

# Installation instructions

for tutorial: “A Tour of the RISC-V ISA Formal Specification”

RISC-V Foundation ISA Formal Spec Technical Committee

At RISC-V Summit, San Jose  
December 12, 2019

# Outline

- 1 About this slide deck
- 2 Installation for reading the spec
  - Cloning the SAIL RISC-V ISA Formal Spec
- 3 Installing for building executable version of the spec
  - Installing Opam
  - Installing OCaml using Opam
  - Installing SAIL using Opam
  - Building RISC-V simulators from the SAIL spec

# About this slide deck

This is a standalone slide deck that accompanies the main slide deck for the tutorial: “A Tour of the RISC-V ISA Formal Specification”, first presented at the RISC-V Summit, December 12, 2019, San Jose.

We recommend taking either Step A, or Steps A and B, depending on your objectives, in advance of the tutorial.

**Step A:** If you just want to learn how to *read and consult the spec*

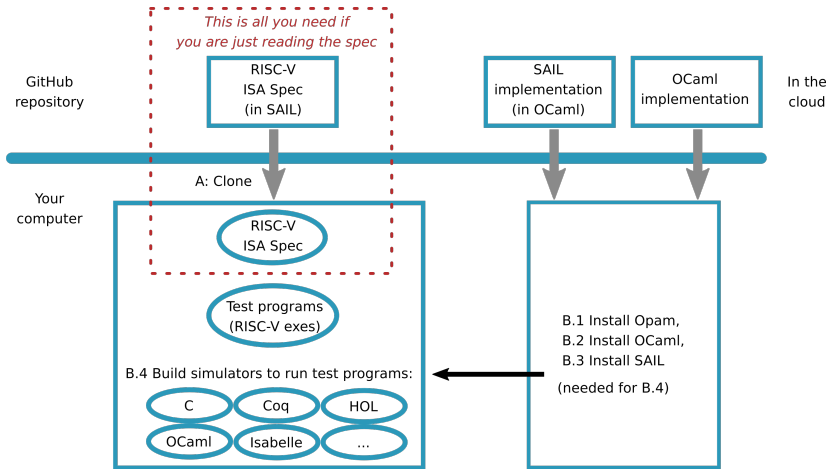
This merely git-clones a certain repo which contains the SAIL source code for the RISC-V ISA Formal Spec.

**Step B:** If you also want to learn how to *execute the spec*

This will compile a RISC-V ISA simulator from the SAIL formal spec, which you can use to execute:

- The standard suite of RISC-V ISA tests
- The standard RISC-V Compliance Test suite
- RISC-V ELF binaries that you create from other source codes

# Installation Overview



## Step A Installation: Cloning the SAIL RISC-V ISA Formal Spec

### Just one step:

```
$ git clone https://github.com/rems-project/sail-riscv
```

### What you get:

```
$ sail-riscv
```

```
$ tree -d
```

```
|-- ...
```

```
|-- model/
```

```
|-- ...
```

*This directory contains all the spec files*

That's all you need, for just reading and consulting (not executing) the spec!

## Step B Installation: to create an executable version of the spec

Reminder: Step B is not necessary if you're only reading and consulting the spec. It's necessary for building an executable version of the spec that can execute RISC-V binaries.

### OS requirements

These instructions are for Debian/Ubuntu Linux. If you are running some other OS:

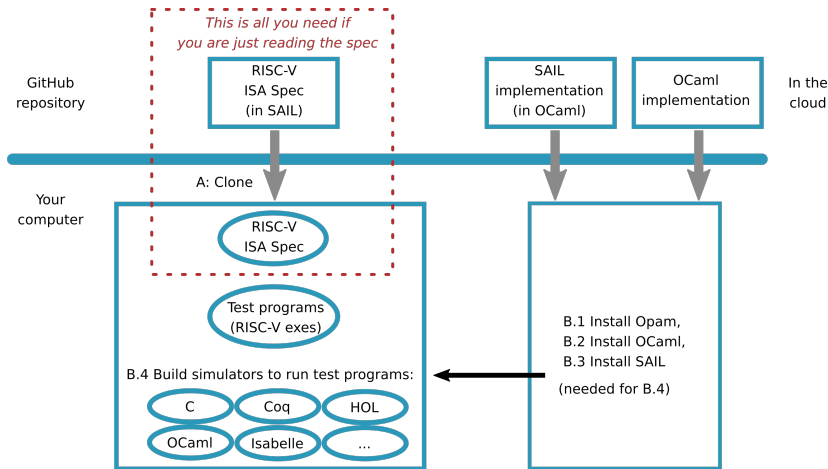
- You could install a virtual machine running Debian/Linux and follow these instructions.
- OCaml and SAIL will also install on other OSes. See the “Safety net” websites (below) for more information.

### Safety net, in case things go wrong:

The instructions in these slides are collected here from various sources for your convenience. In case of trouble, the original full instructions can be found at:

- Installing Opam:  
<https://opam.ocaml.org/doc/Install.html>
- Installing Ocaml for SAIL, and installing SAIL:  
<https://github.com/rem-s-project/sail/wiki/OPAMInstall>

# Installation Overview



## Step B.1: Installing Opam, the package manager for OCaml

### Step B.1: Download the install script and run it

Download <https://raw.githubusercontent.com/ocaml/opam/master/shell/install.sh> Run the script:

```
$ sudo sh install.sh
## Downloading opam 2.0.5 for linux on x86_64...
## ...
## opam 2.0.5 installed to /usr/local/bin
## Run this script again with '--restore' to revert.
```

### Or: Combine the above download-and-run into one line

Install curl if you don't already have it:

```
$ sudo apt-get install curl
```

Then, one line:

```
$ sudo sh <(curl -sL https://raw.githubusercontent.com/ocaml/opam/master/shell/install.sh)
```



## Step B.1: Installing Opam (contd.)

### Step B.1: Verify successful opam installation

```
$ which opam
/usr/local/bin/opam
$ opam --version
2.0.5
```

## Step B.2: Installing OCaml using Opam

Once Opam is installed, you can use it to install OCaml and SAIL. First, OCaml:

### Step B.2: Installing OCaml

```
# Environment setup
$ opam init
$ eval 'opam env'
# Install specific version of OCaml
$ opam switch create ocaml-base-compiler.4.06.1
$ eval 'opam config env'
```

### Verifying we've got OCaml

```
$ which ocaml
/home/nikhil/.opam/ocaml-base-compiler.4.06.1/bin/ocaml
$ ocaml -version

The OCaml toplevel, version 4.06.1
```

Note: 4.06.1 is not the latest version of OCaml, but it is known to be suitable for SAIL (it is the version used during CI of SAIL).

## Step B.3: Installing SAIL using Opam

First, please install certain prerequisite libraries needed by SAIL (if not already installed on your system):

### On Linux (Debian, Ubuntu, ...)

```
$ sudo apt-get install build-essential libgmp-dev z3 m4 pkg-config zlib1g-dev  
$ sudo apt-get install device-tree-compiler Needed by simulator
```

## Step B.3 (contd.): Installing SAIL using Opam

### Set up opam so it knows where to get SAIL

```
$ opam repository add rems https://github.com/rems-project/opam-repository.git
```

### Install SAIL

```
$ opam install sail
```

### Verify we've got it

```
$ which sail
/home/nikhil/.opam/ocaml-base-compiler.4.06.1/bin/sail
$ sail --help
Sail 0.11 (sail2 @ opam)
usage:  sail <options> <file1.sail> ... <fileN.sail>

  -o <prefix> select output filename prefix
  -i start interactive interpreter
...
```

## Step B.4: Building RISC-V simulators from the SAIL spec

A simulator can be used to execute RISC-V binaries.

### Step B.4: Building RISC-V simulators from the SAIL spec:

```
$ cd sail-riscv           i.e., be in the git-cloned directory  
$ make csim  
$ make ARCH=RV32 csim
```

Creates the following executable RISC-V RV64 simulators:

```
c_emulator/riscv_sim_RV64  
c_emulator/riscv_sim_RV32
```

Note: Omitting the 'csim' argument will also build an OCaml-based simulator, and stuff for Coq, Isabelle, HOL4, ... (not necessary for this tutorial).

## Step B.4: Test-drive your RV64 simulator (a smoke-test)

### Example: executing the `rv64ui-p-add` standard ISA test in the simulator

```
$ ./c_emulator/riscv_sim_RV64 test/riscv-tests/rv64ui-p-add.elf
...
tohost located at 0x80001000
...
Running file test/riscv-tests/rv64ui-p-add.elf.
ELF Entry @ 0x80000000
...
[0] [M]: 0x00000000000001000 (0x00000297) auipc t0, 0
...
[1] [M]: 0x00000000000001004 (0x02028593) addi a1, t0, 32
...
[2] [M]: 0x00000000000001008 (0xF1402573) csrrs a0, zero, mhartid
...
[477] [M]: 0x00000000800000044 (0xFC3F2023) sw gp, 4032(t5)
...
SUCCESS
```

During execution of the RISC-V binary, it prints out a trace of instructions executed.

## Step B.4: Test-drive your RV32 simulator (a smoke-test)

### Example: executing the `rv32ui-p-add` standard ISA test in the simulator

```
$ ./c_emulator/riscv_sim_RV32 test/riscv-tests/rv32ui-p-add.elf
...
tohost located at 0x80001000
...
Running file test/riscv-tests/rv32ui-p-add.elf.
ELF Entry @ 0x80000000
...
[0] [M]: 0x00001000 (0x00000297) auipc t0, 0 ...
[1] [M]: 0x00001004 (0x02028593) addi a1, t0, 32 ...
[2] [M]: 0x00001008 (0xF1402573) csrrs a0, zero, mhartid ...
[472] [M]: 0x80000044 (0xFC3F2023) sw gp, 4032(t5) ...
SUCCESS
```

During execution of the RISC-V binary, it prints out a trace of instructions executed.