ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

# 从零开始的RISC-V模拟器开发

## 第0讲 导论

中国科学院软件研究所
PLCT实验室

李威威 liweiwei@iscas.ac.cn
王俊强 wangjunqiang@iscas.ac.cn
吴伟 wuwei2016@iscas.ac.cn

## PLCT Lab 的定位和使命

程序语言与编译技术实验室（**PLCT Lab**）致力于成为编译技术领域的开源领导者，推进开源工具链及运行时系统等软件基础设施的技术革新，具备主导开发和维护重要基础设施的技术及管理能力。与此同时，努力成为编译领域培养尖端人才的黄埔军校，推动先进编译技术在国内的普及和发展。

## PLCT Lab 的定位和使命

程序语言与编译技术实验室（**PLCT Lab**）致力于成为编译技术领域的开源领导者，推进开源工具链及运行时系统等软件基础设施的技术革新，具备主导开发和维护重要基础设施的技术及管理能力。**与此同时，努力成为编译领域培养尖端人才的黄埔军校，推动先进编译技术在国内的普及和发展。**

## 长洲岛计划 （就是人才培养计划，起了个唬人的名字）

**涵盖：**

- ➢ 技术专家培养（学徒制，需要来**PLCT**实验室入职/借调/实习）
- ➢ 本科及研究生教育（高校编译原理课程、高级编译原理讨论班）
- ➢ 慕课及在线教育（面向爱好者和自学者）

## 在2021年四月的PLCT公开课程

- 编译技术入门与实战（**2019·2020·2021**）第三期连载中
- 自己动手写一个 **RISC-V OS** （**2021**）连载中
- 从零开始的**RISC-V**模拟器开发 QQ课程群：**326601025**
- 淦！移植个**V8**不可能这么难

## 在开始学习前需要知道的

- **需要一点计算机组成或体系结构的基础知识，例如知道总线**
- **这门课针对RISC-V讲解Spike和QEMU的技术原理和实现**
- **适合刚刚接到了「扩展RISC-V模拟器」的任务的新人**
- **这是模拟器课程的首次开课，我们摸着学生过河，请多多提问**

如果你有不懂的地方，那么肯定是因为我讲漏掉了知识点 :-)

## 课程介绍

### 课程目标

本课程主要面向初入RISC-V模拟器领域的开发人员，让他们：

了解模拟器的基本运行原理

快速掌握模拟器扩展方法

### 课程安排

每周五20:00 ~22:00 时长不定，小班讨论

B站 录播更新

课程repo：https://github.com/plctlab/writing-your-first-riscv-simulator

## 模拟器是什么?

### 模拟器是:

“A computer simulation (or "sim") is an attempt to model a real-life or hypothetical situation on a computer so that it can be studied to see how the system works.”

参考：https://en.wikipedia.org/wiki/Simulation#Computer_simulation

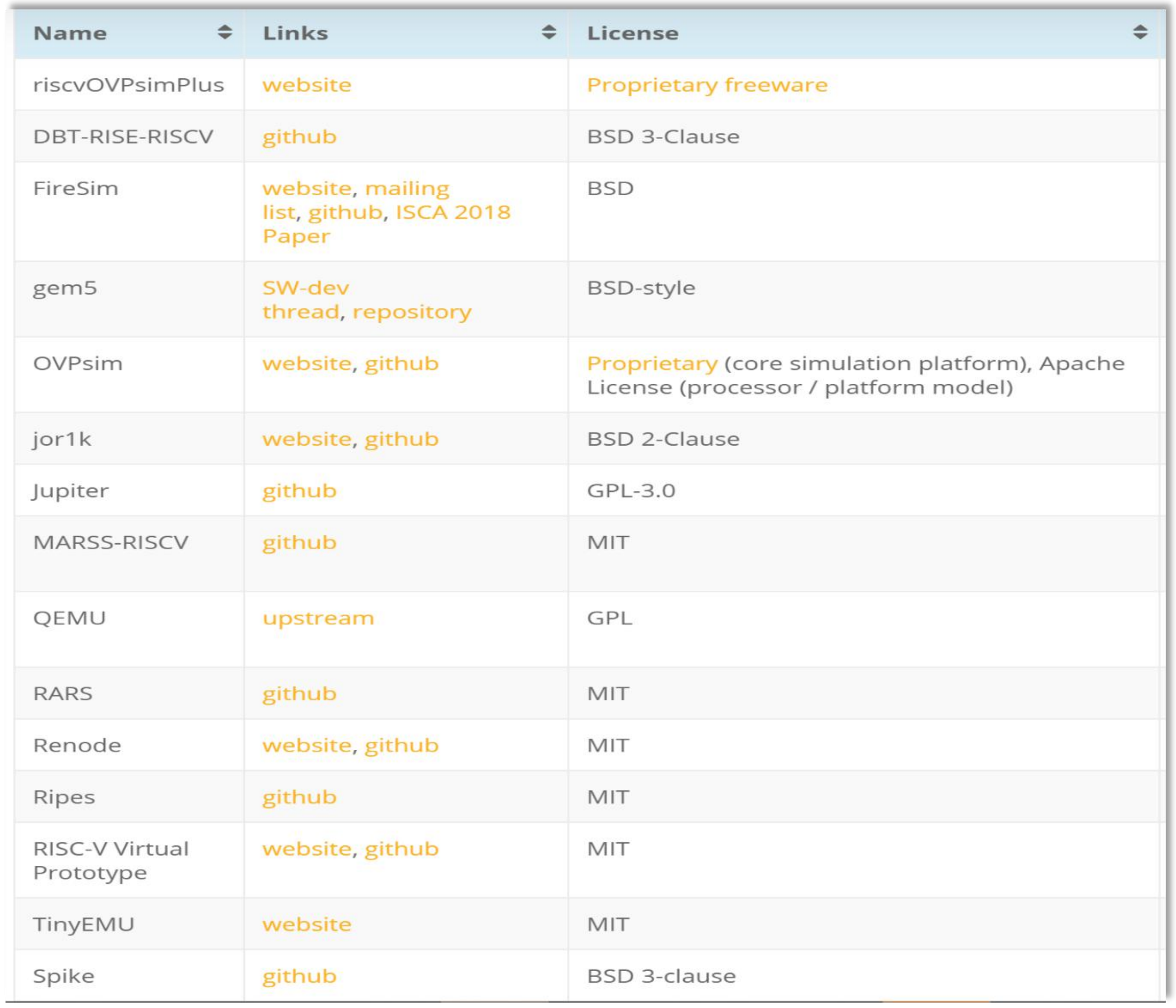
## RISC-V模拟器

### ~~热门模拟器~~(维护力up↑)

- Spike
- QEMU
- riscvOVPsim

### 有兴趣阅读

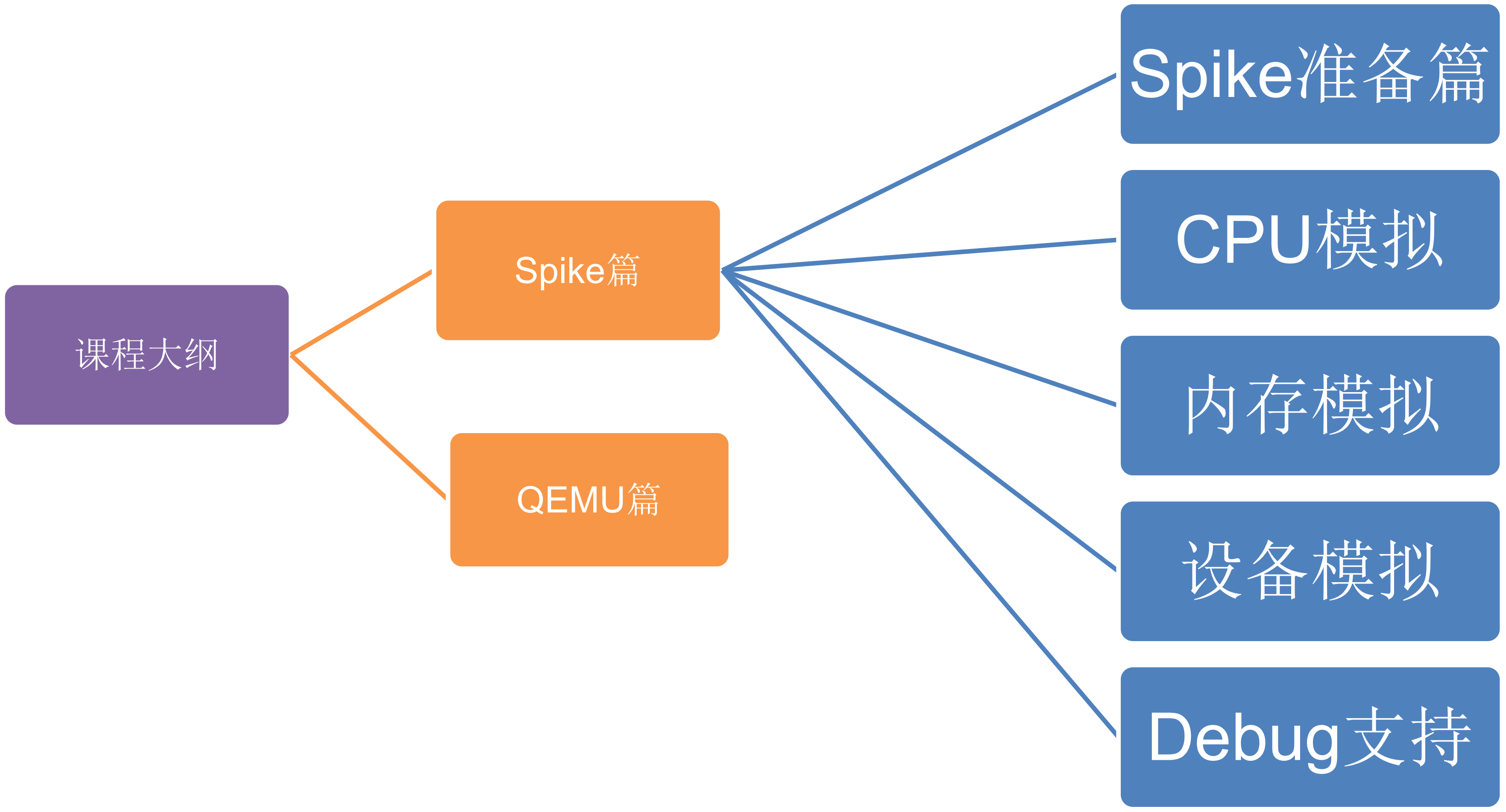
- Gem5
  https://www.gem5.org/
- rv32emu-next
  https://github.com/sysprog21/rv32emu-next.git
- riscv-rust
  https://github.com/takahirox/riscv-rust.git



| Name | Links | License |
|---|---|---|
| riscvOVPsimPlus | website | Proprietary freeware |
| DBT-RISE-RISCV | github | BSD 3-Clause |
| FireSim | website, mailing list, github, ISCA 2018 Paper | BSD |
| gem5 | SW-dev thread, repository | BSD-style |
| OVPsim | website, github | Proprietary (core simulation platform), Apache License (processor / platform model) |
| jor1k | website, github | BSD 2-Clause |
| Jupiter | github | GPL-3.0 |
| MARSS-RISCV | github | MIT |
| QEMU | upstream | GPL |
| RARS | github | MIT |
| Renode | website, github | MIT |
| Ripes | github | MIT |
| RISC-V Virtual Prototype | website, github | MIT |
| TinyEMU | website | MIT |
| Spike | github | BSD 3-clause |

参考：https://riscv.org/exchange/software/#tab-1607480869659-5

## 课程介绍—Spike篇



## Spike模拟器

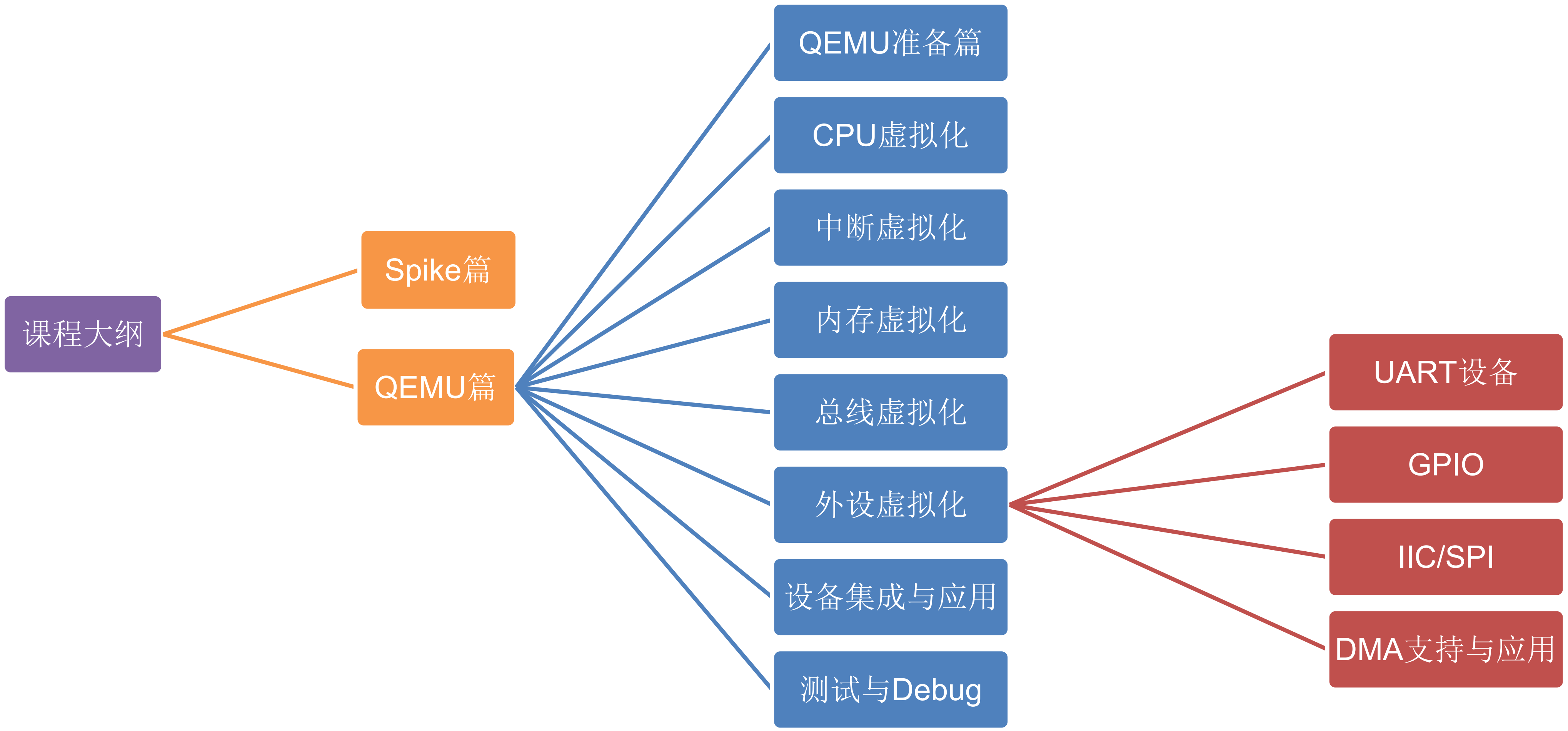
### 基于Spike 实现目标

1.实现对芯来NICE demo指令集的模拟

2.了解Spike对内存，设备模拟以及Debug相关的支持，及相关的扩展方法

| 课程点 | 涉及主要知识点 |
|---|---|
| CPU模拟 | RISC-V CPU实现原理，各种指令集添加方式，指令添加方法 |
| 内存模拟 | Spike cache及内存运行机制 |
| 设备模拟 | Spike的设备模拟机制，plugin mmio设备的添加方法 |
| Debug支持 | Spike的反汇编支持机制以及交互调试机制，及相关扩展方法 |

## 课程介绍—QEMU篇


课程大纲
Spike篇
QEMU篇
QEMU准备篇
CPU虚拟化
中断虚拟化
内存虚拟化
总线虚拟化
外设虚拟化
设备集成与应用
测试与Debug
UART设备
GPIO
IIC/SPI
DMA支持与应用

## QEMU模拟器

### 基于QEMU实现目标

1.实现对芯来HummingBird Evaluation Kit, 200T Board的模拟
2.支持芯来N600 NX600 UX600 CPU运行baremetal demo, RTOS, Linux
3.介绍整体过程中遇到相关实现的原理

| 课程点 | 涉及主要知识点 |
|---|---|
| CPU虚拟化 | RISC-V CPU实现分析，新RISC-V Soc建立，CSR寄存器实现添加，TCG原理，指令添加…… |
| 中断虚拟化 | QEMU RISC-V IRQ机制，Nuclei Timer 和 Eclic 实现，Sifive Clint 和Plic实现…… |
| 内存虚拟化 | QEMU 内存原理介绍，用户态/系统态程序加载运行分析…… |
| 总线虚拟化 | QEMU 总线原理介绍 |
| 外设虚拟化 | 200T上UART实现，GPIO实现与应用， IIC/SPI实现与应用， DMA支持与应用(可能以其他board为例)…… |
| Machine虚拟化 | QEMU的整体运行流程分析，MCU/Linux设备组织方式…… |
| QEMU测试 | 介绍QEMU中主要的测试方法，指令测试方法，设备测试方法…… |
| QEMU Debug | 介绍QEMU中Debug的实现…… |

## 环境准备与介绍

- 交叉工具链

  RISC-V GNU Toolchain
  Nuclei RISC-V Toolchain

- 测试环境

  riscv-tools

  普通应用程序
  Nuclei SDK

- [Spike环境]

- [QEMU环境]

# 环境准备—交叉工具链

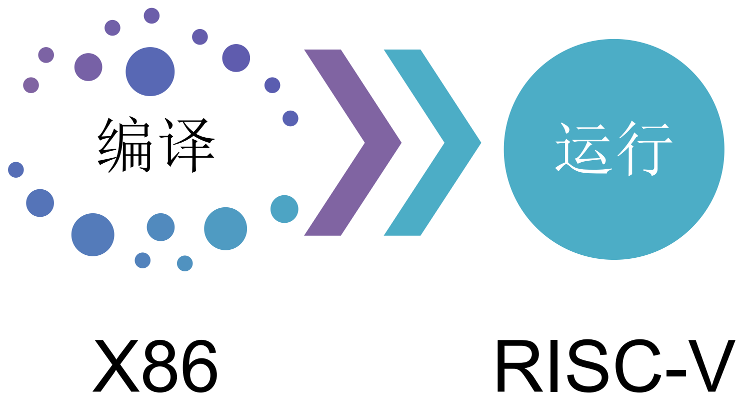
## 交叉工具链

- 交叉（cross）编译

  在一个系统平台上构建一个在另一个系统平台上运行的程序
  比如在x86平台上编译一个RISC-V可执行程序

- 系统类型（system type）

  Build: 程序构建时的系统
  Host: 程序要运行的系统
  Target: 要构建的程序的输出对应的架构



## 环境准备—RISC-V GNU Toolchain

### RISC-V GNU Toolchain 安装

源码获取

```
$git clone https://github.com/riscv/riscv-gnu-toolchain.git
$cd riscv-gnu-toolchain
$git submodule update --init –recursive
```

Warning: git clone takes around 6.65 GB of disk and download size

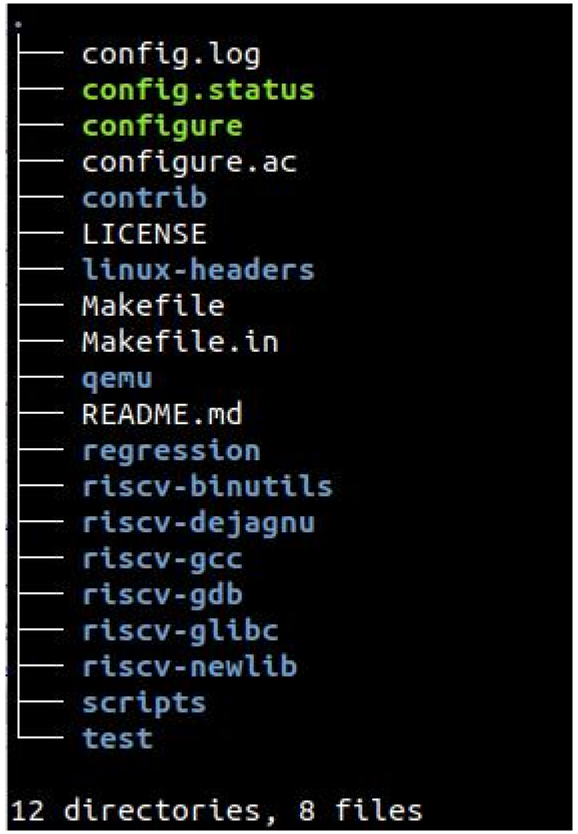
移除qemu(备选)

```
git rm --cached QEMU git submodule
git update --init --recursive
```

安装依赖库(以ubuntu为例)

```
sudo apt-get install autoconf automake autotools-dev curl
python3 libmpc-dev libmpfr-dev libgmp-dev gawk build-
essential bison flex texinfo gperf libtool patchutils bc zlib1g-
dev libexpat-dev
```

Toolchain目录



```
.
├── config.log
├── config.status
├── configure
├── configure.ac
├── contrib
├── LICENSE
├── linux-headers
├── Makefile
├── Makefile.in
├── qemu
├── README.md
├── regression
├── riscv-binutils
├── riscv-dejagnu
├── riscv-gcc
├── riscv-gdb
├── riscv-glibc
├── riscv-newlib
├── scripts
└── test

12 directories, 8 files
```

参考:https://github.com/riscv/riscv-gnu-toolchain/blob/master/README.md

## 环境准备—RISC-V GNU Toolchain

### 编译安装

通用编译安装(32bit and 64bit)

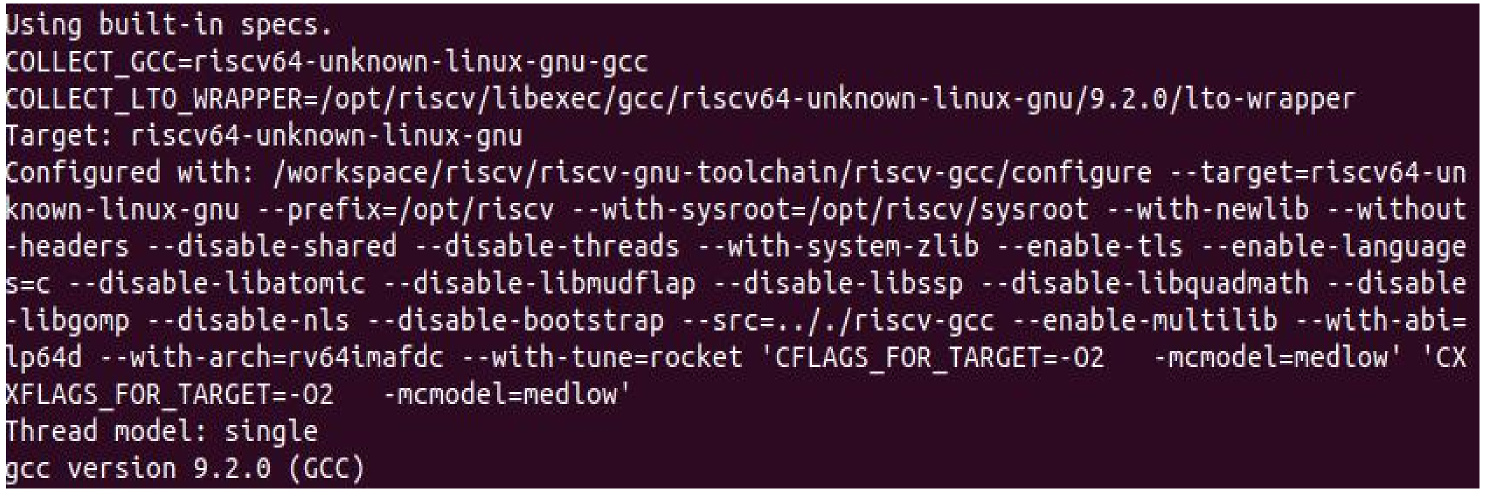
```
$./configure --prefix=/opt/riscv --enable-multilib
$sudo make linux
$sudo make install
$riscv64-unknown-linux-gnu-gcc -v
```

静态编译

```
Installation (Newlib)
./configure --prefix=/opt/riscv
Make
riscv64-unknown-elf-gcc
```

动态编译

```
Installation (Linux)
./configure --prefix=/opt/riscv
make linux
riscv64-unknown-linux-gnu-gcc
```



```
Using built-in specs.
COLLECT_GCC=riscv64-unknown-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/opt/riscv/libexec/gcc/riscv64-unknown-linux-gnu/9.2.0/lto-wrapper
Target: riscv64-unknown-linux-gnu
Configured with: /workspace/riscv/riscv-gnu-toolchain/riscv-gcc/configure --target=riscv64-un
known-linux-gnu --prefix=/opt/riscv --with-sysroot=/opt/riscv/sysroot --with-newlib --without
-headers --disable-shared --disable-threads --with-system-zlib --enable-tls --enable-language
s=c --disable-libatomic --disable-libmudflap --disable-libssp --disable-libquadmath --disable
-libgomp --disable-nls --disable-bootstrap --src=.././riscv-gcc --enable-multilib --with-abi=
lp64d --with-arch=rv64imafdc --with-tune=rocket 'CFLAGS_FOR_TARGET=-O2   -mcmodel=medlow' 'CX
XFLAGS_FOR_TARGET=-O2   -mcmodel=medlow'
Thread model: single
gcc version 9.2.0 (GCC)
```

Configure选项：
--with-arch=
Supported architectures are rv32i or rv64i plus standard extensions (a)tomics, (m)ultiplication and division, (f)loat, (d)ouble, or (g)eneral for MAFD.

--with-abi=
Supported ABIs are
ilp32 (32-bit soft-float), ilp32d (32-bit hard-float),
ilp32f (32-bit with single-precision in registers and double in memory, niche use only),
lp64 lp64f lp64d (same but with 64-bit long and pointers).

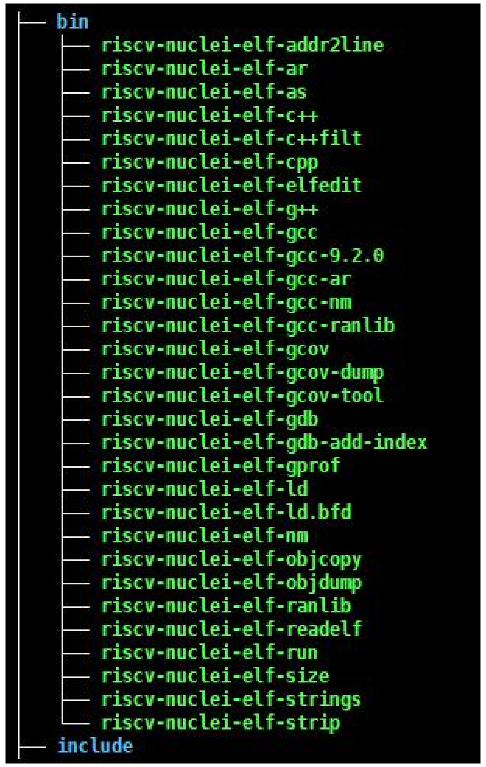
参考:https://github.com/riscv/riscv-gnu-toolchain/blob/master/README.md

## 环境准备—Nuclei RISC-V Toolchain

### NUCLEI RISC-V Toolchain

下载地址:
https://nucleisys.com/download.php

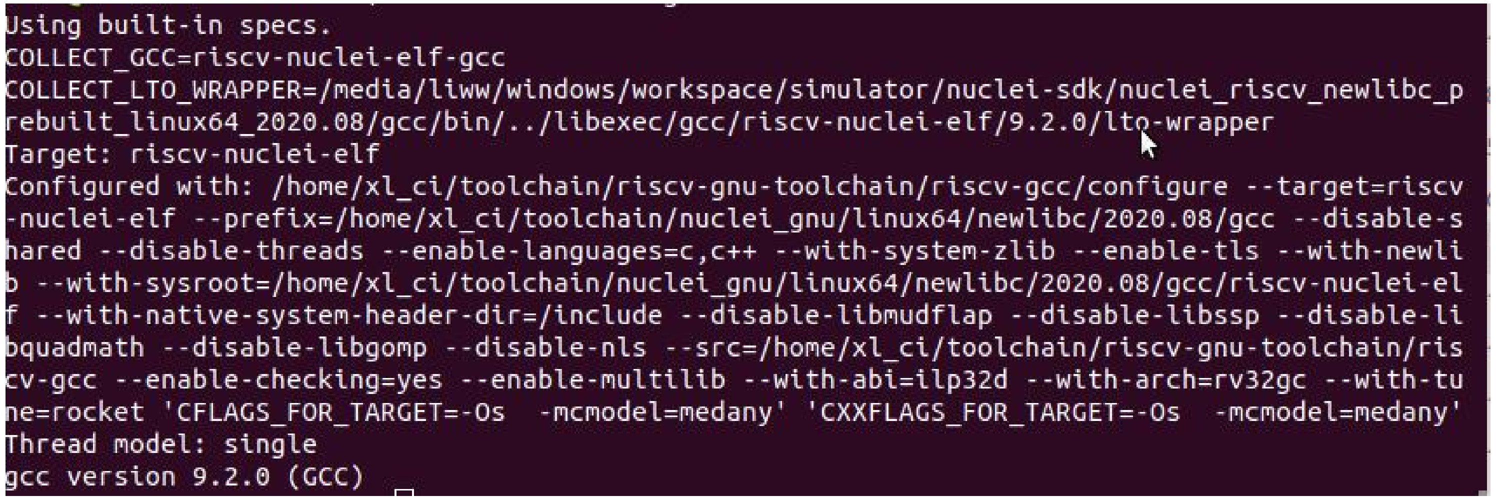
例如: nuclei_riscv_newlibc_prebuilt_linux64_2020.08.tar.bz2



```
├── bin
│   ├── riscv-nuclei-elf-addr2line
│   ├── riscv-nuclei-elf-ar
│   ├── riscv-nuclei-elf-as
│   ├── riscv-nuclei-elf-c++
│   ├── riscv-nuclei-elf-c++filt
│   ├── riscv-nuclei-elf-cpp
│   ├── riscv-nuclei-elf-elfedit
│   ├── riscv-nuclei-elf-g++
│   ├── riscv-nuclei-elf-gcc
│   ├── riscv-nuclei-elf-gcc-9.2.0
│   ├── riscv-nuclei-elf-gcc-ar
│   ├── riscv-nuclei-elf-gcc-nm
│   ├── riscv-nuclei-elf-gcc-ranlib
│   ├── riscv-nuclei-elf-gcov
│   ├── riscv-nuclei-elf-gcov-dump
│   ├── riscv-nuclei-elf-gcov-tool
│   ├── riscv-nuclei-elf-gdb
│   ├── riscv-nuclei-elf-gdb-add-index
│   ├── riscv-nuclei-elf-gprof
│   ├── riscv-nuclei-elf-ld
│   ├── riscv-nuclei-elf-ld.bfd
│   ├── riscv-nuclei-elf-nm
│   ├── riscv-nuclei-elf-objcopy
│   ├── riscv-nuclei-elf-objdump
│   ├── riscv-nuclei-elf-ranlib
│   ├── riscv-nuclei-elf-readelf
│   ├── riscv-nuclei-elf-run
│   ├── riscv-nuclei-elf-size
│   ├── riscv-nuclei-elf-strings
│   └── riscv-nuclei-elf-strip
├── include
```

$tar –xvf nuclei_riscv_newlibc_prebuilt_linux64_2020.08.tar.bz2
得到gcc 目录
在.bashrc 或者其他加入环境变量 新shell生效
export PATH=/path/to/gcc/bin:$PATH

riscv-nuclei-elf-gcc -v



```
Using built-in specs.
COLLECT_GCC=riscv-nuclei-elf-gcc
COLLECT_LTO_WRAPPER=/media/liww/windows/workspace/simulator/nuclei-sdk/nuclei_riscv_newlibc_p
rebuilt_linux64_2020.08/gcc/bin/../libexec/gcc/riscv-nuclei-elf/9.2.0/lto-wrapper
Target: riscv-nuclei-elf
Configured with: /home/xl_ci/toolchain/riscv-gnu-toolchain/riscv-gcc/configure --target=riscv
-nuclei-elf --prefix=/home/xl_ci/toolchain/nuclei_gnu/linux64/newlibc/2020.08/gcc --disable-s
hared --disable-threads --enable-languages=c,c++ --with-system-zlib --enable-tls --with-newli
b --with-sysroot=/home/xl_ci/toolchain/nuclei_gnu/linux64/newlibc/2020.08/gcc/riscv-nuclei-el
f --with-native-system-header-dir=/include --disable-libmudflap --disable-libssp --disable-li
bquadmath --disable-libgomp --disable-nls --src=/home/xl_ci/toolchain/riscv-gnu-toolchain/ris
cv-gcc --enable-checking=yes --enable-multilib --with-abi=ilp32d --with-arch=rv32gc --with-tu
ne=rocket 'CFLAGS_FOR_TARGET=-Os  -mcmodel=medany' 'CXXFLAGS_FOR_TARGET=-Os  -mcmodel=medany'
Thread model: single
gcc version 9.2.0 (GCC)
```

# 环境准备—riscv-tools

## riscv-tools

源码获取：[https://github.com/riscv/riscv-tools.git](https://github.com/riscv/riscv-tools.git)

```
git clone https://github.com/riscv/riscv-tools.git
cd riscv-tools
git submodule update --init –recursive
```

编译过程：

```
export RISCV=/path/to/install/riscv/toolchain
./build.sh
```

- Spike：模拟器
- riscv-tests：RISC-V ISA级的测试集
- riscv-opcodes：所有模拟器中可执行的RISC-V指令操作码的枚举
- riscv-pk：包含了一个linux和其它相似系统的boot loader——bbl, 以及一个能够处理系统调用的代理内核pk
- riscv-openocd：提供on-chip programming 和 debug支持

## 环境准备—普通应用程序

### 普通应用程序

hello.c

```c
#include <stdio.h>

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

RISC-V GNU Toolchain 通用编译/验证

```
$riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 hello.c -o hello32
$file hello32
hello32: ELF 32-bit LSB executable, UCB RISC-V, version 1 (SYSV),
statically linked, with debug_info, not stripped
$riscv64-unknown-elf-gcc hello.c -o hello64
$file hello64
hello64: ELF 64-bit LSB executable, UCB RISC-V, version 1 (SYSV),
statically linked, not stripped
```

Nuclei RISC-V Toolchain 通用编译/验证

```
$riscv-nuclei-elf-gcc  hello.c -o hello32
$file hello32
hello32: ELF 32-bit LSB executable, UCB RISC-V, version 1 (SYSV),
statically linked, not stripped
$riscv-nuclei-elf-gcc -march=rv64i -mabi=lp64 hello.c -o hello64
$file hello64
hello64: ELF 64-bit LSB executable, UCB RISC-V, version 1 (SYSV),
statically linked, with debug_info, not stripped
```

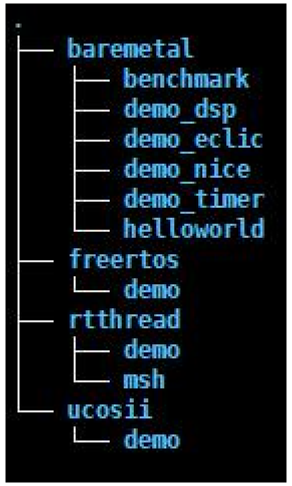
# 环境准备—Nuclei SDK

## Nuclei SDK

源码: https://github.com/Nuclei-Software/nuclei-sdk.git

Build Test For MCU N600 NX600

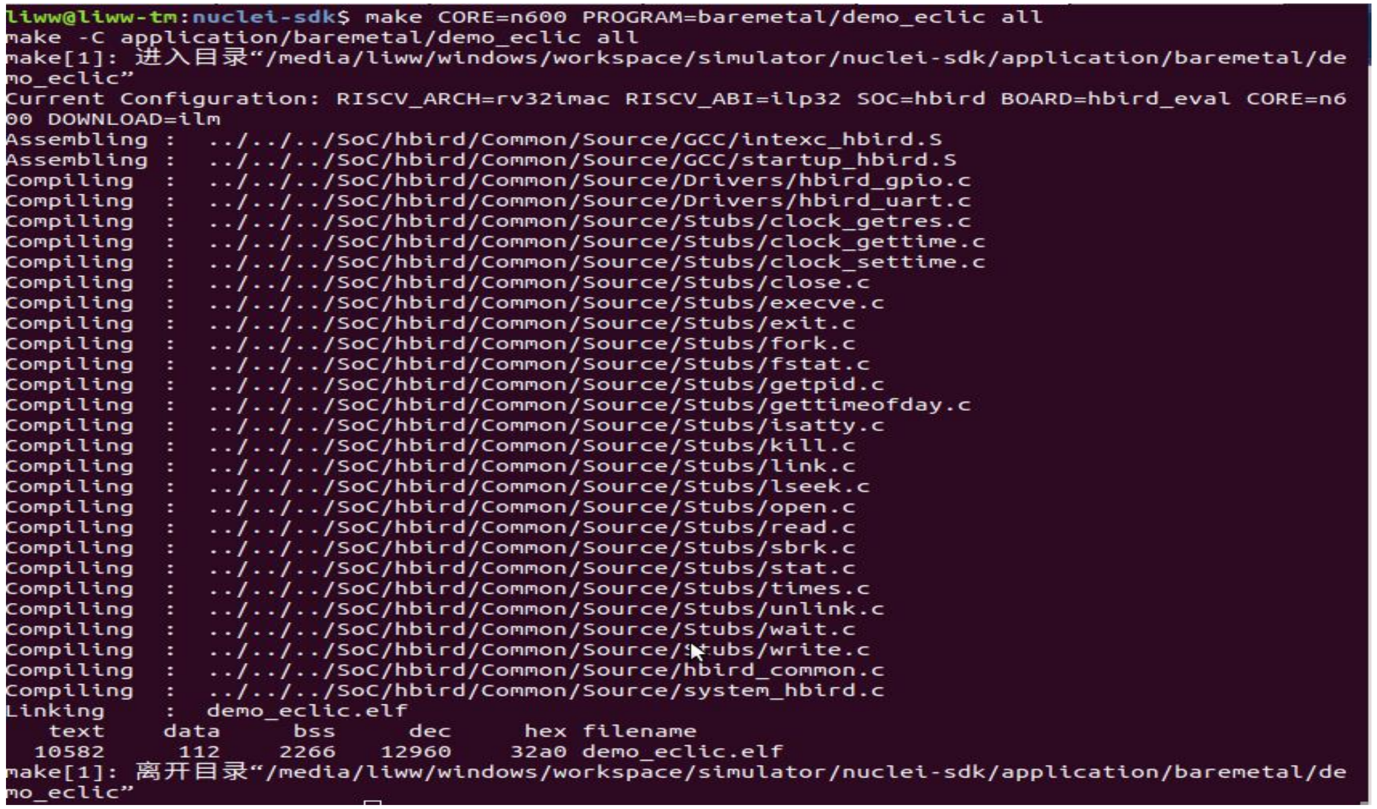
nuclei-sdk/application下



```
.
├── baremetal
│   ├── benchmark
│   ├── demo_dsp
│   ├── demo_eclic
│   ├── demo_nice
│   ├── demo_timer
│   └── helloworld
├── freertos
│   └── demo
├── rtthread
│   ├── demo
│   └── msh
└── ucosii
    └── demo
```

性能测试
~~DSP测试~~
eclic中断测试
自定义指令测试
Timer中断测试
Helloword
RTOS

编译示例:
$make CORE=n600 PROGRAM=baremetal/demo_eclic all

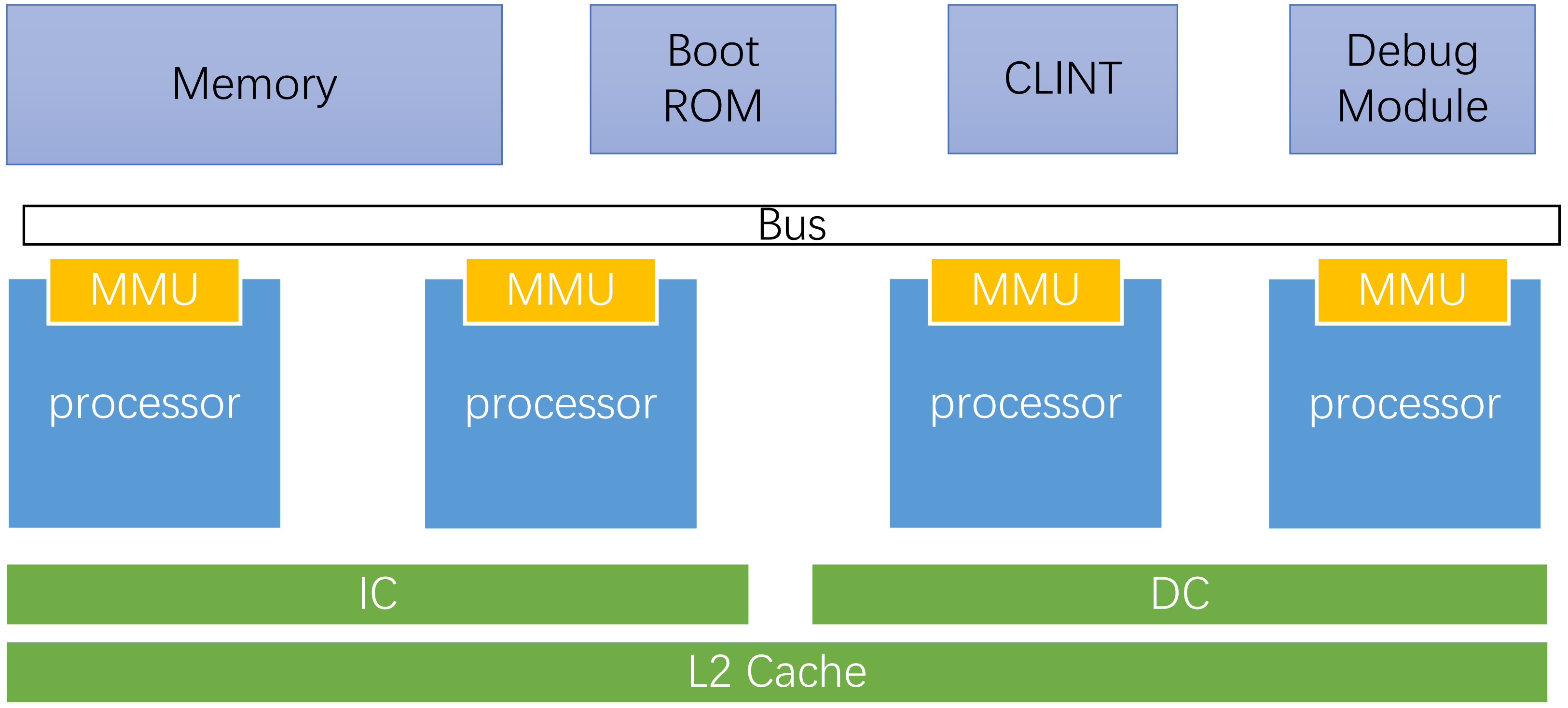


```
liww@liww-tm:nuclei-sdk$ make CORE=n600 PROGRAM=baremetal/demo_eclic all
make -C application/baremetal/demo_eclic all
make[1]: 进入目录“/media/liww/windows/workspace/simulator/nuclei-sdk/application/baremetal/de
mo_eclic”
Current Configuration: RISCV_ARCH=rv32imac RISCV_ABI=ilp32 SOC=hbird BOARD=hbird_eval CORE=n6
00 DOWNLOAD=ilm
Assembling :   ../../../SoC/hbird/Common/Source/GCC/intexc_hbird.S
Assembling :   ../../../SoC/hbird/Common/Source/GCC/startup_hbird.S
Compiling  :   ../../../SoC/hbird/Common/Source/Drivers/hbird_gpio.c
Compiling  :   ../../../SoC/hbird/Common/Source/Drivers/hbird_uart.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/clock_getres.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/clock_gettime.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/clock_settime.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/close.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/execve.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/exit.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/fork.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/fstat.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/getpid.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/gettimeofday.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/isatty.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/kill.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/link.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/lseek.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/open.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/read.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/sbrk.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/stat.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/times.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/unlink.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/wait.c
Compiling  :   ../../../SoC/hbird/Common/Source/Stubs/write.c
Compiling  :   ../../../SoC/hbird/Common/Source/hbird_common.c
Compiling  :   ../../../SoC/hbird/Common/Source/system_hbird.c
Linking    :   demo_eclic.elf
   text    data     bss     dec     hex filename
  10582     112    2266   12960    32a0 demo_eclic.elf
make[1]: 离开目录“/media/liww/windows/workspace/simulator/nuclei-sdk/application/baremetal/de
mo_eclic”
```

参考：https://doc.nucleisys.com/nuclei_sdk/overview.html

## Spike简介

### Spike是针对RISC-V的轻量级指令集模拟器



## Spike简介—ISA支持

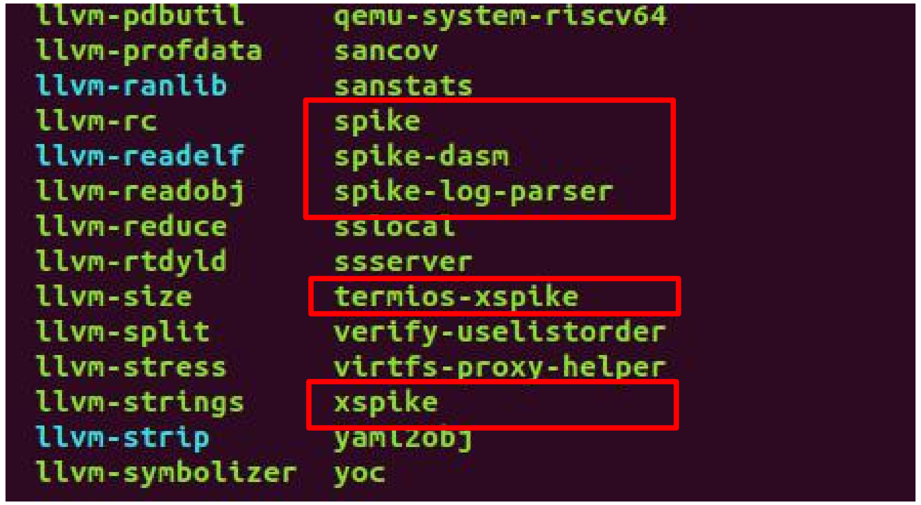
Spike supports the following RISC-V ISA features:
- RV32I and RV64I base ISAs, v2.1
- Zifencei extension, v2.0
- Zicsr extension, v2.0
- M extension, v2.0
- A extension, v2.1
- F extension, v2.2
- D extension, v2.2
- Q extension, v2.2
- C extension, v2.0
- B extension, v0.92
- K extension, v0.8.1 ([Scalar Cryptography](https://github.com/riscv/riscv-crypto))
- V extension, v0.10, w/ Zvlsseg/Zvamo (_requires a 64-bit host_)
- Bi-endianness
- Conformance to both RVWMO and RVTSO (Spike is sequentially consistent)
- Machine, Supervisor, and User modes, v1.11
- Hypervisor extension, v0.6.1
- Svnapot extension, v0.1
- Debug v0.14

| 支持的扩展 | 支持版本 | 最新版本 |
|---|---|---|
| RV32I/RV64I | v2.1 | |
| Zifencei | v2.0 | |
| Zicsr | v2.0 | |
| M | v2.0 | |
| A | v2.1 | |
| F/D/Q | v2.2 | |
| C | v2.0 | |
| B | v0.92 | v0.93 |
| K(scalar crypto) | v0.8.1 | v0.90 |
| V | v0.10 | |
| P | v0.52/v0.92 | v0.93 |
| Hypervisor | v0.6.1 | |
| Svnapot(Zsn) | v0.1 | |
| Debug | v0.14 | |
| M/S/U modes | v1.11 | v1.12 |

## Spike编译安装

### 编译安装

```
$ apt-get install device-tree-compiler
$ mkdir build
$ cd build
$ ../configure --prefix=$RISCV
$ make
$ [sudo] make install
```



```
llvm-pdbutil       qemu-system-riscv64
llvm-profdata      sancov
llvm-ranlib        sanstats
llvm-rc            spike
llvm-readelf       spike-dasm
llvm-readobj       spike-log-parser
llvm-reduce        sslocal
llvm-rtdyld        ssserver
llvm-size          termios-xspike
llvm-split         verify-uselistorder
llvm-stress        virtfs-proxy-helper
llvm-strings       xspike
llvm-strip         yaml2obj
llvm-symbolizer    yoc
```

关键编译选项：

```
--with-isa=RV64IMAFDC   Sets the default RISC-V ISA
--with-priv=MSU         Sets the default RISC-V privilege modes supported
--with-varch=vlen:128,elen:64,slen:128
```

```
--enable-commitlog     Enable commit log generation
--enable-histogram     Enable PC histogram generation
--enable-dual-endian   Enable support for running target in either endianness
```

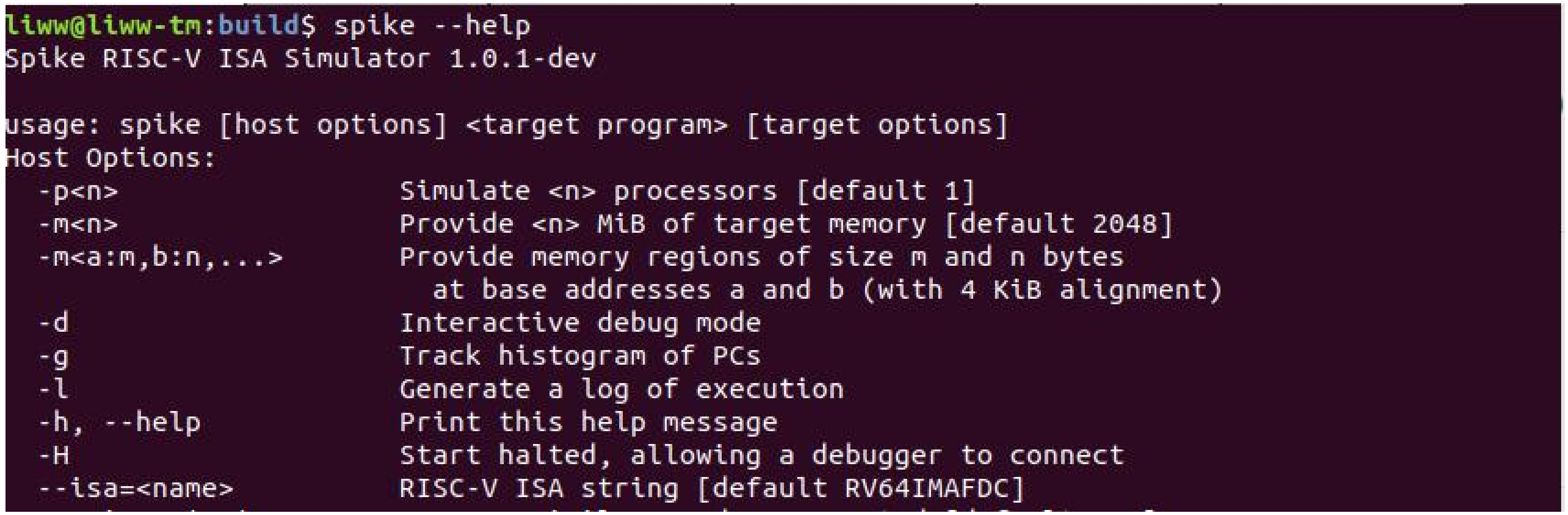
| 程序 | 功能 |
|---|---|
| spike | 模拟器程序 |
| spike-dasm | 二进制反汇编解析程序 |
| xspike | 模拟器程序（带窗口） |
| termios_xspike | 启动一个终端 |
| spike-log-parser | 解析log，提取指令名称 |

## Spike 使用

Spike命令格式:

spike [host options] <target program> [target options]

| 类型 | 命令 | 描述 |
|---|---|---|
| 帮助 | -h， --help | Print this help message |

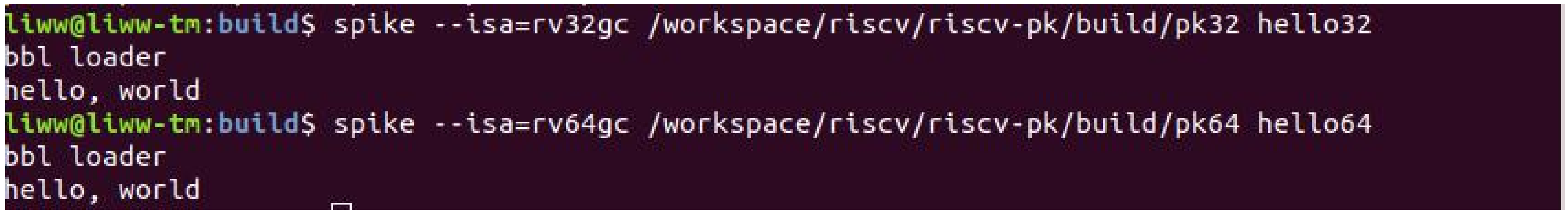


```
liww@liww-tm:build$ spike --help
Spike RISC-V ISA Simulator 1.0.1-dev

usage: spike [host options] <target program> [target options]
Host Options:
  -p<n>                   Simulate <n> processors [default 1]
  -m<n>                   Provide <n> MiB of target memory [default 2048]
  -m<a:m,b:n,...>         Provide memory regions of size m and n bytes
                            at base addresses a and b (with 4 KiB alignment)
  -d                      Interactive debug mode
  -g                      Track histogram of PCs
  -l                      Generate a log of execution
  -h, --help              Print this help message
  -H                      Start halted, allowing a debugger to connect
  --isa=<name>            RISC-V ISA string [default RV64IMAFDC]
```

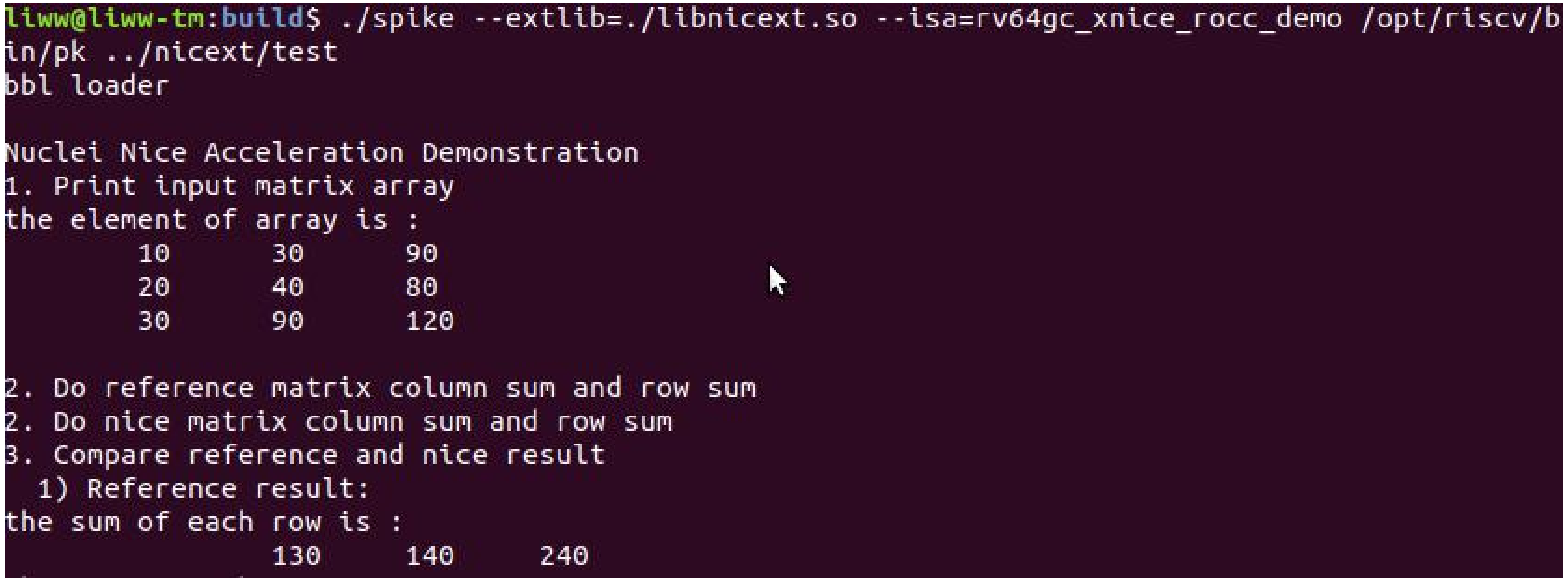
## Spike 使用—配置选项

| 类型 | 命令 | 描述 |
|---|---|---|
| 配置选项 | -p<n><br>--isa=<name><br>--pc=<address><br>-m<n><br>-m<a:m,b:n,...><br><br>-varch=<name><br><br>--ic/id/l2=<S>:<W>:<B> | Simulate <n> processors [default 1]<br>RISC-V ISA string [default RV64IMAFDC]<br>Override ELF entry point<br>Provide <n> MiB of target memory [default 2048]<br>Provide memory regions of size m and n bytes at base addresses a and b (with 4 KiB alignment)<br>RISC-V Vector uArch String [default vlen:128,elen:64,slen:128]<br>Instantiate a cache model with S sets, W ways，B-byte blocks（B和S需要是2的次方） |


```
liww@liww-tm:build$ spike --isa=rv32gc /workspace/riscv/riscv-pk/build/pk32 hello32
bbl loader
hello, world
liww@liww-tm:build$ spike --isa=rv64gc /workspace/riscv/riscv-pk/build/pk64 hello64
bbl loader
hello, world
```

## Spike 使用—扩展选项

| 类型 | 命令 | 描述 |
|---|---|---|
| 扩展配置选项 | --extension=<name><br>--extlib=<name><br>--device=<P,B,A> | Specify RoCC Extension<br>Shared library to load<br>Attach MMIO plugin device from an --extlib library |



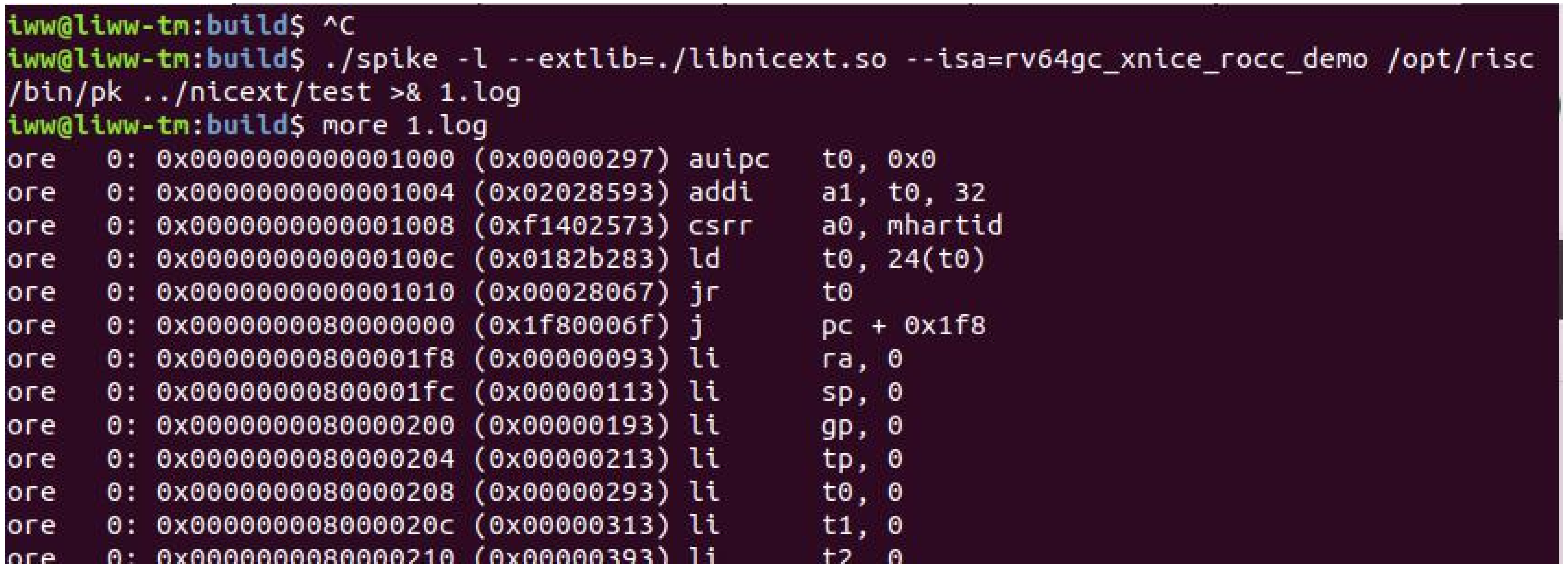
```
liww@liww-tm:build$ ./spike --extlib=./libnicext.so --isa=rv64gc_xnice_rocc_demo /opt/riscv/b
in/pk ../nicext/test
bbl loader

Nuclei Nice Acceleration Demonstration
1. Print input matrix array
the element of array is :
        10      30      90
        20      40      80
        30      90      120

2. Do reference matrix column sum and row sum
2. Do nice matrix column sum and row sum
3. Compare reference and nice result
  1) Reference result:
the sum of each row is :
                130     140     240
```

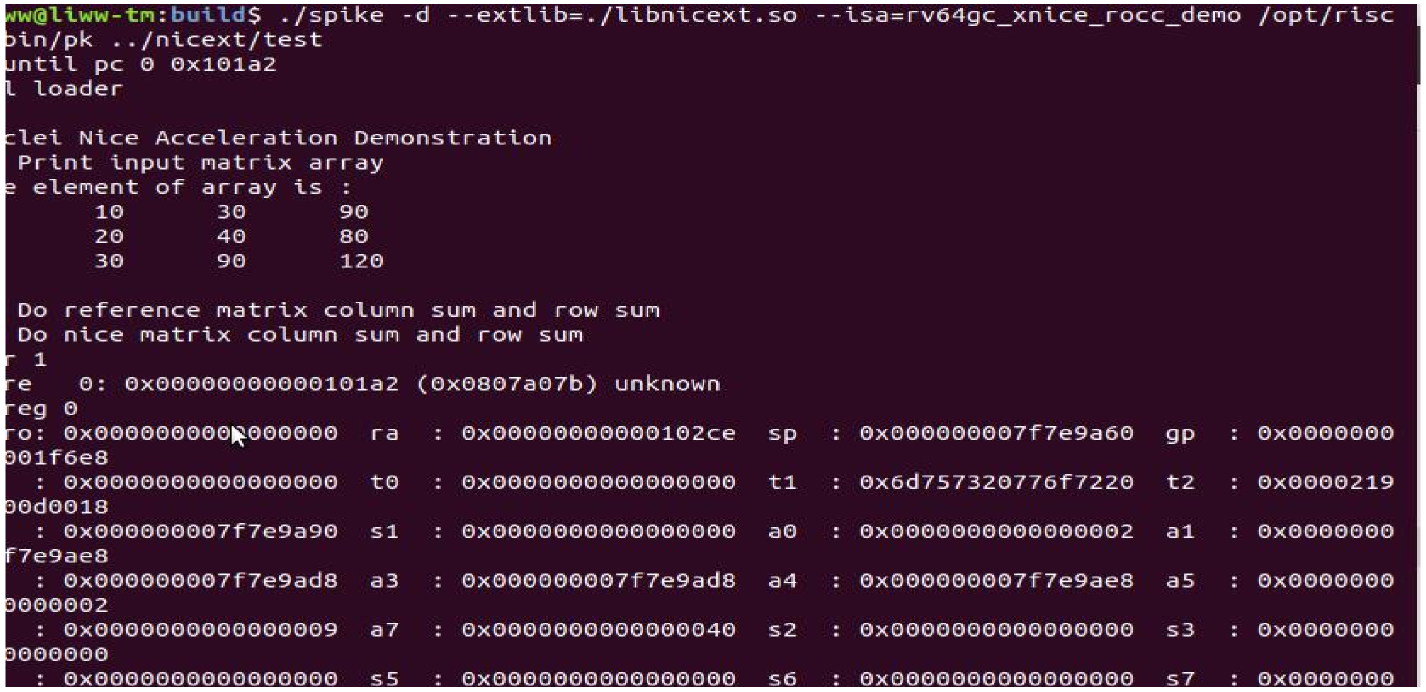
## Spike 使用—Log选项

| 类型 | 命令 | 描述 |
|---|---|---|
| Log选项 | -l<br>--log-cache-miss<br>--dump-dts<br>-g | Generate a log of execution<br>Generate a log of cache miss<br>Print device tree string and exit<br>Track histogram of PCs |


```
iww@liww-tm:build$ ^C
iww@liww-tm:build$ ./spike -l --extlib=./libnicext.so --isa=rv64gc_xnice_rocc_demo /opt/risc
/bin/pk ../nicext/test >& 1.log
iww@liww-tm:build$ more 1.log
ore   0: 0x0000000000001000 (0x00000297) auipc   t0, 0x0
ore   0: 0x0000000000001004 (0x02028593) addi    a1, t0, 32
ore   0: 0x0000000000001008 (0xf1402573) csrr    a0, mhartid
ore   0: 0x000000000000100c (0x0182b283) ld      t0, 24(t0)
ore   0: 0x0000000000001010 (0x00028067) jr      t0
ore   0: 0x0000000080000000 (0x1f80006f) j       pc + 0x1f8
ore   0: 0x00000000800001f8 (0x00000093) li      ra, 0
ore   0: 0x00000000800001fc (0x00000113) li      sp, 0
ore   0: 0x0000000080000200 (0x00000193) li      gp, 0
ore   0: 0x0000000080000204 (0x00000213) li      tp, 0
ore   0: 0x0000000080000208 (0x00000293) li      t0, 0
ore   0: 0x000000008000020c (0x00000313) li      t1, 0
ore   0: 0x0000000080000210 (0x00000393) li      t2, 0
```

## Spike 使用—调试选项

| 类型 | 命令 | 描述 |
|---|---|---|
| 交互调试 | -d | Interactive debug mode |
| OpenOCD + gdb调试 | --rbb-port=<port> | Listen on <port> for remote bitbang connection |



```
ww@liww-tm:build$ ./spike -d --extlib=./libnicext.so --isa=rv64gc_xnice_rocc_demo /opt/risc
bin/pk ../nicext/test
until pc 0 0x101a2
l loader

clei Nice Acceleration Demonstration
 Print input matrix array
e element of array is :
       10       30       90
       20       40       80
       30       90       120

 Do reference matrix column sum and row sum
 Do nice matrix column sum and row sum
r 1
re   0: 0x00000000000101a2 (0x0807a07b) unknown
reg 0
ro: 0x0000000000000000  ra  : 0x00000000000102ce  sp  : 0x000000007f7e9a60  gp  : 0x0000000
001f6e8
  : 0x0000000000000000  t0  : 0x0000000000000000  t1  : 0x6d757320776f7220  t2  : 0x0000219
00d0018
  : 0x000000007f7e9a90  s1  : 0x0000000000000000  a0  : 0x0000000000000002  a1  : 0x0000000
f7e9ae8
  : 0x000000007f7e9ad8  a3  : 0x000000007f7e9ad8  a4  : 0x000000007f7e9ae8  a5  : 0x0000000
0000002
  : 0x0000000000000009  a7  : 0x0000000000000040  s2  : 0x0000000000000000  s3  : 0x0000000
0000000
  : 0x0000000000000000  s5  : 0x0000000000000000  s6  : 0x0000000000000000  s7  : 0x0000000
```

具体调试过程参考：
https://github.com/riscv/riscv-isa-sim/blob/master/README.md

## QEMU介绍

### 简介

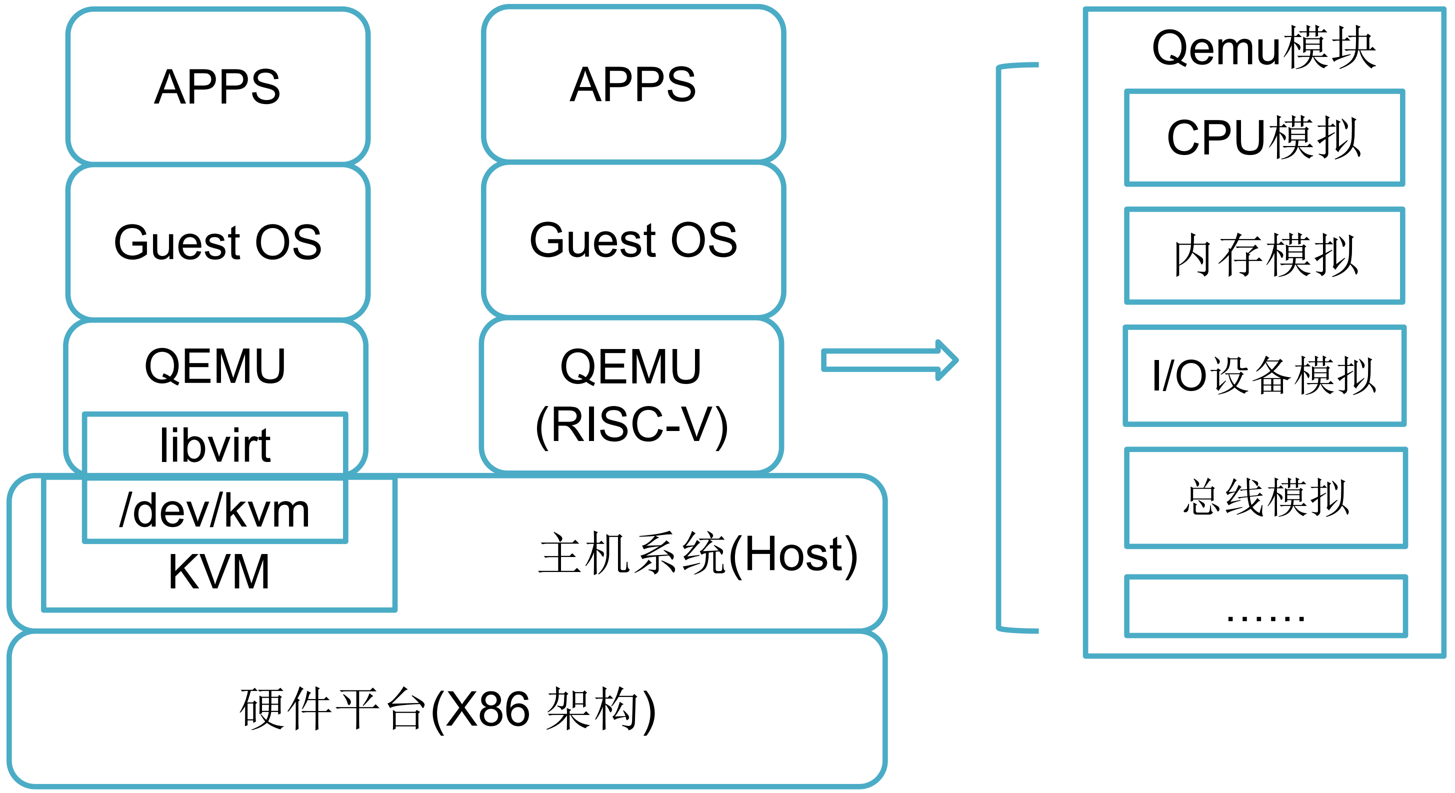
QEMU是由Fabrice Bellard开发的一款开源的模拟器，它通过动态二进制翻译的方式来模拟各种架构的CPU，并提供一系列的硬件模型来模拟各种硬件，模拟真实的硬件平台提供给操作系统和应用进行交互。

官网:https://www.qemu.org/

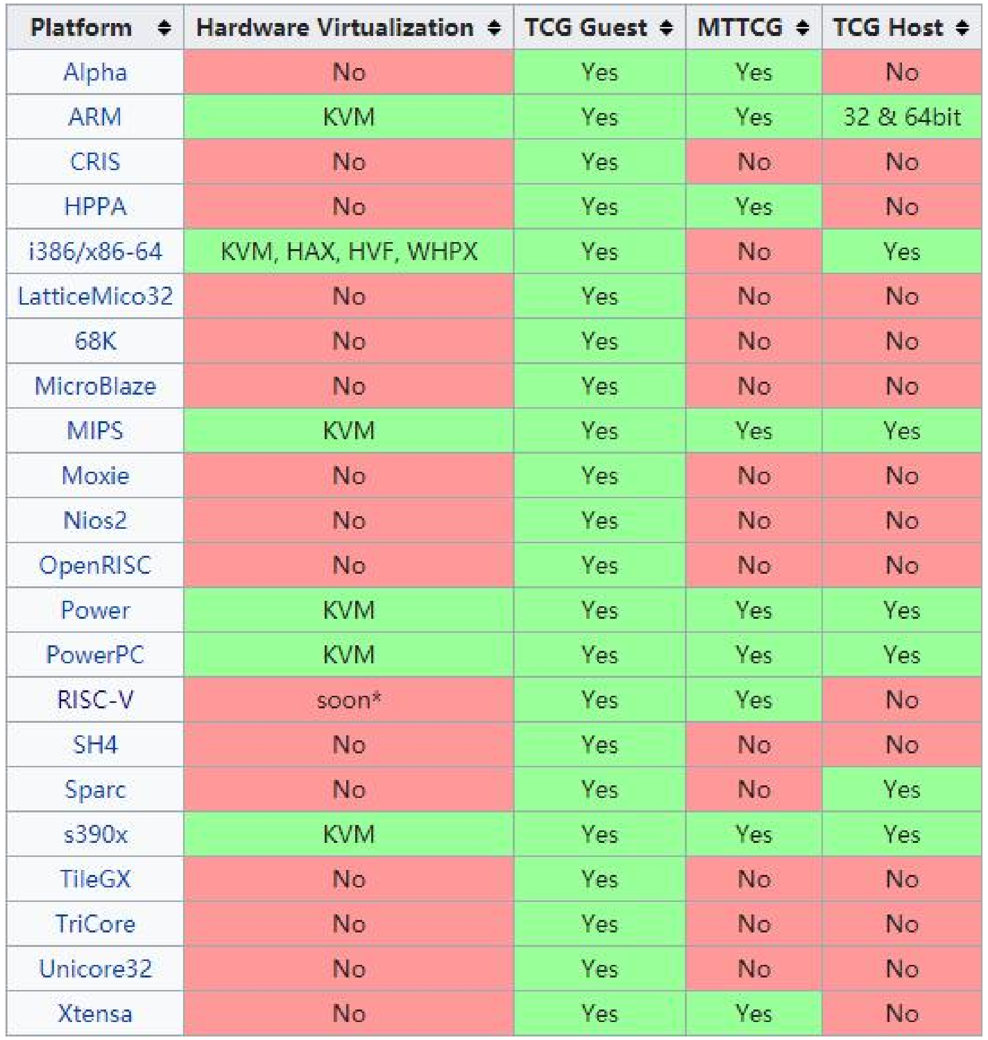
Fabrice Bellard:https://bellard.org/

# QEMU介绍—主要架构

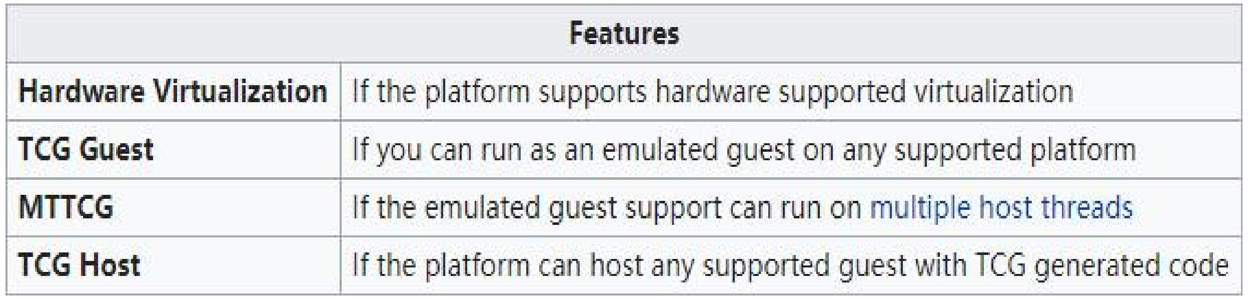
## 主要架构



## QEMU介绍—平台支持



| Platform | Hardware Virtualization | TCG Guest | MTTCG | TCG Host |
|---|---|---|---|---|
| Alpha | No | Yes | Yes | No |
| ARM | KVM | Yes | Yes | 32 & 64bit |
| CRIS | No | Yes | No | No |
| HPPA | No | Yes | Yes | No |
| i386/x86-64 | KVM, HAX, HVF, WHPX | Yes | No | Yes |
| LatticeMico32 | No | Yes | No | No |
| 68K | No | Yes | No | No |
| MicroBlaze | No | Yes | No | No |
| MIPS | KVM | Yes | Yes | Yes |
| Moxie | No | Yes | No | No |
| Nios2 | No | Yes | No | No |
| OpenRISC | No | Yes | No | No |
| Power | KVM | Yes | Yes | Yes |
| PowerPC | KVM | Yes | Yes | Yes |
| RISC-V | soon* | Yes | Yes | No |
| SH4 | No | Yes | No | No |
| Sparc | No | Yes | No | Yes |
| s390x | KVM | Yes | Yes | Yes |
| TileGX | No | Yes | No | No |
| TriCore | No | Yes | No | No |
| Unicore32 | No | Yes | No | No |
| Xtensa | No | Yes | Yes | No |



| Features | |
|---|---|
| **Hardware Virtualization** | If the platform supports hardware supported virtualization |
| **TCG Guest** | If you can run as an emulated guest on any supported platform |
| **MTTCG** | If the emulated guest support can run on multiple host threads |
| **TCG Host** | If the platform can host any supported guest with TCG generated code |

参考：https://wiki.qemu.org/Documentation/Platforms

## QEMU基本使用—编译构建（1）

### 编译构建

源码获取

```
git clone https://git.qemu.org/git/qemu.git
git submodule init
git submodule update –recursive
```

配置编译构建

```
32bit qemu for RISC-V
./configure --target-list=riscv32-softmmu,riscv32-linux-user --prefix=~/tools/qemu
$make && make install
64bit qemu for RISC-V
$./configure --target-list=riscv64-softmmu,riscv64-linux-user --prefix=~/tools/qemu
$make && make install
```

$git log 本课程开始commit

```
commit 9e2e9fe3df9f539f8b6941ceb96d25355fdae47e (HEAD ->
course)
Author: Peter Maydell <peter.maydell@linaro.org>
Date:   Wed Mar 24 19:50:49 2021 +0000

    Update version for v6.0.0-rc0 release

    Signed-off-by: Peter Maydell <peter.maydell@linaro.org>
```

qemu-system-riscv32/64：系统态模拟器程序

qemu-riscv32/64：用户态模拟器程序

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

## QEMU基本使用—编译构建（2）

### 编译构建— Configure关键选项

```
Standard options:
--prefix=PREFIX        install in PREFIX [/usr/local]
--target-list=LIST     set target list (default: build all non-deprecated)
                       riscv32-softmmu riscv64-softmmu
                       riscv32-linux-user riscv64-linux-user
Advanced options (experts only):
--extra-cflags=CFLAGS    append extra C compiler flags QEMU_CFLAGS
--extra-cxxflags=CXXFLAGS append extra C++ compiler flags QEMU_CXXFLAGS
--extra-ldflags=LDFLAGS  append extra linker flags LDFLAGS
--enable-plugins

Optional features, enabled with --enable-FEATURE
sdl          SDL UI
gtk          gtk UI
virtfs       VirtFS
kvm          KVM acceleration support

更多 configure help
```

中国科学院软件研究所 Institute of Software Chinese Academy of Sciences　智能软件研究中心 Intelligent Software Research Center

## QEMU基本使用—RISC-V相关目录和Meson

### 编译构建— RISC-V相关目录

cpu相关(target/riscv)

```
arch_dump.c   cpu-param.h   helper.h         insn32.decode  meson.build  trace-events
cpu_bits.h    cpu_user.h   insn16-32.decode  insn_trans     monitor.c    trace.h
cpu.c         csr.c        insn16-64.decode  instmap.h      op_helper.c  translate.c
cpu.h         fpu_helper.c insn16.decode     internals.h    pmp.c        vector_helper.c
cpu_helper.c  gdbstub.c    insn32-64.decode  machine.c      pmp.h
```

Board相关hw/riscv

```
boot.c   meson.build       numa.c       riscv_hart.c  sifive_u.c  virt.c
Kconfig  microchip_pfsoc.c opentitan.c  sifive_e.c    spike.c
```

设备相关hw

```
gpio  intc   mem      ssi      char  meson.build  timer    sd  i2c
isa   mips   tricore  Kconfig  misc  riscv display  rtc ……
```

### 关联:基于Meson的构建系统

```
ls default-configs/devices/riscv*
default-configs/devices/riscv32-softmmu.mak
default-configs/devices/riscv64-softmmu.mak
```

```
ls default-configs/targets/riscv*
default-configs/targets/riscv32-linux-user.mak
default-configs/targets/riscv64-linux-user.mak
default-configs/targets/riscv32-softmmu.mak
default-configs/targets/riscv64-softmmu.mak
```

编译选项开关Kconfig(以sifive spi支持为例)

```
config SIFIVE_SPI
    bool
    select SSI
```

编译文件开关meson.build

```
softmmu_ss.add(when:
'CONFIG_SIFIVE_SPI', if_true:
files('sifive_spi.c'))
```

## QEMU基本使用—QEMU模式

### QEMU模式

- 用户态仿真模拟器(User mode emulation)

  QEMU可以在当前CPU上执行其它架构的执行程序（例如：QEMU可以在x86机器上执行一个RISC-V二进制可执行程序）

- 全系统仿真模拟器(Full system emulation)

  在这种模式下，qemu完整的仿真目标硬件平台，此时，qemu就相当于一台完整的pc机，可以包括一个或多个处理器以及各种外围设备。

## QEMU基本使用 —CPU及Machine支持

用户态的CPU支持

```
$qemu-riscv32 -cpu help
any
lowrisc-ibex
rv32
sifive-e31
sifive-e34
sifive-u34
```

```
$qemu-riscv64 -cpu help
any
rv64
sifive-e51
sifive-u54
```

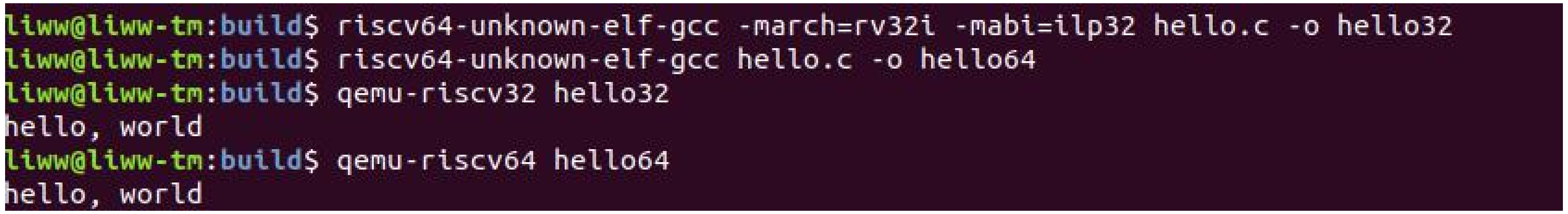
系统态的支持

```
$qemu-system-riscv64 -M ?
Supported machines are:
microchip-icicle-kit Microchip PolarFire SoC Icicle Kit
none              empty machine
sifive_e          RISC-V Board compatible with SiFive E SDK
sifive_u          RISC-V Board compatible with SiFive U SDK
spike             RISC-V Spike board (default)
virt              RISC-V VirtIO board
```

```
$qemu-system-riscv32 -M ?
Supported machines are:
none              empty machine
opentitan         RISC-V Board compatible with OpenTitan
sifive_e          RISC-V Board compatible with SiFive E SDK
sifive_u          RISC-V Board compatible with SiFive U SDK
spike             RISC-V Spike board (default)
virt              RISC-V VirtIO board
```

## QEMU基本使用 —用户态运行

### 用户态运行



```
liww@liww-tm:build$ riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 hello.c -o hello32
liww@liww-tm:build$ riscv64-unknown-elf-gcc hello.c -o hello64
liww@liww-tm:build$ qemu-riscv32 hello32
hello, world
liww@liww-tm:build$ qemu-riscv64 hello64
hello, world
```

## QEMU基本使用—系统模拟运行

### 系统模拟—运行

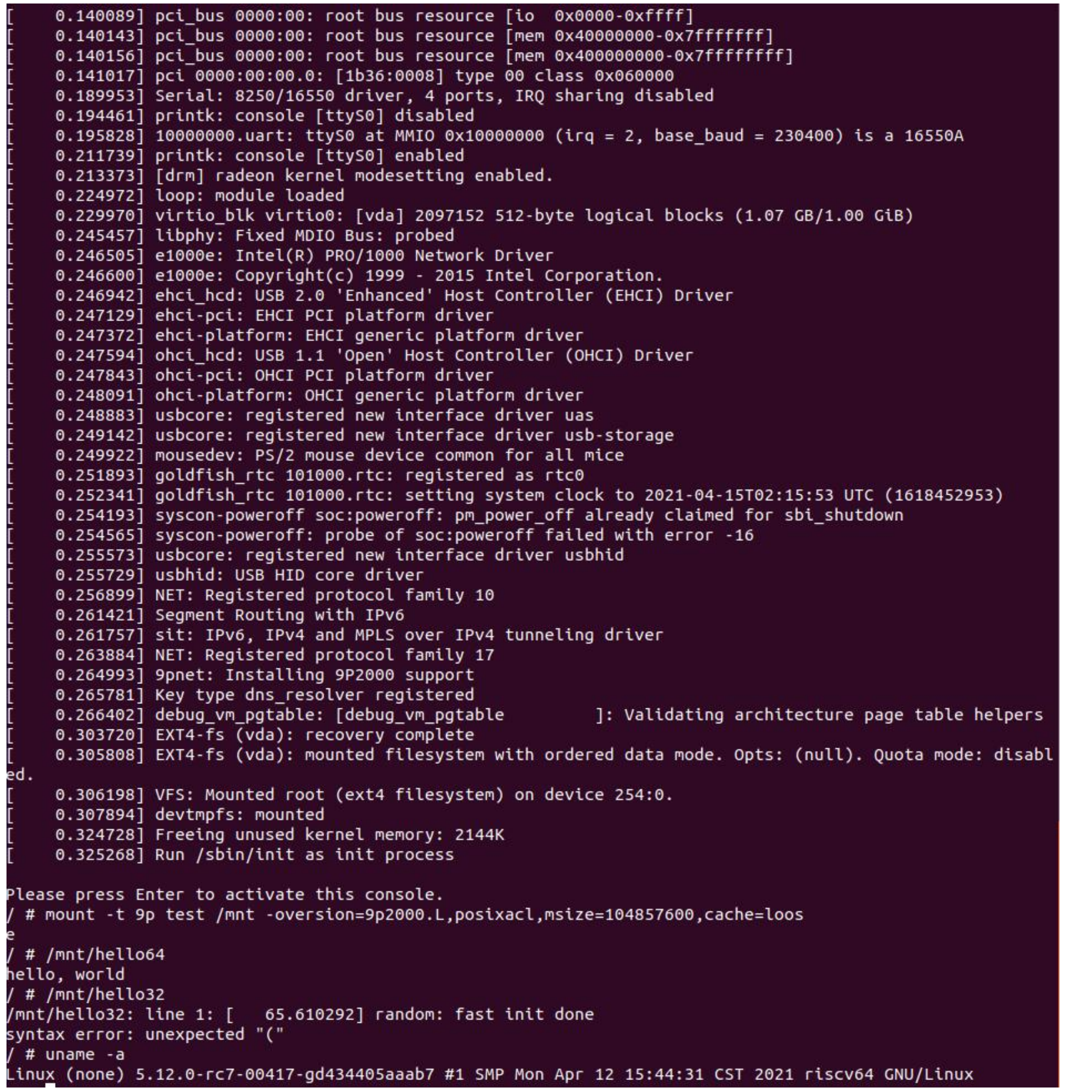
```
$qemu-system-riscv64 \
-M virt -nographic \
-kernel <path-to-linux-image>/Image \
-append "root=/dev/vda rw console=ttyS0" \
-drive file=<path-to-rootfs>/rootfs.img,format=raw,id=hd0  \
-device virtio-blk-device,drive=hd0 \
-device virtio-net-device,netdev=net0 -netdev
user,id=net0,hostfwd=tcp::10000-:22 \
-fsdev local,id=testdev,path=os/shared,security_model=none \
-device virtio-9p-device,fsdev=testdev,mount_tag=test
```

挂载9p

```
mount -t 9p test /mnt
```

或

```
mount -t 9p test /mnt -
oversion=9p2000.L,posixacl,msize=104857600,cache=loose
```



```
[    0.140089] pci_bus 0000:00: root bus resource [io  0x0000-0xffff]
[    0.140143] pci_bus 0000:00: root bus resource [mem 0x40000000-0x7fffffff]
[    0.140156] pci_bus 0000:00: root bus resource [mem 0x400000000-0x7ffffffff]
[    0.141017] pci 0000:00:00.0: [1b36:0008] type 00 class 0x060000
[    0.189953] Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
[    0.194461] printk: console [ttyS0] disabled
[    0.195828] 10000000.uart: ttyS0 at MMIO 0x10000000 (irq = 2, base_baud = 230400) is a 16550A
[    0.211739] printk: console [ttyS0] enabled
[    0.213373] [drm] radeon kernel modesetting enabled.
[    0.224972] loop: module loaded
[    0.229970] virtio_blk virtio0: [vda] 2097152 512-byte logical blocks (1.07 GB/1.00 GiB)
[    0.245457] libphy: Fixed MDIO Bus: probed
[    0.246505] e1000e: Intel(R) PRO/1000 Network Driver
[    0.246600] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    0.246942] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[    0.247129] ehci-pci: EHCI PCI platform driver
[    0.247372] ehci-platform: EHCI generic platform driver
[    0.247594] ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
[    0.247843] ohci-pci: OHCI PCI platform driver
[    0.248091] ohci-platform: OHCI generic platform driver
[    0.248883] usbcore: registered new interface driver uas
[    0.249142] usbcore: registered new interface driver usb-storage
[    0.249922] mousedev: PS/2 mouse device common for all mice
[    0.251893] goldfish_rtc 101000.rtc: registered as rtc0
[    0.252341] goldfish_rtc 101000.rtc: setting system clock to 2021-04-15T02:15:53 UTC (1618452953)
[    0.254193] syscon-poweroff soc:poweroff: pm_power_off already claimed for sbi_shutdown
[    0.254565] syscon-poweroff: probe of soc:poweroff failed with error -16
[    0.255573] usbcore: registered new interface driver usbhid
[    0.255729] usbhid: USB HID core driver
[    0.256899] NET: Registered protocol family 10
[    0.261421] Segment Routing with IPv6
[    0.261757] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[    0.263884] NET: Registered protocol family 17
[    0.264993] 9pnet: Installing 9P2000 support
[    0.265781] Key type dns_resolver registered
[    0.266402] debug_vm_pgtable: [debug_vm_pgtable         ]: Validating architecture page table helpers
[    0.303720] EXT4-fs (vda): recovery complete
[    0.305808] EXT4-fs (vda): mounted filesystem with ordered data mode. Opts: (null). Quota mode: disabl
ed.
[    0.306198] VFS: Mounted root (ext4 filesystem) on device 254:0.
[    0.307894] devtmpfs: mounted
[    0.324728] Freeing unused kernel memory: 2144K
[    0.325268] Run /sbin/init as init process

Please press Enter to activate this console.
/ # mount -t 9p test /mnt -oversion=9p2000.L,posixacl,msize=104857600,cache=loos
e
/ # /mnt/hello64
hello, world
/ # /mnt/hello32
/mnt/hello32: line 1: [   65.610292] random: fast init done
syntax error: unexpected "("
/ # uname -a
Linux (none) 5.12.0-rc7-00417-gd434405aaab7 #1 SMP Mon Apr 12 15:44:31 CST 2021 riscv64 GNU/Linux
```

## QEMU基本使用—Log选项

```
qemu-riscv32 -d  or qemu-system-riscv32 -d
Log items (comma separated):
out_asm        show generated host assembly code for each compiled TB
in_asm         show target assembly code for each compiled TB
op             show micro ops for each compiled TB
op_opt         show micro ops after optimization
op_ind         show micro ops before indirect lowering
int            show interrupts/exceptions in short format
exec           show trace before each executed TB (lots of logs)
cpu            show CPU registers before entering a TB (lots of logs)
fpu            include FPU registers in the 'cpu' logging
mmu            log MMU-related activities
pcall          x86 only: show protected mode far calls/returns/exceptions
cpu_reset      show CPU state before CPU resets
unimp          log unimplemented functionality
guest_errors   log when the guest OS does something invalid (eg accessing a
non-existent register)
page           dump pages at beginning of user mode emulation
nochain        do not chain compiled TBs so that "exec" and "cpu" show
complete traces
strace         log every user-mode syscall, its input, and its result
trace:PATTERN  enable trace events
```

注:用户态调试，裸机程序调试

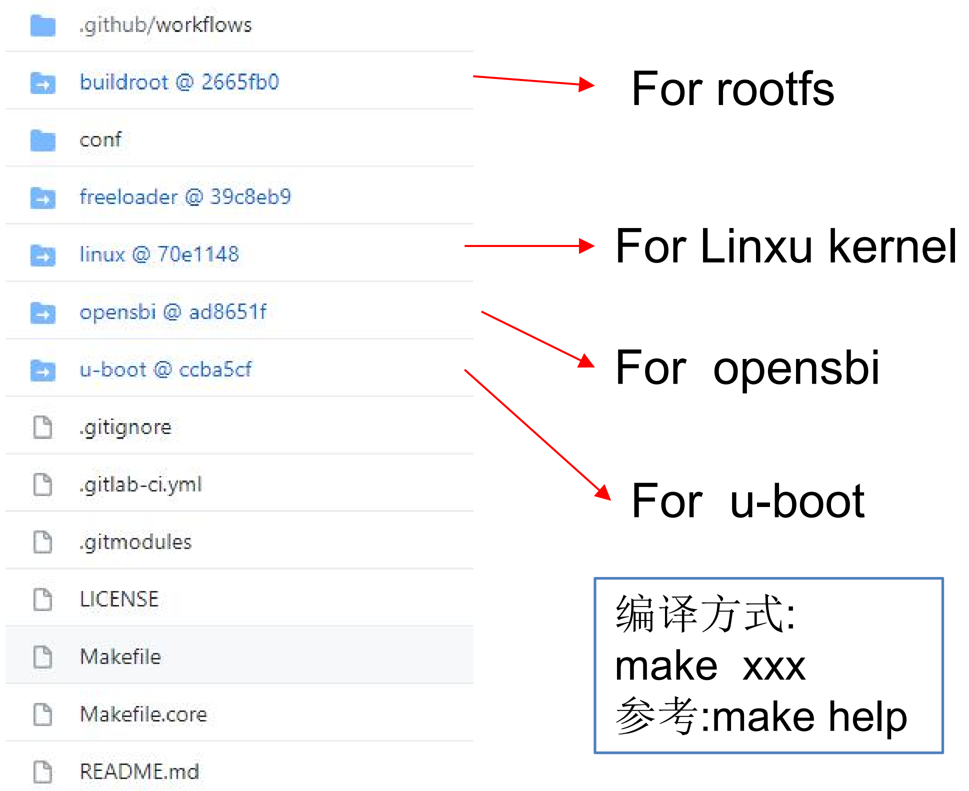
# 谢谢各位

欢迎提问、讨论、交流合作

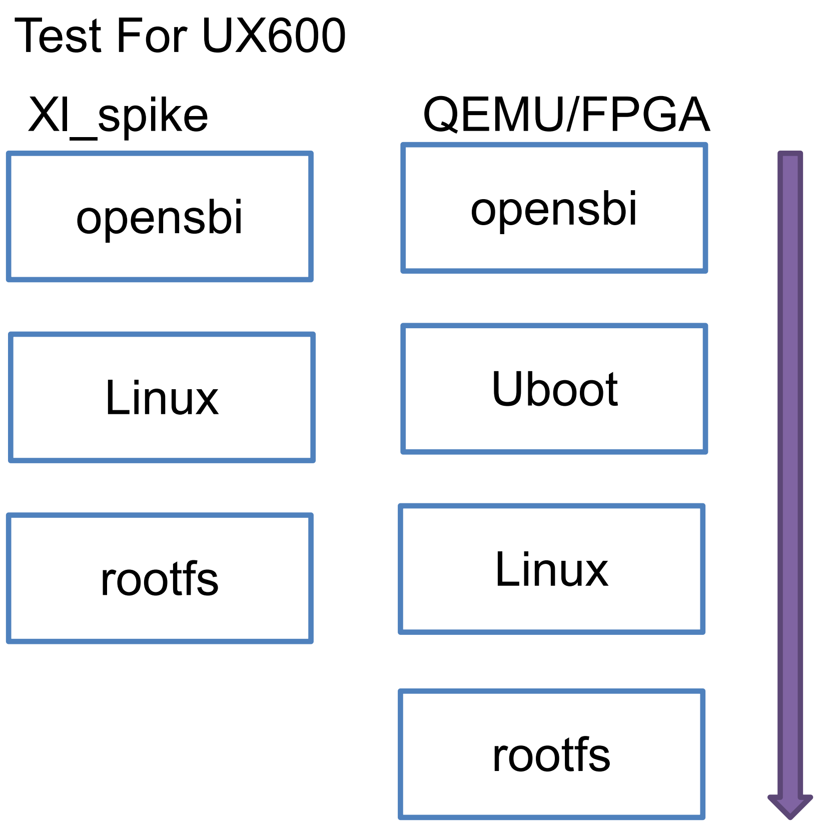
# 环境准备—NUCLEI Linux SDK

## NUCLEI Linux SDK

获取地址:
https://github.com/Nuclei-Software/nuclei-linux-sdk.git


.github/workflows
buildroot @ 2665fb0
conf
freeloader @ 39c8eb9
linux @ 70e1148
opensbi @ ad8651f
u-boot @ ccba5cf
.gitignore
.gitlab-ci.yml
.gitmodules
LICENSE
Makefile
Makefile.core
README.md

编译方式:
make xxx
参考:make help

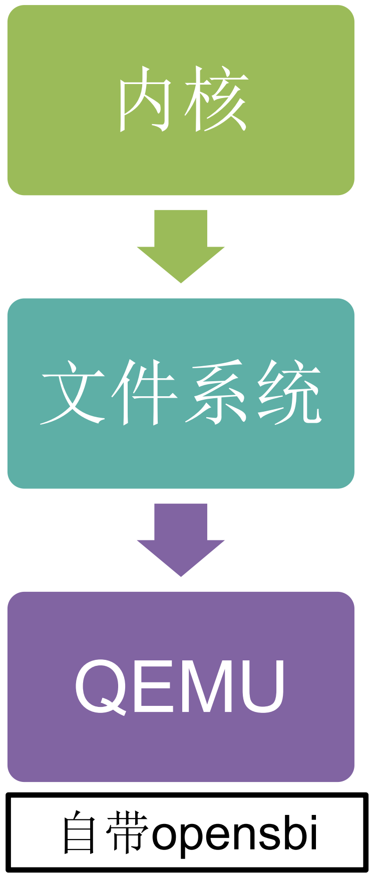


参考: https://github.com/Nuclei-Software/nuclei-linux-sdk/blob/dev_nuclei_next/README.md

opensbi模式:
https://github.com/riscv/opensbi/blob/master/docs/platform/qemu_virt.md

## Linux 构建

### 系统模拟—内核



源码下载:

```
$git clone https://github.com/torvalds/linux.git
```

```
$git log
commit d434405aaab7d0ebc516b68a8fc4100922d7f5ef
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Sun Apr 11 15:16:13 2021 -0700

    Linux 5.12-rc7
```

内核配置

```
$make ARCH=riscv CROSS_COMPILE=riscv64-unknown-linux-gnu- defconfig
```

内核编译

```
$make ARCH=riscv CROSS_COMPILE=riscv64-unknown-linux-gnu- -j $(nproc)
$ ls arch/riscv/boot/
dts  Image  Image.gz  install.sh  loader.lds.S  loader.S  Makefile
```

# 文件系统构建—busybox构建

## 系统模拟—文件系统(基于busybox)

源码下载

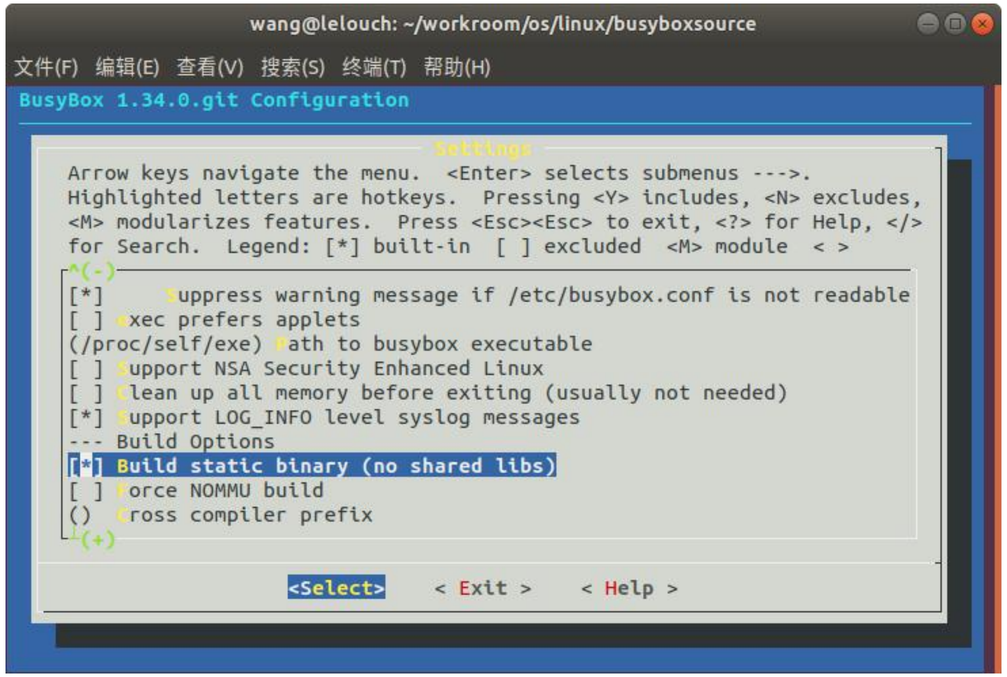
```
$git clone https://gitee.com/mirrors/busyboxsource.git
```

```
$git log
commit 9f017d9db0eb5522eb9c140a2d839461c677eb8e
Author: Ron Yorston <rmy@pobox.com>
Date:   Tue Apr 6 22:11:21 2021 +0100
```

配置编译

```
$CROSS_COMPILE=riscv64-unknown-linux-gnu- make menuconfig
$CROSS_COMPILE=riscv64-unknown-linux-gnu- make -j $(nproc)
$CROSS_COMPILE=riscv64-unknown-linux-gnu- make install
$ls _install
bin  linuxrc  sbin  usr
```

选择静态编译 “Settings“=>”Build Options“


wang@lelouch: ~/workroom/os/linux/busyboxsource
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)

```
BusyBox 1.34.0.git Configuration
                          Settings
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
 ┌^(-)──────────────────────────────────────────────────────────────────
 │[*]     Suppress warning message if /etc/busybox.conf is not readable
 │[ ] exec prefers applets
 │(/proc/self/exe) Path to busybox executable
 │[ ] Support NSA Security Enhanced Linux
 │[ ] Clean up all memory before exiting (usually not needed)
 │[*] Support LOG_INFO level syslog messages
 │--- Build Options
 │[*] Build static binary (no shared libs)
 │[ ] Force NOMMU build
 │()  Cross compiler prefix
 └┴(+)
            <Select>    < Exit >    < Help >
```

添加常见目录

```
$cd _install
$mkdir mnt proc sys dev etc etc/init.d lib
```

## 文件系统构建—rootfs文件系统创建

### 系统模拟—文件系统(基于busybox)

制作文件系统

```
$dd if=/dev/zero of=rootfs.img bs=1048576 count=128
或
$qemu-img create rootfs.img  1g
$mkfs.ext4 rootfs.img
$mkdir rootfs
$sudo mount -o loop rootfs.img  rootfs
$cd rootfs
$sudo cp -r ../busyboxsource/_install/* .
$cd etc/init.d/
$sudo touch rcS
$sudo chmod +x rcS
$sudo umount rootfs
```

```
$cat rcS
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
```

制作支持9p QEMU

```
./configure \
--enable-kvm --enable-virtfs \
--target-list=riscv32-softmmu,riscv64-softmmu,riscv32-linux-user,riscv64-linux-user \
--prefix=~/path/output
$make -j 8
$make install
```

## Fedora 系统

### 系统仿真(ferdora)

Fedora是一个 Linux 发行版，是一款由全球社区爱好者构建的面向日常应用的快速、稳定、强大的操作系统。它允许任何人自由地使用、修改和重发布，无论现在还是将来。它由一个强大的社群开发，这个社群的成员以自己的不懈努力，提供并维护自由、开放源码的软件和开放的标准。

下载镜像:

```
Fedora-Minimal-Rawhide-20200108.n.0-fw_payload-uboot-qemu-virt-smode.elf
Fedora-Minimal-Rawhide-20200108.n.0-sda.raw.xz
```

```
step 1: Download the Fedora disk images Decompress the disk image
$unxz Fedora-Minimal-Rawhide-*-sda.raw.xz
```

参考：
https://wiki.qemu.org/Documentation/Platforms/RISCV
https://wiki.debian.org/RISC-V

镜像地址 https://dl.fedoraproject.org/pub/alt/risc-v/repo/virt-builder-images/images

## QEMU启动Fedora系统

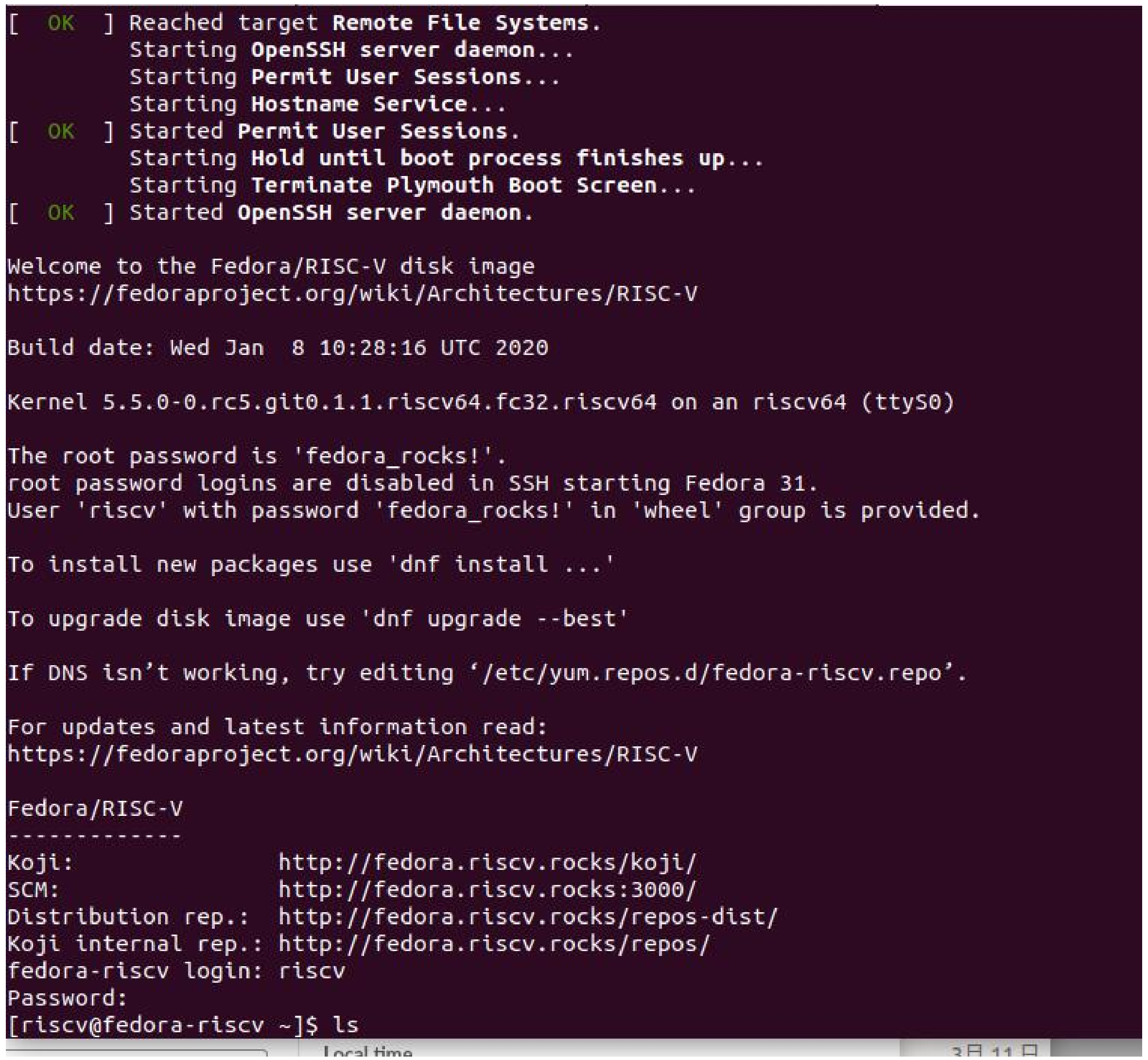
### 系统仿真(ferdora)

step 2: Boot linux using RV64GC qemu

```
qemu-system-riscv64 \
  -nographic -machine virt -smp 8 -m 2G \
  -bios none \
  -kernel fedora/Fedora-Developer-Rawhide-20200108.n.0-fw_payload-uboot-qemu-virt-smode.elf \
  -object rng-random,filename=/dev/urandom,id=rng0 \
  -device virtio-rng-device,rng=rng0 \
  -device virtio-blk-device,drive=hd0 \
  -drive file=fedora/Fedora-Developer-Rawhide-20200108.n.0-sda.raw,format=raw,id=hd0 \
  -device virtio-net-device,netdev=usernet \
  -netdev user,id=usernet,hostfwd=tcp::10000-:22
```

## QEMU启动Fedora系统截图

系统仿真(ferdora)



```
[  OK  ] Reached target Remote File Systems.
         Starting OpenSSH server daemon...
         Starting Permit User Sessions...
         Starting Hostname Service...
[  OK  ] Started Permit User Sessions.
         Starting Hold until boot process finishes up...
         Starting Terminate Plymouth Boot Screen...
[  OK  ] Started OpenSSH server daemon.

Welcome to the Fedora/RISC-V disk image
https://fedoraproject.org/wiki/Architectures/RISC-V

Build date: Wed Jan  8 10:28:16 UTC 2020

Kernel 5.5.0-0.rc5.git0.1.1.riscv64.fc32.riscv64 on an riscv64 (ttyS0)

The root password is 'fedora_rocks!'.
root password logins are disabled in SSH starting Fedora 31.
User 'riscv' with password 'fedora_rocks!' in 'wheel' group is provided.

To install new packages use 'dnf install ...'

To upgrade disk image use 'dnf upgrade --best'

If DNS isn’t working, try editing ‘/etc/yum.repos.d/fedora-riscv.repo’.

For updates and latest information read:
https://fedoraproject.org/wiki/Architectures/RISC-V

Fedora/RISC-V
-------------
Koji:               http://fedora.riscv.rocks/koji/
SCM:                http://fedora.riscv.rocks:3000/
Distribution rep.:  http://fedora.riscv.rocks/repos-dist/
Koji internal rep.: http://fedora.riscv.rocks/repos/
fedora-riscv login: riscv
Password:
[riscv@fedora-riscv ~]$ ls
```

## 下节课内容预告

### Spike 之CPU模拟

- Spike的指令集模拟原理
- 几种指令集扩展方式