Vector Pack in VeGen

1 What's in a Vector Pack

A vector pack is not a collection of vectors, but rather a collection of instructions, each of which computes a scalar value.

2 Kinds of Vector Packs

2.1 General

To create a general vector pack, we need a `VectorPackContext`, an array of `Matches`, two bit vectors of elements and their dependencies, an intrinsic that is the producer of this pack, and the LLVM `TargetTransformInfo`.

2.2 Phi

To create a Phi pack, we do not need `Matches` as we do for creating a general pack.

2.3 Load

To create a Load pack, we do not need `Matches` since we already set it to be the load operation. Additionally, we need a condition pack to prevent unwanted reads from improper addresses and a flag to handle non-consecutive loads.

2.4 Store

Same as Load.

2.5 Reduction

2.6 GEP

GEP stands for "get element pointer".

2.7 Gamma

A Gamma pack is also called a Gated Phi Pack. It is a Phi node with incoming blocks replaced with explicit control conditions.

2.8 Cmp

3 Construction of Vector Packs

To construct a vector pack, we need to perform three steps in common, in addition to filling the vector pack context.

3.1 computeOperandPacks

This step consists of two sub-steps: compute and canonicalize. In general, the compute step collects matched values into an array of values, primarily using a structure called `OperandPack`. The canonicalize step wraps the `OperandPack` with a C++ unique pointer and uses a map to ensure uniqueness.

3.1.1 computeOperandPacksForGeneral

3.1.2 computeOperandPacksForLoad

3.1.3 computeOperandPacksForStore

3.1.4 computeOperandPacksForPhi

3.2 computeOrderedValues

This step performs various tasks. For a general pack, it checks the `Matches` and filters out the unmatched operands, setting them to `null`. For Load, Store, Phi, GEP, and Cmp, it simply copies values from the vector pack variants' own data structure to `OrderedValues`, creating a starting point for later processing.

Reduction has only one value, and Gamma places only Phi nodes contained in it to the `OrderedValues`.

3.3 computeCost

This step is self-contained. The cost is either read from an intrinsic guide or estimated using LLVM `TargetTransformInfo` (primarily for load and store).