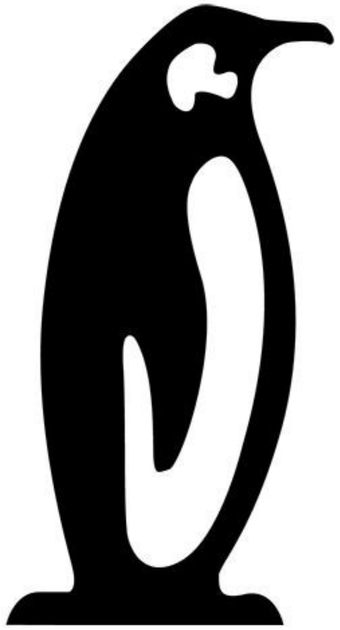


Linux Plumbers Conference

Dublin, Ireland September 12-14, 2022

rustc_codegen_gcc: A gcc codegen for the Rust compiler

Linux Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

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.
- Merged into the Rust repository.

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.

Progress since last year

- rustc_codegen_gcc was merged into the rust repository.
- Complete support for global variables.
- Support for 128-bit integers (-endianness)
- SIMD (stdarch tests).
- Bootstrap rustc.
- Rust for Linux.

Progress since last year (continued)

- Alignment.
- Packed structs.
- Inline asm improvements.
- Symbol visibility.
- Function and variable attributes.
- Many intrinsics.
- Many crashes at compile-time and at run-time.

Progress since last year (continued)

UI tests improvements

Tests	Last year	This year	Delta
Passed	4326	4787	+461
Failed	102	52	-50

Progress since last year (continued)

Summary of the failing UI tests

Category	Number of failing tests
Simd	19
Allocator	9
LTO	10
Asm	3
Other	11

Progress since last year (continued)

SIMD progress

Feature	Completion
Target-specific built-ins support in libgccjit	Done
Support for vector shuffle in libgccjit	Done
LLVM SIMD intrinsics	~99% for x86
Rust SIMD intrinsics	~50%

Progress since last year (continued)

SIMD tests result

```
Test result: FAILED. 4564 passed;
12 failed; 0 ignored; 0 measured;
0 filtered out; finished in 1.03s
```

Progress since last year (continued)

GCC patches

- Add some reflection functions.
- Add support for types used by atomic built-ins.
- Add support for TLS variables.
- Add support for the link section of global variables.
- Add support for bitcasts.
- Add support for register variables.

Progress since last year (continued)

GCC patches (continued)

- Add support for sized integer types, including 128-bit integers.
- Add function to hide stderr logs.
- Add support for setting the alignment.
- Support getting the size of a float.
- Fix bug where unary_op will return an integer type instead of the correct type.
- target: Fix asm generation for AVX built-ins when using `-masm=intel`.

Progress since last year (continued)

libgccjit 12 feature flag

Features implemented

- Basic and aggregate types.
- Operations, local and global variables, constants, functions, basic blocks.
- Atomics.
- Thread-local storage.
- Inline assembly.
- Many intrinsics.
- Metadata.

Features implemented (continued)

- Setting optimization level.
- Support in GodBolt, the Compiler Explorer.
- Packed structs.
- Alignment, symbol visibility, attributes.
- 128-bit integers.
- SIMD (x86).

What needs to be done?

- Unwinding.
- Debug info.
- LTO.
- Endianness support for non-native 128-bit integers.
- Add support for new architectures in libraries (libc, objects, …) and rustc.
- SIMD for targets other than x86.

What needs to be done? (continued)

- More function and variable attributes.
- GCC constraint code.
- Target features (to detect what is supported in an architecture, like SIMD).
- Distribution via rustup.

What could be improved?

- rustc API:
  - Rvalue vs lvalue.
  - Landing pads (unwinding).
  - Handling of basic blocks.
  - Function vs value.
  - AST-based IR vs instruction-based IR:
    - Example: dereference of pointers.
  - Separate aggregate operations (structs, arrays, vectors).

What could be improved? (continued)

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

What’s required to compile Rust for Linux

- CPU features detection.
- Some compiler flags (`-Crelocation-model=static` vs `-mcmodel=kernel -fno-pie`).

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.

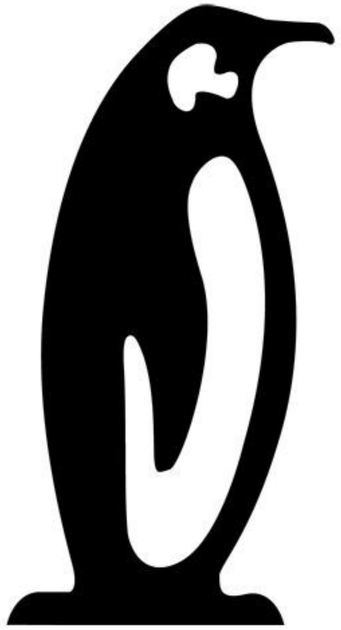
How you can help

- 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

Thanks

- Sponsors.
- Contributors.

Questions / discussion



Linux Plumbers Conference

Dublin, Ireland September 12-14, 2022