RISC Zero

https://risczero.com/

https://dev.risczero.com/api/zkvm

RISC Zero - ChatGPT

zkSTARK の証明データ(sealと呼ばれる？)を作成する zkVMのようだ。
しかし zkSTARK の証明データは大きいので、それをさらに zkSNARK で証明して圧縮している。
Recursive Proofs という手法だそうだ(Recursive Proving)。
(RISC Zeroとして統合しているのはzkSTARKで、zkSNARKでの圧縮は `recursion` モジュールの機能)

最近のペースでは、月に1回くらいマイナーアップデートされているようだ。

メモ

- method ID と Image ID は別物
  - method ID
    - RISC Zero で証明対象としてビルドされた RISC-V プログラム（メソッド）ごとの ID で ELFファイルベースのハッシュ値
    - 期待するプログラムかどうかのチェックに使う
  - Image ID
    - 証明対象となる「RISC-Vプログラムのイメージ全体」の固有識別子で、より低レベルで扱われるID
    - 実際に実行された証明のメタ情報やブロックチェーン上の検証・識別に用いる

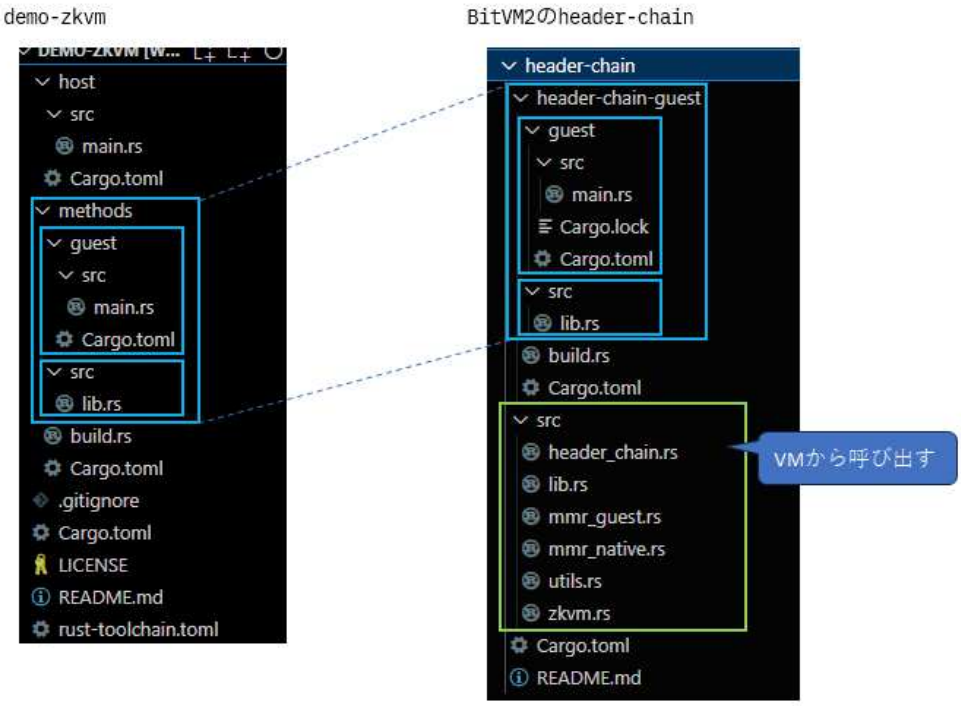
やってみよう

https://zenn.dev/naizo01/articles/5bd666b160aab6

`sample`

```
$ cargo risczero new demo-zkvm
Guest name was not supplied through the --guest-name option. Please enter package name for your template or press [enter] to use default guest package name "method"
Enter package name >
$ cd demo-zkvm
```

guest package name で Enter だけ打つと "method" という名前が使われる。BitVM2 だと `header-chain-guest` のようなのがそれである。



自動生成される `README` にはこうある。

```
project_name
├── Cargo.toml
├── host
│   ├── Cargo.toml
│   └── src
│       └── main.rs                       <-- [Host code goes here]
└── methods
    ├── Cargo.toml
    ├── build.rs
    ├── guest
    │   ├── Cargo.toml
    │   └── src
    │       └── method_name.rs            <-- [Guest code goes here]
    └── src
        └── lib.rs
```

`ELF`

```
$ file target/riscv-guest/methods/method/riscv32im-risc0-zkvm-elf/release/method
target/riscv-guest/methods/method/riscv32im-risc0-zkvm-elf/release/method: ELF 32-bit LSB executable, UCB RISC-V, soft-float ABI, version 1 (SYSV), statically linked, not stripped
```

やってみよう2

hello-world

quick startの手順でプロジェクトを作ったのだがファイル構成が違う。デフォルトのバイナリクレートではなく `host` に置いただけのようだ。

new

```
$ cargo risczero new hello-world --guest-name guest
```

これでつくると git の設定はされていないようだった。

チュートリアルは今回やろうとしている hello-world のようだ。ただ RISC Zeroリポジトリの中にあるので Cargo.toml で指す先が相対パスになっている。実際は公開クレート？のような参照の仕方になるだろうから、その辺は `cargo risczero new` で作った方をそのまま使えばよかろう。
それ以外のRSファイルは GitHub の方のファイルをコピペした。Cargo.toml はパッケージ名を変更した程度だ。 `host/src/lib.rs` はREADME.mdのパスを指定していたので適当に作る。

この状態で `cargo run` するとビルドエラーになる。
エラーになっているのは `methods/src/lib.rs` 。 `OUT_DIR` 環境変数が未設定なせいなのだが、 `new` したものもこう書いているのでよくわからん。 `OUT_DIR` はRustの環境変数らしい。
`cargo clean` したらうまくいくようになった？ vscodeの rust-analyzer が自動でビルドしたりしているので、それが影響している気もする。

`cargo run` は動いたのだが実行すると panic になった。

panic

```
       Finished `dev` profile [optimized + debuginfo] target(s) in 49.14s
        Running `target/debug/hello-world`

   thread 'main' panicked at host/src/lib.rs:40:51:
   called `Result::unwrap()` on an `Err` value: Your installation of the r0vm server
   is not compatible with your host's risc0-zkvm crate.
   Do one of the following to fix this issue:

   1. Install r0vm server version 2.1
   2. Change the risc0-zkvm and risc0-build dependencies in your project to 2.1

   risc0-zkvm version: 2.2.0
   r0vm server version: 2.1.0
   note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

この行の `unwrap()` である。

`risc0-zkvm` や `r0vm server` のバージョンが書かれているので、その不一致？

昨日 v2.2.0 がリリースされたばかりなのでその影響も考えられる。上でやった demo-zkvm はエラーにならないし。

version

```
$ cargo risczero --version
cargo-risczero 2.1.0
```

`r0vm` がコマンド名で `$HOME/.cargo/bin/` にあった。これがサーバ的な役割を果たす。自動で起動して終了するのであまり気にしなくてよい。常駐もできるらしい。

version

```
$ r0vm --version
risc0-r0vm 2.1.0
```

`risc0-zkvm = { version = "^2.1.0" }` となっていたので `risc0-zkvm = { version = "2.1.0" }` と固定にする。

これでも変わらないので `Cargo.lock` を削除。それでも `risc0-zkvm` は"2.2.0"のままになる。

これはRustのしくみがそうなっているからで、たとえば `"0.1.12"` と書いても `0.1.12 <= version < 0.2.0` だそうな。

固定するときは `risc0-zkvm = { version = "=2.1.0" }` のように頭に `=` を付ける。

cargo run

```
    Compiling hello-world v0.1.0 (/mnt/wsl/home2/ueno/Bitcoin/risc0/hello-world/host)
     Finished `dev` profile [optimized + debuginfo] target(s) in 1m 25s
      Running `target/debug/hello-world`
I know the factors of 391, and I can prove it!
```

よし。