Code Generation in VeGen

1 Vector Pack Set

VectorPackSet has a method called codegen. The method takes two parameters: a Builder and Paker (Pkr).

- If AllPacks is not empty, print "Vectorized" and the name of the function from Pkr.
- For each VP in AllPacks, and for each pair of instructions in VP, fuse or co-iterate their loops.
- Create a ControlReifier with the context and the data analysis from Paker. Get the loop info and the vector loop info from Packer.
- Define a function ReifyOneHots that takes a vector loop VL as a parameter. For each instruction I in VL, if I is a phi node and has a one-hot phi, reify its condition.
- For each loop L in LI in preorder, get the vector loop for L, reify the back edge condition of VL, and call ReifyOneHots on VL. Then Call ReifyOneHots on the top vector loop.
- For each VP in AllPacks, reify divergent loads and stores conditions.

2 Lower everything

VeGen defines a class VectorCodeGen with a method run, which takes no parameters and returns nothing.

2.1 emit loop

The function emitLoop lowers a VLoop and returns the loop-header and exit blocks. It takes a VLoop and a BasicBlock as parameters, and returns a pair of BasicBlocks, which are the Header and the Exit blocks.

A VLoop is an information add-on data structure that adds information to the LLVM Loop. It holds the dependency analysis results of all the instructions in the loop.

At the beginning, emitLoop creates three blocks and one instruction: a Header, a Latch, and an Exit block, and a Branch instruction. The Latch block will be wired with the Exit and Header blocks later.

In VeGen, the top-level loop has the Header block as the entry block.

The following steps are performed to generate the code for the loop:

- Schedule the instructions and loops according to data dependence.
- Pick out the reduction packs, which will be emitted last.
- Generate code according to the schedule.
- Emit reductions.

The function fixScalarUses patches scalar mu nodes.