10 Commits Towards GlobalISel for PowerPC

Kai Nacke
Amy Kwan
Nemanja Ivanovic

IBM

Challenges Encountered

Initially, understanding some legalization details were difficult.

Finding out how many type indices an instruction has was not obvious.

Creating new machine basic blocks in the Legalizer does not seem possible.

This is normally not required, but would be useful in certain circumstances.

LLTs are unable to differentiate between different scalar types.

Unlike MVTs, we do not know if we are dealing with an integer or floating point.

Some targets implement utilities that check for FP-related opcodes to know if the LLT we are dealing with is floating point or not.

```
/// Returns whether opcode \p Opc is a pre-isel generic floating-point opcode,
/// having only floating-point operands.
static bool isPreISelGenericFloatingPointOpcode(unsigned Opc) {
  switch (Opc) {
  case TargetOpcode::G_FADD:
  case TargetOpcode::G_FSUB:
  case TargetOpcode::G_FMUL:
  case TargetOpcode::G_FMA:
  case TargetOpcode::G_FDIV:
  case TargetOpcode::G_FCONSTANT:
  case TargetOpcode::G_FPEXT:
  case TargetOpcode::G_FPTRUNC:
  case TargetOpcode::G_FCEIL:
  case TargetOpcode::G_FFLOOR:
  case TargetOpcode::G_FNEARBYINT:
  case TargetOpcode::G_FNEG:
  case TargetOpcode::G_FCOS:
  case TargetOpcode::G_FSIN:
  case TargetOpcode::G_FLOG10:
  case TargetOpcode::G_FLOG:
  case TargetOpcode::G_FLOG2:
  case TargetOpcode::G_FSQRT:
  case TargetOpcode::G_FABS:
  case TargetOpcode::G_FEXP:
  case TargetOpcode::G_FRINT:
  case TargetOpcode::G_INTRINSIC_TRUNC:
  case TargetOpcode::G_INTRINSIC_ROUND:
  case TargetOpcode::G_FMAXNUM:
  case TargetOpcode::G_FMINNUM:
  case TargetOpcode::G_FMAXIMUM:
  case TargetOpcode::G_FMINIMUM:
    return true;
  }
  return false;
}
```

Challenges Encountered *(continued)*

When generating a combiner, using the option to add an additional parameter ended up with a missing space between the type and the name.

There is an easy workaround to this issue.

The generated matcher from the SDAG does not always behave as expected.

In our case, the isCommutable flag is ignored.

Furthermore, immediates used as left hand side operands are not matched.

Pattern Selection Complications.

This is seen when attempting to select certain extend patterns within the PowerPC backend.

The following `sext` pattern can be recognized by the GISel instruction selector:

```
let Interpretation64Bit = 1, isCodeGenOnly = 1 in
defm EXTSW_32_64 : XForm_11r<31, 986, (outs g8rc:$rA), (ins gprc:$rS),
                             "extsw", "$rA, $rS", IIC_IntSimple,
                             [(set i64:$rA, (sext i32:$rS))]>, isPPC64,
                             SExt32To64;
```

But the following simple `zext` pattern can not be recognized.

```
def : Pat<(i64 (zext i32:$in)),
          (RLDICL (INSERT_SUBREG (i64 (IMPLICIT_DEF)), $in, sub_32),
                  0, 32)>;
```

Interesting Discoveries

G_MERGE_VALUES and **G_UNMERGE_VALUES** always takes operands in Little Endian ordering.

This is still the case, even if the target is in Big Endian.

This may come up as a surprise for developers who are working on Big Endian targets.

Not every SDNode has an equivalent GMIR opcode.

For example, umulhilo.

It is not always clear if this is just the current state of development (where certain SDNodes do not map to a single generic opcode), or if there is another reason for it not being available.

Lowering for the **G_SELECT** opcode is only implemented for vector operands at this time.

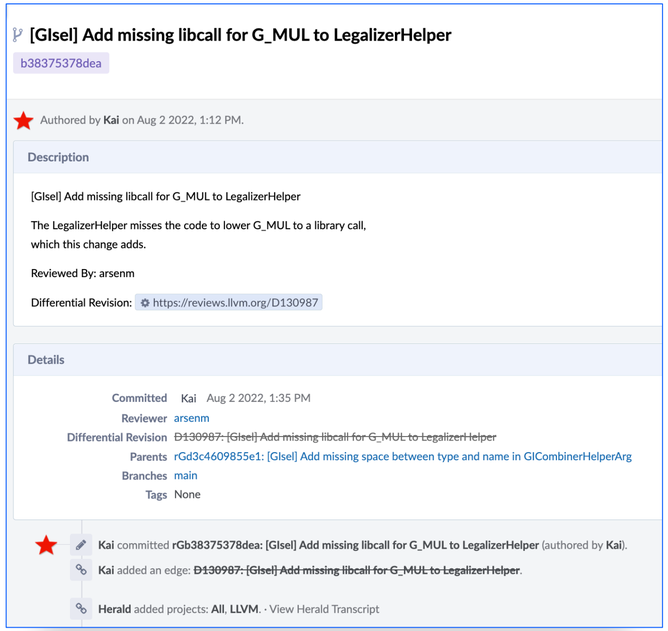
This can be a straightforward upstream fix.

Interesting Discoveries *(continued)*

Libcalls for generic opcodes may be missing.

We discovered that the libcall for G_MUL was missing.

The libcall has been added for the G_MUL opcode upstream.



[GIsel] Add missing libcall for G_MUL to LegalizerHelper

b38375378dea

Authored by **Kai** on Aug 2 2022, 1:12 PM.

Description

[GIsel] Add missing libcall for G_MUL to LegalizerHelper

The LegalizerHelper misses the code to lower G_MUL to a library call, which this change adds.

Reviewed By: arsenm

Differential Revision: https://reviews.llvm.org/D130987

Details

Committed	Kai Aug 2 2022, 1:35 PM
Reviewer	arsenm
Differential Revision	~~D130987: [GIsel] Add missing libcall for G_MUL to LegalizerHelper~~
Parents	rGd3c4609855e1: [GIsel] Add missing space between type and name in GICombinerHelperArg
Branches	main
Tags	None

Kai committed **rGb38375378dea: [GIsel] Add missing libcall for G_MUL to LegalizerHelper** (authored by **Kai**).

Kai added an edge: ~~**D130987: [GIsel] Add missing libcall for G_MUL to LegalizerHelper**~~.

Herald added projects: **All**, **LLVM**. · View Herald Transcript

One of the calling conventions in the PowerPC backend does not use TableGen definitions.

Only a stub definition for this calling convention is available.

This has caused issues in cases where the ABI does not support passing vector parameters.

Specifically, we cannot scalarize vectors in order to pass vector function arguments as scalar values.

```
$ cat and.ll
define <8 x i16> @test_v8i16(<8 x i16> %a, <8 x i16> %b) {
  %res = and <8 x i16> %a, %b
  ret <8 x i16> %res
}

$ llc < and.ll -global-isel -mattr=-altivec
        .text
        .abiversion 2
        .file   "<stdin>"
LLVM ERROR: unable to lower arguments: ptr (in function: test_v8i16)
Stack dump:
0.      Program arguments: llc -global-isel -mattr=-altivec
1.      Running pass 'Function Pass Manager' on module '<stdin>'.
2.      Running pass 'IRTranslator' on function '@test_v8i16'
. . .
```

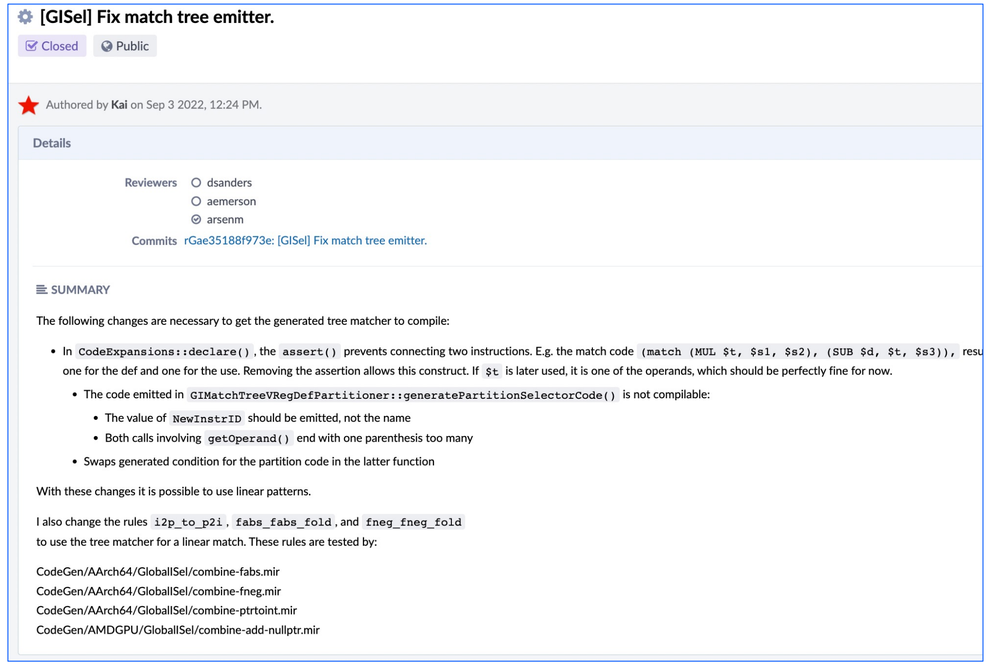
Tool Crashes Experiences

Using the tree matcher for a GICombiner

- `llvm-tblgen` crashes when using the tree matcher for a GICombiner.
- This occurs when creating a combiner when trying to match a sequence of instructions.
- “Declared variable twice” assertion message is displayed in these scenarios.

Discourse post:
https://discourse.llvm.org/t/gicombiner-and-tree-matcher/65014

Phabricator Reviews:
https://reviews.llvm.org/D133257
https://reviews.llvm.org/D134192



[GISel] Fix match tree emitter.

Closed Public

Authored by Kai on Sep 3 2022, 12:24 PM.

Details

Reviewers
- dsanders
- aemerson
- arsenm

Commits rGae35188f973e: [GISel] Fix match tree emitter.

SUMMARY

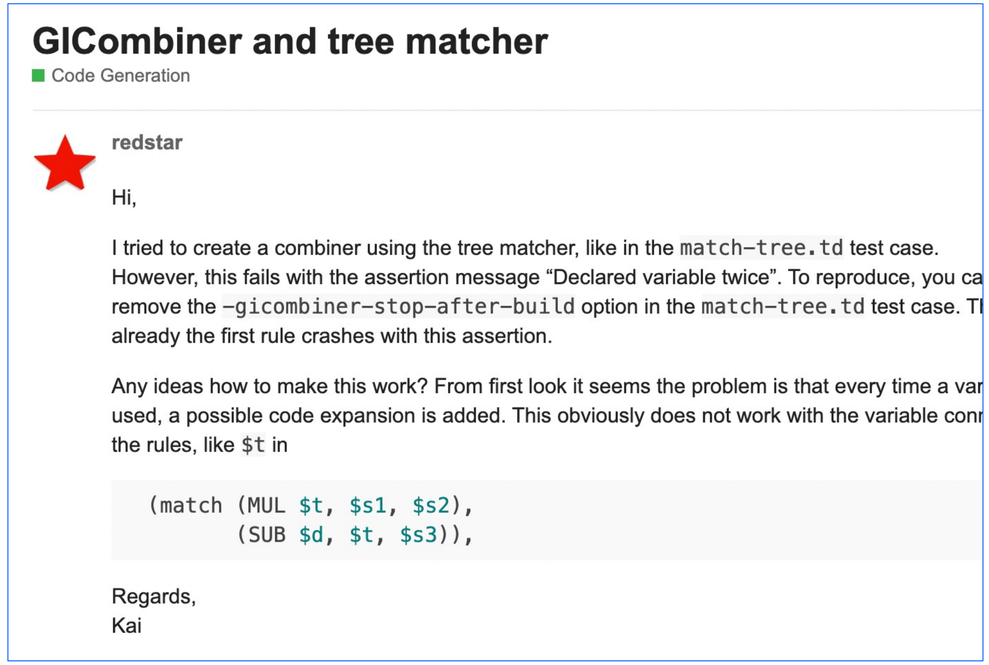
The following changes are necessary to get the generated tree matcher to compile:

- In `CodeExpansions::declare()`, the `assert()` prevents connecting two instructions. E.g. the match code `(match (MUL $t, $s1, $s2), (SUB $d, $t, $s3)),` resu one for the def and one for the use. Removing the assertion allows this construct. If `$t` is later used, it is one of the operands, which should be perfectly fine for now.
  - The code emitted in `GIMatchTreeVRegDefPartitioner::generatePartitionSelectorCode()` is not compilable:
    - The value of `NewInstrID` should be emitted, not the name
    - Both calls involving `getOperand()` end with one parenthesis too many
  - Swaps generated condition for the partition code in the latter function

With these changes it is possible to use linear patterns.

I also change the rules `i2p_to_p2i`, `fabs_fabs_fold`, and `fneg_fneg_fold` to use the tree matcher for a linear match. These rules are tested by:

CodeGen/AArch64/GlobalISel/combine-fabs.mir
CodeGen/AArch64/GlobalISel/combine-fneg.mir
CodeGen/AArch64/GlobalISel/combine-ptrtoint.mir
CodeGen/AMDGPU/GlobalISel/combine-add-nullptr.mir



GICombiner and tree matcher

Code Generation

redstar

Hi,

I tried to create a combiner using the tree matcher, like in the `match-tree.td` test case. However, this fails with the assertion message “Declared variable twice”. To reproduce, you ca remove the `-gicombiner-stop-after-build` option in the `match-tree.td` test case. Tl already the first rule crashes with this assertion.

Any ideas how to make this work? From first look it seems the problem is that every time a var used, a possible code expansion is added. This obviously does not work with the variable conr the rules, like `$t` in

```
(match (MUL $t, $s1, $s2),
       (SUB $d, $t, $s3)),
```

Regards,
Kai

Tool Crashes Experiences *(continued)*

From Hexagon target:

```
def: Pat<(int_hexagon_A2_addi IntRegs:$Rs, timm:$s16),
         (A2_addi IntRegs:$Rs, imm:$s16)>;
```

From SystemZ target:

```
  def : Pat<(int_s390_vcfn VR128:$x, imm32zx4_timm:$m),
            (VCFN VR128:$x, 1, imm32zx4:$m)>;
```

TableGen crashes within the GlobalISel Emitter

It appears that whenever we have:

- An input pattern with an intrinsic using a target constant as a source, and
- An output pattern of an instruction with an immediate as an operand

`llvm-tblgen` will hit an `llvm_unreachable` in `CopyConstantAsImmRenderer::emitRendererOpcodes()` with the message:

`Failed to lookup instruction!`

Discourse Post: https://discourse.llvm.org/t/tablegen-globaliselemitter-crashes-failed-to-lookup-instruction/66196

Our overall GlobalISel Experience...

Implementing GlobalISel for a new target is straightforward for the most part

This is especially true if the target has a complete SDAG implementation.

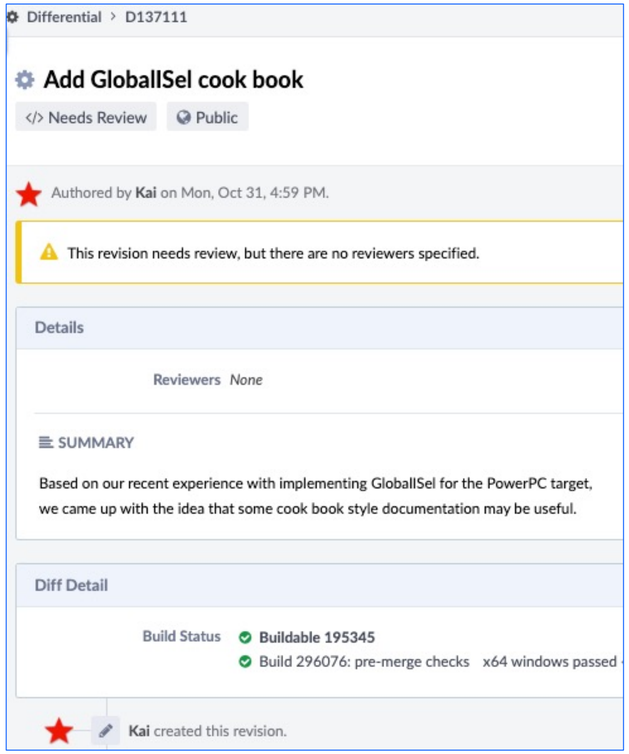
The previous LLVM Dev Meeting Talk (“In 100 Commits to GlobalISel”) is a helpful resource for development.

GlobalISel Community Documentation Contribution

To contribute back to the community, we plan to create a “GlobalISel cookbook” – which aims to assist other targets in adopting the GlobalISel framework to their backends.

The cookbook includes the first “recipes” to follow to adding GlobalISel to a target’s backend, and tips on getting started. This documentation will also include feedback and contributions by other targets.

Phabricator Review:
https://reviews.llvm.org/D137111



Framework Improvements

During our journey in adopting GlobalISel to our backend, we plan to contribute to the framework and to the documentation.