RISC-V の拡張を仮想化できるハイパーバイザの開発

- ハイパーバイザによるエミュレーションと Sail による実装の自動生成 -

1. 背景

オープンソースの命令セットアーキテクチャである RISC-V では仕様を「拡張」という単位でモジュール化しており，設計者はこの拡張を組み合わせることで無駄な機能を入れずに仕様を作ることができる．

拡張は盛んに策定されており，2024 年末までに **122 個**策定されている．一方で実際に利用されている数は，実際にハードウェアの仕様書を調べたところ **35 個**にとどまっている．

このような状況は **RISC-V コミュニティ全体にとって大きな損失**である．

2. 目的

本プロジェクトの目的は，拡張をサポートした環境を手軽にハードウェアへ導入できるようにすることである．

これにより，

- **今まで使われてこなかった拡張の活用**を可能にする．
- 仕様策定の際に迅速にソフトウェアの開発者やユーザに環境を提供することでフィードバックを得やすくし，**エコシステムの開発を促進**させる．

という 2 つの狙いを達成することを目指す．

3. 開発の内容

3.1. ハイパーバイザ

RISC-V H 拡張向けの Type-1 ハイパーバイザをフルスクラッチで開発した．また，ハイパーバイザ内で利用するデコーダやアロケータについてもフルスクラッチで開発した．

ハイパーバイザの概要を図 1 に示す．

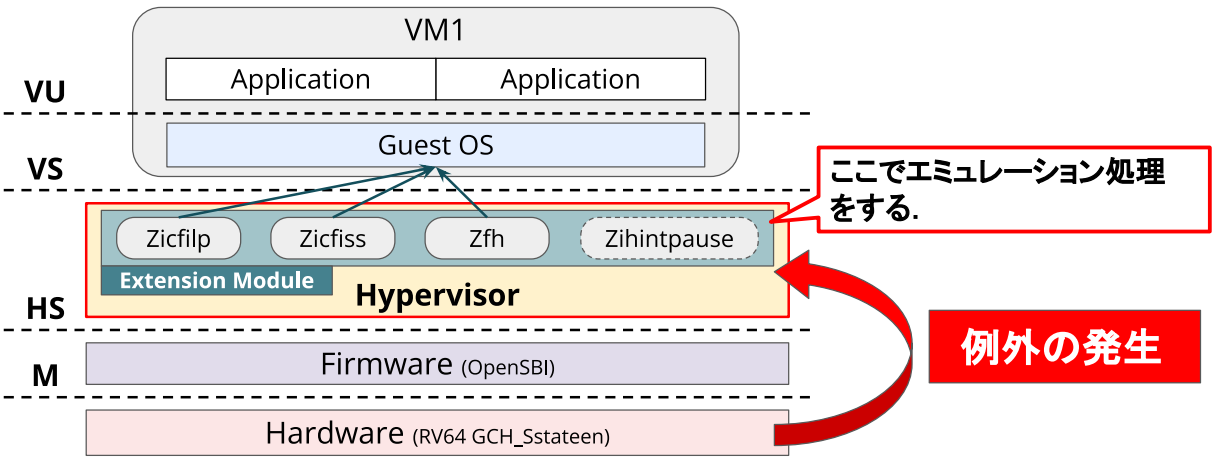

図 1: ハイパーバイザの概要

このハイパーバイザはゲストを 1 つだけ持ち，拡張の振る舞いをエミュレーションするための機能を有する．エミュレーションはハードウェアが出す例外を利用して行う．

例えば，拡張で新しく定義されている命令を実行しようとすると，ハードウェアはそのような命令について知らないので図 2 のように“Illegal Instruciton”という例外を送出する．

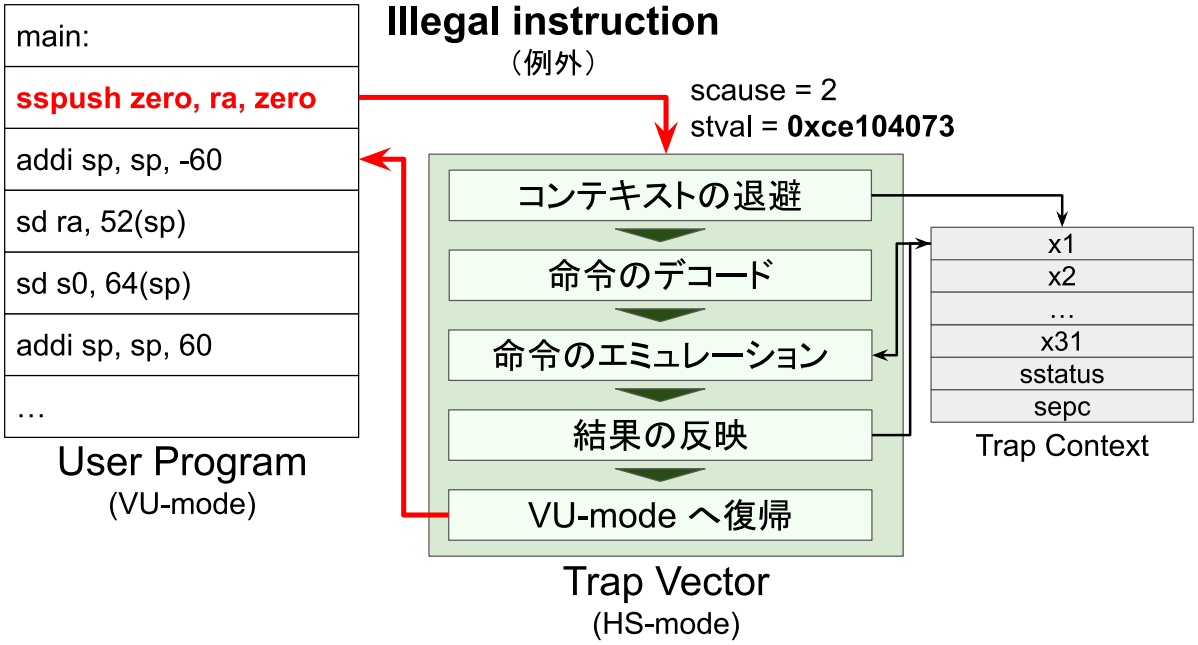

図 2: 例外のトラップによる命令のエミュレーション

このとき，例外処理をハイパーバイザのトラップベクタに移譲するように設定することでハイパーバイザ側で任意の処理を行うことができる．

命令のエミュレーションの場合，デコードによってエミュレーション対象となる命令を解析し，ハイパーバイザ側で演算をしメモリやレジスタの更新をした後，例外を起こした命令の次の命令に復帰することでゲストから見ると正しく命令が実行されたかのように見せかけることができる．

その他，CSR や例外のエミュレーションを行うことで，拡張の振る舞いを再現することに成功した．

エミュレーション処理はハイパーバイザモジュールとして実装されており，拡張と 1 対 1 で対応しているので，ユーザは導入したい拡張に対応したモジュールをインストールするだけで環境を構築できる．

3.2. 自動生成ツール

拡張を新たに追加する際には仕様書に沿ってデコーダやハイパーバイザモジュールを作成する必要があり，これが導入の障壁になるという懸念があった．

また，命令を解釈するデコーダや拡張に記述された仕様をエミュレーションするハイパーバイザモジュールはシステムの根本に関わる部分であるため，**バグや実装漏れ**

は許されない．もし仮にこの部分に脆弱性やバックドアが存在するとハイパーバイザが動作しているよりも上のレイヤに重大な影響を与えてしまう．

実装負荷の低減と実装ミス防止の双方を達成するためにデコーダの実装とハイパーバイザモジュールの雛形を自動で生成するツールを開発した．

自動生成ツールの概要を図 3 に示す．

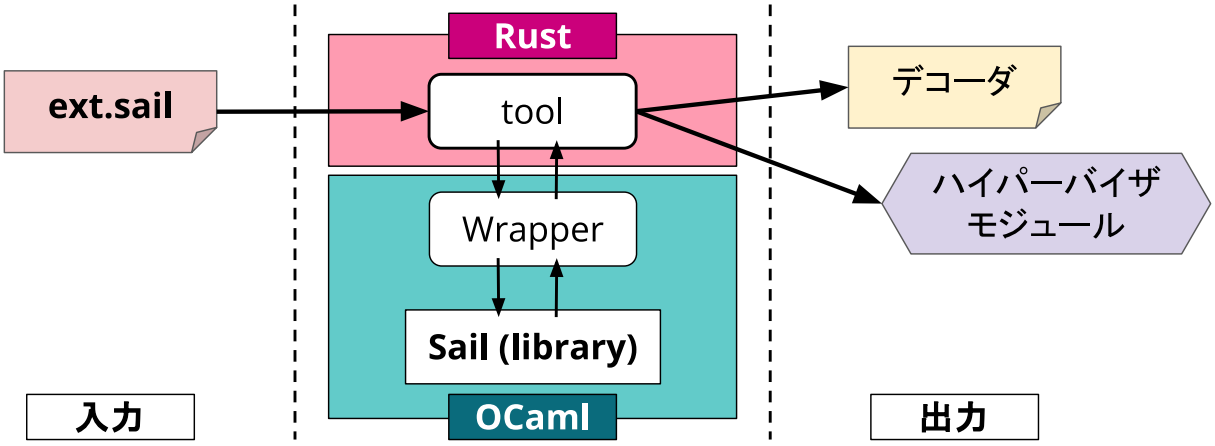

図 3: 自動生成ツールの概要

ツールを作成するにあたり Sail という言語[1]に着目した． Sail は ISA の意味論を記述するための言語で，既に RISC-V の仕様策定でも一部利用されている．

この Sail によって記述された拡張の仕様が書かれたファイルを入力にし，Sail の処理系から AST を取得して，デコーダの実装やハイパーバイザモジュールの雛形を自動で出力するようにした．

デコーダは完全に実装を自動化し，自動生成の正しさをある程度保証するためにデコーダ自身のユニットテストまで自動生成することに成功した．

ハイパーバイザモジュールはグローバル変数の宣言や構造体の定義，trait の実装など全拡張に共通するコードを自動で生成し，それぞれの拡張に依存する処理を todo マクロで指示することで漸進的に実装を進められるようにした．

4. 従来の技術との相違

本プロジェクトは RISC-V の拡張をハイパーバイザのレイヤでエミュレーションする初の試みである．RISC-V H 拡張によるハードウェアの支援を受けることでファームウェアレベルのエミュレーションよりも高速に処理を実現可能である．

また，Sail を用いてデコーダの実装とハイパーバイザモジュールの雛形の自動生成についても取り組み，フルスクラッチ実装の利点を活かして深い連携による高機能な生成を実現した．

[1] https://github.com/rems-project/sail

5. 期待される効果

本システムによって手軽に拡張を使えるようになれば，今まで策定されるだけされて使われてこなかった拡張を有効活用することができる．

手軽に拡張を利用できる環境を作ることで，ユーザはボードが販売される前に拡張を試すことができる．

ソフトウェア開発者は実機の上で検証ができる上，ユーザに使ってもらうことでフィードバックを得ながら開発を進めやすい．

ハードウェア開発者はハードウェアの動作に不可欠なソフトウェアツールチェインが整いやすく，ユーザの動向をみて需要を予測しやすい．

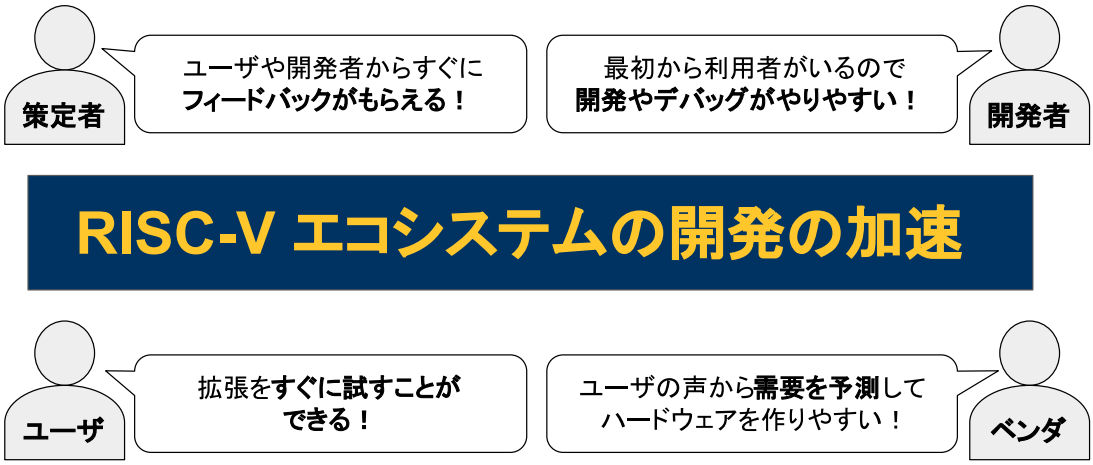

図 4: 本システムによる RISC-V エコシステムの開発の加速

これにより，RISC-V エコシステムの根本的な問題であった**開発サイクルが上手く回らないという問題を解決することができる**（図 4）．本成果は，単に RISC-V の拡張が使えるようになるというだけでなく，エコシステムのあり方に一石を投じるシステムとなっている．

6. 普及の見通し

本プロジェクトの成果は全て MIT ライセンス[2]で公開済みである．オープンソースなので RISC-V との相性が良い．コミュニティに積極的に働きかけることでシステムの周知と普及を進めていきたい．

また，RISC-V H 拡張の実機は今年の 2 月頃からユーザに届き始めており，これに対応するハイパーバイザは少ないことから今が普及のチャンスだと考えている．

本ハイパーバイザが要求する H 拡張や Ssstateen 拡張が RISC-V のプロファイルに昨年入ったので，今後どんどんと対応する実機が出てくるはずであり，更なるシステムの普及が期待できる．より多くの機能や実機に対応することでこの機会を活かしていきたい．

[2] https://opensource.org/license/mit

7. クリエータ名（所属）

- 髙名 典雅（筑波大学 理工情報生命学術院）

（参考）関連 URL

ハイパーバイザ: https://github.com/Alignof/hikami
デコーダ: https://crates.io/crates/raki
アロケータ: https://github.com/Alignof/wild-screen-alloc
自動生成ツール: https://github.com/Alignof/ozora