Assignment 2 Write-Up

Hongpu Ma

Mar 3, 2016

Originally I planned to write a pass to transform some LLVM SIMD intrinsics into LLVM native vector instructions, which will serve as a fundamental of the intrinsic hoisting project. Yet I did not have enough time to investigate and test some related details. Currently, there is one simple example of hoisting `llvm.x86.sse2.psll.q` in the code using `IRBuilder` to insert instructions before the intrinsic call and replace the call with value. The final implementation is expected to have more structural code and will try different approaches and choose the appropriate one (e.g. replacing the intrinsics in line, or implementing the intrinsics in custom functions, etc.).

While writing this pass, I played with the "extension point", which allows me to choose where to insert the pass in the optimization pipeline. I found that Extension Point `EP_EnabledOnOptLevel0` was the earliest point to produce IR with some of the intrinsics hoisted by LLVM. According to the document, this extension point is right after the inlining pass (strictly speaking, several passes inside the Inter-Procedural Transformations). Further investigation on the Intel intrinsic header file shows that intrinsic calls are inline functions: some are implemented via compiler built-in functions; the other are thin wrappers around arithmetic operators, which are the intrinsics that can be "hoisted" automatically (there is no real intrinsic at the beginning). It turns out most automatic hoisting is simply the natural result of function inlining, instead of transforms inside clang or LLVM.