命令のビットパターン

山本昌志*

2004 年 6 月 28 日

1　これまでの復習と本日の学習

本日は、命令をビットパターンに変換することを学習する。

1.1　復習

これまでの授業で、教科書の p.24 まで終了した。これまで学習したことで、大事なことをまとめると、次のようになる。

- コンピューターの動作
  - コンピューターの仕事は、データの処理である。
  - そのためのハードウェアーとして、メモリーと CPU がある。これが、コンピューターの基本構成である。
  - 処理の対象をデータといい、それは命令に従い処理される。
  - データも命令も、メモリーに書かれている。
  - メモリーに書かれた命令に従い、CPU がメモリーに書かれたデータを処理する。
- メモリー
  - 役割は、命令とデータを格納することである。
  - メモリーの内容は、アドレスを指定することにより、読み書きできる。
  - MOMET II の場合、アドレスは 16 ビットで指定する。したがって、$(0000)_{16}$ から $(FFFF)_{16}$ 番地まで指定でき、65536 個の記憶領域がある。
  - 一つの記憶領域には、16 ビットの内容を記憶できる。この 16 ビットを 1 ワード (語) と言う。
- CPU
  - 役割は、命令に従い、レジスタやメモリーの内容を書き換えることである。

*国立秋田工業高等専門学校　電気工学科

  - CPU 内のメモリーをレジスターと言う。

- 高級言語 (FORTRAN や C 言語など) を実行する場合、そのプログラムは翻訳 (コンパイル) され、最終的には、2 進数のビット列になる。アセンブラ言語もビットパターンに変換できる。
- プログラムは、命令とデータから構成される。実行時、プログラムは、メモリーに格納される。したがって、実行時、命令とデータはビットパターンに変換しなくてはならない。
- コンピューターは、このビットパターンに従って、電圧を変化させている自動機械に過ぎない (チューリング機械)。

1.2 本日の学習

これまで、学習してきたことは以上の通りである。ここで、メモリーの内容について、ひとつ学習していないことがある。それは、命令のビットパターンである。整数や文字のビットパターンを学習してきたが、命令については、何も言っていない。本日は、命令をビットパターンに変える方法を示す。これができると、アセンブラ言語で書かれたプログラムをマシン語に変換することができる。

ほとんど、コンピューターが発明された初期のプログラムの方法である。初期のプログラマーと同じことをするのである。

2 簡単なプログラム

命令が機械語に変換する方法を学習する。一般的なことは言わないで、実際の例でそれを示すことにする。ここでの学習の例に用いるプログラムは、図 1 のとおりである。このプログラムが行うことは、

- 3+5 を計算する

だけである。FORTRAN では `C=3+5`、C 言語では `c=3+5;` と書けばすむことを、アセンブラではこのようにいろいろ書かなくてはならない。理由は、後でわかるだろう。

```
1 PGM   START
2       LD    GR1,A
3       ADDA  GR1,B
4       ST    GR1,C
5       RET
6 A     DC    3
7 B     DC    5
8 C     DS    1
9       END
```

図 1: 3+5 を計算するプログラム

このプログラムの動作を詳しく説明すると、以下のようになる。フローチャートを図 2 に示す。これらが、ハードウェアーでどのように実現されるか、授業で説明するので、良く理解してほしい。

行	文	機能	種類
1	PGM START	プログラムは、START から開始	
2	LD GR1,A	A の値をレジスタ GR1 に格納	命令
3	ADDA GR1,B	GR1 と B の値を加算して、GR1 に格納	命令
4	ST GR1,C	GR1 の値を C に格納	命令
5	RET	呼び出し元へ戻る	命令
6	A DC 3	値 $(3)_{10}$ を格納	データ
7	B DC 5	値 $(3)_{10}$ を格納	データ
8	C DS 1	1 ワード予約	データ
9	END	プログラムの終わりを示す	

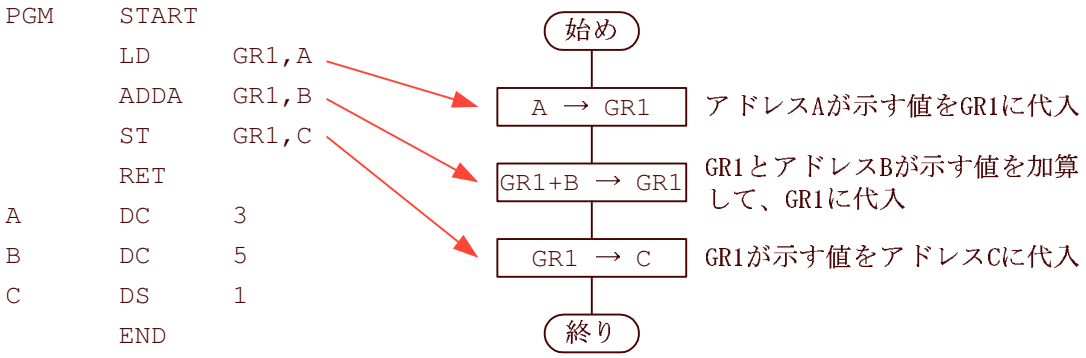

図 2: プログラムとフローチャート

本日の授業は、コンピューターの動作ではなく、その命令がどのようにビットパターンに変換されるか学習することである。この簡単なプログラムの命令の詳細は知らなくても良いから、命令もビットパターンに変換されることを理解してほしい。

3 アセンブラ言語を機械語に変換

3.1 マシン語変換表

今まで、さんざん言ったようにコンピューターのプログラムは、全てビットパターンに変換される。データである整数や文字の変換の仕方は、すでに学習した。後は、命令の変換の仕方だけである。

文字の変換の仕方が、表 (教科書 p.13 JIS X0201) になっていたように、命令も表になっている。教科書の p.213 の命令語の構成に書かれている。全ての命令がこの表に書かれている。40 個弱しかないのである。

ただ、表の見方が、文字のコード表よりちょっと難しい。整数や文字は、16 ビットのビットパターンであったが、命令の場合は 16 ビットであったり、32 ビットだったりする。少し厄介であるが、慣れればたいしたことない。

それでは、実際の表の見方を示す。まずは、表の中央より右側に機械語命令が書かれている。その左側がマシン語を表し、右側がその動作を記述している。今は、動作はどうでもよいので、アセンブラの命令と機械語の対応を考える。たとえば、LD 命令を例にとる。表の機械語命令 LD を見ると、2 つあることに気が付く。それは、

```
LD   r,adr,x
LD   r1,r2
```

である。LD は分かるとして、それ以外 (オペランド) が分からない。詳しいことは、今後の学習に譲るとして、それを簡単にまとめると、次のようになる。

r	汎用レジスター	GR0〜GR7
r1	1 つの命令で 2 つの汎用レジスターを使うときの一方	GR0〜GR7
r2	もう一方の汎用レジスター	GR0〜GR7
adr	アドレスを示す。	レベル名が書かれることが多い。
x	アドレスをシフトするインデックスレジスタ。	GR0〜GR7

これで表の見方がわかった。アセンブラのプログラムをマシン語に変換できるようになった。

たとえば、ラベル A が $(A007)_{16}$ として、LD GR1,A,GR2 という命令は、

$$\texttt{LD GR1,A,GR2} \quad \Rightarrow \quad \begin{matrix}(1012)_{16}\\(A007)_{16}\end{matrix}$$

と変換される。また、LD GR1,GR2 という命令は、

$$\texttt{LD GR1,GR2} \quad \Rightarrow \quad (1412)_{16}$$

と変換される。これで、命令が 1 語の場合と 2 語の場合があることが分かるであろう。表を見て分かるように、2 語を使う命令場合、その 2 語目は必ず、アドレスとなっている。

これで、全て終わるのはまだ早い。賢い者は、LD GR1,A という命令の変換方法に疑問が湧くであろう。インデックスレジスターが無い場合である。これは、

$$\texttt{LD GR1,A} \quad \Rightarrow \quad \begin{matrix}(1010)_{16}\\(A007)_{16}\end{matrix}$$

と変換される。すなわち、命令を構成する 2 語の最初の 1 語の第 0〜3 ビットがゼロの場合、インデックスレジスターが無いと判断されるのである。もし、インデックスレジスターに GR0 が使えると、インデックスレジスターが無い場合と GR0 を使っている場合の区別ができなくなる。そのような理由から、インデックスレジスターに GR0 が使えないのである。ハードウェアー (CPU) がそうなっているからである。

3.2 ハンド アセンブル

準備が整ったので、図 1 のプログラムを機械語に変換する。これをビットパターンに変換したものが、教科書の p.17 の図 2.4 に書かれている。ただし、この表には間違いがあるので、注意が必要である。プログラムの最初の PGM START はアセンブラ命令と言って、機械語に変換されない。これについては来週の授業で説明する。したがって、最初に機械語に変換される命令は、LD GR1,A となる。その変換は、次のように行う。

1. LD という命令から、16 進数 4 桁の表示の最上位の桁は $(1)_{16}$ と分かる。すなわちビットパターンは、$(0001)_2$ である。
2. 次の桁は、LD には、$(0)_{16}$ か $(4)_{16}$ である。ここでは、LD r,adr,x のパターンとなっているので、次の桁は $(0)_{16}$ と分かる。すなわちビットパターンは、$(0000)_2$ である。
3. 次の桁は、汎用レジスターを示す。$(1)_{16}$ と分かる。ビットパターンは、$(0001)_2$ である。
4. 次の桁は、インデックスレジスターを示す。インデックスレジスターは無いので、$(0)_{16}$ である。ビットパターンは、$(0000)_2$ となる。

ここの命令は、LD r,adr,x のパターンであるので、命令語長は 2 語である。最初の 1 語は今示したとおり、$(1010)_{16}$ である。次の 1 語は、ラベル A のアドレスである。これは、プログラムが格納されるアドレスに依存する。ここでは、教科書 (p.17 の図 2.4) に沿って、$(A000)_{16}$ からプログラムは格納されるとすると、A のアドレスは $(A007)_{16}$ となる。これが第 2 語のビットパターンとなる。以上をまとめると、

命令	2 進数	16 進数
LD GR1,A	0001000000010000	1010
	1010000000000111	A007

となる。

このようにアセンブラー言語を人間が表を見ながら、マシン語に変換することをハンドアセンブル (ほとんど死語か?) と言う。これは単純作業なので、通常は、コンピューターの仕事である。ただし、コンピューターを学習する者にとっては、一度は経験しておきたいことである。

4 課題

以下の通り課題を出すので、次回の授業までに、レポートにまとめて提出すること。

- 内容
  - 残りの命令

    ADDA GR1,A

    ST GR1,C

    をハンドアセンブルしなさい。結果は分かっているので、その過程をきちんと書くこと。
  - 教科書 P.17 の図 2.4 は間違っている。間違いを見つけて、訂正しなさい。
- 期日
  - 次回の授業 (7 月 6 日) まで