第 2 巻：RISC-V 特権アーキテクチャ V1.10

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-privileged-v1.10.pdf です。

https://github.com/shibatchii/RISC-V
に置いてあります。

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。
Twitter: @shibatchii

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。訳した日付とか、集中力が切れた時に書くヨタ話とか。

2018/09/28 @shibatchii

RISC-V 命令セットマニュアル第 2 巻：特権付きアーキテクチャ
特権アーキテクチャバージョン 1.10
ドキュメントバージョン 1.10

警告！ このドラフト仕様は、RISC-V 財団によって標準として承認される前に変更される可能性があります。
編集者はこの仕様の今後の変更が順方向互換になるよう意図していますが、
この仕様書の実装が将来の標準に準拠しない可能性があります。

編集者：アンドリュー ウォーターマン 1、クレステ アサノビッチ 1,2
1SiFive Inc.、
2 カリフォルニア大学バークレー校 EECS 学科 CS 課
andrew@sifive.com、krste@berkeley.edu
2017 年 5 月 7 日

第２巻：RISC-V 特権アーキテクチャ V1.10

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、アレン・バウム、パオロ・ボンツィーニ、ルスラン・ブーキン、クリストファー・セリオ、デビッド・キスナー、アンソニー・コールター、パーマー・ダッベルト、モンテ・ダリムプリ、デニス・ファーガソン、マイク・フライシンガー、ジョン・ハウザー、デヴィッド・ホーナー、オロフ・ヨハンソン、リー・ユンサプ、アンドリュー・ルトミルスキー、ジョナサン・ノイシェファー、リシュユール・ニヒル、ステファン・オレア、アルバート・ウー、ジョン・オースターハウト、デビッド・パターソン、コリン・シュミット、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・ヴァンデウォーカー、メガン・ワックス、アンドリュー・ウォーターマン、レイノード・ザンディク

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２巻：特権アーキテクチャ、バージョン 1.10」、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。

--2018/10/12

第 2 巻：RISC-V 特権アーキテクチャ V1.10

序文

これは、RISC-V 特権アーキテクチャ提案のバージョン 1.10 です。
バージョン 1.9.1 からの変更点：

- 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。
- シャドー CSR アドレスに関する明示的な規則は、CSR スペースを再利用するために削除されました。
  シャドー CSRs は、必要に応じて追加することができます。
- mvendorid レジスタには、財団が提供するコードとは対照的に、コアプロバイダの JEDEC コードが含まれるようになりました。
  これにより、財団からの冗長性とオフロード作業が回避されます。 ←JEDEC コードってなんじゃろ？
- 割込み可能なスタック規律は単純化されています。
- 監督者モードとユーザモードで使用されるベース ISA を変更するためのオプションのメカニズムが mstatus CSR に追加されました。misa の以前に Base と呼ばれていたフィールドは、整合性のために MXL に改名されました。
- mstatus の追加の拡張状態のステータスフィールドを要約するために XS を使用することを明確にしました。
- mtvec および stvec CSRs には、オプションのベクタ割り込みのサポートが追加されています。
- mip CSR の SEIP ビットと UEIP ビットは、外部割り込みのソフトウェア注入をサポートするために再定義されています。
- mbadaddr レジスタはより一般的な mtval レジスタに含まれています。これにより、命令エミュレーションの速度を上げるために不正な命令フォルトで不良命令ビットを取り込むことができます。
- 仮想メモリ構成を sptbr（now satp）に移行する過程の一環として、マシンモードのベースと境界の変換と保護のスキームが仕様から削除されました。
  ベースとバインドされたスキームのモチベーションのいくつかは PMP レジスタでカバーされていますが、mstatus で利用可能なまま残っていれば、後で役立ちます。
- M モードのみ、または M モードと U モードの両方で U モードトラップをサポートしていないシステムでは、medeleg レジスタと mideleg レジスタは存在しませんでしたが、以前はゼロを返していました。
- 仮想メモリページ違反は、物理メモリアクセス例外とは別の mcause 値を持つようになりました。
  ページ違反例外は、PMA および PMP チェックによって生成された例外を委任することなく、
  S モードに委任できるようになりました。
- オプションの物理メモリ保護（PMP）方式が提案されています。
- 監督者の仮想メモリ構成は、mstatus レジスタから sptbr レジスタに移動されました。

したがって、sptbr レジスタは satp（監督者アドレスの転送と保護）に名前が変更され、広がった役割が反映されます。

↑レジスタ名は整理しとかないとどれがどれかわからなくなりそう。

- 改善された SFENCE.VMA 命令のために、SFENCE.VM 命令が削除されました。
- mstatus ビット MXR は sstatus を介して S モードにさらされています。
- sstatus の PUM ビットの極性は、MXR を含むコードシーケンスを短縮するために反転されています。
  ビットの名前が SUM に変更されました。
- ページテーブルエントリのハードウェア管理 Accessed ビットと Dirty ビットはオプションです。簡単な実装では、ソフトウェアをトラップして設定することができます。
- カウンタイネーブル方式が変更されたため、S モードはカウンタの U モードへの可用性を制御できます。
- S モードでの再帰的仮想化サポートに焦点を当てているため、H モードは削除されています。
- エンコーディング空間は予約されており、後で再利用することができます。
- S モード仮想メモリ管理操作をトラップして仮想化パフォーマンスを向上させるメカニズムが追加されました。
- 監督者バイナリインターフェイス（SBI）の章が削除され、別の仕様として維持することができます。

第 2 巻：RISC-V 特権アーキテクチャ V1.10

バージョン 1.9.1 の序文

これは、RISC-V 特権アーキテクチャ提案のバージョン 1.9.1 です。
バージョン 1.9 からの変更点：

- 解説セクションへの多数の追加と改良。
- 設定文字列の提案を変更して、デバイスツリーストリングやフラット化されたデバイスツリーなどのさまざまなフォーマットをサポートする検索プロセスを使用します。
- misa は、ベースおよびサポートされている ISA 拡張の変更をサポートするように、オプションで書き込み可能になりました。
  misa の CSR アドレスが変更されました。
- デバッグモードとデバッグ CSRs の説明が追加されました。
- ハードウェアパフォーマンス監視スキームを追加しました。
  既存のハードウェアカウンタの処理を簡素化し、カウンタの特権バージョンと対応するデルタレジスタを削除しました。
- ユーザーレベルの割り込みがある場合の SPIE の説明が修正されました。

↑misa ってなんだっけ。後で確認

-- 1 ページ空き、次ページへ

内容

↑冪等性 (とうべきせい) 「大雑把に言って、ある操作を１回行っても複数回行っても結果が同じであることをいう概念である。」って。なんか難しい言葉やね。初めて聞いた。プログラムでいうといつも同じ答えが出るってこと。あたりまえじゃんと思うけど、割り込み処理が入って意図せずメモリ内容買い換えられちゃったりすると崩れちゃう。

-- 2018/10/13

第 2 巻：RISC-V 特権アーキテクチャ V1.10

第 1 章

前書き

これは、RISC-V の特権付きアーキテクチャ記述文書の草案です。
フィードバックを歓迎します。
変更は最終リリース前に発生します。←行われます。位の意味か

このドキュメントでは、RISC-V 特権アーキテクチャについて説明します。特権命令やオペレーティングシステムの実行、外部デバイスの接続に必要な追加機能など、ユーザーレベルの ISA を超えた RISC-V システムのあらゆる側面をカバーしています。

私たちの意思決定に関する解説は、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。
↑riscv-spec-v2.2 にはこれ書いてなかったような。字下げしてあるところは経緯なので、飛ばしても OK だよって。あ、いや、すまん、書いてあった。

ここでは、このドキュメントで説明している特権レベルの設計全体を、ユーザーレベルの ISA を変更することなく、おそらく ABI を変更することなく、まったく異なる特権レベルの設計に置き換えることができます。
特に、この特権仕様は、既存の一般的なオペレーティングシステムを実行するように設計されており、従来のレベルベースの保護モデルを具体化しています。
代替の特権的な仕様は、他のより柔軟な保護ドメインモデルを具体化することができます。

1.1 RISC-V ハードウェアプラットフォームの用語

RISC-V ハードウェアプラットフォームには、他の非 RISC-V 互換コア、固定機能アクセラレータ、さまざまな物理メモリ構造、I / O デバイス、およびコンポーネントが通信できるようにする相互接続構造とともに、1 つまたは複数の RISC-V 互換プロセッシングコアを含めることができます。

コンポーネントが独立した命令フェッチユニットを含む場合、コンポーネントはコアと呼ばれます。
RISC-V 互換のコアは、マルチスレッド化により、複数の RISC-V 互換ハードウェアスレッドまたはハートをサポートしている可能性があります。←可能性があります。より サポートする場合があります。ぐらい？

RISC-V コアには、追加の特殊な命令セット拡張機能や追加されたコプロセッサが追加されている場合があります。
コプロセッサという用語は、RISC-V コアに接続されたユニットを指し、ほとんどが RISC-V 命令ストリームによってシーケンスされますが、追加のアーキテクチャ状態および命令セット拡張、および場合によってはプライマリ RISC-V 命令ストリームに対するいくつかの限定された自律性を含みます。

我々は、非プログラマブルな固定機能ユニットまたは自律的に動作することができますが、特定のタスクに特化されたコアのいずれかを指すように用語アクセラレータを使用しています。
RISC-V システムでは、多くのプログラム可能なアクセラレータが、専用の命令セット拡張および/またはカスタマイズされたコプロセッサを備えた RISC-V ベースのコアになると期待しています。
RISC-V アクセラレータの重要なクラスは I / O アクセラレータであり、メインアプリケーションコアから I / O 処理タスクの荷をおろします。←I/O 処理タスクを切り離す 位の意味か

RISC-V ハードウェアプラットフォームのシステムレベルの構成には、シングルコアマイクロコントローラから数千ノードの共有メモリの manycore サーバーノードのクラスタまでが含まれます。
小さなシステム・オン・チップでも、開発努力(作業)をモジュール化したり、サブシステム間の安全な分離を提供するために、マルチコンピュータおよび/またはマルチプロセッサの階層として構成することができます。←構造化される可能性があります。

このドキュメントでは、ユニプロセッサまたは共有メモリマルチプロセッサ内で動作する各ハードウェア（ハードウェアスレッド）が認識できる特権アーキテクチャに焦点を当てています。

-- 2018/10/14

1.2 RISC-V 特権ソフトウェアスタックの用語

このセクションでは、RISC-V のさまざまな可能な特権ソフトウェアスタックのコンポーネントを説明するために使用する用語について説明します。

図 1.1 は、RISC-V アーキテクチャでサポート可能なソフトウェアスタックのいくつかを示しています。
左側には、アプリケーション実行環境（AEE）上で実行される単一のアプリケーションのみをサポートするシンプルなシステムが示されています。
アプリケーションは、特定のアプリケーションバイナリインタフェース（ABI）で動作するようにコード化されています。
ABI には、サポートされているユーザーレベルの ISA と、AEE と対話する一連の ABI 呼び出しセットが含まれています。
AEE は、AEE の詳細を柔軟に適用できるように、アプリケーションから AEE の詳細を隠しています。←遮蔽している
同じ ABI を複数の異なるホスト OSs にネイティブに実装することも、異なるネイティブ ISA を持つマシン上で実行されるユーザー・モード・エミュレーション環境でサポートすることもできます。

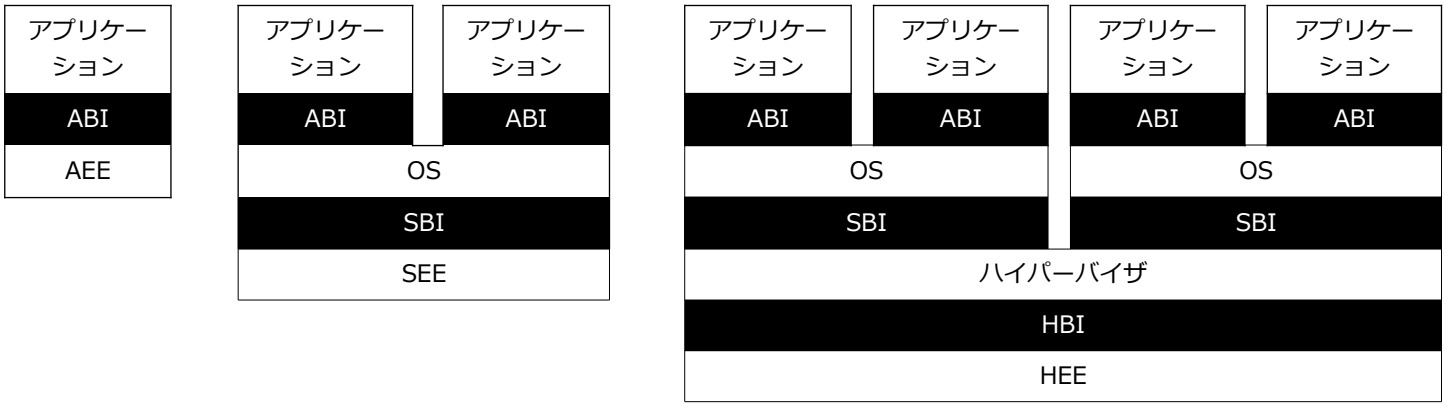

図 1.1：さまざまな形式の特権実行をサポートする異なる実装スタック

私たちのグラフィカルな表記法は、白いテキストの黒いボックスを使用して抽象的なインターフェイスを表し、インターフェイスを実装するコンポーネントの具体的なインスタンスからそれらを分けます。

中央の構成は、複数のアプリケーションのマルチプログラム実行をサポートできる従来のオペレーティングシステム（OS）を示しています。
各アプリケーションは、ABI を介して AEE を提供する OS と通信します。
アプリケーションが ABI 経由で AEE とインタフェースするのと同様に、RISC-V オペレーティングシステムは、監督者バイナリインタフェース（SBI）を介して監督者実行環境（SEE）とインタフェースします。
SBI は、ユーザレベルおよび監督者レベルの ISA と、一連の SBI ファンクションコールを備えています。←組み合わせて構成されています。

すべての SEE 実装で単一の SBI を使用することで、単一の OS バイナリイメージを任意の SEE で実行することができます。
SEE は、ローエンドのハードウェアプラットフォームで単純なブートローダと BIOS スタイルの IO システムにすることができ、またはハイエンドサーバにハイパーバイザが提供する仮想マシンを使用して、またはアーキテクチャシミュレーション環境でホストオペレーティングシステム上の薄い転送層を使用することができます。←シントランスレーションレイヤー

ほとんどの監督者レベルの ISA 定義では、SBI と実行環境および/またはハードウェアプラットフォームが分離されておらず、新しいハードウェアプラットフォームの仮想化および構築が複雑になります。

右端の構成は、複数のマルチプログラミングされた OSs が単一のハイパーバイザによってサポートされる仮想マシンモニタ構成を示しています。
各 OS は SBI を介して SEE を提供するハイパーバイザと通信します。
ハイパーバイザは、ハイパーバイザバイナリインターフェイス（HBI）を使用してハイパーバイザ実行環境（HEE）と通信し、ハードウェアプラットフォームの詳細からハイパーバイザを分離します。

ABI、SBI、および HBI はまだ進行中ですが、SBI が S モード OS によって再帰的に提供されるタイプ 2 ハイパーバイザーのサポートに優先順位を付けるようになっています。

RISC-V ISA のハードウェア実装では、通常、さまざまな実行環境（AEE、SEE、または HEE）をサポートするために、特権 ISA 以外の追加機能が必要になります。

1.3 特権レベル

いつでも、RISC-V ハードウェアスレッド（ハート）は、1 つまたは複数の CSR（コントロールおよびステータスレジスタ）のモードとしてエンコードされた権限レベルで実行されています。
3 つの RISC-V 特権レベルが現在表 1.1 に示すように定義されています。

レベル	エンコーディング	名前	略語
0	00	ユーザー/アプリケーション	U
1	01	スーパーバイザ(監督者)	S
2	10	予約	
3	11	マシン	M

表 1.1：RISC-V 特権レベル。

特権レベルは、ソフトウェアスタックのさまざまなコンポーネント間で保護を提供するために使用され、現在の特権モードで許可されていない操作を実行しようとすると、例外が発生します。
これらの例外は、通常、トラップを基になる実行環境に持ち込みます。←トラップを発生させます。

マシンレベルは最高の特権を持ち、RISC-V ハードウェアプラットフォームの唯一の必須の特権レベルです。
マシンモード（M モード）で実行されるコードは、マシン実装に対する低レベルのアクセス権を持つため、通常は本質的に信頼されています。
M モードは、RISC-V 上の安全な実行環境を管理するために使用できます。
ユーザモード（U モード）および監督者モード（S モード）は、それぞれ従来のアプリケーションおよびオペレーティングシステムの使用を意図しています。←対象としています。

タイプ 1 ハイパーバイザをサポートするように設計された以前のハイパーバイザモード（H モード）は削除され、そして、
第 5 章で説明したタイプ 1 とタイプ 2 の両方のハイパーバイザーに適した拡張 S モードを使用してハイパーバイザーのサポートに焦点を当てているため、予約されているエンコードスペースが必要です。

H のエンコーディング空間は、将来の使用のために予約されており、様々な状態記録者のビット位置の後方互換性のない変更を回避するために予約されています。
ビットポジションは、異なる Type-1 ハイパーバイザのサポート、または可能であれば付加的な安全な実行モードのために将来再利用される可能性があります。

各特権レベルには、オプションの拡張子とバリアント(変形、異形)を持つ特権 ISA 拡張のコアセットがあります。
たとえば、マシンモードでは、アドレス変換とメモリ保護のためのいくつかのオプションの標準バリアントがサポートされています。
また、第 5 章で説明したように、監督者モードを拡張してタイプ 2 ハイパーバイザの実行をサポートすることもできます。

表 1.2 に示すように、インプリメンテーションでは、実装コストを削減するために、1 つから 3 つの特権モードを削減することができます。

この説明では、コードが書き込まれる特権レベルを、それが実行される特権モードから分離しようとしますが、2 つは頻繁に結びついています。
たとえば、監督者レベルのオペレーティングシステムは、3 つの特権モードを持つシステム上で監督者モードで実行できますが、2 つ以上の特権モードを持つシステム上の従来の仮想マシンモニタでは、ユーザモードで実行することもできます。
いずれの場合も、同じ監督者レベルのオペレーティングシステムバイナリコードを使用し、監督者レベルの SBI にコード化し、監督者レベルの特権命令および CSR s を使用できるようにすることを期待しています。
ゲスト OS をユーザモードで実行する場合、すべての監督者レベルのアクションは、より高い特権レベルで実行されている SEE によってトラップとエミュレートされます。

レベル数	サポートされるモード	使用目的
1	M	シンプルな組み込みシステム
2	M, U	安全な組み込みシステム
3	M, S, U	Unix ライクなオペレーティングシステムを実行するシステム

表 1.2：サポートされている特権モードの組み合わせ

すべてのハードウェア実装は、M モードを提供する必要があります。これは、マシン全体に自由にアクセスできる唯一のモードであるためです。
最も単純な RISC-V 実装では、M モードのみが提供されますが、不正または悪意のあるアプリケーションコードに対する保護は提供されません。

オプションの PMP 設備のロック機能は、M モードのみが実装されていても、ある程度の保護を提供することができます。

多くの RISC-V 実装は、アプリケーションコードからシステムの残りの部分を保護するために、少なくともユーザーモード（U モード）もサポートします。
監督者・モード（S モード）を追加すると、監督者・レベルのオペレーティング・システムと SEE の間の分離が可能になります。

ハートは通常、あるトラップ（例えば監督者コールまたはタイマー割込み）が通常より特権モードで動作するトラップハンドラへの切り替えを強制するまで、U モードでアプリケーションコードを実行します。
ハートはトラップハンドラを実行し、最終的に U モードで元のトラップされた命令の実行後に実行を再開します。
特権レベルを上げるトラップは垂直トラップと呼ばれ、同じ特権レベルに留まるトラップは水平トラップと呼ばれます。
RISC-V 特権アーキテクチャは、異なる特権層へのトラップの柔軟なルーティングを提供します。

水平トラップは、特権レベルの低いモードで水平トラップハンドラに制御を戻す垂直トラップとして実装できます。

-- ABI,AEE,SBI,SEE とかの略語はどこかで表にしよう。

1.4 デバッグモード

実装はまた、オフチップデバッグおよび/または製造テストをサポートするためのデバッグモードを含むことができます。
デバッグモード（D モード）は、M モードよりもさらに多くのアクセス権を持つ追加の特権モードと考えることができます。
個別のデバッグ仕様の提案では、デバッグモードでの RISC-V ハートの動作について説明します。
デバッグモードでは、D モードでしかアクセスできないいくつかの CSR アドレスが予約されます。また、プラットフォーム上の物理メモリ領域の一部を予約することもできます。

-- 2018/10/21

-- 1 ページ空き、次ページへ

第 2 章

制御およびステータスレジスタ（CSR）

システムの主要なオペコードは、RISC-V ISA 内のすべての特権命令をエンコードするために使用されます。
これらは 2 つの主要なクラスに分けることができます：
これらは、アトミックにリード・モディファイ・ライト・コントロールおよびステータス・レジスタ（CSRs）、およびその他のすべての特権命令です。
このマニュアルの第 1 巻で説明されているユーザーレベルの状態に加えて、実装には追加の CSRs が含まれ、ユーザレベルのマニュアルに記載された CSR 命令を使用して権限レベルのいくつかのサブセットによってアクセス可能です。
この章では、CSR アドレス空間をマッピングします。
以下の章では、特権レベルによる各 CSRs の機能、および一般に特定の特権レベルと密接に関連するその他の特権命令について説明します。
CSR と命令は 1 つの特権レベルに関連付けられていますが、それらはまた、すべての上位の特権レベルでアクセス可能であることに注意してください。

2.1 CSR アドレスマッピング規則

標準の RISC-V ISA は、最大 4,096 の CSR に対して 12 ビットの符号化空間（csr [11：0]）を設定します。
表 2.1 に示すように、CSR アドレスの上位 4 ビット（csr [11：8]）を使用して、CSR の読み取りおよび書き込みアクセシビリティを特権レベルに従ってエンコードします。
上位 2 ビット（csr [11:10]）は、レジスタが読み/書き（00,01,10）か読み取り専用（11）かを示します。
次の 2 ビット（csr [9：8]）は、CSR にアクセスできる最低の特権レベルをエンコードします。

CSR アドレス規約では、CSR アドレスの上位ビットを使用して、規定のアクセス特権をエンコードします。
これにより、ハードウェアのエラーチェックが簡素化され、より大きな CSR スペースが提供されますが、CSR のアドレス空間へのマッピングが制限されます。

実装により、特権レベルでは、許可されていない CSR アクセスを低特権レベルでトラップして、これらのアクセスを傍受することができます。
この変更は、特権の低いソフトウェアにとっては透過的でなければなりません。

存在しない CSR にアクセスしようとすると、不正な命令例外が発生します。
適切な権限レベルを持たない CSR にアクセスしたり、読み取り専用レジスタを書き込んだりすると、不正な命令例外も発生します。
読み出し/書き込みレジスタには、読み出し専用ビットが含まれている場合があります。この場合、読み出し専用ビットへの書き込みは無視されます。

CSR アドレス			16 進	使用とアクセシビリティ
[11:10]	[9:8]	[7:6]		
ユーザー CSRs				
00	00	XX	0x000-0x0FF	標準 読み／書き
01	00	XX	0x400-0x4FF	標準 読み／書き
10	00	XX	0x800-0x8FF	非標準 読み／書き
11	00	00-10	0xC00-0xCBF	標準 読み取り専用
11	00	11	0xCC0-0xCFF	非標準 読み取り専用
監督者 CSRs				
00	01	XX	0x100-0x1FF	標準 読み／書き
01	01	00-10	0x500-0x5BF	標準 読み／書き
01	01	11	0x5C0-0x5FF	非標準 読み／書き
10	01	00-10	0x900-0x9BF	標準 読み／書き
10	01	11	0x9C0-0x9FF	非標準 読み／書き
11	01	00-10	0xD00-0xDBF	標準 読み取り専用
11	01	11	0xDC0-0xDFF	非標準 読み取り専用
予約 CSRs				
XX	10	XX	予約	
マシン CSRs				
00	11	XX	0x300-0x3FF	標準 読み／書き
01	11	00-10	0x700-0x79F	標準 読み／書き
01	11	10	0x7A0-0x7AF	標準 読み／書き デバッグ CSRs
01	11	10	0x7B0-0x7BF	デバッグモードのみ CSRs
01	11	11	0x7C0-0x7FF	非標準 読み／書き
10	11	00-10	0xB00-0xBBF	標準 読み／書き
10	11	11	0xBC0-0xBFF	非標準 読み／書き
11	11	00-10	0xF00-0xFBF	標準 読み取り専用
11	11	11	0xFC0-0xFFF	非標準 読み取り専用

表 2.1：RISC-V CSR アドレス範囲の割り当て。

表 2.1 はまた、標準と非標準の使用の間に CSR アドレスを割り当てる規則を示しています。
非標準的な使用のために予約された CSR アドレスは、将来の標準拡張によって再定義されません。

私たちは、シャドー CSRs のための CSR スペースの明示的な割り当てを廃止し、他の CSRs を割り当てた方がより柔軟になるようにしました。
シャドー CSRs は、適切な R / W 空間に追加することができます。
カウンタは、現在の仕様では唯一のシャドーされた CSR です。

シャドー CSRs は読み取り／書き込み・アドレスを提供し、より高い特権レベルでより低い特権レベルで読取り専用のレジスタを変更することができます。
1 つの特権レベルで既に読取り/書込みシャドウ・アドレスが割り当てられている場合は、より高い特権レベルであれば、同じレジスタへの読み取り/書き込みアクセスに同じ CSR アドレスを使用できます。

効果的な仮想化では、仮想化された環境内で可能な限り多くの命令をネイティブに実行する必要があり、特権アクセスは仮想マシンモニタ[1]にトラップします。
より低い特権レベルで読み取り専用の CSRs は、より高い特権レベルで読み書き可能にされている場合、別の CSR アドレスにシャドーイングされます。
これにより、許可された低特権アクセスのトラップを回避しながら、不正アクセスのトラップを引き起こします。←発生させます。

マシンモード標準の読み書き可能な CSR 0x7A0-0x7BF は、デバッグシステムで使用するために予約されています。
実装は、これらのレジスタへのマシンモードアクセスで不正な命令例外を発生させるべきです。←必要があります。

-- 2018/10/27

2.2 CSR リスト

表 2.2-2.5 に、現在 CSR アドレスが割り当てられている CSRs の一覧を示します。
タイマー、カウンタ、および浮動小数点 CSRs は、標準的なユーザレベルの CSR であり、N 拡張によって追加されたユーザトラップレジスタも追加されています。
他のレジスタは、次の章で説明するように、特権コードによって使用されます。
すべての実装ですべてのレジスタが必要というわけではないことに注意してください。

番号	特権	名前	説明
			ユーザートラップ設定
0x000	URW	ustatus	ユーザー状態レジスタ。
0x004	URW	uie	ユーザー割り込み有効レジスタ。
0x005	URW	utvec	ユーザトラップハンドラのベースアドレス。
			ユーザートラップ処理
0x040	URW	uscratch	ユーザトラップハンドラのためのスクラッチレジスタ。
0x041	URW	uepc	ユーザー例外プログラムカウンタ。
0x042	URW	ucause	ユーザトラップの原因。
0x043	URW	utval	ユーザーのアドレスまたは命令が不正。
0x044	URW	uip	ユーザー割り込み保留中。
			ユーザ浮動小数点 CSRs
0x001	URW	fflags	浮動小数点発生例外。
0x002	URW	frm	浮動小数点動的丸めモード。
0x003	URW	fcsr	浮動小数点制御および状態レジスタ(frm + fflags)。
			ユーザカウンタ/タイマ
0xC00	URO	cycle	RDCYCLE 命令のサイクルカウンタ。
0xC01	URO	time	RDTIME 命令のタイマ。
0xC02	URO	insret	RDINSTRET 命令の命令リタイヤカウンタ。
0xC03	URO	hpmcounter3	パフォーマンス監視カウンタ
0xC04	URO	hpmcounter4	パフォーマンス監視カウンタ
		：	
0xC1F	URO	hpmcounter31	パフォーマンス監視カウンタ
0xC80	URO	cycleh	cycle の上位 32 ビット、RV32I のみ
0xC81	URO	timeh	time の上位 32 ビット、RV32I のみ。
0xC82	URO	instreth	instret の上位 32 ビット、RV32I のみ。
0xC83	URO	hpmcounter3h	hpmcounter3 の上位 32 ビット、RV32I のみ。
0xC84	URO	hpmcounter4h	hpmcounter4 の上位 32 ビット、RV32I のみ。
		：	
0xC9F	URO	hpmcounter31h	hpmcounter31 の上位 32 ビット、RV32I のみ。

表 2.2：現在割り当てられている RISC-V ユーザーレベルの CSR アドレス。

番号	特権	名前	説明
			監督者トラップ設定
0x100	SRW	sstatus	監督者状態レジスタ。
0x102	SRW	sedeleg	監督者例外委任登録。
0x103	SRW	sideleg	監督者割り込み委譲レジスタ。
0x104	SRW	sie	監督者割り込み有効レジスタ。
0x105	SRW	stvec	監督者トラップハンドラのベースアドレス。
0x106	SRW	scountern	監督者カウンタ有効。
			監督者トラップの処理
0x140	SRW	sscratch	監督者トラップハンドラのスクラッチレジスタ。
0x141	SRW	sepc	監督者例外プログラムカウンタ。
0x142	SRW	scause	監督者トラップの原因。
0x143	SRW	stval	監督者の不良アドレスまたは命令。
0x144	SRW	sip	監督者割り込み保留中。
			監督者保護と翻訳
0x180	SRW	satp	監督者のアドレス変換と保護

表 2.3：現在割り当てられている RISC-V 監督者レベルの CSR アドレス。

-- scratch register ってなんだろ。scratch って削るとか、引っ掻くっていういみだよね。

番号	特権	名前	説明
			マシン情報レジスタ
0xF11	MRO	mvendorid	ベンダー ID。
0xF12	MRO	marchid	アーキテクチャ ID。
0xF13	MRO	mimpid	実装 ID。
0xF14	MRO	mhartid	ハードウェアスレッド ID。
			マシントラップ設定
0x300	MRW	mstatus	マシン状態レジスタ。
0x301	MRW	misa	ISA と拡張機能
0x302	MRW	medeleg	マシン例外委譲レジスタ。
0x303	MRW	mideleg	マシン割り込み委譲レジスタ。
0x304	MRW	mie	マシン割込み許可レジスタ。
0x305	MRW	mtvec	マシントラップハンドラのベースアドレス。
0x306	MRW	mcounteren	マシンカウンター有効。
			マシントラップの処理
0x340	MRW	mscratch	マシントラップハンドラのスクラッチレジスタ。
0x341	MRW	mepc	マシン例外プログラムカウンタ。
0x342	MRW	mcause	マシントラップ原因。
0x343	MRW	mtval	マシンのアドレスまたは命令が不正。
0x344	MRW	mip	マシン割り込み保留中。
			マシンの保護と翻訳
0x3A0	MRW	pmpcfg0	物理メモリ保護構成。
0x3A1	MRW	pmpcfg1	物理メモリ保護構成、RV32 のみ。
0x3A2	MRW	pmpcfg2	物理メモリ保護構成。
0x3A3	MRW	pmpcfg3	物理メモリ保護構成、RV32 のみ。
0x3B0	MRW	pmpaddr0	物理メモリ保護アドレスレジスタ。
0x3B1	MRW	pmpaddr1	物理メモリ保護アドレスレジスタ。
		：	：
0x3BF	MRW	pmpaddr15	物理メモリ保護アドレスレジスタ。

表 2.4：現在割り当てられている RISC-V マシンレベルの CSR アドレス。

番号	特権	名前	説明
		マシンカウンタ/タイマ	
0xB00	MRW	mcycle	マシンサイクルカウンタ。
0xB02	MRW	minstret	マシン命令 – 退役カウンタ。
0xB03	MRW	mhpmcounter3	マシン性能監視カウンタ。
0xB04	MRW	mhpmcounter4	マシン性能監視カウンタ。
		:	
0xB1F	MRW	mhpmcounter31	マシン性能監視カウンタ。
0xB80	MRW	mcycleh	mcycle の上位 32 ビット、RV32I のみ。
0xB82	MRW	minstreth	minstret の上位 32 ビット、RV32I のみ。
0xB83	MRW	mhpmcounter3h	mhpmcounter3 の上位 32 ビット、RV32I のみ。
0xB84	MRW	mhpmcounter4h	mhpmcounter4 の上位 32 ビット、RV32I のみ。
		:	
0xB9F	MRW	mhpmcounter31h	mhpmcounter31 の上位 32 ビット、RV32I のみ。
		マシンカウンタ設定	
0x323	MRW	mhpmevent3	マシン性能監視イベントセレクタ
0x324	MRW	mhpmevent4	マシン性能監視イベントセレクタ
0x33F	MRW	mhpmevent31	マシン性能監視イベントセレクタ
		デバッグ/追跡レジスタ（デバッグモードと共有）	
0x7A0	MRW	tselect	デバッグ/追跡トリガレジスタの選択。
0x7A1	MRW	tdata1	最初のデバッグ/追跡トリガデータレジスタ。
0x7A2	MRW	tdata2	第 2 のデバッグ/追跡トリガデータレジスタ。
0x7A3	MRW	tdata3	第 3 のデバッグ/ト追跡トリガデータレジスタ。
		デバッグモードレジスタ	
0x7B0	DRW	dcsr	デバッグ制御および状態レジスタ。
0x7B1	DRW	dpc	デバッグ PC。
0x7B2	DRW	dscratch	デバッグスクラッチレジスタ。

表 2.5：現在割り当てられている RISC-V マシンレベルの CSR アドレス。

2.3 CSR フィールドの仕様

以下の定義および略語は、CSR 内のフィールドの動作を指定する際に使用されます。

予約済み書き込み無視、読み込み無視値（WIRI） ← リードした値は無視してねってこと

読み取り専用と読み取り/書き込みレジスタの中には、将来の使用のために予約された読み取り専用フィールドがあるものもあります。
これらの予約済みの読み取り専用フィールドは、読み取り時に無視する必要があります。
これらのフィールドへの書き込みは、CSR 全体が読み取り専用でない限り、無効です。この場合、書き込みによって不正な命令例外が発生する可能性があります。
これらのフィールドは、レジスタ記述で WIRI とラベル付けされています。

予約済み書き込みは値を保持し、無視値を読み込みます（WPRI）

一部の読み取り/書き込みフィールド全体は、将来の使用のために予約されています。← some はいくつか、whole は全体 約合ってるるかな
ソフトウェアはこれらのフィールドから読み取った値を無視し、同じレジスタの他のフィールドに値を書き込むときにこれらのフィールドに保持されている値を保持する必要があります。←予約されてるので読んだ値は無視してね。書いた時に書き換わっちゃダメよってことかな。
これらのフィールドは、レジスタ記述で WPRI と表示されます。

ソフトウェアモデルを簡素化するために、CSR 内の以前に予約されたフィールドの将来互換性のある将来の定義は、非原子的な読み取り/修正/書き込みシーケンスが CSR の他のフィールドを更新するために使用される可能性に対処しなければならなりません。←対処する必要があります。
あるいは(また)、元の CSR 定義では、サブフィールドをアトミックにしか更新できない(アトミックにのみ更新できるように)ことを指定しなければならず、中間値が合法(有効)でない場合に問題となる 2 命令のクリアビット/セットビットシーケンスが一般的に必要となる可能性があります。← うーん、いまいちわからん

書き込み/読み取り 専用(のみ)の有効値（WLRL）

いくつかの読み取り/書き込み CSR フィールドは、可能なビットエンコーディングのサブセットのみの動作を指定し、その他のビットエンコーディングは予約されています。
ソフトウェアは、そのようなフィールドに正当な値以外を書き込むべきではなく、最後の書き込みが正当な値でないか、別の操作（例えばリセット）からレジスタが書き込まれていない限り、 レジスタを正しい値に設定します。
これらのフィールドは、レジスタの説明で WLRL というラベルが付けられています。

ハードウェアの実装では、サポートされている値を区別するのに十分な状態ビットを実装するだけで済みますが(実装する必要がありますが)、読み込み時にサポートされている値の完全な指定されたビットエンコーディングを常に返す必要があります。

命令がサポートされていない値を CSR フィールドに書き込もうとすると、実装は許可されますが、不正命令例外を発生させる必要はありません。
ハードウェア実装は、最後の書き込みが不正な値であったときに CSR フィールドの読み込みで任意のビットパターンを返すことができますが、返される値は前回書き込まれた値に依存します。

すべての値を書き込み、有効値を読み取り（WARL）

いくつかの読み取り/書き込み CSR フィールドは、ビットエンコーディングのサブセットに対してのみ定義されますが、読み取られるたびに正しい値を返すことを保証しながら任意の値を書き込むことができます。
CSR の記述に他に副作用がないと仮定すると、サポートされている値の範囲は、希望の設定を書き込んだり、その値が保持されているかどうかを調べることによって判断できます。
これらのフィールドは、レジスタの説明で WARL というラベルが付けられています。

実装では、サポートされていない値の WARL フィールドへの書き込みに関する例外は発生しません。
実装は、特定の不正な値が書き込まれた後、常に確定的に同じ正当な値を返す必要があります。

-- 2018/10/28

第２巻：RISC-V 特権アーキテクチャ V1.10

第３章

マシンレベル ISA、バージョン 1.10

--ここまで。これ以降はこれから翻訳。 @shibatchii