仕様書どおりに動作するか？

Appendix 2　このテスト・パターンをパスしないとRISC-Vとは言えない?!

RISC-V検証用テスト・プログラムによる動作確認

石原 ひでみ Hidemi Ishihara

● テスト・プログラムの必要性

実装したRISC-Vとビルドしたツールチェーンが，ともに正常であることを確認しておく必要があります．

もし，実装したRISC-Vに不具合があったり，ビルドしたツールチェーンが実装していないバイナリ・コードを生成したりすると，いずれの場合も作成したプログラムは正常に動作しません．

また，RISC-V仕様書のバージョンとツールチェーンの対象バージョンによっては，意図しない実行バイナリが生成される可能性もあるので，テスト・プログラムを実行してツールチェーンと実装したRISC-Vモジュールが正常な組み合わせになっていることを確認する必要があります．

Appendix 3の手順に従ってRISC-Vのクロス開発環境の整備が整ったところで，RISC-Vのテスト・プログラム（RISC-Vではテスト・パターンと呼ぶ）を使用して，RISC-Vの実行バイナリを生成してみましょう．

● テスト・パターンの入手

RISC-Vのテスト・パターンは標準構成にも含まれていますが，次のようにgitリポジトリからダウンロードして構築することが可能です．

表1　テスト・パターンの種類とフォルダ

フォルダ	テスト内容		
	ビット数	モード名	テスト命令系
rv32mi	32	マシン	基本命令系
rv32si		スーパバイザ	基本命令系
rv32ui		ユーザ	基本命令系
rv32um			乗除算命令系
rv32ua			アトミック命令系
rv32uc			圧縮命令系
rv32uf			単精度浮動小数点演算命令系
rv64mi	64	マシン	基本命令系
rv64si		スーパバイザ	基本命令系
rv64ui		ユーザ	基本命令系
rv64um			乗除算命令系
rv64ua			アトミック命令系
rv64uc			圧縮命令系
rv64uf			単精度浮動小数点演算命令系
rv64ud			倍精度浮動小数点演算命令系

```
$ git clone git://github.com/riscv/riscv-tests
$ cd riscv-tests
$ git submodule update --init --recursive
```

● テスト・パターンの種類

RISC-Vのテスト・パターンはisaフォルダに格納されており，RV32とRV64のテスト・パターンが含まれています．RISC-Vに実装した命令セットのテスト・パターンを使用してください（**表1**）．今回実装したRISC-Vでは，rv32uiとrv32umの2つを使用します．

● 各テスト・パターンの構成

テスト・パターンは各テスト・パターンであるアセンブラ・ファイル（拡張子が.S）とヘッダ・ファイル（拡張子が.h）で構成されます．rv32uiのadd命令のテスト・パターンの場合，isa/rv32ui/add.Sがテスト・パターンの本体であり（実際はrv64ui/add.Sをインクルードしているだけ），env/p/riscv_test.hをインクルードしています．

riscv_test.hにはテスト・パターンのスタート・コード（_start）などがあり，テスト・パターンは次のシーケンスで実行されます．テスト・パターンの終了は.write_tohostで0x8000_1000（.tohostのアドレス）に結果が書き込まれた時点で終了です．

● テスト・パターンのコンパイル設定

ダウンロードしたテスト・パターンのフォルダで次のようにconfigureを実行します．

```
$ ./configure
```

configureの実行が完了するとMakefileが生成されていますが，生成されたMakefileは64ビット用のテスト・パターンをビルドする設定になっています．今回実装したRISC-Vは32ビットであるため，RV32のテスト・パターンを使用するため，次のようにMakefileを修正します．

```
XLEN    := 64 → 32に変更する
```

● リンカ・ファイルの修正

デフォルトのテスト・パターンは，別のメモリ・マップをターゲットとしているため，今回実装したRISC-Vの環境に適合していません．そこで，テスト・パターンのコンパイル前に，実装したRISC-Vのメモリ・マップに適合させるためにリンカ・ファイルを修