第4章 メモリ・アクセスのアドレッシング・モードから四則演算の原理まで

どんなCPUでも理解できるアセンブリ言語の基礎知識

大貫 徹

究極的には，CPUは2進数である機械語しか理解できない．しかし2進数だけでプログラムを記述するのは困難である．そのため，人間にも読みやすい英語で命令を表記したアセンブリ言語が開発された．アセンブリ言語を使えば，LDやADDなどの表記でプログラムを記述できる．ここではアセンブリ言語の基礎知識について，特定のCPUにとらわれない概念を解説する．（編集部）

最近は8ビットCPUといえども，C言語でプログラムを記述するのが当たり前になってきています．ですがそれでは，実際にCPUの中でどのような処理が行われているかわかりにくいのも事実です．だからといって世の中に沢山あるCPUの命令を一から勉強するのは大変です．

この章で最終的に読者の皆さんに理解してもらいたいのは，メモリ上に表現されたデータの構造であったり，数値や文字，そのほかのさまざまな対象物の表現方法や処理方法です．それぞれのCPU固有の命令セットではありません．

CPUが違っても，メモリへのアクセス方法や演算の考え方は共通であり，そこが理解できていればCPUが違っても必ず応用が利くはずです．

1．機械語とアセンブリ言語

● 機械語とは何か

CPUはプログラムを読み込み，その命令に従って動作します．そのプログラムとはどのようなものでしょうか．

ディジタル回路はすべて0/1の世界です．もちろんプログラムも0と1で示されます．具体的な例を示しましょう．

```
～略～
1000000000100010
1100011001000111
～略～
```

このように，CPUが直接理解できる命令（言葉）のことを機械語と呼びます．2進数では目がチカチカしてきますね．では16進数で表現してみましょうか．

```
～略～
8022
C467
～略～
```

人間から見ると，これでもさっぱりわかりません．

実はこのプログラムは，第5章で解説されているCPU用の機械語です．機械語だけでは何が何やらわからないので，もう少し人間にわかりやすい表現をしてみましょう．

```
～略～
8022h → AND, D1:D2
C467h → JZE, D3:D7
～略～
```

矢印の右側に示したのが「ニモニック」と呼ばれるものです．ニモニックは機械語を英単語や記号の組み合わせに置き換えたもので，機械語とニモニックは1対1で対応します．

8022hという機械語は，レジスタR1とレジスタR2の値の論理積（AND）を計算してレジスタR2に代入するという命令です．C467hは，ゼロ・フラグがセットされていれば，レジスタR3の値をプログラム・カウンタ（レジスタR7）に代入，つまりレジスタR3の値のアドレスにジャンプする命令です．この2行は，レジスタ同士で計算を行い，その結果によってプログラムの流れを切り替える処理となります．

● CPUが異なれば機械語も異なる

最近の携帯電話は高機能で，機能を拡張するためのプログラムをダウンロードして実行できるものもあります．

しかしこの携帯電話用のプログラムは，パソコンでは動

KEYWORD ── 機械語，アセンブリ言語，アドレス・モード，2進数，10進数，加減乗除演算，分岐命令，スタック・ポインタ

作しません．それは携帯電話とパソコンとでは，CPUの種類が異なるからです．CPUの種類が異なると，機械語にも互換性はありません．

機械語が異なるわけですから，それに対応するニモニックもCPUごとに異なります．一見似たような表記だとしても，細部の記述ルールが異なることがほとんどです．

CPUの種類はそれこそ星の数ほどある（というのは言いすぎか？）ので，そのすべての機械語について説明はできません．しかし，ニモニックの記述が異なるにせよ，基本的な命令の動作や考え方は共通のものがあります．

● アセンブリ言語とは

機械語が1命令あるだけではプログラムとしてほとんど意味はありません．何十行，何百行とニモニックを並べて，何らかの意味のある動作をするプログラムを記述します．これを「アセンブリ言語プログラム」と呼びます．

一般的なアセンブリ言語は次のような形式となっています．

- 1行に1命令ずつ記述する
- 1行の中を次のように区切って使っている
  ＜ラベル＞＜命令＞＜オペランド＞＜コメント＞

```
SERCHXX:   LD   A,VAR_NAME  ；検索名称の最初の文字
```

ラベルはこの行にある命令の位置を名前で覚えやすくするために使います．何かを探す処理部の場合は，「SERCHXXX」というように自分でわかりやすい名前を考えます（英数字で記述するのが慣例）．一般的にラベルの後にはコロン「:」を置きます．

命令にはニモニックを記述します．例えば，メモリからレジスタへデータを読み込む＝ロードする場合は「LD」（Loadの略）などと記述します．ニモニックはCPUごとに違う表現が指定されているので，自分で決めることはできません．

オペランドは，主に場所と操作の細部を指定するのに使います．例えばメモリのVAR_NAMEと名付けた場所からAレジスタにデータをロードする場合は「A,VAR_NAME」などと書きます．この書き方もCPUごとに違い，レジスタや構造の違いが反映されます．

コメントはこの行が何を目的にしているとか，後からプログラムを見る人（自分のためでもある）にわかりやすく説明を書いておくのに利用します．コメントは多くの処理系で日本語が使えます．

2. 命令の種類と動作

アセンブリ言語でプログラムを書くには，そのCPUが持っている命令を覚えなくてはなりません．しかしこの世の中には非常に多くの種類のCPUがあります．そしてそれらは皆異なる命令をもっています．つまり，お互いに互換性がありません．とてもすべてのCPUの命令を覚えられるものではありません．

しかしどんなCPUでも，命令を種類別に分類していくと共通するものがあり，それらを知っているだけで，さまざまなCPUを扱えるようになります．

● 命令を大まかに分類

命令数の多少にかかわらず，ほとんどのCPUは個々の命令を次のように分類できます．このような分類で覚えると，命令セットの表を見ながらでもアセンブリ言語でプログラムが作れるようになります．

- データ移動命令
- 論理演算命令
- 算術演算命令
- 分岐命令
- そのほか（制御命令）

プログラムはこれら5種類の組み合わせで表現可能なので，あるCPUでアセンブリ言語の経験があれば，CPUが異なってもゼロから覚え直す必要はなく，それまでの経験が役に立ちます．

● データ移動命令

CPU内部のレジスタやメモリ間でデータの移動を行います．後述するようにさまざまなアドレッシング・モードがあります．

CPUによっては特殊なメモリ空間を用意しているものがあり，それらのアドレッシング・モードを使うと処理サイクルが短縮できたりします．また一部のCPUは入出力空間を別なアドレス空間として捉えており，入出力命令が存在します．これも基本的にはデータの移動命令のグループと考えて差し支えありません．

ニモニックとしては，LD/ST（ロード/ストア），MOV（ムーブ：移動），Tx（トランスファ：転送）といった書き方をするものが代表的です．