Computer Organization HW 0: Environmental Setup

Overview

This document can guide you through the process of preparing the experimental environment required by the programming assignments (HWs) in the Computer Organization course.

In particular, to build the experimental environment, we will show the procedures to install:

1. GNU toolchian for RISC-V (commit hash 59ab58e),
2. RISC-V ISA simulator (Spike; commit hash a22119e), and
3. Proxy kernel (commit hash f036859).

The environmental setup can be done either:

by installing the **virtual machine** or

by building from the **source codes** (of the above three software projects).

If you choose to install the virtual machine, you can leverage our prebuilt environment, including Ubuntu Linux 20.04.05 and the above three tools. Please refer to "**A. Install the pre-built environment**" for more details.

If you choose to build from the source codes, you will need to download the source projects directly from their websites and build the projects from scratch. Please refer to "**B. Build from scratch**" for more details.

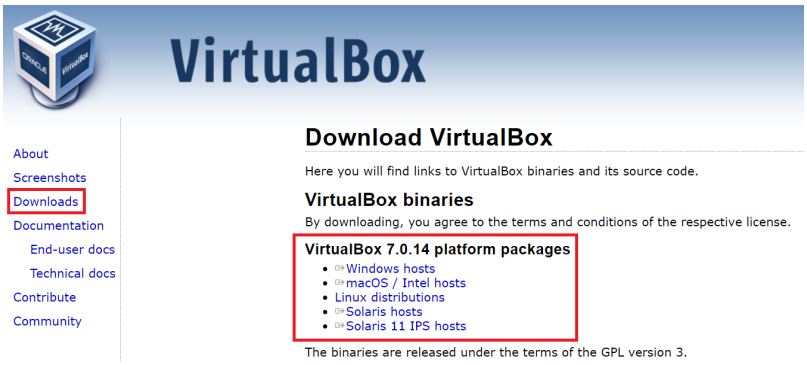
After you installed the tools, you can refer to "**Test the RISC-V tools**" to make sure that the tools are installed successfully

A. Install the pre-builded environment

I. Install VirtualBox (the virtual machine software) Download link

Click the above *download link*, and you will enter the download page.

Please choose, download, and install the proper version (e.g., Windows, Linux, or macOS/Intel hosts) on your host machine, as highlighted in the image below.



II. Import the prebuilt virtual machine image

You will need to download and uncompress the prebuilt image.

Then, you should import the uncompressed image to VirtualBox.

The steps to bring up the virtualized Ubuntu environment are listed as follows.

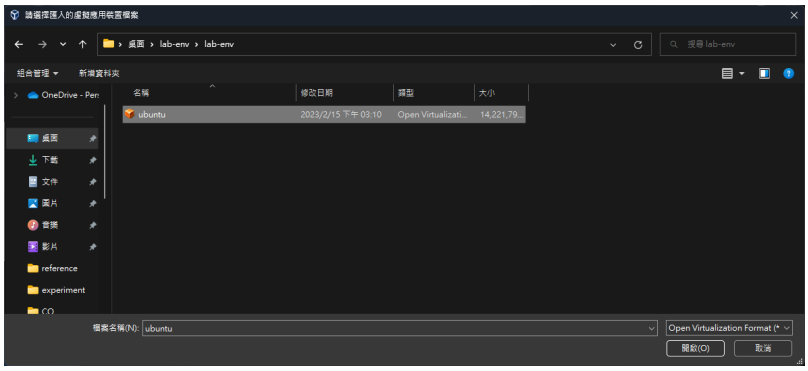
1. Download the prebuilt image `Computer Organization HW.zip` via the link.
2. Uncompress `Computer Organization HW.zip`

> **user id**: ubuntu
>
> **password**: ubuntu

3. Open your installed VirtualBox software
4. Click `Import` button

5. Click the `Browse file` button

6. Choose the file: `Computer Organization HW.ova` from the folder for the uncompressed zip file

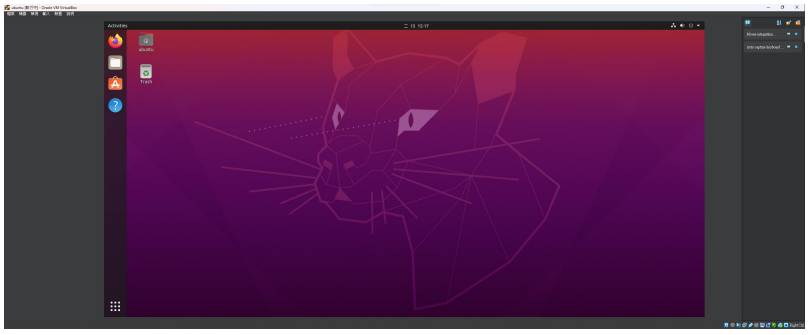


7. Click `Finish` button

8. Wait for VirtualBox to load the virtual environment

9. You will see the **Computer Organization HW** virtual machine of the prebuilt environment on the left side bar This means you have imported the prebuilt environment successfully Click 啟動 to power on the virtual machine

10. A new window is created shown the desktop environment of the ubuntu system This means you successfully start the Ubuntu Linux system



B. Build from scratch

I. Ubuntu installation

Before installing the three RISC-V tools, you need to make sure that your host machine runs Ubuntu Linux (e.g., 20.04.05).

This means that you can either install the Ubuntu Linux on the virtual machine (created by VirtualBox) by yourself. You can refer to the document to know how to create a Ubuntu Linux system with VirtualBox on your own.

Or, it means that you have a machine installed with a Ubuntu Linux environment.

At this point, we assume that you have a workable Ubuntu Linux environment. Please proceed with the following instructions to install the three RISC-V tools

II. Install GNU Toolchain for RISC-V (GCC tools)

- Create a package folder under the `/opt` folder (i.e., `/opt/riscv`) and add the `/opt/riscv` into the environment variable `$RISCV`. Later, you should also create a subfolder `bin` within your package folder (e.g., `/opt/riscv/bin`) and add the `/opt/riscv/bin` into the environment variable `$PATH`. The related commands are listed below.

  ```
  $ sudo mkdir /opt/riscv && sudo mkdir /opt/riscv/bin
  $ echo "export RISCV=/opt/riscv" >> ~/.bashrc
  $ echo "export PATH="$PATH":/opt/riscv/bin" >> ~/.bashrc
  $ source ~/.bashrc
  ```

- After that, you should create a project folder (e.g., `$HOME/riscv`). You should clone the three tools beneath and write your homework in the project folder.

  ```
  $ mkdir ~/riscv
  ```

  The directory tree is shown as below.

  ```
  $HOME/riscv/
          ├── riscv-gnu-toolchain
          ├── riscv-pk
          ├── riscv-isa-sim
          └── CO_StudentID_HW1/
  ```

- Run the following commands to Install the tool:

  - The `sudo apt install` command retrieve and install the necessary packages for the installation of the GNU toolchian.

- The `git clone` command pulls the *latest* version from the **riscv-gnu-toolchain** project repository (the latest version would be different from the version listed in the top of this document).
- The `./configure` command is used to set up the environment variables before building the project.
- The `sudo make linux -j4` command uses four threads to build the project in parallel. The number of threads can be adjusted to fit the cpu.

```
$ cd ~/riscv
$ sudo apt update
$ sudo apt-get install autoconf automake autotools-dev curl python3 python3-
pip libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex
texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev ninja-build git
cmake libglib2.0-dev
$ git clone https://github.com/riscv/riscv-gnu-toolchain
$ cd riscv-gnu-toolchain
$ ./configure --prefix=$RISCV --enable-multilib
$ sudo make linux -j4
```

- After the above operations, the **riscv-gnu-toolchain** will be installed under the path: `$RISCV/riscv-gnu-toolchain` and the **riscv64-unknown-linux-gnu** packages will be installed under the path: `$RISCV`. Please double check the content of `$PATH` before you run the following command to test if the toolchain is installed properly The following command is used to show the version information of the installed GCC compiler (the expected output message is shown below.)

```
$ riscv64-unknown-linux-gnu-gcc -v
```

> gcc version 12.2.0 (xPack GNU RISC-V Embedded GCC x86_64)

III. Install the RISC-V ISA simulator (Spike)

The following commands are used to install the simulator and are similar to those shown above.

Please change the working directory to `$HOME/riscv` (e.g., `cd ~/riscv/`) before running the following commands.

The commands below build the Spike simulator in the folder `~/riscv/riscv-isa-sim/build/spike`, and Spike will be installed under the path: `~/riscv/riscv-isa-sim`.

```
$ cd ~/riscv
$ sudo apt install device-tree-compiler libboost-regex-dev libboost-all-dev
$ git clone https://github.com/riscv/riscv-isa-sim.git
$ cd riscv-isa-sim
$ mkdir build
$ cd build
$ ../configure --prefix=$RISCV
```

```
$ make
$ sudo make install
```

The correctness of the Spike installation can be tested via the instructions listed in "**Test the RISC-V tools**"

IV. Install Proxy Kernel

The proxy kernel is used to redirect I/O system in Spike to the host machine, so that the system calls made in the virtualized RISC-V environment can be emulated and run properly. This is because there is no external I/O device simulated by Spike and Spike needs proxy kernel so that the I/O operations can be done properly

Please change the working directory to `$HOME/riscv` (e.g., `cd ~/riscv/`) before running the following commands.

The commands below build the PK in the folder `~/riscv/riscv-pk`, and Proxy Kernel will be installed under the path: `~/riscv/riscv-pk`.

```
$ cd ~/riscv
$ git clone https://github.com/riscv/riscv-pk.git
$ cd riscv-pk
$ mkdir build
$ cd build
$ ../configure --prefix=$RISCV --host=riscv64-unknown-linux-gnu --with-
arch=rv64gc_zifencei
$ make
$ sudo make install
```

> Checkout your riscv toolchain architecture (riscv64-unknown-linux-gnu-gcc or riscv32-unknown-elf-gcc) in $RISCV/bin, and make sure the toolchain architecture is compatible with your --host flag.

The correctness of the PK installation can be tested via the instructions listed in "**Test the RISC-V tools**"

Test the RISC-V tools

To validate the above tools are installed correctly, you can compile (with the compiler `riscv64-unknown-linux-gnu`) and run an example program (e.g., `hello.c`) with Spike.

- You should prepare a valide C program; e.g., `hello.c` as shown below

  ```
  #include <stdio.h>

  int main(){
      printf("Hello\n");
      return 0;
  }
  ```

- Compile the above C program with the installed RISCV GNU Toolchain, i.e., the GCC compiler If the terminal cannot find the `riscv64-unknown-linux-gnu-gcc` command, you need to check if your environment vairable `$PATH` is set properly

  The command suggests that the Spike simulator supports the `RV64GC` RISC-V variant (More about the RISC-V ISA base and extensions can refer to the page) NOTE: in order to use the Spike to simulate the program, the C program(s) should be built **statically** (with the `-static` flag)

  ```
  $ riscv64-unknown-linux-gnu-gcc -static -o hello hello.c
  ```

- Run the executable file `hello` (generated by the above command) with Spike with the supported of *pk* The binary executable `hello` is run with the support of the built proxy kernel software under the path: `$RISCV/riscv64-unknown-linux-gnu/bin/pk`

  ```
  $ spike --isa=RV64GC $RISCV/riscv64-unknown-linux-gnu/bin/pk hello
  ```

- The simulation result after you run the above command looks like below