LINUX PLUMBERS CONFERENCE September 20-24, 2021

# rustc_codegen_gcc: A gcc codegen for the Rust compiler

## A gcc codegen for Rust

- rustc is based on LLVM.
- rustc provides an API for codegen.
- rustc can load a codegen dynamic library.
- libgccjit can be plugged to rustc via this mechanism.
- PR for inclusion in rustc in review.

## Why do we need this?

- Rust is becoming more and more popular.
- Support more architectures.
- Rust for Linux.
- Embedded programming.
- Some projects (Firefox, librsvg) won't run on architectures not supported by Rust.

## What is implemented?

- Basic and aggregate types.
- Operations, local and global variables, constants, functions, basic blocks.
- Atomics.
- Thread-Local Storage.
- Inline assembly.
- Many intrinsics.
- Metadata.
- Setting optimization level.
- Support in GodBolt, the Compiler Explorer.

## Rust Test Suite

- libcore tests pass.
- Most of the UI tests pass:

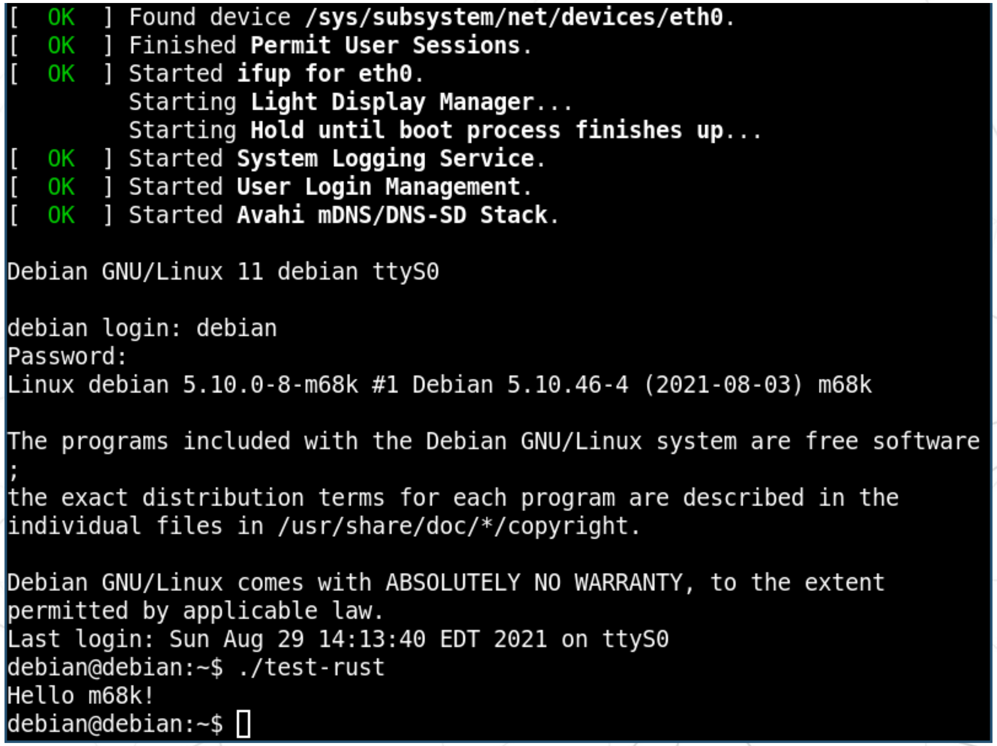
```
test result: FAILED. 4326 passed; 102
failed; 48 ignored; 0 measured; 0
filtered out; finished in 1793.45s
```

## Experiment: running Rust code on m68k

- Still early stage, but proves that it's possible to run Rust on platforms unsupported by LLVM.

LINUX PLUMBERS CONFERENCE September 20-24, 2021

# Experiment: running Rust code on m68k



```
[  OK  ] Found device /sys/subsystem/net/devices/eth0.
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started ifup for eth0.
         Starting Light Display Manager...
         Starting Hold until boot process finishes up...
[  OK  ] Started System Logging Service.
[  OK  ] Started User Login Management.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.

Debian GNU/Linux 11 debian ttyS0

debian login: debian
Password:
Linux debian 5.10.0-8-m68k #1 Debian 5.10.46-4 (2021-08-03) m68k

The programs included with the Debian GNU/Linux system are free software
;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 29 14:13:40 EDT 2021 on ttyS0
debian@debian:~$ ./test-rust
Hello m68k!
debian@debian:~$ 
```

https://github.com/rust-lang/rustc_codegen_gcc

## What needs to be done?

- Some attributes (`#[inline]`, ...).
- Debug info.
- Fix bad code generation.
- 128-bit (and non-power of two) integers on platforms not supporting them.
- Add support for new architectures in libraries (libc, object, …) and rustc.
- LTO.
- SIMD.
- Unwinding.

## What needs to be done?

- GCC constraint code.
- Fix initialization of global variables.
- Target features (to detect what is supported in an architecture, like SIMD).
- Poison value.
- Handle alignment and flags (like volatile).
- Packed structs.

## What could be improved?

- rustc API:
  - Rvalue vs lvalue.
  - Landing pads (unwinding).
  - Handling of basic blocks (mostly an issue for intrinsics that don't exist in gcc).
  - Function vs value.
  - AST-based IR vs instruction-based IR:
    - Example: dereference of pointers in wrong basic block.
  - Separate aggregate operations (structs, arrays, vectors).

## What could be improved?

- libgccjit:
  - Types introspection (with attributes).
- Compilation time.
- Missed optimizations.
- Binary size.

## Patches to libgccjit

- Handle truncation and extension for casts (merged).
- Initialization of global variable (WIP).
- Add support for setting the link section of global variables.
- Add support for sized integer types, including 128-bit integers.
- Add support for TLS variables.
- Add support for types used by atomic builtins.
- Add some reflection functions.
- Implement bitcast.
- Add support for register variables.

# Potential issues

- Distribution of libgccjit.so (gcc binary targets a particular architecture).
- Requires a patched gcc until the patches are merged.
- Different ABI on some platforms.
- `rustc --target=sh2` that just works.
- Backporting to older gcc (for the Linux kernel).
- Running the Rust test suite on new architectures (CI, crater runs).
- Target triples.

## Questions / discussion

https://github.com/rust-lang/rustc_codegen_gcc

## How you can help

- rustc_codegen_gcc:

1) Run the tests locally.
2) Choose a test that fails.
3) Investigate why it fails.
4) Fix the problem.

- Crates:
  - object
  - libc
- Test this project:
  - On new platforms.
  - To compare the assembly with LLVM.
- Good first issue