

RISC-V 生态介绍

PLCT GNU小队 陈嘉炜

介绍内容

- RISC-V指令集介绍
 - [RISC-V 指令集](#)
 - [RISC-V 指令分类](#)
 - [RISC-V Profiles](#)
 - [RISC-V 特性](#)
- RISC-V软件介绍
 - [ABI](#)
 - [API](#)
 - [工具链](#)
 - [模拟器](#)
 - [调试器](#)
 - [IDE/SDK](#)
 - [操作系统](#)
- [RISC-V厂商介绍](#)

什么是RISC-V

- RISC-V 是一种**开放标准指令集**，由RISC-V International(RVI)基金会进行管理维护，致力于设计**自由、可选择、灵活**的开放式RISC-V ISA架构扩展
- RVI作为一个非营利性组织，RISC-V对产品或服务没有任何商业利益。RISC-V作为一个开放标准，任何人任何地方都可以从RISC-V贡献和制作的IP中受益
- 官方网站: riscv.org



RISC-V简介

了解 RISC-V。我们的进展。社区。会员计划以及如何参与其中。

幻灯片



RISC-V: 开放计算时代

Calista Redmond 介绍了 RISC-V 的进展。我们所处的位置以及我们要去的地方。

幻灯片

RISC-V指令集规范

- RISC-V指令集规范由RISC-V [技术工作组](#)(TWG)的[RISC-V国际贡献成员](#)开发、批准和维护。规范的发布和维护工作是在 [GitHub 上](#) [执行的](#)，GitHub [问题机制](#)可用于为规范提供输入
- 目前指令集规范分可为
 - **非特权指令集**规范(用户模式)
 - **特权指令集**规范(特权模式或机器模式)

The RISC-V Instruction Set Manual Volume I

Unprivileged Architecture

Editors: Andrew waterman, Krste Asanovic, SiFive, Inc., CS Division, EECS Department,
University of California, Berkeley

Version 20191214, Revised 20230723

The RISC-V Instruction Set Manual: Volume II

Privileged Architecture

Andrew waterman, waterman@eecs.berkeley.edu; Krste Asanović, krste@berkeley.edu;
John Hauser, jh.riscv@jhauser.us, SiFive Inc., CS Division, EECS Department, University of
California, Berkeley

Version 20211203, Revised 20230731

RISC-V指令格式

- RISC-V指令采用32位编码，意味着指令必须在内存中的4字节边界上对齐 (IALIGN=32) (当使用压缩指令扩展时，对齐变为2字节)
- 指令有以下六种形式R-I-S-B-U-J，其操作码(opcode)为固定7位长度

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

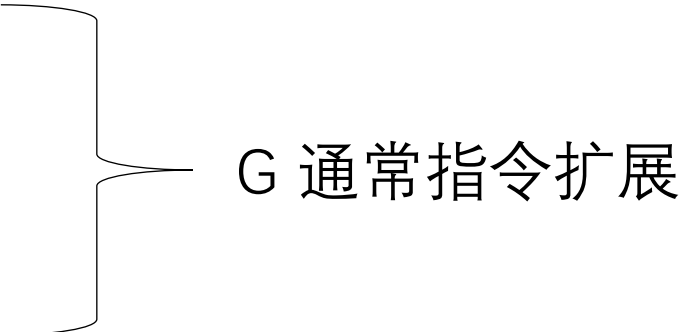
RISC-V指令编码

- RISC-V指令编码定义在对应的RISC-V ISA SPEC文档中
- 编码将寄存器位替换为0后即可得到对应的**操作码编码**
对非寄存器部分取1即可得到**操作码掩码**

例如add rd, rs1, rs2 的操作码为0x33,掩码为0xfe00707f

0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

RISC-V指令分类

- RISC-V指令目前以模块化的扩展(extension)形式进行使用, 这是RISC-V指令集的一大特色, 指令扩展以字母命名
 - I 基础指令扩展
 - M 乘法指令扩展
 - A 原子操作指令扩展
 - F 单精度浮点指令扩展
 - D 双精度浮点指令扩展
 - C 压缩指令扩展
 - V 向量指令扩展
 - Z*** 额外指令扩展 (子扩展)
 - 自定义指令

G 通常指令扩展
- 通常使用G(general-purpose)来代替IMAFD_Zicsr_Zifencei的扩展组合

RISC-V子扩展

- RISC-V子扩展作为RISC-V指令集的延申，以丰富的指令组合增强了RISC-V的使用场景与性能表现

子扩展	扩展名
位操作扩展	Zba,Zbb,Zbc,Zbs
密码学扩展	Zkn,Zks,Zvkn,Zvks
半精度浮点扩展	Zfh,Zvfh
SIMD扩展(草案)	Zpn,Zpsf
指令体积减小扩展	Zca,Zcb,Zcmp,Zcmt
浮点整型寄存器扩展	Zfinx,Zdinx,Zhinx
.....	

RISC-V自定义指令

- RISC-V在设计时预留了大量的指令编码空间，用于用户自定义指令，目前一些RISC-V厂商已向RVI提交通过了厂商自定义扩展
- 与标准指令不同，厂商自定义指令需要在每条指令前注明厂商名称

厂商名	前缀名	厂商链接
Open Hardware Group	cv	https://www.openhwgroup.org/
SiFive	sf	https://www.sifive.com/
T-Head(平头哥)	th	https://www.t-head.cn/
Ventana Micro Systems	vt	https://www.ventanamicro.com/
Nuclei(芯来)	xl	https://nucleisys.com/

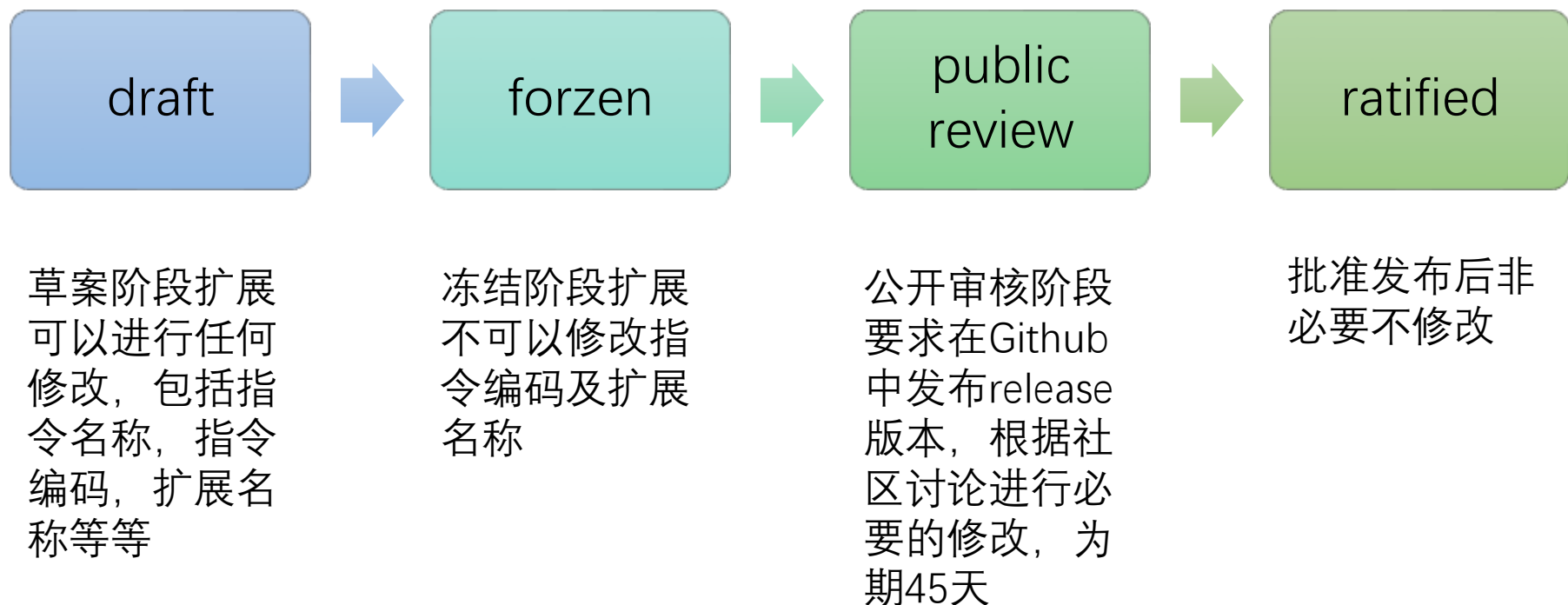
RISC-V厂商自定义扩展

- 厂商的自定义扩展通过RVI审核批准后就可以进入开源社区上游进行维护使用，这可以减小厂商的指令维护成本

厂商名	自定义指令扩展	自定义指令版本	指令介绍
OpenHW	Xcv*	1.0.0	CORE-V Instruction Set Extensions
SiFive	XSFVCP	1.0	SiFive Vector Coprocessor Interface Software Specification
T-Head	Xthead*	1.0	T-Head ISA extension specification
Ventana	XVentanaCondOps	1.0	VTx-family custom instructions

RISC-V扩展周期

- 一个新的扩展通常由四个周期组成



- 目前已经通过批准的扩展——[Recently Ratified Extensions](#)

RISC-V Profiles

- [RISC-V Profiles](#)是RISC-V为了方便进行软硬件设计开放的一套设计标准，其本质是一系列RISC-V扩展的组合
- 其命名规则为

RV + 属性 + 年份 + 模式 + 机器位长

[I/M/A] [20/22] [U/M/S] [32/64]

例如RVA20U64表示——

IMAFDC_Zicsr_Zicntr_Ziccif_Ziccrse_Ziccamoa_Za128rs_Zicclsm

- Profiles包含强制支持扩展及可选支持扩展两个部分，也可在其基础上添加任意需要的扩展

RISC-V 软硬件支持

- 随着RISC-V的普及，RISC-V的软硬件生态发展迅速，目前已有来自70 多个国家/地区，超过3,950 名 RISC-V 组织成员正式加入RISC-V社区
- 支持的硬件包括MCU,CPU,开发板,笔记本电脑等，软件包括工具链， 模拟器， 调试器， 操作系统等

软硬件技术栈



RISC-V ABI

- [psABI](#)(processor-specific)是RISC-V应用运行接口的规范,其中定义了RISC-V的调用约定(Calling Convention), ELF文件格式, DWARF调试信息格式,根据机器位长与寄存器使用情况将ABI分为如下情况

ABI Name	Status
ILP32	Ratified
ILP32F	Ratified
<u>ILP32D</u>	Ratified
ILP32E/LP64E	Ratified
LP64	Ratified
LP64F	Ratified
<u>LP64D</u>	Ratified
LP64Q	Ratified

RISC-V 寄存器调用规范

- ABI中定义寄存器调用规范，包括整型寄存器，浮点寄存器与向量寄存器三种类型

Name	ABI Mnemonic	Meaning	Preserved across calls?
x0	zero	Zero	— (Immutable)
x1	ra	Return address	No
x2	sp	Stack pointer	Yes
x3	gp	Global pointer	— (Unallocatable)
x4	tp	Thread pointer	— (Unallocatable)
x5 - x7	t0 - t2	Temporary registers	No
x8 - x9	s0 - s1	Callee-saved registers	Yes
x10 - x17	a0 - a7	Argument registers	No
x18 - x27	s2 - s11	Callee-saved registers	Yes
x28 - x31	t3 - t6	Temporary registers	No

- Unallocatable**表示调用不应修改寄存器值，因为CPU信号处理对其有依赖，**Immutable**表示所有修改都是无效的，因为x0寄存器硬连线为0

RISC-V 寄存器调用规范

- ABI中定义寄存器调用规范，包括整型寄存器，浮点寄存器与向量寄存器三类

Name	ABI Mnemonic	Meaning	Preserved across calls?
f0 - f7	ft0 - ft7	Temporary registers	No
f8 - f9	fs0 - fs1	Callee-saved registers	Yes*
f10 - f17	fa0 - fa7	Argument registers	No
f18 - f27	fs2 - fs11	Callee-saved registers	Yes*
f28 - f31	ft8 - ft11	Temporary registers	No

- 浮点运算的状态保存在特权寄存器fcsr(floating-point control and status register)中

RISC-V 寄存器调用规范

- ABI中定义寄存器调用规范，包括整型寄存器，浮点寄存器与向量寄存器三类

Name	ABI Mnemonic	Meaning	Preserved across calls?
v0-v31		Temporary registers	No
vl		Vector length	No
vtype		Vector data type register	No
vxrm		Vector fixed-point rounding mode register	No
vxsat		Vector fixed-point saturation flag register	No

RISC-V C/C++ 类型规范

RV32

Type	Size (Bytes)	Alignment (Bytes)
bool/_Bool	1	1
char	1	1
short	2	2
int	4	4
long	4	4
long long	8	8
void *	4	4
_Float16	2	2
float	4	4
double	8	8
long double	16	16
float _Complex	8	4
double _Complex	16	8
long double _Complex	32	16

RV64

Type	Size (Bytes)	Alignment (Bytes)
bool/_Bool	1	1
char	1	1
short	2	2
int	4	4
long	8	8
long long	8	8
__int128	16	16
void *	8	8
_Float16	2	2
float	4	4
double	8	8
long double	16	16
float _Complex	8	4
double _Complex	16	8
long double _Complex	32	16

RISC-V API

- [C-API](#)是RISC-V开发时的应用程序接口规范，其规定了RISC-V预处理器宏与C语言内联函数格式
- RISC-V宏定义以__riscv开头，后接扩展名称
- 内联函数以__riscv_开头，后接指令名称，厂商自定义内联函数需补充厂商名称__riscv_<vendor>_

- [RVV intrinsic手册](#)

```
vint8mf8_t __riscv_vadd_vv_i8mf8(vint8mf8_t vs2, vint8mf8_t vs1, size_t vl);
vint8mf8_t __riscv_vadd_vx_i8mf8(vint8mf8_t vs2, int8_t rs1, size_t vl);
vint8mf4_t __riscv_vadd_vv_i8mf4(vint8mf4_t vs2, vint8mf4_t vs1, size_t vl);
vint8mf4_t __riscv_vadd_vx_i8mf4(vint8mf4_t vs2, int8_t rs1, size_t vl);
vint8mf2_t __riscv_vadd_vv_i8mf2(vint8mf2_t vs2, vint8mf2_t vs1, size_t vl);
vint8mf2_t __riscv_vadd_vx_i8mf2(vint8mf2_t vs2, int8_t rs1, size_t vl);
vint8m1_t __riscv_vadd_vv_i8m1(vint8m1_t vs2, vint8m1_t vs1, size_t vl);
vint8m1_t __riscv_vadd_vx_i8m1(vint8m1_t vs2, int8_t rs1, size_t vl);
vint8m2_t __riscv_vadd_vv_i8m2(vint8m2_t vs2, vint8m2_t vs1, size_t vl);
vint8m2_t __riscv_vadd_vx_i8m2(vint8m2_t vs2, int8_t rs1, size_t vl);
vint8m4_t __riscv_vadd_vv_i8m4(vint8m4_t vs2, vint8m4_t vs1, size_t vl);
vint8m4_t __riscv_vadd_vx_i8m4(vint8m4_t vs2, int8_t rs1, size_t vl);
vint8m8_t __riscv_vadd_vv_i8m8(vint8m8_t vs2, vint8m8_t vs1, size_t vl);
vint8m8_t __riscv_vadd_vx_i8m8(vint8m8_t vs2, int8_t rs1, size_t vl);
vint16mf4_t __riscv_vadd_vv_i16mf4(vint16mf4_t vs2, vint16mf4_t vs1, size_t vl);
vint16mf4_t __riscv_vadd_vx_i16mf4(vint16mf4_t vs2, int16_t rs1, size_t vl);
vint16mf2_t __riscv_vadd_vv_i16mf2(vint16mf2_t vs2, vint16mf2_t vs1, size_t vl);
vint16mf2_t __riscv_vadd_vx_i16mf2(vint16mf2_t vs2, int16_t rs1, size_t vl);
```

RISC-V 工具链

- [riscv-gnu-toolchain](#) 是由 RVI 管理的官方工具链仓库，包含支持 RISC-V 指令集架构的一些列工具（**编译器**，**汇编器**，**链接器**，**调试器**，**模拟器**，**C语言库**，**分析测试工具**），是进行RISC-V开发必备选择
- 其构建包括仓库下载，依赖环境安装，工具编译三个步骤，也提供可直接使用的[release版本](#)

RISC-V 编译工具

- 目前RISC-V的主流编译器包括GCC与LLVM两类，对RISC-V均有较好的支持,OpenJDK,V8等编译器也支持了RISC-V后端
- 其使用方式与其他架构相同，可使用-march与-mabi参数开启特定的RISC-V扩展与使用的ABI(默认参数为rv64gc与lp64d)
gcc -march=rv64gcv -mabi=lp64d
- 注意当使用向量扩展时需要启用-O3与-ffast-math参数，以保证程序中向量模块被更好的优化

RISC-V 模拟器

- 目前QEMU与Spike已较好的支持RISC-V特性，可以模拟RISC-V机器环境运行RISC-V二进制程序，gem5也已支持RISC-V后端，目前一些特性仍在开发中

qemu-riscv64 -cpu rv64,v=true,vlen=256,vext_spec=v1.0 rvv.elf

spike --isa=rv64gcv pk64 rvv.elf

RISC-V 调试工具

- GCC中自带的GDB工具可以用来调试大部分RISC-V程序
- SEGGER等厂商也对RISC-V进行了较好的支持([J-Linker](#))
 - [Imperas](#) Debbuger
 - [Ashling RiscFree](#) Debbuger
 - [Lauterbach JTAG](#) Debbuger
 - [IAR I-jet](#)

RISC-V IDE/SDK

- RISC-V厂商提供了丰富的IDE/SDK开发平台

厂商名称	IDE/SDK
Microchip	SoftConsole
RT-Thread	RT-Thread Studio
Nuclei(芯来)	Nuclei Studio
Thead(平头哥)	Jianchi SDK
Sifive	Freedom Studio
Andes	AndeSight IDE

RISC-V OS

- 许多常用的Linux发行版已支持RISC-V

- [Freertos](#)
- [Ubuntu](#)
- [Archlinux](#)
- [FreeBSD](#)
- [OpenEuler](#)
- [RT-Thread](#)
- [Gentoo](#)
- [Fedora](#)

RISC-V厂商介绍

- 国际厂商：Sifive,Andes,Rivos,Synopsys,Ventana micro,Codasip
- 国内厂商:平头哥，芯来科技，算能，ESWIN，中科昊芯，沁恒微，先楫半导体，赛昉科技，睿思新科，彭峰科技.....
- 科研机构：中科院软件所/计算所，山东大学，上海交通大学，深圳先进技术研究院，剑桥大学，东京大学，UC伯里克利.....
- 具体成员可参考[Members – RISC-V International](#)

RISC-V厂商产品介绍

RISC-V产品众多，下面两个链接是目前市场上产品的汇总

- CPU/SoC [RISC-V Core](#)
- 开发板 [RISC-V Hardware](#)