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。

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 、Krste 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、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・テイラー、ウェズリー・ターブストラ、 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・ザンヂジク。

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 RV32E RV64I RV128I	2.0 1.9 2.0 1.7	Y N Y N
拡張	バージョン	凍結?
M A F D Q L C B J T P V N	2.0 2.0 2.0 2.0 2.0 0.0 2.0 0.0 0.0 0.0 0.1 0.2 1.1	Y Y Y Y Y N Y N N N N N 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 のところでててくるかな。

-2018/03/27

・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 仕様に取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の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アント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ーは 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アント 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

↑Atomic ってどういう意味だろう？ なんて訳すのがいいのか。

-2018/04/02

-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を可能にします。
・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アントをサポートする 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リ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的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、大きな懸案事項です。
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う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、間違った、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