Project Title: SLP Vectorization

Group Info:

Jonny Rogers, jsr2@andrew.cmu.edu
Enrico Green, eygreen@andrew.cmu.edu

URL for Project Web Page: https://eptgen.github.io/

Plan of Attack and Schedule:

10/28-11/4: Literature Search on SLP. In particular, closely read the paper which introduced SLP vectorization. We also plan on taking this time to better familiarize ourselves with the LLVM IR (and particularly, how it represents SIMD instructions).
11/4-11/18: Implement a "basic" version of an SLP vectorizer. This should produce correct code, but may not be particularly fast.
11/18-11/25: Implement testing framework and begin optimizing SLP pass
11/25-12/2: Finish SLP optimization begin work on write-up. If ahead of schedule, work on 125% goal.
12/2-12/4: Finish up write-up and work on poster

Milestone:

By 11/20, we will have a working SLP vectorization pass which can identify sets of isomorphic instructions and combine them into SIMD instructions. We will also implement rudimentary transformations (notably, loop unrolling, likely using LLVM's built-in pass) to the code to increase the opportunities for straight-line vectorization. Since we don't plan on implementing a testing framework until the week of the 18th, we plan on creating some "toy" programs to run our pass on in order to demonstrate that the pass is working. However, we don't expect to have any serious analysis done by this point.

Literature Search:

We will base our initial SLP vectorization pass off of the following paper:

Samuel Larsen and Saman Amarasinghe. 2000. Exploiting Superword Level Parallelism with Multimedia Instruction Sets. In *Programming Language Design and Implementation.*

For the purpose of optimization, we may reference some of the papers from the class discussion:

Yishen Chen, Charith Mendis, and Saman Amarasinghe. "All you need is superword-level parallelism: systematic control-flow vectorization with SLP," in Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI 2022), June 2022.

Vasileios Porpodas, Rodrigo C. O. Rocha, Evgueni Brevnov, Luis F. W. Goes, and Timothy Mattson. “Super-Node SLP: optimized vectorization for code sequences containing operators and their inverse elements,” in Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2019), February 2019.

Resources Needed: We plan on implementing our pass on the LLVM IR and testing it on our own computers (likely from the virtual machine provided for previous assignments). As such, we shouldn’t need any additional resources.

Getting Started: We already skimmed the papers for our discussion, and tried to determine what would be useful to implement. For example, implementing the entire tree framework in the Super-Node SLP paper seems tedious and ultimately will most likely not see any improvement, so we might do something simpler instead. Also, the control-flow vectorization seems interesting to do, so we are thinking of implementing parts of the paper. Before we get started, we want to know exactly how to insert SIMD instructions as LLVM code, as well as see if there is an operation in the standard LLVM library to unroll loops.