2019 LLVM DEV MEETING

Supporting 64 bit pointers in RISCV 32 bit LLVM backend

Reshabh Sharma

Background:

Prof. Taylor’s Bespoke Silicon Group is developing a GP-GPU based on RISC-V 32 bit ISA (RV32)

Why 32 bit ISA?

It is good for very high energy efficiency and density.

What is a good first thing you would expect from a GP-GPU?

Access to 64 bit addressable DRAM

We provide access to 64 bit addresses in DRAM using custom load and store instructions.

LDW rd, rs1, rs2

SDW rd, rs1, rs2

Where rs1 and rs2 always store 32 bit halves of a 64 bit address

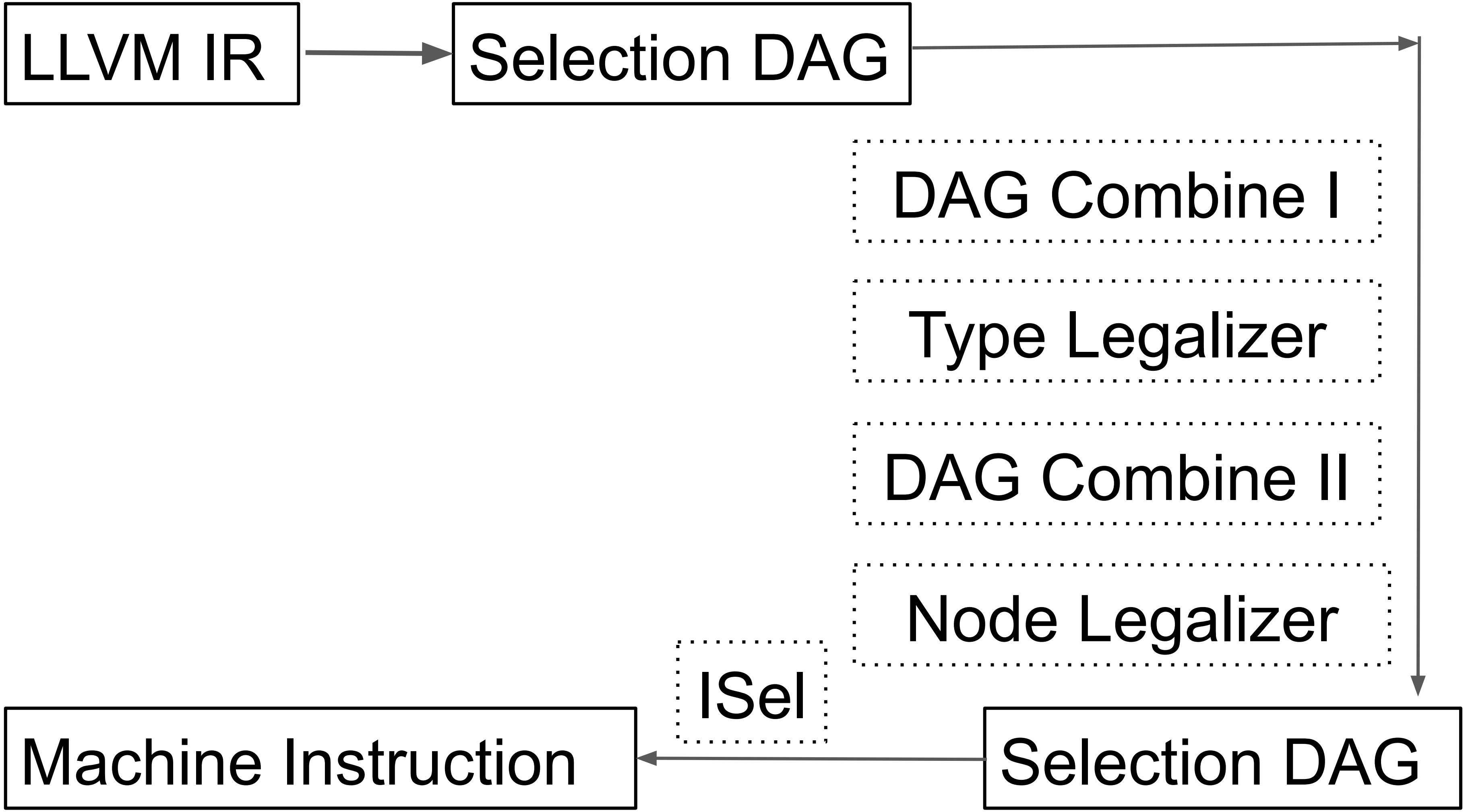
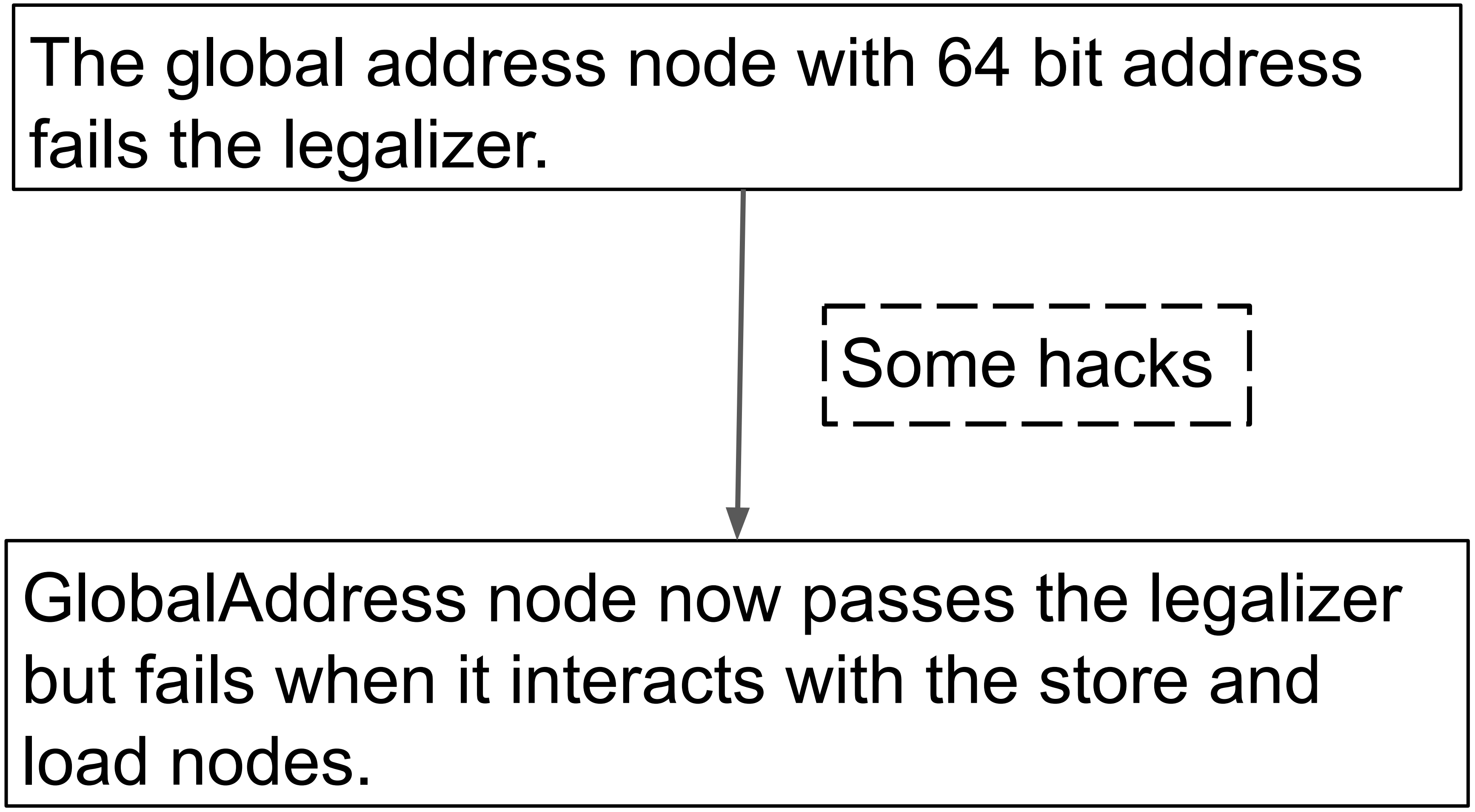
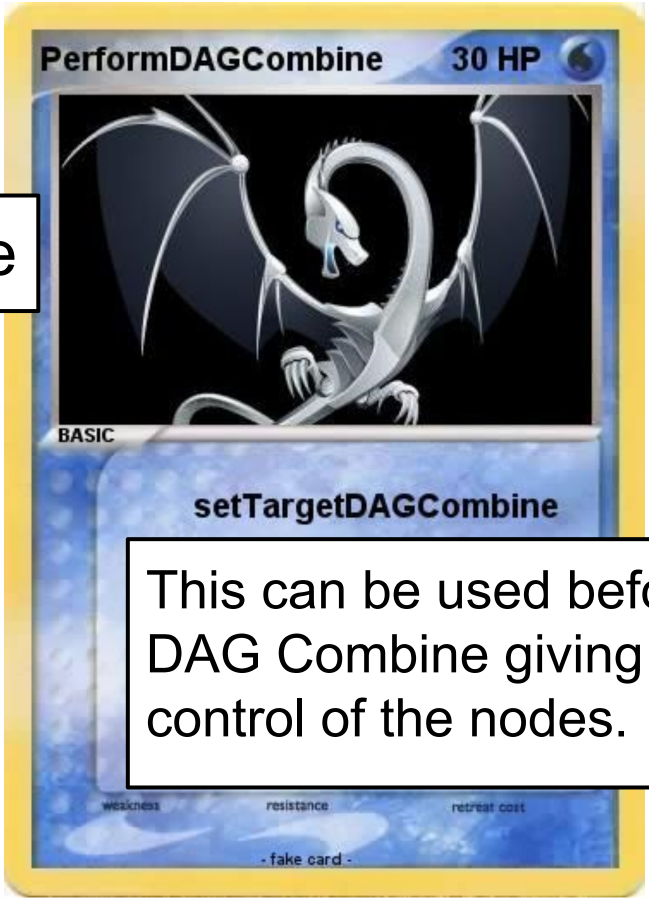

LLVM IR
Selection DAG
DAG Combine I
Type Legalizer
DAG Combine II
Node Legalizer
ISel
Selection DAG
Machine Instruction

Image credits:
Dragon illustration: Vintage vector created by stockgiu - www.freepik.com
Smiley image: The logo belong to the awesome band nirvana


The global address node with 64 bit address fails the legalizer.
Some hacks
GlobalAddress node now passes the legalizer but fails when it interacts with the store and load nodes.

LLVM gives us many good ways to manipulate the nodes before ISel

PerformDAGCombine



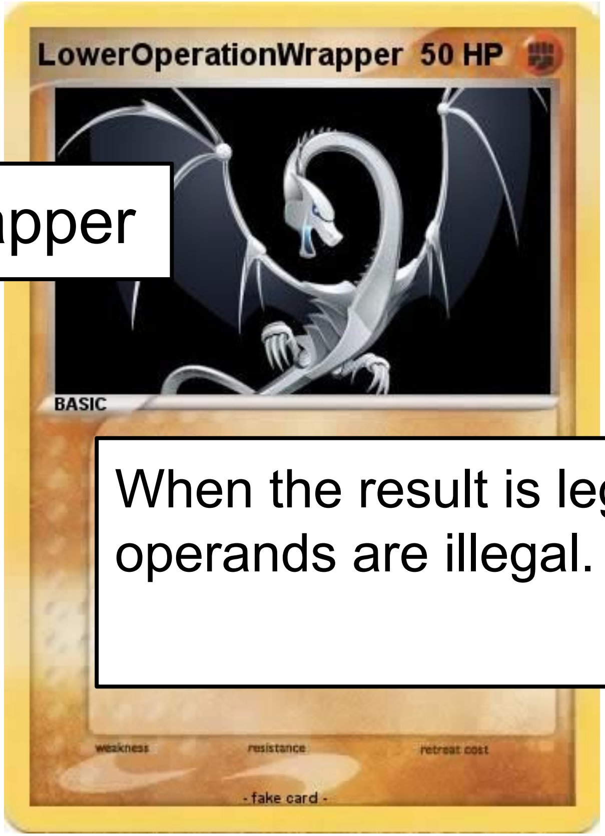
This can be used before any DAG Combine giving complete control of the nodes.

LLVM gives us many good ways to manipulate the nodes before ISel

ReplaceNodeResults

Replaces illegal return type.

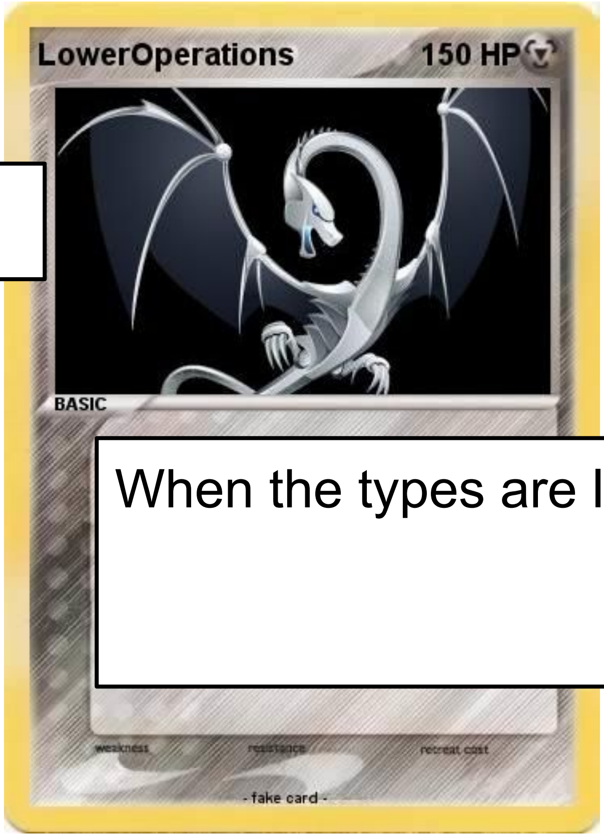
LLVM gives us many good ways to manipulate the nodes before ISel



LowerOperationWrapper

When the result is legal and operands are illegal.

LLVM gives us many good ways to manipulate the nodes before ISel

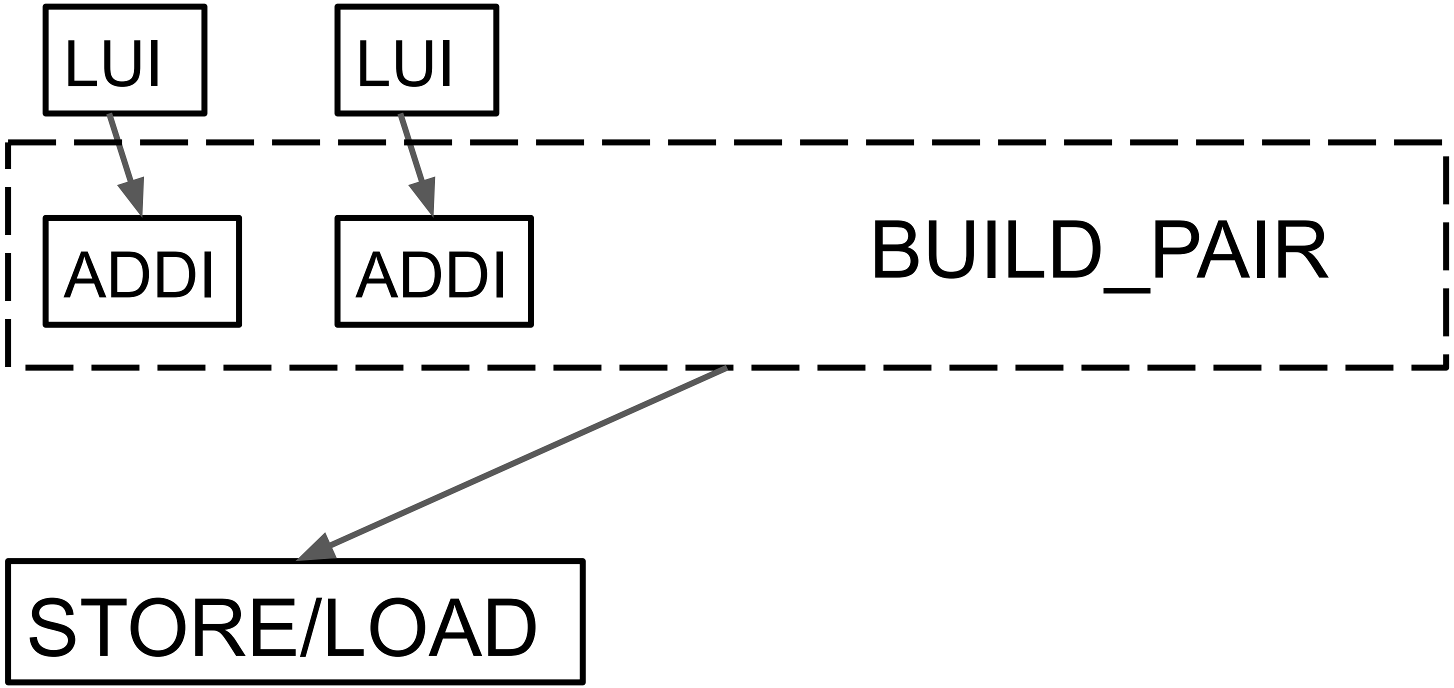


LowerOperations

When the types are legal

How we lowered global address?

GlobalAddress 0x64-bit-address



How we managed to handle store and load with GlobalAddress node?

The only option we had was to lower at the farthest point possible.

Load node is lowered before the Type Legalizer at DAG Combine 1

Store node is lowered at the Node Legalizer

Thank you!