第4章 方法ごとのロード時間差をプログラムで調査

Armプロセッサの即値ロード命令の使い方

永井 健一

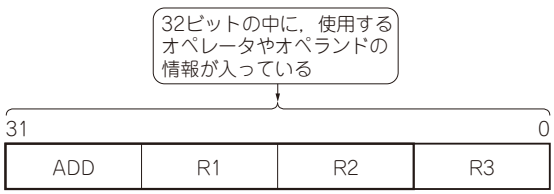

図1 Armアセンブラの**ADD**命令の構造

Armプロセッサなどで行う処理をアセンブリ言語で書くことはあまり多くないでしょう．しかし，アセンブリ言語で書かないとCPUが持っている一部の機能は使えません．それらはOSやドライバのソフトウェアのように低レイヤで処理するには必要です．

現在のArmの命令セットには，Thumb，Thumb-2，A32，A64と複数あります．本稿では，A32とA64について取り上げます．

コンパクトなA32命令セット

ラスベリー・パイなどで32ビット・モードと言われていたもので，従来よく使われていた命令セットです．A32の命令長は32ビット固定で，CPU上のレジスタも32ビットです．

例えばオペランドの値を足すADD命令は**図1**のような構造になっています．ADD命令には複数の記述方法があるのですが基本形は，

```
ADD {Rd}, <Rn>, <Rm>
```

と書きます．Rd，Rn，RmはCPUの持つレジスタを指定する変数です．この命令は，{Rd}＝<Rn>+<Rm>という計算を実行します．

リスト1 32ビットの即値を16ビットずつロードする

```
MOVW    Rd, #imm16      ; 下位16ビットをセット
MOVT    Rd, #imm16      ; 上位16ビットをセット

MOVW    Rd, #0x5678
MOVT    Rd, #0x1234
```

①

②

図2に，32ビット命令へのエンコーディングを示します．ADD命令では，単純に加算を行うだけでなく条件付きで加算したり，シフトした値を加算したりもできます．

レジスタに32ビットのイミディエイト値（即値）をロードする場合を考えます．命令長が32ビットなので，普通に考えると32ビットの命令の中に即値が入りきらず，1回の命令実行ではロードできません．

● 任意の値をロードする方法

▶方法1：16ビットずつロード

即値を，上位16ビットと下位16ビットに分けてロードする方法があります．MOVW/MOVTという命令を使います．これをアセンブリ言語の書式で示すと**リスト1**の①のようになります．0x12345678という即値をロードする例を**リスト1**の②に示します．

▶方法2：リテラル・プールを使ってロード

いったんメモリ領域に即値を置いて，そのアドレスのメモリから値を読み出す方法としてLDR命令が用意されています（**リスト2**の①）．

GNUアセンブラではこれを**リスト2**の②のように書くこともできます．自動的にリテラル値を配置して，PC（プログラム・カウンタ）相対でのロード命令

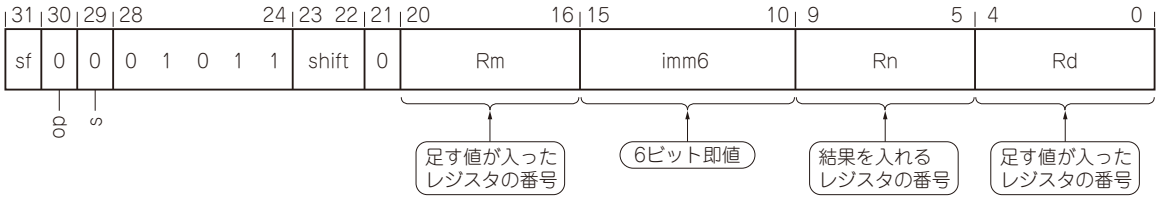

図2 **ADD**命令の各ビットが持つ意味