rustc

A wonderful (ongoing) journey

@ereslibre - Rafael Fernández López

What we’ll see today?

- Rust structure
- Bootstrap process
- The driver
- AST
    - What checks are performed using the AST?
- HIR
    - How the AST becomes the HIR?
    - What checks are performed using the HIR?
- MIR
    - How MIR is built
    - What checks are performed using the MIR?
- Compiler plugins

Best friends

- `rustup`
    - `rustup toolchain link stage2 ~/projects/rust/build/x86_64-unknown-linux-gnu/stage2`
- `ctags generator`
    - Easier code navigation
    - https://github.com/nikomatsakis/rust-ctags
- `git {log,blame,show}`
    - Understand some immediate context
- `rustc +stage2 -Z help`
- `RUST_LOG=something rustc +stage2 ...`

Structure

- Rust is a set of crates
  - `libsyntax` (Lexer, Parser)
  - `librustc_driver`
  - `librustc`
  - `librustc_codegen_llvm`
  - `librustc_borrowck`
  - `librustc_traits`
  - `librustc_mir`
  - `libcore` (Generic data structures)
  - `libstd`
  - Many others...

Bootstrap process (`src/bootstrap`)

- Built-in rust build system
- `x.py` wrapper
- Stages
  - Stage 0
    - Download rustc and libstd
    - Build compiler artifacts
  - Stage 1
    - Build compiler with Stage 0 compiler
      - Suitable for most rustc developing tasks (except procedural macros, custom derive)
  - Stage 2
    - Build final compiler with Stage 1 compiler
    - Optimized version -- it compiled itself
  - Stage 3 (optional)
    - Sanity check: should be identical to Stage 2

Building

- As of today, general case
- First run
  - `./x.py build -i -j<N>`
- Later runs
  - `./x.py build -i -j<N> --keep-stage 0 src/libstd`
- Testing
  - `./x.py test -i -j<N> --keep-stage 0 src/test/.../whatever.rs`

The driver (`src/librustc_driver`)

- Handles the compilation process
- Compiler drop-in replacements
  - Wrapper around rustc that allows you to handle compiler callbacks
  - Example: https://github.com/nrc/stupid-stats
- 4 main phases
  - Parser
  - Configure and expand
  - Analysis
  - Codegen

Patterns

- Visitor
  - Visits AST, HIR, MIR nodes
  - Behavior can be overridden
- Folder
  - Can transform AST nodes (e.g. during macro expansion)

Session (`src/librustc/session`)

- Contains the state of the compilation process

Phase 1: Parser (`src/libsyntax`)

- `ast::Crate`
    - Explore `ast::*`

Phase 2: Configure and Expand (`src/librustc`)

- Compute crate features
- Crate injection (`core, std prelude`)
- Compiler plugin loading/registration
- Register built-in syntax extensions
- Expand macros
    - This includes removing conditional code -- `#[cfg], #[test]`
- Build test harness -- if necessary
- AST validation
- Name resolution
- Lower `AST -> HIR`

Phase 3: Analysis

- Initial checks
- Load providers
- Incremental compilation initialization
- Create `TyCtxt`
  - Type checking
  - Borrow checking
    - HIR (`src/librustc_borrowck`)
    - MIR (`src/librustc_mir/borrow_check`)
  - Many other HIR based checks
- Builds MIR

MIR (Tomorrow is Today now)

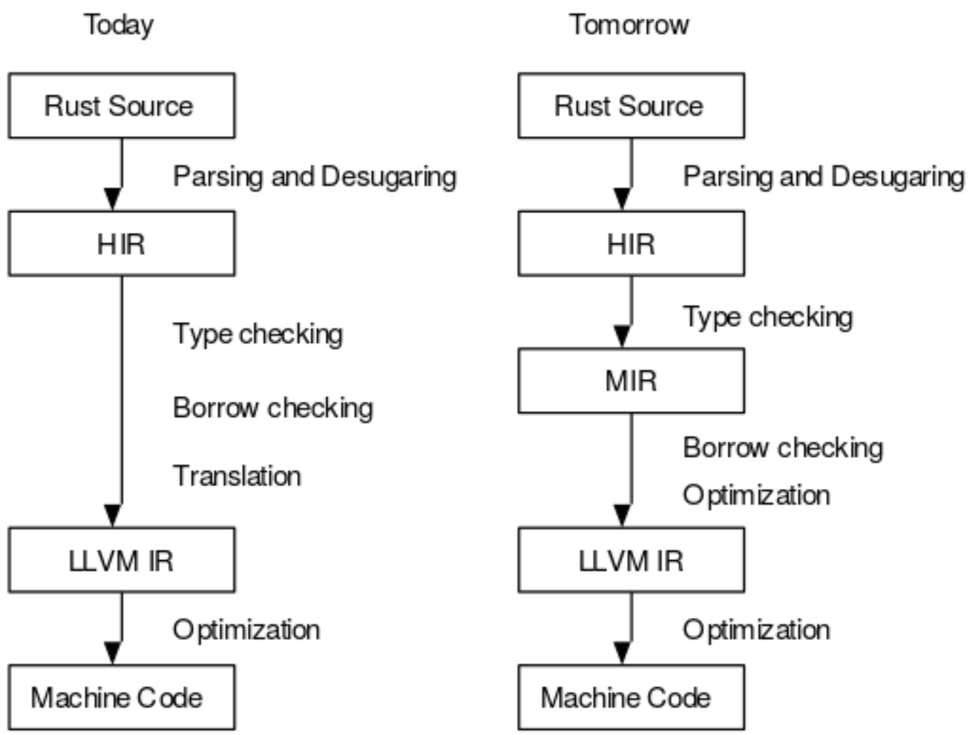
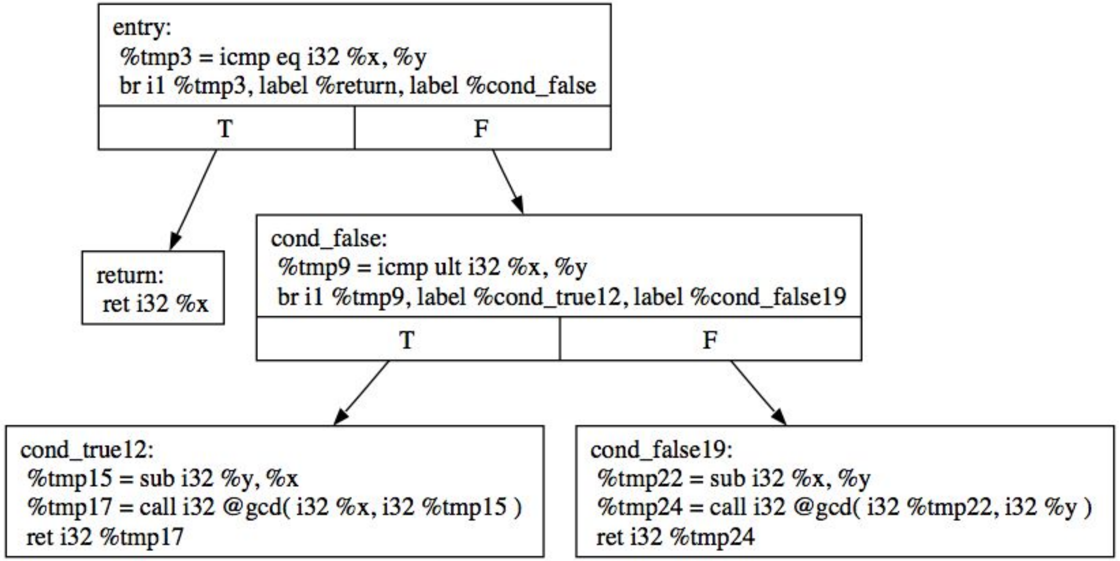


Image: https://blog.rust-lang.org/2016/04/19/MIR.html

Phase 4: Codegen (`src/librustc_codegen_llvm`)

- MIR basic blocks are translated into LLVM basic blocks



```
entry:
 %tmp3 = icmp eq i32 %x, %y
 br i1 %tmp3, label %return, label %cond_false
```

```
return:
 ret i32 %x
```

```
cond_false:
 %tmp9 = icmp ult i32 %x, %y
 br i1 %tmp9, label %cond_true12, label %cond_false19
```

```
cond_true12:
 %tmp15 = sub i32 %y, %x
 %tmp17 = call i32 @gcd( i32 %x, i32 %tmp15 )
 ret i32 %tmp17
```

```
cond_false19:
 %tmp22 = sub i32 %x, %y
 %tmp24 = call i32 @gcd( i32 %tmp22, i32 %y )
 ret i32 %tmp24
```

CFG for 'gcd' function

- LLVM optimizations
- Linking

Image: http://releases.llvm.org/2.6/docs/tutorial/JITTutorial2.html

Resources

```
~/projects/rust/src (master) > find . -name README.md | wc -l
50
```

- https://internals.rust-lang.org/
- Discord
    - https://discordapp.com/invite/rust-lang
- https://github.com/rust-lang/rust
- https://github.com/rust-lang/rfcs
- https://github.com/rust-lang-nursery/rustc-guide
    - Rendered: https://rust-lang-nursery.github.io/rustc-guide/

Bonus track

- https://github.com/rust-lang/rust/pull/54161