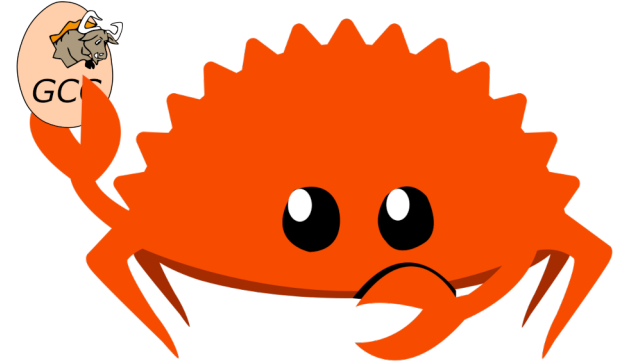




OPEN
SOURCE
SECURITY



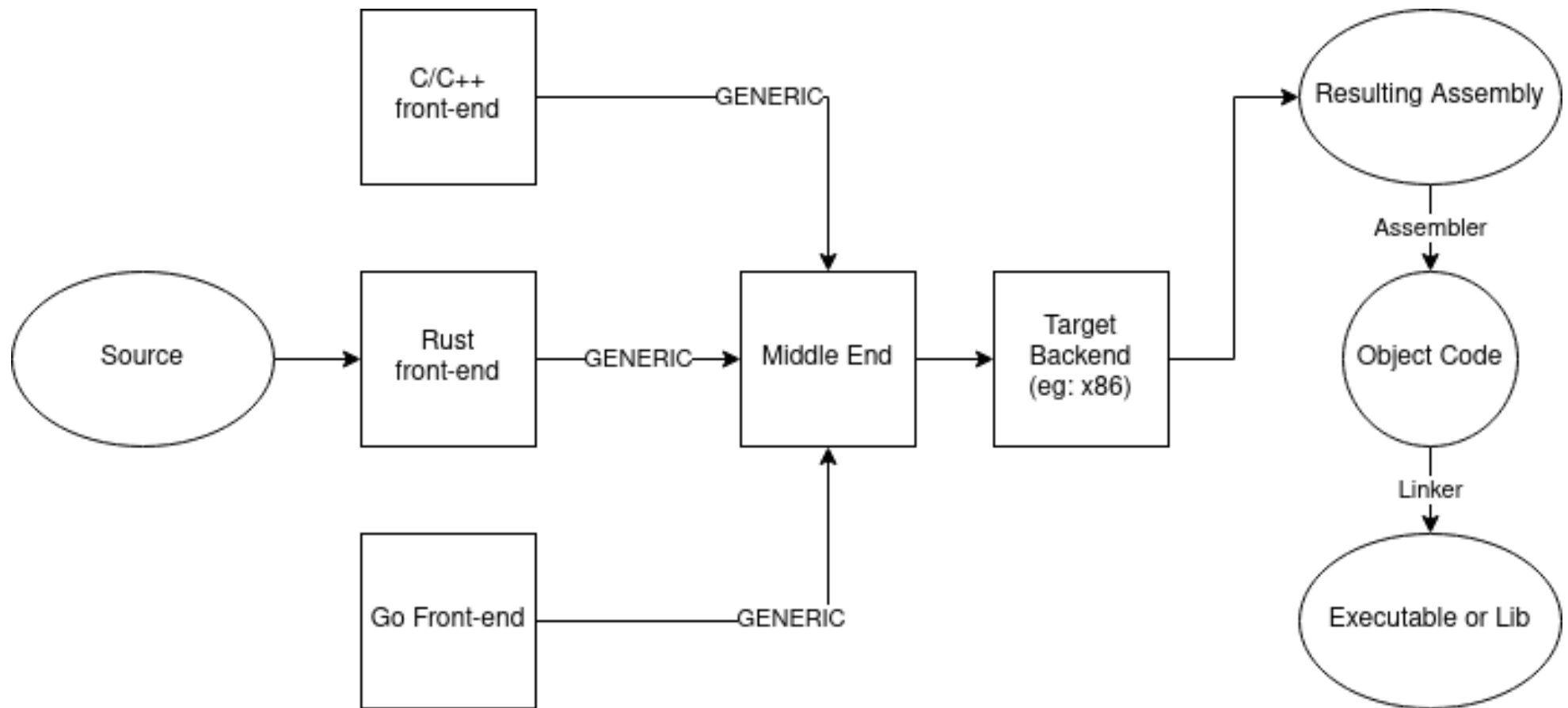
GCC Front-End for Rust

@the_philbert

Overview

- Motivation
 - What, Why, How
- Progress
 - Status
 - Community
 - Future Work
- Questions

What is a Front-End



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 provides 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 through 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
- Ininsics/builtins



Demo

```
0:2gdb - "philbert-beast"
make[2]: Entering directory '/home/philbert/workspace/gccrs-build/gcc'
(rootme=${PWD_CMD-pwd}); export rootme; \
srcdir='cd ../../gccrs/gcc; ${PWD_CMD-pwd}'; export srcdir; \
if [ -n "" ] \
&& [ -n "$GCC_RUNTEST_PARALLELIZE_DIR" ] \
&& [ -f testsuite/rust-parallel/finished ]; then \
rm -rf testsuite/rust; \
else \
cd testsuite/rust; \
rm -f tmp-site.exp; \
sed '/set tmpdir/ s|testsuite|testsuite/rust|' \
< ../../site.exp > tmp-site.exp; \
/bin/bash ${srcdir}/../move-if-change tmp-site.exp site.exp; \
EXPECT='if [ -f ${rootme}/../expect/expect ] ; then echo ${rootme}/../expect/expect ; else echo expect ; fi' ; export EXPECT ; \
if [ -f ${rootme}/../expect/expect ] ; then \
TCL_LIBRARY='cd .. ; cd ${srcdir}/../tcl/library ; ${PWD_CMD-pwd}'; \
export TCL_LIBRARY ; \
fi ; \
if [ -f ${srcdir}/../dejagnum/runtest ] ; then echo ${srcdir}/../dejagnum/runtest ; else echo runtest ; fi --tool rust ; \
if [ -n "$GCC_RUNTEST_PARALLELIZE_DIR" ] ; then \
touch ${rootme}/testsuite/rust-parallel/finished; \
fi ; \
fi )
Using /home/philbert/workspace/gccrs/gcc/testsuite/lib/rust.exp as tool init file.
Test run by philbert on Sun Aug 22 16:36:42 2021
Native configuration is x86_64-pc-linux-gnu

=== rust tests ===

Schedule of variations:
unix

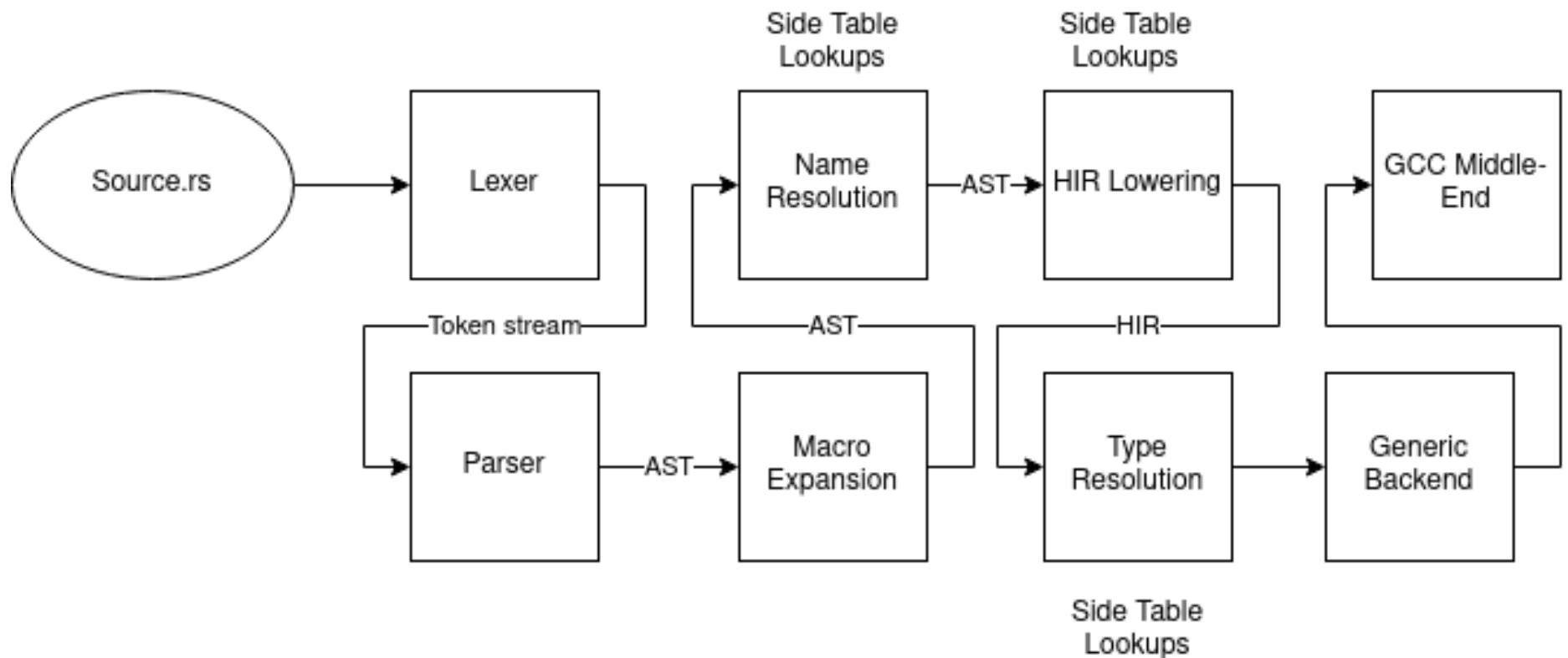
Running target unix
Using /usr/share/dejagnum/baseboards/unix.exp as board description file for target.
Using /usr/share/dejagnum/config/unix.exp as generic interface file for target.
Using /home/philbert/workspace/gccrs/gcc/testsuite/config/default.exp as tool-and-target-specific interface file.
Running /home/philbert/workspace/gccrs/gcc/testsuite/rust/compile/compile.exp ...
Running /home/philbert/workspace/gccrs/gcc/testsuite/rust/compile/torture/compile.exp ...
Running /home/philbert/workspace/gccrs/gcc/testsuite/rust/compile/xfail/xfail.exp ...
Running /home/philbert/workspace/gccrs/gcc/testsuite/rust/debug/debug.exp ...
Running /home/philbert/workspace/gccrs/gcc/testsuite/rust/execute/torture/execute.exp ...

=== rust Summary ===

# of expected passes      4059
# of expected failures    21
make[2]: Leaving directory '/home/philbert/workspace/gccrs-build/gcc'
make[1]: Leaving directory '/home/philbert/workspace/gccrs-build/gcc'
(gdb) cd
[0] 1:bash 2:gdb 3:bash 4:bash 5:bash 6:irssi-
```



Compiler Pipeline



Community



Photo courtesy: [Marc Poulhiès](#)



OPEN
SOURCE
SECURITY



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/>
- Jeremy Bennett
 - <https://www.embecosm.com/>
- David Edelsohn
 - <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
- <https://gcc.gnu.org/mailman/listinfo/gcc-rust>
- Thanks!