先端計算機アーキテクチャレポート
Arm アーキテクチャの構造

坪井正太郎（101830245）

2020 年 11 月 20 日

1　概要

このレポートでは、社会で活用されている計算機アーキテクチャとして、Arm アーキテクチャについて記述する。まず、社会における活用の例について書き、その後 MIPS との違いを中心にアーキテクチャの構造を分析する。

Arm アーキテクチャの中でも、コアの実装、拡張に複数のラインナップがある。ここでは、実装を限定せずに、どのような拡張がアーキテクチャに含まれるのか記述する。基本的に、Armv8 アーキテクチャを前提とした。

2　社会における活用

2.1　ライセンス

Arm アーキテクチャは、他のアーキテクチャとは異なり、ARM によってライセンスを受けることで利用できる。ライセンスを受けたプロセッサベンダは、提供を受けた設計に従ってプロセッサを製造することができる。

ライセンスには、命令セットレベルで互換性があるプロセッサを製造できるものや、RTL 記述が提供されるもの、ネットリストからの配線をしてあるものまである。命令レベルのライセンスは、用途に最適化できる部分が大きいが、設計の負担や製造までの時間がかかる。一方でハードに近い設計の提供を受けると、そのような問題が少ない。このように、ライセンスの選択によってベンダの都合に合った設計手法を選択することができる。

2.2　設計方針

Arm アーキテクチャは、ハードウェアを小さくすることで消費電力を削減している。そのため、携帯機器や組み込み向けのアーキテクチャとして採用されることが多い。具体的には、キャッシュの容量や、パイプラインの段数を削減することでハードウェア要素を削り、省電化している。

一方で、高度なアプリケーションを実行するための拡張を一部では導入し、コード密度や処理性能を向上させている。例として、SIMD 命令、64bit 命令のサポートの他に、Java のバイトコードを直接実行する機能を

サポートしている。

この 2 つを両立させるための設計として、少電力のコアと、高性能のコアを組み合わせる実装が設計され、big.LITTLE という名前で実現されている。

3 アーキテクチャの特徴、構造

3.1 cpsr による条件フラグ

cpsr(プログラムステータスレジスタ) という条件フラグが格納されたレジスタを参照することで、1 命令単位で命令の実行、非実行を制御することができる。たとえば、EQ,NE 命令や演算によるオーバーフロー、演算結果の符号によって cpsr の値を更新できる。ARM 命令は、全ての命令に条件フィールドがあるので、全ての命令実行をプログラムステートによって制御できる。

さらに cpsr は、プロセッサの実行状態や、エンディアン形式などのフラグを持っている。

少数命令のみの分岐や、後続の処理は必ず実行する if-then 構造のように、コード密度や、分岐予測の失敗時にパイプラインに与える影響がある記述を削減することができる。

3.2 Thumb/Thumb-2 命令

32bit の ARM 命令セットから、さらにコード密度を向上させ、省電力化するために、16bit の Thumb 命令が定義され、ARM 命令セットと相互に呼び出し可能としている。これは、頻繁に実行される手続きを ARM 命令で記述し、あまり頻繁に実行されない手続きでは全体のコード密度を向上させるために Thumb 命令で記述する、といった利用がされる。実際に、コードベースで少ない部分が、実行時間のほとんどを占めるので、この削減には効果がある。

また、Thumb の拡張で、Thumb-2 が設計された。これは、ARM ←→ Thumb が別のモードで実行することによるオーバーヘッドを、32bit の Thumb-2 命令を実行することで、削減される。

3.3 Jazelle

Arm は、3 つ目の命令セットとして、Java のバイトコードをプロセッサ上の専用デコーダで解して、直接ほとんどの命令を実行することができる。cspr のフラグにモードの選択ビットがあるので、切替命令によって ARM 命令→ Java バイトコードに命令セットを変更することができる。

3.4 DSP

一部の Arm プロセッサには、SIMD 演算を行うことができる演算器が搭載されている。大量のデータに対して、一度に同じ演算を高速に行うことができる。

命令セットとして、NEON 命令セットが用意されている。NEON 用のレジスタ幅に対して、処理するデータを決めることで、一度に対象になるデータ数が決まる。例えば、64bit のレジスタ 2 本を 16bit データの加算オペランドにとると、1 度に 4 つのデータに対して加算処理をして、64bit ディスティネーションレジスタにできる。

NEON によるベクタ処理は、マルチメディア処理の他に、マシンラーニングの目的でも広範に利用される。特に、高い精度を要求されない処理に対応するために、高速、省データの半精度浮動小数点演算がサポートさ

れた。

今回調査した中では、Arm の DSP を利用してマシンラーニングを行っている例を見つけることができなかった。しかし、Qualcom の SnapDragon シリーズには、Qualcom が内製した Hexagon という DSP が搭載されているなど、各ベンダによって DSP が追加されている。

Hexagon が、NEON 命令セットをサポートしているかについては、今回の調査で資料を見つけることができなかった。DSP を使った開発には Hexagon DSP SDK が、SD 上のマシンラーニング向け開発は、SNPE(Snapdragon Neural Processing Engine) を使用して開発される。SNPE では、コンパイルは Arm をターゲットとしていて、ソース内でアクセラレータを指定するようになっていたので、おそらく NEON 命令を実行する DSP として、Hexagon が搭載されていると予想できる。

参考文献

[1] David A.Patterson/John L.Hennessy コンピュータの構成と設計 (上), 付録 E

[2] Hexagon DSP SDK, `https://developer.qualcomm.com/software/hexagon-dsp-sdk`