SecHack365

RVCore

学習駆動コース 今岡ゼミ 松見湧斗

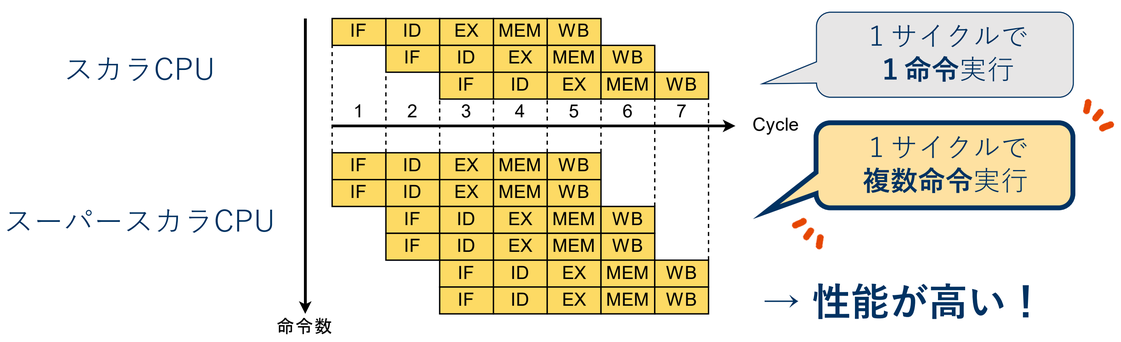
～アウトオブオーダー実行機能を備えたCPUの制作～

リアルなCPUの技術を学びたい！！

・CPUの**動作原理を深く理解**し
・CPUの**設計に関するノウハウ**を手に入れる

アウトオブオーダー
×
スーパースカラ
搭載のCPUを自作して全部達成する！

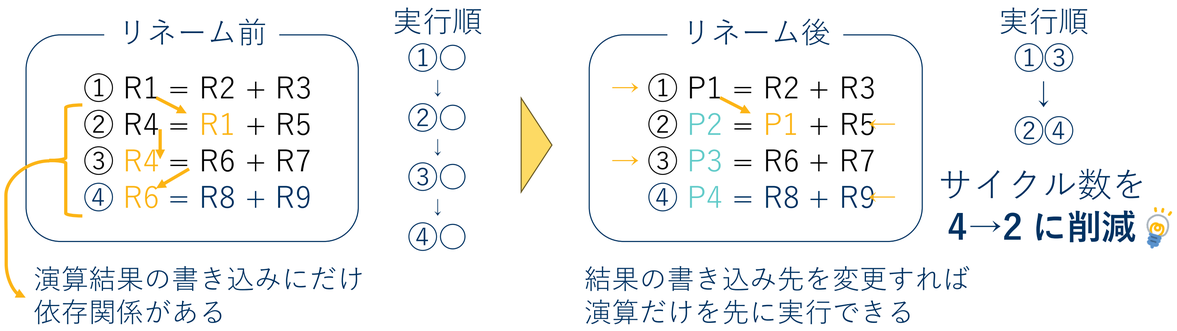
そもそも… スーパースカラって？



→ **性能が高い！**

- 命令レベルの並列性を取り出し**複数の命令を並列実行**する技術
- 同じサイクル数で**より多くの命令を実行可能**に！
- 命令間の依存関係により**同時に実行できない命令が存在**

アウトオブオーダー実行って？



- ハードウェアによって**命令レベル並列性を引き出す**技術
  （スーパースカラの課題：命令間の依存関係を解消）
- オペランド準備状況を監視、準備が完了した命令から先に実行）
- RVCore では**レジスタリネーミング**を用いてアウトオブオーダー実行を実現

RVCore の構成

- ディジタル回路シミュレータ上で実装（FPGAには未実装）
- 32bit RISC-V アーキテクチャ（RV32I）のサブセット
  →メモリアクセス以外の整数演算命令を実装
- ２命令同時デコード・発行
- アウトオブオーダー実行
- Re order Buffer（ROB）/Reservation Station（RS）
  →32エントリ（最大32命令の幅で依存関係を解消）
- 分岐予測・1レベルの投機的実行をサポート

フロントエンド

命令デコードに加えて以下の処理が**フェッチ順で**行われる
- 依存関係の除去（レジスタのリネーム）
- ROB（命令追跡テーブル）への登録

Rename Map Table は最新の結果を持つ物理レジスタ番号が格納され、後続の命令がどこを参照すれば正しいデータが得られるかを示している。すべての書き込み先レジスタは Free List から取得された空き物理レジスタにリネームされる。

依存関係が取り除かれた命令は
Reservation Station にバッファされる

バックエンド

データ依存が取り除かれた命令から順に演算を実行
このとき、先にバッファされた命令を優先して発行する

発行された命令はインオーダーCPU同様に処理される
- 演算結果を物理レジスタに格納
- ROBとRSに命令実行が終了したことを通知
- 通知をもとに次に発行する命令を決定

コミット　演算結果をフェッチ順に並べなおす処理

ROBの先頭命令が実行完了すると以下の動作を行う
- 最新の命令の結果を保持する物理レジスタ番号を Commit Map Table に書き込む
- ROB から命令エントリを削除
- 不要になった物理レジスタを Free List に返却

投機的実行時に分岐予測ミスが発覚した場合、分岐以後の命令をROBから削除し、Commit Map Table の内容を Rename Map Table にコピーすることでロールバックする

先行命令の結果をROBで待機

検証機能　命令実行完了までの流れが複雑なため、バグが埋め込まれやすい
→CPU 検証作業の効率化が重要となる

自動テスト

ソフトウェアで用いられるテストツールを流用したテスト環境を用意。
各モジュールに対してテストコードを記述することで変更時にバグが生じていないことを検証する

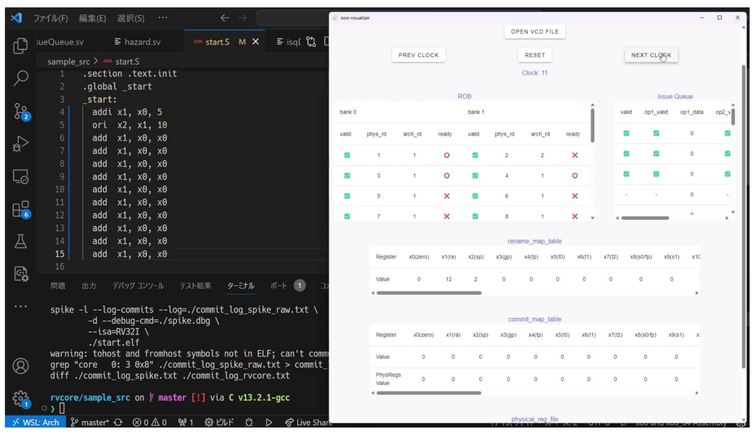
既存のエミュレータとの実行結果比較

命令の順序を入れ替えながら実行する性質上、命令を実行する順番に依存して顕在化するバグが混入する可能性がある。
→一般的なアサーションではどの時点でバグが発生したか調べるのが難しい
Spike を用いて全命令の実行結果を検証することでバグ発生個所の特定を容易化

デバッグ支援ツール

CPUの内部状態（各レジスタ、ROB、RS）を可視化できるツールを作成
従来の波形表示によるデバッグと組み合わせ、以下のフローでバグ修正を行えるようになり、デバッグ効率の改善に寄与した。

1. Spikeとの比較でバグ検出
2. 支援ツールを用いて内部状態の解析
3. 波形表示ツールを用いて原因となっている信号を特定



デバッグ支援ツールを用いると１命令目が実行完了後、２命令目が追い越されて３・４命令目が実行されている様子が確認できる

今後の課題

実行可能な命令種別の拡張
　メモリアクセス命令の実装や割り込みの実装など。OSが動作できる程度の機能になるまでは開発を続けたい。
FPGAへの移植
　現在はシミュレータ上での実行にとどまる。クリティカルパスや回路規模が現実的でない部分が含まれるため、最適化が必要。
性能の向上
　単純にスーパースカラ・アウトオブオーダー実行を実装しただけでは性能が下がってしまった（現在はIPC=0.5～0.6程度）
　フォワーディング等の基礎的な対策や各リソース量の調整が必要
ノウハウの共有
　先駆者が少なく、情報収集に苦労した。
　得られた知見をブログ等で公開したい。

NICT National Institute of Information and Communications Technology