Usage

rustc_codegen_cranelift can be used as a near-drop-in replacement for `cargo build` or `cargo run` for existing projects.

Assuming `$cg_clif_dir` is the directory you cloned this repo into and you followed the instructions (`y.rs prepare` and `y.rs build` or `test.sh`).

Cargo

In the directory with your project (where you can do the usual `cargo build`), run:

```
$ $cg_clif_dir/build/cargo-clif build
```

This will build your project with rustc_codegen_cranelift instead of the usual LLVM backend.

Rustc

> *You should prefer using the Cargo method.*

```
$ $cg_clif_dir/build/bin/cg_clif my_crate.rs
```

Jit mode

> *⚠⚠⚠ The JIT mode is highly experimental. It may be slower than AOT compilation due to lack of incremental compilation. It may also be hard to setup if you have cargo dependencies. ⚠⚠⚠*

In jit mode cg_clif will immediately execute your code without creating an executable file.

> *This requires all dependencies to be available as dynamic library. The jit mode will probably need cargo integration to make this possible.*

```
$ $cg_clif_dir/build/cargo-clif jit
```

or

```
$ $cg_clif_dir/build/bin/cg_clif -Zunstable-features -Cllvm-args=mode=jit -Cprefer-dynamic my_crate.rs
```

There is also an experimental lazy jit mode. In this mode functions are only compiled once they are first called.

```
$ $cg_clif_dir/build/cargo-clif lazy-jit
```

Shell

These are a few functions that allow you to easily run rust code from the shell using cg_clif as jit.

```
function jit_naked() {
    echo "$@" | $cg_clif_dir/build/bin/cg_clif - -Zunstable-features -Cllvm-
args=mode=jit -Cprefer-dynamic
}

function jit() {
    jit_naked "fn main() { $@ }"
}

function jit_calc() {
    jit 'println!("0x{:x}", ' $@ ');';
}
```