

はじめに

この文章は RISC-V の命令マニュアルを @shibatchii が RISC-V アーキテクチャ勉強のためメモしながら訳しているものです。  
原文は <https://riscv.org/specifications/> にある riscv-spec-v2.2.pdf です。

原文のライセンス表示

The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2", Editors Andrew Waterman and Krste Asanovic, RISC-V Foundation, May 2017.

Creative Commons Attribution 4.0 International License

英語は得意でないので誤訳等あるかもしれません。ご指摘歓迎です。

Google 翻訳、Bing 翻訳、Webilo 翻訳、Exclite 翻訳 を併用しながら翻訳し、勉強しています。

まずは意味が分からないところもあるかもしれませんが、ざっくり訳して 2 周位回ればまともになるかなと。  
体裁とかは後で整えようと思います。

文章は以下の様に色分けしてます。

黒文字：翻訳した文書。

赤文字：@shibatchii コメント。わからないところとか、こう解釈したとか。

青文字：RISC-V にあまり関係なし。集中力が切れた時に書くヨタ話とか。

2018/03/26 @shibatchii

RISC-V 命令セットマニュアル  
第 I 巻：ユーザーレベルの ISA  
ドキュメントバージョン 2.2

編集者：Andrew Waterman 1、Krstje Asanovic 1,2  
1 SiFive Inc.、  
カリフォルニア大学バークレー校の EECS 部門 2 CS 課  
[andrew@sifive.com](mailto:andrew@sifive.com)、[krste@berkeley.edu](mailto:krste@berkeley.edu)  
2017 年 5 月 7 日

アルファベット順の仕様全バージョン貢献者（訂正があれば編集者に連絡してください）：クレステ・アサノビック、リマス・アヴィジエニス、ジェイコブ・バッハマイヤー、クリストファー・エフ・バッテン、アレン・ジェイ・バウム、アレックス・ブラッドベリー、スコット・ビーマー、プレストン・ブリッグス、クリストファー・セリオ、デイビッド・キスナル、ポール・クレイトン、パーマー・ダッベルト、ステファン・フロイデンベルガー、ジャン・グレイ、マイケル・ハンブルク、ジョン・ハウザー、デイヴィッド・ホーナー、オルフ・ヨハンソン、ベン・ケラー、ユンサップ・リー、ジョセフ・マイヤーズ、リシュユール・ニヒル、ステファン・オレア、アルバート・オー、ジョン・オースターハウト、デヴィッド・パターソン、コリン・シュミット、マイケル・ティラー、ウェズリー・タープストラ、マット・トーマス、トミー・ソーン、レイ・バン・デーウォーカー、メガン・ワックス、アンドリュー・ウォーターマン、ロバート・ワトソン、そして、レイノルド・ザンチジク。

このドキュメントはクリエイティブコモンズ帰属 4.0 国際ライセンスの下で公開されています。

このドキュメントは、「RISC-V 命令セットマニュアル第 1 巻：ユーザーレベル ISA バージョン 2.1」を次のライセンスの下でリリースした派生物です：(c) 2010-2017 アンドリュー・ウォーターマン、ユンサップ・リー、デビッド・パターソン、クレステ・アサノビック クリエイティブコモンズ帰属 4.0 国際ライセンス。

次のように引用してください：「RISC-V 命令セットマニュアル、第 1 巻：ユーザーレベル ISA、ドキュメントバージョン 2.2」、編集者アンドリュー・ウォーターマンとクレステ・アサノビック、RISC-V 財団、2017 年 5 月。

配布条件はこれですね。<https://creativecommons.org/licenses/by/4.0/deed.ja>  
制限が少ない大変良いですね。

-2018/03/26

序文

これは、RISC-V ユーザーレベルのアーキテクチャを説明するドキュメントのバージョン 2.2 です。

このドキュメントには RISC-V ISA モジュールの次のバージョンが含まれています。

ベース	バージョン	凍結?
RV32I	2.0	Y
RV32E	1.9	N
RV64I	2.0	Y
RV128I	1.7	N
拡張	バージョン	凍結?
M	2.0	Y
A	2.0	Y
F	2.0	Y
D	2.0	Y
Q	2.0	Y
L	0.0	N
C	2.0	Y
B	0.0	N
J	0.0	N
T	0.0	N
P	0.1	N
V	0.2	N
N	1.1	N

今日まで、RISC-V 財団によって正式に批准された規格はありませんが、上記の "凍結" と表示されているコンポーネントは、曖昧さや仕様の穴を解決する以上に批准プロセス中に変更されることはありません。

凍結ってなっているところは「曖昧なところや穴があったときは修正される」けど、仕様は今後変わらないよ。ってことらしい。

このバージョンのドキュメントの主な変更点は次のとおりです。

- ・このドキュメントの以前のバージョンは、元の作者が作成したクリエイティブコモンズ帰属 4.0 国際ライセンスの下でリリースされました。このドキュメントの今後のバージョンは、同じライセンスでリリースされる予定です。
- ・すべての拡張子を標準的な順序で最初に置くように章を再編成しました。
- ・説明と解説の改善。
- ・LUI / JALR と AUIPC / JALR ペアのより高度なマクロオペレーションの融合をサポートするための JALR に関する暗黙のヒント提案を修正しました。

↑なんじゃろ。JALR のところででてくるかな。

- ・ロード予約/ストア条件付きシーケンスに対する制約の明確化。
- ・コントロールとステータスレジスタ（CSR）のマッピングの新しいテーブル。
- ・fcsr の上位ビットの目的と動作を明確にしました。
- ・FNMADD.fmt および FNMSUB.fmt 命令の説明が修正されました。これは、ゼロ結果の符号が正しくないことを示唆していました。
- ・命令 FMV.S.X と FMV.X.S は FMV.W.X と FMV.X.W に変わらなかったそれらの意味論とより一致するようにそれぞれ名前が変更されました。古い名前は引き続きツールでサポートされます。

↑意味が合うように命令を変えたってことか。

- ・より狭い（<FLEN）浮動小数点値の指定された作用は、NaN-ボクシング・モデルを使っているより広いfレジスターをおさえました。

↑広いfレジスタに拡張(移行)されたってことかな。

- ・FMA（1、0、qNaN）の例外作用を定めました。
- ・P 拡張が整数レジスタを使用した固定小数点演算の整数パック SIMD 提案に再加工される可能性があることを示すノートを追加しました。
- ・Vベクトル命令セット拡張のドラフト提案。
- ・N ユーザレベルのトラップ拡張の早期草案。
- ・拡張された疑似命令リスト。
- ・RISC-V ELF psABI 仕様にとって代わった呼び出し規約の章の削除 [1]。
- ・C の拡張機能は凍結され、バージョン番号は 2.0 に変更されました。

-2018/03/28

## 文書 版 2.1 の序文

これは、RISC-V ユーザーレベルアーキテクチャを記載している文書の版 2.1 です。  
凍結されたユーザーレベル ISA ベースと拡張 IMAFDQ 版 2.0 がこの文書の前の版から変わらなかった点に注意してください [36]、しかし、いくつかの仕様の穴は修正されました、そして、文章は改善されました。  
ソフトウェアの規則にいくつかの変更が加えられました。

- ・解説セクションへの多数の追加と改良。
- ・各々の章のための別々のバージョン番号
- ・非常に長い命令フォーマットで rd 指定子を移動させないようにするために、64 ビットを超える長い命令符号化への変更。
- ・CSR 命令は、後で浮動小数点セクション（および付随する特権アーキテクチャマニュアル）に導入されるのとは対照的に、カウンタレジスタが導入される基本整数フォーマットで記述されます。
- ・SCALL 命令と SBREAK 命令は、それぞれ ECALL と EBREAK に名前が変更されました。それらのエンコーディングと機能は変更されていません。
- ・浮動小数点 NaN 処理の明確化、および新しい標準 NaN 値
- ・オーバーフローする浮動小数点から整数への変換によって返される値の明確化。
- ・LR / SC の明確化は、シーケンス内の圧縮命令の使用を含む、成功と必要な失敗を許しました。
- ・整数レジスタ数を削減し、MAC 拡張をサポートする新しい RV32E ベースの ISA 提案。
- ・改訂された呼び出し規約。
- ・ソフトフロート呼び出し規約のためのリラックスしたスタックアライメント、 および RV32E 呼び出し規約の説明。

-2018/03/29

- ・ C の圧縮された拡張のための修正案、バージョン 1.9。

## バージョン 2.0 の序文

これは、ユーザー isa の仕様の 2 番目のリリースであり、我々は、ベースユーザーの isa plus の一般的な拡張機能 (すなわち、IMAFD) の仕様は、将来の開発のために固定されたままにするつもりです。

この ISA 仕様のバージョン 1.0 [35] 以降では、以下の変更が行われています。

- ・ ISA はいくつかの標準拡張を持つ整数ベースに分割されています。
  - ・ 命令フォーマットは、即時エンコードをより効率的にするために再編成されています。
  - ・ ベース ISA は、リトルエンディアンメモリシステムを持ち、ビッグエンディアンまたはバイエンディアンが非標準のバリエーションであると定義されています。
  - ・ アトミック命令拡張で、Load-Reserved / Store-Conditional (LR / SC) 命令が追加されました。
  - ・ AMO および LR / SC は、リリース一貫性モデルをサポートできます。
  - ・ フェンス命令は、より細かいメモリと I / O 順序を提供します。
  - ・ fetch-and-XOR (AMOXOR) の AMO が追加され、部屋を作るために AMOSWAP のエンコーディングが変更されました。
- ↑ 部屋を作るって場所を空けるっていう感じか。
- ・ AUIPC 命令は、PC に 20 ビットの上位の即値を追加し、現在の PC 値のみを読み取る RDNPC 命令を置き換えます。これにより、位置に依存しないコードが大幅に節約されます。
  - ・ JAL 命令は、明示的なデスティネーションレジスタを持つ U タイプフォーマットに移動し、J 命令は、RAL = x0 の JAL に置き換えられ、なくなりました。
- これにより、暗黙のデスティネーションレジスタを持つ唯一の命令が削除され、ベース ISA から J-Type 命令フォーマットが削除されます。これに伴い、JAL の到達範囲が縮小されますが、ベース ISA の複雑さは大幅に軽減されます。
- ・ JALR 命令の静的ヒントが削除されました。このヒントは、標準の呼び出し規約に準拠したコードの rd および rs1 レジスタ指定子では冗長です。
  - ・ ハードウェアを単純化し、補助情報を関数ポインタに格納できるように、JALR 命令は計算されたターゲットアドレスの最下位ビットをクリアするようになりました。
  - ・ MFTX.S 命令と MFTX.D 命令は、それぞれ FMV.X.S と FMV.X.D に名前が変更されました。同様に、MXTF.S および MXTF.D 命令は、それぞれ FMV.S.X および FMV.D.X に名前が変更されました。
  - ・ MFFSR および MTFSR 命令は、それぞれ FRCSR および FSCSR に名前が変更されました。
- fcsr の丸めモードおよび例外フラグのサブフィールドに個別にアクセスするために、FRRM、FSRM、FRFLAGS、および FSFLAGS 命令が追加されました。
- ・ FMV.X.S および FMV.X.D 命令は、rs2 ではなく rs1 からオペランドがソースになります。この変更により、データパス設計が簡素化されます。
  - ・ FCLASS.S および FCLASS.D 浮動小数点分類命令が追加されました。
  - ・ より単純な NaN 生成および伝播方式が採用されています。
  - ・ RV32I の場合、システムパフォーマンスカウンタは 64 ビット幅に拡張されており、上位および下位 32 ビットへの個別の読み取りアクセスが可能です。
  - ・ 標準的な NOP および MV エンコーディングが定義されています。

-2018/03/30

- ・ 標準的な命令長エンコーディングは、48 ビット、64 ビット、および >64 ビット命令で定義されています。
- ・ 128 ビットアドレス空間バリエーション RV128 の説明が追加されました。
- ・ 32 ビットの基本命令フォーマットの主なオペコードは、ユーザ定義のカスタム拡張のために割り当てられています。
- ・ rd のデータをストアすることを示唆している誤植は、rs2 を参照するように修正されました。

-2018/04/01

## 内容

## 序文

1 前書き	1
1.1 RISC-V ISA の概要。	3
1.2 命令長符号化。	5
1.3 例外、トラップ、および割り込み。	7
2 RV32I ベース整数命令セット、バージョン 2.0	9
2.1 基本整数サブセットのプログラマーのモデル。	9
2.2 基本命令フォーマット。	11
2.3 即値エンコーディングの変形。	11
↑ valiant をどう訳すか	
2.4 整数計算命令。	13
2.5 コントロール転送命令。	15
2.6 ロードとストア命令。	18
2.7 メモリモデル。	20
2.8 制御と状態レジスタ命令。	21
2.9 環境呼び出しとブレークポイント。	24
3 RV32E ベース整数命令セット、バージョン 1.9	27
3.1 RV32E プログラムのモデル。	27
3.2 RV32E 命令セット。	27
3.3 RV32E 拡張。	28



4 RV64I ベース整数命令セット、バージョン 2.0	29
4.1 レジスタの状態。	29
4.2 整数計算命令。	29
4.3 ロードおよびストア命令。	31
4.4 システム命令。	32
5 RV128I 基本整数命令セット、バージョン 1.7	33
6 "M"整数乗算および除算標準拡張、バージョン 2.0	35
6.1 乗算演算。	35
6.2 除算演算。	36
7 "A"原子命令の標準拡張、バージョン 2.0	39
7.1 原子命令の順序付けの指定。	39
7.2 ロード予約/ストア条件付き命令。	40
7.3 原子メモリ操作。	43
↑ Atomic ってどういう意味だろう？ なんて訳すのがいいのか。	
8 単精度浮動小数点"F"標準拡張 バージョン 2.0	45
8.1 F レジスタ状態。	45
8.2 浮動小数点制御およびステータスレジスタ。	47
8.3 NaN の生成と伝播。	48
8.4 非正常な算術。	49
8.5 単精度ロード命令とストア命令。	49
8.6 単精度浮動小数点計算命令。	49
8.7 単精度浮動小数点変換および移動命令。	51
8.8 単精度浮動小数点比較命令。	52
8.9 単精度浮動小数点分類命令。	53
9 "D"倍精度浮動小数点標準拡張 バージョン 2.0	55
9.1 D レジスタの状態。	55

9.2 NaN のより狭い値の箱詰め。	55
9.3 倍精度ロード命令とストア命令。	56
9.4 倍精度浮動小数点計算命令。	57
9.5 倍精度浮動小数点変換および移動命令。	57
9.6 倍精度浮動小数点比較命令。	59
9.7 倍精度浮動小数点分類命令。	59
10 "Q"四倍精度浮動小数点の標準拡張 バージョン 2.0	
10.1 四倍精度ロード命令とストア命令。	61
10.2 四倍精度計算命令。	62
10.3 四倍精度変換および移動命令。	62
10.4 四倍精度浮動小数点比較命令。	63
10.5 四倍精度浮動小数点分類命令。	63
11 "L" 10 進浮動小数点の標準拡張、バージョン 0.0	65
11.1 10 進浮動小数点レジスタ。	65
12 "C" 圧縮された命令の標準拡張、バージョン 2.0	67
12.1 概要。	67
12.2 圧縮された命令フォーマット。	69
12.3 ロードとストア命令。	71
12.4 制御転送命令。	74
12.5 整数計算命令。	76
12.6 LR / SC シーケンスにおける C 命令の使用。	80
12.7 RVC 命令セットリスト。	81
13 "B" ビット操作の標準拡張、バージョン 0.0	85
14 "J" 動的に翻訳された言語の標準拡張、バージョン 0.07	
15 "T" トランザクションメモリの標準拡張、バージョン 0.0	

16 "P" 圧縮された(パックされた)SIMD 命令の標準拡張、バージョン 0.1	91
17 "V"ベクトル演算標準拡張、バージョン 0.2	93
17.1 ベクターユニットの状態。	93
17.2 要素のデータ型と幅。	93
17.3 ベクトル構成レジスタ (vcmaxw、vctype、vcp)	95
17.4 ベクトルの長さ。	97
17.5 迅速な設定手順。	97
18 "N" ユーザーレベル割り込みの標準的な拡張、バージョン 1.1	101
18.1 追加の CSR。	101
18.2 ユーザステータスレジスタ(ustatus)	102
18.3 その他の CSR。	102
18.4 N 拡張命令。	102
18.5 前後関係交換オーバーヘッドの削減。	102
19 RV32 / 64G 命令セット一覧	103
20 RISC-V アセンブリプログラマーズハンドブック	109
21 RISC-V 拡張	113
21.1 拡張用語。	113
21.2 RISC-V 拡張設計哲学(の考え方)。	116
21.3 固定幅の 32 ビット命令フォーマット内の拡張。	116
21.4 整列した(アライン)64 ビット命令拡張の追加。	118
21.5 VLIW エンコーディング支援(提供?)。	118
22 ISA サブセット命名規則	121
22.1 大文字小文字の区別。	121
22.2 基本整数 ISA。	121
22.3 命令拡張名。	121

22.4 バージョン番号。	122
22.5 非標準拡張名。	122
22.6 管理レベルの命令サブセット。	122
22.7 管理レベルの拡張。	122
22.8 サブセット命名規則。	123

## 23 歴史と謝辞

23.1 ISA マニュアルのリビジョン 1.0 履歴。	。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。	125
23.2 ISA マニュアルのリビジョン 2.0 履歴。	。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。	126
23.3 改訂履歴 2.1。	。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。	128
23.4 改訂履歴 2.2。	。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。	128
23.5 資金調達。	。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。	129

-2018/04/03

1 ページ空き。次ページへ。

## 第1章

### 前書き

RISC-V（「リスク5」と発音）は、コンピュータアーキテクチャの研究と教育をサポートするためにもともと設計された新しい命令セットアーキテクチャ（ISA）であり、

私たちが今望んでいるのは、業界標準の自由で解放されたアーキテクチャです。

RISC-Vを定義する私たちの目標は次のとおりです。：

- ・ 学界や産業界が自由に利用できる完全に解放された ISA です。
- ・ 実際の ISA シミュレーションまたはバイナリ変換だけでなく、直接ネイティブハードウェア実装に適しています。
- ・ ISA は特定のマイクロアーキテクチャスタイル（例えばマイクロコード化、インオーダー、デカップル、アウトオブオーダー）や実装技術（フルカスタム、ASIC、FPGA など）を「オーバーアーキテクチャー」することなく、効率的にこれらのいずれかの実装が可能にします。
- ・ ISA は、汎用のソフトウェア開発をサポートするために、カスタマイズされたアクセラレータまたは教育目的のためのベースとして使用可能な小さな基本整数 ISA とオプションの標準拡張子に分かれています。
- ・ 改訂された 2008 IEEE-754 浮動小数点標準のサポート [14]。
- ・ 広範なユーザーレベルの ISA 拡張機能と特殊なバリエーションをサポートする ISA。  
↑ specialized variants. ってどう訳す？ 特別な変形、変異体、変数、展開
- ・ アプリケーション、オペレーティングシステムカーネル、およびハードウェア実装の 32 ビットおよび 64 ビットアドレス空間の両バリエーション。
- ・ 異種マルチプロセッサを含む、高度に並列なマルチコアまたは多くのコアの実装を支援する ISA。  
↑ multicore と manycore の違いは？ どっちも沢山のコアのように思えるが
- ・ オプションの可変長命令は、使用可能な命令エンコーディング空間を拡張と、オプションの高密度命令エンコーディングを提供し、パフォーマンス、静的コードサイズ、およびエネルギー効率を向上させます。
- ・ ハイパーバイザー開発を容易にする完全仮想化 ISA
- ・ 新しいスーパーバイザーレベルとハイパーバイザーレベルの ISA デザインを使用して実験を簡単にする ISA。  
↑ experiments 実験、試み なんだけどなんかしっくりこない

---

私たちのデザイン決定に関する論評(解説)は、この段落のように書式化されており、読者が仕様書そのものに興味があればスキップすることができます。

---

RISC-V という名前は、UC Berkeley (RISC-I [23]、RISC-II [15]、SOAR [32]、SPUR [18]の最初の4つ) から5番目の主要な RISC ISA 設計を代表するものです。

さまざまなデータ並列アクセラレータを含む一連のアーキテクチャ研究の支援が ISA 設計の明白な目標で、"バリエーション"と "ベクトル"を表すローマ数字 "V"の使用についてもしゃれ(だじゃれ)を言います。

↑ 1つの文になってるけど、2つの文として考えた方がいいような。～明確な目標です。と ～言います。

-----  
我々は、実際のハードウェア実装の研究アイデア（この仕様書の初版以来 11 種類のシリコン製作を完了している）に特に関心のある研究と教育のニーズを支援するために RISC-V を開発しました。（RISC-V プロセッサの RTL デザインは、Berkeley の複数の学部および大学院クラスで使用されています）。

我々の現在の研究では、従来のトランジスタスケールングの終了によって課された電力制約によって推進される、特殊(専門)で異種なアクセラレータへの移行に特に関心があります。

私たちは、私たちの研究努力を築くための非常に柔軟で拡張性のあるベース ISA を求めました。

我々が繰り返し尋ねられた質問は、なぜ新しい ISA を開発するのかということです。既存の商用 ISA を使用する最大の明白な利点は、研究と教育に活用できる開発ツールと移植されたアプリケーションの両方の、大規模で広くサポートされているソフトウェアエコシステムです。

↑既存の ISA はツールとアプリケーションがたくさんあるよ。ってことかな

その他の利点としては、大量の文章と例題(チュートリアル)があります。

しかし、商用の命令セットを研究と教育に使用した経験では、実際にはこれらの利点が小さく、欠点を上回らないことです。

-2018/04/05

・商用 ISA は専有です。

オープンな IEEE 標準[2]である SPARC V8 を除き、商用 ISA のほとんどの所有者は、知的財産を慎重に保護し、自由に利用可能な競争の実装を歓迎しません。

これは、ソフトウェアシミュレータのみを使用した学術研究や教育では大きな問題は少ないですが、実際の RTL 実装を共有しようとするグループにとっては大きな懸念事項でした。

また、商用 ISA 実装のいくつかのソースを信頼したくないが、独自のクリーンルーム実装を作成することは禁止されているエンティティにとって、大きな懸念事項です。

すべての RISC-V 実装に第三者の特許侵害がないことを保証することはできませんが、RISC-V 実装者を訴えるしよとしないことを保証することができます。

・商用 ISA は特定の市場分野でのみ普及しています。

執筆時点での最も明白な例は、ARM アーキテクチャがサーバ空間で十分にサポートされていないことと、インテル x86 アーキテクチャ（または他のほとんどすべてのアーキテクチャ）がモバイル領域で十分にサポートされていないことです。インテルと ARM は互いの市場区分(領域)に参入しようとしています。

もう 1 つの例は、拡張可能なコアを提供するが、埋め込み領域に焦点を当てている ARC と Tensilica です。

この市場分割は、特定の商用 ISA を実際にはソフトウェアエコシステムが特定のドメインにのみ存在し、他のユーザーのために構築しなければならないという利点を希釈します。

↑特定の所だけのソフトウェアエコシステムなので、ほかのユーザーのためにならないよ ってことかな

・商業的な ISA が出て行きます。

以前の研究基盤は、もはや普及していない（SPARC、MIPS）、あるいはもはや生産段階でない（Alpha）、商用の ISA まわりに構築されました。

これらは、活発なソフトウェア生態系の利益を失い、ISA および支援ツールに関する長引く知的財産の問題は、関心のある第三者が ISA の支援を継続する能力を妨げます。

オープン ISA は人気を失うかもしれないが、利害関係者はいずれもエコシステムの使用と開発を続けることができます。

・一般的な商用 ISA は複雑です。

主要な商用 ISAs（x86 および ARM）は、ハードウェアで一般的なソフトウェアスタックおよびオペレーティングシステムをサポートするレベルを実装するのに非常に複雑です。

さらに悪いことに、ほとんどすべての複雑さは、真に効率を改善する機能ではなく、間違っ、あるいは少なくとも旧式の ISA デザインの決定によるものです。

・商用 ISA だけでは、アプリケーションを起動するには十分ではありません。

市販の ISA を実装する努力を費やしたとしても、その ISA 用に既存のアプリケーションを実行するだけでは不十分です。

ほとんどのアプリケーションでは、ユーザーレベルの ISA だけでなく、実行するための完全な ABI（アプリケーション バイナリ インターフェイス）が必要です。

ほとんどの ABI はライブラリに依存しており、オペレーティングシステムの支援に依存しています。

既存のオペレーティングシステムを実行するには、OS が期待する管理レベルの ISA とデバイスインタフェースを実装する必要があります。

これらは通常、ユーザレベルの ISA よりもはるかに明確ではなく、実装がかなり複雑です。

・一般的な商用 ISA は拡張性のために設計されていませんでした。

支配的な商用 ISA は、特に拡張性のために設計されておらず、その結果、命令セットが大きくなったため、命令エンコーディングの複雑さが増しています。

Tensilica（Cadence 社買収）や ARC（Synopsys 社買収）などの企業では、拡張性を重視して ISA とツールチェーンを構築していますが、汎用コンピューティングシステムではなく組み込みアプリケーションに注力しています。

・変更された商用 ISA は新しい ISA です。

私たちの主な目標の 1 つは、主要な ISA 拡張を含むアーキテクチャ研究をサポートすることです。

小さな拡張機能でもコンパイラを修正し、拡張機能を使用するためにソースコードからアプリケーションを再構築する必要があるため標準の ISA を使用する利点は少なくなります。

新しいアーキテクチャ状態を導入するより大きな拡張では、オペレーティングシステムを変更する必要があります。

最終的に、変更された商用 ISA は新しい ISA になりますが、ベース ISA のすべての遺産障害を引継ぎます。

私たちの立場は、ISA はおそらくコンピューティングシステムの中で最も重要なインターフェースであり、そのような重要なインターフェースが独自(専有)のものでなければならない理由はありません。

支配的な商用 ISA は、30 年以上前に既によく知られている命令セットの概念に基づいています。

ソフトウェア開発者はオープンな標準ハードウェア目標を目標とすることができ、商用プロセッサ設計者は実装品質を競うべきです。

我々はハードウェアの実装に適したオープンな ISA 設計を最初に検討しているところはありません。

↑我々は最初から遠いです。って言ってるんだけどいまいちわからん。

我々はまた、OpenRISC アーキテクチャ[22]に最も近いものとして、他の既存のオープン ISA 設計を検討しました。

私たちはいくつかの技術的な理由から OpenRISC ISA を採用することに反対しました。

・ OpenRISC はより高性能な実装が複雑な条件コードと分岐遅延スロットを持っています。

・ OpenRISC は、固定の 32 ビットエンコーディングと 16 ビット即値を使用しており、これにより、より高密度の命令エンコーディングが排除され、後で ISA を拡張するためのスペースが制限されます。

・ OpenRISC は 2008 年の IEEE 754 浮動小数点標準の改訂をサポートしていません。

・ OpenRISC 64 ビットデザインは、私たちが始めたときに完成していませんでした。

白紙の状態から始めることによって、すべての目標を達成した ISA を設計することができましたが、当初計画していたよりもはるかに労力がかかりました。

ドキュメント、コンパイラツールチェーン、オペレーティングシステムポート、参照(基準)ISA シミュレータ、FPGA 実装、効率的な ASIC 実装、アーキテクチャ試験項目群、教材など、RISC-V ISA インフラストラクチャを構築するために多大な労力を投じました。このマニュアルの最後の版以降、学界と産業界で RISC-V ISA かなりの取り込みが行われており、我々は標準を保護し促進するために非営利の RISC-V 財団を創設しました。

RISC-V 財団ウェブサイト (<http://riscv.org>) には、RISC-V を使用した財団メンバーシップと様々なオープンソースプロジェクトに関する最新情報が掲載されています。

RISC-V マニュアルは 2 つの巻で構成されています。

この巻は、オプションの ISA 拡張を含むユーザーレベルの ISA 設計について説明します。

2 番目の巻は特権アーキテクチャを提供します。

---

このユーザーレベルのマニュアルでは、特定のマイクロアーキテクチャー機能や特権アーキテクチャーの詳細に依存することを排除することを目指しています。

これは、明快さと、代替実装のための最大の柔軟性を可能にするためです。

-2018/04/05

## 1.1 RISC-V ISA の概要

RISC-V ISA は、任意の実装に存在しなければならない基本整数 ISA と、基本 ISA へのオプションの拡張として定義されます。

基本整数 ISA は、分岐遅延スロットがなく、オプションの可変長命令エンコーディングをサポートしている点を除いて、初期の RISC プロセッサのものと非常によく似ています。



ベースはコンパイラ、アセンブラ、リンカ、およびオペレーティングシステム（追加の管理者レベルの操作を含む）のための合理的なターゲットを提供するのに十分な最小限の命令セットに注意深く制限されているので、便利な ISA およびソフトウェアツールチェーン "スケルトン" よりカスタマイズされたプロセッサ ISA を構築することができます。

各基本整数命令セットは、整数レジスタの幅および対応するユーザアドレス空間のサイズによって特徴付けられます。2つの主要な基本整数の変形、RV32I と RV64I があり、それぞれ第2章と第4章で説明します。これらは32ビットまたは64ビットのユーザーレベルのアドレス空間を提供します。ハードウェアの実装およびオペレーティングシステムは、ユーザープログラムに対して RV32I と RV64I の一方または両方のみを提供する場合があります。第3章では、小型マイクロコントローラをサポートするために追加された RV32I ベース命令セットの RV32E サブセットの変形について説明します。第5章では、将来の128ビットユーザアドレス空間をサポートする基本整数命令セットの RV128I 変形について説明します。基本整数命令セットは、符号付き整数値に対して2の補数表現を使用します。

大規模システムでは64ビットのアドレス空間が必要ですが、32ビットのアドレス空間は多くのエンベデッド・デバイスやクライアント・デバイスにとって今後も数十年の間十分なままであり、メモリ・トラフィックとエネルギー消費を削減することが望めます。さらに、教育目的では32ビットのアドレス空間で十分です。最終的には128ビットのアドレス空間が必要になりますので、これを RISC-V ISA フレームワークに収めることができます。

基本整数 ISA はハードウェア実装によってサブセットになるかもしれませんが、より特権層によるオペコードトラップとソフトウェアエミュレーションは、ハードウェアによって提供されない機能を実装するために使用する必要があります。

基本整数 ISA のサブセットは教育目的には役立つかもしれませんが、ベースが逸脱しているため、実際のハードウェアの実装をサブセット化するインセンティブはほとんどなく、メモリの不整合のサポートを省略し、すべての SYSTEM 命令を単一のトラップとして扱います。

↑ここは何をいっているのかいまいちわからん。要約(意識)すると、基本整数 ISA は教育には役立つかも。でもこの基本はメモリのミスアライン(整列)を省略してるし、すべてのシステム命令を1トラップのように扱ってるので、実際のハードウェア実装のサブセットの動機(報酬)は少ないよ。 ってとこか

RISC-V は、豊富なカスタマイズと特殊化をサポートするように設計されています。基本整数 ISA は、1つまたは複数のオプションの命令セット拡張で拡張できますが、基本整数命令は再定義できません。RISC-V 命令セット拡張機能を標準および非標準拡張に分割します。標準拡張は一般的に有用であり、他の標準拡張と競合しないようにする必要があります。非標準拡張は高度に専門化されているか、他の標準または非標準拡張と競合する可能性があります。命令セット拡張は、基本整数命令セットの幅に応じて若干異なる機能を提供することがあります。第21章では、RISC-V ISA を拡張するさまざまな方法について説明します。また、RISC-V ベース命令と命令セット拡張の命名規則も開発しました（第22章で詳しく説明しています）。

より一般的なソフトウェア開発をサポートするために、整数の乗算/除算、アトミック演算、単精度浮動小数点演算と倍精度浮動小数点演算を提供する標準拡張のセットが定義されています。

基本整数 ISA は "I"（整数レジスタの幅に応じて RV32 または RV64 が前に付く）という名前で、整数計算命令、整数ロード、整数ストア、および制御フロー命令を含み、すべての RISC-V 実装で必須です。標準の整数の乗算および除算の拡張は「M」と命名され、整数レジスタに保持された値を乗算および除算するための命令を追加します。「A」で示される標準的な原子命令拡張は、プロセッサ間同期のためにメモリを原子的に読み取り、修正し、書き込む命令を追加します。「F」で示される標準単精度浮動小数点拡張は、浮動小数点レジスタ、単精度計算命令、および単精度ロードおよびストアを追加します。「D」で示される標準倍精度浮動小数点拡張は、浮動小数点レジスタを拡張し、倍精度の計算命令、ロード、およびストアを追加します。整数ベースに加えて、これら4つの標準拡張（「IMAFD」）には、略語「G」が与えられ、汎用スカラ命令セットが提供される。現在、RV32G と RV64G はコンパイラツールチェーンのデフォルトターゲットです。後の章では、これらおよびその他の計画された標準的な RISC-V 拡張について説明します。

↑アトミックとアトミカルってどう訳すべきか。ARM AMBA AXI 規格に アトミック転送ってあったような。あれと同じ？

ベース整数 ISA および標準の拡張を超えて、新しい命令がすべてのアプリケーションに大きな利点をもたらすことはまれですが、特定のドメインにとっては非常に有益です。

エネルギー効率の懸念により、より専門化が強まっているので、ISA 仕様の必要部分を簡素化することが重要であると考えています。他のアーキテクチャでは通常、ISA を単一のエンティティとして扱いますが、命令が時間の経過と共に追加されるにつれて新しいバージョンに変更されますが、RISC-V はベースと各標準拡張を時間の経過とともに一定に保ち、その代わりにさらにオプションの拡張として新しい命令を階層化します。

たとえば、基本整数 ISAs は、後続の拡張機能に関係なく、完全に支持されている独立型 ISAs として引き続き使用されます。

ユーザー ISA 仕様の 2.0 リリースでは、将来の開発のために、"RV32IMAFD"と "RV64IMAFD"ベースと標準拡張（別名 "RV32G"と "RV64G"）を一定に保つ予定です。

1.2 命令長エンコーディング

ベース RISC-V ISA は、固定長の 32 ビット命令を備えています。これらの命令は、32 ビット境界で自然に整列する必要があります。しかし、標準の RISC-V エンコーディング方式は、可変長命令を持つ ISA 拡張をサポートするように設計されています。各命令は任意の数の 16 ビット命令区切りになり、16 ビット境界で自然に整列されます。

第 12 章で説明した標準の圧縮 ISA 拡張命令は、圧縮された 16 ビット命令を提供することでコードサイズを縮小し、すべての命令（16 ビットと 32 ビット）を 16 ビット境界で整列させてコード密度を向上させるための整列制約を緩和します。

図 1.1 に、標準的な RISC-V 命令長エンコード規則を示します。

ベース ISA のすべての 32 ビット命令は、最下位の 2 ビットが 11 に設定されています。

オプションの圧縮された 16 ビット命令セット拡張は、00,01、または 10 に等しい最低 2 ビットを有しています。

32 ビット以上でエンコードされた標準命令セット拡張では、図 1.1 に示す 48 ビットおよび 64 ビットの長さ規則に従って、下位ビットが 1 に設定され追加されます。

80 ビットと 176 ビットの間の命令長は、最初の 5x16 ビットワードに加えて、16 ビットワードの数を与えるビット [14:12] の 3 ビットフィールドを使用して符号化されます。

ビット[14:12]が 111 にセットされた符号化は、将来のより長い命令符号化のために予約されています。

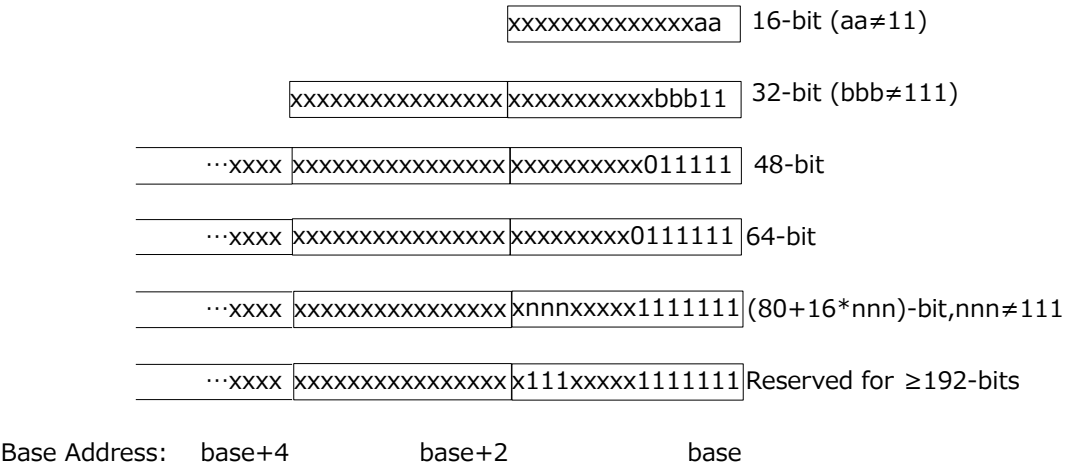


図 1.1 RISC-V 命令長符号化

↑要するに、命令コードは 16bit ずつの区切りで、下位ビットから見っていくと何ビットの命令かわかるよ、ってこと

圧縮されたフォーマットのコードサイズと省エネルギーを考えると、これを補足として追加するのではなく、ISA 符号化方式の圧縮フォーマットをサポートしたいと考えましたが、より簡単な実装を可能にするために、必須です。

また、実験とより大きな命令セット拡張をサポートするために、より長い命令をオプションで許可したいと考えました。

私たちの符号化規約では、コア RISC-V ISA の符号化がより厳密に要求されていましたが、これにはいくつかの有益な効果があります。

標準 G ISA の実装では、命令キャッシュに最上位 30 ビットしか保持する必要がありません（6.25%の節約）。命令キャッシュリフィルでは、不正な命令例外の動作を保持するためにキャッシュに格納する前に、ロービットクリアのいずれかの命令が不正 30bit 命令に記録される必要があります。

↑ここは何を言っているのかよくわからない。例外が起こった時には 30bit は保持してねということかな。???

おそらくもっと重要なのは、基本 ISA を 32 ビット命令語のサブセットに集約することによって、カスタム拡張で使用可能な空間が増えたことです。

第 21 章で説明したように、標準の圧縮命令拡張機能のサポートを必要としない実装では、 $\geq 32$  ビット命令セット拡張を超える標準のサポートを維持しながら、3 つの 30 ビット命令スペースを 32 ビットの固定幅フォーマットにマップできます。さらに、実装はまた、長さが  $> 32$  ビットを超える命令を必要としない場合、それはさらに 4 つの主要なオペコードを回復することができます。

-----  
すべてのゼロビットを含む任意の長さの命令が正当でないという特徴は、ゼロメモリ領域への誤ったジャンプを素早くトラップするためです。

同様に、プログラムされていない不揮発性メモリデバイス、切断されたメモリバス、または壊れたメモリデバイスで観察される他の一般的パターンを補足するために、すべてのものを含む命令符号化は不正命令として確保します。

ベース RISC-V ISA にはリトルエンディアンのメモリシステムがありますが、非標準のバリエーションではビッグエンディアンまたはバイエンディアンのメモリシステムを提供できます。

命令はメモリに格納され、各 16 ビット区画は、実装の自然なエンディアンに従ってメモリハーフワードに格納されます。

1 つの命令を形成する区画は、命令仕様の最も低い番号のビットを保持する最下位の区画を有します、すなわちメモリシステムのエンディアンに関係なく小規模の区画配列に常に格納されます。

図 1.2 のコードシーケンスは、メモリシステムのエンディアンに関係なく、32 ビット命令をメモリに正しく格納します。

// x2 レジスタの 32 ビット命令を x3 が指し示す位置に格納します。

sh x2, 0(x3) // 命令の下位ビットを第 1 区画に格納します。

srl x2, x2, 16 // 上位ビットを下位ビットに移動し、x2 を上書きします。

sh x2, 2(x3) // 上位ビットを 2 番目の区画に格納します。

図 1.2 : レジスタからメモリへの 32 ビット命令を格納する推奨コードシーケンス。

ビッグエンディアンとリトルエンディアンの両方のメモリシステムで正しく動作し、可変長命令セット拡張で 사용되는場合の誤整列アクセスを回避します。

---

リトルエンディアンシステムが現在商業的に支配的しているため、我々は RISC-V メモリシステムのためにリトルエンディアンバイトオーダリングを選択しました（すべての x86 システム、iOS、Android、Windows for ARM）。

目立たない点は、我々はハードウェア設計者にとっては、リトルエンディアンのメモリシステムがより自然なものであることがわかったということです。

しかし、IP ネットワーキングなどの特定のアプリケーション分野は、ビッグエンディアンのデータ構造で動作するため、非標準のビッグエンディアンまたはバイエンディアンの可能性を残しています。

長さエンコーディングビットが必ず最初にハーフワードアドレス順に現れるようにするために、メモリシステムエンディアンとは独立して命令区画がメモリに格納される順序を修正する必要があります。

これにより、最初の 16 ビット命令区画の最初の数ビットのみを調べることによって、命令フェッチユニットによって可変長命令の長さを迅速に決定することが可能になります。

リトルエンディアンのメモリシステムと命令区画の順序付けを修正した後は、命令形式の LSB 位置に長さエンコードビットを配置し、オペコードフィールドの分割を避けることができるようになりました。

### 1.3 例外、トラップ、および割り込み

現在の RISC-V スレッドの命令に関連付けられた実行時に発生する異常な状態を参照するために、例外という用語を使用します。

トラップという用語は、RISC-V スレッド内で発生する例外的条件によって引き起こされるトラップハンドラへの制御の同期転送を参照するために使用します。

トラップハンドラは通常より特権のある環境で実行されます。

現在の RISC-V スレッドとは非同期に発生する外部イベントを参照するために、割り込みという用語を使用します。

処理されなければならない割り込みが発生すると、割り込み例外を受け取るためにいくつかの命令が選択され、続いてトラップが発生します。

次章の命令説明では、実行中に例外が発生する条件について説明しています。

これらがトラップに変換されるかどうかは実行環境に依存しますが、例外が通知されたときにはほとんどの環境で正確なトラップが実行されることが予想されます（ただし、浮動小数点例外を除き、標準浮動小数点数の拡張は、トラップを引き起こしません）。

---

私たちの "例外" と "トラップ" の使用は、IEEE-754 浮動小数点標準と一致します。

-2018/04/14

1 ページ空き。次ページへ。

## 第2章

### RV32I ベース整数命令セット、 バージョン 2.0