第2章

高級言語によるプログラミングの流れを理解する

C言語プログラムを開発する手順を理解しよう

ここではC言語によるプログラムを始める前に，機械語とC言語との関係について解説する．そしてC言語によるプログラム開発の手順や「関数」の考え方など，C言語プログラミングに必要な予備知識について解説する．（編集部）

大中 邦彦

1．C言語は高級言語，機械語は低級言語？

● 電卓のプログラムをより詳しく理解する

第1章では，CPUが動く基本的なしくみを説明しました．第1章で示した電卓のプログラムは少し複雑でしたね．読むだけであれば1行ずつ丁寧に追っていけば理解できますが，あの書き方でプログラムを書くとしたらちょっと頭が痛くなってくるのではないかと思います．

リスト1は第1章の電卓のプログラムを実際に存在するARMという名前のCPU用に書き直してみたものです．ARMプロセッサにはI/Oポート入出力命令がなく，第1章で説明したメモリ・マップドI/O方式を使って周辺装置とデータを受け渡します．今回は「電卓のボタン」が「メモリの0x40000000番地」に割り当てられており，「7セグメントLED」が「メモリの0x40000004番地」にあるものとしてプログラムを作成しています．

リスト1の右側にある数値の一覧は，このCPUが直接理解できる機械語です．このような数値を見ても人間にはなんだかさっぱりわからないため，もう少しわかりやすく記述したのが真ん中の列です．

真ん中の列は「アセンブリ言語」と呼ばれる言語で書かれています．アセンブリ言語はCPUが直接解釈できる機械語を，人間が読みやすい英単語の形で表現したものです．右側の列のような機械語の数値を見ても人間にはわかりにくいため，機械語は通常アセンブリ言語で記述します．

● アセンブリ言語を読んでみる

このアセンブリ言語で書かれた機械語を，少し詳しく見てみましょう．まず，日本語の文章で書かれた「レジスタA」，「レジスタB」，「レジスタC」は，それぞれアセンブリ言語の中のR1，R2，R3に相当します．最初の2行にあるR4とR5もレジスタで，これらは電卓のボタンと7セグメントLEDにアクセスするための番地の記憶用として使用しています．例えば，

```
LDR     R4, Button_Addr
```

という命令は，リストの最後の方に書かれている，

```
Button_Addr: DCD 0x40000000
```

という値をR4というレジスタに読み込む（ロードする）という命令です．これで，R4に0x40000000が入ります．同じように，R5には0x40000004が入ります．

続くMOV命令は，レジスタに定数を覚えさせるための命令です．

```
MOV     R1,#0
```

は，R1に0を入れる命令です．LDR命令もレジスタに値を入れる命令でしたが，LDR命令はプログラムの別の場所に書かれた値を読み込むのに対し，MOV命令は#0という値を直接代入します．

次のSTR命令はレジスタの内容をメモリに書く命令です．例えば，

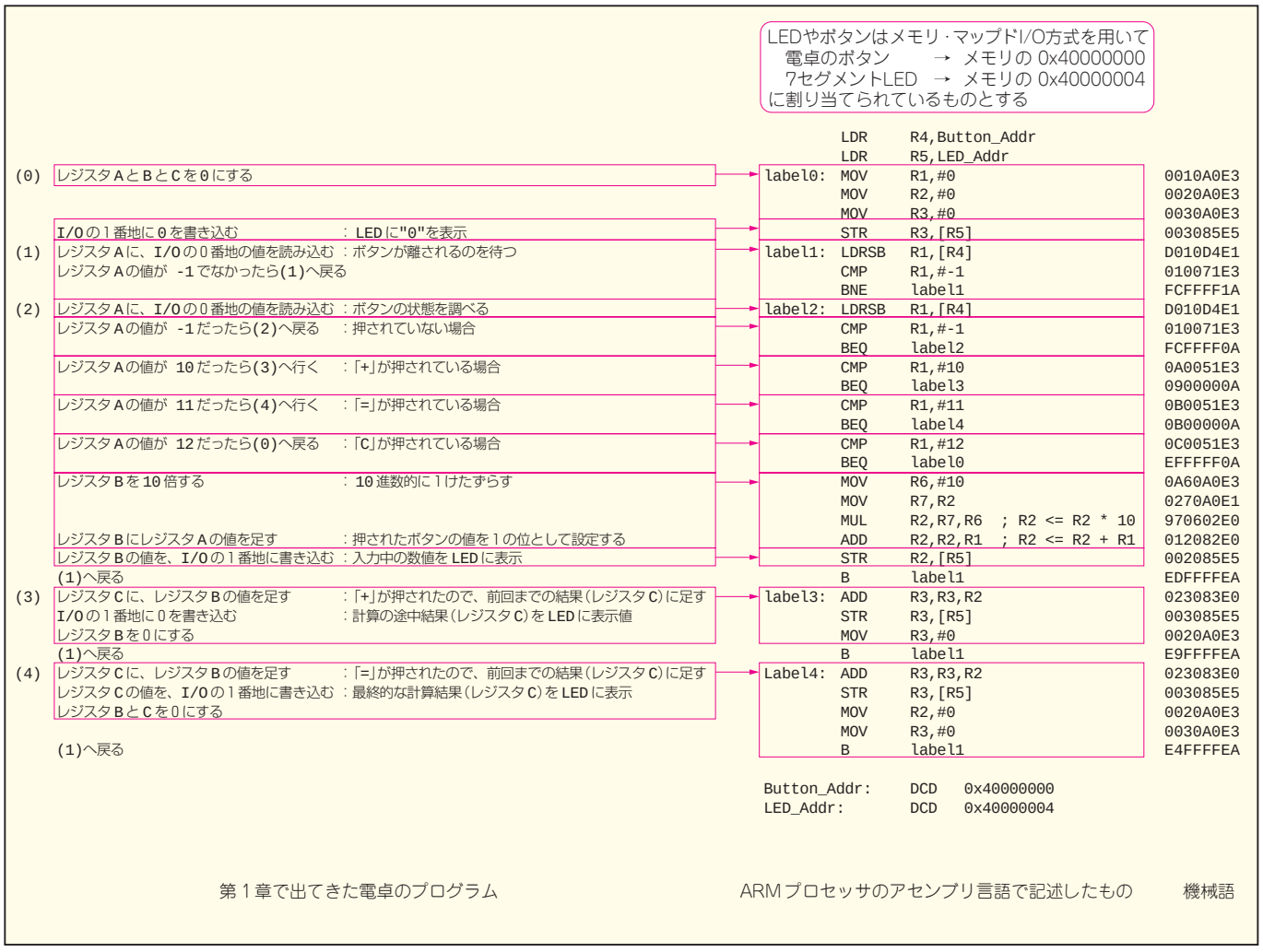
```
STR  R3,[R5]
```

という命令は，「R3の内容をR5が覚えている番地に書く」という命令です．R5にはI/Oの1番地に相当する0x40000004が書かれているため，上記命令は「R3の内容を7セグメントLEDに表示する」という動作をします．

CMP命令はレジスタの値を何かと比較する命令です．比較した後にあるBEQという命令は，「もし直前の比較に

KEYWORD ── 高級言語，低級言語，C言語，機械語，アセンブリ言語，移植性，関数，コンパイル，リンク，I/Oアクセス，ライブラリ

リスト1　第1章の電卓を実際のアセンブリ言語で記述



よって同じ値だということがわかったら○○へ行く」という命令です．BNE は BEQ の逆で「もし直前の比較によって同じ値で“ない”ということがわかったら○○へ行く」という命令になります．頭文字の「B」は分岐するという意味の「Branch」から，「EQ」は等しいという意味の「Equal」からきています．同じように「NE」は「Not Equal」です．なお，**リスト1**の最後にある「B」という1文字の命令は「条件に関係なく常に○○へ行く」という意味です．

残る命令は ADD と MUL だけです．ADD はその名のとおり「足し算せよ」という意味，MUL は「Multiply」の略で「かけ算せよ」です．例えば，

```
MUL R2,R7,R6
```

は，「R7 と R6 を掛けた結果を R2 に保存せよ」という意味になり，

```
ADD R2,R2,R1
```

という命令は，「R2 に R1 を足した結果を R2 に保存せよ」という意味です．

これで，このアセンブリ言語で書かれたプログラムが，**リスト1**の左の列に書かれた日本語のプログラムと同じ内容になっていることがおわかりいただけたと思います．

● もっと簡単に書きたい！ C言語で書いてみよう

説明を聞いた後であったとしても，**リスト1**のプログラムはお世辞にも「わかりやすい」ものではありません．右側の機械語よりは圧倒的にわかりやすいのはいうまでもないのですが，実際，説明をする側の筆者としてもアセンブリ言語は大変です．

リスト2は同様のことをC言語で書いたものです．C言語の解説は第5章以降に詳しく書かれているのでここでは