GCC Front-End for Rust

@the_philbert
Overview

- Motivation
  - What, Why, How
- Progress
  - Status
  - Community
  - Future Work
- Questions
What is a Front-End

- C/C++ front-end
- Rust front-end
- Go Front-end
- Middle End
- Target Backend (e.g., x86)
- Resulting Assembly
- Assembler
- Object Code
- Linker
- Executable or Lib
What is this?

- Full Implementation of Rust on top of GNU Toolchain
  - Goal to be up streamed with mainline GCC
  - Reuses Binutils (ld, as, gdb)
  - Written in C++
  - Reusing official Rust libcore, libstd, libproc
Personal Motivations

- I enjoy big projects, especially compilers
- GCC will provide a contrast to LLVM
  - Code Size
  - Register Allocation
  - Energy efficiency
  - Security Features
  - Performance characteristics
Benefits

- Independent implementation of Rust
- Tight integration with GCC
- Rustc bootstrapping
- GCC Plugins support
- LTO and CFI
- Drive adoption of Rust though GCC via backporting
- Retargeting
- Backend support for more systems
How

- Community effort began back in 2014
  - Progress stalled with the frequency of language changes
  - Community effort restarted in early 2019
- Recent interest in driving Rust into Linux
  - Open Source Security, inc and Embecosm
  - MVP compiler is fully planned with weekly reporting and milestones tracked
Milestones

- Core Data Structures - Done
- Core Control Flow 1 - Done
- Generics - Done
- Core Trait Resolution – Done
- Control Flow 2 – In Progress
- Macros and cfg-expansion
- Imports and Visibility
- Unstable Features
- Intrinsics/builtins
Demo
Compiler Pipeline
Community

Photo courtesy: Marc Poulhiès
Community

• Goal is to make working on compilers fun
• Status reporting
  – Weekly and Monthly
  – Shout out to contributors
  – Open and transparent
• Monthly Community Call
  – 1st Friday of the Month 09h00 UTC
  – Open to everyone who is interested
  – Hosted on Jitsi
GSoC 2021

- Part of the GCC Organisation
- Accepted two projects
  - Cargo GCCRS
    - Arthur Cohen
  - Dead Code Analysis
    - Wenzhang Yang
Get Involved 1

- Lots of scope to make your mark on the compiler
  - Joel wrote the Parser and AST
  - Mark Wielaard wrote support for unions
  - Marc Poulhiès added module support
  - Arthur Cohen extended this to support for module expansions
  - Thomas Schwinge merges from upstream GCC and built our testsuite with Marc
Get Involved 2

• We keep a list of good first PR’s
  – Task guides and mentorship is offered
• Thanks for the support from:
  – flip1995
  – bjorn3
• Not all contributions must be code
  – Testcases
  – Bug Reports
  • We are on compiler explorer!
Future Work

- Borrow checker
  - https://github.com/rust-lang/polonius
- Incremental Compilation
- Retarget the code onto other infrastructure
- Drive Rust compiler compatibility testing
- Backport the front-end
Special Thanks

- Brad Spengler
  - [https://opensrcsec.com/](https://opensrcsec.com/)
- Jeremy Bennett
  - [https://www.embecosm.com/](https://www.embecosm.com/)
- David Edelsohn
  - [https://gcc.gnu.org/steering.html](https://gcc.gnu.org/steering.html)
Questions

- Github: https://rust-gcc.github.io/
- Email: philip.herron@embecosm.com
- Zulip: https://gcc-rust.zulipchat.com/
- IRC: irc.oftc.net #gccrust
- Thanks!