

# RISC-V\_概略説明

**2021年11月30日**

# RISC-Vとは

- ☑ もともとカリフォルニア大学バークレイで教育用として作られていた。今はRISC-V財団が管理
  - ☑ CPUの命令セット・アーキテクチャ (ISA)
  - ☑ グーグル、オラクル、ヒューレット・パッカード・エンタープライズ (HPE) などが開発に参加
  - ☑ 完全にオープンで自由に使える命令セットアーキテクチャ
  - ☑ アドレッシングは32/64/128bitサポート
- 
- ☑ 上記、Wikipedia参照しました。 <https://ja.wikipedia.org/wiki/RISC-V>
  - ☑ 詳細は FPGAマガジンNo.18とかRISC-VのWebページを参考にしてください。

# 特徴

- ☑ 自由度が高い
  - ▶ 必要なバス幅、命令セットを選んで構成できる
  - ▶ バスも自由：AMBA, Sonic, VME, etc
  - ▶ メモリマップも自由：スタートベクタが途中にあってもOK

この並べ方にも規則有

- ☑ 命令セット 構成例
  - ▶ RV32I MAFDQC

■ RV32I:基本命令 ■ M:乗除算命令 ■ A:アトミック命令 ■ C:圧縮

■ F:単精度小数点 ■ D:倍精度小数点 ■ Q:4倍精度小数点 ■他にも L,B,J,T,P,V,Nなど有

ここまで必要ないなら、例えばRV32IMACだけでもOK

# モジュール一覧

ベース	バージョン	状態	内容
RVWMO	2.0	批准	弱いメモリの順序付けメモリモデル
RV32I	2.1	批准	32bitバス 32レジスタ
RV64I	2.1	批准	64bitバス 32レジスタ
RV32E	1.9	草案	32bitバス 16レジスタ
RV128I	1.7	草案	128bitバス 32レジスタ
拡張	バージョン	状態	内容
M	2.0	批准	乗除算
A	2.1	批准	アトミック
F	2.2	批准	浮動小数
D	2.2	批准	倍精度浮動小数
Q	2.2	批准	4倍精度浮動小数
C	2.0	批准	圧縮命令
Counters	2.0	草案	カウンタ
L	0.0	草案	10進浮動小数
B	0.0	草案	ビット操作
J	0.0	草案	動的翻訳言語
T	0.0	草案	トランザクション的メモリ
P	0.2	草案	パックドSIMD命令
V	0.7	草案	ベクトル演算
Zicsr	2.0	批准	状態と制御レジスタ
Zifencei	2.0	批准	命令フェッチフェンス
Zam	0.1	草案	ミスアラインドアトミック
Ztso	0.1	凍結	トータルストアオーダー

G

**凍結(Freeze)となっているところは今後仕様が変わらない。**

注：RV32G とか"G"の場合がある。GはGeneral purpose (汎用、一般的用途)  
これは G=(IMAFD) と読み替える。つまりRV32G = RV32IMAFD

# | 基本命令一覧

命令	説明	命令	説明
<u>SLL rd,rs1,rs2</u>	論理左シフト	<u>JAL rd,imm</u>	PC保存とジャンプ
<u>SLLI rd,rs1,shamt</u>	論理側値左シフト	<u>JALR rd,rs1,imm</u>	PC保存とレジスタ加算ジャンプ
<u>SRL rd,rs,rs2</u>	論理右シフト	<u>LB rd,rs1,imm</u>	メモリバイトロード
<u>SRLI rd,rs1,shamt</u>	論理即値右シフト	<u>LH rd,rs1,imm</u>	メモリハーフワードロード
<u>SRA rd,rs1,rs2</u>	算術右シフト	<u>LBU rd,rs1,imm</u>	メモリバイト符号なしロード
<u>SRAI rd,rs1,shamt</u>	算術即値右シフト	<u>LHU rd,rs1,imm</u>	メモリハーフワード符号なしロード
<u>ADD rd,rs1,rs2</u>	加算	<u>LW rd,rs1,imm</u>	メモリロードワード
<u>ADDI rd,rs1,imm</u>	加算即値	<u>SB rs1,rs2,imm</u>	メモリバイトストア
<u>SUB rd,rs1,rs2</u>	減算	<u>SH rs1,rs2,imm</u>	メモリハーフワードストア
<u>LUI rd,imm</u>	ロード即値上位	<u>SW rs1,rs2,imm</u>	メモリワードストア
<u>AUIPC rd,imm</u>	PCに上位即値加算	<u>CSRRW rd,csr,rs1</u>	CSRリードライト
<u>XOR rd,rs1,rs2</u>	排他的論理和	<u>CSRRS rd,csr,rs1</u>	CSRリードセットビット
<u>XORI rd,rs1,imm</u>	排他的論理和即値	<u>CSRRC rd,csr,rs1</u>	CSRリードクリアビット
<u>OR rd,rs1,rs2</u>	論理和	<u>CSRRWI rd,csr,imm</u>	CSR即値リードライト
<u>ORI rd,rs1,imm</u>	論理和即値	<u>CSRRSI rd,csr,imm</u>	CSR即値リードセットビット
<u>AND rd,rs1,rs2</u>	論理積	<u>CSRRCI rd,csr,imm</u>	CSR即値リードクリアビット
<u>ANDI rd,rs1,imm</u>	論理積即値	<u>ECALL</u>	実行呼び出し例外
<u>SLT rd,rs1,rs2</u>	符号付き比較	<u>EBREAK</u>	デバッガ呼び出し例外
<u>SLTI rd,rs1,imm</u>	符号付き即値比較	<u>FENCE</u>	メモリとIOの同期
<u>SLTU rd,rs1,rs2</u>	符号なし比較	<u>FENCE.I</u>	メモリとIOの同期即値指定
<u>SLTIU rd,rs1,imm</u>	符号なし即値比較		
<u>BEQ rs1,rs2,imm</u>	分岐 =		
<u>BNE rs1,rs2,imm</u>	分岐 ≠		
<u>BLT rs1,rs2,imm</u>	分岐 <		
<u>BGE rs1,rs2,imm</u>	分岐 ≥		
<u>BLTU rs1,rs2,imm</u>	符号なし分岐 <		
<u>BGEU rs1,rs2,imm</u>	符号なし分岐 ≥		

全47命令、ただし簡易な実装では黄、桃、赤色の所はSYSTEM  
命令1個、NOPに置き換えられるので38命令でOK  
CSRはコントロール・ステータス・レジスタ

# 疑似命令

- NOP命令が無い？ `nop`
  - `addi x0,x0,0` で置き換え
- 単純JUMPしたい？ `jmp offset`
  - `jal x0,offset` で可能
- 戻ってくる命令がない？ `ret`
  - `jalr x0,x1,0` を使用
- 2の補数を生成したい？ `neg rd,rs`
  - `sub rd,x0,rs` で可能

これらはアセンブラが置き換えてくれる。他にも疑似命令いろいろ有。

★ **x0レジスタは0固定になっているレジスタ**

# レジスタ一覧

レジスタ	ABI 名	説明	セーバー
x0	zero	ハードワイヤードゼロ	—
x1	<u>ra</u>	リターン アドレス	呼び出し元
x2	<u>sp</u>	スタック ポインタ	呼び出し先
x3	<u>gp</u>	広域 ポインタ	—
x4	<u>tp</u>	スレッドポインタ	—
x5	t0	一時的/代替リンク・レジスタ	呼び出し元
x6-7	t1-2	一時変数	呼び出し元
x8	s0/ <u>fp</u>	保存レジスタ/フレームポインタ	呼び出し先
x9	s1	保存レジスタ	呼び出し先
x10-11	a0-1	関数の引数/戻り値	呼び出し元
x12-17	a2-7	関数の引数	呼び出し元
x18-27	a2-11	保存レジスタ	呼び出し先
x28-31	t3-6	一時変数	呼び出し元

# オススメの書籍、Web

FPGAマガジン No.18 Googleも推すオープンソースCPU RISC-Vづくり  
CQ出版社

- [http://cc.cqpub.co.jp/lib/system/doclib\\_item/1149/](http://cc.cqpub.co.jp/lib/system/doclib_item/1149/)
- <https://shop.cqpub.co.jp/hanbai/books/46/46281.html>

RISC-V原典 オープンアーキテクチャのススメ 日経BP社

- <https://www.nikkeibp.co.jp/atclpubmkt/book/18/269170/>

プログラマのためのFPGAによるRISC-Vマイコンの作り方 Kindle

堀江 徹也さん @tetsuya\_horie

- [https://www.amazon.co.jp/gp/product/B07G2CHSK3/ref=oh\\_aui\\_d\\_detailpage\\_o00\\_?ie=UTF8&psc=1](https://www.amazon.co.jp/gp/product/B07G2CHSK3/ref=oh_aui_d_detailpage_o00_?ie=UTF8&psc=1)

FPGA開発日記 msyksphinzさん @dev\_msyksphinz

- <http://msyksphinz.hatenablog.com/>



# RISC-V仕様書

RISC-Vの仕様書はここを参照。ただし英語。  
<https://riscv.org/technical/specifications/>

The screenshot shows the RISC-V International website's 'Specifications' page. The page header includes the RISC-V logo and navigation links: About RISC-V, Membership, RISC-V Exchange, Technical, News & Events, and Community. The main heading is 'Specifications'. Below it, a paragraph explains that 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. A link to the membership page is provided for more information on becoming a member.

The page is divided into four columns, each representing a different specification:

- 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]
- Compatibility Test Framework**: The RISC-V Architectural Models against a reference and currently covers MC unprivileged models only. Tests for the ratified Crypto Scalar extension and RV32EMC extensions are also available.

Callouts from the image:

- ISA仕様そのもの**: Points to the ISA Specification column.
- 割り込みとかコントロールレジスタとか**: Points to the ISA Specification column.
- デバッグ仕様**: Points to the Debug Specification column.
- トレース仕様**: Points to the Trace Specification column.

# 日本語訳の場所 @shibatachii翻訳

## GitHub

- <https://github.com/shibatchii/RISC-V>
  - RISC-V\_spec\_manual\_v2.2\_jp.pdf
  - riscv-privileged-v1.10\_jp.pdf
  - MSoffice Word、 LibreOffice Writer 形式も有
- 注：少し古いバージョンの仕様書です。

ANDI、ORI、XORI は、レジスタ rs1 と符号拡張 12 ビットの即値をビット単位で AND、OR、XOR し、その結果を rd に格納する論理演算です。

注：XORI rd、rs1、-1 は、レジスタ rs1 のビット単位の論理反転を実行します（アセンブラ疑似命令 NOT rd、rs）。

31	25 24	20 19	15 14	12 11	7 6	0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode	
7	5	5	3	5	7	
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM	
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM	
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM	

定数によるシフトは、I 型形式の特殊化としてエンコードされます。

シフトされるオペランドは rs1 であり、シフト量は I-即値フィールドの下位 5 ビットにエンコードされます。

右シフト型は、I 即値の上位ビットで符号化されます。

SLLI は論理左シフトです（ゼロは下位ビットにシフトされます）。SRLI は論理右シフトです（0 は上位ビットにシフトされます）。

SRAI は算術右シフトです（元の符号ビットは空いている上位ビットにコピーされます）。