CSCI 4975 — LLVM: A Compiler Case Study
Fall 2016 — Final Project Proposal

Owen Stenson

November 7, 2016

- ## Proposal

  The Rust language aims to compete with C++ in terms of performance, but Rust's lack of maturity and C++'s flexibility pose difficulties. For example, the `__builtin_expect/likely` functions can give some C/C++ compilers important information useful to optimizations that they would otherwise have no way of knowing. I would like to use Rust's framework for writing Compiler Plugins and one or more LLVM Passes to construct a functional implementation of this.

- ## Deliverables

  The main goal I would like to accomplish is to use syntax in the source code to inform branch prediction during compilation.
  An interesting addition would be informing code paths in more complex scenarios; for example, Rust returns generic `enum`s instead of throwing exceptions or returning `null`, meaning that many functions will return an `Ok<A>(A)` type far more often than `Err<B>(B)`. It would be interesting and useful to indicate this to the compiler from the source code level, as described in a Rust RFC.

- ## Novelty

  The idea of `__builtin_expect` and `likely/unlikely` has been around for a while and there are several implementations in different compilers, but adding something similar into the Rust compiler poses a good opportunity to gain familiarity with LLVM and possibly create something that is useful for many people. As noted above, there are also some ways this could be extended that are unique to Rust, but again, those are stretch goals.

- ## Target Audience

  This feature has been requested from Rust for about two years, and there is an RFC that has been accepted to implement it. However, it was blocked by the implementation of MIR and the developer who was initially pushing for it seems to be too busy to work on it now. It would be another interesting stretch goal to get this project to a state in which it can be merged into Rust, but that is probably out of scope in terms of difficulty and completeness. Nevertheless, a functional version or proof of concept could certainly be useful to many of the people who have expressed interest and could be a step toward an official implementation.

- ## Breakdown

  - Owen Stenson
    - Develop a Rust Compiler plugin to analyze and inform the compiler of expected outcomes
    - Develop an LLVM pass to use source code level information to adjust branch branching
    - Develop a method of testing the result to determine the effect of the previous components