10 Commits Towards GlobalISel for PowerPC

Kai Nacke
Amy Kwan
Nemanja Ivanovic
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.
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.
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.
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.

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.
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
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!**

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.