\
(40000_{16} \times 2/10) = 1446
\]
Formulation

\[ \text{trunc}_b(f(x, y)) = f(\text{trunc}_b(x), \text{trunc}_b(y)) \]
Formulation

\[ \text{trunc}_b(f(x, y)) == f(\text{trunc}_b(x), \text{trunc}_b(y)) \]

\[ b(f(x, y), i) = g(b(x, j), b(y, j) \ldots), j \leq i \]
Formulation

\[ \text{trunc}_b(f(x, y)) == f(\text{trunc}_b(x), \text{trunc}_b(y)) \]

\[ b(f(x, y), i) = g(b(x, j), b(y, j))..., j \leq i \]
Formulation

\[ \text{trunc}_b(f(x, y)) == f(\text{trunc}_b(x), \text{trunc}_b(y)) \]
Formulation

\[ \text{trunc}_b(f(x, y)) == f(\text{trunc}_b(x), \text{trunc}_b(y)) \]

Just plug it in and check

\[ \text{trunc}_8(2070/8) = 2 == \text{trunc}_8(2070)/\text{trunc}_8(8) \]
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

\[ \text{trunc}_b(f(x, y)) = f(\text{trunc}_b(x), \text{trunc}_b(y)) \]

\[ \text{trunc}_8([180, 200] \times [1, 2]) = [180, 144] \]
Integer range analysis

- Almost the same as folding
  
  \[ \text{trunc}_b(f(x, y)) =/= f(\text{trunc}_b(x), \text{trunc}_b(y)) \]

  \[ \text{trunc}_8([180, 200] \times [1, 2]) = [180, 144] \]

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

  \[ f(\text{trunc}_b(x), \text{trunc}_b(y)) \cup f(x, y) \]
Thanks!