RISC-V Bootstrapping in Guix and Live-Bootstrap

Status report for 2023, and what's coming up next

Ekaitz Zárraga

Before we start

If you want details and don't mind reading large text walls full of nitty-gritty details, you can leave this talk and read all the blogposts I did on the process[1]

You can still stay here and smile, too.

[1] https://ekaitz.elenq.tech/tag/bootstrapping-gcc-in-risc-v.html

Who I am

- Telecommunication engineer (EEE equivalent)
- Freelance engineer/programmer at ElenQ.Tech[2]
- Guix user and contributor
- You might remember me from my talk last year: *"Bringing RISC-V to Guix's bootstrap"*

[2] https://elenq.tech

Context

- In 2021 NlNet[3] funded me to do some bootstrapping work: I backported RISC-V support to GCC 4.6.4 from 7.5, and to our TinyCC-Boot from upstream TinyCC. These were fundamental points in our bootstrap chain.
- I explained that in FOSDEM 2023[4].

[3] https://nlnet.nl/project/GNUMes-RISCV/index.html
[4] https://archive.fosdem.org/2023/schedule/event/guixriscv/

Intro

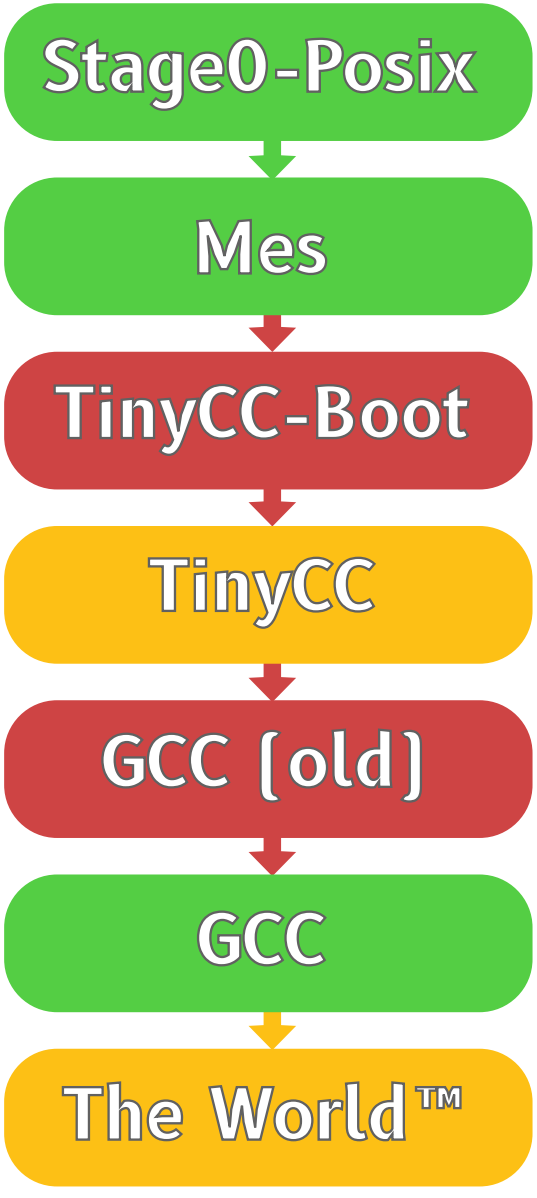
- The work wasn't finished so I asked for a second grant to NlNet[5].
- I was tired of the previous effort and I needed help. I decided to bring more people to the project:
  - **Andrius Štikonas**: Involved in `live-bootstrap`, `stage-0` and the related projects. Has the context awareness I lack.
  - **Janneke**: `Mes` maintainer and author, `TinyCC-boot` maintainer and `guix` contributor. All these projects are structural to the bootstrapping process.
  - More people
- My goal was to pay people for their good work, using the success of my previous effort as a lever to get funded.

[5] https://nlnet.nl/project/GNUMes-RISCV-bootstrap/

In pictures

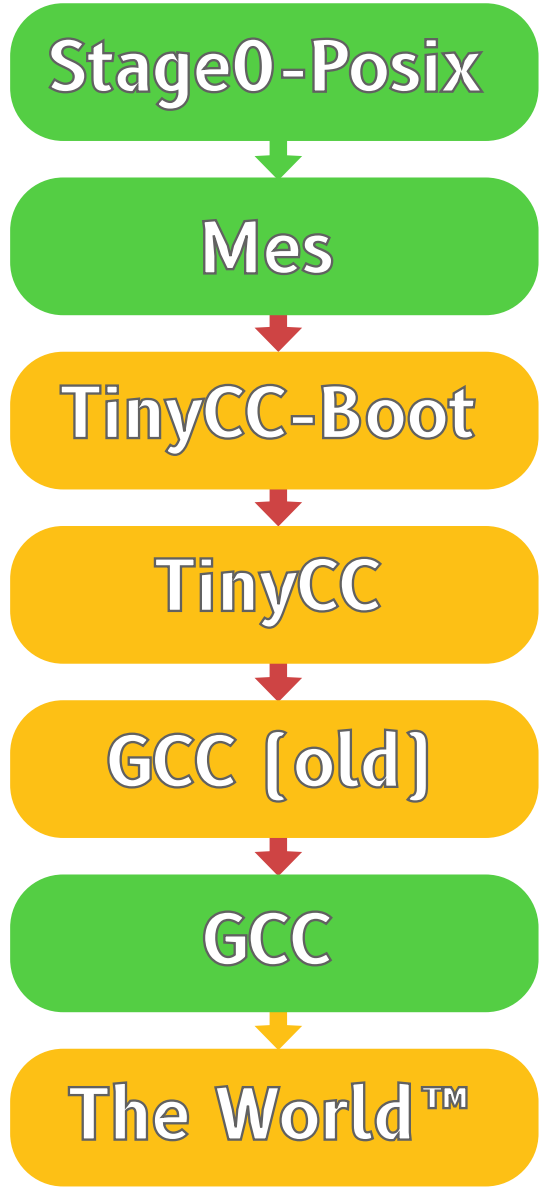
Bootstrapping process - Before

- Colors represent RISC-V support:
  - **RED**: no RISC-V support
  - **ORANGE**: some RISC-V support
  - **GREEN**: full RISC-V support
- **TinyCC-boot** and **GCC (old)** are **RED**.



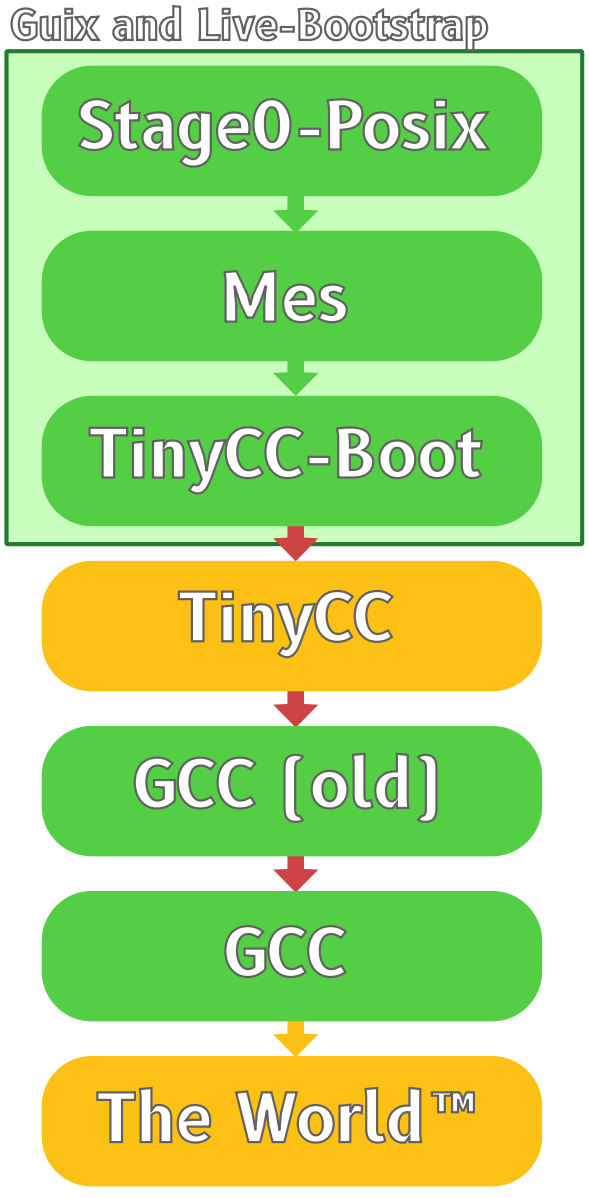
Bootstrapping process - After the Backport (2022)

- **Arrows** are still **RED**
- At that time I didn't know **TinyCC** was **ORANGE**, that was discovered later
- **GCC (old)** was actually **GREENER** than I thought



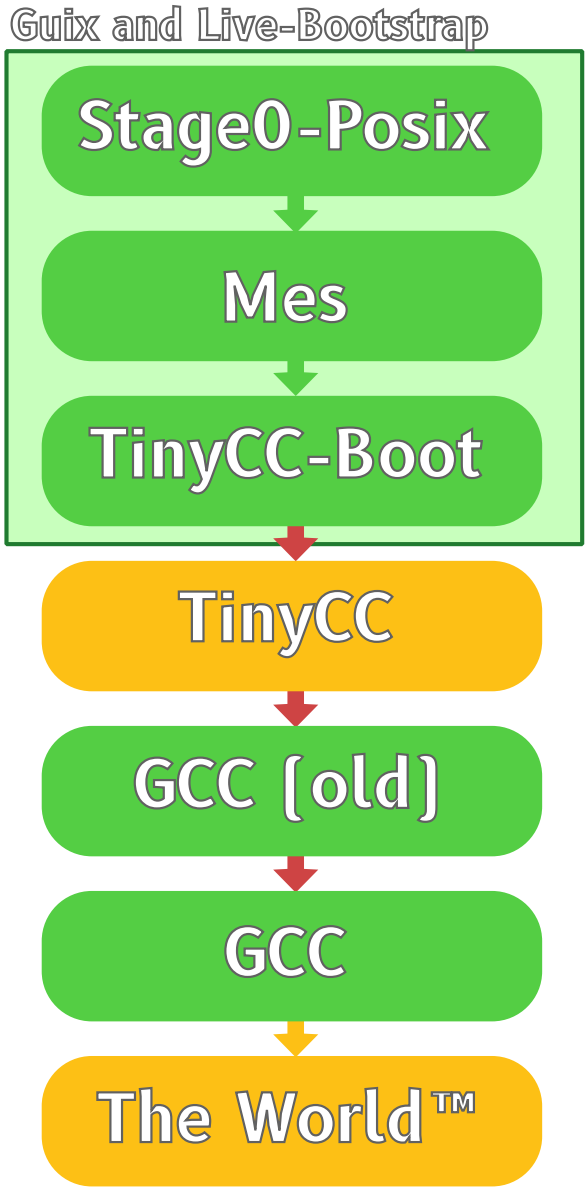
Bootstrapping process - Now

- **TinyCC-Boot** is **GREEN** now
- The **GREEN** rectangle is included in `Guix` and `live-bootstrap`
- **GCC (old)** is **GREEN** now: Tested in Debian in real RISC-V hardware.



Bootstrapping process - Future

- -> **TinyCC** requires changes in MesLibC
- -> **GCC (old)**:
  - requires `make`
  - TinyCC claims to be able to build GCC 4, but we didn't test that (i386 bootstrap uses GCC 2.95 in between).
- -> **GCC** should *just work*
- A powerful `libc` is needed, built with TinyCC, which doesn't support Extended Assembly.



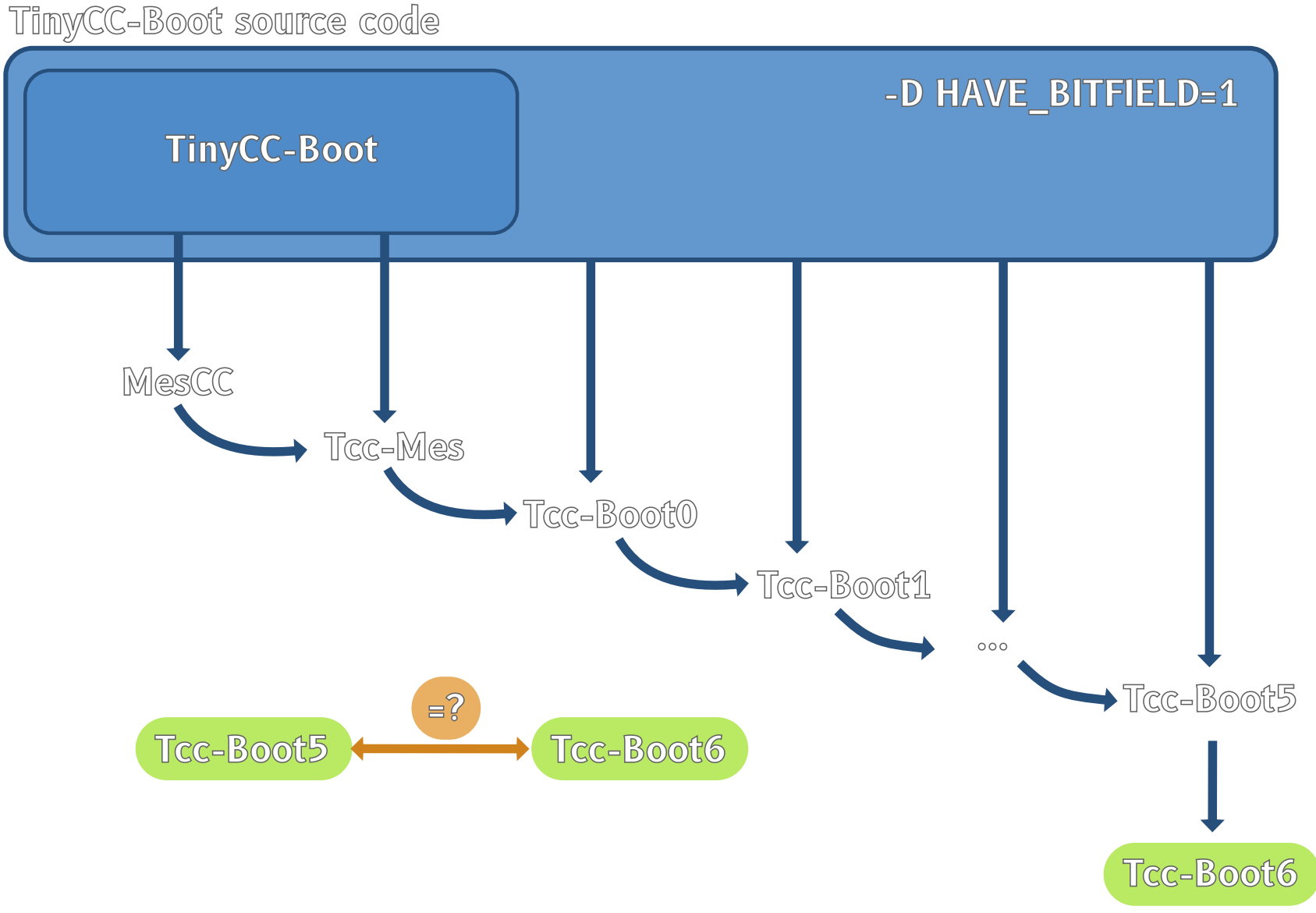
Questions?

Limitations of the backport

TinyCC

- I only tested the backend, in an emulated environment, using a using TinyCC as a cross-compiler
- I tested on **GLibC**, but in the bootstrapping process it uses **MesLibC**, a minimal **LibC**, part of Mes.
- **TinyCC-Boot** is in fact compiled several times (up to 6) with different features (using conditional compilation via macro definitions like `-D...`).
  - In a cross compiled environment this is not possible to test
- I didn't test the TinyCC **RISC-V assembler**, which happened to be unimplemented (yeah... LOL), and it's needed for the **LibC**.

TinyCC



TinyCC-Boot bootstrapping process

GCC

- I tested it as a Cross-Compiler, so only the backend was tested. Again, no bootstrap[6].
- It never compiled itself
- I didn't work on the C++ support

[6] GCC has a bootstrap process that checks if the compiler works correctly. First, it's compiled with your compiler of choice. Then, the resulting binary is used to compile the GCC codebase again. The result of that compiles GCC again. Then the binaries are compared, to make sure they are identical. This can't be done in a cross-compiled environment, for obvious reasons.

The work

TinyCC-Boot

We focused on this, and let the other projects be carried by this effort.

- Andrius and I spent many nights debugging crazy things in here.
- TinyCC's codebase is hard to read
- Many errors we got were not reproducible using a TinyCC compiled with GCC, so we needed to build it from MesCC, which is very slow
- MesCC doesn't provide very helpful debug symbols
- We learned many things as we went

  I wouldn't have the energy to make this without Andrius.

TinyCC-Boot - Crazy things

1. Body is never executed:

```
if ( x < 8 ) {
    // body
}
```

2. `A << 20 >> 20`
3. `__global_pointer$ is not a valid symbol`
4. Assembler uses a weird syntax and doesn't support Extended Assembly
5. `cannot cast from/to void`
6. And segfaults, segfaults everywhere!

Read more about them here[7]

[7] https://ekaitz.elenq.tech/bootstrapGcc8.html

TinyCC-Boot - Results

We finally managed to bootstrap TinyCC-Boot.

- Andrius included the build recipe in `live-bootstrap`[8]
- I did the `guix` recipe[9]

[8] https://github.com/fosslinux/live-bootstrap/blob/master/steps/tcc-0.9.26/pass1.kaem

[9] https://issues.guix.gnu.org/68222

GNU Mes

The problems we found in **TinyCC-Boot** made us improve Mes in several ways:

- **MesCC**, the C compiler, had some limitations that were unnoticed in the i386 bootstrap that appeared in RISC-V: faulty `switch-cases`, wrong `struct` initializations, some integer weirdness…
- **MesLibC**, had to be split for our TinyCC as it doesn't support Extended Assembly, and it also needed some extra tweaks: `char` signedness, `va_args` support, build script support for the changes…

  Having Janneke with us made the process very smooth.

GCC

- We compiled GCC with itself in a Debian machine in real RISC-V hardware
- Including C++ support
- Only needed a few commits![10]
- Guix recipe is pending, as we need to fix other steps to be able to add this: `make`, a proper `libc`, TinyCC...

[10] https://github.com/ekaitz-zarraga/gcc/compare/working-compiler...riscv

Last words

Last words

- People is important: I felt alone, and I didn't know how to continue. If I had to spend another year working alone I would've just rejected the project. **Bringing people gave me energy, knowledge and emotional support**
- Money is important: thanks to NlNet I had the chance to pay people for their work and buy some hardware. Covering people's basic needs is fundamental. **Getting paid makes people independent, so they can focus on what they are good at instead of struggling to survive**

So

- Thanks, Andrius and Janneke
- Thanks, NlNet
- And…

Thank you

Contact and take part

- Email me: ekaitz@elenq.tech[11]
- Relevant IRC channels: `#bootstrappable`, `#guix`, `#guix-risc-v`

[11] mailto:ekaitz@elenq.tech