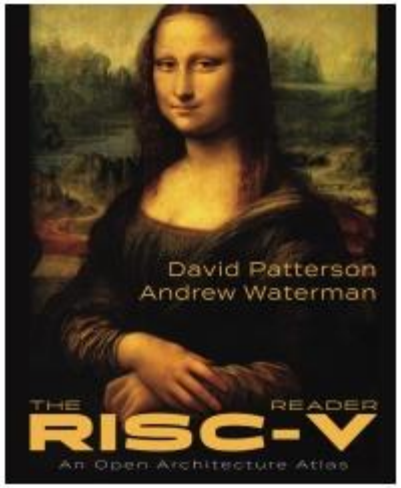
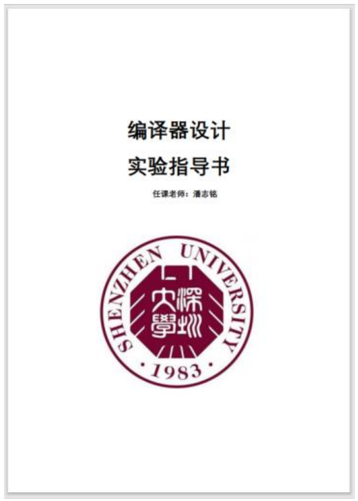
0 引论

潘志铭

0.1 课程概要

- 教材
  - *The RISC-V Reader: An Open Architecture Atlas*
  - *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*
  - 《编译器设计：实验教程书》




编译器设计
实验指导书
任课老师：潘志铭

0.1 课程概要

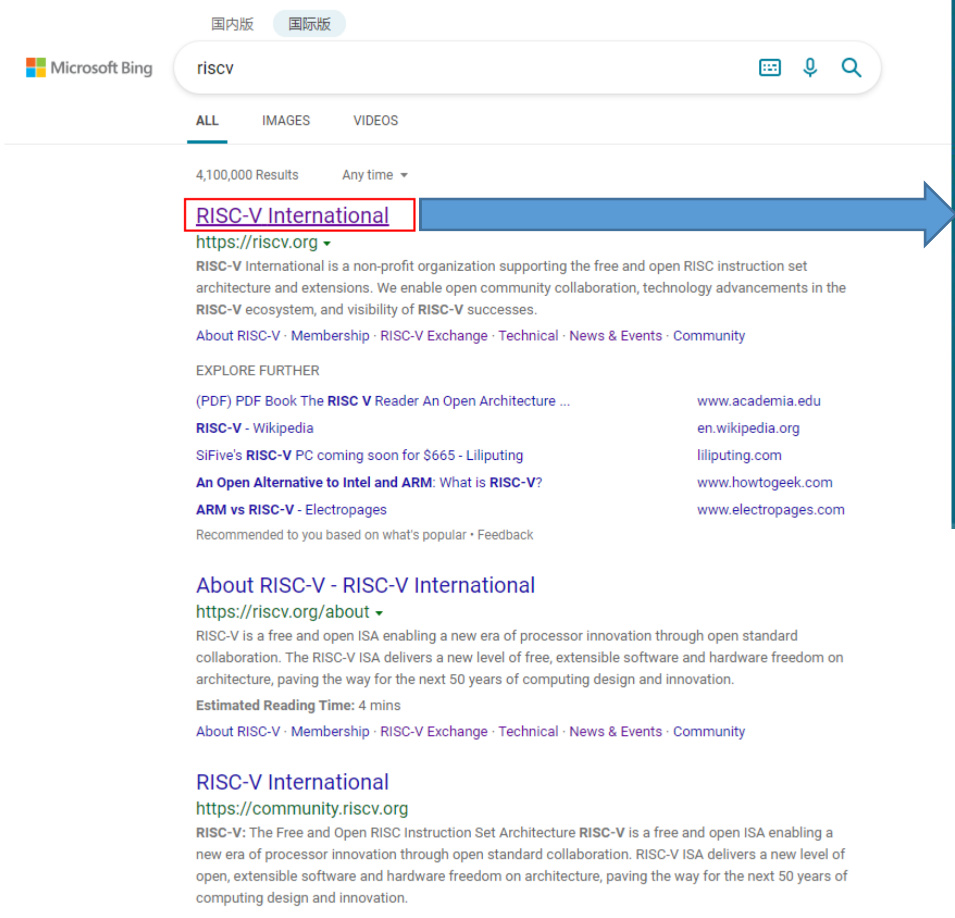
- 安排
  - 网站： https://github.com/xicongye/compiler-design
  - 学时： 36
  - 学分： 2
  - 先修课程：C语言、计算机组成原理
  - 其他： 上课要求自带笔记本电脑

- 课程结构
  - 理论部分：上课听讲，下课作业，交书面作业
  - 实践部分：按照《实验指导书》步骤完成实验报告

0.2 课程内容

- 0. 引论
- 1. RISC-V工具链介绍
- 2. RISC-V汇编语言
- 3. RISC-V工具链的使用
- 4. RISC-V工具链的开发

0.3 什么是RISC-V?



国内版 国际版

Microsoft Bing riscv

ALL IMAGES VIDEOS

4,100,000 Results Any time

RISC-V International
https://riscv.org
RISC-V International is a non-profit organization supporting the free and open RISC instruction set architecture and extensions. We enable open community collaboration, technology advancements in the RISC-V ecosystem, and visibility of RISC-V successes.
About RISC-V · Membership · RISC-V Exchange · Technical · News & Events · Community

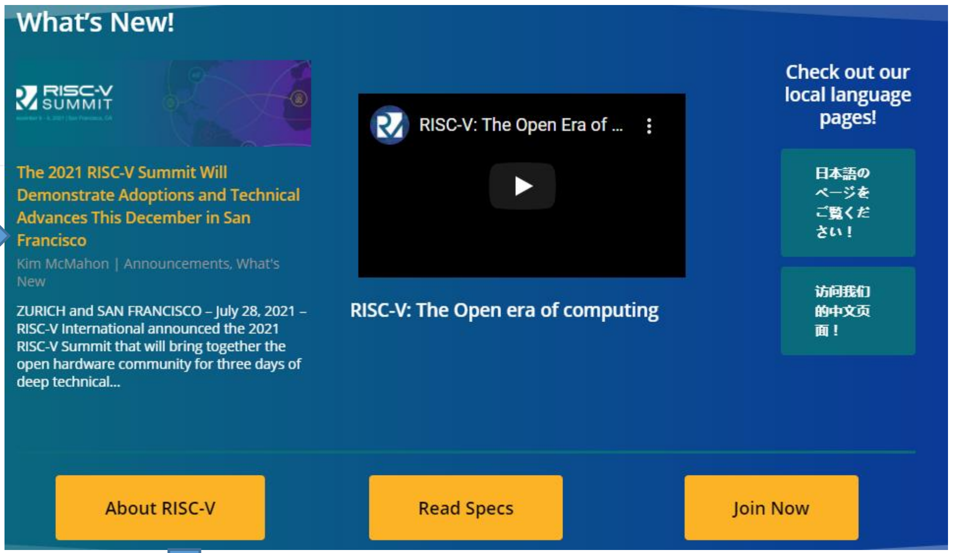
EXPLORE FURTHER

(PDF) PDF Book The RISC V Reader An Open Architecture ...	www.academia.edu
RISC-V - Wikipedia	en.wikipedia.org
SiFive's RISC-V PC coming soon for $665 - Liliputing	liliputing.com
An Open Alternative to Intel and ARM: What is RISC-V?	www.howtogeek.com
ARM vs RISC-V - Electropages	www.electropages.com

Recommended to you based on what's popular • Feedback

About RISC-V - RISC-V International
https://riscv.org/about
RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. The RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.
Estimated Reading Time: 4 mins
About RISC-V · Membership · RISC-V Exchange · Technical · News & Events · Community

RISC-V International
https://community.riscv.org
RISC-V: The Free and Open RISC Instruction Set Architecture RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. RISC-V ISA delivers a new level of free, open, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.



What's New!

RISC-V SUMMIT

The 2021 RISC-V Summit Will Demonstrate Adoptions and Technical Advances This December in San Francisco
Kim McMahon | Announcements, What's New

ZURICH and SAN FRANCISCO – July 28, 2021 – RISC-V International announced the 2021 RISC-V Summit that will bring together the open hardware community for three days of deep technical...

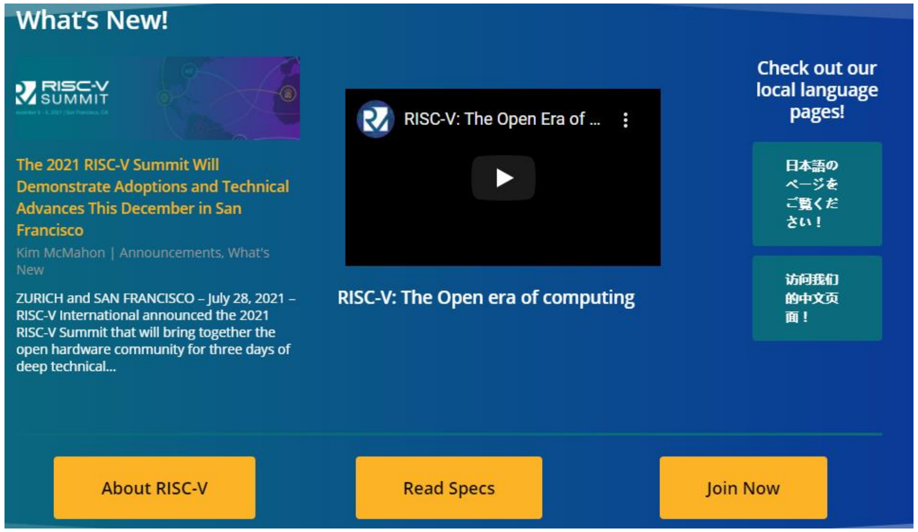
RISC-V: The Open era of computing

Check out our local language pages!

RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration.

The RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

0.4 哪里找RISC-V文档？



What's New!

The 2021 RISC-V Summit Will Demonstrate Adoptions and Technical Advances This December in San Francisco

Kim McMahon | Announcements, What's New

ZURICH and SAN FRANCISCO – July 28, 2021 – RISC-V International announced the 2021 RISC-V Summit that will bring together the open hardware community for three days of deep technical...

RISC-V: The Open Era of ...

RISC-V: The Open era of computing

Check out our local language pages!

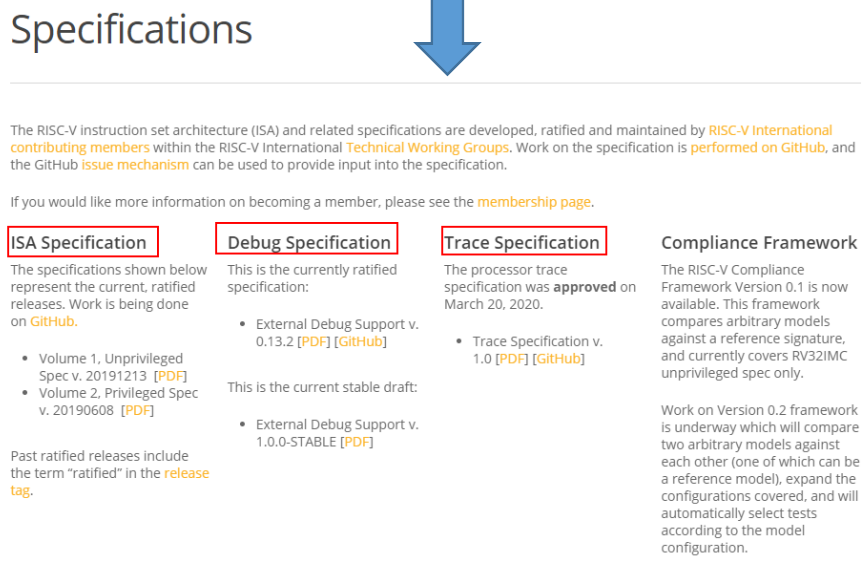
日本語のページをご覧ください！

访问我们的中文页面！

About RISC-V

Read Specs

Join Now



Specifications

The RISC-V instruction set architecture (ISA) and related specifications are developed, ratified and maintained by RISC-V International contributing members within the RISC-V International Technical Working Groups. Work on the specification is performed on GitHub, and the GitHub issue mechanism can be used to provide input into the specification.

If you would like more information on becoming a member, please see the membership page.

ISA Specification

The specifications shown below represent the current, ratified releases. Work is being done on GitHub.

- Volume 1, Unprivileged Spec v. 20191213 [PDF]
- Volume 2, Privileged Spec v. 20190608 [PDF]

Past ratified releases include the term "ratified" in the release tag.

Debug Specification

This is the currently ratified specification:

- External Debug Support v. 0.13.2 [PDF] [GitHub]

This is the current stable draft:

- External Debug Support v. 1.0.0-STABLE [PDF]

Trace Specification

The processor trace specification was **approved** on March 20, 2020.

- Trace Specification v. 1.0 [PDF] [GitHub]

Compliance Framework

The RISC-V Compliance Framework Version 0.1 is now available. This framework compares arbitrary models against a reference signature, and currently covers RV32IMC unprivileged spec only.

Work on Version 0.2 framework is underway which will compare two arbitrary models against each other (one of which can be a reference model), expand the configurations covered, and will automatically select tests according to the model configuration.

ISA Specification

- Volume 1, Unprivileged Spec
- Volume 2, Privileged Spec

*Unprivileged Spec*描述的是RISC-V用户级（user-level）架构，例如RISC-V指令编码和用法；
*Privileged Spec*描述的是RISC-V特权架构，例如机器模式（machine-level）、管理员模式（supervisor-level）；

Debug Specification

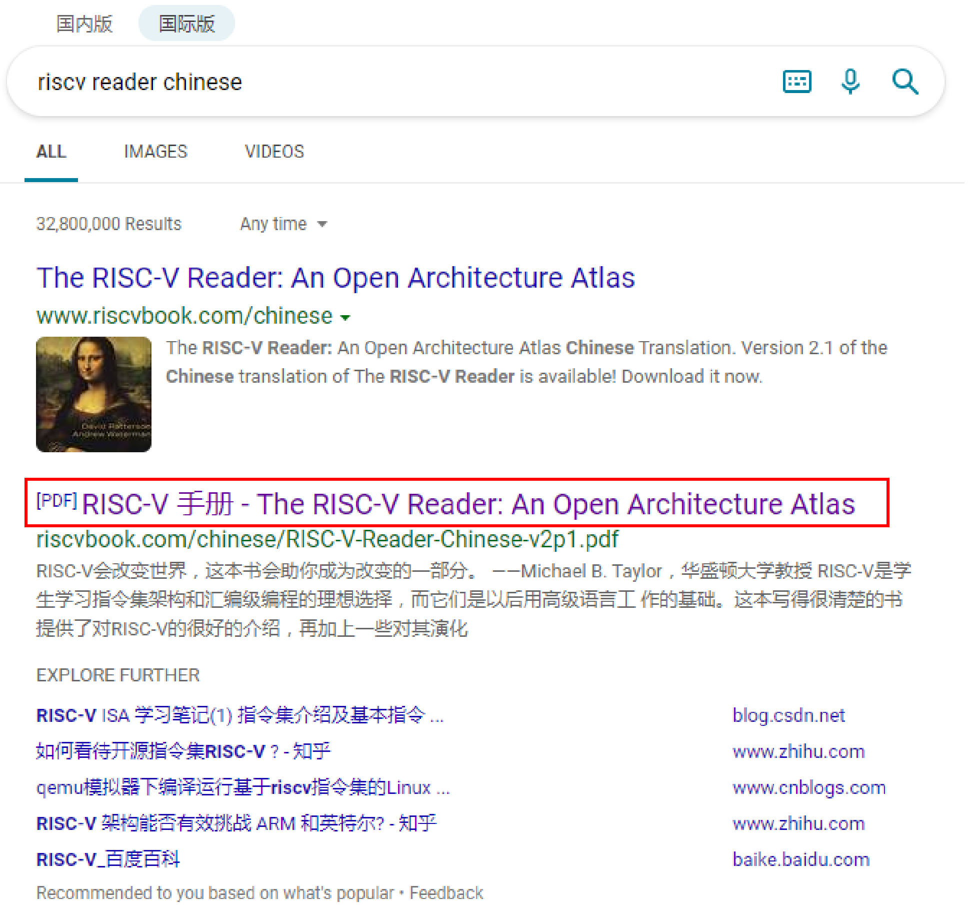
- External Debug Support

Trace Specification

- Trace Specification

*Debug*和*Trace*描述的是RISC-V软硬件的调试机制规范

0.5 哪里找RISC-V文档？



国内版 国际版

riscv reader chinese

ALL IMAGES VIDEOS

32,800,000 Results Any time

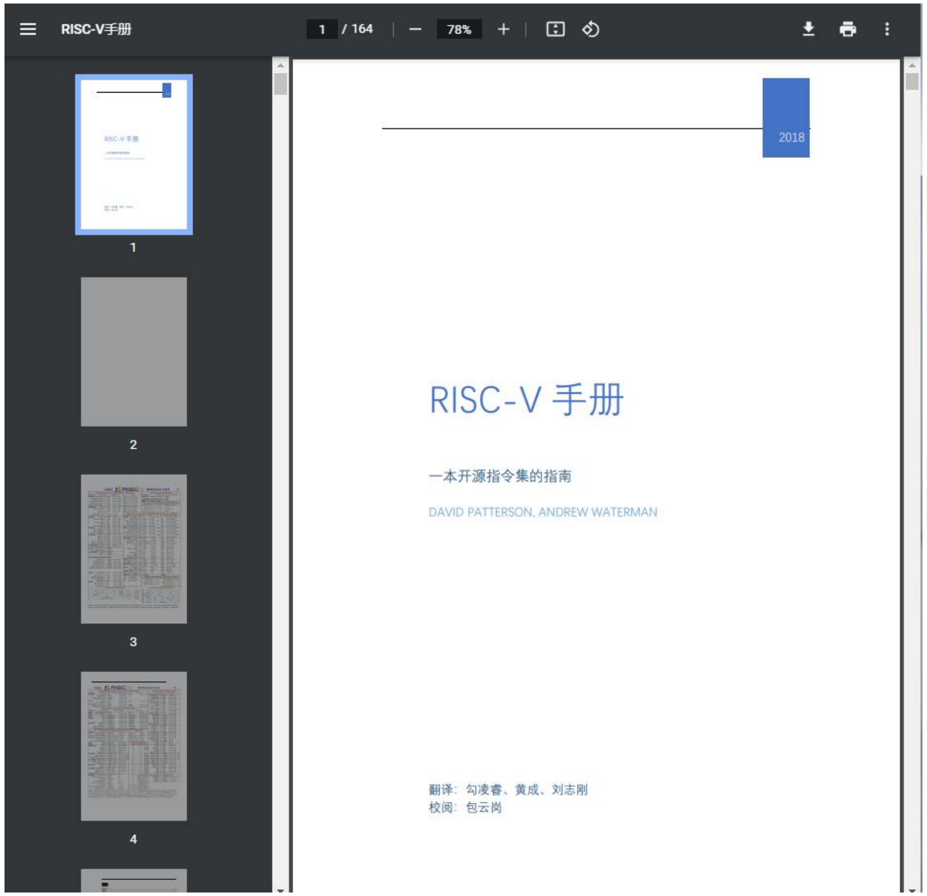
The RISC-V Reader: An Open Architecture Atlas
www.riscvbook.com/chinese
The RISC-V Reader: An Open Architecture Atlas Chinese Translation. Version 2.1 of the Chinese translation of The RISC-V Reader is available! Download it now.

[PDF] RISC-V 手册 - The RISC-V Reader: An Open Architecture Atlas
riscvbook.com/chinese/RISC-V-Reader-Chinese-v2p1.pdf
RISC-V会改变世界，这本书会助你成为改变的一部分。 ——Michael B. Taylor，华盛顿大学教授 RISC-V是学生学习指令集架构和汇编级编程的理想选择，而它们是以后用高级语言工 作的基础。这本写得很清楚的书提供了对RISC-V的很好的介绍，再加上一些对其演化

EXPLORE FURTHER

RISC-V ISA 学习笔记(1) 指令集介绍及基本指令 ...	blog.csdn.net
如何看待开源指令集RISC-V？- 知乎	www.zhihu.com
qemu模拟器下编译运行基于riscv指令集的Linux ...	www.cnblogs.com
RISC-V 架构能否有效挑战 ARM 和英特尔? - 知乎	www.zhihu.com
RISC-V_百度百科	baike.baidu.com

Recommended to you based on what's popular • Feedback



RISC-V手册 1 / 164 78%

2018

RISC-V 手册

一本开源指令集的指南

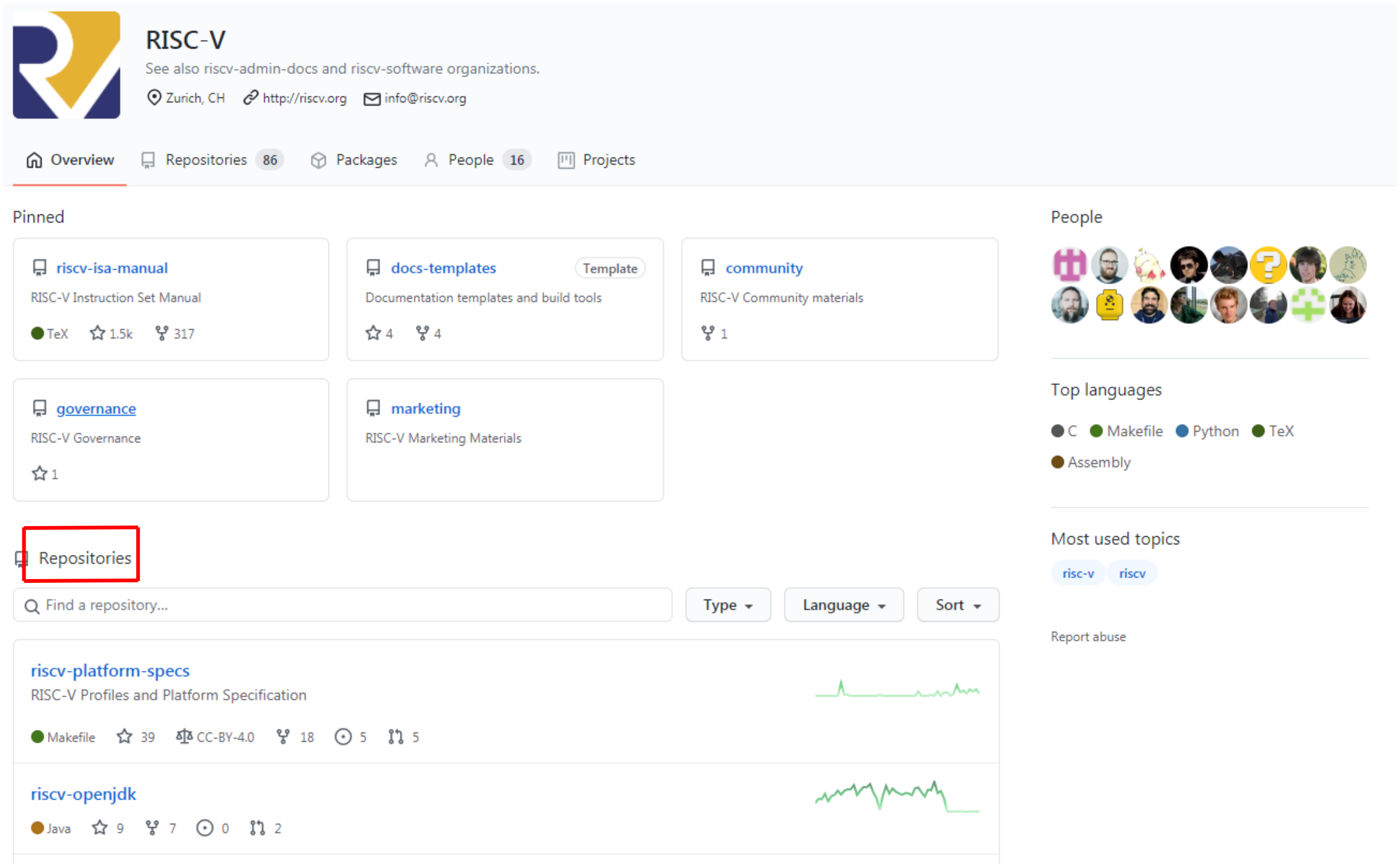
DAVID PATTERSON, ANDREW WATERMAN

翻译：勾凌睿、黄成、刘志刚
校阅：包云岗

英文原本是*The RISC-V Reader: An Open Architecture Atlas*

0.6 哪里找RISC-V软件工具？

https://github.com/riscv



RISC-V

See also riscv-admin-docs and riscv-software organizations.

Zurich, CH http://riscv.org info@riscv.org

Overview Repositories 86 Packages People 16 Projects

Pinned

riscv-isa-manual
RISC-V Instruction Set Manual
TeX ☆ 1.5k 317

docs-templates Template
Documentation templates and build tools
☆ 4 4

community
RISC-V Community materials
1

governance
RISC-V Governance
☆ 1

marketing
RISC-V Marketing Materials

Repositories

Find a repository... Type Language Sort

riscv-platform-specs
RISC-V Profiles and Platform Specification
Makefile ☆ 39 CC-BY-4.0 18 5 5

riscv-openjdk
Java ☆ 9 7 0 2

People

Top languages
C Makefile Python TeX Assembly

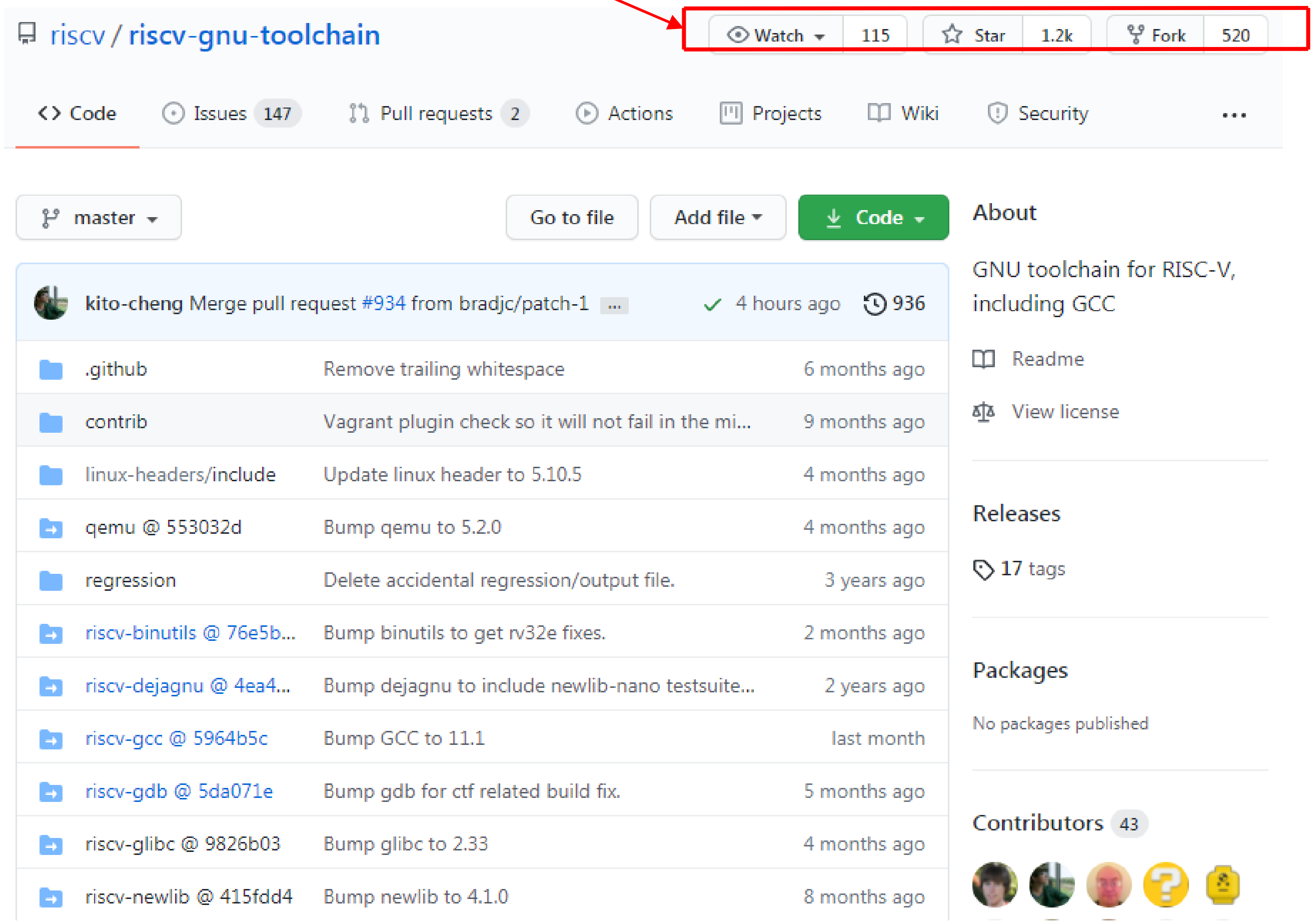
Most used topics
risc-v riscv

Report abuse

RISC-V相关的文档、软件工具都可以在RISC-V的官方GitHub账号上找到

0.7 GitHub使用界面说明

Watch是关注的意思，如果当前仓库有更新，GitHub会通过邮件推送给自己
Star是收藏的意思，方便自己查找
Fork是开发选项，点击Fork会把当前仓库拷贝到自己的GitHub账号上


riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security | ...

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

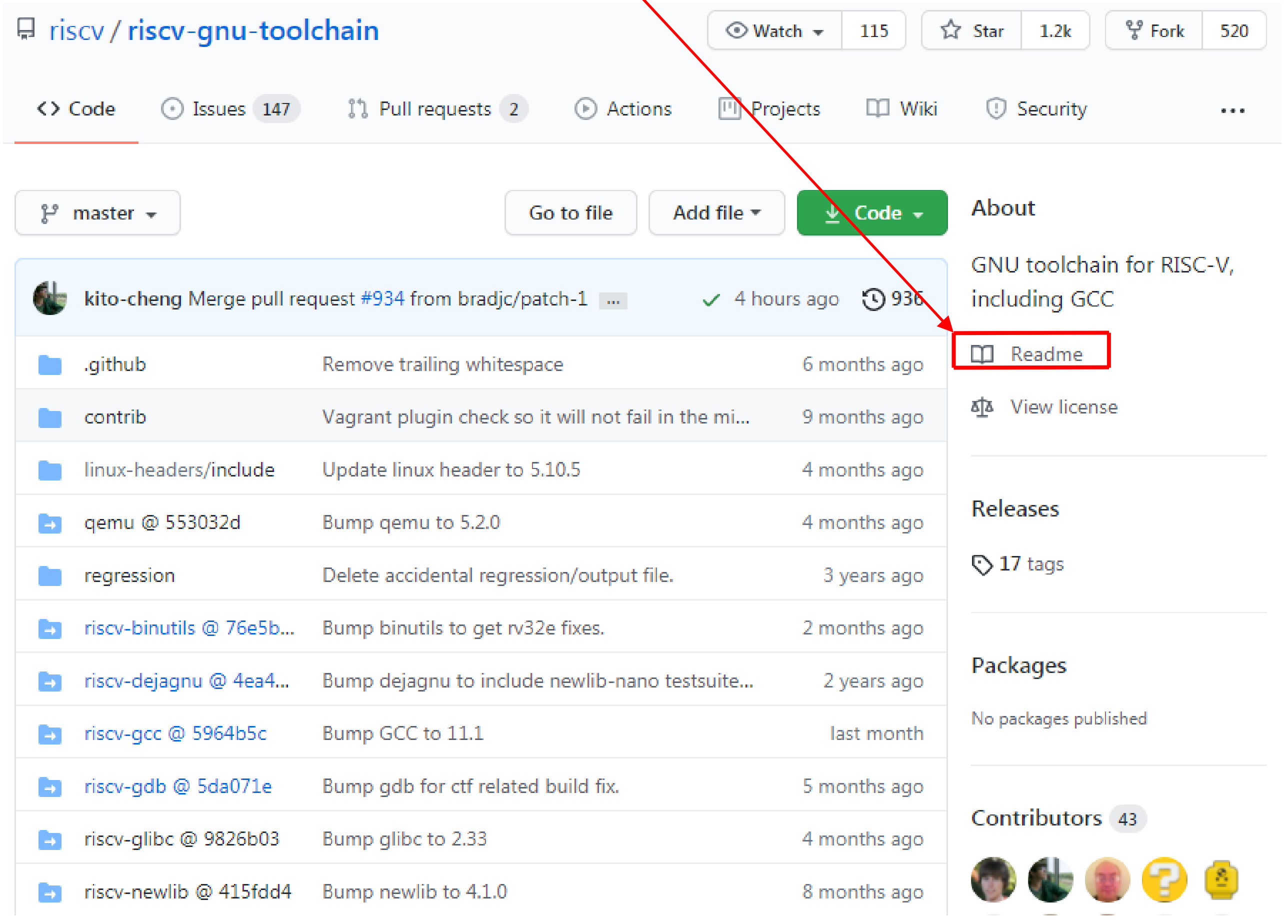
Packages

No packages published

Contributors 43

0.7 GitHub使用界面说明

想要了解一个仓库的内容，必须先阅读Readme文件



riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security | ...

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

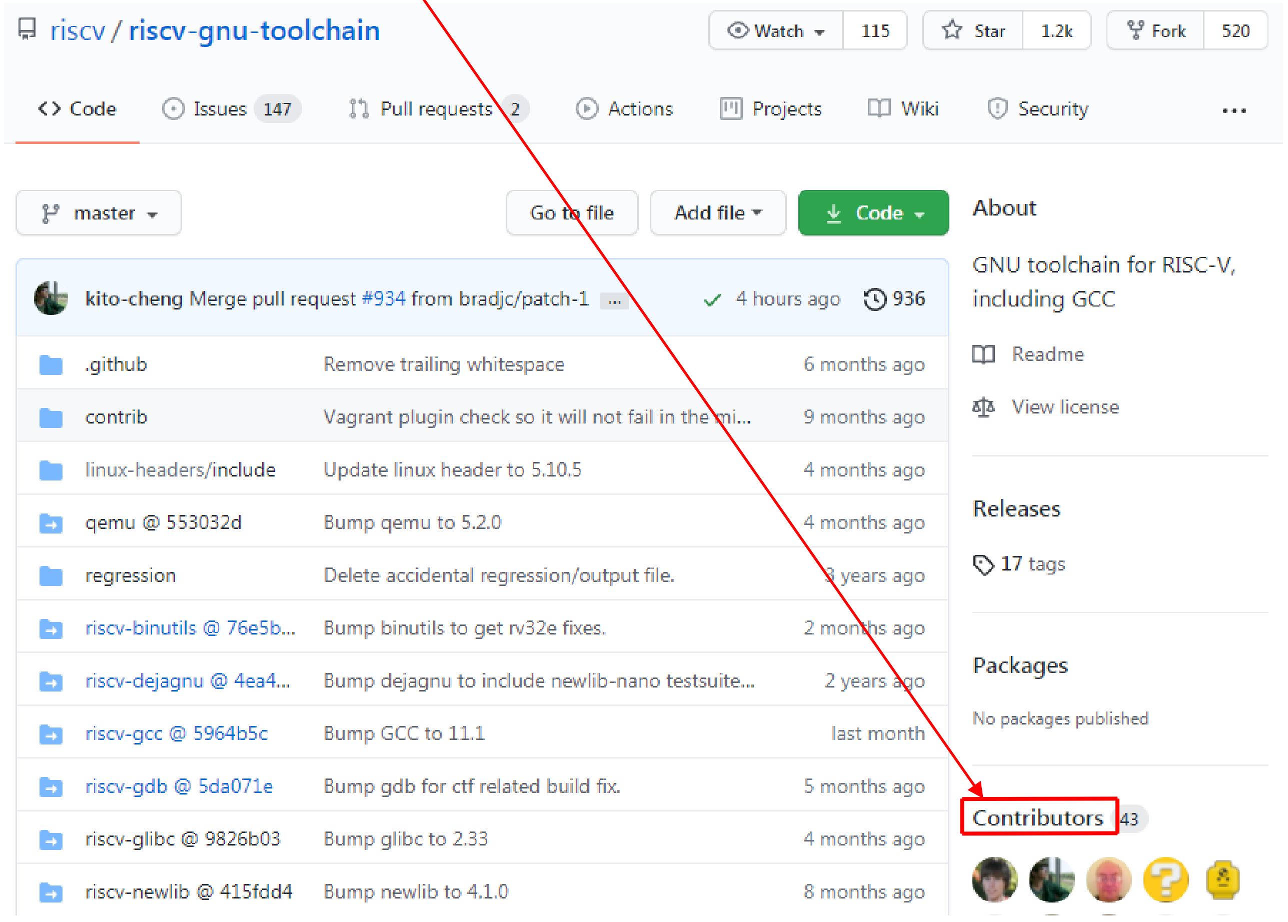
Packages

No packages published

Contributors 43

0.7 GitHub使用界面说明

当前仓库的贡献者



riscv / riscv-gnu-toolchain

Watch 115 · Star 1.2k · Fork 520

Code · Issues 147 · Pull requests 2 · Actions · Projects · Wiki · Security · ...

master · Go to file · Add file · Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

Packages

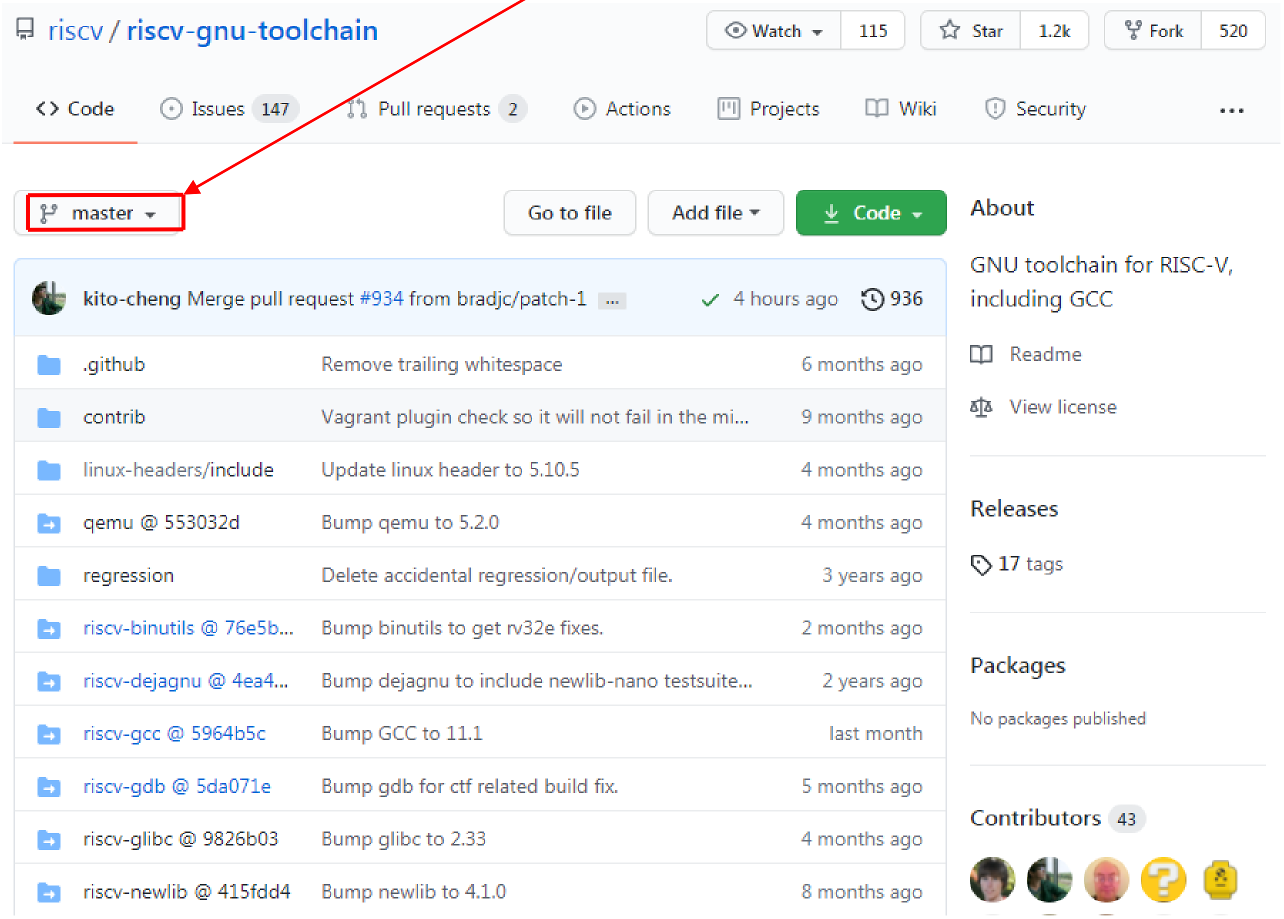
No packages published

Contributors 43

深圳大学 SHENZHEN UNIVERSITY

0.7 GitHub使用界面说明

当前分支名称；一个仓库可能会有若干个分支，不同的分支可能不一样，所以在使用该仓库之前，必须确定自己选对正确的分支


riscv / riscv-gnu-toolchain

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

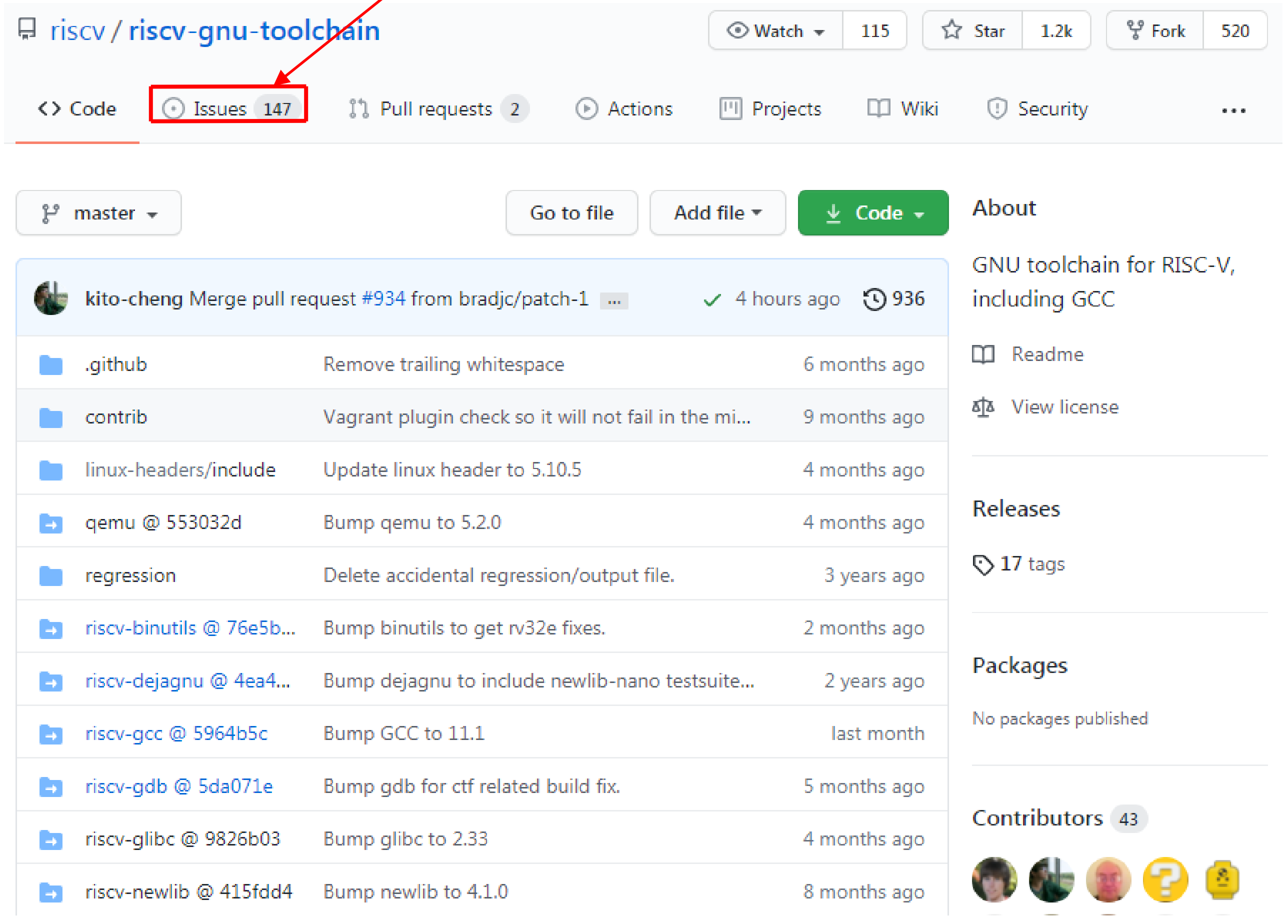
Packages

No packages published

Contributors 43

0.7 GitHub使用界面说明

如果在使用或者开发过程中遇到问题，可以在这里提出


riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

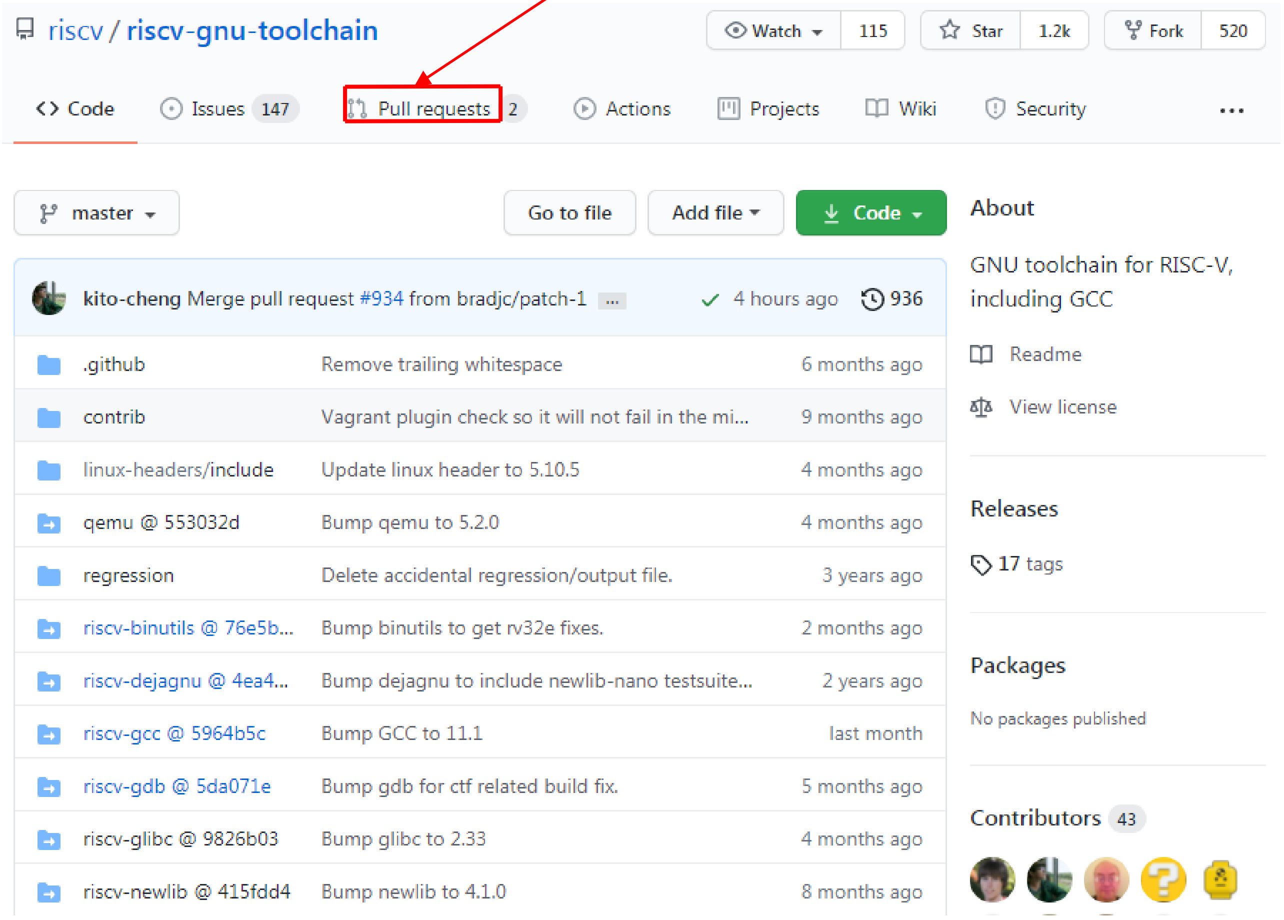
Packages

No packages published

Contributors 43

0.7 GitHub使用界面说明

如果想修改仓库的内容，可以通过Pull requests提交修改后的代码，经过仓库维护者审核后，会直接合并到仓库中



riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security | ...

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

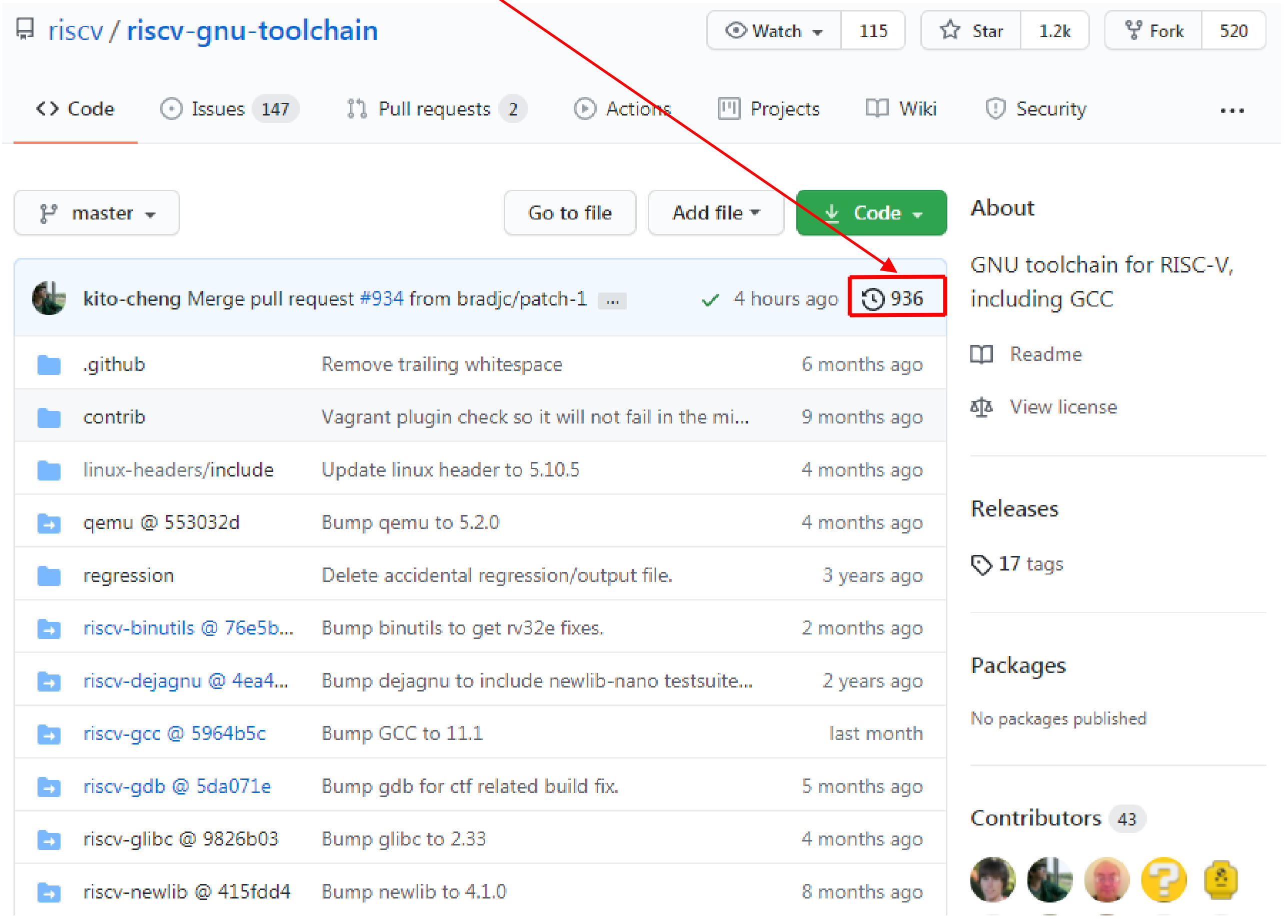
Packages

No packages published

Contributors 43

0.7 GitHub使用界面说明

当前分支的代码提交历史



riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security | ...

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

17 tags

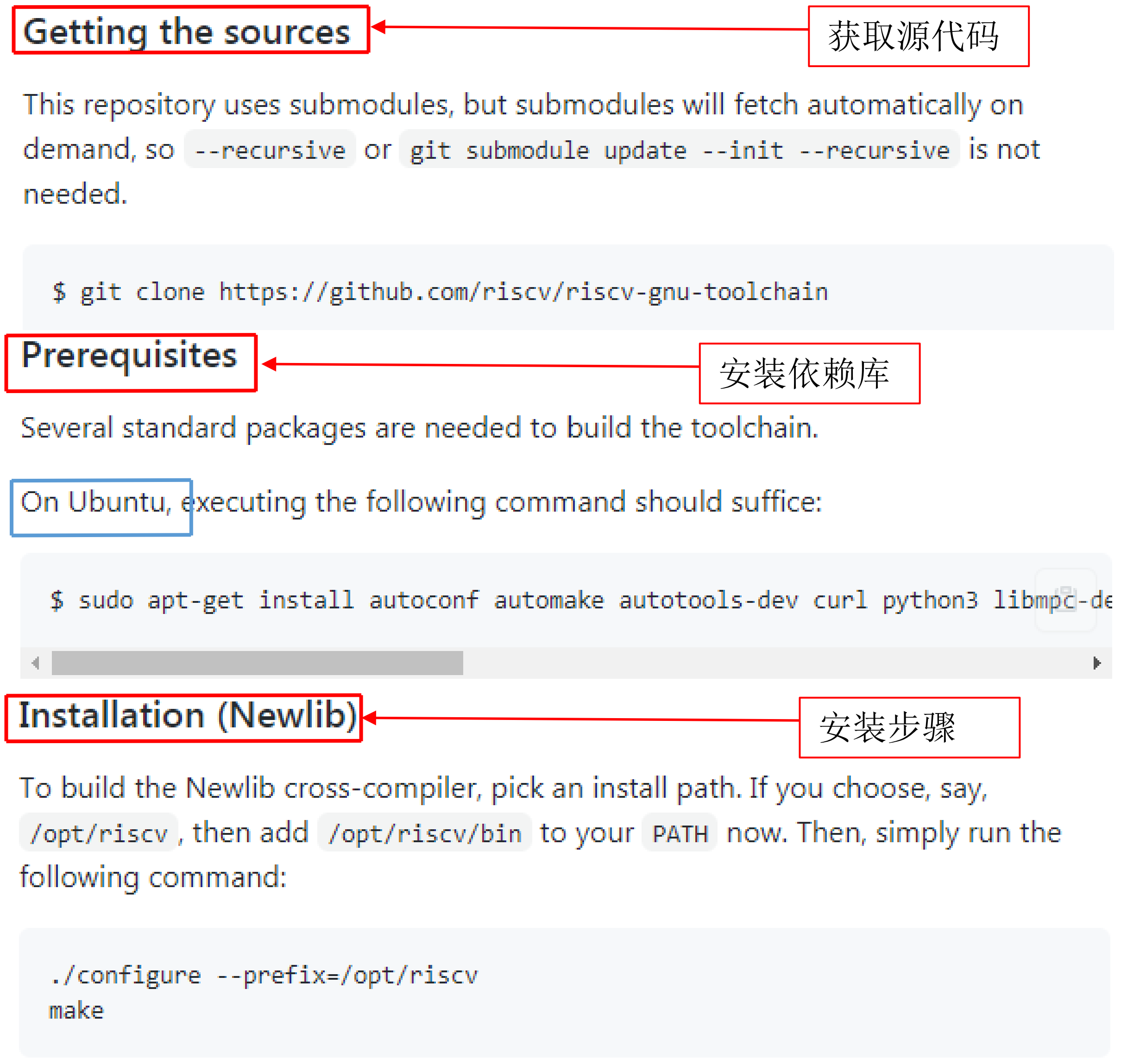
Packages

No packages published

Contributors 43

0.8 如何阅读README？

README文件中描述了具体的构建方法



Getting the sources

获取源代码

This repository uses submodules, but submodules will fetch automatically on demand, so `--recursive` or `git submodule update --init --recursive` is not needed.

```
$ git clone https://github.com/riscv/riscv-gnu-toolchain
```

下载过程可能会奇慢，所以最好科学上网

Prerequisites

安装依赖库

Several standard packages are needed to build the toolchain.

注意这里要求操作系统是Ubuntu

On Ubuntu, executing the following command should suffice:

```
$ sudo apt-get install autoconf automake autotools-dev curl python3 libmpc-de
```

Installation (Newlib)

安装步骤

To build the Newlib cross-compiler, pick an install path. If you choose, say, `/opt/riscv` , then add `/opt/riscv/bin` to your `PATH` now. Then, simply run the following command:

```
./configure --prefix=/opt/riscv
make
```

0.9 作业

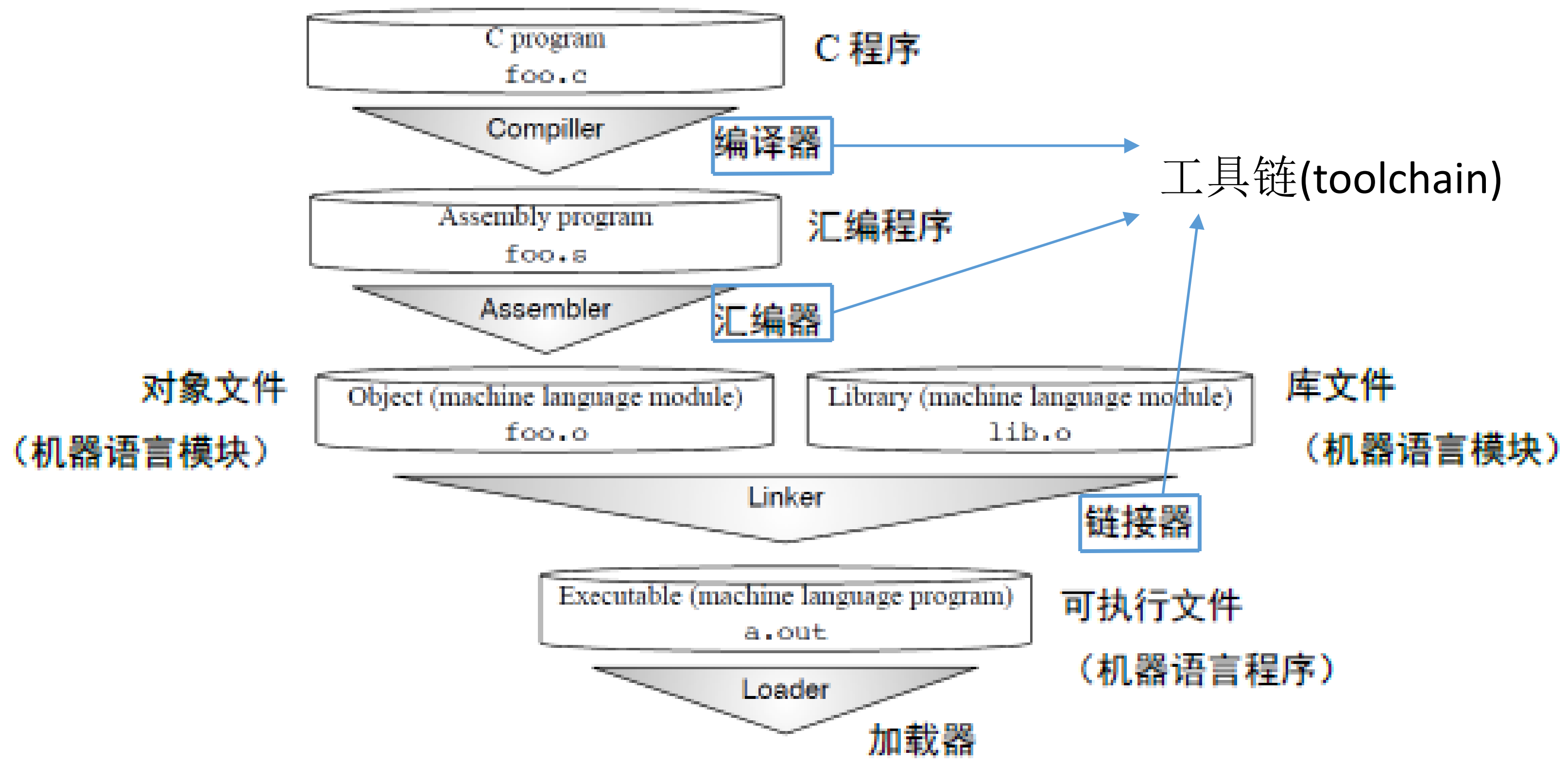
- 在GitHub上创建一个仓库
- 安装Ubuntu系统（建议在Win10上安装WSL）

Thank You!

1 RISC-V工具链介绍

潘志铭

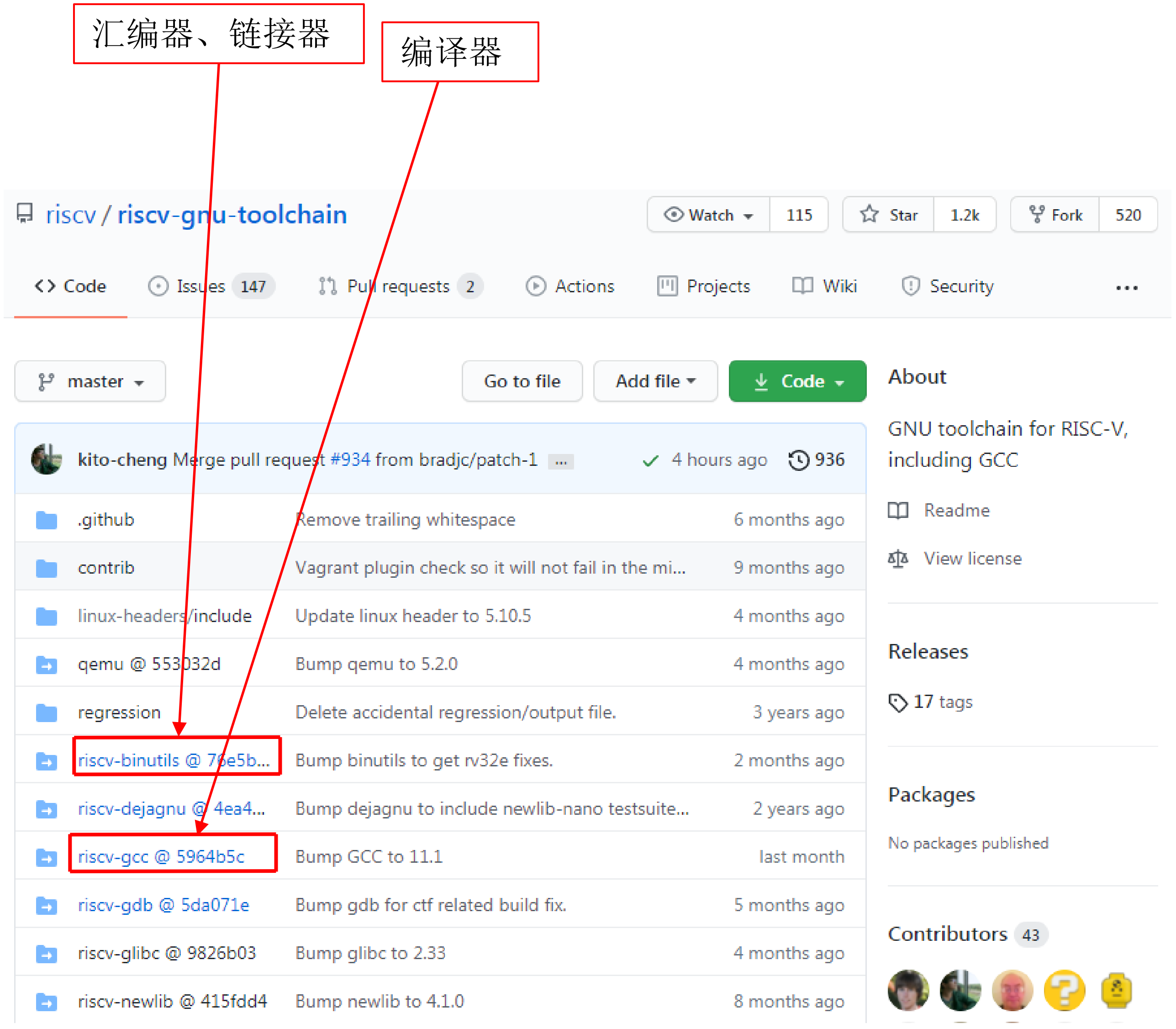
1.1 RISC-V的工具链



从C源代码翻译为可运行程序的步骤

--- 上图来自《RISC-V手册》第41页

1.1 RISC-V的工具链



riscv / riscv-gnu-toolchain

Watch 115 | Star 1.2k | Fork 520

Code | Issues 147 | Pull requests 2 | Actions | Projects | Wiki | Security | ...

master | Go to file | Add file | Code

kito-cheng Merge pull request #934 from bradjc/patch-1 ... ✓ 4 hours ago 936

.github	Remove trailing whitespace	6 months ago
contrib	Vagrant plugin check so it will not fail in the mi...	9 months ago
linux-headers/include	Update linux header to 5.10.5	4 months ago
qemu @ 553032d	Bump qemu to 5.2.0	4 months ago
regression	Delete accidental regression/output file.	3 years ago
riscv-binutils @ 76e5b...	Bump binutils to get rv32e fixes.	2 months ago
riscv-dejagnu @ 4ea4...	Bump dejagnu to include newlib-nano testsuite...	2 years ago
riscv-gcc @ 5964b5c	Bump GCC to 11.1	last month
riscv-gdb @ 5da071e	Bump gdb for ctf related build fix.	5 months ago
riscv-glibc @ 9826b03	Bump glibc to 2.33	4 months ago
riscv-newlib @ 415fdd4	Bump newlib to 4.1.0	8 months ago

About

GNU toolchain for RISC-V, including GCC

Readme

View license

Releases

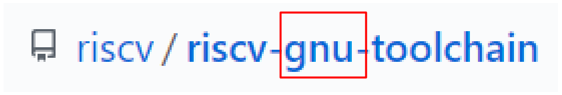
17 tags

Packages

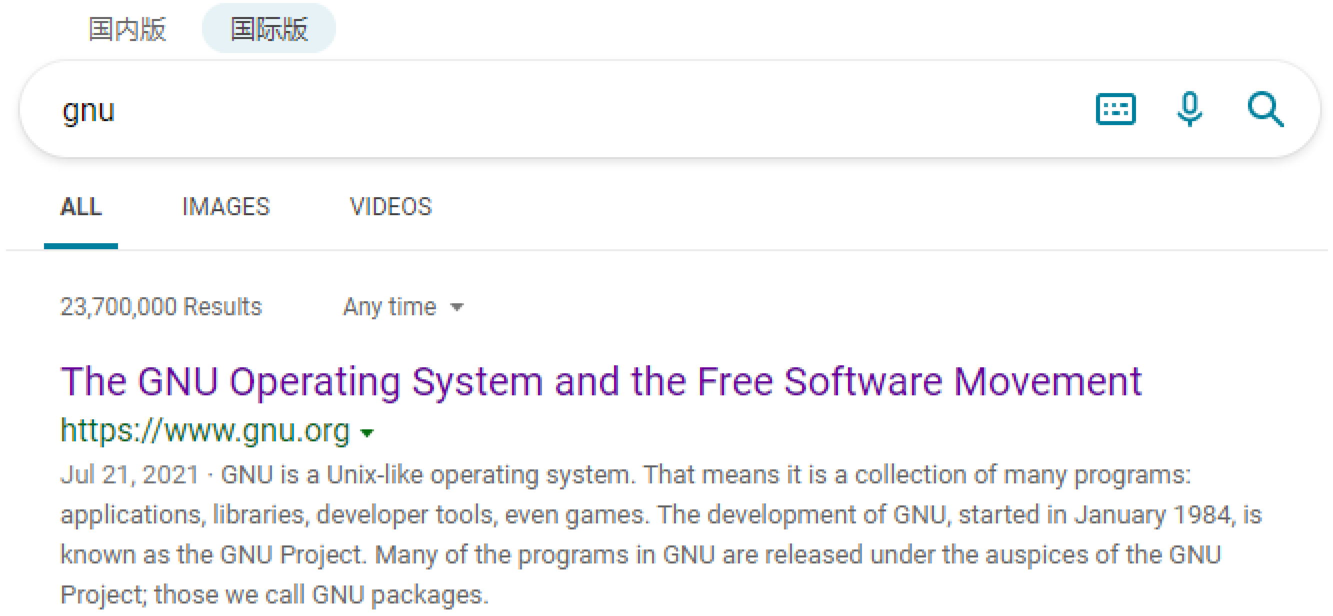
No packages published

Contributors 43

1.2 什么是GNU？



riscv / riscv-gnu-toolchain



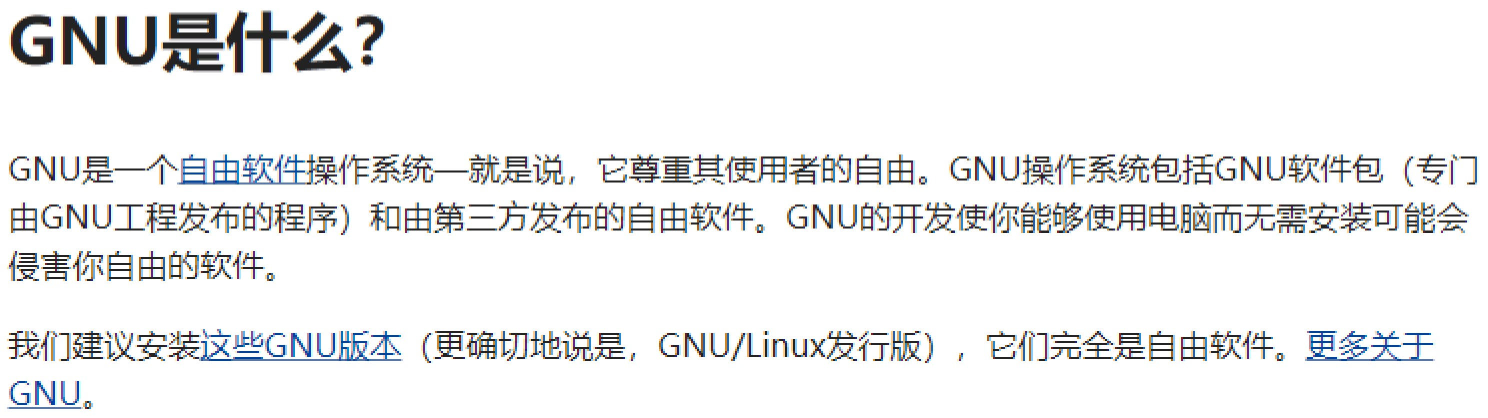
国内版 国际版

gnu

ALL IMAGES VIDEOS

23,700,000 Results Any time

The GNU Operating System and the Free Software Movement
https://www.gnu.org
Jul 21, 2021 · GNU is a Unix-like operating system. That means it is a collection of many programs: applications, libraries, developer tools, even games. The development of GNU, started in January 1984, is known as the GNU Project. Many of the programs in GNU are released under the auspices of the GNU Project; those we call GNU packages.

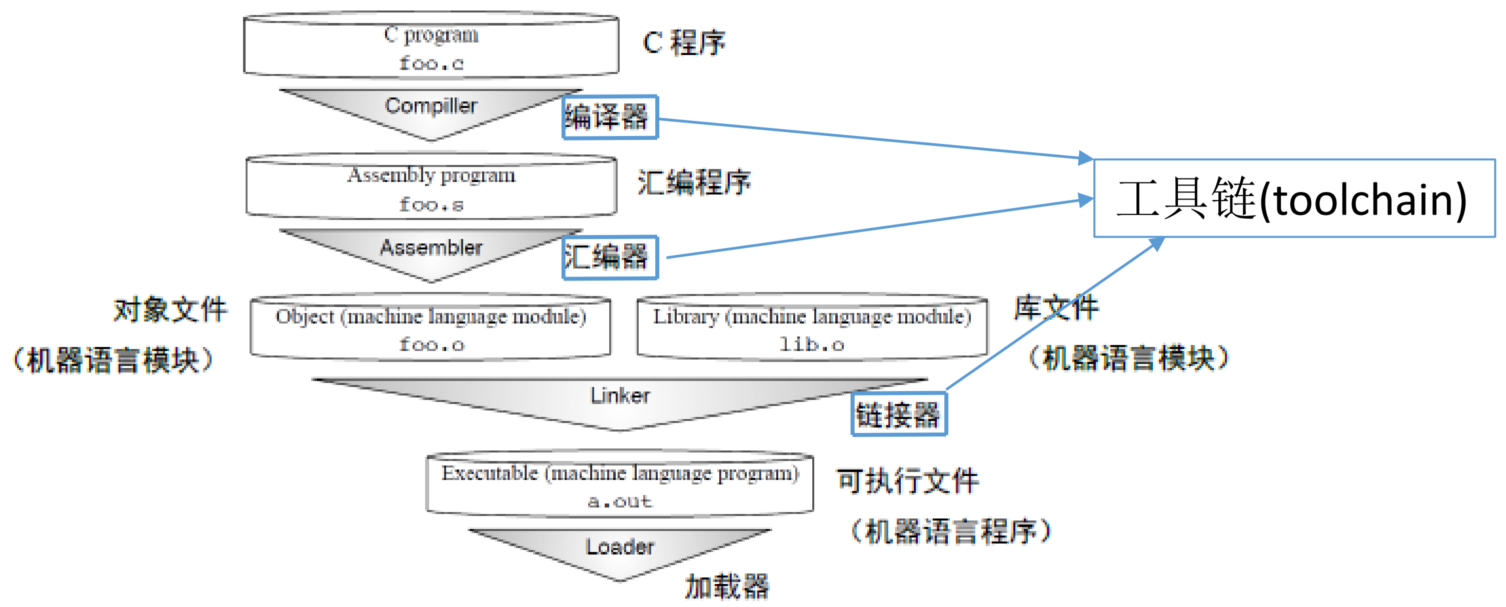


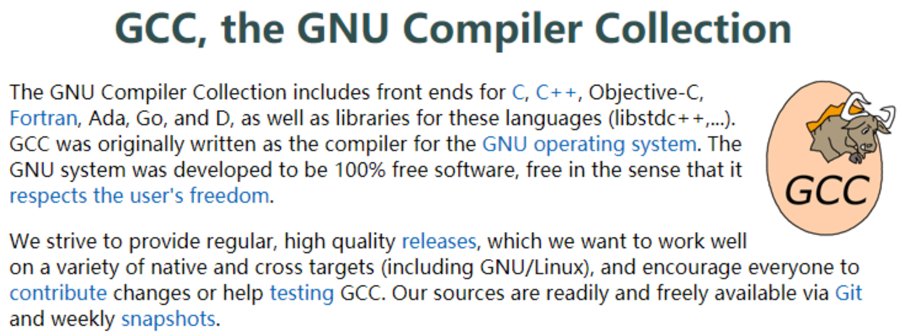
GNU是什么?

GNU是一个自由软件操作系统—就是说，它尊重其使用者的自由。GNU操作系统包括GNU软件包（专门由GNU工程发布的程序）和由第三方发布的自由软件。GNU的开发使你能够使用电脑而无需安装可能会侵害你自由的软件。

我们建议安装这些GNU版本（更确切地说是，GNU/Linux发行版），它们完全是自由软件。更多关于GNU。

1.3 GNU有哪些重要的软件工具？



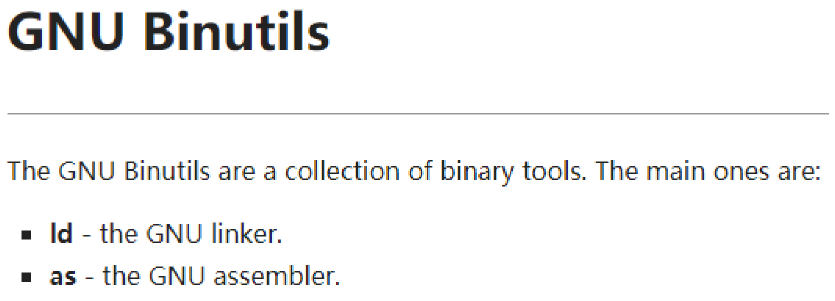


GCC, the GNU Compiler Collection

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Ada, Go, and D, as well as libraries for these languages (libstdc++,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100% free software, free in the sense that it respects the user's freedom.

We strive to provide regular, high quality releases, which we want to work well on a variety of native and cross targets (including GNU/Linux), and encourage everyone to contribute changes or help testing GCC. Our sources are readily and freely available via Git and weekly snapshots.

https://gcc.gnu.org



GNU Binutils

The GNU Binutils are a collection of binary tools. The main ones are:

- **ld** - the GNU linker.
- **as** - the GNU assembler.

https://www.gnu.org/software/binutils

--- 上图来自《RISC-V手册》第41页

1.4 GNU Compiler Collection(GCC)

- 广义的GCC实质上是多个程序的集合：
  - GCC(GNU C Compiler)是编译工具，能够将C/C++语言编写的程序转换成处理器能够执行的二进制代码；
  - GCC既支持本地编译（即在一个平台上编译该平台运行的程序），也支持交叉编译（即在一个平台上编译供另外一个平台运行的程序）；
  - Binutils(Binary Utilities)是一组二进制程序处理工具，包含汇编器as，链接器ld，反汇编器objdump，查看ELF文件信息的readelf，查看ELF文件大小的size等等；
  - C运行库，应用最为广泛的有glibc(GNU C Library)，以及newlib（相比glibc更小，适合单片机系统使用）

1.5 如何构建RISC-V工具链？

- 获取源代码

```
$ git clone https://github.com/riscv/riscv-gnu-toolchain.git
```

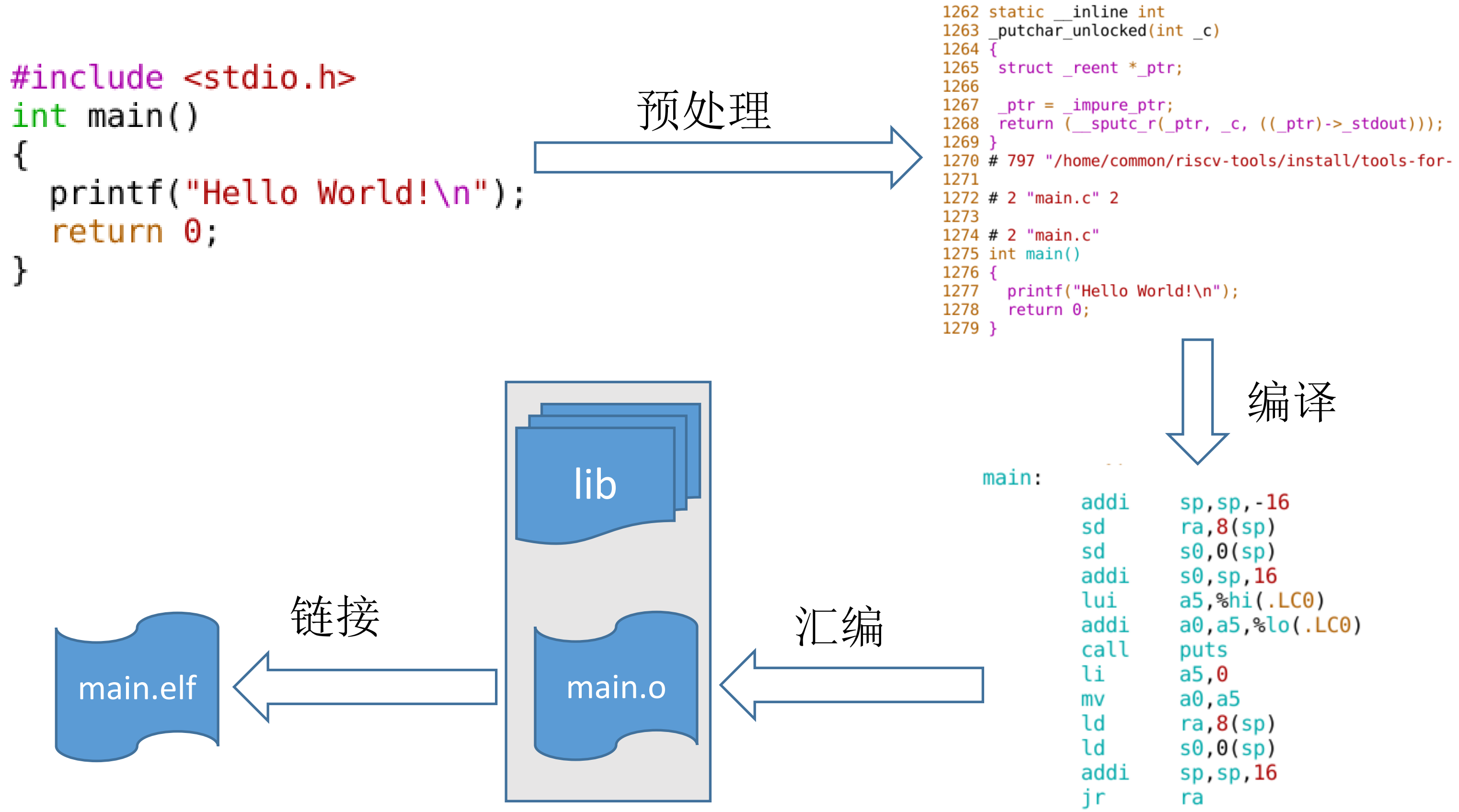
- 安装依赖文件

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

- 开始构建

```
$ cd riscv-gnu-toolchain
$ mkdir build && cd build
$ ../configure --prefix=/opt/riscv --enable-multilib
$ make
$ make install
```

1.6 RISC-V工具链的简单使用



```c
#include <stdio.h>
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

```c
static __inline int
_putchar_unlocked(int _c)
{
 struct _reent *_ptr;

 _ptr = _impure_ptr;
 return (__sputc_r(_ptr, _c, ((_ptr)->_stdout)));
}
# 797 "/home/common/riscv-tools/install/tools-for-

# 2 "main.c" 2

# 2 "main.c"
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

```asm
main:
          addi    sp,sp,-16
          sd      ra,8(sp)
          sd      s0,0(sp)
          addi    s0,sp,16
          lui     a5,%hi(.LC0)
          addi    a0,a5,%lo(.LC0)
          call    puts
          li      a5,0
          mv      a0,a5
          ld      ra,8(sp)
          ld      s0,0(sp)
          addi    sp,sp,16
          jr      ra
          .size   main, .-main
          .ident  "GCC: (GNU) 10.1.0"
```

上面的操作可以通过一个命令实现：

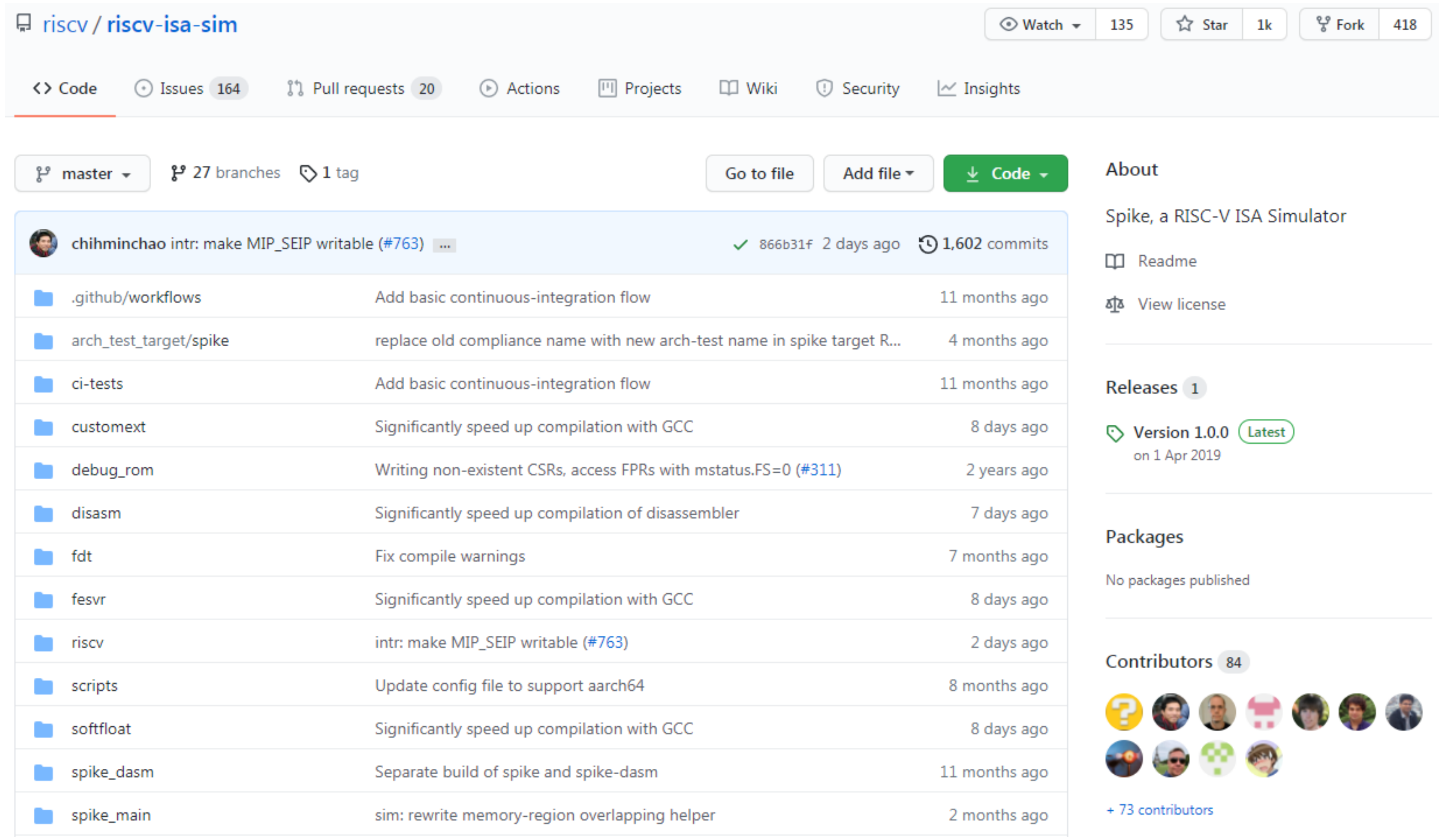
```
$ riscv64-unknown-elf-gcc main.c -o main.elf
```

如果出现`riscv64-unknown-elf-gcc: command not found`的错误，需要将工具链路径加入PATH中：

```
$ export PATH=/opt/riscv:$PATH
```

1.7 指令模拟器spike的构建

spike是一个RISC-V指令模拟器，它能够模拟一个或者多个RISC-V硬件线程的功能。


riscv / riscv-isa-sim

Watch 135 | Star 1k | Fork 418

Code | Issues 164 | Pull requests 20 | Actions | Projects | Wiki | Security | Insights

master | 27 branches | 1 tag | Go to file | Add file | Code

chihminchao intr: make MIP_SEIP writable (#763) ... ✓ 866b31f 2 days ago 1,602 commits

.github/workflows	Add basic continuous-integration flow	11 months ago
arch_test_target/spike	replace old compliance name with new arch-test name in spike target R...	4 months ago
ci-tests	Add basic continuous-integration flow	11 months ago
customext	Significantly speed up compilation with GCC	8 days ago
debug_rom	Writing non-existent CSRs, access FPRs with mstatus.FS=0 (#311)	2 years ago
disasm	Significantly speed up compilation of disassembler	7 days ago
fdt	Fix compile warnings	7 months ago
fesvr	Significantly speed up compilation with GCC	8 days ago
riscv	intr: make MIP_SEIP writable (#763)	2 days ago
scripts	Update config file to support aarch64	8 months ago
softfloat	Significantly speed up compilation with GCC	8 days ago
spike_dasm	Separate build of spike and spike-dasm	11 months ago
spike_main	sim: rewrite memory-region overlapping helper	2 months ago

About

Spike, a RISC-V ISA Simulator

Readme

View license

Releases 1

Version 1.0.0 Latest
on 1 Apr 2019

Packages

No packages published

Contributors 84

+ 73 contributors

1.7 指令模拟器spike的构建

- 获取源码

```
$ git clone https://github.com/riscv/riscv.isa-sim.git
```

- 安装依赖文件

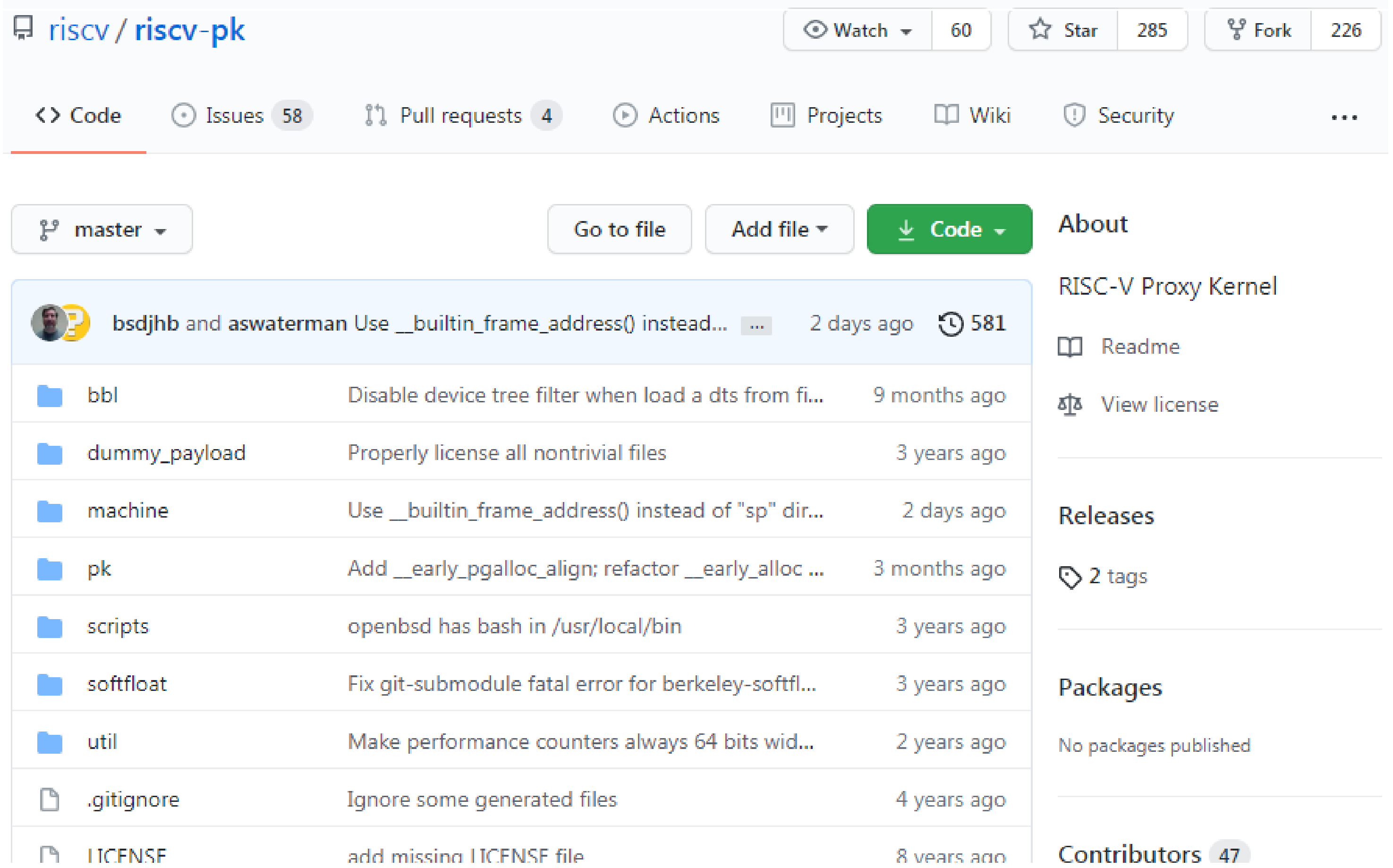
```
$ sudo apt-get install device-tree-compile
```

- 开始构建

```
$ cd riscv-isa-sim
$ mkdir build
$ cd build
$ ../configure –prefix=/opt/riscv -enable-histogram
$ make
$ make install
```

1.8 代理内核pk的构建

RISC-V Proxy Kernel（RISC-V代理内核）用于给RISC-V执行文件提供执行环境。


riscv / riscv-pk

Watch 60 | Star 285 | Fork 226

Code | Issues 58 | Pull requests 4 | Actions | Projects | Wiki | Security

master | Go to file | Add file | Code

bsdjhb and aswaterman Use __builtin_frame_address() instead... … 2 days ago 581

bbl	Disable device tree filter when load a dts from fi...	9 months ago
dummy_payload	Properly license all nontrivial files	3 years ago
machine	Use __builtin_frame_address() instead of "sp" dir...	2 days ago
pk	Add __early_pgalloc_align; refactor __early_alloc ...	3 months ago
scripts	openbsd has bash in /usr/local/bin	3 years ago
softfloat	Fix git-submodule fatal error for berkeley-softfl...	3 years ago
util	Make performance counters always 64 bits wid...	2 years ago
.gitignore	Ignore some generated files	4 years ago
LICENSE	add missing LICENSE file	8 years ago

About

RISC-V Proxy Kernel

Readme

View license

Releases

2 tags

Packages

No packages published

Contributors 47

1.8 代理内核pk的构建

- 获取源码

```
$ git clone https://github.com/riscv/riscv-pk.git
```

- 开始构建

```
$ cd riscv-pk
$ mkdir build
$ cd build
$ ../configure -prefix=/opt/riscv -host=riscv64-unknown-elf
$ make
$ make install
```

构建pk之前需要先完成riscv-gnu-toolchain的安装

如果出现`riscv64-unknown-elf-gcc: command not found`的错误，需要将工具链路径加入PATH中：

```
$ export PATH=/opt/riscv:$PATH
```

1.9 结合spike和pk运行程序

- 编译helloword程序

```
$ riscv64-unknown-elf-gcc main.c -o main.elf
```

- 将Spike路径加入PATH环境变量中

```
$ export PATH=/opt/riscv:$PATH
```

- 将pk路径加入PATH环境变量中

```
$ export PATH=/opt/riscv/riscv64-unknown-elf/bin:$PATH
```

- 运行spike和pk

```
$ spike pk main.elf
```

1.10 作业

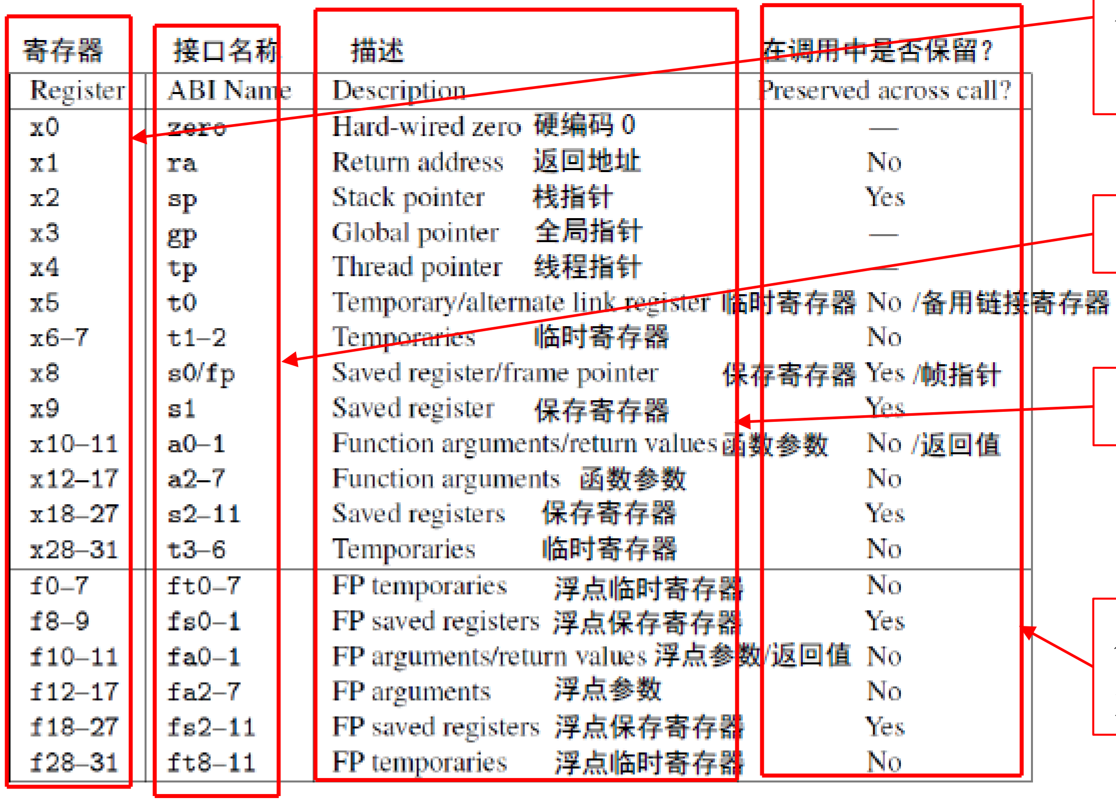
- 完成riscv-gnu-toolchain的构建
- 完成spike和pk的构建
- 编写一个简单的C语言程序，并且在spike上运行

Thank You!

2 RISC-V汇编语言

潘志铭

2.1 RISC-V通用寄存器



寄存器 Register	接口名称 ABI Name	描述 Description	在调用中是否保留? Preserved across call?
x0	zero	Hard-wired zero 硬编码 0	—
x1	ra	Return address 返回地址	No
x2	sp	Stack pointer 栈指针	Yes
x3	gp	Global pointer 全局指针	—
x4	tp	Thread pointer 线程指针	—
x5	t0	Temporary/alternate link register 临时寄存器	No /备用链接寄存器
x6–7	t1–2	Temporaries 临时寄存器	No
x8	s0/fp	Saved register/frame pointer 保存寄存器	Yes /帧指针
x9	s1	Saved register 保存寄存器	Yes
x10–11	a0–1	Function arguments/return values 函数参数	No /返回值
x12–17	a2–7	Function arguments 函数参数	No
x18–27	s2–11	Saved registers 保存寄存器	Yes
x28–31	t3–6	Temporaries 临时寄存器	No
f0–7	ft0–7	FP temporaries 浮点临时寄存器	No
f8–9	fs0–1	FP saved registers 浮点保存寄存器	Yes
f10–11	fa0–1	FP arguments/return values 浮点参数/返回值	No
f12–17	fa2–7	FP arguments 浮点参数	No
f18–27	fs2–11	FP saved registers 浮点保存寄存器	Yes
f28–31	ft8–11	FP temporaries 浮点临时寄存器	No

RISC-V定义了32个64-bit位宽的整型寄存器，和32个64-bit位宽的浮点寄存器[1]

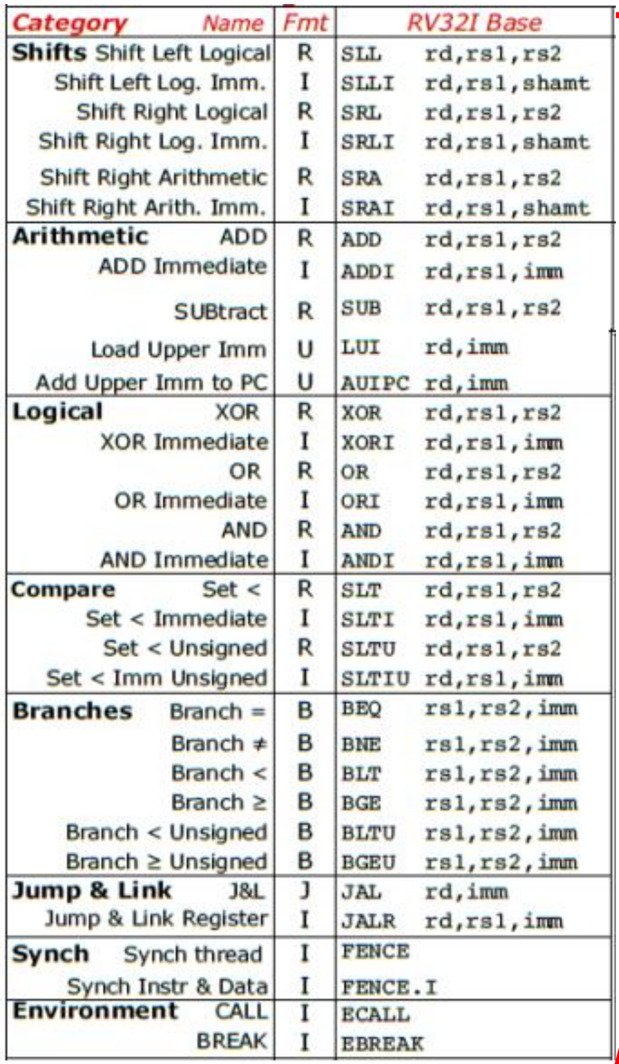
RISC-V各个寄存器的ABI别名

RISC-V各个寄存器的作用描述

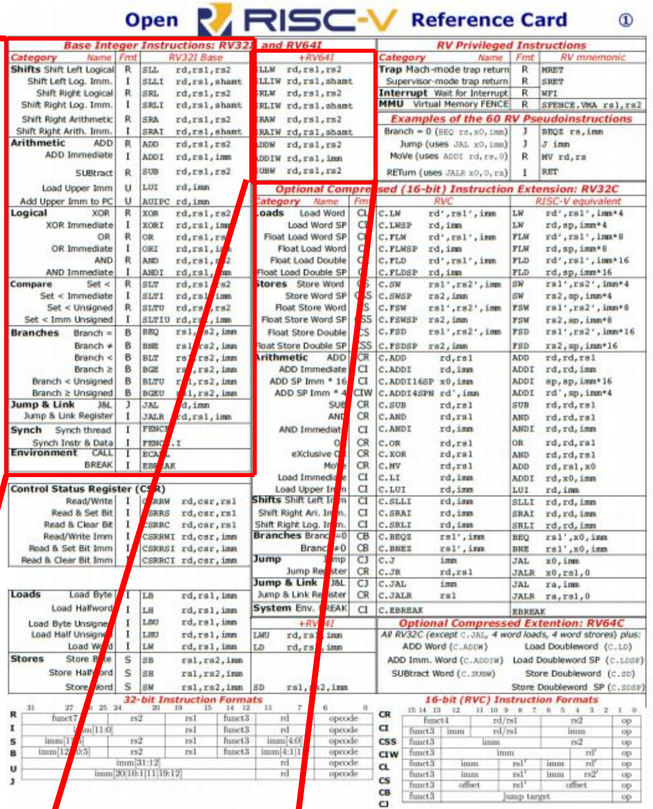
在函数调用前后，寄存器的值是否需要保持不变

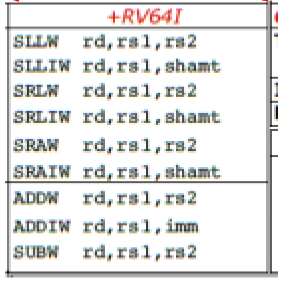
--- 上图来自《RISC-V手册》第42页
--- [1]本课程以rv64指令集讲解

2.2 RISC-V汇编指令



Category	Name	Fmt	RV32I Base	
Shifts	Shift Left Logical	R	SLL	rd,rs1,rs2
	Shift Left Log. Imm.	I	SLLI	rd,rs1,shamt
	Shift Right Logical	R	SRL	rd,rs1,rs2
	Shift Right Log. Imm.	I	SRLI	rd,rs1,shamt
	Shift Right Arithmetic	R	SRA	rd,rs1,rs2
	Shift Right Arith. Imm.	I	SRAI	rd,rs1,shamt
Arithmetic	ADD	R	ADD	rd,rs1,rs2
	ADD Immediate	I	ADDI	rd,rs1,imm
	SUBtract	R	SUB	rd,rs1,rs2
	Load Upper Imm	U	LUI	rd,imm
	Add Upper Imm to PC	U	AUIPC	rd,imm
Logical	XOR	R	XOR	rd,rs1,rs2
	XOR Immediate	I	XORI	rd,rs1,imm
	OR	R	OR	rd,rs1,rs2
	OR Immediate	I	ORI	rd,rs1,imm
	AND	R	AND	rd,rs1,rs2
	AND Immediate	I	ANDI	rd,rs1,imm
Compare	Set <	R	SLT	rd,rs1,rs2
	Set < Immediate	I	SLTI	rd,rs1,imm
	Set < Unsigned	R	SLTU	rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm
Branches	Branch =	B	BEQ	rs1,rs2,imm
	Branch ≠	B	BNE	rs1,rs2,imm
	Branch <	B	BLT	rs1,rs2,imm
	Branch ≥	B	BGE	rs1,rs2,imm
	Branch < Unsigned	B	BLTU	rs1,rs2,imm
	Branch ≥ Unsigned	B	BGEU	rs1,rs2,imm
Jump & Link	J&L	J	JAL	rd,imm
	Jump & Link Register	I	JALR	rd,rs1,imm
Synch	Synch thread	I	FENCE	
	Synch Instr & Data	I	FENCE.I	
Environment	CALL	I	ECALL	
	BREAK	I	EBREAK	



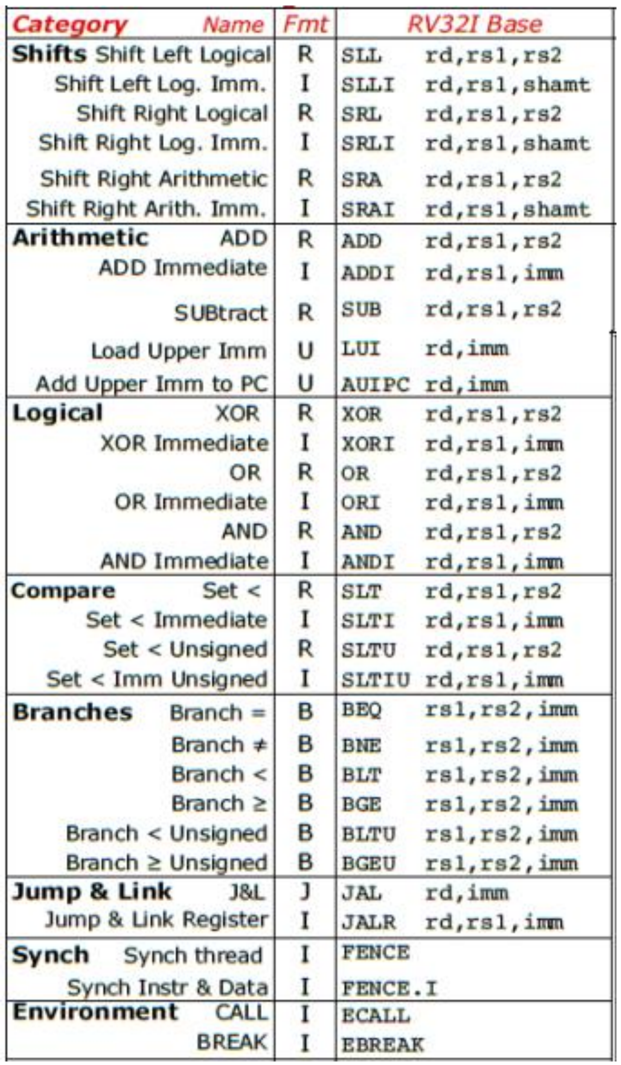


+RV64I	
SLLW	rd,rs1,rs2
SLLIW	rd,rs1,shamt
SRLW	rd,rs1,rs2
SRLIW	rd,rs1,shamt
SRAW	rd,rs1,rs2
SRAIW	rd,rs1,shamt
ADDW	rd,rs1,rs2
ADDIW	rd,rs1,imm
SUBW	rd,rs1,rs2

RV64I指令在RV32I指令基础上增加了部分移位和算术指令

--- 上图来自《RISC-V手册》第1页

2.2 RISC-V汇编指令



Category	Name	Fmt	RV32I Base	
Shifts	Shift Left Logical	R	SLL	rd,rs1,rs2
	Shift Left Log. Imm.	I	SLLI	rd,rs1,shamt
	Shift Right Logical	R	SRL	rd,rs1,rs2
	Shift Right Log. Imm.	I	SRLI	rd,rs1,shamt
	Shift Right Arithmetic	R	SRA	rd,rs1,rs2
	Shift Right Arith. Imm.	I	SRAI	rd,rs1,shamt
Arithmetic	ADD	R	ADD	rd,rs1,rs2
	ADD Immediate	I	ADDI	rd,rs1,imm
	SUBtract	R	SUB	rd,rs1,rs2
	Load Upper Imm	U	LUI	rd,imm
	Add Upper Imm to PC	U	AUIPC	rd,imm
Logical	XOR	R	XOR	rd,rs1,rs2
	XOR Immediate	I	XORI	rd,rs1,imm
	OR	R	OR	rd,rs1,rs2
	OR Immediate	I	ORI	rd,rs1,imm
	AND	R	AND	rd,rs1,rs2
	AND Immediate	I	ANDI	rd,rs1,imm
Compare	Set <	R	SLT	rd,rs1,rs2
	Set < Immediate	I	SLTI	rd,rs1,imm
	Set < Unsigned	R	SLTU	rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm
Branches	Branch =	B	BEQ	rs1,rs2,imm
	Branch ≠	B	BNE	rs1,rs2,imm
	Branch <	B	BLT	rs1,rs2,imm
	Branch ≥	B	BGE	rs1,rs2,imm
	Branch < Unsigned	B	BLTU	rs1,rs2,imm
	Branch ≥ Unsigned	B	BGEU	rs1,rs2,imm
Jump & Link	J&L	J	JAL	rd,imm
	Jump & Link Register	I	JALR	rd,rs1,imm
Synch	Synch thread	I	FENCE	
	Synch Instr & Data	I	FENCE.I	
Environment	CALL	I	ECALL	
	BREAK	I	EBREAK	

--- 上图来自《RISC-V手册》第1页

- RISC-V指令的一般格式：
  op dst, src1, src2
  - op = 操作符名称（operator）
  - dst = 结果寄存器（destination）
  - src1 = 操作数1（source 1）
  - src2 = 操作数2（source 2）

- RV32I指令的分类：
  - Shift：移位指令
  - Arithmetic：算术指令
  - Logical：逻辑操作
  - Compare：比较操作
  - Branches：分支指令
  - Jump & Link：跳转指令
  - Synch：同步指令（不涉及）
  - Environment：环境调用（不涉及）

- RV64I指令在RV32I指令基础上增加了部分移位和算术指令

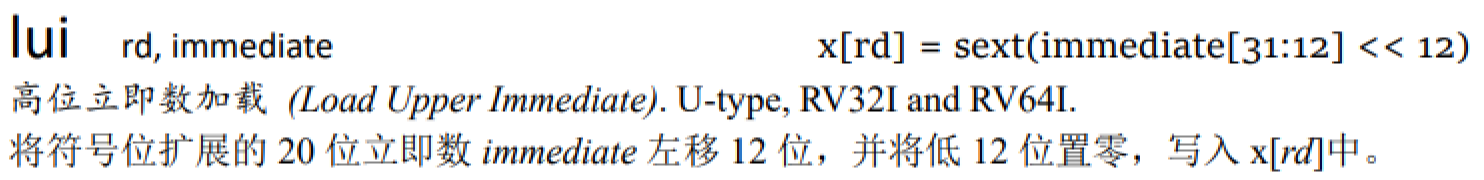
2.3 RISC-V移位指令

Instructions Name	RISC-V
Shift Left Logical	sll rd, rs1, rs2
Shift Left Logical immediate	slli rd, rs1, shamt
Shift Right Logical	srl rd, rs1, rs2
Shift Right Logical immediate	srli rd, rs1, shamt
Shift Right Arithmetic	sra rd, rs1, rs2
Shift Right Arithmetic immediate	srai rd, rs1, shamt

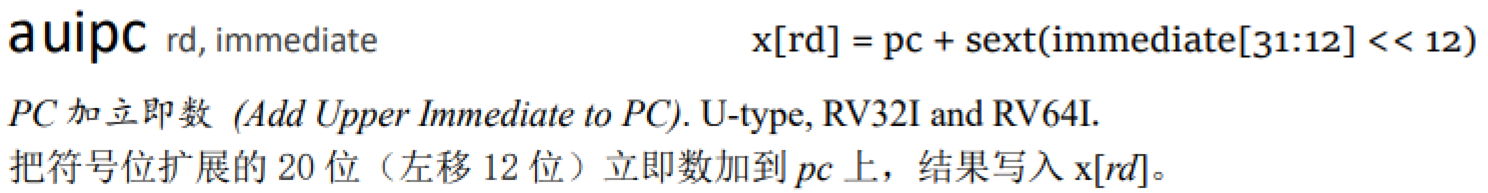
- Logical shift: 逻辑移位，高位填0
- Arithmetic shift: 算术移位，高位填符号位

2.4 RISC-V算术指令

Instructions Name	RISC-V
Add	add rd, rs1, rs2
Add Immediate	addi rd, rs1, imm
Subtract	sub rd, rs1, rs2
Load Upper Immediate	lui rd, imm
Add Upper Immediate to PC	auipc rd, imm


lui rd, immediate   x[rd] = sext(immediate[31:12] << 12)

高位立即数加载 *(Load Upper Immediate)*. U-type, RV32I and RV64I.
将符号位扩展的 20 位立即数 *immediate* 左移 12 位，并将低 12 位置零，写入 x[*rd*]中。


auipc rd, immediate   x[rd] = pc + sext(immediate[31:12] << 12)

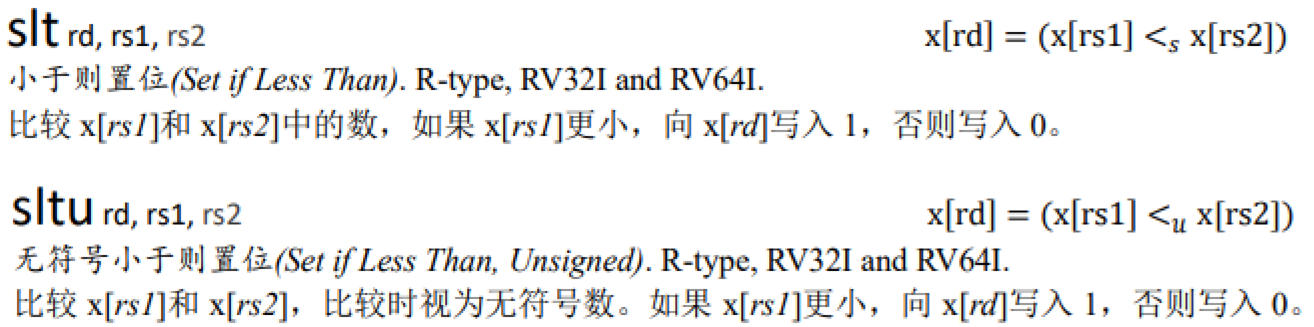
PC 加立即数 (Add Upper Immediate to PC). U-type, RV32I and RV64I.
把符号位扩展的 20 位（左移 12 位）立即数加到 *pc* 上，结果写入 x[*rd*]。

2.5 RISC-V逻辑操作指令

Instructions Name	RISC-V
XOR	xor rd, rs1, rs2
XOR Immediate	xori rd, rs1, imm
OR	or rd, rs1, rs2
OR Immediate	ori rd, rs1, imm
AND	and rd, rs1, rs2
AND Immediate	andi rd, rs1, imm

2.6 RISC-V比较指令

Instructions Name	RISC-V
Set <	slt rd, rs1, rs2
Set < Immediate	slti rd, rs1, imm
Set < Unsigned	sltu rd, rs1, rs2
Set < Imm Unsigned	sltiu rd, rs1, imm



slt rd, rs1, rs2 $x[rd] = (x[rs1] <_s x[rs2])$

小于则置位*(Set if Less Than)*. R-type, RV32I and RV64I.

比较 x[*rs1*]和 x[*rs2*]中的数，如果 x[*rs1*]更小，向 x[*rd*]写入 1，否则写入 0。

sltu rd, rs1, rs2 $x[rd] = (x[rs1] <_u x[rs2])$

无符号小于则置位*(Set if Less Than, Unsigned)*. R-type, RV32I and RV64I.

比较 x[*rs1*]和 x[*rs2*]，比较时视为无符号数。如果 x[*rs1*]更小，向 x[*rd*]写入 1，否则写入 0。

2.7 RISC-V分支指令

Instructions Name	RISC-V
Branch =	beq rs1, rs2, imm
Branch ≠	bne rs1, rs2, imm
Branch <	blt rs1, rs2, imm
Branch ≧	bge rs1, rs2, imm
Branch < Unsigned	bltu rs1, rs2, imm
Branch ≧ Unsigned	bgeu rs1, rs2, imm

beq rs1, rs2, offset　　　　if (rs1 == rs2) pc += sext(offset)

相等时分支 *(Branch if Equal)*. B-type, RV32I and RV64I.

若寄存器 x[*rs1*]和寄存器 x[*rs2*]的值相等，把 *pc* 的值设为当前值加上符号位扩展的偏移 *offset*。

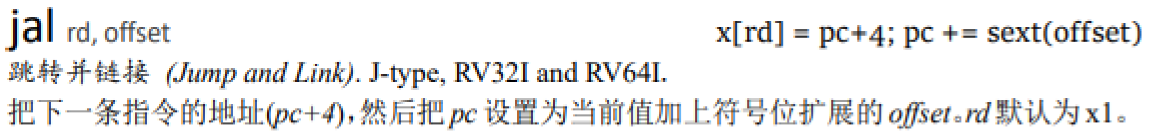
bltu rs1, rs2, offset　　　　if (rs1 $<_u$ rs2) pc += sext(offset)

无符号小于时分支 *(Branch if Less Than, Unsigned)*. B-type, RV32I and RV64I.

若寄存器 x[*rs1*]的值小于寄存器 x[*rs2*]的值（均视为无符号数），把 *pc* 的值设为当前值加上符号位扩展的偏移 *offset*。

2.8 RISC-V跳转指令

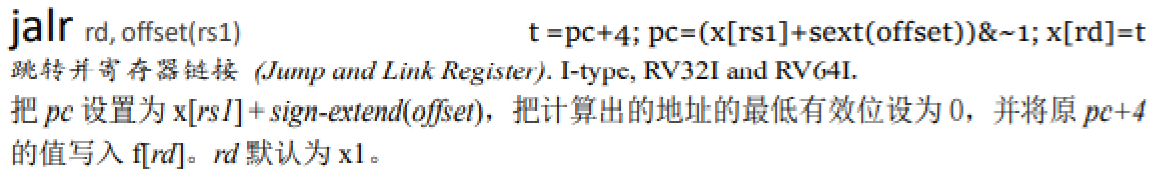
Instructions Name	RISC-V
Jump and Link	jal rd, imm
Jump and Link Register	jalr rd, rs1, imm



jal rd, offset   x[rd] = pc+4; pc += sext(offset)

跳转并链接 *(Jump and Link)*. J-type, RV32I and RV64I.

把下一条指令的地址(*pc*+*4*)，然后把 *pc* 设置为当前值加上符号位扩展的 *offset*。*rd* 默认为 x1。



jalr rd, offset(rs1)   t =pc+4; pc=(x[rs1]+sext(offset))&~1; x[rd]=t

跳转并寄存器链接 *(Jump and Link Register)*. I-type, RV32I and RV64I.

把 *pc* 设置为 x[*rs1*]+*sign-extend*(*offset*)，把计算出的地址的最低有效位设为 0，并将原 *pc*+*4* 的值写入 f[*rd*]。*rd* 默认为 x1。

2.9 RISC-V伪指令

- 伪指令不是真实存在的RISC-V指令，它只是为了方便程序员理解
- 伪指令会被编译器翻译成真实的RISC-V指令
  - 伪指令move
    例子：　　　　　mv　dst, reg1
    基础指令：　　　addi　dst, reg1, 0
  - 伪指令no operation
    例子：　　　　　nop
    基础指令：　　　addi x0, x0, 0
  - 伪指令load address
    例子：　　　　　la dst, label
    基础指令：　　　auipc dst, <offset to label>
  - 伪指令jump
    例子：　　　　　j offset
    基础指令：　　　jal x0, offset
- 更多伪指令可以看《RISC-V手册》第44、45页

2.10 一个简单的例子

```
# Fibonacci Sequence
main:
        add   t0, x0, x0
        addi  t1, x0, 1
        la    t3, n
        lw    t3, 0(t3)
fib:
        beq   t3, x0, finish
        add   t2, t1, t0
        mv    t0, t1
        mv    t1, t2
        addi  t3, t3, -1
        j     fib
finish:
        addi  a0, x0, 1
        addi  a1, t0, 0
        ecall   # print integer ecall
        addi  a0, x0, 10
        ecall   # terminate ecall
```

注释使用#符号

2.10 一个简单的例子

```
# Fibonacci Sequence
main:
        add   t0, x0, x0
        addi  t1, x0, 1
        la    t3, n
        lw    t3, 0(t3)
fib:
        beq   t3, x0, finish
        add   t2, t1, t0
        mv    t0, t1
        mv    t1, t2
        addi  t3, t3, -1
        j     fib
finish:
        addi  a0, x0, 1
        addi  a1, t0, 0
        ecall   # print integer ecall
        addi  a0, x0, 10
        ecall   # terminate ecall
```

标签用于标记代码段

2.10 一个简单的例子

a[n] = a[n-1] + a[n-2], n>=2, a[0]=0, a[1]=1

```
# Fibonacci Sequence
main:
        add   t0, x0, x0
        addi  t1, x0, 1
        la    t3, n
        lw    t3, 0(t3)
fib:
        beq   t3, x0, finish
        add   t2, t1, t0
        mv    t0, t1
        mv    t1, t2
        addi  t3, t3, -1
        j     fib
finish:
        addi  a0, x0, 1
        addi  a1, t0, 0
        ecall  # print integer ecall
        addi  a0, x0, 10
        ecall  # terminate ecall
```

```
t0=x0+x0          ===>  t0=0;
t1=x0+1           ===>  t1=1;
t3=address_of(n)  ===>  此时t3指向n变量的地址;
t3=n              ===>  t3=n;
```

```
c如果t3为0，说明已经将a[n] 计算出来，可以结束循环
t2=t1+t0          ===>  斐波那契计算公式
t0=t1             ===>  更新t0
t1=t2             ===>  更新t1
t3=t3-1           ===>  计数器减1
开始下一次循环
```

```
a0=x0+1           ===>  a0=1
a1=t0+0           ===>  a1=t0
调用printf函数
a0=x0+10          ===>  a0=10
调用exit函数
```

2.11 作业

- 用RISC-V汇编语言实现一个从1加到100的程序

Thank You!

3 RISC-V工具链的使用

潘志铭

3.1 GCC工具链介绍

- 命名方式：arch[-vendor][-os][-(gnu)eabi]-gcc
  - arm-none-eabi-gcc
  - arm-none-linux-gnueabi-gcc
  - arm-linux-gnueabi-gcc
  - arm-linux-gnueabihf-gcc
  - riscv64-unknown-linux-gnu-gcc
  - riscv32-unknown-linux-gnu-gcc
  - riscv64-unknown-elf-gcc
  - riscv32-unknown-elf-gcc

ARM architecture, no vendor, not target an operating system, complies with the ARM EABI

用于编译 ARM 架构的裸机系统（包括 ARM Linux 的 boot、kernel，不适用编译 Linux 应用 Application），一般适合 ARM7、Cortex-M 和 Cortex-R 内核的芯片使用，所以不支持那些跟操作系统关系密切的函数（比如fork），该工具链使用的是 newlib 库

3.1 GCC工具链介绍

- 命名方式：arch[-vendor][-os][-(gnu)eabi]-gcc
  - arm-none-eabi-gcc
  - arm-none-linux-gnueabi-gcc
  - arm-linux-gnueabi-gcc
  - arm-linux-gnueabihf-gcc
  - riscv64-unknown-linux-gnu-gcc
  - riscv32-unknown-linux-gnu-gcc
  - riscv64-unknown-elf-gcc
  - riscv32-unknown-elf-gcc

ARM architecture, no vendor, creates binaries that run on the Linux operating system, and uses the GNU EABI

主要用于基于ARM架构的Linux系统，可用于编译 ARM 架构的 u-boot、Linux内核、linux应用等。该工具链使用的是glibc库，一般ARM9、ARM11、Cortex-A 内核，带有 Linux 操作系统的会用到。

3.1 GCC工具链介绍

- 命名方式：arch[-vendor][-os][-(gnu)eabi]-gcc
  - arm-none-eabi-gcc
  - arm-none-linux-gnueabi-gcc
  - arm-linux-gnueabi-gcc
  - arm-linux-gnueabihf-gcc
  - riscv64-unknown-linux-gnu-gcc
  - riscv32-unknown-linux-gnu-gcc
  - riscv64-unknown-elf-gcc
  - riscv32-unknown-elf-gcc

两者区别在于选项-mfloat-abi的默认值不同，前者默认的选项是-mfloat-abi=softfp，也就是用fpu计算，但是传参数用普通寄存器传，这样中断的时候，只需要保存普通寄存器，中断负荷小，但是参数需要转换成浮点的再计算。后者的默认选项是-mfloat-abi-hard，也就是用fpu计算，传参数也用fpu中的浮点寄存器传，省去了转换，性能最好，但是中断负荷高。

深圳大学 SHENZHEN UNIVERSITY

3.1 GCC工具链介绍

- 命名方式：arch[-vendor][-os][-(gnu)eabi]-gcc
  - arm-none-eabi-gcc
  - arm-none-linux-gnueabi-gcc
  - arm-linux-gnueabi-gcc
  - arm-linux-gnueabihf-gcc
  - riscv64-unknown-linux-gnu-gcc
  - riscv32-unknown-linux-gnu-gcc
  - riscv64-unknown-elf-gcc
  - riscv32-unknown-elf-gcc

"riscv64-unknown-linux-gnu-"前缀表示该工具链是64位RISC-V架构的Linux版本工具链。这里的linux指的是该工具链会使用glibc库。

同理，" riscv32-unknown-linux-gnu-"前缀则是指32位RISC-V架构的Linux版本工具链。

前缀riscv64（还有riscv32的版本）与运行在64位或者32位电脑上毫无关系，此处的64和32是指如果没有通过-march和-mabi选项指定RISC-V架构的位宽，默认将会按照64位还是32位的RISC-V架构来编译程序。有关-march和-mabi选项的含义会在后面描述。

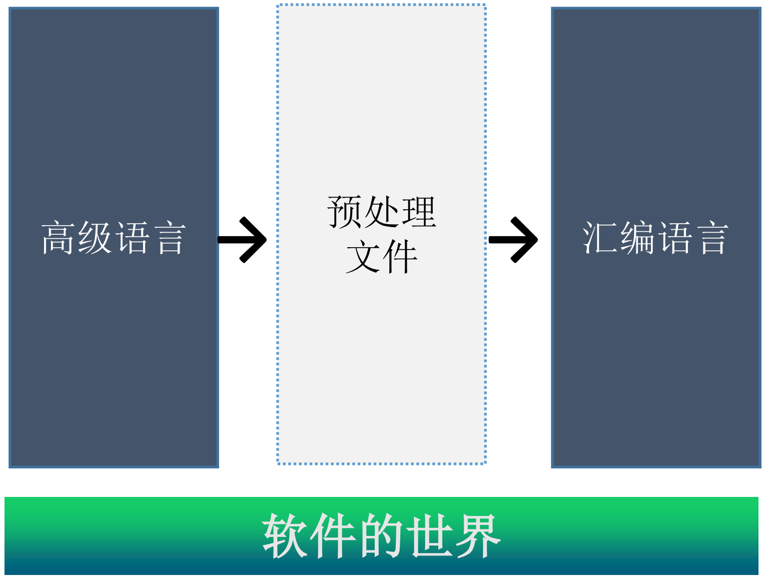
3.1 GCC工具链介绍

- 命名方式：arch[-vendor][-os][-(gnu)eabi]-gcc
  - arm-none-eabi-gcc
  - arm-none-linux-gnueabi-gcc
  - arm-linux-gnueabi-gcc
  - arm-linux-gnueabihf-gcc
  - riscv64-unknown-linux-gnu-gcc
  - riscv32-unknown-linux-gnu-gcc
  - riscv64-unknown-elf-gcc
  - riscv32-unknown-elf-gcc

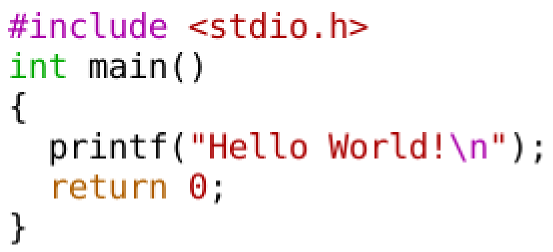
以”riscv64-unknown-elf-”/”riscv32-unknown-elf” 为前缀则表示该工具链为非Linux(Non-linux)版本的工具链。Non-Linux不是指当前版本工具链一定不能运行在Linux操作系统的电脑上，Non-Linux是指该GCC工具链会使用newlib作为C运行库。

3.1 GCC工具链介绍

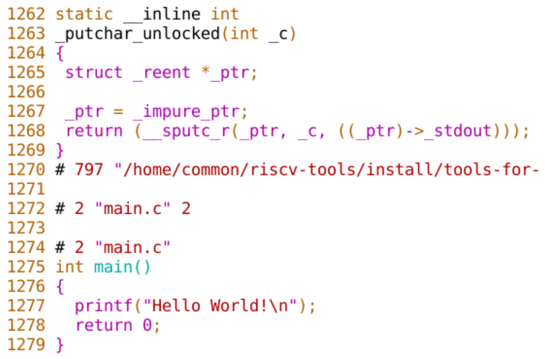
- 预处理 $ riscv64-unknown-elf-gcc -E main.c -o main.i



预处理器（cpp）根据以字符#号开头的命令，修改原始的c程序。比如main.c中的第一行#include <stdio.h>命令告诉预处理器读取系统头文件stdio.h的内容，并把它直接插入到程序文本中，结果得到了另一个C程序，通常是以.i为扩展名。



```
#include <stdio.h>
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

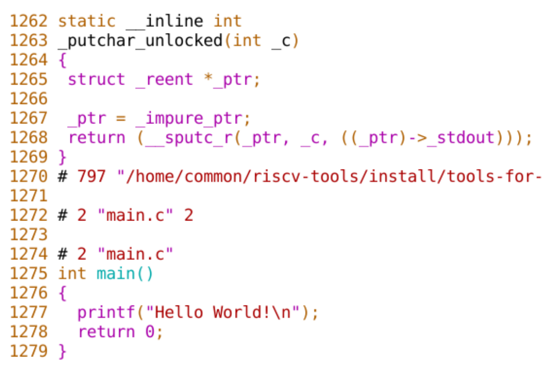


```
static __inline int
_putchar_unlocked(int _c)
{
 struct _reent *_ptr;

 _ptr = _impure_ptr;
 return (__sputc_r(_ptr, _c, ((_ptr)->_stdout)));
}
# 797 "/home/common/riscv-tools/install/tools-for-

# 2 "main.c" 2

# 2 "main.c"
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

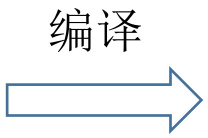
3.1 GCC工具链介绍

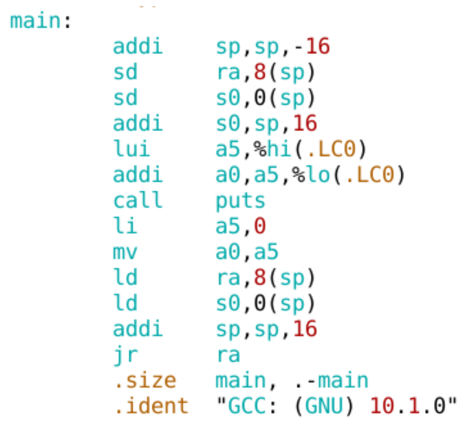
- 编译 $ riscv64-unknown-elf-gcc -S main.i -o main.s

  编译器将文本文件main.i翻译成文本文件main.s，它包含一个汇编语言程序。汇编语言中每条语句都以一种标准的文本格式确切地描述了一条低级机器语言指令。其实汇编语言是非常有用的，它为所有的高级语言提供了一种通用的输出语言。比如C编译器和Fortran编译器产生的输出文件用的都是一样的汇编语言。



```
static __inline int
_putchar_unlocked(int _c)
{
 struct _reent *_ptr;

 _ptr = _impure_ptr;
 return (__sputc_r(_ptr, _c, ((_ptr)->_stdout)));
}
# 797 "/home/common/riscv-tools/install/tools-for-

# 2 "main.c" 2

# 2 "main.c"
int main()
{
  printf("Hello World!\n");
  return 0;
}
```



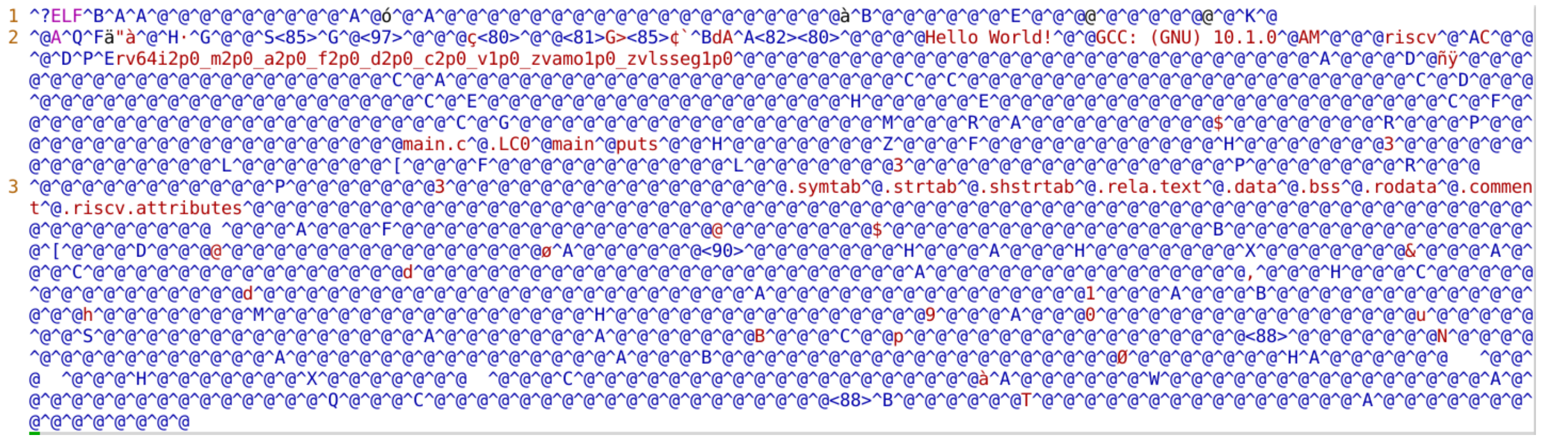


```
main:
        addi    sp,sp,-16
        sd      ra,8(sp)
        sd      s0,0(sp)
        addi    s0,sp,16
        lui     a5,%hi(.LC0)
        addi    a0,a5,%lo(.LC0)
        call    puts
        li      a5,0
        mv      a0,a5
        ld      ra,8(sp)
        ld      s0,0(sp)
        addi    sp,sp,16
        jr      ra
        .size   main, .-main
        .ident  "GCC: (GNU) 10.1.0"
```

3.1 GCC工具链介绍

- 汇编　　$ riscv64-unknown-elf-gcc -c main.s -o main.o

  汇编器（as）将main.s翻译成机器语言指令，把这些指令打包成一种叫做可定位目标程序的格式，并将结果保存在目标文件main.o中，main.o是一个二进制文件，它的字节编码是“机器语言指令”而不是“字符”，所以，我们用文本编辑器打开main.o文件看到是会是一堆乱码。

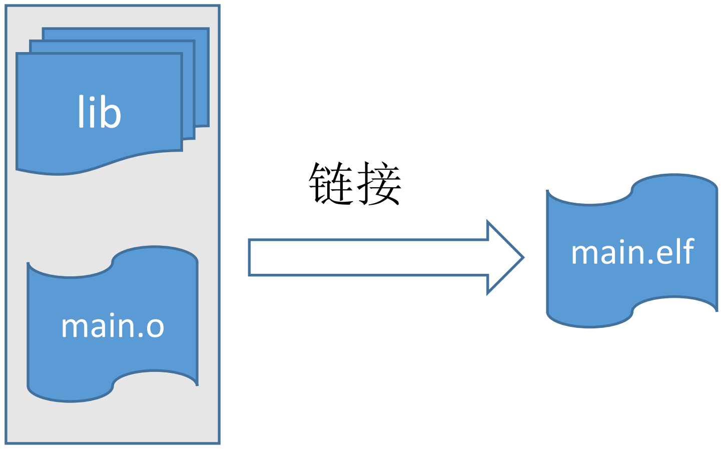


```
^?ELF^B^A^A^@^@^@^@^@^@^@^@^@^A^@ó^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@à^B^@^@^@^@^@^E^@^@^@@^@^@^@^@^@@^@^K^@
^@A^Q^Fä"à^@^H·^G^@^@^S<85>^G^@<97>^@^@^@ç<80>^@^@<81>G><85>¢`^BdA^A<82><80>^@^@^@^@Hello World!^@^@GCC: (GNU) 10.1.0^@AM^@^@^@riscv^@^AC^@^@
^@^D^P^Erv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0_v1p0_zvamo1p0_zvlsseg1p0^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^D^@ñÿ^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^D^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^@^@^E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^F^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^G^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^M^@^@^@^R^@^A^@^@^@^@^@^@^@^@$^@^@^@^@^@^@^R^@^@^@^P^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@main.c^@.LC0^@main^@puts^@^@^H^@^@^@^@^@^@^@^Z^@^@^@^F^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^@^@^@3^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^L^@^@^@^@^@^@^@^[^@^@^@^F^@^@^@^@^@^@^@^@^@^@^L^@^@^@^@^@^@3^@^@^@^@^@^@^@^@^@^@^@^@^@^P^@^@^@^@^@^@^R^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^P^@^@^@^@^@^@3^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@.symtab^@.strtab^@.shstrtab^@.rela.text^@.data^@.bss^@.rodata^@.commen
t^@.riscv.attributes^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@ ^@^@^@^A^@^@^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@@^@^@^@^@^@^@$^@^@^@^@^@^@^@^@^@^@^@^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^[^@^@^@^D^@^@^@@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ø^A^@^@^@^@^@^@<90>^@^@^@^@^@^@^H^@^@^@^A^@^@^@^H^@^@^@^@^@^@^X^@^@^@^@^@^@&^@^@^@^A^@^
@^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@d^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@,^@^@^@^H^@^@^@^C^@^@^@^@
^@^@^@^@^@^@^@^@^@d^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@1^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^
@^@^@h^@^@^@^@^@^@^@^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@9^@^@^@^A^@^@^@0^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@u^@^@^@^@
^@^@^S^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@B^@^@^@^C^@^@p^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@<88>^@^@^@^@^@^@^@N^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@Ø^@^@^@^@^@^@^@^H^A^@^@^@^@^@^@   ^@^@^
@  ^@^@^@^H^@^@^@^@^@^@^@^X^@^@^@^@^@^@^@  ^@^@^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@à^A^@^@^@^@^@^@^W^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^Q^@^@^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@<88>^B^@^@^@^@^@^@T^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^
@^@^@^@^@^@^@
```

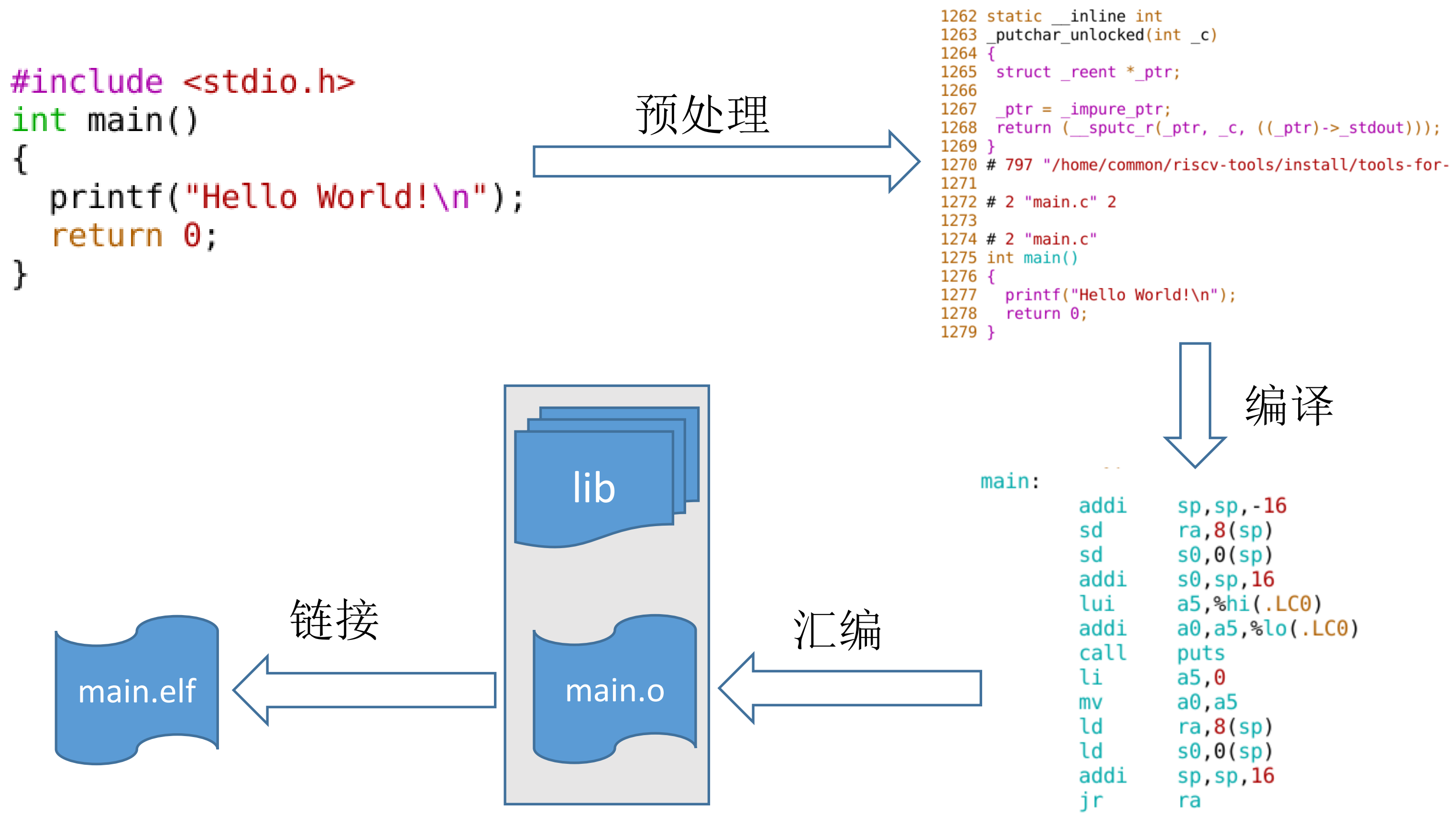
3.1 GCC工具链介绍

- 链接　　`$ riscv64-unknown-elf-gcc main.o -o main.elf`

我们注意到，main.c中有一个printf函数，它是每个C编译器都会提供的标准库中的一个函数。printf函数存在于一个名为printf.o的单独的预编译好的目标文件中，而这个文件必须以某种方式合并到我们的main.o程序中。链接器（ld）就是负责处理这种合并。最后得到main文件，一个可执行目标文件（可执行文件），可被加载到内存中，由系统执行。



3.1 GCC工具链介绍



```c
#include <stdio.h>
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

```c
static __inline int
_putchar_unlocked(int _c)
{
 struct _reent *_ptr;

 _ptr = _impure_ptr;
 return (__sputc_r(_ptr, _c, ((_ptr)->_stdout)));
}
# 797 "/home/common/riscv-tools/install/tools-for-

# 2 "main.c" 2

# 2 "main.c"
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

```
main:
        addi    sp,sp,-16
        sd      ra,8(sp)
        sd      s0,0(sp)
        addi    s0,sp,16
        lui     a5,%hi(.LC0)
        addi    a0,a5,%lo(.LC0)
        call    puts
        li      a5,0
        mv      a0,a5
        ld      ra,8(sp)
        ld      s0,0(sp)
        addi    sp,sp,16
        jr      ra
        .size   main, .-main
        .ident  "GCC: (GNU) 10.1.0"
```

上面的操作可以通过一个命令实现：
$ riscv64-unknown-elf-gcc main.c -o main.elf

如果出现`riscv64-unknown-elf-gcc: command not found`的错误，需要将工具链路径加入PATH中：
$ export PATH=/opt/riscv:$PATH

3.2 RISC-V GCC Options

- -march=*ISA-string*

由于RISC-V的指令集是模块化的指令集，因此在为目标RISC-V平台进行交叉编译之时，需要通过选项指定目标RISC-V平台所支持的模块化指令集组合，该选项为(-march=)，有效的选项值如下：

  - rv32i[m][a][f[d]][c]
  - rv32g[c]
  - rv64i[m][a][f[d]][c]
  - rv64g[c]

注意：在上述选项中rv32表示目标平台是32位架构，rv64表示目标平台是64位架构，其他i/m/a/f/d/c/g分别代表了RISC-V模块化指令子集的字母简称。

3.2 RISC-V GCC Options

- -mabi=*ABI-string*

由于RISC-V的指令集是模块化的指令集，因此在为目标RISC-V平台进行交叉编译之时，需要通过选项指定RISC-V目标平台所支持的ABI函数调用规则。RISC-V定义了两种整数的ABI调用规则和三种浮点ABI调用规则，通过选项(-mabi=)指明，有效的选项值如下：

- ilp32
- ilp32f
- ilp32d
- lp64
- lp64f
- lp64d

前缀ilp32表示目标平台是32位架构，在此架构下，C语言的“int”和“long”变量长度为32比特，“long long”变量为64位；前缀lp64表示目标平台是64位架构，C语言的“int”变量长度为32比特，而“long”变量长度为64比特。

3.2 RISC-V GCC Options

- 查看合法支持的march/mabi组合：

```
$ riscv64-unknown-elf-gcc --print-multi-lib
.;
rv32e/ilp32e;@march=rv32e@mabi=ilp32e
rv32ec/ilp32e;@march=rv32ec@mabi=ilp32e
rv32em/ilp32e;@march=rv32em@mabi=ilp32e
rv32emc/ilp32e;@march=rv32emc@mabi=ilp32e
rv32ema/ilp32e;@march=rv32ema@mabi=ilp32e
rv32emac/ilp32e;@march=rv32emac@mabi=ilp32e
rv32i/ilp32;@march=rv32i@mabi=ilp32
rv32ic/ilp32;@march=rv32ic@mabi=ilp32
rv32im/ilp32;@march=rv32im@mabi=ilp32
rv32imc/ilp32;@march=rv32imc@mabi=ilp32
rv32iac/ilp32;@march=rv32iac@mabi=ilp32
rv32imac/ilp32;@march=rv32imac@mabi=ilp32
rv32imfc/ilp32f;@march=rv32imfc@mabi=ilp32f
rv32imafc/ilp32f;@march=rv32imafc@mabi=ilp32f
rv32imafdc/ilp32d;@march=rv32imafdc@mabi=ilp32d
rv64imac/lp64;@march=rv64imac@mabi=lp64
rv64imafdc/lp64d;@march=rv64imafdc@mabi=lp64d
```

3.2 RISC-V GCC Options

- -cmodel=medlow

medlow选项用于指示该程序的寻址范围固定只能在-2GB至+2GB的空间内。注意：地址区间没有负数可言，-2GB是指整个64位地址空间最高2GB地址区间。由于此模式的寻址空间是固定的-2GB至+2GB的空间内，因此编译器能够相对生成比较高效的代码，但是由于寻址空间固定，对于整个64位的大多数地址空间都无法访问到，用户需小心使用。

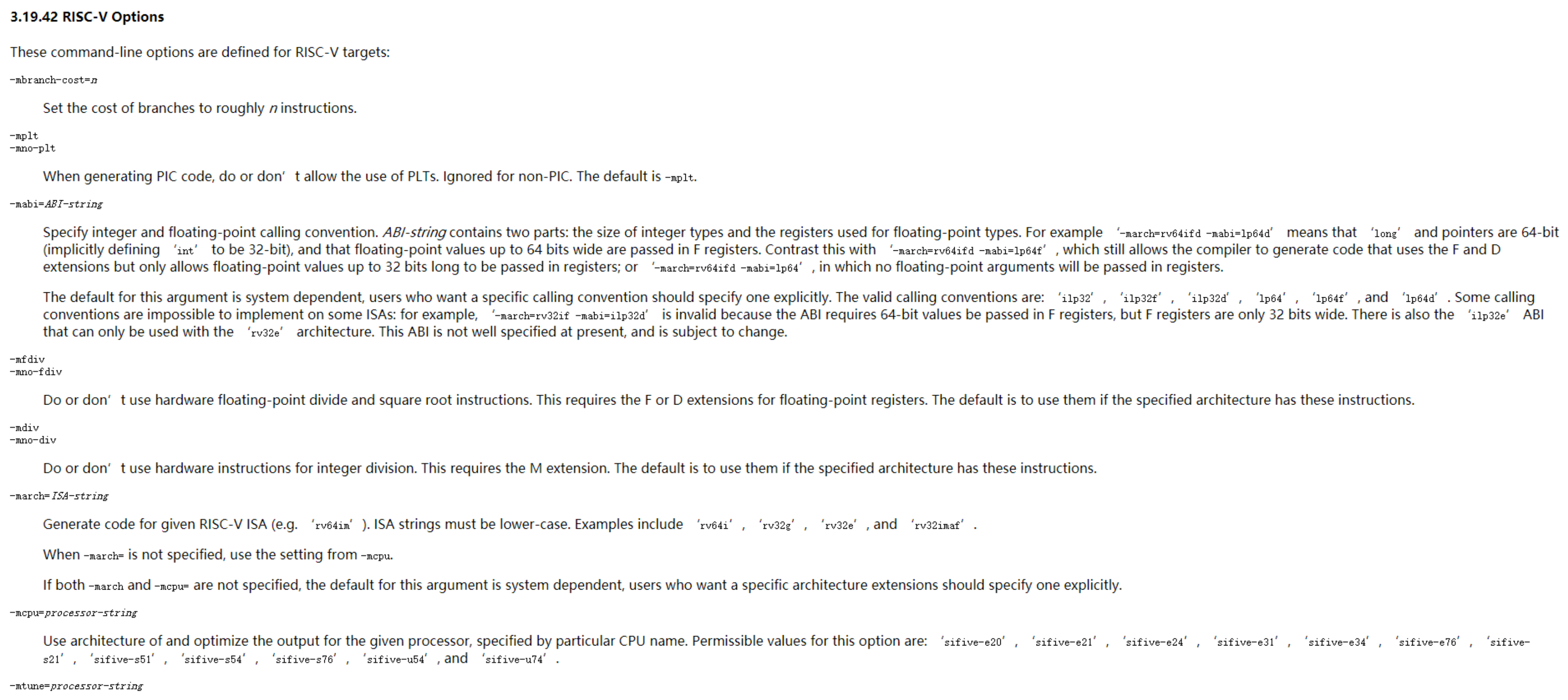
- -cmodel=medany

medlow选项用于指示该程序的寻址范围可以在任意的一个4G空间内。由于此模式的寻址空间不是固定的，所以相对比较灵活。

3.2 RISC-V GCC Options

- 其他RISC-V GCC选项

  https://gcc.gnu.org/onlinedocs/gcc/RISC-V-Options.html



3.19.42 RISC-V Options

These command-line options are defined for RISC-V targets:

`-mbranch-cost=n`

Set the cost of branches to roughly *n* instructions.

`-mplt`
`-mno-plt`

When generating PIC code, do or don't allow the use of PLTs. Ignored for non-PIC. The default is `-mplt`.

`-mabi=ABI-string`

Specify integer and floating-point calling convention. *ABI-string* contains two parts: the size of integer types and the registers used for floating-point types. For example '`-march=rv64ifd -mabi=lp64d`' means that '`long`' and pointers are 64-bit (implicitly defining '`int`' to be 32-bit), and that floating-point values up to 64 bits wide are passed in F registers. Contrast this with '`-march=rv64ifd -mabi=lp64f`', which still allows the compiler to generate code that uses the F and D extensions but only allows floating-point values up to 32 bits long to be passed in registers; or '`-march=rv64ifd -mabi=lp64`', in which no floating-point arguments will be passed in registers.

The default for this argument is system dependent, users who want a specific calling convention should specify one explicitly. The valid calling conventions are: '`ilp32`', '`ilp32f`', '`ilp32d`', '`lp64`', '`lp64f`', and '`lp64d`'. Some calling conventions are impossible to implement on some ISAs: for example, '`-march=rv32if -mabi=ilp32d`' is invalid because the ABI requires 64-bit values be passed in F registers, but F registers are only 32 bits wide. There is also the '`ilp32e`' ABI that can only be used with the '`rv32e`' architecture. This ABI is not well specified at present, and is subject to change.

`-mfdiv`
`-mno-fdiv`

Do or don't use hardware floating-point divide and square root instructions. This requires the F or D extensions for floating-point registers. The default is to use them if the specified architecture has these instructions.

`-mdiv`
`-mno-div`

Do or don't use hardware instructions for integer division. This requires the M extension. The default is to use them if the specified architecture has these instructions.

`-march=ISA-string`

Generate code for given RISC-V ISA (e.g. '`rv64im`'). ISA strings must be lower-case. Examples include '`rv64i`', '`rv32g`', '`rv32e`', and '`rv32imaf`'.

When `-march=` is not specified, use the setting from `-mcpu`.

If both `-march` and `-mcpu=` are not specified, the default for this argument is system dependent, users who want a specific architecture extensions should specify one explicitly.

`-mcpu=processor-string`

Use architecture of and optimize the output for the given processor, specified by particular CPU name. Permissible values for this option are: '`sifive-e20`', '`sifive-e21`', '`sifive-e24`', '`sifive-e31`', '`sifive-e34`', '`sifive-e76`', '`sifive-s21`', '`sifive-s51`', '`sifive-s54`', '`sifive-s76`', '`sifive-u54`', and '`sifive-u74`'.

`-mtune=processor-string`

3.3 RISC-V Binutils

• riscv64-unknown-elf-gdb	调试器
• riscv64-unknown-elf-as	汇编器
• riscv64-unknown-elf-ld	链接器
• riscv64-unknown-elf-ar	用于打包静态库
• riscv64-unknown-elf-objdump	反汇编器
• riscv64-unknown-elf-readelf	查看有关ELF文件的信息
• riscv64-unknown-elf-size	查看有关ELF文件大小信息
• riscv64-unknown-elf-objcopy	将ELF文件转成另外一种格式

3.4 C运行库

为了解释C运行库，需要先回忆一下C语言标准。C语言标准主要由两部分组成：一部分描述C的语法，另一部分描述C标准库。C标准库定义了一组标准头文件，每个头文件中包含一些相关的函数、变量、类型声明和宏定义，譬如常见的printf函数便是一个C标准库函数，其原型定义在stdio头文件中。

C语言标准仅仅定义了C标准库函数原型，并没有提供实现。因此，C语言编译器通常需要一个C运行时库（C Run Time Libray，CRT）的支持。C运行时库又常简称为C运行库。与C语言类似，C++也定义了自己的标准，同时提供相关支持库，称为C++运行时库。

3.4 C运行库

要在一个平台上支持C语言，不仅要实现C编译器，还要实现C标准库，这样的实现才能完全支持C标准。glibc（GNU C Library）是Linux下面C标准库的实现，其要点如下：

- glibc本身是GNU旗下的C标准库，后来逐渐成为了Linux的标准C库。glibc的主体分布在Linux系统的/lib与/usr/lib目录中，包括 libc 标准 C 函式库、libm数学函式库等等，都以.so做结尾；
- Linux系统通常将libc库作为操作系统的一部分，它被视为操作系统与用户程序的接口。譬如：glibc不仅实现标准C语言中的函数，还封装了操作系统提供的系统服务，即系统调用的封装；
- 对于C++语言，常用的C++标准库为libstdc++；

newlib是一个面向嵌入式系统的C运行库。相对于glibc，newlib实现了大部分的功能函数，但体积却小很多。newlib独特的体系结构将功能实现与具体的操作系统分层，使之能够很好地进行配置以满足嵌入式系统的要求。由于专为嵌入式系统设计，newlib具有可移植性强、轻量级、速度快、功能完备等特点，已广泛应用于各种嵌入式系统中。

3.5 作业

- 查看你的RISC-V工具链支持哪些march/mabi组合
- 利用不同组合的march/mabi选项编译helloworld
- 使用反汇编器objdump生成反汇编文件

Thank You!

4 RISC-V工具链的开发

潘志铭

4.1 为什么要进行工具链开发？

- RISC-V指令集的特点是什么？
  - 开源；模块化；可扩展性强

- 什么情况下需要设计自定义指令？
  - 特定应用场景：如PS5游戏加速指令

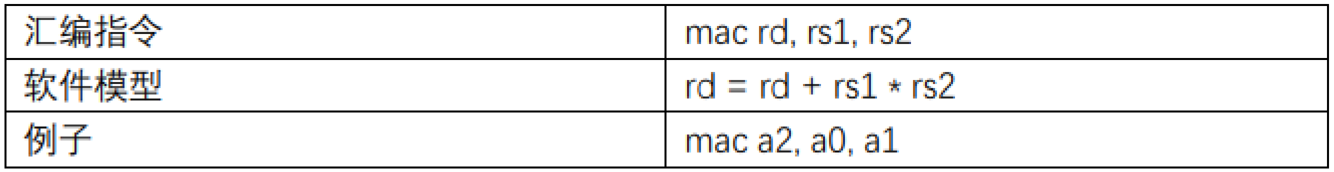
- 硬件实现自定义指令后，软件如何使用？
  - 修改GNU工具链，增加对自定义指令的支持

4.2 如何进行工具链开发？

- 1. 描述自定义指令
- 2. 确定指令码
- 3. 为binutils增加自定义指令
- 4. 为gcc增加自定义指令
- 5. 在spike上增加自定义指令
- 6. 重新构建toolchain和spike
- 7. 测试toolchain和spike

4.3 描述自定义指令

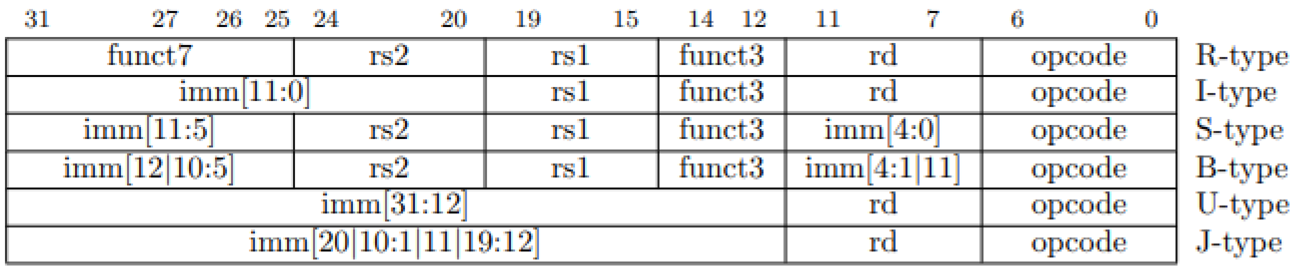
- 增加一条乘加指令mac，具体描述如下：



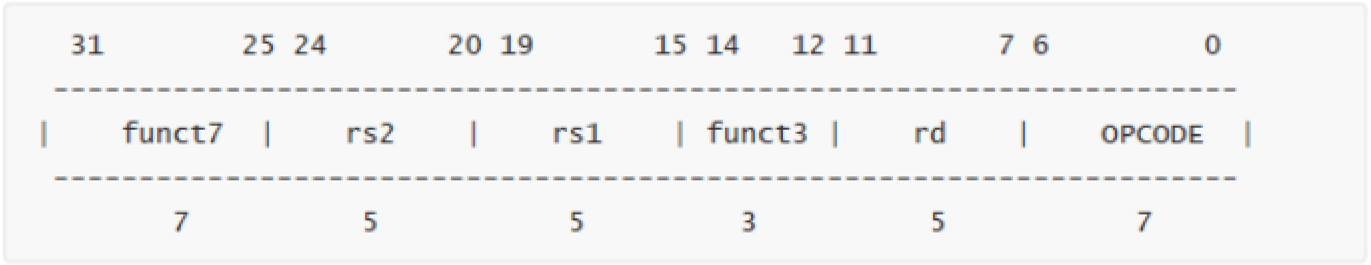
汇编指令	mac rd, rs1, rs2
软件模型	rd = rd + rs1 * rs2
例子	mac a2, a0, a1

4.4 确定指令码

- 依据 RISC-V SPEC， mac 属于 R-Type 指令



31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

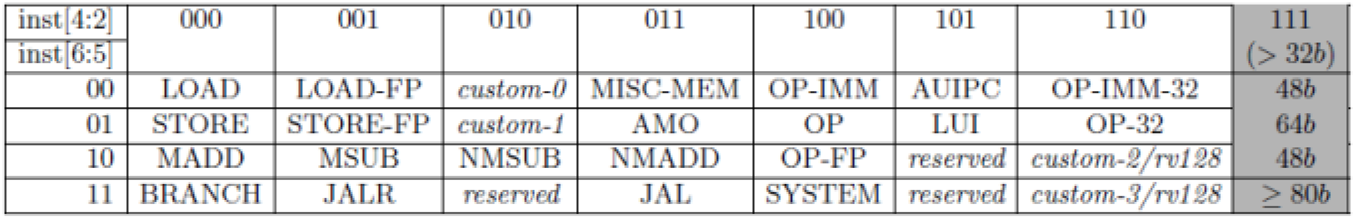


```
  31          25 24         20 19         15 14     12 11         7 6          0
 ---------------------------------------------------------------------------------
|    funct7    |     rs2     |     rs1     | funct3 |     rd     |    OPCODE    |
 ---------------------------------------------------------------------------------
        7             5             5           3          5             7
```

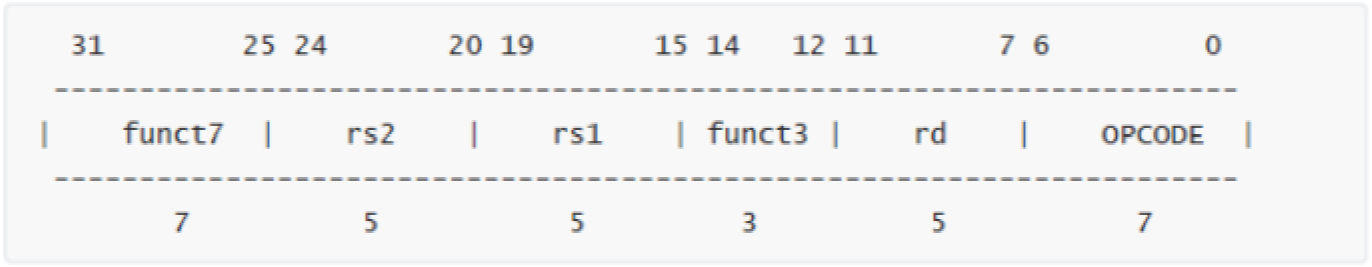
4.4 确定指令码

- 确定OPCODE

RISC-V SPEC规定了四种自定义指令的OPCODE，分别是7’b000_1011，7’b010_1011，7’b101_1011，7’b111_1011，这里我们选用CUSTOM0编码7’b000_1011



inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 (> 32b)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	48b
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	≥ 80b

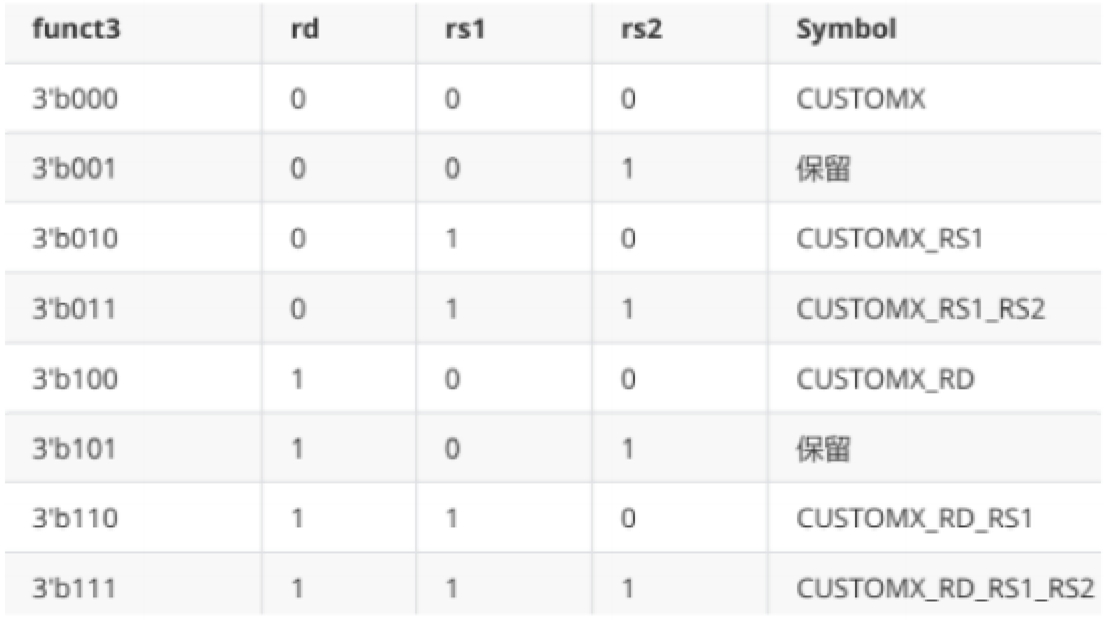


```
 31          25 24         20 19         15 14    12 11         7 6           0
----------------------------------------------------------------------
|     funct7   |     rs2     |     rs1     | funct3 |     rd     |    OPCODE   |
----------------------------------------------------------------------
       7              5             5           3          5            7
```

4.4 确定指令码

- 确定funct3

为了进一步规范自定义指令码格式，我们可以约定funct3的编码格式：3’b010表示只使用了RS1寄存器，3’b011表示只使用了RS1和RS2 寄存器，3’b100表示只使用了RD寄存器，3’b110表示只使用RD和RS1寄存器，而3’b111则表示RD、RS1和RS2寄存器都用到了，所以这里的funct3应该是3’b111.

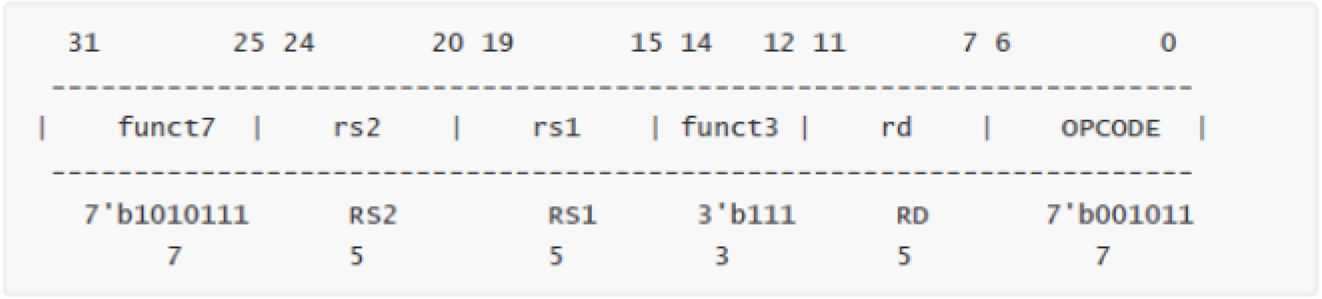


funct3	rd	rs1	rs2	Symbol
3'b000	0	0	0	CUSTOMX
3'b001	0	0	1	保留
3'b010	0	1	0	CUSTOMX_RS1
3'b011	0	1	1	CUSTOMX_RS1_RS2
3'b100	1	0	0	CUSTOMX_RD
3'b101	1	0	1	保留
3'b110	1	1	0	CUSTOMX_RD_RS1
3'b111	1	1	1	CUSTOMX_RD_RS1_RS2

4.4 确定指令码

- 最终确定的指令码

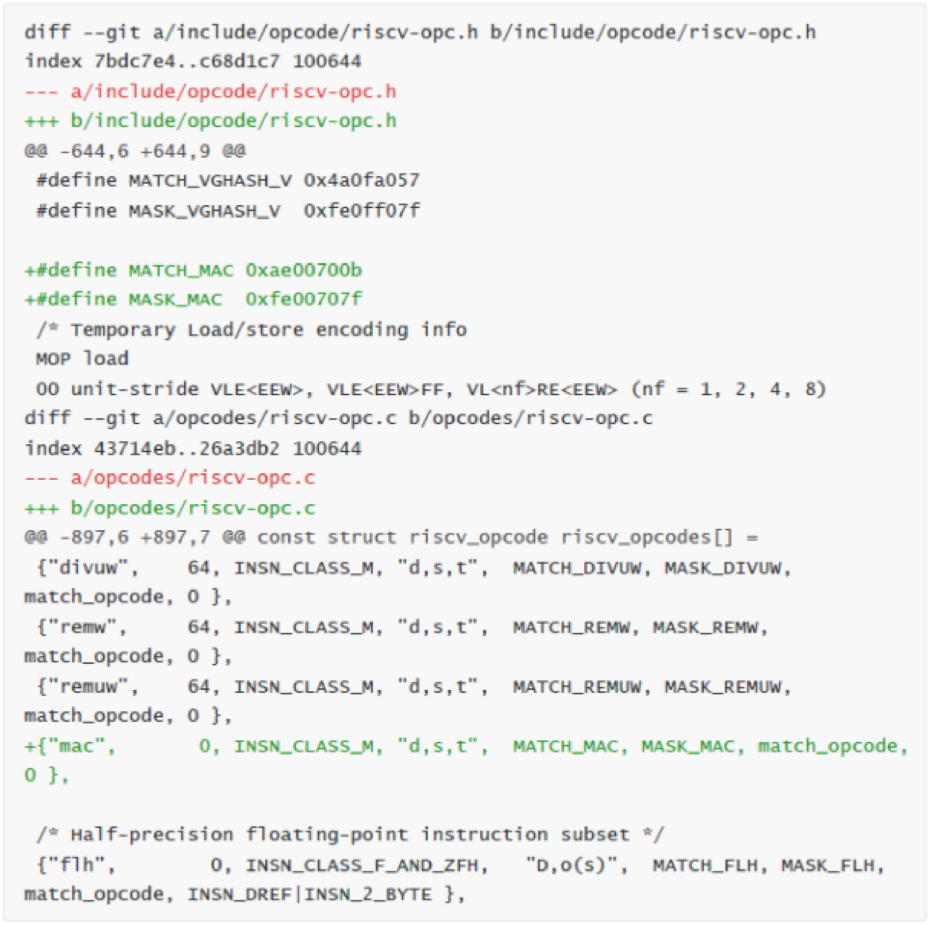
由于funct7没有特别的约束，我们这里选定为7’b101_0111.



```
 31         25 24        20 19        15 14    12 11       7 6         0
---------------------------------------------------------------------
|    funct7   |     rs2     |     rs1     | funct3 |    rd     |    OPCODE   |
---------------------------------------------------------------------
  7'b1010111       RS2          RS1       3'b111      RD       7'b001011
       7            5            5           3         5           7
```

4.5 为binutils增加自定义指令

- 修改riscv-opc.c文件和riscv-opc.h文件：



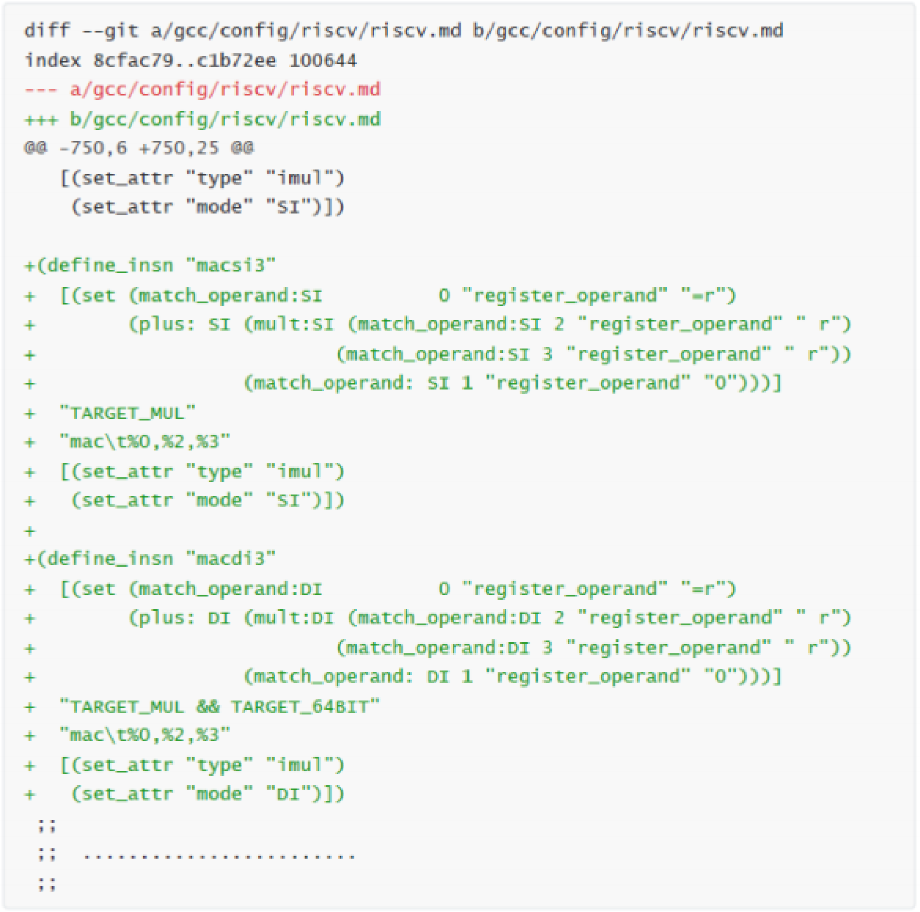
```
diff --git a/include/opcode/riscv-opc.h b/include/opcode/riscv-opc.h
index 7bdc7e4..c68d1c7 100644
--- a/include/opcode/riscv-opc.h
+++ b/include/opcode/riscv-opc.h
@@ -644,6 +644,9 @@
 #define MATCH_VGHASH_V 0x4a0fa057
 #define MASK_VGHASH_V  0xfe0ff07f

+#define MATCH_MAC 0xae00700b
+#define MASK_MAC  0xfe00707f
 /* Temporary Load/store encoding info
 MOP load
 00 unit-stride VLE<EEW>, VLE<EEW>FF, VL<nf>RE<EEW> (nf = 1, 2, 4, 8)
diff --git a/opcodes/riscv-opc.c b/opcodes/riscv-opc.c
index 43714eb..26a3db2 100644
--- a/opcodes/riscv-opc.c
+++ b/opcodes/riscv-opc.c
@@ -897,6 +897,7 @@ const struct riscv_opcode riscv_opcodes[] =
 {"divuw",    64, INSN_CLASS_M, "d,s,t",  MATCH_DIVUW, MASK_DIVUW,
match_opcode, 0 },
 {"remw",     64, INSN_CLASS_M, "d,s,t",  MATCH_REMW, MASK_REMW,
match_opcode, 0 },
 {"remuw",    64, INSN_CLASS_M, "d,s,t",  MATCH_REMUW, MASK_REMUW,
match_opcode, 0 },
+{"mac",       0, INSN_CLASS_M, "d,s,t",  MATCH_MAC, MASK_MAC, match_opcode,
0 },

 /* Half-precision floating-point instruction subset */
 {"flh",        0, INSN_CLASS_F_AND_ZFH,   "D,o(s)",  MATCH_FLH, MASK_FLH,
match_opcode, INSN_DREF|INSN_2_BYTE },
```

汇编器需要定义一些宏来确定指令编码，其中有两个宏是一定要定义的，分别是 MATCH 和 MASK，其中MATCH宏定义用来识别指令码，MASK宏定义用来获取指令码，在生成指令码时，汇编器会这样使用这些宏定义：

((insn ^ MATCH) & MASK) == 0.

4.6 为gcc增加自定义指令

- 修改riscv.md文件：

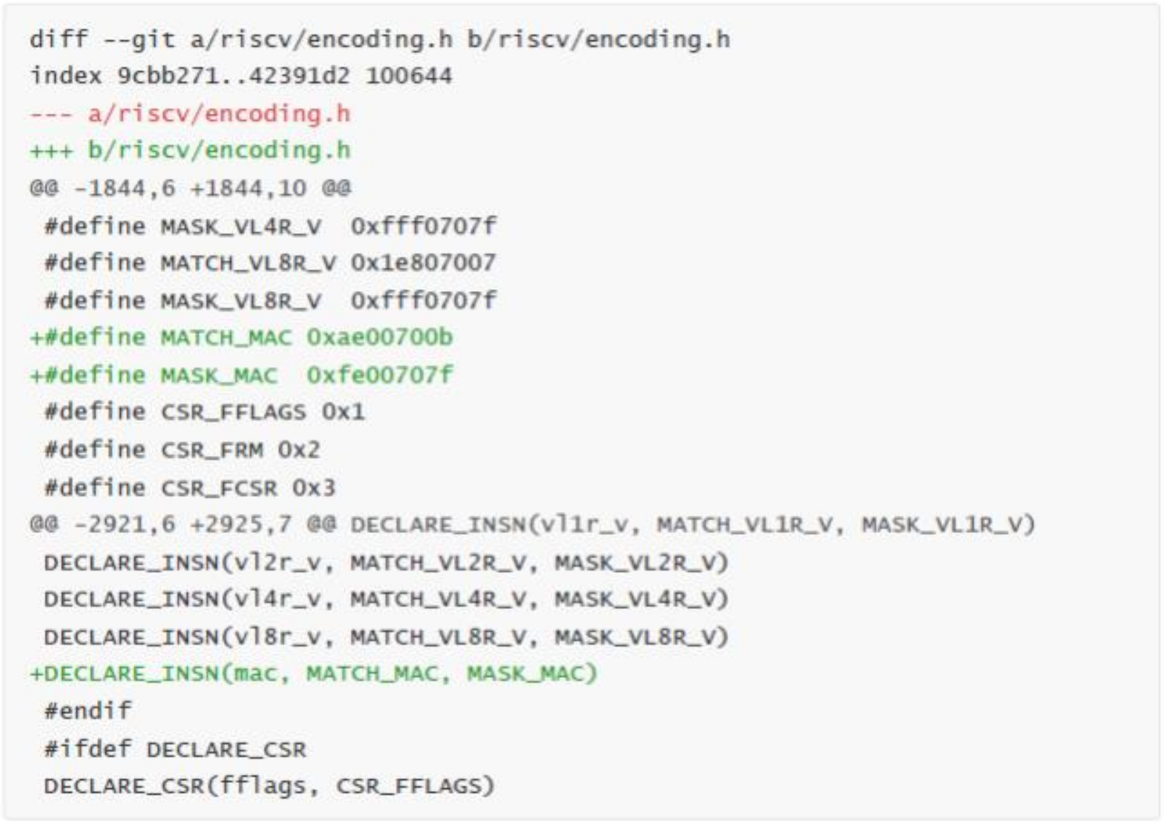


```
diff --git a/gcc/config/riscv/riscv.md b/gcc/config/riscv/riscv.md
index 8cfac79..c1b72ee 100644
--- a/gcc/config/riscv/riscv.md
+++ b/gcc/config/riscv/riscv.md
@@ -750,6 +750,25 @@
   [(set_attr "type" "imul")
    (set_attr "mode" "SI")])

+(define_insn "macsi3"
+  [(set (match_operand:SI          0 "register_operand" "=r")
+        (plus: SI (mult:SI (match_operand:SI 2 "register_operand" " r")
+                           (match_operand:SI 3 "register_operand" " r"))
+                  (match_operand: SI 1 "register_operand" "0")))]
+  "TARGET_MUL"
+  "mac\t%0,%2,%3"
+  [(set_attr "type" "imul")
+   (set_attr "mode" "SI")])
+
+(define_insn "macdi3"
+  [(set (match_operand:DI          0 "register_operand" "=r")
+        (plus: DI (mult:DI (match_operand:DI 2 "register_operand" " r")
+                           (match_operand:DI 3 "register_operand" " r"))
+                  (match_operand: DI 1 "register_operand" "0")))]
+  "TARGET_MUL && TARGET_64BIT"
+  "mac\t%0,%2,%3"
+  [(set_attr "type" "imul")
+   (set_attr "mode" "DI")])
 ;;
 ;;  ........................
 ;;
```

为了让 GCC 识别 mac 指令，我们需要在 GCC 后端添加指令模板，由于 mac 指令可以描述成乘、加两种指令的混合体，所以我们可以直接使用 GCC 中自带的 SPN 来描述指令模板（对于 AES、SHA 这种加密指令，是无法直接使用 GCC 中自带的 SPN 来描述的，只能用 intrinsic，然后通过 intrinsic 与 GCC 后端的对应指令模板做匹配），经过修改后的 riscv.md 文件如左图所示。

4.7 在spike上增加自定义指令

- 修改encoding.h文件



```
diff --git a/riscv/encoding.h b/riscv/encoding.h
index 9cbb271..42391d2 100644
--- a/riscv/encoding.h
+++ b/riscv/encoding.h
@@ -1844,6 +1844,10 @@
 #define MASK_VL4R_V  0xfff0707f
 #define MATCH_VL8R_V 0x1e807007
 #define MASK_VL8R_V  0xfff0707f
+#define MATCH_MAC 0xae00700b
+#define MASK_MAC  0xfe00707f
 #define CSR_FFLAGS 0x1
 #define CSR_FRM 0x2
 #define CSR_FCSR 0x3
@@ -2921,6 +2925,7 @@ DECLARE_INSN(vl1r_v, MATCH_VL1R_V, MASK_VL1R_V)
 DECLARE_INSN(vl2r_v, MATCH_VL2R_V, MASK_VL2R_V)
 DECLARE_INSN(vl4r_v, MATCH_VL4R_V, MASK_VL4R_V)
 DECLARE_INSN(vl8r_v, MATCH_VL8R_V, MASK_VL8R_V)
+DECLARE_INSN(mac, MATCH_MAC, MASK_MAC)
 #endif
 #ifdef DECLARE_CSR
 DECLARE_CSR(fflags, CSR_FFLAGS)
```

spike 是一款 RISC-V 指令模拟器，我们可以在 spike 上增加自定义指令，运行包含自定 义指令 mac 的应用程序，从而验证修改后的 GNU 工具链是否能够正常工作。

4.7 在spike上增加自定义指令

- 增加指令描述：
  - 在riscv/insns目录下新建文件mac.h，内容如下：

```
require_extension('M');
WRITE_RD(sext_xlen(RD + RS1 * RS2));
```

  - 修改riscv/riscv.mk.in文件：

```
diff --git a/riscv/riscv.mk.in b/riscv/riscv.mk.in
index d4422fe..aa6f68b 100644
--- a/riscv/riscv.mk.in
+++ b/riscv/riscv.mk.in
@@ -195,6 +195,7 @@ riscv_insn_ext_m = \
        remu \
        remuw \
        remw \
+       mac \

 riscv_insn_ext_f = \
        fadd_s \
```

4.8 重新构建toolchain和spike

- 重新构建toolchain

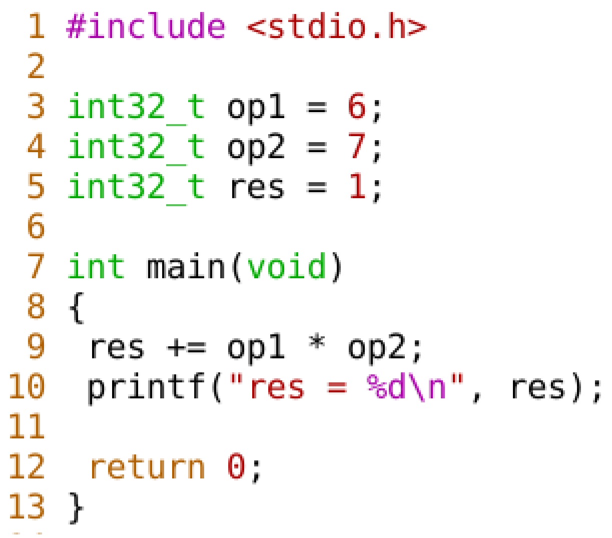
```
$ cd riscv-gnu-toolchain
$ mkdir build && cd build
$ ../configure --prefix=/opt/riscv --enable-multilib
$ make && make install
```

- 重新构建spike

```
$ cd riscv-isa-sim
$ mkdir build && cd build
$ ../configure –prefix=/opt/riscv -enable-histogram
$ make && make install
```

4.9 测试toolchain和spike

- 编写测试文件main.c

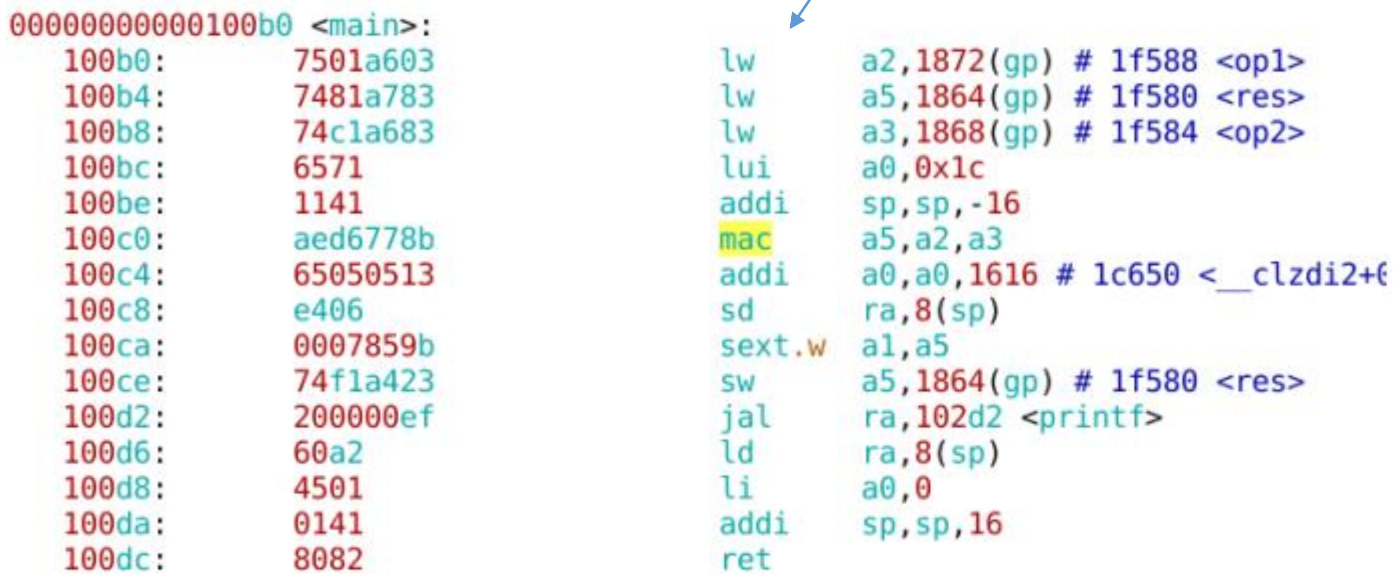


```c
#include <stdio.h>

int32_t op1 = 6;
int32_t op2 = 7;
int32_t res = 1;

int main(void)
{
 res += op1 * op2;
 printf("res = %d\n", res);

 return 0;
}
```

4.9 测试toolchain和spike

- 测试步骤

```
$ riscv64-unknown-elf-gcc -O2 main.c -o main.elf
$ riscv64-unknown-elf-objdump -D main.elf > main.asm
$ spike pk main.elf
```



```
00000000000100b0 <main>:
   100b0:       7501a603                lw      a2,1872(gp) # 1f588 <op1>
   100b4:       7481a783                lw      a5,1864(gp) # 1f580 <res>
   100b8:       74c1a683                lw      a3,1868(gp) # 1f584 <op2>
   100bc:       6571                    lui     a0,0x1c
   100be:       1141                    addi    sp,sp,-16
   100c0:       aed6778b                mac     a5,a2,a3
   100c4:       65050513                addi    a0,a0,1616 # 1c650 <__clzdi2+
   100c8:       e406                    sd      ra,8(sp)
   100ca:       0007859b                sext.w  a1,a5
   100ce:       74f1a423                sw      a5,1864(gp) # 1f580 <res>
   100d2:       200000ef                jal     ra,102d2 <printf>
   100d6:       60a2                    ld      ra,8(sp)
   100d8:       4501                    li      a0,0
   100da:       0141                    addi    sp,sp,16
   100dc:       8082                    ret
```

4.10 作业

- 根据本节内容完成实验报告

深圳大学
SHENZHEN UNIVERSITY

Thank You!