オープンソース時代の2枚看板を一石二鳥研究

注目のモダン言語Rustで作るRISC-Vシミュレータ

@msyksphinz

今回すること

● 注目RISC-Vのシミュレータを作る

筆者は，コンピュータ・アーキテクチャや命令セット・アーキテクチャ（ISA）について長年研究しており，特に数年前からはRISC-V（リスク-ファイブ）というISAに注目しています．RISC-V ISAは，他のISAに比べて比較的シンプルで，仕様書もシンプルにまとめられており初心者にも分かりやすいです．しかし，そうは言ってもISAの仕様を十分に理解するためには，仕様書を読むだけでは不十分です．真にISAを理解するには自分で手を動かしてみるのが一番，そのためにはいきなりハードウェアを作り込むよりも命令セット・シミュレータ（Instruction Set Simulator；ISS）を作ってみるのが良いと考えています．

● C/C++の代わりになると注目のRustで作るのがピッタリ

筆者はこれまでC/C++でISSを開発してきた経験がありますが，今回はC/C++の代わりの選択肢として注目のRustを使ってISSの開発に挑戦しました．自作したISSでRISC-Vのテスト・パターンが動くまでのステップと，そのときに感じたRustの利点についてまとめてみます．

RustはCやC++と同様に，コンパイル型のプログラミング言語でかつ低レイヤ向けの開発に適しているため，ISSのような高速動作が要求される分野には最適です．また，コンパイル時に安全性がチェックされるため，多くの問題を開発段階で抽出できます．

RISC-Vの命令セット・シミュレータとは

● ソフト作りに欠かせないCPUシミュレータ

命令セット・シミュレータ（ISS）は，CPUの動作をシミュレーションするためのソフトウェアで，主に以下の役割を持っています．

- ハードウェア現物がまだ存在していない段階でソフトウェアの動作を確認する
- ソフトウェアの動作を命令レベルに落とし込むことで，効率的にデバッグする

つまり，現物のCPUがなかなか入手できないときでも，ISSを使ってシミュレーションすることで効率的に解析できます．まだ開発ボードが存在していないCPUでOSを動かすテストなどを行う開発現場では非常に有効で，QEMUなどのエミュレータが使われています．

一方で，（レアなケースだが）CPU開発者にとってはISSの存在は次の2つがより重要になってきます．

- 開発しているCPUが正しく動作しているかどうかを確認するためのゴールデン・モデルとしてISSを使用する
- ハードウェアの開発前に，ISSを用いてCPUの性能を見積もる

CPUの開発は非常に複雑ですので，本当に正しく動作しているのかを確認するためのゴールデン・モデルが必要です．そのために通常はISSが先に開発され，それがハードウェア開発時のゴールデン・モデルとして使用されます．RTLシミュレーション時の実行結果とISSの実行結果とを比較することでハードウェアの正しさを確認します．

ISSのもう1つの利点として，ハードウェアの開発前に開発ターゲットとなるハードウェアの性能を見積もることが挙げられます．これは一般的にサイクル精度を持つ命令セット・シミュレータを使います．

サイクル精度とは，ISSがハードウェアの実行サイクル数まで見積もる機能で，CPUの機能的な正しさだけ

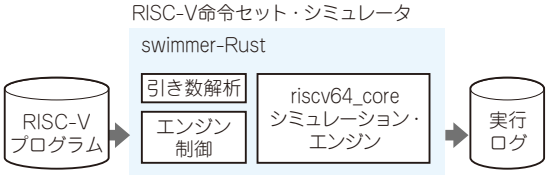

図1　RISC-V 命令セット・シミュレータにコンパイルしたプログラムを入力すると実行して結果を出力する