



RISC_V目标平台介绍

软件所智能软件中心PLCT实验室郑志文实习生

2020/02/12

目录

01 RISC_V 介绍

02 Spike 介绍和安装

03 RISCV-QEMU 介绍、安装和使用

初识 RISC_V

RISC-V(读作“RISC-FIVE”)是基于**精简指令集计算(RISC)原理**建立的开放指令集架构(ISA)，V表示为第五代RISC(精简指令集计算机),表示此前已经四代RISC处理器原型芯片。



RISC-V标识图

初识 RISC_V


- 每一代RISC处理器都是在同一人带领下完成，那就是加州大学伯克利分校的David A. Patterson教授。
- RISC-V 可以免费地用于所有希望的设备中，允许任何人设计、制造和销售RISC-V芯片和软件。
- 基于RISC-V指令集架构可以设计服务器CPU，家用电器cpu，工控cpu和用在微型传感器中的cpu。

01 RISC_V 介绍

初识 RISC_V

RISC-V基金会的网站: <https://riscv.org/specifications/>

[Join the Mailing Lists](#)info@riscv.org[📄](#)[🐦](#)[in](#)[▶](#)[📡](#) | [Member Login](#)

[ABOUT](#)[MEMBERSHIP](#)[SPECS & SUPPORT](#)[CORES & TOOLS](#)[NEWS](#)[EVENTS](#)[🔍](#)

RISC-V Cores

[🏠 / RISC-V Cores](#)

RISC-V SPECIFICATIONS

- [Unprivileged Specification](#)
- [Privileged ISA Specification](#)
- [Debug Specification](#)

RISC-V SOFTWARE

- [🔗 Software Status](#)

RISC-V CORES

- [🔗 RISC-V Cores](#)

RISC-V EDUCATION

RISC-V Cores and SoC Overview

This document captures the status of various cores and SoCs that endeavor to implement the RISC-V specification. Note that none of these cores/SoCs have passed the in-development RISC-V compliance suite.

Please add to the **list** and fix inaccuracies.

[↑](#)

RISC_V 特点

- **完全开源** 开源采用宽松的**BSD协议**，企业完全自由免费使用，同时也容许企业添加自有指令集拓展而不必开放共享以实现差异化发展。
- **架构简单** RISC-V基础指令集则只有**40多条**，加上其他的模块化扩展指令总共**几十条指令**。RISC-V的规范文档仅有**145页**。
- **易于移植*nix** RISC-V提供了**特权级指令**和**用户级指令**，使开发者能非常方便的移植linux和unix系统到RISC-V平台。

RISC_V 特点

- **模块化设计** 用户能够灵活选择不同的模块组合，来实现自己定制化设备的需要。
- **完整的工具链** RISC-V社区提供了完整的工具链，并且RISC-V基金会持续维护该工具链。当前RISC-V的支持已经合并到主要的工具中，比如编译工具链gcc, 仿真工具qemu等。
- **社区贡献** 完整的工具链维护，大量的开源项目。risc-v的google讨论组(名称: **RISC-V ISA Dev**)吸引各地自愿者参与讨论来不断改进risc-v架构。

RISC和CISC架构

CISC(复杂指令集计算机)

CISC要用最少的机器语言指令来完成所需的计算任务

早期的CPU全部是CISC架构

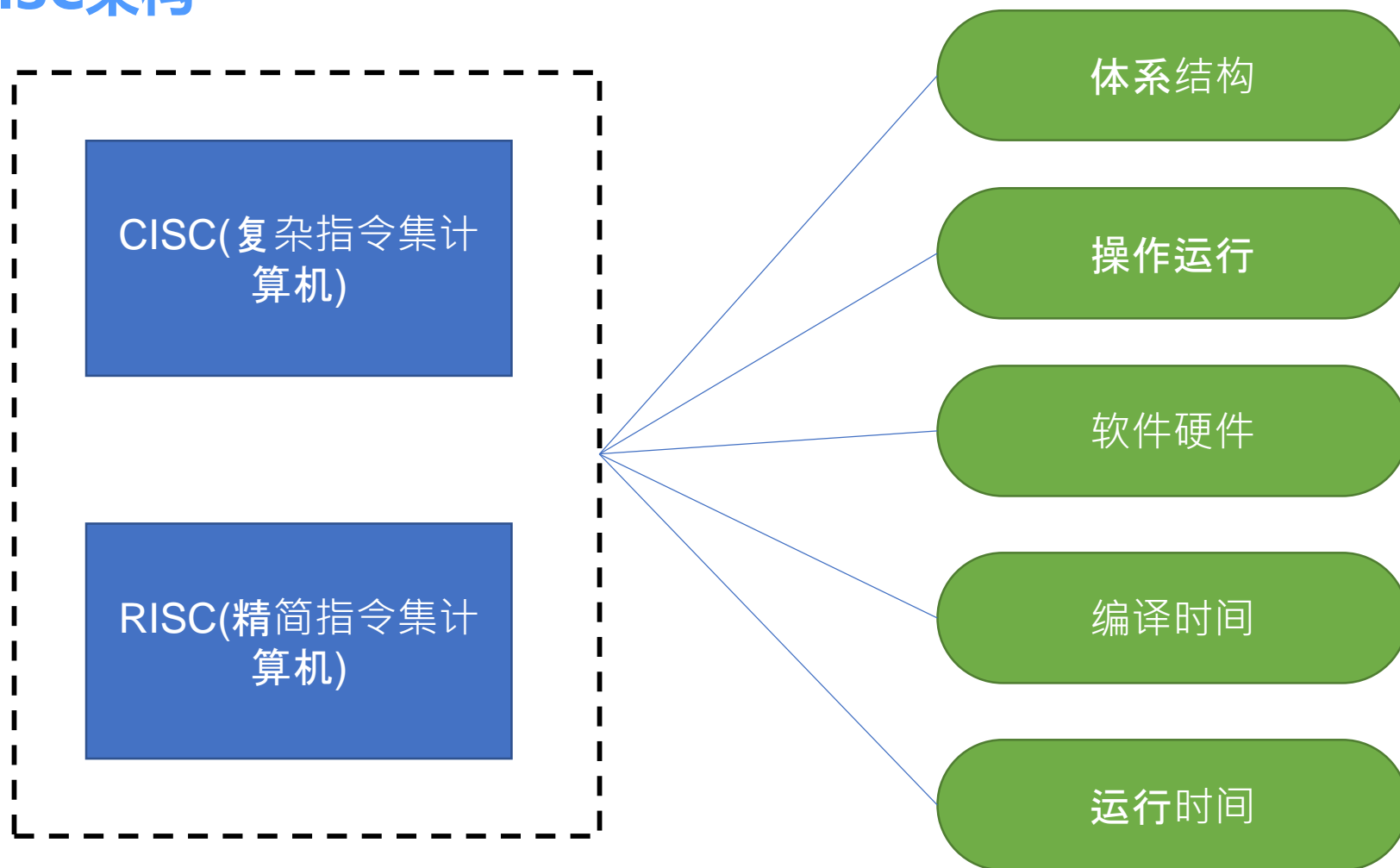
X86架构采用CISC

RISC(精简指令集计算机)

非常适用于移动通信领域，具有低成本、高性能和低耗电的特性

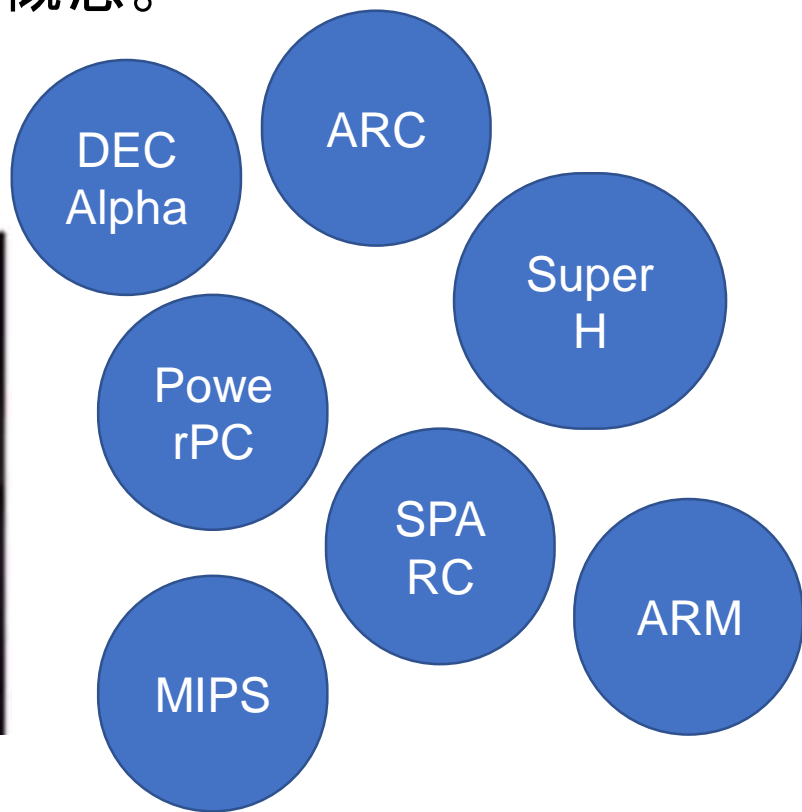
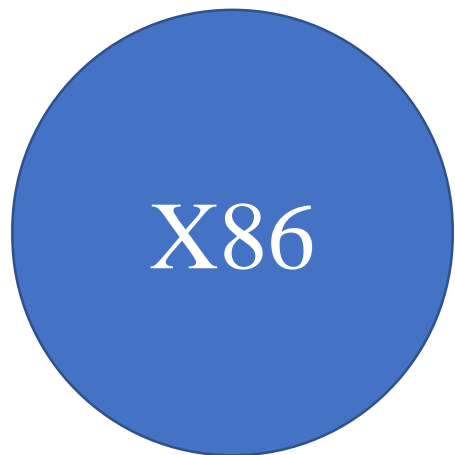
大幅简化架构，仅保留所需要的指令，可以让整个处理器更为简化，拥有小体积、高效能的特性。

RISC和CISC架构



RISC和CISC架构

纽约约克镇IBM研究中心的John Cocke证明，计算机中约20%的指令承担了80%的工作，他于1974年提出了RISC的概念。



RISC和CISC架构

总结一下CISC和RISC的主要区别

指令系统类型	指令	寻址方式	实现方式	其它
CISC (复杂)	数量多，使用频率差别大，可变长格式	支持多种	微程序控制技术(微码)	研制周期长
RISC (精简)	数量少，使用频率接近,定长格式，大部分为单周期指令，操作寄存器，只有Load/Store操作内存	支持方式少	增加了通用寄存器;硬布线逻辑控制为主;适合采用流水线	优化编译，有效支持高级语言

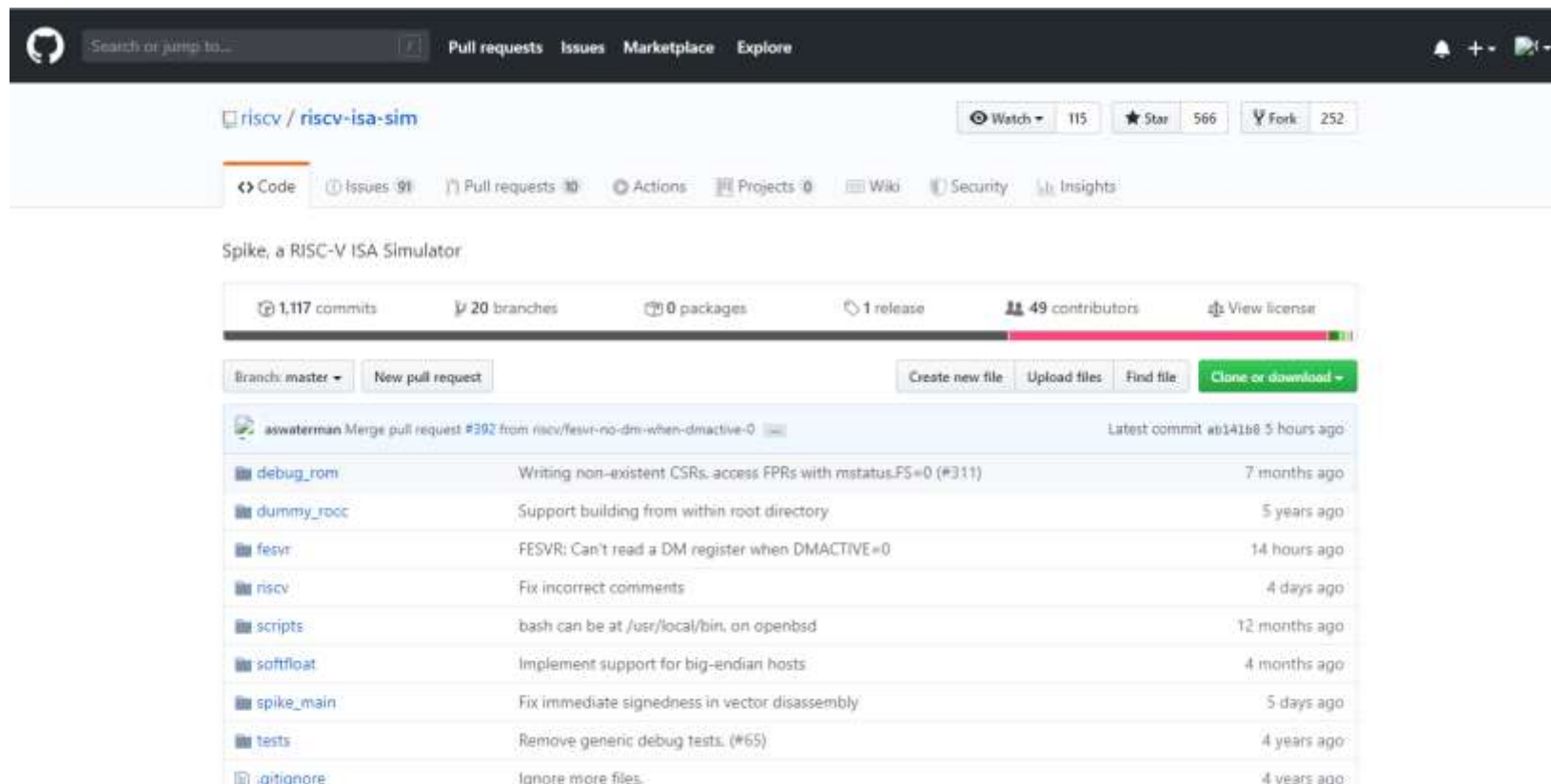
Spike 介绍

Spike RISC-V ISA Simulator 是RISC-V ISA的模拟器，可以运行经过交叉编译工具链编译过的二进制文件，简单来说是一款模拟器。

02 Spike 介绍和安装

Spike 介绍

spike源代码网址: <https://github.com/riscv/riscv-tools>

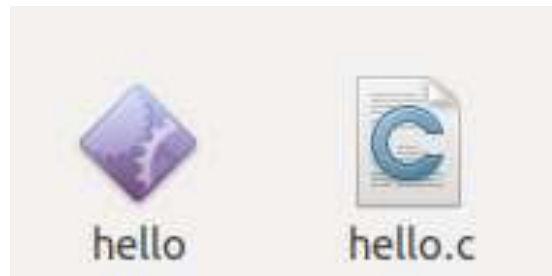


Spike 安装

Spike 是集成在RISC-V编译工具箱中的，往往是在搭建好RISC-V环境之后安装Spike编译工具。

上节课我们编译安装了RISC-V的交叉编译工具riscv64-unknown-linux-gnu-gcc，这节课我们在此基础上安装Spike。

目前路径下有这样两个文件



Spike 安装

riscv-tools安装中已包含了Spike和pk的相关文件。首先安装Spike（顺序不影响）。
在riscv-isa-sim路径下打开终端：

```
$ apt-get install device-tree-compiler
```

```
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim$ sudo apt-get
install device-tree-compiler
[sudo] zzw 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装:
  device-tree-compiler
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 293 个软件包未被升
级。
需要下载 239 kB 的归档。
解压缩后会消耗 441 kB 的额外空间。
获取:1 http://cn.archive.ubuntu.com/ubuntu bionic/main amd64 device-tree-compile
r amd64 1.4.5-3 [239 kB]
已下载 239 kB，耗时 4秒 (65.7 kB/s)
正在选中未选择的软件包 device-tree-compiler。
(正在读取数据库 ... 系统当前共安装有 187489 个文件和目录。)
正准备解包 .../device-tree-compiler_1.4.5-3_amd64.deb ...
正在解包 device-tree-compiler (1.4.5-3) ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
正在设置 device-tree-compiler (1.4.5-3) ...
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim$
```

Spike 安装

riscv-tools安装中已包含了Spike和pk的相关文件。首先安装Spike（顺序不影响）。
在riscv-isa-sim路径下打开终端：

```
$ mkdir build  
$ cd build  
$ ../configure --prefix=$RISCV  
$ make  
$ [sudo] make install
```


02 Spike 介绍和安装

Spike 安装

```
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim$ mkdir build
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim$ cd build
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim/build$ ../configure --prefix=$RISCV
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking for ar... ar
checking for ranlib... ranlib
checking for dtc... /usr/bin/dtc
```

```
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-isa-sim/build$ sudo make install
../scripts/mk-install-dirs.sh /include
mkdir /include
for file in fesvr/elf.h fesvr/elfloader.h fesvr/htif.h fesvr/dtm.h fesvr/memif.h
fesvr/syscall.h fesvr/context.h fesvr/htif_pthread.h fesvr/htif_hexwriter.h fesvr/option_parser.h fesvr/term.h fesvr/device.h fesvr/rfb.h fesvr/tsi.h; \
do \
    ../scripts/mk-install-dirs.sh /include/`dirname $file`; \
    /usr/bin/install -c -m 444 ../$file /include/`dirname $file`; \
done
mkdir /include/fesvr
../scripts/mk-install-dirs.sh /lib
for file in libfesvr.a; \
do \
    /usr/bin/install -c -m 644 $file /lib; \
done
../scripts/mk-install-dirs.sh /bin
for file in elf2hex spike spike-dasm xspike termios-xspike; \
do \
    /usr/bin/install -c -m 555 $file /bin; \
```

运行结果截图

02 Spike 介绍和安装

Spike 安装

然后安装pk。

在riscv-pk路径下打开终端：

```
$ mkdir build  
$ cd build  
$ ../configure --prefix=$RISCV --host=riscv64-unknown-linux  
$ make  
$ make install
```

02 Spike 介绍和安装

Spike 安装

```
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-pk$ mkdir build
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-pk$ cd build
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-pk/build$ ../configure
--prefix=$RISCV --host=riscv64-unknown-elf
checking build system type... x86_64-pc-linux-gnu
checking host system type... riscv64-unknown-elf
checking for riscv64-unknown-elf-gcc... riscv64-unknown-elf-gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... yes
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether riscv64-unknown-elf-gcc accepts -g... yes
checking for riscv64-unknown-elf-gcc option to accept ISO C89... none needed
checking for riscv64-unknown-elf-g++... riscv64-unknown-elf-g++
checking whether we are using the GNU C++ compiler... yes
checking whether riscv64-unknown-elf-g++ accepts -g... yes
checking for riscv64-unknown-elf-ar... riscv64-unknown-elf-ar
checking for riscv64-unknown-elf-ranlib... riscv64-unknown-elf-ranlib
checking for riscv64-unknown-elf-readelf... riscv64-unknown-elf-readelf
checking for riscv64-unknown-elf-objcopy... riscv64-unknown-elf-objcopy
checking for a BSD-compatible install... /usr/bin/install -c
checking how to run the C preprocessor... riscv64-unknown-elf-gcc -E
```

```
(base) zzw@zzw-ThinkCentre-M828z-D101:~/riscv-tools/riscv-pk/build$ make
riscv64-unknown-elf-gcc -MMD -MP -Wall -Werror -D__NO_INLINE__ -mcmodel=medany -
O2 -std=gnu99 -Wno-unused -Wno-attributes -fno-delete-null-pointer-checks -fno-P
IE -DBBL_PAYLOAD=\"bbl_payload\" -DBBL_LOGO_FILE=\"bbl_logo_file\" -I. -I../p
k -I../bbl -I../softfloat -I../dummy_payload -I../machine -I../util -c ../pk/fil
e.c
riscv64-unknown-elf-gcc -MMD -MP -Wall -Werror -D__NO_INLINE__ -mcmodel=medany -
O2 -std=gnu99 -Wno-unused -Wno-attributes -fno-delete-null-pointer-checks -fno-P
IE -DBBL_PAYLOAD=\"bbl_payload\" -DBBL_LOGO_FILE=\"bbl_logo_file\" -I. -I../p
k -I../bbl -I../softfloat -I../dummy_payload -I../machine -I../util -c ../pk/sys
call.c
riscv64-unknown-elf-gcc -MMD -MP -Wall -Werror -D__NO_INLINE__ -mcmodel=medany -
O2 -std=gnu99 -Wno-unused -Wno-attributes -fno-delete-null-pointer-checks -fno-P
IE -DBBL_PAYLOAD=\"bbl_payload\" -DBBL_LOGO_FILE=\"bbl_logo_file\" -I. -I../p
k -I../bbl -I../softfloat -I../dummy_payload -I../machine -I../util -c ../pk/han
dlers.c
riscv64-unknown-elf-gcc -MMD -MP -Wall -Werror -D__NO_INLINE__ -mcmodel=medany -
O2 -std=gnu99 -Wno-unused -Wno-attributes -fno-delete-null-pointer-checks -fno-P
IE -DBBL_PAYLOAD=\"bbl_payload\" -DBBL_LOGO_FILE=\"bbl_logo_file\" -I. -I../p
```

运行结果截图

02 Spike 介绍和安装

Spike 安装

此时该模拟器已完成安装，在可执行文件hello的路径中打开终端，输入：


```
(base) zzw@zzw-ThinkCentre-M828z-D101:~$ spike pk hello  
bbl loader  
Hello, World!  
(base) zzw@zzw-ThinkCentre-M828z-D101:~$
```

说明此时spike已经安装成功。

RISC-V-QEMU 介绍

RISC-V-QEMU 也是集成在RISC-V编译工具箱中的，往往是在搭建好RISC-V环境之后安装RISC-V-QEMU编译工具。

[Join the Mailing Lists](#)info@riscv.org[📧](#)[🐦](#)[in](#)[▶](#)[📺](#) | [Member Login](#)

[ABOUT](#) [MEMBERSHIP](#) [SPECS & SUPPORT](#) [CORES & TOOLS](#) [NEWS](#) [EVENTS](#) [🔍](#)

riscv-qemu

🏠 / riscv-qemu

RISC-V SPECIFICATIONS

- [Unprivileged Specification](#)
- [Privileged ISA Specification](#)
- [Debug Specification](#)

RISC-V SOFTWARE

- [🔗 Software Status](#)

RISC-V CORES

- [🔗 RISC-V Cores](#)

RISC-V EDUCATION

- [🔗 RISC-V Educational Materials](#)

[🔗 View on GitHub](#) [🔗 Visit riscv on GitHub](#)

The RISC-V QEMU Port is Upstream

The RISC-V QEMU port is developed in the **upstream QEMU repository**. You may be more interested in the **official releases**.

RISC-V-QEMU 介绍

在risc-tools/riscv-gnu-toolchain目录下，包含了qemu，所以直接cd进这个目录

```
$ ./configure --target-list=riscv64-linux-user  
$ make -j4  
$ make install
```

RISC-V-QEMU 介绍

```
zww@zww-ThinkCentre-M828z-D101: ~/riscv-tools/riscv-gnu-toolchain/qemu
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
(base) zww@zww-ThinkCentre-M828z-D101:~/riscv-tools/riscv-gnu-toolchain/qemu$ ./
configure --target-list=riscv64-linux-user && make -j4
Install prefix      /usr/local
BIOS directory      /usr/local/share/qemu
firmware path       /usr/local/share/qemu-firmware
binary directory    /usr/local/bin
library directory   /usr/local/lib
module directory    /usr/local/lib/qemu
libexec directory   /usr/local/libexec
include directory   /usr/local/include
config directory    /usr/local/etc
local state directory /usr/local/var
Manual directory     /usr/local/share/man
ELF interp prefix   /usr/gnemul/qemu-%M
Source path         /home/zww/riscv-tools/riscv-gnu-toolchain/qemu
GIT binary          git
GIT submodules      ui/keycodemapdb tests/fp/berkeley-testfloat-3 tests/fp/berkeley-softfloat-3 dtc capstone slirp
C compiler           cc
Host C compiler      cc
C++ compiler         c++
Objective-C compiler cc
ARFLAGS             rv
CFLAGS              -O2 -U FORTIFY_SOURCE -D FORTIFY_SOURCE=2 -g
```

运行结果截图

RISC-V-QEMU 介绍

测试方法

```
$ ./riscv64-linux-user/qemu-riscv64 /home/zzw/hello
```


谢 谢

欢迎交流合作

2020/02/12