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。

原文のライセンス表示

”The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version

1.10", Editors Andrew Waterman and Krste Asanovi#c, RISC-V Foundation, May 2017.

Creative Commons Attribution 4.0 International License

この日本語訳のライセンスも原文のライセンスを引き継いで

RISC-V命令セットマニュアル 第二巻：特権付きアーキテクチャ、文書 1.10版 日本語訳 @shibatchii

Creative Commons Attribution 4.0 International License

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

2019/07/27

@Roy\_n\_royさんより誤記指摘の連絡がありましたので修正しました。@Roy\_n\_royさんありがとうございます。

・レジスタ名の一部で「counteren」が「countern」となっている箇所があった。

・「図4.14 Sv32物理アドレス」が34bit長であるところ、32bitとなっていた。

・「表4.4 PTE R / W / Xフィールドの符号化」の下4行、X(実行)の値が 1 でなく 0 となっていた。

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

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・タープストラ、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・ザンデイク

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特権仕様バージョン1.9.1」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、 リマス・アヴィジエニス、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

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。

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

序文

1 はじめに　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　1

1.1 RISC-Vハードウェアプラットフォームの用語。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 1

1.2 RISC-V特権ソフトウェアスタック用語。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 2

1.3 特権レベル。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 3

1.4 デバッグモード。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 5

2 制御およびステータスレジスタ（CSR）　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　7

2.1 CSRアドレスマッピング規則。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 7

2.2 CSRリスト。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 9

2.3 CSR分野の仕様。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 13

3 マシンレベルISA、バージョン1.10 　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　15

3.1マシンレベルのCSR。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。。 。 。 。 。 。 。 。 。 。 。 。 15

3.1.1 マシンISAレジスタ misa。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 15

3.1.2 マシンベンダID mvendoridレジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 18

3.1.3 マシンアーキテクチャID marchidレジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 18

3.1.4 マシン実装ID mimpidレジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 19

3.1.5 ハートID mhartidレジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 19

3.1.6 マシンステータスレジスタ（mstatus）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 19

3.1.7 mstatusレジスタの特権およびグローバル割り込みイネーブルスタック。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 20

3.1.8 mstatusレジスタのベースISA制御。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 21

3.1.9 mstatusレジスタのメモリ特権。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 22

3.1.10 mstatusレジスタの仮想化サポート。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 22

3.1.11 mstatusレジスタの拡張コンテキスト状態。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 23

3.1.12 マシントラップベクトルベースアドレスレジスタ（mtvec）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 26

3.1.13 マシントラップ委任レジスタ（medelegとmideleg）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 27

3.1.14 マシン割り込みレジスタ（mipとmie）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 28

3.1.15 マシンタイマレジスタ（mtimeとmtimecmp）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 30

3.1.16 ハードウェアパフォーマンスモニタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 31

3.1.17 カウンタイネーブルレジスタ（[m | h | s] counteren）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 32

3.1.18 マシンスクラッチレジスタ（mscratch）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 33

3.1.19 マシン例外プログラムカウンタ（mepc）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 34

3.1.20 マシン要因レジスタ（mcause）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 34

3.1.21 マシントラップ値（mtval）レジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 35

3.2 マシンモード特権命令。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 37

3.2.1 環境呼び出しとブレークポイント。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 37

3.2.2 トラップリターン命令。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 37

3.2.3 割り込みを待ち。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 38

3.3 リセット。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 39

3.4 マスカブルでない割り込み。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 39

3.5 物理メモリ属性。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 39

3.5.1 メインメモリとI / Oと空レジスタの比較。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 41

3.5.2 サポートされているアクセスタイプPMAs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 41

3.5.3 原子性PMAs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 41

3.5.4 メモリオーダーPMAs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 42

3.5.5 一貫性とキャッシュ可能性のPMAs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 43

3.5.6 冪等性 PMAs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 44

3.6 物理メモリ保護。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 44

↑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。

3.6.1 物理メモリ保護のCSRs。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 45

4 監督者レベルISA、バージョン1.10 　　　 　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　49

4.1 監督者CSRs。 。。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 49

4.1.1 監督者状態レジスタ（sstatus）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 49

4.1.2 sstatusレジスタのベースISA制御。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 50

4.1.3 sstatusレジスタのメモリ特権。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 51

4.1.4 監督者トラップベクタベースアドレスレジスタ（stvec）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 51

4.1.5 監督者割り込みレジスタ（sip and sie）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 52

4.1.6 監督者タイマとパフォーマンスカウンタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 53

4.1.7 カウンタイネーブルレジスタ（scounteren）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 53

4.1.8 監督者スクラッチレジスタ（スクラッチ）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 54

4.1.9 監督者例外プログラムカウンタ（sepc）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 54

4.1.10 監督者原因登録（scause）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 54

4.1.11 監督者トラップ値（stval）レジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 55

4.1.12 監督者アドレス変換および保護（satp）レジスタ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 56

4.2 監督者命令。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 58

4.2.1 監督者メモリ管理フェンス命令。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 58

4.3 Sv32：ページベースの32ビット仮想メモリシステム。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 59

4.3.1 アドレッシングとメモリ保護。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 59

4.3.2 仮想アドレス変換プロセス。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 62

4.4 Sv39：ページベースの39ビット仮想メモリシステム。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 62

4.4.1 アドレッシングとメモリ保護。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 63

4.5 Sv48：ページベースの48ビット仮想メモリシステム。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 63

4.5.1アドレッシングとメモリ保護。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 64

5 ハイパーバイザー 拡張、バージョン0.0 　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　65

6 RISC-V特権命令セットのリスト　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　67

7 プラットフォームレベル割り込みコントローラ（PLIC）　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　69

7.1 PLICの概要。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 69

7.2 割込みソース。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 69

7.2.1ロ ーカル割込みソース。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 70

7.2.2 グローバル割込みソース。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 71

7.3 割り込みターゲットとハートコンテキスト。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 71

7.4 割り込みゲートウェイ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 71

7.5 割込み識別子（ID）。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 72

7.6 割り込みの優先順位。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 72

7.7 割り込みが有効。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 73

7.8 割り込み優先度しきい値。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 73

7.9 割り込み通知。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 74

7.10 割り込みクレーム。 。 。 。。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 74

7.11 割り込み完了。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 75

7.12 割り込みの流れ。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 75

7.13 PLICコア仕様。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 76

7.14 PLICへのアクセスの制御。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 76

8 マシン設定説明　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　77

8.1 設定文字列検索手順。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 77

9 ヒストリー　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　79

9.1 UC Berkeleyにおける研究資金。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 79

-- 2018/10/13

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。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| アプリケーション |  | アプリケーション |  | アプリケーション |  | アプリケーション |  | アプリケーション |  | アプリケーション |  | アプリケーション |
| ABI |  | ABI |  | ABI |  | ABI |  | ABI |  | ABI |  | ABI |
| AEE |  | OS | | |  | OS | | |  | OS | | |
|  |  | SBI | | |  | SBI | | |  | SBI | | |
|  |  | SEE | | |  | ハイパーバイザ | | | | | | |
|  |  |  |  |  |  | HBI | | | | | | |
|  |  |  |  |  |  | HEE | | | | | | |

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  1  2  3 | 00  01  10  11 | ユーザー/アプリケーション  スーバーバイザ(監督者)  予約  マシン | U  S  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CSRｓ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  2  3 | M  M, U  M, S, U | シンプルな組み込みシステム  安全な組み込みシステム  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  01  10  11  11 | 00  00  00  00  00 | XX  XX  XX  00-10  11 | 0x000-0x0FF  0x400-0x4FF  0x800-0x8FF  0xC00-0xCBF  0xCC0-0xCFF | 標準 読み／書き  標準 読み／書き  非標準 読み／書き  標準 読み取り専用  非標準 読み取り専用 |
| 監督者 CSRs | | | | |
| 00  01  01  10  10  11  11 | 01  01  01  01  01  01  01 | XX  00-10  11  00-10  11  00-10  11 | 0x100-0x1FF  0x500-0x5BF  0x5C0-0x5FF  0x900-0x9BF  0x9C0-0x9FF  0xD00-0xDBF  0xDC0-0xDFF | 標準 読み／書き  標準 読み／書き  非標準 読み／書き  標準 読み／書き  非標準 読み／書き  標準 読み取り専用  非標準 読み取り専用 |
| 予約 CSRs | | | | |
| XX | 10 | XX | 予約 |  |
| マシン CSRs | | | | |
| 00  01  01  01  01  10  10  11  11 | 11  11  11  11  11  11  11  11  11 | XX  00-10  10  10  11  00-10  11  00-10  11 | 0x300-0x3FF  0x700-0x79F  0x7A0-0x7AF  0x7B0-0x7BF  0x7C0-0x7FF  0xB00-0xBBF  0xBC0-0xBFF  0xF00-0xFBF  0xFC0-0xFFF | 標準 読み／書き  標準 読み／書き  標準 読み／書き デバッグ CSRs  デバッグモードのみ CSRs  非標準 読み／書き  標準 読み／書き  非標準 読み／書き  標準 読み取り専用  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  0x004  0x005 | URW  URW  URW | ustatus  uie  utvec | ユーザー状態レジスタ。  ユーザー割り込み有効レジスタ。  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  0x041  0x042  0x043  0x044 | URW  URW  URW  URW  URW | uscratch  uepc  ucause  utval  uip | ユーザトラップハンドラのためのスクラッチレジスタ。  ユーザー例外プログラムカウンタ。  ユーザトラップの原因。  ユーザーのアドレスまたは命令が不正。  ユーザー割り込み保留中。 |
| ユーザ浮動小数点CSRs | | | |
| 0x001  0x002  0x003 | URW  URW  URW | fflags  frm  fcsr | 浮動小数点発生例外。  浮動小数点動的丸めモード。  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  0xC01  0xC02  0xC03  0xC04  0xC1F  0xC80  0xC81  0xC82  0xC83  0xC84  0xC9F | URO  URO  URO  URO  URO  URO  URO  URO  URO  URO  URO  URO | cycle  time  insret  hpmcounter3  hpmcounter4  ：  hpmcounter31  cycleh  timeh  instreth  hpmcounter3h  hpmcounter4h  ：  hpmcounter31h | RDCYCLE命令のサイクルカウンタ。  RDTIME命令のタイマ。  RDINSTRET命令の命令リタイヤカウンタ。  パフォーマンス監視カウンタ  パフォーマンス監視カウンタ  パフォーマンス監視カウンタ  cycleの上位32ビット、RV32Iのみ  timeの上位32ビット、RV32Iのみ。  instretの上位32ビット、RV32Iのみ。  hpmcounter3の上位32ビット、RV32Iのみ。  hpmcounter4の上位32ビット、RV32Iのみ。  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  0x102  0x103  0x104  0x105  0x106 | SRW  SRW  SRW  SRW  SRW  SRW | sstatus  sedeleg  sideleg  sie  stvec  scounteren | 監督者状態レジスタ。  監督者例外委任登録。  監督者割り込み委譲レジスタ。  監督者割り込み有効レジスタ。  監督者トラップハンドラのベースアドレス。  監督者カウンタ有効。 |
| 監督者トラップの処理 | | | |
| 0x140  0x141  0x142  0x143  0x144 | SRW  SRW  SRW  SRW  SRW | sscratch  sepc  scause  stval  sip | 監督者トラップハンドラのスクラッチレジスタ。  監督者例外プログラムカウンタ。  監督者トラップの原因。  監督者の不良アドレスまたは命令。  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  0xF12  0xF13  0xF14 | MRO  MRO  MRO  MRO | mvendorid  marchid  mimpid  mhartid | ベンダーID。  アーキテクチャID。  実装ID。  ハードウェアスレッドID。 |
| マシントラップ設定 | | | |
| 0x300  0x301  0x302  0x303  0x304  0x305  0x306 | MRW  MRW  MRW  MRW  MRW  MRW  MRW | mstatus  misa  medeleg  mideleg  mie  mtvec  mcounteren | マシン状態レジスタ。  ISAと拡張機能  マシン例外委譲レジスタ。  マシン割り込み委譲レジスタ。  マシン割込み許可レジスタ。  マシントラップハンドラのベースアドレス。  マシンカウンター有効。 |
| マシントラップの処理 | | | |
| 0x340  0x341  0x342  0x343  0x344 | MRW  MRW  MRW  MRW  MRW | mscratch  mepc  mcause  mtval  mip | マシントラップハンドラのスクラッチレジスタ。  マシン例外プログラムカウンタ。  マシントラップ原因。  マシンのアドレスまたは命令が不正。  マシン割り込み保留中。 |
| マシンの保護と翻訳 | | | |
| 0x3A0  0x3A1  0x3A2  0x3A3  0x3B0  0x3B1  0x3BF | MRW  MRW  MRW  MRW  MRW  MRW  MRW | pmpcfg0  pmpcfg1  pmpcfg2  pmpcfg3  pmpaddr0  pmpaddr1  ：  pmpaddr15 | 物理メモリ保護構成。  物理メモリ保護構成、RV32のみ。  物理メモリ保護構成。  物理メモリ保護構成、RV32のみ。  物理メモリ保護アドレスレジスタ。  物理メモリ保護アドレスレジスタ。  ：  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  0xB02  0xB03  0xB04  0xB1F  0xB80  0xB82  0xB83  0xB84  0xB9F | MRW  MRW  MRW  MRW  MRW  MRW  MRW  MRW  MRW  MRW | mcycle  minstret  mhpmcounter3  mhpmcounter4  ：  mhpmcounter31  mcycleh  minstreth  mhpmcounter3h  mhpmcounter4h  ：  mhpmcounter31h | マシンサイクルカウンタ。  マシン命令 – 退役カウンタ。  マシン性能監視カウンタ。  マシン性能監視カウンタ。  マシン性能監視カウンタ。  mcycleの上位32ビット、RV32Iのみ。  minstretの上位32ビット、RV32Iのみ。  mhpmcounter3の上位32ビット、RV32Iのみ。  mhpmcounter4の上位32ビット、RV32Iのみ。  mhpmcounter31の上位32ビット、RV32Iのみ。 |
| マシンカウンタ設定 | | | |
| 0x323  0x324  0x33F | MRW  MRW  MRW | mhpmevent3  mhpmevent4  mhpmevent31 | マシン性能監視イベントセレクタ  マシン性能監視イベントセレクタ  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  0x7A1  0x7A2  0x7A3 | MRW  MRW  MRW  MRW | tselect  tdata1  tdata2  tdata3 | デバッグ/追跡トリガレジスタの選択。  最初のデバッグ/追跡トリガデータレジスタ。  第2のデバッグ/追跡トリガデータレジスタ。  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  0x7B1  0x7B2 | DRW  DRW  DRW | dcsr  dpc  dscratch | デバッグ制御および状態レジスタ。  デバッグPC。  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3章

マシンレベルISA、バージョン1.10

この章では、マシンモード（Mモード）で使用可能なマシンレベルの操作について説明します。これは、RISC-Vシステムで最高の権限モードです。

RISC-Vハードウェア実装では、Mモードのみが必須(唯一)の特権モードです。

Mモードは、ハードウェアプラットフォームへの低レベルアクセスに使用され、リセット時に最初に入力されるモードです。

Mモードは、ハードウェアに直接実装するにはあまりにも困難または高価な機能を実装するためにも使用できます。

RISC-VマシンレベルのISAには、サポートされている他の特権レベルとハードウェア実装のその他の詳細に応じて拡張された共通のコアが含まれています。

3.1マシンレベルのCSR

このセクションで説明するマシンレベルのCSRに加えて、Mモードコードはより低い特権レベルですべてのCSRにアクセスできます。

3.1.1マシンISAレジスタミサ

ミサCSRは、ハートでサポートされているISAを報告するXLENビットのWARL読み書きレジスタです。

このレジスタはどの実装でも読み込み可能でなければなりませんが、misaレジスタが実装されていないことを示すために0の値が返されるため、別個の非標準メカニズムによってCPU能力を判断する必要があります。

↑ゼロが返されるとmisaレジスタが実装されていないことがわかる。

XLEN-1 XLEN-2 XLEN-3 26 25 0

|  |  |  |
| --- | --- | --- |
| MXL[1:0] (WARL) | WIRI | Extensions[25:0] (WARL) |

2 XLEN-28 26

図3.1：マシンISAレジスタ（ミサ）。

MXL（Machine XLEN）フィールドは、表3.1に示すようにネイティブベースの整数ISA幅をエンコードします。

MXLフィールドは、複数のベースISA幅をサポートする実装では書き込み可能です。

Mモードの有効XLEN、M-XLENはMXLの設定で与えられ、ミサがゼロの場合は固定値です。

MXLフィールドは、リセット時に常にサポートされている最も広いISAバリアントに設定されます。

|  |  |
| --- | --- |
| MXL | XLEN |
| 1  2  3 | 32  64  128 |

表3.1：ミサのXMLフィールドのエンコーディング

-----------------------------------------------------------------------------------------------------------------------------------

ベース幅は返されたミサ値の符号による分岐と、場合によっては１つだけ左にシフトと、符号による2つ目の分岐をすることによって、迅速に確認できます。

↑ここはsignを符号とするか、なにか目印、標識のような意味にするかでちょと迷う。

これらのチェックは、マシンのレジスタ幅（XLEN）を知らずにアセンブリコードで記述することができます。

ベース幅はXLEN = 2MXL + 4で与えられます。

ミサがゼロの場合は、即値4をレジスタに置き、レジスタを一度に31ビットだけ左にシフトすることによって、ベース幅を見つけることもできます。

1回シフトした後にゼロになると、マシンはRV32になります。

2回シフトした後にゼロになると、マシンはRV64、それ以外の場合はRV128になります。

MXLがサポートされているXLENよりも小さい値に設定されている場合、すべての演算は、設定されたXLENよりも上位のソース・オペランド・レジスタ・ビットを無視し、サポートされている最も広いXLENをデスティネーション・レジスタに埋めるために結果を符号拡張する必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

実実装では、実装で定義された動作を避けるために、常に、基本ハードウェア・レジスタ全体を定義された値で埋める必要があります。

拡張フィールドは、アルファベット1文字あたり1ビットのビットを有する標準拡張の存在を符号化する（ビット0は拡張子「A」の存在を符号化し、ビット1は拡張子「B」を符号化し、「Z」を符号化するビット25まで符号化する） ）。

"I"ビットはRV32I、RV64I、RV128IベースISAに設定され、 "E"ビットはRV32Eに設定されます。

拡張機能は、実装がサポートされているISAを変更できるようにする書き込み可能なビットを含むことができるWARLフィールドです。

リセット時に、拡張フィールドにはサポートされている拡張機能の最大セットが含まれている必要があり、両方が利用可能であるなら、私はEより上を選ぶべきです。

"G"ビットはエスケープとして使用され、標準拡張名のより大きなスペースに拡張することができます。

-----------------------------------------------------------------------------------------------------------------------------------

Gは組み合わせIMAFDを示すために使用されるため、ミサCSRには冗長であるため、追加標準拡張が存在することを示すビットを予約します。

"U"ビットと "S"ビットは、それぞれユーザモードとスーパバイザモードがサポートされている場合に設定されます。

非標準の拡張があれば "X"ビットがセットされます。

-----------------------------------------------------------------------------------------------------------------------------------

misa CSRは、CPU機能の基本カタログをマシンモードコードに公開します。

マシンモードでは、他のマシンレジスタを調査し、ブートプロセスの一部としてシステム内の他のROMストレージを調べることにより、より詳細な情報を得ることができます。

各特権レベルで利用可能な機能を決定するために、CPU レジスタを読み取る代わりに、低い特権レベルで環境呼び出しを実行する必要があります。

これにより、仮想化レイヤーはあらゆるレベルで観察されるISAを変更でき、ハードウェア設計に負担をかけることなく、より豊富なコマンドインターフェイスをサポートします。

|  |  |  |
| --- | --- | --- |
| ビット | 文字 | 説明 |
| 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z | 原子の拡張  ビット演算拡張用に仮予約  圧縮拡張  倍精度浮動小数点拡張  RV32EベースISA  単精度浮動小数点拡張  追加の標準拡張機能が存在  予約  RV32I/64I/128I ベース ISA  動的に翻訳された言語拡張のために仮予約  予約  10進浮動小数点数の拡張のために仮予約  整数乗算/除算拡張  サポートされているユーザーレベルの割り込み  予約  パック SIMD 拡張の仮予約  四倍精度浮動小数点数の拡張  予約  監督者モードの実装  トランザクションメモリ拡張用に仮予約  ユーザーモードの実装  ベクター拡張のために仮予約  予約  標準以外の拡張機能が存在  予約  予約 |

表3.2：ミサの拡張フィールドのエンコード。将来使用するために予約されているすべてのビットは、読み込み時にゼロを返す必要があります。

3.1.2マシンベンダーIDレジスタ mvendorid

mvendorid CSRは、コアのプロバイダのJEDEC製造元IDを提供するXLENビットの読み取り専用レジスタです。

このレジスタは、どの実装でも読み込み可能でなければなりませんが、フィールドが実装されていないこと、または非商用実装であることを示すために値0を返すことができます。

XLEN-1 7 6 0

|  |  |
| --- | --- |
| バンク | オフセット |

XLEN-7 7

図3.2：ベンダーIDレジスタ（mvendorid）。

JEDEC製造者IDは、通常、各バイトの最上位ビットに奇数パリティビットを有する0x7fに等しくない1バイトIDによって終端される1バイト連続コード0x7fのシーケンスとして符号化されます。

mvendoridは、バンクフィールドの1バイトの連続コードの数をエンコードし、オフセットフィールドの最後のバイトをエンコードして、パリティビットを破棄します。

たとえば、JEDEC製造元ID 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x7f 0x8a（12個の継続コードの後ろに0x8a）は、mvendoridフィールドに0x60aとしてエンコードされます。

↑0x60aは 0x7fの数が12、16進で0xc。8aの最上位パリティを飛ばすと0x0a、0xcをパリティの位置から上に詰めると06a。

うーん、合ってるだろうか。

-----------------------------------------------------------------------------------------------------------------------------------

以前はベンダーIDはRISC-V財団によって割り当てられた番号でしたが、これはJEDECが製造者ID標準を維持する作業を複製しています。

執筆時点では、JEDECにメーカーIDを登録するには1回限りの500ドルのコストがかかります。

3.1.3マシンアーキテクチャIDレジスタmarchid

marchid CSRは、ハートのベースマイクロアーキテクチャをエンコードするXLENビットの読み取り専用レジスタです。

このレジスタはどの実装でも読み込み可能でなければなりませんが、フィールドが実装されていないことを示す値0を返すことができます。

mvendoridとmarchidの組み合わせは、実装されているハートマイクロアーキテクチャのタイプを一意に識別する必要があります。

XLEN-1 0

|  |
| --- |
| アーキテクチャ ID |

XLEN

図3.3：マシンアーキテクチャIDレジスタ（marchid）。

オープンソースのプロジェクトアーキテクチャIDは、RISC-V 財団によってグローバルに割り当てられ、ゼロ以外のアーキテクチャIDとゼロの最上位ビット（MSB）を持っています。

商用アーキテクチャIDは、各商用ベンダーによって個別に割り当てられますが、MSBが設定されていて、残りのXLEN-1ビットにゼロを含めることはできません。

↑ 残りのXLEN-1ビットにゼロを含めることはできません って All 1 なん？ ID区別できんじゃんって思ったけど、ここはXLEN-1は0にしてはダメってことだと思う。

-----------------------------------------------------------------------------------------------------------------------------------

その目的は、アーキテクチャーIDが、特定の組織ではなく開発が行われるレポに関連するマイクロアーキテクチャーを表すことです。

オープンソース設計の商業的製作では、元のアーキテクチャIDを保持する必要があります。（ライセンスによって必要となる可能性があります）

これにより、断片化やツールサポートのコストを削減するだけでなく、帰属を提供するのに役立ちます。

オープンソースのアーキテクチャーIDは財団によって管理されるべきであり、リリースされ機能しているオープンソースプロジェクトにのみ割り当てられるべきです。

商用アーキテクチャーIDは、登録されたベンダーによって個別に管理することができますが、ベンダーがクローズドソースとオープンソースの両方のマイクロアーキテクチャーを使用したい場合には、オープンソースアーキテクチャーID（MSBセット）とは別のIDが必要です。

次の実装フィールド内で採用された規約は、組織を含む同じアーキテクチャ設計のブランチを分離するために使用できます。

ミサレジスタはまた、構成文字列が存在する場合、デザインの異なるバリアントを区別するのにも役立ちます。

3.1.4マシン実装IDレジスタmimpid

mimpid CSRは、プロセッサ実装のバージョンを一意にエンコードします。

このレジスタはどの実装でも読み込み可能でなければなりませんが、フィールドが実装されていないことを示すために値0を返すことができます。

実装の値は、RISC-Vプロセッサ自体の設計を反映するものであり、周囲のシステムではありません。

↑この文はGoogle翻訳とBing翻訳で真逆の訳をする。多分こっちのGoogle翻訳が正しいと思う。

XLEN-1 0

|  |
| --- |
| 実装 |

XLEN

図3.4：マシン実装IDレジスタ（mimpid）。

-----------------------------------------------------------------------------------------------------------------------------------

このフィールドのフォーマットは、アーキテクチャーのソースコードのプロバイダーに委ねられていますが、先行または後続の0を持たない16進文字列として標準ツールによって印刷されることがよくありますが、人間の可読性を容易にするために、インプリメンテーション値をニブル境界で整列されたサブフィールドで左揃えにすることができます。  （すなわち、最も重要なニブルから下に埋められる）

3.1.5ハートIDレジスタmhartid

mhartid CSRは、コードを実行するハードウェアスレッドの整数IDを含むXLENビットの読み取り専用レジスタです。

このレジスタは、どの実装でも読み込み可能でなければなりません。

ハートIDは必ずしもマルチプロセッサーシステムでは連続して番号付けされるとは限りませんが、少なくとも1つのハートはハートIDがゼロでなければなりません。

XLEN-1 0

|  |
| --- |
| ハート ID |

XLEN

図3.5：ハートIDレジスタ（mhartid）。

-----------------------------------------------------------------------------------------------------------------------------------

特定のケースでは、正確に1つのハートが（例えばリセット時に）いくつかのコードを実行することを保証するようにする必要があります。したがって、1つのハートには既知のハートIDがゼロである必要があります。

効率のために、システム実装者は、システムで使用される最大のhartIDの大きさを減らすことを目指すべきです。

3.1.6マシンステータスレジスタ（mstatus）

mstatusレジスタは、RV32では図3.6、RV64とRV128では図3.7のようにフォーマットされたXLENビットの読み書き可能レジスタです。

mstatusレジスタは、ハートの現在の動作状態を追跡し、制御します。

mstatusレジスタの制限されたビューは、それぞれSレベルおよびUレベルのISAにsstatusおよびustatusレジスタとして表示されます。

31 30 23 22 21 20 19 18 17

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SD | WPRI | TSR | TW | TVM | MXR | SUM | MPRV |  |

1 8 1 1 1 1 1 1

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | XS[1:0] | FS[1:0] | MPP[1:0] | WPRI | SPP | MPIE | WPRI | SPIE | UPIE | MIE | WPRI | SIE | UIE |

2 2 2 2 1 1 1 1 1 1 1 1 1

図3.6：RV32のマシンモードステータスレジスタ（mstatus）

XLEN-1 XLEN-2 36 35 34 33 32 31 23 22 21 20 19 18 17

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SD | WPRI | SXL[1:0] | UXL[1:0] | WPRI | TSR | TW | TVM | MXR | SUM | MPRV |  |

1 XLEN-37 2 2 9 1 1 1 1 1 1

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | XS[1:0] | FS[1:0] | MPP[1:0] | WPRI | SPP | MPIE | WPRI | SPIE | UPIE | MIE | WPRI | SIE | UIE |

2 2 2 2 1 1 1 1 1 1 1 1 1

図3.7：RV64とRV128のマシンモードステータスレジスタ（mstatus）

3.1.7 mstatusレジスタの特権とグローバル割り込みイネーブルスタック

割込み許可ビット、MIE、SIE、およびUIEは、特権モードごとに提供されます。

これらのビットは、現在の特権レベルで割り込みハンドラに関するアトミック性を保証するために主に使用されます。

ハートが特権モードxで実行されている場合、x IE = 1のとき割り込みが有効になります。

低特権モードの割込みは常に無効になりますが、高特権モードの割込みは常に有効です。

より高い特権レベルのコードは、より低い特権レベルに制御を委譲する前に、割込みごとのイネーブルビットを使用するこ選択された割込みを無効にすることができます。

-----------------------------------------------------------------------------------------------------------------------------------

xIEビットは、mstatusの下位ビットに配置され、単一のCSR命令でアトミックにセットまたはクリアすることができます。

ネストされたトラップをサポートするために、各特権モードxには、割り込みイネーブルビットと特権モードの2つのレベルのスタックがあります。

x PIEは、トラップの前に割り込みイネーブルビットの値を保持し、xPPは前の特権モードを保持します。

xPPフィールドはxまでの特権モードしか保持できないため、MPPは2ビット幅、SPPは1ビット幅、UPPは暗黙的にゼロです。

トラップが特権モードyから特権モードxに移されると、x PIEはx IEの値に設定されます。 x IEは0に設定されます。 xPPはyに設定されます。

-----------------------------------------------------------------------------------------------------------------------------------

低特権モードでは、通常、エントリ（エントリ）時に割り込みが禁止された状態で、上位の特権モードでトラップ（同期または非同期）が行われます。

上位レベルのトラップハンドラは、スタックされた情報を使用してトラップを処理して戻り、または、中断されたコンテキストに直ちに戻ってこない場合、割り込みを再び有効にする前に特権スタックを保存し、スタックごとに1つのエントリだけが必要です。

MRET、SRET、またはURET命令は、それぞれMモード、Sモード、またはUモードのトラップから戻るために使用されます。

xRP命令を実行するとき、xPPが値yを保持すると仮定すると、x IEはx PIEに設定される。 特権モードはyに変更されます。 x PIEは1に設定されます。 xPPはU（ユーザーモードがサポートされていない場合はM）に設定されます。

-----------------------------------------------------------------------------------------------------------------------------------

スタックがポップされると、サポートされている最低限の割り込み特権モードがスタックの最下部に追加され、無効なエントリがポップされる原因となるエラーを捕捉するのに役立ちます。

xPPフィールドは、サポートされている特権モード（xおよびxより小さい実装された特権モードを含む）だけを格納できるようにする必要があるWLRLフィールドです。

-----------------------------------------------------------------------------------------------------------------------------------

マシンがUモードとMモードのみを提供する場合、MPPの00または11のいずれかを表すためにハードウェア記憶ビットは1つだけ必要です。

ユーザーレベルの割り込みは、オプションの拡張であり、ISA拡張文字Nが割り当てられています。

ユーザーレベルの割り込みを省略すると、UIEビットとUPIEビットはゼロに固定されます。

他のすべてのサポートされている特権モードxでは、x IEとx PIEはハードワイヤードであってはなりません。

↑ここもGoogle翻訳とBing翻訳で真逆の意味に翻訳される。Bing翻訳は否定形に弱いのかな。

-----------------------------------------------------------------------------------------------------------------------------------

ユーザーレベルの割り込みは主に、MモードとUモードのみのセキュアな組み込みシステムをサポートすることを目的としていますが、ユーザーレベルのトラップ処理をサポートする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でもサポートできます。

-----------------------------------------------------------------------------------------------------------------------------------

以前mstatusでHモードサポートのために割り当てられていたフィールドは、現在WPRIフィールドとして予約されています。

既存の実装との後方互換性を低下させるために、これらのフィールドを削除した後は、レジスタを圧縮しませんでした。

3.1.8 mstatusレジスタのベースISA制御

RV64およびRV128システムの場合、SXLおよびUXLフィールドは、それぞれSモードおよびUモードのXLENの値を制御するWARLフィールドです。

これらのフィールドの符号化は、表3.1に示すミサのMXLフィールドと同じです。

SモードおよびUモードで有効なXLENは、それぞれS-XLENおよびU-XLENと呼ばれます。

RV32システムの場合、SXLおよびUXLフィールドは存在せず、S-XLEN = 32およびU-XLEN = 32です。

RV64およびRV128システムの場合、Sモードがサポートされていない場合、SXLはゼロにハードワイヤードされます。

それ以外の場合は、S-XLENの現在の値をエンコードするのはWARLフィールドです。

特に、S-XLEN = M-XLENとなるようにSXLを実装することがあります。

RV64およびRV128システムの場合、Uモードがサポートされていない場合、UXLはゼロに固定配線されます。

それ以外の場合は、U-XLENの現在の値をエンコードするのはWARLフィールドです。

特に、U-XLEN = M-XLENとなるようにUXLを実装することができます。

どのモードでもXLENがサポートされているXLENよりも小さい値に設定されると、すべての演算は、設定されたXLENより上のソース・オペランド・レジスタ・ビットを無視しなければならず、サポートされている最も広いXLENをデスティネーション・レジスタに埋めるために結果を符号拡張する必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

ハードウェアの複雑さを軽減するため、アーキテクチャでは、低特権モードでXLEN設定が次の高位特権モード以下であることを確認しません。

実際には、このような設定はほとんど常にエラーになりますが、この場合でもマシンの動作は明確に定義されています。

3.1.9 mstatusレジスタのメモリ特権

MPRV（Modify PRiVilege）(権限の変更)ビットは、すべての特権モードでロードおよびストアが実行される特権レベルを変更します。

MPRV = 0の場合、変換と保護は正常に動作します。

MPRV = 1の場合、現在の特権モードがMPPに設定されているかのように、読み込みと保存のメモリアドレスは変換され、保護されます。

命令のアドレス変換と保護は影響を受けません。

Uモードがサポートされていない場合、MPRVは0にハードワイヤードされます。

MXR（Make eXecutable Readable）(実行可能ファイルを読めるようにする) ビットは、ロードが仮想メモリにアクセスする際の特権を変更します。

MXR = 0の場合、読み込み可能とマークされたページからの読み込みのみが成功します（図4.15のR = 1）。

MXR = 1の場合、読み取り可能または実行可能（R = 1またはX = 1）のいずれかのマークが付けられたページからの読み込みは成功します。

ページベースの仮想メモリが有効でない場合、MXRは効果がありません。

Sモードがサポートされていない場合、MXRは0にハードワイヤードされます。

-----------------------------------------------------------------------------------------------------------------------------------

MPRVおよびMXR機構は、ミスアラインされた読み込みおよび保存などの欠落したハードウェア機能をエミュレートするMモードルーチンの効率を改善するために考案されました。

MPRVは、ソフトウェアでアドレス変換を実行する必要性を排除します。

MXRを使用すると、実行専用とマークされたページから命令語をロードできます。

簡単にするため、MPRVとMXRは特権モードに関係なく有効ですが、通常の使用ではマシンモードの短いシーケンスに対してのみ有効になります。

SUM（スーパーバイザユーザメモリアクセス許可）ビットは、Sモードの読み込み、書き込み、および命令フェッチが仮想メモリにアクセスするための特権を変更します。

SUM = 0のとき、Uモードでアクセス可能なページへのSモードメモリアクセスは、エラーになります。（図4.15のU = 1）。

↑ここはBing翻訳の方が正しい。Google翻訳だと最後のfaultを飛ばす。うーん。

SUM = 1の場合、これらのアクセスは許可されます。

ページベースの仮想メモリが有効でない場合、SUMは効果がありません。

SUMはSモードで実行されていないときは通常無視されますが、MPRV = 1かつMPP = Sのときには有効です。

Sモードがサポートされていない場合、SUMは0にハードワイヤードされます。

3.1.10 mstatusレジスタの仮想化サポート

TVM（トラップ仮想メモリ）ビットは、スーパバイザ仮想メモリ管理操作の代行受信(インターセプト)をサポートします。

TVM = 1の場合、S-modeで実行中にsatp CSRを読み書きしようとするかSFENCE.VMA命令を実行しようとすると、不正な命令例外が発生します。

TVM = 0の場合、これらの操作はSモードで許可されます。

Sモードがサポートされていない場合、TVMは0にハードワイヤードされています。

-----------------------------------------------------------------------------------------------------------------------------------

TVMメカニズムは、ゲストオペレーティングシステムをUモードで古典的に仮想化するのではなく、Sモードで実行できるようにすることで、仮想化の効率を向上させます。

このアプローチは、ほとんどのSモードCSRsへのアクセスをトラップする必要性を排除します。

satpアクセスとSFENCE.VMA命令をトラッピングすると、シャドウページテーブルを遅延読み込みするのに必要なフックが提供されます。

TW（タイムアウト待機）ビットは、WFI命令のインターセプトをサポートします（3.2.3項を参照）。

TW = 0のとき、WFI命令はSモードで許可されます。

TW = 1の場合、WFIがSmodeで実行され、実装固有の制限時間内に完了しない場合、WFI命令は不正な命令トラップを引き起こします。

制限時間は常に0にすることができ、この場合TW = 1のときにWFIはSモードで常に不正な命令トラップを引き起こします。

Sモードがサポートされていない場合、TWは0にハードワイヤードされます。

-----------------------------------------------------------------------------------------------------------------------------------

WFI命令をトラップすると、現在のゲストで無駄にアイドリングするのではなく、別のゲストOSに切り替えることができます。

TSR（トラップSRET）ビットは、スーパバイザ例外リターン命令SRETのインターセプトをサポートします。

TSR = 1のとき、Sモードで実行中にSRETを実行しようとすると、不正な命令例外が発生します。

TSR = 0のときは、Sモードで許可されます。

Sモードがサポートされていない場合、TSRは0にハード配線されます。

-----------------------------------------------------------------------------------------------------------------------------------

SRETをトラッピングすることは、それを提供しない実装で拡張仮想化機構（第5章参照）をエミュレートするために必要です。

3.1.11 mstatusレジスタの拡張コンテキスト状況

実質的な拡張をサポートすることは、RISC-Vの主な目標の1つです。したがって、変更されていない特権モードのコード、特にスーパバイザレベルのOSが任意のユーザーモードの状態拡張をサポートできるようにする標準インターフェイスを定義します。

-----------------------------------------------------------------------------------------------------------------------------------

今日まで、浮動小数点CSRおよびデータレジスタ以外の追加の状態を定義する標準的な拡張はありません。

FS [1：0]読み取り/書き込みフィールドとXS [1：0]読み取り専用フィールドは、浮動小数点ユニットの現在の状態、および他のユーザモードの拡張をそれぞれ示す設定と追跡によってコンテキストの保存と復元のコストを削減するために使用されます

FSフィールドは、CSR fcsrおよび浮動小数点データレジスタf0〜f31を含む浮動小数点ユニットのステータスをエンコードし、XSフィールドは、追加のユーザモード拡張および関連するステートのステータスをエンコードします。

これらのフィールドは、状態の保存または復元が必要かどうかを迅速に判断するためのコンテキスト切り替えルーチンによってチェックできます。

保存または復元が必要な場合は、通常、プロセスを効果的に最適化するために追加の指示とCSRsが必要です。

-----------------------------------------------------------------------------------------------------------------------------------

この設計では、ほとんどのコンテキストスイッチが浮動小数点ユニットまたはその他の拡張のいずれかまたは両方で状態を保存/復元する必要がないため、SDビットによる高速チェックが可能になると予想しています。

FSフィールドとXSフィールドは、表3.3に示されているものと同じステータスエンコーディングを使用し、4つの可能なステータス値はオフ、イニシャル、クリーン、およびダーティです。

|  |  |  |
| --- | --- | --- |
| 状態 | FSの意味 | XSの意味 |
| 0  1  2  3 | オフ  イニシャル  クリーン  ダーティ | すべてオフ  いずれもダーティではないか、クリーンではダーティでない、いくつかはクリーン  いくつかダーティ |

表3.3：FS [1：0]およびXS [1：0]ステータスフィールドのエンコード

Sモードを実装せず、浮動小数点ユニットを持たないシステムでは、FSフィールドはゼロにハードワイヤードされています。

新しい状態を必要とする追加のユーザー拡張がないシステムでは、XSフィールドはゼロにハードワイヤードされます。

状態付きのすべての追加拡張は、XS状態に相当するmstatusにローカルフィールドを追加します。

XSフィールドは、表3.3に示すように、すべての内線番号のステータスの要約を表します。

-----------------------------------------------------------------------------------------------------------------------------------

XSフィールドは、すべてのユーザー拡張ステータスフィールドにわたって最大ステータス値を効果的に報告しますが、個々の拡張機能はXSとは異なるエンコードを使用できます。

SDビットは、FSフィールドまたはXSフィールドのいずれかが、拡張されたユーザーコンテキストをメモリに保存する必要があるダーティ状態の存在を示すかどうかを要約する読み取り専用ビットです。

XSとFSの両方がゼロにハードワイヤードされていると、SDも常にゼロになります。

拡張の状態がオフに設定されている場合、対応するステートを読み書きしようとする命令は例外を発生させます。

ステータスがイニシャルの場合、対応する状態は初期の定数値を持つ必要があります。

状態がクリーンの場合、対応する状態は初期値と異なる可能性がありますが、コンテキストスワップに格納されている最後の値と一致します。

ステータスがダーディの場合、対応するステートは、最後のコンテキスト保存以降に変更されている可能性があります。

コンテキスト保存中に責任ある特権コードは、その状態がダーティの場合にのみ対応する状態を書き出す必要があり、その後、その状態をクリーンにリセットすることができます。

コンテキスト復元時には、状態がクリーン（復元時にダーティであってはならない）の場合にのみコンテキストをメモリからロードする必要があります。

ステータスがイニシャルの場合、セキュリティホールが発生しないようコンテキストを復元時に初期の定数値に設定する必要がありますが、これはメモリにアクセスせずに行うことができます。

たとえば、浮動小数点レジスタはすべて即値0に初期化できます。

FSおよびXSフィールドは、コンテキストを保存する前に特権コードによって読み取られます。

FSフィールドは、ユーザコンテキストを再開するときには特権コードによって直接設定され、XSフィールドは個々の拡張のステータスレジスタに書き込むことによって間接的に設定されます。

ステータスフィールドは、特権モードに関係なく、命令の実行中に(も)更新されます。

ユーザーモードISAへの拡張には、追加のユーザーモード状態が含まれることが多く、この状態は基本整数レジスタよりもかなり大きくなる可能性があります。

拡張機能は、一部のアプリケーションでのみ使用することも、単一のアプリケーション内で短期間だけ必要とすることもあります。

パフォーマンスを向上させるために、ユーザモード拡張は、ユーザモードソフトウェアがユニットを初期状態に戻すか、またはユニットをオフにすることを可能にする追加の命令を定義することができます。

↑使わない時はそのユーザーモードをOFFにできる。例えばそのブロックのクロックを止めたり、電源をOFFしたりできるってこと。

たとえば、コプロセッサを使用する前に構成する必要があり、使用後にコプロセッサを”未構成”することができます。

未構成状態は、コンテキスト保存の初期状態として表されます。

未構成と次の構成（状態をダーティに設定する）の間で同じアプリケーションが実行中のままである場合、すべての状態がユーザープロセスに対してローカルであるため、実際に未構成命令で状態を再初期化する必要はありません。 すなわち、初期状態は、コンテキスト・リストア時にコプロセッサ・ステートがすべての未構成時ではなく一定値に初期化されるだけでよいです。

ユニットをディスエーブルしてオフ状態にするユーザモード命令を実行すると、後続の命令が再びオンになる前にユニットを使用しようとすると、不正な命令例外が発生します。

ユニットをオンにするユーザモード命令は、そのユニットが別のコンテキストによって使用されている可能性があるため、ユニットの状態が適切に初期化されていることを保証しなければなりません。(確認する必要があります。)

FSの設定を変更しても、浮動小数点レジスタの状態は変化しません。

特に、FS = オフを設定しても状態が破壊されることはなく、FS = イニシャルが内容をクリアするように設定することもありません。

オフに設定すると、他の拡張機能が状態を保持しないことがあります。(保持できません。)

表3.4に、FSまたはXSステータスビットのすべての可能な状態遷移を示します。

標準の浮動小数点拡張は、ユーザモードの構成解除または無効/有効命令をサポートしていないことに注意してください。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 現在の状態  動作 | オフ | イニシャル | クリーン | ダーティ |
| 特権コードでのコンテキスト保存時 | | | | |
| 保存状態？  次の状態 | いいえ  オフ | いいえ  イニシャル | いいえ  クリーン | はい  クリーン |
| 特権コードでのコンテキスト復元時 | | | | |
| 復元状態？  次の状態 | いいえ  オフ | はい、イニシャルに(へ)  イニシャル | はい、メモリから  クリーン | N/A  N/A |
| 状態を読み出すための命令を実行 | | | | |
| 動作？  次の状態 | 例外  オフ | 例外  イニシャル | 例外  クリーン | 例外  ダーティ |
| コンフィグレーションを含むステートの変更命令を実行 | | | | |
| 動作？  次の状態 | 例外  オフ | 実行  ダーティ | 実行  ダーティ | 実行  ダーティ |
| 未構成ユニットへの命令実行 | | | | |
| 動作？  次の状態 | 例外  オフ | 実行  イニシャル | 実行  イニシャル | 実行  イニシャル |
| ユニットを無効にする命令を実行 | | | | |
| 動作？  次の状態 | 実行  オフ | 実行  オフ | 実行  オフ | 実行  オフ |
| ユニットを有効にする命令を実行 | | | | |
| 動作  次の状態 | 実行  イニシャル | 実行  イニシャル | 実行  イニシャル | 実行  イニシャル |

表3.4：FSおよびXSの状態遷移。

-- 2018/11/24

拡張状態を初期化、保存、復元するための標準的な特権命令は、状態を不透明なオブジェクトとして扱うことによって、拡張された拡張状態の詳細から特権コードを隔離するために提供されます。

-----------------------------------------------------------------------------------------------------------------------------------

多くのコプロセッサ拡張は、制限されたコンテキストでのみ使用されます。これにより、ソフトウェアがユニットの設定を安全に解除するか、実行時にユニットを無効にすることさえできます。

これにより、大規模なステートフルコプロセッサのコンテキストスイッチオーバーヘッドが削減されます。

浮動小数点状態を他の拡張状態から分離し、浮動小数点ユニットが存在する場合、浮動小数点レジスタは標準呼び出し規約の一部であるため、ユーザモードソフトウェアは浮動小数点ユニットを無効にすることが安全であるかどうかを知ることができません。

XSフィールドは、追加されたすべての拡張状態の要約を提供しますが、コンテキストの保存と復元のオーバーヘッドをさらに削減するために、拡張で追加のマイクロアーキテクチャビットが維持される可能性があります。

SDビットは読み出し専用であり、FSまたはXSビットのいずれかがダーティ状態（すなわち、SD =（FS == 11）OR（XS == 11））を符号化するときに設定されます。

これにより、特権コードは、整数レジスタセットとPC以外のコンテキスト保存が不要になったときを素早く判断することができます。

浮動小数点ユニットの状態は、標準命令（F、D、および/またはQ）を使用して常に初期化、保存、および復元され、特権付きコードはFLENを認識して各fレジスタに予約する適切な領域を決定する必要があります。

スーパバイザレベルのOSでは、追加のユーザモードの状態は、追加のコンテキストを一定の最大サイズの不透明なオブジェクトとして扱うSBI呼び出しを使用して初期化、保存、復元する必要があります。

SBIの初期化、保存、および復元の実装では、コプロセッサ内のマイクロアーキテクチャ状態を初期化、保存、および復元するために、インプリメンテーション依存の特権命令が追加で必要になる場合があります。

すべての特権モードはFSとXSビットの1つのコピーを共有します。

複数の特権モードを持つシステムでは、スーパーバイザモードでは通常、スーパーバイザレベルのセーブされたコンテキストに関してステータスを記録するためにFSビットとXSビットを直接使用します。

他のより特権のあるアクティブモードは、対応するバージョンのコンテキストで拡張状態を保存して復元する際に、より慎重でなければなりません。

-----------------------------------------------------------------------------------------------------------------------------------

任意の合理的なユースケースでは、ユーザとスーパーバイザレベル間のコンテキストスイッチの数は、他の権限レベルへのコンテキストスイッチの数よりはるかに多いはずです。(上回る必要があります。)

コプロセッサは、割り込みによってユーザレベルのコンテキストスワップが発生しない限り、非同期割り込みを処理するためにコンテキストを保存および復元する必要はないことに注意してください。

3.1.12マシントラップベクトルベースアドレスレジスタ（mtvec）

mtvecレジスタは、ベクトルベースアドレス（BASE）とベクタモード（MODE）で構成されるトラップベクタの構成を保持するXLENビットの読み/書きレジスタです。

XLEN-1 2 1 0

|  |  |
| --- | --- |
| BASE[XLEN-1:2](WARL) | MODE(WARL) |

XLEN-2 2

図3.8：マシントラップベクタベースアドレスレジスタ（mtvec）。

mtvecレジスタは常に実装する必要がありますが、ハードワイヤードの読み取り専用値を含めることができます。

mtvecが書き込み可能な場合、レジスタが保持できる値のセットは実装によって異なる可能性があります。

BASEフィールドの値は常に4バイトの境界に揃えなければならず、MODE設定はBASEフィールドの値に追加の配置制約を課す可能性があります。

-----------------------------------------------------------------------------------------------------------------------------------

トラップベクタベースアドレスの実装にはかなりの柔軟性があります。

一方では、多数の状態ビットを持つローエンドの実装に負担をかけることは望ましくありませんが、一方で、より大きいシステムでは可用性を可能にしたいと考えています。(柔軟性を許可することを希望しています。)

|  |  |  |
| --- | --- | --- |
| 値 | 名前 | 説明 |
| 0  1  ≧2 | ダイレクト  ベクタード  － | すべての例外はpcをBASEに設定  非同期割り込みは、pcをBASE + 4 \*causeに設定  予約 |

表3.5：mtvec MODEフィールドのエンコーディング

MODEフィールドのエンコーディングを表3.5に示します。

MODE = ダイレクトの場合、マシンモードに入るすべてのトラップが、PCをベースフィールドのアドレスに設定します。

MODE = ベクターの場合、マシンモードへの同期例外はすべてPCをベースフィールドのアドレスに設定しますが、割り込みはPCをベースフィールドのアドレス+割り込み要因番号の4倍に設定します。

たとえば、マシンモードタイマ割り込み（表3.6を参照）は、PCをBASE + 0x1cに設定します。

MODE =ベクタードを設定すると、ベースに追加のアラインメント制約が課されることがあり、最大4 \* XLENバイトのアライメントが必要です。

-----------------------------------------------------------------------------------------------------------------------------------

ベクタ割り込みが有効にされると、ユーザモードのソフトウェア割り込みに対応する割り込み要因0は、同期例外と同じ場所にベクトル化されます。

この曖昧さは、実際には発生しません。これは、ユーザーモードのソフトウェア割り込みが無効になっているか、権限の低いモードに委任されているためです。

-----------------------------------------------------------------------------------------------------------------------------------

リセットおよびNMIベクタの場所は、プラットフォーム仕様に記載(指定)されています。

3.1.13マシントラップ代理レジスタ（medelegとmideleg）

デフォルトでは、任意の特権レベルのすべてのトラップはマシンモードで処理されますが、マシンモードハンドラはトラップをMRET命令（3.2.2項）で適切なレベルにリダイレクトできます。

パフォーマンスを向上させるために、実装はmedelegおよびmideleg内の個々の読み取り/書き込みビットを提供して、特定の例外および割り込みを低い特権レベルで直接処理する必要があることを示します。

マシン例外委譲レジスタ（medeleg）およびマシン割り込み委譲レジスタ（mideleg）は、XLENビットの読み取り/書き込み・レジスタです。

3つの特権モード（M / S / U）を持つすべてのシステムでは、medelegまたはmidelegにビットを設定すると、SモードまたはUモードの対応するトラップがSモードトラップハンドラに委任されます。

Uモードトラップがサポートされている場合、Sモードは、sedelegレジスタとsidelegレジスタに対応するビットを順番に設定して、Uモードで発生するトラップをUモードトラップハンドラに委任することができます。

2つの特権モード（M / U）とUモードトラップをサポートするシステムでは、medelegまたはmidelegにビットを設定すると、Uモードの対応するトラップがUモードトラップハンドラに委任されます。

Mモードのみ、またはMモードとUモードの両方を持つが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とmidelegレジスタは存在してはいけません。(存在しません)

-----------------------------------------------------------------------------------------------------------------------------------

1.9.1以前のバージョンでは、ここれらのレジスタは存在していましたが、MモードでのみまたはNシステムなしのM / Uでゼロに固定配線されていました。

これらのケースでは、ミサレジスタが存在するかどうかを示すので、ゼロを返す必要はありません。

トラップが特権の低いモードxに委任されると、x原因レジスタにトラップ原因が書き込まれます。

x epcレジスタには、トラップを取った命令の仮想アドレスが書き込まれます。

mstatusのxPPフィールドは、トラップ時にアクティブ特権モードで書き込まれます。

mstatusのx PIEフィールドには、トラップ時にアクティブな割り込みイネーブルビットの値が書き込まれます。

mstatusのx IEフィールドがクリアされます。

mcauseおよびmepcレジスタとmstatusのMPPおよびMPIEフィールドは書き込まれません。

実装は、委譲ビットを1つにハードワイヤしてはなりません。すなわち、委任可能なトラップが委任されていないことをサポートしなければなりません。

実装では、デリゲート可能なトラップをサブセット化することを選択できます。サポートされている委託可能なビットは、どのビット位置が1を保持しているかを見るために、medelegまたはmidelegの値を読み戻します。

トラップは、より特権モードから低特権モードに移行することはありません。

たとえば、MモードがSモードに不正な命令トラップを委譲し、Mモードソフトウェアが後で不正命令を実行すると、Sモードに委譲されるのではなく、Mモードでトラップが取られます。

対照的に、トラップは水平に取られてもよい。(取られることがあります)

同じ例を使用して、Mモードが不正な命令トラップをSモードに委譲し、Sモードソフトウェアが後で不正命令を実行すると、トラップはSモードで取られます。(行われます)

XLEN-1 0

|  |
| --- |
| 同期例外（WARL） |

XLEN

図3.9：マシン例外委譲登録medeleg。

medelegは、表3.6に示すすべての同期例外に対して割り当てられたビット位置を持ち、ビット位置のインデックスはmcauseレジスタに返された値と等しくなります。（すなわち、ビット8を設定することにより、ユーザモード環境呼出しをより低い特権のトラップハンドラに委任することができます）。

XLEN-1 0

|  |
| --- |
| 割り込み（WARL） |

XLEN

図 3.10: マシン例外の委任登録 mideleg。

midelegは個々の割り込みのトラップ委譲ビットを保持し、ビットのレイアウトはmipレジスタのものと一致します。

（すなわち、STIP割り込み委譲制御はビット5に位置します）。

いくつかの例外は、特権の低いモードでは発生することができず(発生しませんし)、対応するx edelegビットはゼロにハードワイヤードされるべきです。(する必要があります。)

特に、medeleg [11]とsedeleg [11：9]はすべてゼロに固定されています。

3.1.14マシン割り込みレジスタ（mipおよびmie）

mipレジスタは、保留中の割り込みに関する情報を含むXLENビットの読み/書き・レジスタです。一方、mieは、割り込みイネーブル・ビットを含む対応するXLENビットの読み/書き・レジスタです。

このCSRアドレスによって、低特権ソフトウェア割り込み（USIP、SSIP）、タイマ割り込み（UTIP、STIP）、およびmipの外部割り込み（UEIP、SEIP）に対応するビットだけが書き込み可能です。

  残りのビットは読み取り専用です。

mipレジスタとmieレジスタの制限されたビューはsip / sieとして表示され、uip / uieはそれぞれSモードとUモードで登録されます。

割り込みがmidelegレジスタにビットをセットすることによって特権モードxに委任された場合、x ipレジスタに表示され、x ieレジスタを使用してマスク可能です。

それ以外の場合、x ipとx ieの対応するビットはゼロにハードワイヤードされているように見えます。

XLEN-1 12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WIRI | MEIP | WIRI | SEIP | UEIP | MTIP | WIRI | STIP | UTIP | MSIP | WIRI | SSIP | USIP |

XLEN-12 1 1 1 1 1 1 1 1 1 1 1 1

図3.11：マシン割込み保留レジスタ（mip）。

MTIPビット、STIPビット、UTIPビットは、マシン、スーパバイザ、ユーザタイマ割り込みのタイマ割り込み保留ビットにそれぞれ対応します。

MTIPビットは読み出し専用で、メモリマップされたマシンモードタイマコンペアレジスタへの書き込みによってクリアされます。

UTIPビットとSTIPビットは、低レベルの特権レベルにタイマー割込みを送るためにMモードソフトウェアによって書き込まれます。

ユーザおよびスーパバイザソフトウェアは、それぞれAEEおよびSEEへのコールにより、UTIPおよびSTIPビットをクリアすることができます。

XLEN-1 12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WPRI | MEIE | WPRI | SEIE | UEIE | MTIE | WPRI | STIE | UTIE | MSIE | WPRI | SSIE | USIE |

XLEN-12 1 1 1 1 1 1 1 1 1 1 1 1

図3.12：マシン割込み許可レジスタ（mie）。

Mモード、Sモード、Uモードの各タイマ割り込みには、それぞれMTIE、STIE、UTIEという名前の別々のタイマ割り込みイネーブルビットがあります。

各下位特権レベルには、独立した(個別の)ソフトウェア割り込み保留ビット（SSIP、USIP）があります。これは、関連するまたはそれ以上の特権レベルでローカルハートで実行されているコードからCSRアクセスによって読み書きすることができます。

マシンレベルのMSIPビットは、メモリマップ制御レジスタへのアクセスによって書き込まれ、これはリモートハートがマシンモードのプロセッサ間割り込みを提供するために使用されます。

より低い特権レベルのためのプロセッサ間割り込みは、それぞれAEEまたはSEEへのABIおよびSBIコールを介して実装され、それは最終的に受信側ハートのMSIPビットへのマシンモード書き込みをもたらす可能性があります。

ハートは、同じメモリマップ制御レジスタを使用して、独自のMSIPビットを書き込むことができます。

-----------------------------------------------------------------------------------------------------------------------------------

私たちは、適切なモードで動作している場合、ハートが独自のSSIPまたはUSIPビットを直接書き込むことを許可しますが、他のハートが仮想化され、より高い特権レベルによって予定されている可能性があります。

この理由から、ABIとSBIの呼び出しによってプロセッサ間割り込みが提供されています。

マシンモードのハーツは仮想化されておらず、MSIPビットを設定することで他のハートに直接割り込みをかけることができます。

通常、プラットフォーム仕様に応じてメモリマップ制御レジスタへのキャッシュされていないI / O書き込みを使用します。

mipのMEIPフィールドは、マシンモード外部割り込みが保留中であることを示す読み出し専用ビットです。

MEIPは、第7章で指定されている標準プラットフォームレベルの割り込みコントローラなど、プラットフォーム固有の割り込みコントローラによって設定およびクリアされます。

mieのMEIEフィールドは、設定時にマシンの外部割り込みを有効(可能)にします。

mipのSEIPフィールドには、単一の読み書きビットが含まれます。

SEIPは、外部割込みが保留中であることをSモードに示すために、Mモードソフトウェアによって書かれることがあります。

さらに、プラットフォームレベルの割り込みコントローラは、スーパーバイザレベルの外部割り込みを生成することがあります。

ソフトウェア書き込み可能ビットと外部割り込みコントローラからの信号の論理和は、スーパーバイザへの外部割り込みを生成するために使用されます。

SEIPビットがCSRRW、CSRRS、またはCSRRC命令で読み取られると、rd宛先レジスタに返される値には、ソフトウェア書き込み可能ビットと割り込みコントローラからの割り込み信号の論理和が含まれます。

ただし、CSRRS命令またはCSRRC命令のリード・モディファイ・ライト・シーケンスで使用される値は、ソフトウェア書き込み可能なSEIPビットのみであり、外部割り込みコントローラからの割り込み値は無視されます。

-----------------------------------------------------------------------------------------------------------------------------------

SEIPフィールドの動作は、上位の特権層が外部の割り込みをきちんと模倣できるように設計されており、実際の外部割り込みを失うことはありません。

その結果、CSR命令の動作は通常のCSRアクセスから少し変更されています。

MIPのUEIPフィールドは、ユーザモード割り込みのためのN個の拡張が実装されているとき、ユーザモードの外部割り込みを提供します。

SEIPと同様に定義されます。

mie CSRのMEIE、SEIE、およびUEIEフィールドは、それぞれMモード外部割り込み、Sモード外部割り込み、およびUモード外部割り込みをそれぞれ有効にします。

-----------------------------------------------------------------------------------------------------------------------------------

非マスク可能割り込みは、NMIトラップハンドラを実行するときにその存在が暗黙的に知られている(認識される)ので、mipレジスタを介して見ることはできません。(表示されません)

すべての様々な割り込みタイプ（ソフトウェア、タイマ、外部）に対して、特権レベルがサポートされていない場合、関連する保留中および割り込み可能ビットは、それぞれmipおよびmieレジスタでゼロにハードワイヤードされます。

したがって、これらはすべて効果的にWARLフィールドです。

-----------------------------------------------------------------------------------------------------------------------------------

これらのレジスタの上位ビットには、プラットフォーム固有のマシンレベルの割り込みソースを追加することができますが、ほとんどの外部割り込みはプラットフォーム割り込みコントローラを経由してルーティングされ、MEIP経由で配信されることが期待されます。(配線されます。)

割り込みiは、ビットiがmipとmieの両方に設定されている場合、および割り込みがグローバルにイネーブルされている場合に行われます。

デフォルトでは、ハートの現在の特権モードがMより小さい場合、または現在の特権モードがMでmstatusレジスタのMIEビットが設定されている場合、Mモード割込みはグローバルにイネーブルされます。

しかし、midelegのビットiがセットされている場合、hartの現在の特権モードが委譲特権モード（SまたはU）に等しく、そのモードの割込み許可ビット（mstatusのSIEまたはUIE）が設定されている場合、または現在の特権モードが委譲特権モードよりも小さい場合、割り込みはグローバルに有効とみなされます。

同じ特権レベルの複数の同時割り込みとトラップは、次の優先順位の低い順に処理されます。：外部割込み、ソフトウェア割込み、タイマー割込み、次にすべての同期トラップがあります。

3.1.15マシンタイマレジスタ（mtimeとmtimecmp）

プラットフォームは、メモリマップされたマシンモードレジスタmtimeとして公開されるリアルタイムカウンタを提供します。

mtimeは一定の頻度(周波数)で実行されなければならず、プラットフォームはmtimeのタイムベースを決定するためのメカニズムを提供しなければなりません。(必要があります)

mtimeレジスタは、すべてのRV32、RV64、およびRV128システムで64ビット精度を持ちます。

プラットフォームは、64ビットのメモリマップされたマシンモードタイマコンペアレジスタ（mtimecmp）を提供し、mtimeレジスタにmtimecmpレジスタの値以上の値が含まれている場合に、タイマ割り込みがポストされます。

割り込みは、mtimecmpレジスタを書き込むことによってクリアされるまでポストされたままです。

割り込みは、割り込みが許可され、MTIEビットがmieレジスタに設定されている場合にのみ行われます。

63 0

|  |
| --- |
| mtime |

64

図3.13：マシンタイムレジスタ（メモリマップ制御レジスタ）。

-----------------------------------------------------------------------------------------------------------------------------------

タイマ機能は、動的な電圧および周波数スケーリングによってエネルギーを節約するために、非常に可変なクロック周波数で動作する最新のプロセッサをサポートするサイクルカウンタではなく、ウォールクロック時間を使用するように定義されています。

↑wall-clock(壁クロック)ってRTCとか場合によって周波数(周期)が変わらないクロックのことだと思う。

63 0

|  |
| --- |
| mtimecmp |

64

図3.14：マシン時間比較レジスタ（メモリマップ制御レジスタ）。

正確なリアルタイムクロック（RTC）は、（水晶またはMEMS発振器を必要とする）比較的高価であり、システムの残りの部分がパワーダウンしても実行する必要がありますので、通常、プロセッサとは異なる周波数/電圧領域に位置するシステムには1つしか存在しません。

したがって、RTCはシステム内のすべてのハートによって共有されなければならず、RTCへのアクセスは潜在的に電圧レベルシフタとクロックドメイン交差のペナルティを招くことがあります。

したがって、mtimeをCSRよりもメモリマップされたレジスタとして公開する方が自然です。

より低い特権レベルには独自のtimecmpレジスタはありません。

代わりに、マシンモードソフトウェアは、次のタイマー割り込みをmtimecmpレジスタに多重化することによって、ハート上に任意の数の仮想タイマを実装できます。

シンプルな固定周波数システムでは、サイクルカウントとウォールクロックの両方に単一のクロックを使用できます。

RV32では、mtimecmpへのメモリマップ書き込みはレジスタの1つの32ビット部分のみを変更します。

次のコードシーケンスは、comparandの中間値のために、タイマ割り込みを誤って生成することなく、64ビットのmtimecmp値を設定します。：

# New comparand is in a1:a0.

li t0, -1

sw t0, mtimecmp # No smaller than old value.

sw a1, mtimecmp+4 # No smaller than new value.

sw a0, mtimecmp # New value.

図3.15：レジスタが強く順序付けられたI / O領域にあると仮定して、RV32の64ビットの時間比較値を設定するためのサンプルコード。

3.1.16ハードウェアパフォーマンスモニタ

Mモードには、基本的なハードウェアパフォーマンス監視機能が含まれています。

mcycle CSRは、過去の任意の時間以来、ハートが実行したサイクル数のカウントを保持します。

minstret CSRには、ハートが退職(引退)した命令の数が過去の任意の時間からカウントされます。

↑retiredをどう訳すか。実行がすでに終わったぐらいでいいのかな。

mcycleレジスタとminstretレジスタは、すべてのRV32、RV64、およびRV128システムで64ビットの精度を持ちます。

ハードウェアパフォーマンスモニタには、29個の追加イベントカウンタmhpmcounter3-mhpmcounter31が含まれています。

イベントセレクタCSRs mhpmevent3-mhpmevent31は、どのイベントが対応するカウンタを増加させるかを制御するWARLレジスタです。

これらのイベントの意味はプラットフォームによって定義されますが、イベント0は「イベントなし」を意味するために予約されています。 すべてのカウンタを実装する必要がありますが、カウンタと対応するイベントセレクタの両方を0にハードワイヤリングするのが法的な実装です。

↑leagalをどう訳すか。法的というより、正当なとか適切な。とかかな。

これらのカウンタはすべて、RV32、RV64、およびRV128で64ビットの精度を持ちます。

RV32のみでは、mcycle、minstret、およびmhpmcounternのCSRの読み取りによって32ビットが返され、一方、mcycleh、minstreth、およびmhpmcounternh CSRの読み取りは、対応するカウンタのビット63-32を返します。

63 0

|  |  |  |
| --- | --- | --- |
| mcycle |  |  |
| minstret |  |  |
|  |  | XELN-1 0 |
| mhpmcounter3 |  | mphmevent3 |
| mhpmcounter4 |  | mphmevent4 |
| : |  | ： |
| mhpmcounter30 |  | mhpmevent30 |
| mhpmcounter31 |  | mhpmevent31 |

64 XLEN

図3.16：ハードウェアパフォーマンスモニタカウンタ。

31 0

|  |
| --- |
| mcycleh |
| minstreth |
| mhpmcounter3h |
| mhpmcounter4h |
| : |
| mhpmcounter30h |
| mhpmcounter31h |

32

図3.17：上位32ビットのハードウェアパフォーマンスモニタカウンタ、RV32のみ。

RV128システムでは、mcycle、minstret、およびmhpmcounternの64ビット値は、読み込み時に128ビットに符号拡張されます。

-----------------------------------------------------------------------------------------------------------------------------------

RV128システムでは、符号付き64ビット値と符号なし64ビット値の両方が正準形式で保持され、ビット63はすべての上位ビット位置で繰り返されます。

↑63bit目が符号拡張されるということ

カウンタはRV128でも64ビットの値であるため、カウンタCSRは符号拡張拡張不変量を保持して読み込みます。

実装は、カウンタのより少ないビットを実装することを選択することができ、（例えば、263命令が実行される）ラップアラウンドを経験する可能性は低く、それによって符号拡張回路を実際に実装する必要がなくなる。

↑この例では263まで数えられれば良いので9bit512まで数えられるカウンタでOK

3.1.17カウンタイネーブルレジスタ（[m | h | s] counteren）

31 30 29 28 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HPM31 | HPM30 | HPM29 | ．．． | HPM5 | HPM4 | HPM3 | IR | TM | CY |

1 1 1 23 1 1 1 1 1 1

図3.18：カウンタイネーブルレジスタ（mcounterenとscounteren）。

カウンタイネーブルレジスタmcounterenおよびscounterenは、ハードウェアパフォーマンスモニタリングカウンタの可用性を、次に低い特権モードに制御します。

mcounterenレジスタのCY、TM、IR、またはHPMnビットがクリアされている場合、SモードまたはUモードで実行中に、cycle、time、instret、またはhpmcounternレジスタを読み込もうとすると、不正な命令例外が発生します。

これらのビットの1つがセットされると、次の実装された特権モード（実装されている場合はSモード、それ以外の場合はUモード）で対応するレジスタへのアクセスが許可されます。

Sモードが実装されている場合、Uモードで実行している間は、scounterenレジスタの同じビット位置がこれらのレジスタへのアクセスを同様に制御します。

Sモードがカウンタ・レジスタへのアクセスを許可され、対応するビットがscounterenに設定されている場合、Uモードもそのレジスタにアクセスすることができます。

レジスタmcounterenとscounterenは、UモードとSモードが実装されている場合に実装する必要があるWARLレジスタです。

いずれのビットにもハードワイヤードの値0が含まれている可能性があります。これは、対応するカウンタへの読み込みによって、特権レベルの低いモードで実行されているときに例外が発生することを示します。

-----------------------------------------------------------------------------------------------------------------------------------

カウンタイネーブルビットは、最小限のハードウェアで2つの一般的な使用例をサポートします。

高性能タイマとカウンタが不要なシステムの場合、マシンモードソフトウェアはアクセスをトラップし、ソフトウェアのすべての機能を実装できます。

高性能タイマとカウンタを必要とするが、基盤となるハードウェアカウンタの難読化には関係(考慮)しないシステムでは、カウンタを低特権モードに直接さらす(公開する)ことができます。

cycle、instret、およびhpmcounternのCSRsは、それぞれmcycle、minstret、およびmhpmcounternの読み取り専用のシャドウです。

time CSRは、メモリマップされたmtimeレジスタの読み取り専用の影(シャドウ)です。

-----------------------------------------------------------------------------------------------------------------------------------

実装は、time CSRの読み込みをメモリマップされたmtimeレジスタへのロードに変換するか、またはmxcounterenのTMビットを0にハードワイヤリングし、M-modeソフトウェアでこの機能をエミュレートすることができます。

3.1.18マシンスクラッチレジスタ（mscratch）

mscratchレジスタは、マシンモードで使用するために専用のXLENビットのリード/ライトレジスタです。

通常は、マシンモードハートローカルコンテキスト空間へのポインタを保持するために使用され、Mモードトラップハンドラへのエントリ時にユーザレジスタとスワップされます。

XLEN-1 0

|  |
| --- |
| mscratch |

XLEN

図3.19：マシンモードスクラッチレジスタ。

-----------------------------------------------------------------------------------------------------------------------------------

MIPS ISAは、オペレーティングシステムで使用するために2つのユーザーレジスタ（k0 / k1）を割り当てました。

MIPSスキームは、迅速かつ簡単な実装を提供しますが、使用可能なユーザー・レジスタを減らし、さらに特権レベルやネストされたトラップに拡張されません。

潜在的なセキュリティホールを回避し、確定的なデバッグ動作を提供するために、ユーザーレベルに戻る前に両方のレジスタをクリアする必要があります。

RISC-VユーザーISAは、多くの可能な特権システム環境をサポートするように設計されているため、ユーザーレベルのISAにはOS依存の機能を感染させたくありませんでした。

RISC-V CSRスワップ命令は、値をmscratchレジスタにすばやく保存/復元できます。

MIPSデザインとは異なり、OSはユーザコンテキストの実行中にmscratchレジスタに値を保持することができます。

3.1.19マシン例外プログラムカウンタ（mepc）

mepcは、図3.20のように書式設定されたXLENビットの読取り/書込みレジスタです。

mepc（mepc [0]）の下位ビットは常にゼロです。

16ビット命令アライメントを持つ命令セット拡張をサポートしない実装では、2つの下位ビット（mepc [1：0]）は常にゼロです。

-----------------------------------------------------------------------------------------------------------------------------------

mepcレジスタは、命令アドレス不一致の例外を発生させるPC値を保持することはできません。

mepcは、有効な物理アドレスと仮想アドレスをすべて保持できる必要があるWARLレジスタです。

すべての可能な無効アドレスを保持できる必要はありません。

実装は、無効なアドレスパターンをmepcに書き込む前に他の無効なアドレスに変換する可能性があります。

トラップがMモードになると、mepcには例外が発生した命令の仮想アドレスが書き込まれます。

それ以外の場合、mepcは実装によって書き込まれることはありませんが、ソフトウェアによって明示的に記述されることがあります。

XLEN-1 0

|  |
| --- |
| mepc |

XLEN

図3.20：マシン例外プログラムカウンタレジスタ。

3.1.20マシン要因レジスタ（mcause）

mcauseレジスタは、図3.21に示すようにフォーマットされたXLENビットの読み書き可能レジスタです。

トラップをMモードにすると、トラップの原因となったイベントを示すコードがmcauseに書き込まれます。

それ以外の場合、mcauseは実装によって書き込まれませんが、ソフトウェアによって明示的に書き込まれることがあります。

トラップが割り込みによって発生した場合、mcauseレジスタの割り込みビットがセットされます。

例外コードフィールドには、最後の例外を識別するコードが含まれています。

表3.6に、可能なマシンレベルの例外コードを示します。

例外コードはWLRLフィールドなので、サポートされている例外コードを保持することのみが保証されています。

XLEN-1 XLEN-2 0

|  |  |
| --- | --- |
| 割り込み | 例外コード(WLRL) |

1 XLEN-1

図3.21：マシン要因レジスタmcause。

-----------------------------------------------------------------------------------------------------------------------------------

特権命令例外と不正命令コード例外を区別しません。

これにより、アーキテクチャが簡素化され、インプリメンテーションによってどのより高い特権の命令がサポートされているかが詳細に隠されます。(非表示になります。)

トラップをサービスする特権レベルは、これらを区別する必要があるかどうか、もしそうであれば、与えられたオペコードを違法(不当)または特権として扱うべきかどうかに関するポリシーを実装できます。

-----------------------------------------------------------------------------------------------------------------------------------

割り込みは、mcauseレジスタ値の符号で単一の分岐を持つ他のトラップと分離することができます。

左にシフトすると、割込みビットを除去し、例外コードをスケーリングしてトラップベクタテーブルにインデックスすることができます。

|  |  |  |
| --- | --- | --- |
| 割り込み | 例外コード | 説明 |
| 1  1  1  1 | 0  1  2  3 | ユーザーソフトウェア割り込み  監督者ソフトウェア割り込み  予約  マシンソフトウェア割り込み |
| 1  1  1  1 | 4  5  6  7 | ユーザータイマ割り込み  監督者タイマ割り込み  予約  マシンタイマ割り込み |
| 1  1  1  1 | 8  9  10  11 | ユーザー外部割込み  監督者外部割込み  予約  マシン外部割込み |
| 1 | ≧12 | 予約 |
| 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  ≧16 | 命令アドレス不整列  命令アクセス障害  違法(違反)命令  ブレークポイント  読み込みアドレス不整列  読み込みアクセス障害  保存/AMOアドレス不整列  保存/AMOアドレス障害  Uモードからの環境呼び出し  Sモードからの環境呼び出し  予約  Mモードからの環境呼び出し  命令ページ障害  読み込みページ障害  予約  保存/AMOページ障害  予約 |

表3.6トラップ後のマシン要因レジスタ（mcause）の値

3.1.21マシントラップ値（mtval）レジスタ

mtvalレジスタは、図3.22に示すようにフォーマットされたXLENビットの読み書きレジスタです。

トラップをMモードにすると、mtvalに例外固有の情報が書き込まれ、ソフトウェアがトラップを処理する際に役立ちます。

それ以外の場合、mtvalは実装によって書き込まれることはありませんが、ソフトウェアによって明示的に書き込まれることがあります。

ハードウェアブレークポイントがトリガされるか、命令フェッチ、読み込み、または保存アドレスの不整列、アクセス、またはページフォルトの例外が発生すると、mtvalにはフォルトのある実効アドレスが書き込まれます。

不正な命令トラップでは、mtvalはフォールト中の命令の最初のXLENビットで記述されます。

その他の例外については、mtvalはゼロに設定されますが、将来の標準では、他の例外に対してmtvalの設定を再定義すること(可能性)があります。

-----------------------------------------------------------------------------------------------------------------------------------

mtvalレジスタは、前の仕様のmbadaddrレジスタを置き換えます。

不正なアドレスを提供することに加えて、レジスタは不正な命令トラップをトリガした（そして将来他の情報を返すために使用されるかもしれない）不良命令を提供できるようになりました。

命令ビットを戻す(返す)と命令エミュレーションが加速され、不正命令をエミュレートしようとするときに存在する可能性のある競合も取り除かれます。

-----------------------------------------------------------------------------------------------------------------------------------

ページベースの仮想メモリが有効な場合、物理メモリアクセスの例外の場合でも、mtvalにはフォールト仮想アドレスが書き込まれます。

この設計は、ほとんどの実装、特にハードウェアページテーブルウォーカーを持つ実装のデータパスコストを削減します。

XLEN-1 0

|  |
| --- |
| mtval |

XLEN

図3.22：マシントラップ値レジスタ。

可変長命令を持つRISC-Vシステム上の命令フェッチアクセス障害の場合、mtvalは、命令の先頭を指し示す間に、障害の原因となった命令の部分へのポインタを含みます。

オプションで、mtvalレジスタを使用して、不正命令例外（mepcがメモリ内のフォルト命令を指し示す）でフォルト命令ビットを返すこともできます。

この機能が提供されていない場合、mtvalは不正な命令フォルトでゼロに設定されます。

この機能が提供されている場合、不正な命令トラップの後で、命令がXLENビットを超えない限り、mtvalには障害発生命令全体が含まれます。

命令がXLENビットより短い場合、mtvalの上位ビットはゼロにクリアされます。

命令の長さがXLENビット以上の場合、mtvalには命令の最初のXLENビットが格納されます。

-----------------------------------------------------------------------------------------------------------------------------------

mtvalでフォールト中の命令をキャプチャすると、命令エミュレーションのオーバヘッドが減少し、命令が不整列された場合にいくつかの部分的な命令のロードが回避される可能性があり、そして、読み込みがデータレジスタに命令をフェッチするために使用されたときにデータキャッシュミスまたは低速のキャッシュされていないアクセスが発生する可能性があります。

動的翻訳システムで発生する可能性があるように、他のエージェントが命令メモリを操作している場合は、原子性の問題もあります。

必要なのは、命令全体（または少なくとも最初のXLENビット）がトラップを取る前にmtvalにフェッチされることです。

これは実装を制限するものではなく、命令のデコードを試みる前に命令全体をフェッチし、ソフトウェアハンドラの複雑さを避けることができます。

mtvalの値が0の場合、その機能がサポートされていないか、不正なゼロ命令がフェッチされたことが示されます。

mepcが指す命令メモリからの読み込みを使用して、これらの2つのケースを区別することができます（または、実行時に適切なトラップ処理をインストールするためにシステム設定情報を調べることができます）。

mtvalは、有効なすべての物理アドレスと仮想アドレスと値0を保持できる必要があるWARLレジスタです。

可能なすべての無効なアドレスを保持できる必要はありません。

実装は、無効なアドレスパターンをmtvalに書き込む前に他の無効なアドレスに変換する可能性があります。

障害のある命令ビットを返す機能が実装されている場合、mtvalは2N未満のすべての値も保持できる必要があり、NはXLENのうち小さい方とサポートされる最長命令の幅です。

– 2018/11/25

3.2マシンモード特権命令

3.2.1環境呼び出しとブレークポイント

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| funct12 | rs1 | funct3 | rd | opcode |

12 5 3 5 7

ECALL 0 PRIV 0 SYSTEM

EBREAK 0 PRIV 0 SYSTEM

ECALL命令は、支援実行環境への要求を行うために使用されます。

Uモード、Sモード、またはMモードで実行されると、Uモードからの環境呼び出し例外、Sモードからの環境呼び出し例外、またはMモードからの環境呼び出しを生成します。 他の操作を実行しません。

-----------------------------------------------------------------------------------------------------------------------------------

ECALLは、元の特権モードごとに異なる例外を生成し、環境呼び出しの例外を選択的に委任することができます。

Unixライクなオペレーティングシステムの典型的な使用例は、U モード例外からの環境呼び出しを S モードに委譲することですが、他のものではありません。

デバッガでEBREAK命令を使用して、制御をデバッグ環境に戻します。

ブレークポイント例外が生成され、他の操作は実行されません。

-----------------------------------------------------------------------------------------------------------------------------------

このマニュアルの第1巻の「C」圧縮命令の標準拡張に記載されているように、C.EBREAK命令はEBREAK命令と同じ動作を実行します。

ECALLおよびEBREAKは、受信特権モードのepcレジスタに、次の命令のアドレスではなく、ECALLまたはEBREAK命令のアドレスに設定されます。

3.2.2トラップリターン命令

トラップから戻る命令は、PRIVマイナーオペコードの下にエンコードされます。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| funct12 | rs1 | funct3 | rd | opcode |

12 5 3 5 7

MRET/SRET/URET 0 PRIV 0 SYSTEM

トラップを処理した後に戻るには、特権レベルごとに個別のトラップリターン命令があります。：MRET、SRET、およびURET。

MRETは常に提供され、スーパーバイザモードがサポートされている場合はSRETを提供する必要があります。

URETは、ユーザーモードのトラップがサポートされている場合にのみ提供されます。

xRET命令は、特権モードx以上で実行することができます。下位(低)特権xRET命令を実行すると、関連する下位(低)特権割込みの有効化および特権モードスタックがポップされます。

セクション3.1.7で説明した特権スタックの操作に加えて、xRETはpcをx epcレジスタに格納された値に設定します。

-----------------------------------------------------------------------------------------------------------------------------------

以前は、ERET命令が1つしかありませんでした（以前はSRETとも呼ばれていました）。

ユーザーレベルの割り込みの追加をサポートするために、ERET命令を使用してOSコードの古典的な仮想化を引き続き許可するために、別々のURET命令を追加する必要がありました。

次に、特権レベルごとに異なるxRET命令をサポートするために、より直交的になりました。

3.2.3割り込み待ち

Wait for Interrupt命令（WFI）は、割り込みがサービスを必要とするまで、現在のハートが停止する可能性があるというヒントを実装に提供します。

WFI命令の実行は、適切な割込みがこのハートに優先的にルーティングされるべきであることをハードウェアプラットフォームに知らせるためにも使用することができます。

WFIはサポートされているすべてのSおよびM特権モードで利用できますが、

オプションでUモード割り込みをサポートする実装のためにUモードで使用できます。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| funct12 | rs1 | funct3 | rd | opcode |

12 5 3 5 7

WFI 0 PRIV 0 SYSTEM

ハートが停止している間に、有効な割り込みが存在するか、または後で存在する場合、割り込み例外は次の命令で取られます。

すなわち、トラップハンドラおよびmepc = pc + 4で実行が再開されます。

-----------------------------------------------------------------------------------------------------------------------------------

次の命令は割り込み例外とトラップを受け取り、トラップハンドラからの単純な戻りはWFI命令の後にコードを実行します。

WFI命令はほんのヒントであり、法的な実装はWFIをNOPとして実装することです。

-----------------------------------------------------------------------------------------------------------------------------------

実装が命令の実行時にハートを停止させない場合、割り込みはWFIを含むアイドルループ内の何らかの命令で取られ(ループの命令に対して行われ)、ハンドラからの単純な復帰時には、アイドルループは実行を再開します。

-----------------------------------------------------------------------------------------------------------------------------------

我々は実装がrs1とrdフィールドを無視するという以前の要件を削除しましたが、これらのフィールドのゼロ以外の値が不正な命令例外を発生させるはずです。(0以外の値は、不正な命令例外を発生させる必要があります。)

WFI命令は、割り込みが禁止されているときに(ときにも)実行することもできます。

WFIの動作は、mstatus（MIE / SIE / UIE）および委譲レジスタ[m | s | u] idelegのグローバル割り込みビットの影響を受けないようにする必要があります

（すなわち、局所的に使用可能な割込みが保留になった場合にはあまり特権のないモードに委任されたとしてもハートを再開しなければなりません。）、しかし個々の割り込みイネーブル（例えば、MTIE）を尊重するべきです。

（すなわち、割り込みが保留されているが個々に有効にされていない場合、実装はハートを再開しないでください）。

また、WFIは、各特権レベルでのグローバル割り込み有効化に関係なく、任意の特権レベルで保留中のローカルで有効な割り込みの実行を再開する必要があります。

ハートの実行を再開させるイベントによって割り込みが発生しない場合は、PC + 4で実行が再開され、アクション可能なイベントがない場合にWFIを繰り返すためのループバックを含む、ソフトウェアが取るべきアクションを決定する必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

割り込みが無効のときにウェイクアップを許可することにより、WFIが実行される前に現在のコンテキストを保存または破棄できるため、現在のコンテキストを保存する必要がない割り込みハンドラへの代替エントリポイントを呼び出すことができます。

実装はWFIをNOPとして自由に実装できるため、ソフトウェアは、WFIに続くコード内の関連する保留中の割り込みを無効にする必要があるかどうかを明示的にチェックする必要があります。そして、適切な割り込みが検出されなかった場合はWFIにループバックする必要があります。

MIP、SIP、またはUIPレジスタは、任意のそれぞれのマシン、スーパーバイザ、またはユーザ・モードの割り込みの存在を決定するために調べることができます。

WFIの操作は、委任レジスタ設定の(によって)影響を受けません。

WFIは、実装がより高い特権モードにトラップできるように定義されており、WFIに直面(発生)した直後か或る(一定の)時間間隔の後に、例えば低電力状態への機械モード移行を開始します。

-----------------------------------------------------------------------------------------------------------------------------------

同じ「イベント待機」テンプレートは、メモリの場所(ロケーション)の変更またはメッセージ到着を待つ可能性のある将来の拡張機能に使用される可能性があります。

-- 2018/12/02

3.3リセット

リセットすると、ハートの特権モードがMに設定されます。

mstatusフィールドMIEおよびMPRVは0にリセットされます。

pcは、実装定義のリセットベクタに設定されます。

mcauseレジスタは、リセットの原因を示す値に設定されます。

他のハートの状態はすべて定義されていません。

リセット後のmcause値は実装固有の解釈を持っていますが、異なるリセット条件を区別しない実装では値0を返す必要があります。

異なるリセット条件を区別する実装では、最も完全なリセット（ハードリセットなど）を示すために0を使用する必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

一部のデザインでは、リセットの原因が複数ある場合があります。（例えば、パワーオンリセット、外部ハードリセット、ブラウンアウト検出、ウォッチドッグタイマ経過、スリープモードウェイクアップ）どのマシンモードソフトウェアとデバッガが区別したいかを知ることができます。

mcauseリセット値は、同期例外に続いてmcause値に別名を付けることがあります(指定できます)。

こリセット時にpcが他のトラップと異なる値に設定されるため、このオーバーラップにあいまい性はありません。

3.4マスカブルでない割り込み

ノンマスカブル割り込み（NMI）は、ハードウェアエラー条件でのみ使用され、ハートの割り込みイネーブルビットの状態に関係なく、Mモードで動作する実装定義のNMIベクタに即座にジャンプします。

mepcレジスタには、NMIが取られた時点で実行される次の命令のアドレスが書き込まれ、mcauseにはNMIの元を示す値が設定されます。

したがって、NMIは、アクティブなマシンモード割り込みハンドラで状態を上書きすることができます。

NMI上でmcauseに書き込まれる値は実装で定義されていますが、0の値は「不明な原因」を意味するために予約されており、mcauseレジスタを介してNMIのソースを区別しない実装は0を返すべきです(返す必要があります)。

リセットとは異なり、NMIはプロセッサの状態をリセットせず、診断、レポート、およびハードウェアエラーの包含を可能にします。

3.5物理メモリの属性

完全なシステムの物理メモリマップには、さまざまなアドレス範囲を含み、いくつかはメモリ領域に対応し、いくつかはメモリマップ制御レジスタに、いくつかはアドレス空間の空の穴になります。

一部のメモリ領域では、読み込み、書き込み、または実行をサポートしない場合があります。

サブワードアクセスやサブブロックアクセスをサポートしていないものもあります。

原子(アトミックな)操作をサポートしていないものもあります。

キャッシュ一貫性(コヒーレンス)をサポートしていないか、メモリモデルが異なる可能性があります。

同様に、メモリマップされた制御レジスタは、サポートされているアクセス幅、アトミック操作のサポート、および読み書きアクセスに関連する副作用があるかどうかによって異なります。

RISC-Vシステムでは、マシンの物理アドレス空間の各領域のこれらのプロパティと機能を物理メモリ属性（PMAs）と呼びます。

このセクションでは、RISC-V PMAの用語と、RISC-VシステムがPMAを実装して確認する方法について説明します。

PMAは、基本となるハードウェアの固有の特性であり、システム動作中にはほとんど変更されません。

セクション3.6で説明されている物理メモリ保護値とは異なり、PMAsは実行コンテキストによって変化しません。

いくつかのメモリ領域のPMAsは、例えばオンチップROM用のチップ設計時に固定されます。

他は、例えば、他のチップがオフチップバスに接続されているかどうかに応じて、基板設計時に固定されます。

オフチップ・バスは、電源サイクル（コールド・プラグ可能）またはシステム稼動中（ホット・プラガブル）に動的に変更できるデバイスもサポートします(サポートする場合もあります)。

いくつかのデバイスは、異なるPMAsを暗示(意味)する異なる用途をサポートするために、実行時に構成可能であり(可能な場合があります)、オンチップのスクラッチパッドRAMは、1つのエンドアプリケーション内の1つのコアによってプライベートにキャッシュされるか、別のエンドアプリケーション内の共有された非キャッシュメモリとしてアクセスされる可能性があります。

ほとんどのシステムでは、少なくとも一部のPMAsが物理アドレスが判明した後、実行パイプラインの後のハードウェアで動的にチェックされることが必要です。一部の操作はすべての物理メモリアドレスでサポートされないため、いくつかの操作では、設定可能なPMA属性の現在の設定を知る必要があります。

他の多くのシステムでは、仮想メモリページテーブルにPMAをいくつか指定しており、TLBを使用してこれらのプロパティをパイプラインに通知していますが、このアプローチでは、プラットフォーム固有の情報を仮想化されたレイヤーに注入し、各物理メモリ領域の各ページテーブルエントリで属性が正しく初期化されない限り、システムエラーを引き起こす可能性があります。

さらに、使用可能なページサイズが物理メモリ空間の属性を指定するのに最適ではないため(ない場合があり)、アドレス空間の断片化や高価なTLBエントリの非効率的な使用が発生する可能性があります。

RISC-Vでは、PMAsの仕様とチェックを別のハードウェア構造、PMAチェッカーに分けています。

多くの場合、属性は各物理アドレス領域のシステム設計時に認識され、PMAチェッカーにハードワイヤード接続することができます。

属性が実行可能である場合、プラットフォーム固有のメモリマップ制御レジスタを提供して、プラットフォーム上の各領域に適切な粒度でこれらの属性を指定することができます。

（例えば、キャッシュ可能なメモリとキャッシュ不可能なメモリとの間で柔軟に分割できるオンチップSRAMの場合）。

PMAsは、仮想メモリから物理メモリへの変換を経たアクセスを含む、物理メモリへのアクセスについてチェックされます。

システムのデバッグを助けるため、可能であれば、RISC-Vプロセッサは、PMAチェックに失敗した物理メモリアクセスを正確にトラップすることを強く推奨します。

正確にトラップされたPMA違反は、仮想メモリのページフォルト例外とは異なり、ロード、ストア、または命令フェッチアクセス例外として現れます。

たとえば、検出メカニズムの一部としてアクセス障害を使用する従来のバスアーキテクチャを調査する場合など、正確なPMAトラップは常に可能ではない可能性があります。

この場合、スレーブデバイスからのエラー応答は不正確なバスエラー割り込みとして報告されます。

PMAsは、特定のデバイスに正しくアクセスするため、またはDMAエンジンなど、メモリにアクセスする他のハードウェアコンポーネントを正しく構成するために、ソフトウェアで読み取る必要があります。

PMAsは特定の物理プラットフォームの組織に緊密に結びついているため、多くの詳細は本質的にプラットフォーム固有であり、ソフトウェアがプラットフォームのPMA値を知ることができる手段です。

コンフィギュレーションストリング(構成文字列)（第8章）は、オンチップデバイス用のPMAsをエンコードすることができ、また、接続されたデバイスPMAsを検出するために動的に質問することができるオフチップバス用のオンチップコントローラを記述することもできます。

いくつかのデバイス、特にレガシーバスは、PMAsの検出をサポートしていないため、サポートされていないアクセスが試行された場合にエラー応答やタイムアウトが発生します。

通常、プラットフォーム固有のマシンモードコードはPMAsを抽出し、最終的にこの情報をいくつかの標準的な表現を使用して高レベル(上位レベル)で低特権のソフトウェアに提示します。

プラットフォームがPMAsの動的再構成をサポートする場合、プラットフォームを正しく再構成できるマシン・モード・ドライバーに要求を渡すことによって、属性を設定するためのインターフェースが提供されます。

たとえば、一部のメモリー領域でキャッシュ機能を切り替えるには、マシン・モードでのみ使用可能な、キャッシュ・フラッシュなどのプラットフォーム固有の操作が必要になることがあります。

--2018/12/07

3.5.1メインメモリ対I / O対空き領域

与えられた(指定された)メモリアドレス範囲の最も重要な特徴は、それが通常のメインメモリを保持するか、またはI / Oデバイスであるか、または空であるかどうかです。

通常のメインメモリには、以下で指定する多数のプロパティが必要ですが、I / Oデバイスの属性範囲は非常に広いです(広範囲の属性を持つことができます)。

デバイスのスクラッチパッドRAMなど、通常のメインメモリに収まらないメモリ領域は、I / O領域として分類されます。

空の領域もI / O領域として分類されますが、アクセスがサポートされていないことを指定する属性があります。

3.5.2サポートされているアクセスタイプPMA

アクセスタイプは、8ビットバイトから長いマルチワードバーストまでのどのアクセス幅がサポートされているか、また各アクセス幅に対して誤整列アクセスがサポートされているかどうかを指定します。

-----------------------------------------------------------------------------------------------------------------------------------

RISC-Vハート上で動作するソフトウェアは、メモリへのバーストを直接生成することはできませんが、ソフトウェアはI / OデバイスにアクセスするためにDMAエンジンをプログラムする必要があり、したがって、どのアクセスサイズがサポートされているかを知る必要があります。

メインメモリ領域は、接続されたデバイスが必要とするすべてのアクセス幅の読み取り、書き込み、および実行を常にサポートします。

-----------------------------------------------------------------------------------------------------------------------------------

場合によっては、メインメモリにアクセスするプロセッサまたはデバイスの設計が他の幅をサポートすることもありますが、メインメモリがサポートするタイプで機能できる必要があります。

I / O領域は、データ幅がサポートされている読み取り、書き込み、または実行アクセスの組み合わせを指定できます。

3.5.3 原子性(アトミシティ)PMA

原子性PMAは、どのアトミック命令がこのアドレス領域でサポートされているかを記述します。

メインメモリ領域は、取り付けられているプロセッサが必要とするアトミック操作をサポートしなければなりません。(必要があります。)

I / O領域は、プロセッサがサポートするアトミック操作のサブセットのみをサポートすることも、サポートしないこともあります。

アトミック命令のサポートは、LR / SCとAMOsの2つのカテゴリに分かれています。

AMO内には、AMONone、AMOSwap、AMOLogical、AMOArithmeticという4つのサポートレベルがあります。

AMONoneは、AMO操作がサポートされていないことを示します。

AMOSwapは、このアドレス範囲でサポートされているのはamoswap命令だけであることを示します。

AMOLogicalは、スワップ命令とすべての論理AMOs（amoand、amoor、amoxor）がサポートされていることを示します。

AMOArithmeticは、すべてのRISC-V AMOsがサポートされていることを示します。

各レベルのサポートについては、下にあるメモリ領域がその幅の読み書きをサポートする場合、特定の幅の自然に位置合わせされたAMOsがサポートされます。

|  |  |
| --- | --- |
| AMO クラス | サポートされる動作 |
| AMONone AMO無し  AMOSwap AMO交換  AMOLogical AMO論理  AMOArithmetic AMO 算術 | 無し  amoswap  上記 ＋ amoand, amoor, amoxor  上記 ＋ amoadd, amomin, amomax, amominu, amomaxu |

表3.7：I / O領域でサポートされるAMOのクラス

メインメモリ領域は、プロセッサが必要とするすべてのAMOを常にサポートする必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

可能であれば、少なくともI / O領域のAMOLogicalサポートを提供することをお勧めします。

ほとんどのI / O領域は、LR / SCアクセスをサポートしませんが、これは、キャッシュコヒーレンス方式の上に構築するのが最も便利なためです。

3.5.4メモリ順序(オーダー)PMA

アドレス空間の領域は、フェンス命令とアトミック命令の順序付けビットによる順序付けのためにメインメモリまたはI / Oとして分類されます。

1つのハートによる主メモリ領域へのアクセスは、他のハーツだけでなく、主メモリシステム（例えば、DMAエンジン）で要求を開始する能力を有する他の装置によっても観察可能です。

メインメモリ領域は、常に標準のRISC-V緩和メモリモデルを備えています。

1つのハートによるI / O空間へのアクセスは、他のハーツやバスマスタリングデバイスだけでなく、対象のスレーブI / Oデバイスによっても観測されます。

I / O内では、領域はさらに、緩やかな順序または強い順序のいずれかを実装するものとして分類することができる(してもよい)。

リラックスしたI / O領域には、フェンスとAMO命令で強制されたものを超えて(以外の)、異なるハートやI / Oデバイスが1つのハートでのメモリアクセスをどのように観測するかについて、順序保証はありません。

強く注文(厳密に順序づけ)されたI / O領域は、ハートがその領域に行ったすべてのアクセスが、他のすべてのハーツまたはI / Oデバイスによってプログラム順にしか観察できないことを保証します。

強く順序付けられた各I/O領域は、番号付き順序付けチャネルを指定します。これは、異なるI / O領域間で順序保証が提供されるメカニズムです。

チャネル0は、ポイントツーポイントの強力な順序付けのみを示すために使用され、単一の関連するI / O領域へのアクセスのみが強く順序づけされます。

チャネル1は、すべてのI / O領域にわたってグローバルな強力な順序付けを提供するために使用されます。

ハートによるチャネル1に関連する任意のI / O領域へのアクセスは、他のすべてのハートおよびI / Oデバイスによってプログラム順序で発生したことのみが観察され、そのハートがリラックスしたI / O領域または異なるチャネル番号を有する強く順序付けされたI / O領域に対して行ったアクセスに関連するものを含みます。

換言すれば、チャネル1の領域へのアクセスは、命令の前後にフェンス(fence) io、io命令を実行することと等価です。

他のより大きいチャンネル番号は、同じチャンネル番号を有する任意の地域にわたってそのハートによってアクセスするための番組注文を提供する。

システムは、各メモリ領域上の注文プロパティの動的構成をサポートすることができる。

-----------------------------------------------------------------------------------------------------------------------------------

厳密(強力)な順序付け(並べ替え)を使用すると、従来のデバイスドライバコードとの互換性を向上させることができたり、または実装がアクセスの順序を変更しないことがわかっている場合に明示的な順序付け命令の挿入と比較してパフォーマンスを向上させることができます。

ハートとI / Oデバイスの間に1つのインオーダー通信パスしかない場合は、ローカルの強力な順序付け（チャネル0）が強力な順序付けのデフォルト形式です。

一般的に、同一の相互接続パスを共有し、パスが要求を並べ替えることができない場合、厳密(強く)順序付けされた異なるI / O領域は順序付けハードウェアを追加すること無く同じ順序付けチャネルを共有することができます。

3.5.5一貫性とキャッシュ可能性PMA

一貫性は、単一の物理アドレスに対して定義されたプロパティであり、ある(1つの)エージェントによるそのアドレスへの書き込みが、最終的にシステム内の他のエージェントから見えるようになる(する)ことを示します。

一貫性はシステムのメモリ一貫性モデルと混同されるべきではなく、それは、メモリシステム全体への読取りおよび書込みの以前の履歴が与えられたとき(履歴を元に)に、メモリ読取り値が返すことができる値を定義します。

RISC-Vプラットフォームでは、ソフトウェアの複雑さ、パフォーマンス、およびエネルギーの影響により、ハードウェア非干渉性領域の使用は推奨されません。

メインメモリとI / Oの分類、メモリの順序付け、サポートされているアクセスとアトミック操作、一貫性など、他のPMAsに反映されている違いを除いて、メモリ領域のキャッシャビリティは領域のソフトウェアビューに影響しないはずです(影響を与えるべきではありません)。

このために、私たちは、キャッシュ可能性を、マシンモードソフトウェアのみで管理されるプラットフォームレベルの設定として扱います。

プラットフォームがメモリ領域に対する構成可能なキャッシュ可能性設定(構成可能な)をサポートする場合、プラットフォーム固有のマシンモードルーチンは、必要に応じて設定を変更し、キャッシュをフラッシュし、システムはキャッシュ可能性設定間の遷移中にインコヒーレントなものに過ぎません(首尾一貫しないだけです。)。

この一時的な状態は、より低い特権レベルでは見えてはなりません(表示されません)。

-----------------------------------------------------------------------------------------------------------------------------------

RISC-Vキャッシュは、次の3つのタイプに分類されます。：マスター、プライベート、共有、およびスレーブプライベート。

マスター/プライベート・キャッシュは単一のマスター・エージェント(すなわち、メモリシステムに読み出し/書き込み要求を出すもの)に接続されています。

共有キャッシュはマスターとスレーブの間に配置され、階層的に構成されています。

スレーブプライベートキャッシュは、単一のスレーブに対してローカルであり、マスタの他のPMAsに影響を与えないため、一貫性に影響を与えません。したがって、ここでは考慮しません(考慮されません)。

特に明記しない限り、次のセクションでプライベートキャッシュをマスタープライベートキャッシュと呼びます。

一貫性は、どのエージェントによってもキャッシュされない共有メモリ領域を提供するのは簡単です。

このような領域のPMAは、それがプライベートまたは共有キャッシュにキャッシュされるべきでないことを単に示します。

一貫性は、読み取り専用領域でも簡単です。この領域は、キャッシュ一貫性スキームを必要とせずに複数のエージェントによって安全にキャッシュできます。

この領域のPMAはキャッシュできることを示しますが、書き込みはサポートされていません。

一部の読み取り/書き込み領域は単一エージェントによってのみアクセスされる可能性があり、その場合、コヒーレンススキームを必要とせずにそのエージェントによってプライベートにキャッシュされることがあります。

そのような領域のPMAは、それらをキャッシュできることを示します。

他のエージェントがその領域にアクセスすべきでないため(アクセスしないように)、データを共有キャッシュにキャッシュすることもできます。

エージェントが他のエージェントがアクセスできる読み取り/書き込み領域をキャッシュできるかどうかに関わらず、無効な値の使用を避けるためにキャッシュコヒーレンススキームが必要です。

ハードウェア・キャッシュ一貫性のない領域（ハードウェア非干渉性領域）では、キャッシュ・コヒーレンスをソフトウェアで完全に実装できますが、ソフトウェア・コヒーレンス・スキームは正しく実装するのが難しいことが知られており、保守的なソフトウェア指向キャッシュシングの必要性のためにパフォーマンスに大きな影響を与えることがよくあります。

ハードウェアキャッシュコヒーレンス方式は、より複雑なハードウェアを必要とし、キャッシュコヒーレンスプローブによるパフォーマンスに影響を与える可能性がありますが、それ以外の場合はソフトウェアからは見えません。

ハードウェア・キャッシュ一貫性領域ごとに、PMAは、領域にコヒーレント性があり、システムに複数のコヒーレンス・コントローラがある場合にどのハードウェア・コヒーレンス・コントローラを使用するかを示します。

システムによっては(一部のシステムでは)、コヒーレンス・コントローラが外部レベルの共有キャッシュであり、それ自体がさらに外部レベルのキャッシュ・コヒーレンス・コントローラに階層的にアクセスする場合があります。

プラットフォーム内のほとんどのメモリ領域は、キャッシュされていない、読み取り専用の、ハードウェアのキャッシュ一貫性のある、または1つのエージェントだけがアクセスするものとして固定されるため、ソフトウェアに一貫性があります。

3.5.6 冪等性 PMAs

冪等性 PMAは、アドレス領域への読み取りと書き込みが冪等であるかどうかを示します。

主記憶領域は冪等元であると仮定されます。

I / O領域では、読取りおよび書込みの冪等性を別々に指定することができます（例えば、読取りは冪等であるが、書込みはそうでない）。

アクセスが制限されていない(冪等で無い)場合、すなわち、読出しまたは書込みアクセスに副作用が潜在的に存在する場合、投機的または冗長アクセスは避けなければなりません。

冪等性 PMAsを定義するために、冗長アクセスによって作成された観測メモリ順序の変更は、副作用とはみなされません。

-- 冪等性（べきとうせい）は、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。ええっと、再現性があるよ。ってことでいいかな。まえに出てきたよね。（すっかり忘れてる）

-----------------------------------------------------------------------------------------------------------------------------------

ハードウェアは、非冪等としてマークされたメモリ領域への投機的または冗長アクセスを避けるように常に設計されなければなりませんが、ソフトウェアまたはコンパイラの最適化が非冪等のメモリ領域への偽のアクセスを生成しないようにすることも必要です。

-----------------------------------------------------------------------------------------------------------------------------------

非冪等領域は、整列していないアクセスをサポートしていない可能性があり、この場合、ソフトウェアは整列していないアクセスを一連の整列したアクセスとしてエミュレートし、それぞれが副作用を引き起こす可能性があります

したがって、ポータブルソフトウェアは、非冪等領域への不整合アクセスを発行してはいけません。

3.6物理メモリ保護

安全な(セキュリティで保護)処理をサポートし、障害を含むためには、ハート上で動作するソフトウェアによってアクセス可能な物理アドレスを制限することが望ましいです。

オプションの物理メモリ保護（PMP）ユニットは、ハート単位マシンモード制御レジスタを提供し、各物理メモリ領域に対して物理メモリアクセス特権（読取り、書込み、実行）を指定できるようにします。

PMP値は、セクション3.5で説明したPMAチェックと並行してチェックされます。

PMPアクセス制御設定の粒度はプラットフォーム固有であり、プラットフォーム内では物理メモリ領域によって異なる場合がありますが、標準PMPエンコードでは4バイトという小さな領域がサポートされます。

特定の領域の特権は固定式(ハードワイヤ化)にすることができます。たとえば、一部の領域はマシンモードでしか表示されないかもしれませんが、特権レベルの低い層は表示されません(存在しない可能性があります)。

-----------------------------------------------------------------------------------------------------------------------------------

プラットフォームは、物理メモリ保護の要求が大きく異なり、一部のプラットフォームでは、このセクションで説明するスキームに加えて、またはスキームの代わりに他のPMP構造を提供する(プラットフォームも)場合があります。

ハートがSモードまたはUモードで動作している場合、PMPチェックはすべてのアクセスに適用され、

mstatusレジスタにMPRVビットが設定され、mstatusレジスタのMPPフィールドにSまたはUが含まれている場合、ロードおよびストアのために使用されます。

実効特権モードがSである仮想アドレス変換のページテーブルアクセスにもPMPチェックが適用されます。

オプションで、PMPチェックがさらにMモードアクセスに適用される場合があります。その場合、PMPレジスタ自体はロックされているため、Mモードソフトウェアでも(でさえ)システムリセットなしで変更することはできません。

実際には、PMPはSとUモード（デフォルトではnone）に権限を与え、デフォルトで完全な権限を持つMモードから権限を取り消すことができます。

PMP違反は常にプロセッサーに(で)正確にトラップされます。

3.6.1物理メモリ保護のCSR

PMPエントリは、8ビット構成レジスタと1つのXLENビット・アドレス・レジスタによって記述されます。

一部のPMP設定では、前のPMPエントリに関連付けられたアドレスレジスタが追加で使用されます。

最大16のPMPエントリがサポートされています。

PMPエントリが実装されている場合は、すべてのPMP CSRsを実装する必要がありますが、すべてのPMP CSRフィールドはWARLであり、ハードウェアでゼロにすることができます(ゼロに固定される可能性があります。)。

PMP CSRsはMモードでのみアクセスできます。

PMPコンフィギュレーションレジスタはコンテキストスイッチ時間を最小限に抑えるために、密集してCSRにパックされています。

RV32の場合、4つのCSR、pmpcfg0-pmpcfg3は、図3.23に示すように、16個のPMPエントリに対応してコンフィギュレーションpmp0cfg-pmp15cfgを保持します。

RV64の場合、図3.24に示すように、pmpcfg0とpmpcfg2は16個のPMPエントリの設定を保持します。 pmpcfg1とpmpcfg3は不正です。

-----------------------------------------------------------------------------------------------------------------------------------

RV64システムでは、pmpcfg1ではなくpmpcfg2を使用して、PMPエントリ8〜15の設定を保持します。

この設計は、PMPエントリ8-11の設定がRV32とRV64の両方についてpmpcfg2 [31：0]に表示されるため、複数のM-XLEN値をサポートするコストを削減します。

31 24 23 16 15 8 7 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pmp3cfg | pmp2cfg | pmp1cfg | pmp0cfg | pmpcfg0 |

8 8 8 8

31 24 23 16 15 8 7 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pmp7cfg | pmp6cfg | pmp5cfg | pmp4cfg | pmpcfg1 |

8 8 8 8

31 24 23 16 15 8 7 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pmp11cfg | pmp10cfg | pmp9cfg | pmp8cfg | pmpcfg2 |

8 8 8 8

31 24 23 16 15 8 7 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pmp15cfg | pmp14cfg | pmp13cfg | pmp12cfg | pmpcfg3 |

8 8 8 8

図3.23：RV32 PMP構成のCSRレイアウト。

PMPアドレスレジスタは、pmpaddr0-pmpaddr15という名前のCSRsです。

各PMPアドレスレジスタは、図3.25に示すように、RV32の34ビット物理アドレスのビット33-2を符号化します。

RV64の場合、図3.26に示すように、各PMPアドレスレジスタは56ビットの物理アドレスのビット55-2をエンコードします。

すべての物理アドレスビットが実装されるわけではないので、pmpaddrレジスタはWARLです。

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pmp7cfg | pmp6cfg | pmp5cfg | pmp4cfg | pmp3cfg | pmp2cfg | pmp1cfg | pmp0cfg | pmpcfg0 |

8 8 8 8 8 8 8 8

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pmp15cfg | pmp14cfg | pmp13cfg | pmp12cfg | pmp11cfg | pmp10cfg | pmp9cfg | pmp8cfg | pmpcfg2 |

8 8 8 8 8 8 8 8

図3.24：RV64 PMP構成のCSRレイアウト。

-----------------------------------------------------------------------------------------------------------------------------------

セクション4.3で説明したSv32ページベースの仮想メモリスキームは、RV32用の34ビット物理アドレスをサポートしているため、PMPスキームはRV32用のXLENより広いアドレスをサポートする必要があります。

セクション4.4と4.5で説明されているSv39とSv48のページベース仮想メモリスキームは56ビットの物理アドレス空間をサポートしているため、RV64 PMPアドレスレジスタは同じ制限を課しています。

31 0

|  |
| --- |
| address[33:2] (WARL) |

32

63 54 53 0

|  |  |
| --- | --- |
| WIRI | addres[55:2] (WARL) |

10 54

– ここは元の図のビット幅が間違っている WIRIが32になっているが、ここは10。 コピペして修正忘れたのかな。

図3.26：PMPアドレスレジスタ形式、RV64。

図3.27にPMPコンフィグレーションレジスタのレイアウトを示します。

Rビット、Wビット、およびXビットがセットされると、PMPエントリがそれぞれ読み出し、書き込み、および命令の実行を許可することを示します。

これらのビットの1つがクリアされると、対応するアクセスタイプは拒否されます。

残りの2つのフィールドAとLについては、次のセクションで説明します。

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| L (WARL) | WIRI | A (WARL) | X (WARL) | W (WARL) | R (WARL) |

1 2 2 1 1 1

図3.27：PMP構成レジスタのフォーマット

アドレスマッチング(アドレスの一致)

PMPエントリのコンフィギュレーションレジスタのAフィールドは、関連するPMPアドレスレジスタのアドレスマッチングモードをエンコードします。

このフィールドのエンコーディングを表3.8に示します。

A = 0の場合、このPMPエントリは無効になり、アドレスと一致しません。

その他の2つのアドレスマッチングモードがサポートされています。：自然に整列した2の累乗（power-of-2）領域（NAPOT）、

自然にアライメントされた4バイト領域（NA4）の特別な場合を含む。任意の範囲（TOR）の上端の境界です。

これらのモードでは、4バイトの細分性がサポートされます。

NAPOT範囲は、表3.9に示すように、関連するアドレスレジスタの下位ビットを使用して範囲のサイズを符号化します。

TORが選択されると、関連するアドレスレジスタがアドレス範囲の先頭を形成し、先行するPMPアドレスレジスタがアドレス範囲の最下部を形成します。

|  |  |  |
| --- | --- | --- |
| A | 名前 | 説明 |
| 0  1  2  3 | OFF  TOR  NA4  NAPOT | ヌル領域（無効）  範囲のトップ  自然に整列した4バイト領域  自然に整列した2の累乗≧8バイト |

表3.8：PMPコンフィギュレーションレジスタの A フィールドのエンコーディング。

|  |  |  |
| --- | --- | --- |
| pmpaddr | pmpcfg.A | マッチタイプとサイズ |
| aaaa...aaaa  aaaa...aaa0  aaaa...aa01  aaaa...a011  …  aa01...1111  a011...1111  0111...1111 | NA4  NAPOT  NAPOT  NAPOT  ...  NAPOT  NAPOT  NAPOT | 4-byte NAPOT 範囲  4-byte NAPOT 範囲  4-byte NAPOT 範囲  4-byte NAPOT 範囲  …  2XLEN-byte NAPORT 範囲  2XLEN+1-byte NAPORT 範囲  2XLEN+2-byte NAPORT 範囲 |

表3.9：PAPアドレスと構成レジスタのNAPOT範囲の符号化。

PMPエントリi’sのAフィールドがTORに設定されている場合、そのエントリは、pmpaddr i-1 ≦ a <pmpaddr iのような任意のアドレスaに一致する。

PMPエントリ0’sのAフィールドがTORに設定されている場合、下限にゼロが使用されるため、a <pmpaddr0の任意のアドレスに一致します。

ロックモードと特権モード

Lビットは、PMPエントリがロックされていること、すなわち、構成レジスタおよび関連するアドレスレジスタへの書込みは無視される事を示します。

ロックされたPMPエントリは、システムリセットでのみロック解除できます。

PMPエントリiがロックされている場合、pmpicfgおよびpmpaddriへの書き込みは無視されます。

さらに、pmpicfg.AがTORに設定されていると、pmpaddr i-1への書き込みは無視されます。

Lビットは、PMPエントリをロックすることに加えて、MモードアクセスでR / W / Xパーミッション(アクセス許可)が強制(適用)されるかどうかを示します。

Lビットがセットされると、これらのパーミッション(アクセス許可)はすべての特権モードに対して強制(適用)されます。

Lビットがクリアされると、PMPエントリに一致するすべてのMモードアクセスは成功します。 R / W / Xパーミッション(アクセス許可)はSモードとUモードにのみ適用されます。

優先順位とマッチング(一致)ロジック

PMPエントリは静的に優先順位が付けられます。

アクセスの任意のバイトに一致する最も小さい番号のPMPエントリは、そのアクセスが成功するか失敗するかを決定します。

一致するPMPエントリは、アクセスのすべてのバイトと一致する必要があります。またはL、R、W、およびXビットに関係なく、アクセスが失敗します。 -- ここは or the access fails がどこにかかるのかな。これでいいのか。

たとえば、PMPエントリが4バイトの範囲0xC-0xFに一致するように設定されている場合、PMPエントリがそれらのアドレスに一致する最も優先順位の高いエントリであると仮定すると、0x8〜0xxの範囲への8バイトアクセスは失敗します 。

PMPエントリがアクセスのすべてのバイトに一致する場合、L、R、W、およびXビットは、アクセスが成功するか失敗するかを決定します。

Lビットがクリアされ、アクセスの特権モードがMの場合、アクセスは成功します。

それ以外の場合、Lビットがセットされているか、またはアクセスの特権モードがSまたはUである場合、アクセスタイプに対応するR、W、またはXビットがセットされている場合にのみアクセスが成功します。

Mモードアクセスに一致するPMPエントリがない場合、アクセスは成功します。

SモードまたはUモードのアクセスと一致するPMPエントリがないのに少なくとも1つのPMPエントリが実装されている場合、アクセスは失敗します。

失敗したアクセスは、ロード、ストア、または命令アクセス例外を生成します。

1つの命令が複数のアクセスを生成する可能性があり、相互にアトミックでない可能性があります(あることに注意してください)。

ある命令によって生成された少なくとも1つのアクセスが失敗した場合、アクセス例外が生成されますが、その命令によって生成された他のアクセスは可視の副作用で成功する可能性があります。

特に、仮想メモリを参照する命令は、複数のアクセスに分解されます。

いくつかの実装では、ミスアライメント(不整列)されたロード、ストア、および命令フェッチも複数のアクセスに分解され、そのうちのいくつかはアクセス例外が発生する前に成功する場合があります。

特に、別の部分がPMPチェックに失敗しても、PMPチェックをパス(通貨)する整列していないストア(保存)の一部が表示されることがあります。

ストアアドレスが自然に整列されている場合でも、XLENビット（例えば、RV32DのFSD命令）よりも広い浮動小数点ストアについては、同じ挙動が現れることがあります。

– ↑ここ3章は文章はそう長くないのだけど、意味合いがどうも分かりにくいので何となく訳が変

-- 2018/12/08

第4章

監督者レベルISA、バージョン1.10

この章では、RISC-Vスーパーバイザレベルアーキテクチャについて説明します。このアーキテクチャには、さまざまなスーパーバイザレベルアドレス変換および保護方式で使用される共通のコアが含まれています。

スーパーバイザーレベルコードは、スーパーバイザー実行環境に依存して環境を初期化し、システムバイナリインターフェース（ＳＢＩ）によって定義されたエントリポイントにスーパーバイザーコードを入力します。

SBIは、スーパーバイザレベルのコードにスーパーバイザ環境サービスを提供する機能エントリポイントも定義しています。

-----------------------------------------------------------------------------------------------------------------------------------

スーパーバイザモードは、クリーンな仮想化をサポートするために、物理メモリやデバイスの割り込みなど、基盤となる物理ハードウェアとのやり取りに関して故意(意識的)に制限されています。

4.1スーパーバイザーのCSR

スーパーバイザーには多数のCSRが用意されています。

-----------------------------------------------------------------------------------------------------------------------------------

スーパーバイザは、スーパーバイザレベルのオペレーティングシステムに認識されるべきCSRの状態のみを表示する必要があります。

特に、スーパーバイザがアクセス可能なCSRに表示される、より高い特権レベル（ハイパーバイザまたはマシン）の存在（または存在しない）についての情報はありません。

多くのスーパーバイザCSRは、同等のマシンモードCSRのサブセットです。スーパーバイザレベルのCSRの説明を理解するには、最初にマシンモードの章を読む必要があります。

4.1.1スーパーバイザステータスレジスタ（status）

sstatusレジスタは、RV32の場合は図4.1、RV64とRV128の場合は図4.2に示すようにフォーマットされたXLENビットの読み出し/書き込みレジスタです。

sstatusレジスタは、プロセッサの現在の動作状態を追跡します。

SPPビットは、監視プログラムモードに入る前にハートが実行していた特権レベルを示します。

トラップが発生すると、そのトラップがユーザモードから発生した場合はSPPが0に設定され、それ以外の場合は1に設定されます。

SRET命令（3.2.2項を参照）を実行してトラップハンドラから戻ると、特権レベルはSPPビットが0の場合はユーザモードに、SPPビットが1の場合はスーパーバイザモードに設定されます。 その後、SPPは0に設定されます。

31 30 20 19 18 17 16 15 14 13 12 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SD | WPRI | MXR | SUM | WPRI | XS[1:0] | FS[1:0] | WPRI | SPP | WPRI | SPIE | UPIE | WPRI | SIE | UIE |

1 11 1 1 1 2 2 4 1 2 1 1 2 1 1

図4.1：RV32のスーパーバイザモードステータスレジスタ（status）

XLEN-1 XLEN-2 34 33 32 31 20 19 18 17

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SD | WPRI | UXL[1:0] | WPRI | MXR | SUM | WPRI |  |

1 XLEN-35 2 12 1 1 1

16 15 14 13 12 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | XS[1:0] | FS[1:0] | WPRI | SPP | WPRI | SPIE | UPIE | WPRI | SIE | UIE |

2 2 4 1 2 1 1 2 1 1

図4.2：RV64とRV128のスーパーバイザモードステータスレジスタ（sstatus）

SIEビットは、スーパーバイザモードですべての割り込みを有効または無効にします。

SIEがクリアされていると、スーパーバイザモードでは割り込みは発生しません(実行されません)。

ハートがユーザーモードで実行されている場合、SIEの値は無視され、スーパーバイザレベルの割り込みが有効になります。

スーパーバイザは、sieレジスタを使用して個々の割り込み要因を無効にすることができます。

SPIEビットは、スーパーバイザモードにトラップする前にスーパーバイザ割り込みが有効になっていたかどうかを示します。

トラップがスーパーバイザモードになると、SPIEはSIEに設定され、SIEは0に設定されます。

SRET命令が実行されると、SIEはSPIEに設定され、その後SPIEは1に設定されます。

UIEビットは、ユーザーモード割り込みを有効または無効にします。

ユーザーレベルの割り込みは、UIEが設定されていてハートがユーザーモードで実行されている場合にのみ有効になります。

UPIEビットは、ユーザーレベルのトラップを取得する前にユーザーレベルの割り込みが有効になっていたかどうかを示します。

URET命令が実行されると、UIEはUPIEに設定され、UPIEは1に設定されます。

ユーザーレベルの割り込みはオプション(省略可能)です。

省略した場合、UIEビットとUPIEビットはゼロに固定されています。

-----------------------------------------------------------------------------------------------------------------------------------

sstatusレジスタは、mstatusレジスタのサブセットです。

簡単な実装では、sstatusで任意のフィールドを読み書きすることは、mstatusで同名フィールドを読み書きすることと同じです。

4.1.2 sstatusレジスタのベースISA 制御

UXLフィールドは、U-XLENと呼ばれるUモードのXLENの値を制御します。これは、S-XLENと呼ばれるSモードのXLENの値とは異なる場合があります。

UXLの符号化は、表3.1に示すように、misaのMXLフィールドの符号化と同じです。

RV32システムの場合、UXLフィールドは存在せず、U-XLEN = 32です。RV64およびRV128システムの場合、これはU-XLENの現在の値をエンコードするWARLフィールドです。

特に、U-XLEN = S-XLENとなるように、実装はUXLをハードワイヤリングする場合があります。

U-XLEN≠S-XLENの場合、より狭いモードで実行される命令は、構成されたXLENより上のソースレジスタオペランドビットを無視し、デスティネーションレジスタでサポートされる最も広いXLENを満たすように結果を符号拡張する必要があります。

4.1.3 sstatusレジスタのメモリ特権

MXR (Make eXecutable Readable)（eXecutableを読み取り可能にする）ビットは、ロードが仮想メモリにアクセスする際の特権を変更します。

MXR = 0の場合、読み取り可能とマークされたページ（図4.15のR = 1）からのロードのみが成功します。

MXR = 1の場合、読み取り可能または実行可能（R = 1またはX = 1）のいずれかとマークされたページからのロードは成功します。

ページベースの仮想メモリが有効になっていない場合、MXRは無効です(効果がありません)。

SUM (permit Supervisor User Memory access)（スーパーバイザユーザメモリアクセスを許可する）ビットは、Ｓモードが仮想メモリにアクセスし、ロードし、格納し、命令フェッチするための特権を変更します。

SUM = 0の場合、Uモードでアクセス可能なページへのSモードメモリアクセス（図4.15のU = 1）は失敗します。

SUM = 1の場合、これらのアクセスは許可されています。

SUMは、ページベースの仮想メモリが有効になっていないときも、Uモードで実行されているときも無効です(効果がありません)。

-----------------------------------------------------------------------------------------------------------------------------------

SUMメカニズムは、スーパーバイザソフトウェアが誤ってユーザメモリにアクセスするのを防ぎます。

オペレーティングシステムは、SUMをクリアしてコードの大部分を実行できます。； ユーザメモリにアクセスする必要がある少数のコードセグメントは、一時的にSUMを設定できます。

4.1.4スーパーバイザトラップベクタベースアドレスレジスタ（stvec）

stvecレジスタは、ベクタベースアドレス（BASE）とベクタモード（MODE）で構成されたトラップベクタ構成を保持するXLENビットの読み取り/書き込みレジスタです。

XLEN-1 2 1 0

|  |  |
| --- | --- |
| BASE[XLEN-1:2](WARL) | MODE(WARL) |

XLEN-2 2

図4.3スーパーバイザトラップベクタベースアドレスレジスタ（stvec）

stvecのBASEフィールドは、(任意の)有効な仮想アドレスまたは物理アドレスを保持できるWARLフィールドです。これには、次のような配置(整列)(アライメント)制約があります。：

アドレスは常に少なくとも4バイト境界で整列されている必要があり、MODE設定はBASEフィールドの値に追加の境界制約を課すことがあります。

|  |  |  |
| --- | --- | --- |
| 値 | 名前 | 説明 |
| 0  1  ≧2 | ダイレクト  ベクタード  － | すべての例外はpcをBASEに設定  非同期割り込みは、pcをBASE + 4 \*causeに設定  予約 |

表4.1：stvec MODEフィールドのエンコーディング

MODEフィールドの符号化(エンコーディング)を表4.1に示します。

MODE = Directのときは、スーパーバイザモードへのすべてのトラップによって、pcがBASEフィールドのアドレスに設定されます。

MODE = Vectoredの場合、スーパーバイザモードへのすべての同期例外により、pcはBASEフィールドのアドレスに設定され、割り込みが発生すると、pcはBASEフィールドのアドレスに割り込み原因番号の4倍を加えた値に設定されます。

たとえば、スーパーバイザモードのタイマ割り込み（表4.2参照）が発生すると、pcはBASE + 0x14に設定されます。

MODE = Vectoredを設定すると、BASEに追加のアライメント制約が課され、最大4xXLENバイトのアライメントが必要になります。

-----------------------------------------------------------------------------------------------------------------------------------

ベクトル割り込みが有効な場合、割り込み要因0（ユーザーモードのソフトウェア割り込みに対応）は同期例外と同じ場所にベクトル化されます。

ユーザーモードソフトウェア割り込みは無効にされるかユーザーモードに委任されるので、このあいまいさは実際には起こりません。

4.1.5スーパーバイザ割り込みレジスタ（sipとsie）

sipレジスタは、保留中の割り込みに関する情報を含むXLENビットの読み書き可能レジスタです。一方、sieは、割り込み有効ビットを含む対応するXLENビットの読み書き可能レジスタです。

XLEN-1 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| WIRI | SEIP | UEIP | WIRI | STIP | UTIP | WIRI | SSIP | USIP |

XLEN-10 1 1 ２ 1 1 ２ 1 1

図4.4スーパーバイザ割り込み保留レジスタ（sip）

XLEN-1 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| WPRI | SEIE | UEIE | WPRI | STIE | UTIE | WPRI | SSIE | USIE |

XLEN-10 1 1 ２ 1 1 ２ 1 1

図4.5：スーパーバイザ割り込み許可レジスタ（sie）

3種類の割り込みが定義されています。： ソフトウェア割り込み、タイマー割り込み、および外部割り込み。

スーパーバイザレベルのソフトウェア割り込みは、sipレジスタのスーパーバイザソフトウェア割り込み保留（SSIP）ビットに1を書き込むことによって現在のハートトでトリガされます。

保留中のスーパーバイザレベルのソフトウェア割り込みは、sipのSSIPビットに0を書き込むことでクリアできます。

SieレジスタのSSIEビットがクリアされていると、スーパーバイザレベルのソフトウェア割り込みは無効になります。

プロセッサ間割り込みはSBI呼び出しによって他のハートに送信され、それによって最終的に受信者ハートのSIPレジスタにSSIPビットが設定されます。

ユーザーレベルのソフトウェア割り込みは、sipレジスタのユーザーソフトウェア割り込み保留（USIP）ビットに1を書き込むことによって現在のハートトでトリガされます。

保留中のユーザーレベルソフトウェア割り込みは、sipのUSIPビットに0を書き込むことでクリアできます。

SieレジスタのUSIEビットがクリアされていると、ユーザーレベルのソフトウェア割り込みは無効になります。

ユーザレベルの割り込みがサポートされていない場合、USIPとUSIEはゼロに固定されています。

sipレジスタのSSIP、USIP、およびUEIP以外のすべてのビットは読み取り専用です。

SIPレジスタのSTIPビットがセットされている場合、スーパーバイザレベルのタイマ割り込みは保留中です。(保留されます)

SieレジスタのSTIEビットがクリアされている場合、スーパーバイザレベルタイマ割り込みは無効になります。

SEEへのSBI呼び出しを使用して、保留中のタイマー割り込みをクリアできます。

SIPレジスタのUTIPビットがセットされている場合、ユーザレベルのタイマ割り込みは保留中です。(保留されます)

SieレジスタのUTIEビットがクリアされている場合、ユーザレベルのタイマ割り込みは無効です。

ユーザレベルの割り込みがサポートされている場合、ABIはリアルタイムカウンタ値の観点からタイマー割り込みをスケジュールする機能を提供する必要があります。

ユーザーレベルの割り込みがサポートされていない場合、UTIPとUTIEはゼロに固定されています。

SIPレジスタのSEIPビットがセットされている場合、スーパーバイザレベルの外部割り込みは保留中です。(保留されます)

SieレジスタのSEIEビットがクリアされている場合、スーパーバイザレベルの外部割り込みは無効になります。

SBIは、外部割り込みの原因をマスク、マスク解除、および照会する機能を提供する必要があります。

sipのUEIPフィールドは単一の読み書きビットを含みます。(1つの読み取り/書き込みビットが含まれています。)

UEIPは、外部割り込みが保留中であることをUモードに示すためにSモードソフトウェアによって書き込まれます。

さらに、プラットフォームレベルの割り込みコントローラは、ユーザーレベルの外部割り込みを生成することがあります。

ソフトウェア書き込み可能ビットと外部割り込みコントローラからの信号の論理和(OR)を使用して、ユーザモード用の外部割り込みを生成します。

UEIPビットがCSRRW、CSRRS、またはCSRRC命令で読み出されると、rdデスティネーションレジスタに返される値には、ソフトウェア書き込み可能ビットと割り込みコントローラからの割り込み信号の論理和が含まれます。

ただし、CSRRSまたはCSRRC命令のリードモディファイライトシーケンスで使用される値は、ソフトウェアで書き込み可能なUEIPビットのみで、外部割り込みコントローラからの割り込み値は無視されます。

-----------------------------------------------------------------------------------------------------------------------------------

SEIPと同様に、UIEPフィールドの動作は、実際の外部割り込みを失うことなく、より高い特権層で外部割り込みを模倣できるように設計されています。

SieレジスタのUEIEビットがクリアされていると、ユーザーレベルの外部割り込みは無効になります。

ユーザレベル割り込みのためのNの拡張が実装されていない場合、UEIPとUEIEはゼロに固定されています。

-----------------------------------------------------------------------------------------------------------------------------------

sipおよびsieレジスタは、mipおよびmieレジスタのサブセットです。

sip / sieの任意のフィールドの読み取り、または書き込み可能フィールドの書き込みは、mip / mieの同名フィールドの読み取りまたは書き込みに影響します。

4.1.6スーパーバイザータイマーとパフォーマンスカウンター

スーパーバイザソフトウェアは、ユーザモードソフトウェアと同じハードウェアパフォーマンスモニタリング機能を使用します。これには、時間、サイクル、および instret CSRsなどが含まれます。

SBIはカウンター値を修正(変更)するためのメカニズムを提供するべきです。

SBIは、リアルタイムカウンタtimeの観点からタイマー割り込みをスケジュールする機能を提供する必要があります。

4.1.7カウンタイネーブルレジスタ（scounteren）

31 30 29 28 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HPM31 | HPM30 | HPM29 | ．．． | HPM5 | HPM4 | HPM3 | IR | TM | CY |

1 1 1 23 1 1 1 1 1 1

図4.6：カウンタイネーブルレジスタ（scounteren）

カウンタイネーブルレジスタscounterenは、Uモードに対するハードウェアパフォーマンスモニタリングカウンタの可用性を制御します。

scounterenレジスタ内のCY、TM、IR、またはHPMnビットがクリアされているとき、Uモードで実行中にサイクル、時間、instret、またはhpmcounternレジスタを読み込もうとすると、不正命令例外が発生します。

これらのビットの1つが設定されると、対応するレジスタへのアクセスが許可されます。

scounterenを実装する必要があります。

ただし、どのビットにもハードワイヤード値の0が含まれている場合があります。これは、対応するカウンタへの読み出しがUモードでの実行時に例外を引き起こすことを示します。

したがって、それらは事実上WARLフィールドです。← effectivelyは有効に という意味もあるけど、ここは事実上の方がいいね

4.1.8スーパーバイザースクラッチレジスタ（sscratch）

scratchレジスタは、スーパーバイザが使用する専用のXLENビット読み出し/書き込みレジスタです。

通常、sscratchは、ハートがユーザーコードを実行している間にハートローカルスーパーバイザコンテキストへのポインタを保持するために使用されます。

トラップハンドラの開始時に、sscratchはユーザレジスタと交換されて初期ワーキングレジスタを提供します。

XLEN-1 0

|  |
| --- |
| sscratch |

XLEN

図4.7スーパーバイザスクラッチレジスタ

4.1.9スーパーバイザー例外プログラムカウンタ（sepc）

sepcは図4.8に示すようにフォーマットされたXLENビット読み出し/書き込みレジスタです。

sepcの下位ビット（sepc [0]）は常にゼロです。

16ビット命令アライメントを使用した命令セット拡張をサポートしない実装では、下位2ビット（sepc [1：0]）は常にゼロです。

sepcはすべての有効な物理アドレスと仮想アドレスを保持できなければならないWARLレジスタです。

すべての無効なアドレスを保持できる必要はありません。

実装はいくつかの無効なアドレスパターンをsepcに書き込む前に他の無効なアドレスに変換するかもしれません。

トラップがSモードになると、sepcに例外が発生した命令の仮想アドレスが書き込まれます。

それ以外の場合、sepcは実装によって書き込まれることはありませんが、ソフトウェアによって明示的に書き込まれることがあります。

XLEN-1 0

|  |
| --- |
| sepc |

XLEN

図4.8スーパーバイザ例外プログラムカウンタレジスタ

4.1.10 スーパーバイザ原因レジスタ（scause）

scauseレジスタは図4.9に示すようにフォーマットされたXLENビット読み書き可能レジスタです。

トラップがSモードになると(Sモードにすると)、scauseにトラップの原因となったイベントを示すコードが書き込まれます。

そうでなければ、scauseは実装によって書かれることはありませんが、ソフトウェアによって明示的に書かれることがあります。

最後の例外を識別するコードが含まれている場合、scauseレジスタの割り込みビットが設定されます。

表4.2に、現在のスーパーバイザISAで発生する可能性のある例外コードを、優先順位の降順で示します。

例外コードはWLRLフィールドなので、サポートされている例外コードを保持することのみが保証されています。

XLEN-1 XLEN-2 0

|  |  |
| --- | --- |
| 割り込み | 例外コード(WLRL) |

1 XLEN-1

図4.9：スーパーバイザ原因レジスタ scause。

|  |  |  |
| --- | --- | --- |
| 割り込み | 例外コード | 説明 |
| 1  1  1  1  1  1  1  1  1 | 0  1  2-3  4  5  6-7  8  9  ≧10 | ユーザーソフトウェア割り込み  監督者ソフトウェア割り込み  予約  ユーザータイマ割り込み  監督者タイマ割り込み  予約  ユーザー外部割込み  監督者外部割込み  予約 |
| 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  1  2  3  4  5  6  7  8  9-11  12  13  14  15  ≧16 | 命令アドレス不整列  命令アクセス障害  違法(違反)命令  ブレークポイント  予約  読み込みアクセス障害  AMOアドレス不整列  保存/AMOアドレス障害  環境呼び出し  予約  命令ページ障害  読み込みページ障害  予約  保存/AMOページ障害  予約 |

表4.2：トラップ後のスーパーバイザ原因レジスタ（scause）値

4.1.11スーパーバイザートラップ値（stval）レジスタ

stvalレジスタは、図4.10に示すようにフォーマットされたXLENビット読み書き可能レジスタです。

トラップがSモードになると、ソフトウェアがトラップを処理できるように、stvalに例外固有の情報が書き込まれます。

そうでなければ、ソフトウェアによって明示的に書かれるかもしれませんが、stvalは実装によって書かれることは決してありません。

ハードウェアブレークポイントがトリガされた場合、命令フェッチ、ロード、ストアアクセス、ページフォルト例外が発生した場合、あるいは命令フェッチ例外またはAMOアドレスミスアライン例外が発生した場合は、stvalにフォルトアドレスが書き込まれます。

他の例外については、stvalはゼロに設定されますが、将来の規格は他の例外についてstvalの設定を再定義するかもしれません。

XLEN-1 0

|  |
| --- |
| stval |

XLEN

図4.10：スーパーバイザトラップ値レジスタ

可変長命令を使用するRISC-Vシステムでの命令フェッチアクセスフォルトおよびページフォルトの場合、stvalはフォルトを引き起こした命令の部分を指し、sepcは命令の先頭を指します。

stvalレジスタは、オプションで、不正な命令例外（sepcがメモリ内のエラーのある命令を指す）でエラーのある命令ビットを返すためにも使用できます。

この機能が提供されていない場合、不正な命令フォルトが発生するとstvalはゼロに設定されます。

この機能が提供されている場合、不正な命令トラップの後、命令がXLENビットを超えない限り、stvalにはフォルト命令全体が含まれます。

命令の長さがXLENビットより短い場合、stvalの上位ビットはゼロにクリアされます。

命令の長さがXLENビットを超える場合、stvalには命令の最初のXLENビットが含まれます。

stvalは、有効なすべての物理アドレスと仮想アドレス、および値0を保持できなければならないWARLレジスタです。

すべての無効なアドレスを保持できる必要はありません。

実装はいくつかの無効なアドレスパターンをstvalに書き込む前に他の無効なアドレスに変換するかもしれません。

誤った(障害のある)命令ビットを返す機能が実装されている場合、stvalは2N未満のすべての値を保持できなければなりません。ここでNはXLENの小さい方とサポートされる最長命令の幅です。

4.1.12スーパーバイザアドレス変換と保護（satp）レジスタ

satpレジスタは、スーパーバイザモードのアドレス変換と保護を制御する、XLVビットの読み出し/書き込みレジスタで、RV32では図4.11、図4.12では図4.12のようにフォーマットされています。

このレジスタは、ルートページテーブルの物理ページ番号（ＰＰＮ）、すなわち４ＫＢで除算されたそのスーパーバイザ物理アドレス、すなわちアドレス空間識別子（ＡＳＩＤ）を保持します。

これは、アドレススペースごとのアドレス変換フェンスと、現在のアドレス変換方式を選択するMODEフィールドを容易にします。

31 30 22 21 0

|  |  |  |
| --- | --- | --- |
| MODE(WARL) | ASID(WARL) | PPN(WARL) |

1 9 22

図4.11：RV32スーパーバイザアドレス変換および保護レジスタ

-----------------------------------------------------------------------------------------------------------------------------------

PPNを物理アドレスではなくsatpに格納すると、RV32では4 GiBを超える物理アドレス空間がサポートされます。

-----------------------------------------------------------------------------------------------------------------------------------

ASIDとページテーブルのベースアドレスを同じCSRに格納して、コンテキストスイッチでペアをアトミックに変更できるようにします。

それらを非アトミックに交換すると、古い仮想アドレス空間が新しい変換で汚染される可能性があります。その逆も同様です。

このアプローチはまたコンテキストスイッチのコストをわずかに削減します。

63 60 59 44 43 0

|  |  |  |
| --- | --- | --- |
| MODE(WARL) | ASID(WARL) | PPN(WARL) |

4 16 44

図4.12：MODE値Sv39およびSv48のRV64スーパーバイザアドレス変換および保護レジスタsatp。

表4.3にRV32とRV64のMODEフィールドのエンコーディングを示します。

MODE = Bareのとき、スーパーバイザー仮想アドレスはスーパーバイザー物理アドレスと等しく、セクション3.6で説明されている物理メモリー保護方式を超える追加のメモリー保護はありません。

この場合、satpの残りのフィールドは効果がありません。

RV32の場合、MODEの他の唯一の有効な設定は、セクション4.3で説明されているページング仮想メモリ方式であるSv32です。

RV64では、2つのページ仮想メモリ方式が定義されています。；Sv39とSv48で、それぞれセクション4.4と4.5で説明されています。

2つの追加スキーム、Sv57とSv64がこの仕様の今後のバージョンで定義される予定です。

残りのMODE設定は将来の使用のために予約されており、satpの他のフィールドの異なる解釈を定義するかもしれません。

すべてのMODE設定をサポートするための実装は必須ではありません。また、satpがサポートされていないMODEで書き込まれた場合、書き込み全体は効果がありません。 satp内のフィールドは変更されません。

↑ 書き込み全体は効果がありません は 書き込み全体が無効になります。 のような意味かな

|  |  |  |
| --- | --- | --- |
| RV32 | | |
| 値 | 名前 | 説明 |
| 0  1 | Bare  Sv32 | 変換や保護はありません。  ページベースの32ビット仮想アドレス指定。 |

|  |  |  |
| --- | --- | --- |
| RV64 | | |
| 値 | 名前 | 説明 |
| 0  1-7  8  9  10  11  12-15 | Bare  ー  Sv39  Sv48  Sv57  Sv64  ー | 変換や保護はありません。  予約  ページベースの39ビット仮想アドレス指定。  ページベースの48ビット仮想アドレス指定。  ページベースの57ビット仮想アドレス指定用に予約済み。  ページベースの64ビット仮想アドレス指定用に予約済み。  予約 |

表4.3：satp MODEフィールドの符号化

スーパーバイザ物理アドレスビットの数は実装によって定義されています。 未実装のアドレスビットは、satpレジスタで0に固定されています。

ASIDビットの数もインプリメンテーション定義であり、ゼロの場合もあります。

ASIDLENと呼ばれる、実装されたASIDビットの数は、ASIDフィールドのすべてのビット位置に1を書き込むことによって決定できます。

次にsatpの値を読み戻して、ASIDフィールドのどのビット位置に1が入っているかを調べます。

ASIDの最下位ビットが最初に実装されます。つまり、ASIDLEN> 0の場合、ASID [ASIDLEN-1：0]は書き込み可能です。

ASIDMAXと呼ばれるASIDLENの最大値は、Sv32では9、Sv39およびSv48では16です。

-----------------------------------------------------------------------------------------------------------------------------------

多くのアプリケーションにとって、ページサイズの選択はパフォーマンスに大きな影響を与えます。

大きなページサイズはTLBの範囲を広げ、仮想的に索引付けされた、物理的にタグ付けされたキャッシュに対する結合性の制約を緩和します。

同時に、大きなページは内部の断片化を悪化させ、物理メモリとおそらくキャッシュ容量を無駄にします。

多くの検討(審議)の結果、RV32とRV64の両方について、従来の4KiBのページサイズに決定(定住)しました。

この決定により、低レベルのランタイムソフトウェアとデバイスドライバの移植が容易になると期待されます。

TLBリーチの問題は、現代のオペレーティングシステムにおける透過的なスーパーページサポートによって改善されています[2]。

さらに、マルチレベルTLB階層は、アドレス空間がマッピングされているマルチレベルキャッシュ階層に比べて非常に安価です。

satpを書いても、ページテーブルの更新とそれに続くアドレス変換の間に順序の制約があるわけではないことに注意してください。

新しいアドレス空間のページテーブルが変更されている場合は、satpを書き込む前にSFENCE.VMA命令（セクション4.2.1を参照）を実行する必要があるかもしれません。

-----------------------------------------------------------------------------------------------------------------------------------

十分な大きさのASIDスペースが提供されるならば、satt書き込みの際にアドレス変換キャッシュをフラッシュする実装を強制しないことはコンテキストスイッチのコストを削減します。

4.2監督者の指示

セクション3.2.2で定義されているSRET命令に加えて、1つの新しいスーパーバイザレベルの命令が提供されています。

4.2.1スーパーバイザメモリ管理フェンス命令

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| funct7 | rs2 | rs1 | funct3 | rd | opcode |

7 5 5 3 5 7

SFENCE.VMA asid vaddr PRIV 0 SYSTEM

スーパーバイザメモリ管理フェンス命令SFENCE.VMAは、現在の実行とインメモリメモリ管理データ構造の更新を同期させるために使用されます。

命令実行はこれらのデータ構造への暗黙の読み書きを引き起こします。 しかし、これらの暗黙の参照は通常、命令ストリーム内のロードおよびストアに関して順序付けされていません。

SFENCE.VMA命令を実行すると、SFENCE.VMAより前の命令ストリーム内のすべてのストアが、SFENCE.VMAに続くすべての暗黙参照の前に順序付けされることが保証されます。

-----------------------------------------------------------------------------------------------------------------------------------

SFENCE.VMAは、アドレス変換に関連するローカルハードウェアキャッシュを消去するために使用されます。

TLBフラッシュではなくフェンスとして指定されており、どの命令がフラッシュ操作の影響を受けるかに関してより明確なセマンティクスを提供し、より多様な動的キャッシュ構造およびメモリ管理方式をサポートします。

SFENCE.VMAは、ページテーブル書き込みとアドレス変換ハードウェアを同期させるために、より高い特権レベルによっても使用されます。

-----------------------------------------------------------------------------------------------------------------------------------

この命令は他のRISC-Vスレッドの変換には影響しません。これらのスレッドは別々に通知される必要があります。

1つの方法は、1）ローカル書き込みがグローバルに見えるようにするためにローカルデータフェンスを使用し、次に2）他のスレッドへのプロセッサ間割り込み、3）リモートスレッドの割り込みハンドラ内でローカルSFENCE.VMAを使用します。4）操作が完了したことを元のスレッドに知らせます。

これは、もちろん、TLBの撃墜に対するRISC-Vの類似物です。

ネイティブではないが、実装はリモートTLB無効化のための直接的なハードウェアサポートを提供するかもしれません。

TLBの撃墜は、実装の詳細を隠すためのSBI呼び出しによって処理されます。

↑the RISC-V analog to a TLB shootdown. ってどう訳すのがいいんだろ

変換データ構造が単一アドレスマッピング（すなわち、１ページまたはスーパーページ）に対してのみ修正(変更)されているという一般的な場合(ケースでは)には、ｒｓ１はそのマッピング内の仮想アドレスを指定してそのマッピングに対してのみ変換フェンスをもたらす(有効にする)ことができます。

さらに、変換データ構造が単一のアドレス空間識別子に対してのみ変更されているという一般的な場合には、rs2はアドレ​​ス空間を指定できます。

SFENCE.VMAの動作は、次のようにrs1とrs2に依存します。

* rs1 = x0およびrs2 = x0の場合、フェンスはすべてのアドレス空間について、任意のレベルのページテーブルに対して行われたすべての読み取りおよび書き込みを順序付けます。
* rs1 = x0かつrs2≠x0の場合、フェンスはページテーブルの任意のレベルに対して行われたすべての読み取りおよび書き込みを順序付けますが、整数レジスタrs2によって識別されるアドレス空間に対してのみです。

グローバルマッピングへのアクセス（4.3.1項を参照）は順序付けられていません。

* ｒｓ１≠ｘ０かつｒｓ２ ＝ ｘ０の場合、フェンスは、すべてのアドレス空間について、ｒｓ１内の仮想アドレスに対応するリーフページテーブルエントリに対して行われた読み取りおよび書き込みのみを行います。
* ｒｓ１≠ｘ０およびｒｓ２≠ｘ０の場合、フェンスは、整数レジスタｒｓ２によって識別されるアドレス空間に対して、ｒｓ１内の仮想アドレスに対応するリーフページテーブルエントリに対して行われた読み取りおよび書き込みのみを行います。

グローバルマッピングへのアクセスは順序付けられていません。

rs2≠x0の場合、rs2に保持されている値のビットXLEN-1：ASIDMAXは将来の使用のために予約されており、ソフトウェアでゼロにして現在の実装では無視する必要があります。

さらに、ASIDLEN <ASIDMAXの場合、実装は、rs2に保持されている値のASIDMAX-1：ASIDLENビットを無視します。

-----------------------------------------------------------------------------------------------------------------------------------

より単純な実装では、rs1の仮想アドレスとrs2のASID値を無視して、常にグローバルフェンスを実行できます。

4.3 Sv32：ページベースの32ビット仮想メモリシステム

Sv32がsatpレジスタのMODEフィールドに書き込まれると（セクション4.1.12を参照）、スーパーバイザは32ビットページ仮想メモリシステムで動作します。

Sv32はRV32システムでサポートされており、現代のUnixベースのオペレーティングシステムをサポートするのに十分なメカニズムを含むように設計されています。

-----------------------------------------------------------------------------------------------------------------------------------

初期のRISC-Vページ仮想メモリアーキテクチャは、既存のオペレーティングシステムをサポートするための直接的な(簡単な)実装として設計されています。

ハードウェアのページテーブルウォーカーをサポートするために、ページテーブルレイアウトを設計しました。

ソフトウェアTLBの詰め替えは、高性能システムではパフォーマンスのボトルネックになり、特に分離された専用コプロセッサでは面倒です。

実装は、Mモードの拡張としてマシンモードトラップハンドラを使用してソフトウェアTLBリフィルを実装することを選択できます。

4.3.1アドレス指定とメモリ保護

Sv32実装は、4 KiBページに分割された32ビットの仮想アドレス空間をサポートします。

図4.13に示すように、Sv32仮想アドレスは仮想ページ番号（VPN）とページオフセットに分割されています。

satpレジスタのMODEフィールドでSv32仮想メモリモードが選択されている場合、スーパーバイザ仮想アドレスは2レベルのページテーブルを介してスーパーバイザ物理アドレスに変換されます。

20ビットのVPNは22ビットの物理ページ番号（PPN）に変換され、12ビットのページオフセットは変換されません。

結果として得られるスーパーバイザレベルの物理アドレスは、その後、直接マシンレベルの物理アドレスに変換される前に、任意の物理メモリ保護構造（セクション3.6）を使用してチェックされます。

31 22 21 12 11 0

|  |  |  |
| --- | --- | --- |
| VPN[1] | VPN[0] | ページオフセット |

10 10 12

図4.13：Sv32仮想アドレス

33 22 21 12 11 0

|  |  |  |
| --- | --- | --- |
| PPN[1] | PPN[0] | ページオフセット |

12 10 12

図4.14：Sv32物理アドレス

31 20 19 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PPN[1] | PPN[0] | RSW | D | A | G | U | X | W | R | V |

12 10 2 1 1 1 1 1 1 1 1

図4.15：Sv32ページテーブルエントリ

Sv32ページテーブルは、それぞれ4バイトの210ページテーブルエントリ（PTE）で構成されています。

ページテーブルは正確にページのサイズであり、常にページ境界に合わせて配置する必要があります。

ルートページテーブルの物理ページ番号はsatpレジスタに格納されます。

Sv32のPTEフォーマットを図4.15に示します。

VビットはPTEが有効かどうかを示します。 それが0であるならば、PTEのビット31-1は気にせず、ソフトウェアによって自由に使用されるかもしれません。

許可ビットR、W、およびXは、それぞれページが読み取り可能、書き込み可能、および実行可能かどうかを示します。

3つすべてがゼロの場合、PTEはページテーブルの次のレベルへのポインタです。 それ以外の場合は、リーフPTEです。

↑リーフPTE って ずらずらと繋がっているPTEの一番端っこの、最後のPTEってことだね

書き込み可能ページも読み取り可能としてマークする必要があります。 反対の組み合わせは将来の使用のために予約されています。

表4.4に許可ビットの符号化をまとめます。

|  |  |  |  |
| --- | --- | --- | --- |
| X | W | R | 意味 |
| 0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1 | ページテーブルの次のレベルへのポインタ。  読み取り専用ページ。  将来の使用のために予約済み。  読み書きページ。  実行専用ページ。  読み取り実行ページ。  将来の使用のために予約済み。  読み書き実行ページ |

表4.4：PTE R / W / Xフィールドの符号化

Uビットは、ページがユーザーモードにアクセス可能かどうかを示します。

Uモードソフトウェアは、U = 1の場合にのみページにアクセスできます。

sstatusレジスタのSUMビットが設定されている場合、スーパーバイザモードソフトウェアもU = 1のページにアクセスする可能性があります。(アクセスすることもできます。

ただし、スーパーバイザコードは通常SUMビットをクリアした状態で動作します。その場合、スーパーバイザコードはユーザーモードページへのアクセス時に障害になります。（障害を発生させることになります。）

-----------------------------------------------------------------------------------------------------------------------------------

代替PTEフォーマットは、スーパーバイザーとユーザーに対して異なる許可をサポートします。

このメカニズムは、SUMメカニズム（セクション4.1.3を参照）とほぼ重複しているため(大幅に冗長化され)、PTEでより多くのエンコードスペースが必要になるため、この機能を省略しました。

Gビットはグローバルマッピングを指定します。

グローバルマッピングは、すべてのアドレス空間に存在するマッピングです。

非リーフPTEの場合、グローバル設定は、ページテーブルの後続レベルのすべてのマッピングがグローバルであることを意味します。

大域マッピングを大域としてマークしないとパフォーマンスが低下するだけで、非大域マッピングを大域としてマークするとエラーになることに注意してください。

-----------------------------------------------------------------------------------------------------------------------------------

グローバルマッピングは、複数のASIDsのアドレス変換キャッシュに重複して格納する必要はありません。

さらに、SFENCE.VMA命令がrs2≠x0で実行されるときにローカルアドレス変換キャッシュからフラッシュする必要はありません。

RSWフィールドは、スーパーバイザソフトウェアで使用するために予約されています。；

実装はこの分野を無視するものとします。

各リーフＰＴＥはアクセスされた（Ａ）およびダーティ（Ｄ）ビットを含む。

Aビットは、最後にAビットがクリアされてから仮想ページが読み取られた、書き込まれた、または取り出されたことを示します。

Dビットは、最後にDビットがクリアされてから仮想ページが書き込まれたことを示します。

AビットとDビットを管理するための2つの方式が許可されています。

* 仮想ページがアクセスされ、Aビットがクリアされている、または書き込まれていてDビットがクリアされている場合、実装はPTE内の対応するビットを設定します。

ＰＴＥの更新は、ＰＴＥへの他のアクセスに関してアトミックでなければならず、ＰＴＥが有効であることをアトミックにチェックし、十分な許可を与えなければなりません。

ＰＴＥの更新は正確でなければならず（すなわち投機的ではなく）、そして地方のハートによってプログラム順に観察されなければなりません。

フェンス命令によって提供されるロードおよびストアの順序付け、およびアトミック命令の獲得／解放ビットもまた、リモートハートによって観察されるようにそれらのロードおよびストアに関連するＰＴＥ更新を順序付けます。

* 仮想ページがアクセスされてAビットがクリアされているか、または書き込まれていてDビットがクリアされていると、ページフォルト例外が発生します。

一方または両方のPTE更新方式が有効である可能性があると想定して、標準スーパーバイザソフトウェアを作成する必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

PTEの更新を正確、アトミック、プログラム順に更新することを指定(命令)すると、仕様が単純化され、システムソフトウェアにとってこの機能がより便利になります。

単純な実装では、代わりにページフォルト例外が発生する可能性があります。

AおよびDビットは実装によって決してクリアされません。

スーパーバイザソフトウェアがアクセスビットおよび／またはダーティビットに依存しない場合、例えば メモリー・ページを2次記憶域にスワップしない場合、またはページが入出力領域のマップに使用されている場合は、パフォーマンスを向上させるためにPTEで常にそれらを1に設定する必要があります。

任意のレベルのPTEがリーフPTEになる可能性があるため、4 KiBページに加えて、Sv32は4 MiBメガページをサポートします。

メガページは、仮想的かつ物理的に4 MiB境界に合わせて配置する必要があります。

物理アドレスのアライメント(整列)が不十分な場合、ページフォルト例外が発生します。

非リーフPTEの場合、D、A、およびUビットは将来の使用のために予約されており、前方互換性のためにソフトウェアでクリアする必要があります。

4.3.2仮想アドレス変換プロセス

仮想アドレスvaは、次のように物理アドレスpaに変換されます。

1. aをsatp：ppn×PAGESIZEとし、i = LEVELS - 1とする（Sv32の場合、PAGESIZE = 212およびLEVELS = 2）。

2.アドレスa + vaにあるPTEの値をpteとします。vpn [i]×PTESIZE。 （Sv32の場合、PTESIZE = 4）

pteにアクセスしてPMAまたはPMPチェックに違反した場合は、アクセス例外を発生させます。

3. pte.v = 0の場合、またはpte.r = 0かつpte.w = 1の場合、停止してページフォルト例外を発生させます。

4.それ以外の場合、PTEは有効です。

pte：r = 1またはpte：x = 1の場合、手順5に進みます。

それ以外の場合、このPTEはページテーブルの次のレベルへのポインタです。

ｉ ＝ ｉ － １とします。

i <0の場合、停止してページ不在例外を発生させます。

それ以外の場合は、a = pte.ppn×PAGESIZEとし、手順2に進みます。

5.リーフPTEが見つかりました。

現在の特権モードと、mstatusレジスタのSUMおよびMXRフィールドの値を考慮して、要求されたメモリアクセスが

pte.r、pte.w、pte.x、およびpte.uビットによって許可されているかどうかを確認します。

そうでない場合は、停止してページ不在例外を発生させます。

6. i> 0かつpa.ppn [i － 1：0]≠0の場合、これは位置ずれしたスーパーページです。

ページフォルト例外を停止して発生させます。

7. pte.a = 0の場合、またはメモリアクセスがストアでpte.d = 0の場合、ページフォルト例外が発生するか、または、

・pte.aを1に設定し、メモリアクセスがストアの場合はpte.dも1に設定します。

・このアクセスがPMAまたはPMPチェックに違反している場合は、アクセス例外を発生させます。

・この更新と手順2のpteのロードはアトミックでなければなりません。

特に、PTEへの介在ストアがその間に発生したと認識されることはありません。

8.変換は成功しました。

変換された物理アドレスは以下のように与えられます。

・pa.pgoff = va.pgoff

・i> 0の場合、これはスーパーページ変換であり、pa：ppn [i - 1：0] = va：vpn [i - 1：0]です。

・pa.ppn [LEVELS - 1：i] = pte.ppn [LEVELS - 1：i]

4.4 Sv39：ページベースの39ビット仮想メモリシステム

このセクションでは、RV64システム用に設計された、39ビットの仮想アドレス空間をサポートする単純なページ仮想メモリシステムについて説明します。

Sv39の設計はSv32の全体的なスキームに従います、そしてこのセクションはスキーム間の違いだけを詳述します。

-----------------------------------------------------------------------------------------------------------------------------------

RV64用に複数の仮想メモリシステムを指定して、大きなアドレス空間を提供することとアドレス変換コストを最小限に抑えることとの間の緊張(張力、テンション)を緩和しました。

多くのシステムでは、512 GiBの仮想アドレス空間で十分なので、Sv39で十分です。

Sv48は仮想アドレス空間を256 TiBに増やしますが、ページテーブル専用の物理メモリ容量、ページテーブル横断の待ち時間、そして仮想アドレスを格納するハードウェア構造のサイズを増やします。

4.4.1アドレス指定とメモリ保護

Sv39実装は、4 KiBページに分割された39ビットの仮想アドレス空間をサポートします。

図4.16に示すようにSv39アドレスは分割されています。

64ビットである実効アドレスをロードして格納するには、ビット63から39をすべてビット38と等しくなければなりません。そうしないと、ページ不在例外が発生します。

27ビットのVPNは3レベルのページテーブルを介して44ビットのPPNに変換されますが、12ビットのページオフセットは変換されません。

38 30 29 22 21 12 11 0

|  |  |  |  |
| --- | --- | --- | --- |
| VPN[2] | VPN[1] | VPN[0] | ページオフセット |

9 9 9 12

図4.16：Sv39の仮想アドレス

55 30 29 22 21 12 11 0

|  |  |  |  |
| --- | --- | --- | --- |
| PPN[2] | PPN[1] | PPN[0] | ページオフセット |

26 9 9 12

図4.17：Sv39の物理アドレス

63 54 53 28 27 19 18 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 予約 | PPN[2] | PPN[1] | PPN[0] | RSW | D | A | G | U | X | W | R | V |

10 26 9 9 2 1 1 1 1 1 1 1 1

図4.18：Sv39ページテーブルエントリ

Sv39ページテーブルには、それぞれ8バイトの29個のページテーブルエントリ（PTE）が含まれています。

ページテーブルは正確なページのサイズであり、常にページ境界に合わせて配置する必要があります。

ルートページテーブルの物理ページ番号は、satpレジスタのPPNフィールドに格納されています。

Sv39のPTEフォーマットを図4.18に示します。

ビット9〜0はSv32と同じ意味です。

ビット63〜54は将来の使用のために予約されており、前方互換性のためにソフトウェアでゼロにする必要があります。

-----------------------------------------------------------------------------------------------------------------------------------

ページテーブルレベルをスキップできるようにすることでスパースアドレス空間のサポートを改善し、メモリ使用量とTLBリフィル(再充填)待ち時間を減らすことができる拡張のために、いくつかのPTEビットを予約しました。

これらの予約ビットはまた、研究実験を容易にするために使用されてもよいです。

コストは物理アドレス空間を減らしていますが、64 PiBは現在十分です。

それがもはや十分でないとき、未割り当てのままの予約ビットは物理アドレス空間を拡張するために使用されるかもしれません。

どのレベルのPTEでもリーフPTEになる可能性があるため、Sv39は4 KiBページに加えて、2 MiBメガページと1 GiBギガページをサポートしています。それぞれは、そのサイズに等しい境界に仮想的かつ物理的に配置されている必要があります。

物理アドレスのアライメント(整列)が不十分な場合、ページフォルト例外が発生します。

仮想アドレスから物理アドレスへの変換アルゴリズムは、LEVELSが3、PTESIZEが8である点を除いて、4.3.2項と同じです。

4.5 Sv48：ページベースの48ビット仮想メモリシステム

このセクションでは、48ビット仮想アドレス空間をサポートする、RV64システム用に設計された単純なページ仮想メモリシステムについて説明します。

Sv48は39ビットの仮想アドレス空間が不十分なシステムを対象としています。

これはSv39の設計に厳密に従っており、単にページテーブルのレベルを追加しているだけなので、この章では2つの方式の違いについてのみ詳しく説明します。

Sv48をサポートする実装はSv39もサポートするはずです。

-----------------------------------------------------------------------------------------------------------------------------------

Sv48をサポートするシステムは、基本的に無料(コスト無しで)でSv39もサポートできます。したがって、Sv39を前提とするスーパーバイザソフトウェアとの互換性を維持するために、そうする必要があります。

4.5.1アドレス指定とメモリ保護

Sv48の実装は、4 KiBページに分割された48ビットの仮想アドレス空間をサポートしています。

Sv48アドレスは図4.19のように分割されています。

64ビットの実効アドレスをロードして格納するには、ビット63〜48をすべてビット47と等しくなければなりません。そうしないと、ページ不在例外が発生します。

36ビットのVPNは4レベルのページテーブルを介して44ビットのPPNに変換されますが、12ビットのページオフセットは変換されません。

47 39 38 30 29 22 21 12 11 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VPN[3] | VPN[2] | VPN[1] | VPN[0] | ページオフセット |

9 9 9 9 12

図4.19：Sv48の仮想アドレス

55 39 38 30 29 22 21 12 11 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PPN[3] | PPN[2] | PPN[1] | PPN[0] | ページオフセット |

17 9 9 9 12

図4.20：Sv48の物理アドレス

63 54 53 37 36 28 27 19 18 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 予約 | PPN[3] | PPN[2] | PPN[1] | PPN[0] | RSW | D | A | G | U | X | W | R | V |

10 17 9 9 9 2 1 1 1 1 1 1 1 1

図4.21：Sv48ページテーブルエントリ

Sv48のPTEフォーマットを図4.21に示します。

ビット9〜0はSv32と同じ意味です。

どのレベルのPTEでもリーフPTEになる可能性があるため、Sv48は4 KiBページに加えて、2 MiBメガページ、1 GiBギガページ、および512 GiBテラページをサポートしています。それぞれは、そのサイズに等しい境界に仮想的かつ物理的に配置されている必要があります。

物理アドレスのアライメント(整列)が不十分な場合、ページフォルト例外が発生します。

仮想アドレスから物理アドレスへの変換アルゴリズムは、LEVELSが4、PTESIZEが8である点を除いて、4.3.2項と同じです。

-- 2018/12/22

第5章

ハイパーバイザー拡張、バージョン0.0

この章は、拡張Sモードを使用したRISC-Vハイパーバイザーサポートのプレースホルダーです。

↑placeholderってなんだ？

-----------------------------------------------------------------------------------------------------------------------------------

特権アーキテクチャは、少数の特権命令を簡単に検出してトラップできるため、ゲストOSをユーザーレベルで実行するという従来の仮想化技術の使用を簡素化するように設計されています。

-- 1ページ空き、次ページへ

第6章

RISC-V特権命令セットのリスト

この章では、RISC-V特権アーキテクチャで定義されているすべての命令の命令セットの一覧を示します。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| imm[11:0] | rs1 | funct3 | rd | opcode | I-型 |

環境呼び出しとブレークポイント

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 000000000000 | rs1 | 000 | 00000 | 1110011 | ECALL |
| 000000000001 | rs1 | 000 | 00000 | 1110011 | EBREAK |

トラップ返却命令

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0000000 | 00010 | 00000 | 000 | 00000 | 1110011 | URET |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0001000 | 00010 | 00000 | 000 | 00000 | 1110011 | SRET |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0011000 | 00010 | 00000 | 000 | 00000 | 1110011 | MRET |

割り込み管理命令

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0001000 | 00101 | 00000 | 000 | 00000 | 1110011 | WFI |

メモリ管理命令

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0001001 | rs2 | rs1 | 000 | 00000 | 1110011 | SFENCE.VMA |

表6.1：RISC-Vの特権命令

第7章

プラットフォームレベル割り込みコントローラ（PLIC）

この章では、RISC-Vシステムでグローバル割り込みの優先順位付けと分配を行うRISC-Vプラットフォームレベルの割り込みコントローラ（PLIC）の一般的なアーキテクチャについて説明します。

7.1 PLICの概要

図7.1にPLIC動作の概要を示します。

PLICは、通常はI / Oデバイスであるグローバル割り込みソースを、通常はハードコンテキストである割り込みターゲットに接続します。

PLICには、割り込みの優先順位付けとルーティングを実行するPLICコアとともに、割り込みソースごとに1つずつ、複数の割り込みゲートウェイが含まれています。

グローバル割り込みは、各ソースからの割り込み信号を処理してPLICコアに割り込み要求を送信する割り込みゲートウェイに送信され、PLICコアはこれらをコア割り込み保留ビット（IP）にラッチします。

各割り込み要因には個別の優先順位が割り当てられています。

PLICコアには、各ターゲットに対して有効にされている割り込みを選択するためのマトリックスの割り込み有効（IE）ビットが含まれています。

ターゲットで保留中の割り込みが有効になっていて、保留中の割り込みの優先度がターゲットごとのしきい値を超えると、PLICコアは1つまたは複数のターゲットに割り込み通知を転送します。

ターゲットが外部割り込みを受け取ると、そのターゲットに対して保留中の最高優先順位グローバル割り込みソースの識別子を取得するための割り込み要求要求をPLICコアから送信し、PLICコアは対応する割り込みソース保留中ビットをクリアします。

ターゲットは割り込みを処理した後、関連する割り込みゲートウェイに割り込み完了メッセージを送信し、割り込みゲートウェイは同じソースに対する別の割り込み要求をPLICに転送できます。

この章の残りの部分では、これらの各コンポーネントについて詳細に説明しますが、多くの詳細は必ずプラットフォーム固有のものです。

7.2割り込み要因

RISC-Vハートは、ローカルとグローバルの両方の割り込みソースを持つことができます。

グローバル割り込みソースのみがPLICによって処理されます。

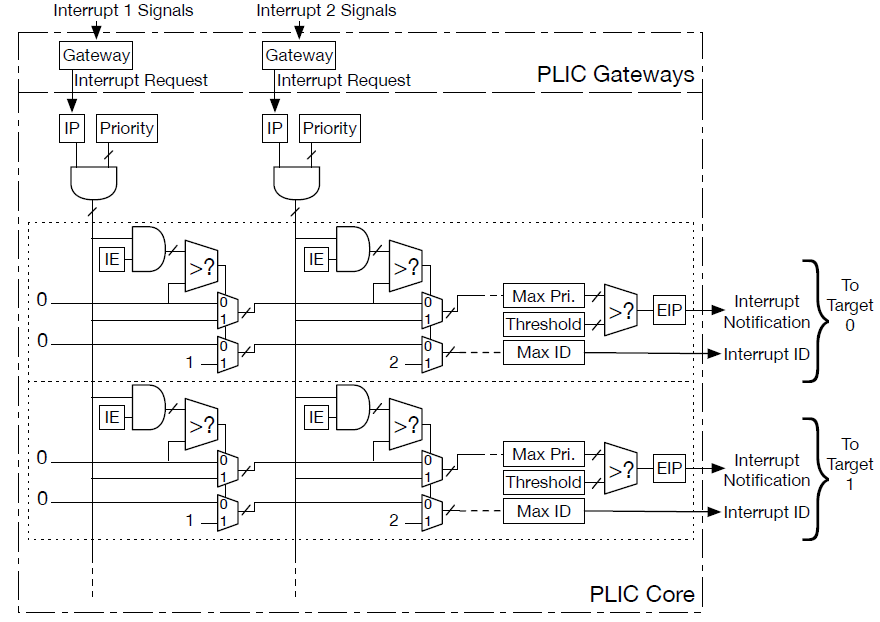


図7.1：プラットフォームレベル割り込みコントローラ（PLIC）の概念ブロック図

この図は、潜在的に多数の割り込みソースの最初の2つ、および潜在的に多数の割り込みターゲットの最初の2つを示しています。

この図は、PLICの動作のロジックを示すことを目的としており、現実的な実装戦略を表すものではありません。

7.2.1ローカル割り込み要因

各ハートには、各特権レベルの標準ソフトウェア割り込みやタイマー割り込みなど、PLICを通過しない多数のローカル割り込みソースがあります。

発信元とサービス提供ハートとの間の待ち時間が最小になるので、ローカル割り込みは迅速にサービス提供され、どのハートが要求を処理するかを決定するために仲裁は必要なく、そして、保守サービス担当者は、mcauseレジスタを使用して割り込みソースを迅速に判断できます。

すべてのローカル割り込みは、mip内の対応するビットが設定されている場合に割り込みが保留されているレベルベースのモデルに従います。

割り込みハンドラは、割り込みハンドラの終了時に割り込みを再度有効にした後に割り込みを取り戻さないようにするために、ミップビットが設定される原因となっているハードウェア条件をクリアする必要があります。

追加の非標準ローカル割り込みソースは、それらをミップ/ミーレジスタの上位ビットに追加することによって、マシンモードから見えるようにすることができ、対応する追加の原因値がmcauseレジスタに返されます。

これらの追加の非標準ローカル割り込みは、midelegレジスタの対応するビットを使用して、低い特権レベルからも見えるようにすることができます。

外部割り込み、タイマー割り込み、ソフトウェア割り込みに対する非標準のローカル割り込みソースの優先順位はプラットフォームによって異なります。

7.2.2グローバル割り込み要因

グローバル割り込みソースは、PLICによって優先順位が付けられ配布されるものです。

プラットフォーム固有のPLIC実装に応じて、任意のグローバル割り込みソースを任意のハートコンテキストにルーティングできます。

グローバル割り込みソースは、レベルトリガ、エッジトリガ、メッセージシグナリングなど、さまざまな形式を取ります。

いくつかのソースは多数の割り込み要求を待ち行列に入れるかもしれません。

すべてのグローバル割り込みソースは、PLIC用の共通の割り込み要求形式に変換されます。

7.3割り込みターゲットとハートコンテキスト

割り込みターゲットは通常ハートコンテキストで、ハートコンテキストは特定のハートに対する特定の特権モードです（ただし、DMAエンジンなど、他の割り込みターゲットも考えられます）。

特にプロセッサコアが外部割り込みを低特権モードに委譲することをサポートしていない場合は、すべてのハートコンテキストが割り込みターゲットになる必要はありません。その場合、特権の低いハートのコンテキストは割り込みの対象にはなりません。

PLICによって生成された割り込み通知は、M / S / Uモード用のmip / sip / uipレジスタのmeip / seip / ueipビットにそれぞれ表示されます。

外部割り込みが低特権モードに委譲されている場合、通知は低特権xipレジスタにのみ表示されます。

各プロセッサコアは、コア上の複数のハートコンテキストによって同時にアクティブな割り込みが発生する方法に関するポリシーを定義する必要があります。

サポートされている特権モードごとに1つずつ、ハートコンテキストの単一スタックの単純な場合では、より高い特権のコンテキストに対する割り込みは、より低い特権のコンテキストに対する割り込みハンドラの実行を横取り(差し替え)することができます。

マルチスレッドプロセッサコアは、異なるハートコンテキストで複数の独立した割り込みハンドラを同時に実行することができます。

プロセッサコアは、割り込み処理の待ち時間を減らすために割り込み処理にのみ使用されるハートコンテキストを提供することもでき、そしてこれらは同じコア上の他のハートの割り込みハンドラを横取り(差し替え)するかもしれません。

PLICは各割り込みターゲットを個別に扱い、複数の割り込みターゲットを含むコンポーネントが使用する割り込み優先順位付け方式は考慮しません。

その結果、PLICは割り込みのプリエンプションやネストの概念を提供していないため、複数の割り込みターゲットコンテキストをホストしているコアでこれを処理する必要があります。

7.4割り込みゲートウェイ

割り込みゲートウェイは、グローバル割り込み信号を共通の割り込み要求フォーマットに変換し、PLICコアへの割り込み要求の流れを制御します。(制御する役割を担います)

割り込みソースごとに最大1つの割り込み要求をPLICコアでいつでも保留できます。これは、ソースのIPビットを設定することによって示されます。

ゲートウェイは、同じソースからの以前の割り込み要求を処理している割り込みハンドラが完了したという通知を受信した後でのみ、新しい割り込み要求をPLICコアに転送します。

グローバル割り込みソースがレベル依存割り込みを使用している場合、ゲートウェイは割り込みレベルの最初のアサーションを割り込み要求に変換しますが、しかしその後、ゲートウェイは割り込み完了メッセージを受信するまで追加の割り込み要求を転送しません。

割り込み完了メッセージを受信したときに、割り込みがレベルトリガされ、それでもまだ割り込みがアサートされている場合は、新しい割り込み要求がPLICコアに転送されます。

ゲートウェイには、いったんPLICコアに転送された割り込み要求を取り消す(撤回する)機能はありません。

PLICコアが要求を受け付けた後、割り込みが処理される前に、レベル依存の割り込みソースが割り込みをディアサートすると、

割り込み要求はPLICコアのIPビットに存在したままで、ハンドラによって処理されます。それは(ハンドラは)、割り込み装置がもはやサービスを必要としなくなったと判断する必要があります。

グローバル割り込みソースがエッジトリガされた場合、ゲートウェイは最初に一致した信号エッジを割り込み要求に変換します。

デバイスと割り込みハンドラの設計によっては、割り込み要求を送信してからそのハンドラの完了通知を受け取るまでの間に、ゲートウェイは、追加の一致エッジを無視するか、保留中の割り込みのカウンタを増分します(インクリメントする可能性があります)。

どちらの場合でも、次の割り込み要求は、前の完了メッセージが受信されるまでPLICコアに転送されません。

ゲートウェイに保留中の割り込みカウンタがある場合、割り込み要求がPLICコアによって受け付けられると、カウンタは減少します。

専用線割り込み信号とは異なり、メッセージ信号割り込み（MSIs）は、どの割り込みがアサートされているかを記述するメッセージパケットを介してシステム相互接続を介して送信されます。

メッセージは、割り込みゲートウェイを選択するためにデコードされ、関連するゲートウェイはエッジトリガー割り込みと同様にMSIを処理します。

7.5割り込み識別子（ID）

グローバル割り込みソースには、値1から始まる小さな符号なし整数の識別子が割り当てられます。

0の割り込みIDは「割り込みなし」を意味するために予約されています。

2つかそれ以上の割り込み要因に同じ優先順位が割り当てられている場合は、割り込み識別子(ID)を使用して関係を解除することもできます。

小さい値の割り込みIDは、大きい値の割り込みIDよりも優先されます。

7.6割り込み優先順位

割り込み優先順位は、小さな符号なし整数で、プラットフォームごとにサポートされている最大レベル数がサポートされています。

優先順位の値0は「割り込まない」ことを意味するために予約されており、割り込みの優先順位は整数値の増加とともに増加します。

各グローバル割り込みソースには、プラットフォーム固有のメモリマップドレジスタに保持されている関連割り込み優先順位があります。

異なる割り込み要因が同じ優先順位値のセットをサポートする必要はありません。

有効な実装は、すべての入力優先順位レベルを配線(ハードワイヤ)することができます。

割り込みソース優先順位レジスタは、ソフトウェアが各優先順位指定内の読み書きビットの数と位置を決定できるようにするためのWARLフィールドであるべきです。

サポートされている優先順位値の検出を簡単にするために、各優先順位レジスタは、レジスタ内で可変であるビット内の値の任意の組み合わせをサポートしている必要があります。

すなわち、レジスタ内に２つの可変ビットがある場合、それらのビット内の４つの値の組み合わせすべてが有効な優先順位レベルとして動作しなければなりません。

-----------------------------------------------------------------------------------------------------------------------------------

縮退の場合、すべての優先順位を値1に固定することができます。この場合、入力優先順位は割り込みIDによって効果的に決定されます。

サポートされている優先順位の値は、次のように決定できます。

1）優先順位レジスタにすべて0を書き込み、次に2）値を読み戻します。

設定したビットは1に固定されています。

次に、3）すべて1をレジスタに書き込み、4）値を読み戻します。

クリアビットはすべて0に固定されています。

ステップ2で配線されていないことが判明した設定ビットはすべて可変です。

サポートされる優先順位レベルは、優先順位フィールド内の可変ビットに1と0のすべての組み合わせを代入することによって取得された値のセットです。

7.7割り込みイネーブル

各ターゲットには、割り込みソースごとに1つの割り込みイネーブル（IE）ビットのベクトルがあります。

ターゲットは無効化されているソースからの割り込みを受信しません。

単一のターゲットに対するＩＥビットは、ターゲットに対するＩＥビットの迅速なコンテキスト切り替えをサポートするために、プラットフォーム特有のメモリマップド制御レジスタ内のビットベクタとしてまとめる必要があります。(られるべきである。)

IEビットはWARLフィールドで、0または1に固定されています。(固定できます)

-----------------------------------------------------------------------------------------------------------------------------------

一部の割り込みソースがターゲットのサブセットにしかルーティングできない場合は、多数の潜在的なIEビットがゼロに固定される可能性があります。

固定割り込みルーティングを使用する組み込みデバイスでは、より多数のビットを1に配線することができます。

グローバル割り込みソースが常に有効になっていても、割り込み優先度、しきい値、および内部割り込みマスキングは、外部割り込みを無視する上でかなりの柔軟性を提供します。

7.8割り込み優先度しきい値

各割り込みターゲットには、プラットフォーム固有のメモリマップドレジスタに保持されている優先順位しきい値が関連付けられています。

しきい値よりも厳密に高い優先順位を持つアクティブな割り込みのみが、割り込み通知をターゲットに送信します。

異なる割り込みターゲットが同じ優先順位しきい値のセットをサポートする必要はありません。

ソフトウェアがサポートされているしきい値を決定できるように、割り込みターゲットしきい値レジスタはWARLフィールドである必要があります。

しきい値レジスタは常に値0を保持できる必要があります。この場合、割り込みはマスクされません。

実装されている場合、しきい地レジスタは通常最大優先レベルを保持することもできます。その場合、すべての割り込みはマスクされます。

-----------------------------------------------------------------------------------------------------------------------------------

単純で有効な実装は、しきい値をゼロに配線することです。この場合、それは効果がなく、個々の有効ビットを保存して復元して同じ効果を得る必要があります。

しきい値の機能は割り込みイネーブルビットを変更することによって達成できますが、単一のしきい値を操作することで、ターゲットは各割り込みソースの個々の優先順位レベルを考慮する必要がなくなり、すべての割り込み有効を保存および復元できます

しきい値を素早く変更することは、電源状態間を頻繁に移動するシステムにとって特に重要です。

7.9割り込み通知

各割り込みターゲットには、対応するターゲットにサービスを待機している保留中の割り込みがあることを示す外部割り込み保留（EIP）ビットがPLICコアにあります。

EIPの値は、割り込みソース、割り込みターゲット、またはPLIC内のレジスタ値を操作する他のエージェントによってもたらされる、PLICコア内の状態の変化の結果として変化する可能性があります。

EIPの値は、割り込み通知として宛先ターゲットに伝達されます。

ターゲットがRISC-Vハートコンテキストの場合、割り込み通知はハートコンテキストの特権レベルに応じてmeip / seip / ueipビットに到着します。

-----------------------------------------------------------------------------------------------------------------------------------

単純なシステムでは、割り込み通知はハートを実装するプロセッサに接続された単純なワイヤになります。

より複雑なプラットフォームでは、通知はシステムインターコネクトを介してメッセージとしてルーティングされる可能性があります。

PLICハードウェアは割り込みのマルチキャストのみをサポートしているため、有効になっているすべてのターゲットが特定のアクティブな割り込みに関する割り込み通知を受信します。

-----------------------------------------------------------------------------------------------------------------------------------

最速のレスポンダが割り込みを要求するので、マルチキャスティングは迅速な応答を提供しますが、ただし、割り込み率が高いシナリオでは、複数のハートが割り込みをトラップして1人しか正常に要求できない場合は無駄になる可能性があります。

ソフトウェアは、各割り込みハンドラの一部としてPLIC IEビットを変調して、割り込みアフィニティ(密接な関係)やラウンドロビンユニキャストなどの代替ポリシーを提供できます。

プラットフォームアーキテクチャおよび割り込み通知の転送に使用される方法によっては、これらがターゲットで受信されるまでに時間がかかる場合があります。

PLICは、PLICコアで介入アクティビティ(する活動)がない限り、最終的にEIPのすべての状態変更をすべてのターゲットに配信することが保証されています。

-----------------------------------------------------------------------------------------------------------------------------------

割り込み通知内の値は、過去のある時点で有効だったEIP値を保持することだけが保証されています。

特に、第２のターゲットは、第１のターゲットへの通知がまだ飛んでいる間に応答して割り込みを要求することができ、その結果、第１(最初)のターゲットがその割り込みを要求しようとするとＰＬＩＣコア内にアクティブな割り込みがないことがわかります。(検出します)

7.10割り込み要求

ターゲットが割り込み通知を受け取った後、割り込みを処理することを決定するかもしれません。。

ターゲットはPLICコアに割り込み要求メッセージを送信します。これは通常、べき等でないメモリマップドI / O制御レジスタの読み出しとして実装されます。

要求メッセージを受信すると、PLICコアはターゲットの最も優先度の高い保留中の割り込みのIDを自動的(アトミック)に決定し、対応する送信元のIPビットをクリアします。

PLICコアはターゲットにIDを返します。

請求が処理されたときにターゲットに対する保留中の割り込みがなかった場合、PLICコアはID 0を返します。

優先順位の最も高い保留中の割り込みがターゲットによって要求され、対応するIPビットがクリアされた後、他の優先順位の低い保留中の割り込みがターゲットから見えるようになるため、PLIC EIPビットは要求後にクリアされません。(クリアされない可能性があります。)

割り込みハンドラは、ハンドラを終了する前にローカルのmeip / seip / ueipビットをチェックして、割り込み(中断)されたコンテキストを復元して別の割り込みトラップを取得(実行)することなく他の割り込みをより効率的に処理できるようにします。

EIPが設定されていなくても、ハートが主張を実行することは常に正当です。

具体的には、ハートはしきい値を最大に設定して割り込み通知を無効にし、代わりに定期的な請求要求を使用してアクティブな割り込みをポーリングすることができますが、ただし、ポーリングを実装するためのより簡単な方法は、特権モード ｘ の対応するxieレジスタの外部割り込みイネーブルをクリアすることです。

7.11割り込み完了

ハンドラが割り込みのサービスを完了した後、関連するゲートウェイは、通常、べき等でないメモリマップドI / O制御レジスタへの書き込みとして、割り込み完了メッセージを送信される必要があります。

ゲートウェイは、完了メッセージを受信した後にのみ、追加の割り込みをPLICコアに転送します。

7.12割り込みフロー

図7.2は、PLIC経由で割り込みを処理するときにエージェント間で流れるメッセージを示しています。

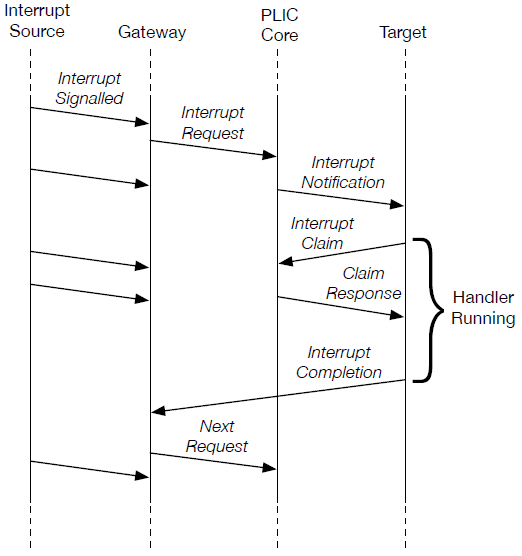


図7.2 PLICによる割り込み処理の流れ

ゲートウェイは、一度に1つの割り込み要求のみをPLICに転送します。割り込みの完了を受信するまで、後続の割り込み要求は転送しません。

PLICは、ゲートウェイからの割り込み要求を受け取るとIPビットを設定し、そしてしばらくしてから割り込み通知をターゲットに転送します。

ターゲットは到着した新しい割り込みに応答するのにしばらく時間がかかるかもしれませんがそれから割り込みIDを得るためにPLICコアに割り込みクレーム要求を送信します。

PLICコアは自動的(アトミック)にIDを返し、対応するIPビットをクリアします。その後、他のターゲットが同じ割り込み要求を要求することはできません。

ハンドラは、割り込みを処理すると、新しい割り込み要求を許可するためにゲートウェイに割り込み完了メッセージを送信します。

7.13 PLICコア仕様

PLICコアの動作は、次のアトミックアクションを使用して、入力メッセージキューと出力メッセージキューを持つ非決定的有限状態マシンとして指定できます。

* レジスタ書き込み：レジスタ書き込み要求を含むメッセージがデキューされます。内部レジスタの1つが書き込まれます。内部レジスタは、優先順位、割り込みイネーブル（IE）、またはしきい値にすることができます。
* 要求を受諾する：割込みソースに対応するＩＰビットがクリアされている場合、ゲートウェイからの割込み要求を含むメッセージがデキューされ、ＩＰビットが設定されます。
* プロセス要求：割り込み要求メッセージがデキューされました。要求応答メッセージは、そのターゲットに対する最も優先順位の高いアクティブな割り込みのIDを持つリクエスタにエンキューされ、この割り込みソースに対応するIPビットはクリアされます。

EIPビットの値は、PLICコア状態の組み合わせ機能として決定されます。

割り込み通知は、EIP値が最終的にターゲットに反映されるようにする自律プロセスを介して送信されます。

割り込みゲートウェイの動作は、PLICコアから切り離されていることに注意してください。

ゲートウェイは、PLICコアの他の操作と同時に割り込み信号の解析と割り込み完了メッセージの処理を処理できます。

-----------------------------------------------------------------------------------------------------------------------------------

図7.1は、PLIC設計のおおまかな概念図です。

PLICコアは、その動作が常にこれらのアトミックアクションの連続的な順序付けに従うと理解できる場合に限り、さまざまな方法で実装できます。

特に、PLICは1クロックサイクルで複数のアクションを処理したり、多数のクロックサイクルにわたって各アクションを処理したりします。

7.14 PLICへのアクセス制御

想定されるユースケースでは、マシンモードのみが割り込みサブシステムを設定するためにソース優先順位、ソースペンディング、およびターゲット割り込みイネーブルにアクセスします。

低特権モードは、ABIまたはSBI呼び出しを介してこれらの機能にアクセスします。

割り込みイネーブルは、ターゲットが現在そのターゲットに対して有効になっている割り込みゲートウェイにのみ完了を通知できる保護メカニズムとして機能します。

マシンモードよりも低い特権で実行される割り込みハンドラは、クレーム読み取りおよび完了書き込みを実行し、それらの目標しきい値を設定することができるだけでよい。

これらのレジスタのメモリマップは、物理メモリ保護または仮想メモリページ保護のいずれかを使用して、マシンモードが互いのアクセスから異なるターゲットを保護できるようにする必要があります。

第8章

マシン構成説明

-----------------------------------------------------------------------------------------------------------------------------------

OSブートのための移植作業を減らすために、プラットフォーム情報をカーネルに伝達するためにデバイスツリーを使うことに戻ったので、この章は時代遅れです。

設定文字列は、他の用途にも設計されていますが、今のところ、標準のデバイスツリーモデルを使用しています。(滞在しています)

RISC-Vプラットフォームには、無数のデバイス、プロセッサコア、および構成パラメータが含まれている場合があります。

ブートローダやオペレーティングシステムを含む、より高度なソフトウェアをサポートするため、ハードウェアプラットフォームは、低レベルのシステムソフトウェア、外部デバッガ、または製造テスト手順で使用するために、プロセッサのリセット後に直接アクセス可能な読み取り専用メモリにコンポーネントの説明を埋め込むことをお勧めします。

この低レベルの埋め込み情報を構成の説明と呼びます。

ここでは、設定情報をエンコードして検索し、設定情報のフォーマットを決定するための標準的なメカニズムを定義します。

8.1構成文字列の検索手順

プラットフォームは、ポインタが存在する固定の物理アドレスを指定するなどして、この文字列を検索するためのポインタの検索方法を記述する必要があります。

製造時に構成オプションをプログラミングすることで、さまざまなプラットフォーム、構成フォーマット、およびチップをサポートするために、プラットフォームで指定された初期ポインターによってシードされた構成情報を見つけるための柔軟な検索手順が定義されます。

プラットフォームによって提供される設定文字列ポインタは、設定文字列の検索が開始される最初のメモリアドレスを指します。

設定文字列は埋め込みバイトで始めることはできません。埋め込みバイトは0x0または0xffのいずれかを含むように定義されますが、無視される最大63の埋め込みバイトを先頭に付けることができます。

64埋め込みバイトに遭遇した場合、検索は設定文字列を見つけずに終了します。

-----------------------------------------------------------------------------------------------------------------------------------

埋め込みバイトは、実装されていないメモリ、バス領域、またはプログラムされていない不揮発性メモリから返される一般的な値を表します。

したがって、構成文字列には、必要に応じて移入またはプログラムされる領域へのポインタを含めることができます。これらの領域は、何も存在しない場合は無視されます。

埋め込みバイトはバイナリデータ構造のアライメントもサポートします。

そうでなければ、最初の非埋め込みバイトは構成情報の始まりです。

たとえば、デバイスツリー文字列形式の設定情報は「/ dts-v1 /」で始まります。

フラット化されたデバイスツリー(Flattened Device Tree)形式の設定情報は、マジックナンバー0xd00dfeedで始まります。

構成文字列形式の構成情報は、 「/ cs-v1 /」で始まります。

-----------------------------------------------------------------------------------------------------------------------------------

設定文字列は（DTS仕様が存在する限り）デバイスツリー文字列と下位互換性がある新しい形式ですが、他のメモリ領域に追加の設定情報を含めることができます。

第9章

歴史

9.1 UC Berkeleyでの研究資金

RISC-Vアーキテクチャと実装の開発は、以下のスポンサーによって部分的に資金提供されています。

* Par Lab：マイクロソフト（Award＃024263）とインテル（Award＃024894）の資金提供と米国カリフォルニア大学のマッチング資金提供による研究支援。ディスカバリー（賞＃DIG07-10227）追加のサポートはPar Lab a liatesのノキア、エヌビディア、オラクル、そしてサムソンから来ました。
* プロジェクトアイシス：DoE賞DE-SC0003624。
* アスパイア Lab：DARPA PERFECTプログラム、HR0011-12-2-0016。DARPA POEMプログラム賞HR0011-11-C-0100。セミコンダクターリサーチコーポレーションによって資金提供されたSTARnetセンター、未来アーキテクチャ研究センター（C-FAR）。アスパイアインダストリアルスポンサー、インテル、およびアスパイア a liates、グーグル、フアウェイ、ノキア、エヌビディア、オラクル、サムソンからの追加サポート。

本稿の内容は必ずしも米国政府の立場や方針を反映するものではなく、公式な支持を推論するものでもありません。

-- 1ページ空き、次ページへ

書誌

[1] ロバート P. ゴールドバーグ。 仮想マシン研究の調査 コンピュータ、７（６）：３４−４５、１９７４年６月。

[2] ジュアン ナヴァロ、シタラム アイヤー、ペーター ドルッシェル、および アラン コックス。 スーパーページに対する実用的で透過的なオペレーティングシステムのサポート。 SIGOPS Oper。 シスト Ｒｅｖ．、３６（ＳＩ）：８９−１０４、２００２年１２月。

-- 最終頁 ここまで。

-- 2018/12/23