rustc_codegen_gcc

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`.

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 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%

SIMD TESTS RESULT

```
test result: FAILED. 4564 passed; 12 failed; 0 ignored; 0 meas
```

PROGRESS SINCE LAST YEAR (CONTINUED)

GCC PATCHES

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

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`, `object`, ...) 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).

WHAT COULD BE IMPROVED? (CONTINUED)

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

DEMO: COMPILING RUST FOR LINUX

WHAT'S REQUIRED TO COMPILE RUST FOR LINUX

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

POTENTIAL ISSUES FOR RUST FOR LINUX

- Different ABI on some platforms.
- Backporting to older `gcc`.
- Requires a patched `gcc` for now.

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`

HOW YOU CAN HELP (CONTINUED)

- Test this project:
  - On new platforms.
  - To compare the assembly with LLVM.
- good first issue

QUESTIONS / DISCUSSION