ハードの理解で差がつく時代

シミュレータの自作からはじめるCPU動作メカニズム

第5回 32ビットArm Cortex-Aシミュレータ作りに挑戦

中森 章

前回は，Cortex-M0（Armv6-M）シミュレータを実際に動作させてみました．今回は，Cortex-A7（Armv7-A）のシミュレータ作りに挑戦します．（編集部）

今回作るもの…32ビットArm代表格Cortex-A7シミュレータ

RISC-V，Cortex-M0と，いい感じでソフトウェア命令セット・シミュレータ（ISS：Instruction Set Simulator 以下ISSと記載）が作成できました．その次に手を出してみたいのは，Cortex-A7（Armv7-A）のISSです．

Armv7-Aは32ビットArmのアーキテクチャ（AArch32）を象徴するものであり，今の時点で（32ビットの）Armと言ったらArmv7-Aのことです．Armv7-Aは，Cortex-A7の前世代であるCortex-A8から採用されていますが，Cortex-A8では除算命令をサポートしないなどArmv7-Aとしては完成形ではありません．そこで，ISSのターゲットとしたのがCortex-A7です．

やはり，32ビットArmのアーキテクチャというのは，CPUのアーキテクチャを語る上で無視することはできません．Armアーキテクチャは，一応，RISCに分類されますが，非常に独特です．この連載では，RISC-VとCortex-M0のISSの規模を比較して，RISC-Vアーキテクチャの単純さを示してきましたが，本当に比較する相手は，Cortex-A7ではないかと思っています．

Cortex-A7はラズベリー・パイ2のCPUです．ラズベリー・パイ3で64ビット化（Cortex-A53採用）になりましたが，本誌の読者にはなじみ深いCPUではないでしょうか？

Armv7-Aアーキテクチャ Cortex-A7の特徴

さて，以下にCortex-A7（Armv7-A）のアーキテクチャの特徴を示します．ここでは，Cortex-M0（Armv6-M）との比較を行います．もっとも，Armv7-AではThumb/Thumb-2命令セットをサポートしますから，Cortex-M0にあってCortex-A7にないものは存在しません．Cortex-M0とCortex-A7の違いは，Thumb/Thumb-2命令セットと32ビットArmのネイティブな命令セットの違いです．

①Cortex-A7の特徴

- 32ビット固定命令長
- 汎用レジスタが16本
- PC（プログラム・カウンタ）が，実行中の命令より8番地先を指している
- ほとんど全ての命令が，条件フラグの値に応じて，実行/不実行を指定できる
- ソース・レジスタの片方をシフトできる
- ロード/ストア命令の実行アドレス計算に，ベース・レジスタに対して，インデックス・レジスタをシフトした値を使うことができる
- ロード/ストア命令で，実行後に実行アドレスを次のアクセス用に変更できる
- 連続ロード/ストア命令（LDM/STM）で，プレデクリメント/プレインクリメント/ポスト・デクリメント/ポスト・インクリメントの全てが指定可能

②Cortex-M0の特徴

- 16ビット/32ビット可変長命令
- 汎用レジスタが8本（命令によっては16本使える）
- PC（プログラム・カウンタ）が，実行中の命令より4番地先を指している
- 連続ロード/ストア命令（LDM/STM）では，ポスト・インクリメントしか指定できないが，代わりに，PUSH/POP命令でプレデクリメント/ポスト・デクリメントを指定する

Cortex-A7のISSを作成する上で少々面倒なのは，オペランドのシフト機能です．これは，デコード時にシフト操作（つまり命令実行）を行わなければならないということです．また，個人的にはロード/ストア命令のバリエーションの多さがISSを実装する上で気になりました．

Cortex-A7シミュレータの構成

● ①命令フェッチ部

Cortex-A7は32ビット固定長の命令ですから，命

第1回 CPUの基本動作メカニズム入門（2018年9月号）
第2回 シンプルRISC-Vシミュレータを自作する（2018年10月号）
第3回 RISC-Vシミュレータを動かしてみる（2018年11月号）