電子計算機 **3E**

3E　2003.7.14

本節の授業のテーマ

本日の事業のテーマは、以下のとおりです。

(1)　CASL II の書き方
(2)　アセンブラ命令
(3)　プログラムの記述順序

本日の授業のゴールは、以下のとおり。

- プログラムの行の書き方が分かる。
  - ラベル欄
  - 命令コード欄
  - オペランド欄
  - 注釈欄
- アセンブラ命令の使い方が分かる。
  - START
  - END
  - DC
  - DS
- プログラムの記述順序が分かる。

1. CASL II の書き方

1.1 コーディングの約束

コーディングとはプログラムを書く動作のことです。ここでは、CASL II のプログラムの書き方の約束を学習します。FORTRAN でもプログラムの書式がありましたよね。あれと同じです。FORTRAN では、

- 第 1 桁目に`C`または`*`を書くと、その行は注釈行となる。
- 文番号は、第 1 桁目から第 5 桁目に書く。
- 第 6 桁目が空白、または **0** 以外の文字であれば前の行からの継続と見なされます。
- 文は、第 7 桁目から 72 桁目までに書く。

というような約束があったと思います。これに対応するものが、CASL II にもあります。

FORTRAN では各桁は、コメントのしるし(* or C)を書く欄(第 1 桁)や文番号を書く欄(第 1～5 桁)、継続のしるしを書く欄(第 6 桁)、文を書く欄(第 7～72 桁)に分かれていました。それと同じように、CASL II でも行は機能毎に欄が分かれています。例えば、先週学習したプログラムを例に取ると、


```
PGM         START
            LD          GR1,A
            ADDA        GR1,B
            ST          GR1,C
            OUT         D,E
            RET
A           DC          3           ;アドレスAに3を格納
B           DC          5           ;アドレスBに5を格納
C           DS          1           ;アドレスCから1語分の領域確保
D           DC          'END'       ;アドレスDから文字'END'を格納
E           DC          3           ;アドレスEに3を格納
            END
```

のようになっています。

各欄は FORTRAN のように桁数で分けられているわけではありません。欄の区切りは、1 個以上の空白で示します。したがって、

- ラベル欄の先頭の空白は許されません。空白があると、それはラベルではなく命令コードと解釈されます。
- 先の例のように、各行の命令コード欄やオペランド欄、注釈欄の書き始めをそろえる必要はありません。しかし、各欄の書き始めの位置はそろえたほうが、プログラムは分かりやすくなります。できるだけ、そろえるようにしましょう。

となります。

1.2 欄の説明

1.2.1 ラベル欄

教科書にも書かれている通り、記述の約束は以下の通りです。

- プログラムのロジックでラベルが不要な場合は、記述しなくても良いです。
- ラベルは、8 文字以内です。先頭はアルファベットの大文字、2 文字以降はアルファベットの大文字、数字いずれでも良いです。
- 必ず先頭 (第 1 文字) から始めます。第 1 文字が空白の場合は、ラベル名は無いものみなされ、命令コードと解釈されます。
- 汎用レジスタの名前の GR0 から GR7 は予約語であり、ラベル名として使用できません。命令コードのオペランドで、ラベルなのかレジスタなのか区別できなくなります。

ラベルは、その記述する位置から FORTRAN の文番号にも似ています。あるいは、先週の例でもわかるように、変数名にも似ています。実際、FORTRAN の文番号や変数名のような使われ方をします。しかし実際には、これはアドレスをあらわします。アドレスは、命令に従い、以下のようになっています。

- 機械語命令のラベルの場合は、その機械語命令が格納されている 2 語分の領域のうち、その先頭アドレスを表します。実際のプログラムでは、ジャンプ命令とともに使われ、そのアドレスに制御が移ります。FORTRAN の GO TO 分でその文番号に制御が移るのと同じです。
- DC 命令[1]の場合、ラベルは定数が格納されている領域[2]の先頭アドレスを示します。使い方は FORTRAN の変数名に似ていますが、実態はアドレスです。
- DS 命令[3]の場合、ラベルはこの命令によって確保されている主記憶の領域[4]のの先頭アドレスを示します。
- IN や OUT のマグロ命令の場合は、ラベルは複数の命令群のうちの先頭の命令が格納されているアドレスを示します。

ラベルがアドレスを表すことが理解できれば、簡単ですよね。常識通りに解釈すればよいのです。

重要なポイント

以前学習した通りアセンブラのプログラムは、主記憶装置 (メインメモリー) の中に格納されているデータを処理 (いろいろな演算) します。また、プログラムの命令も主記憶装置に格納されています。主記憶装置に格納されているデータや命令にアクセスする場合、主記憶装置のアドレスを指定することになります。したがって、アセンブラでは、アドレスが重要になり、プログラマーは意識しなくてはなりません。

高級言語の場合、アドレスに関してはコンパイラーが勝手に処理をしてくれます。例えば、FORTRAN で変数を使った場合、プログラマーがその変数のアドレスに注意を払う必要はありません。これは、コンパイラーが変数とアドレスの関係の表を持っており、それに従い、上手にマシン語に変換してくれるからです。

アセンブラのでは、コンパイラーの代わりにプログラマーが変数とアドレスの関係を考えなくてはなりません。

---

[1] Define Constant データを定義する。実際には、実行時、主記憶装置にデータを格納する。

[2] 1 語の場合もあれば、複数語の場合もある。

[3] Define Storage 主記憶装置にデータを格納する領域を確保する。

[4] 1 語の場合もあれば、複数語の場合もある。

1.2.2 命令コード欄

この欄には、アセンブラ命令(非実行文)、機械語命令、マクロ命令を書きます。教科書にも書かれている通り、記述の約束は以下の通りです。

- ラベルの後に 1 個以上の空白の後、命令コードを書きます。
- ラベルが無い場合は、命令コードの前に 1 個以上の空白の後、記述します。

1.2.3 オペランド欄

この欄には、命令のオペランドを記述します。オペランド(operand：被演算子)とは命令の対象となる値やレジスタのことです。CASL II では汎用レジスタ番号、記号番地(ラベルのこと)、あるいは絶対番地がオペランドとなります。その記述方法は、教科書に書かれているように、以下の通りです。

- 命令コードの後に 1 個以上の空白の後、オペランドを書きます。
- 複数のオペランドは、カンマ`,`で区切って、連続して書きます。

1.2.4 注釈欄

行中にセミコロン`;`を書くことのより、それから行末まで注釈(コメント)として扱うことができます。FORTRAN の注釈文と同じで、プログラムの実行に何ら影響を与えません。プログラムの内容を分かりやすくするために書きます。あるいは、その行を実行させないときに行頭にセミコロンを`;`を追加してデバック作業を進めることがあります。

- 行の先頭、あるいはセミコロンの前に空白しかない場合は、行全体が注釈となります。
- オペランドの後に 1 個以上の空白があれば、そこ以降は注釈となります。

2. 非実行文(アセンブラ命令)

2.1 プログラムの開始(START)

書式

ラベル欄	命令コード欄	オペランド欄
ラベル	*START*	*[実行開始番地]*

プログラムの実行開始番地(アドレス)をアセンブラーに対して指示するアセンブラ命令です。プログラムの先頭に必ず書く必要があります。これが無いと、どこからプログラムを実行するか分かりません。この START 命令が示すアドレスがプログラムを実行するときに最初の PR(プログラムレジスタ)の値になります。これが実行開始番地です。

実際の図 1 のプログラムで START 命令の役割を示します。図 1 に示すように、START はアセンブラ命令であるため、マシン語に変換されません。しかし、アセンブラーにこのプログラムは、ラベル BEGIN から実行しますということを示しています。ラベル BEGIN はアドレスです。そのアドレスがプログラム実行開始時にプログラムレジスターPR にセットされます。

もし、オペランド欄に記述が無い場合、START 命令に次の行からプログラムの実行は開始されます。

START 命令のラベルは、そのプログラム自身で参照されることはありませんが、絶対に必要です。ほかのプログラムがこのプログラムを実行したい場合、このラベルが使われます。そしてこのラベルが示すアドレスは、このプログラムの実行開始番地になります。したがって、図 1 のラベル PGM と BEGIN は同じアドレス 0020 を示します。

2.2 プログラムの終わり (END)

書式

ラベル欄	命令コード欄	オペランド欄
	END	

プログラムの最後に必ず書く必要があります。アセンブラーに対して、プログラムの終わりを示します。FORTRAN の END 文と同じで、FOORTRAN の場合はコンパイラーにプログラムの終わりを示す役割があります。

END 文はラベルをつけることはできません。アセンブラ命令である END 文は、主記憶装置に格納されません。そのため、対応するアドレスがありません。また、当然、処理の対象となるオペランドもありませんので、書くことはできません。しかし、END 文の後に注釈が有ってもかまいません。アセンブラーは無視します[5]。

2.3 データの定義 (DC)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*DC*	*定数,[定数]*

プログラムに必要な定数を定義する命令です。主記憶装置の領域を確保して、そこに定数を格納します。カンマで区切ることにより、複数の定数を格納可能です。その場合、1 語毎に主記憶装置に格納されます。文字の場合は、表 1 に示すようにシングルクォーテーションで囲むと、1 文字ずつ 1 語毎に格納されます。いずれにしても、複数の文字や数字を格納する場合、連続した領域にデータは格納されます。定数の種類と書き方を表 1 に示します。

ラベルは、そのデータの先頭アドレスを示します。ラベルは無くても良いですが、通常はラベルを付けます。ラベルが無いと、目的のデータにアクセスが困難になります

定数の定義と言いましたが、実態はメモリーの領域を確保して初期値を決めているだけです。従って、そのデータを書き換えることもできます。

表 1 DC 命令の定数

定数の種類	オペランド	命令の説明
10 進数定数	n	10 進数定数を n で指定
16 進数定数	#h	4 桁の 16 進数定数を#の後に記述
文字定数	`文字列`	文字列を指定
アドレス定数	ラベル	アドレスをラベルで指定

[5] END 命令の注釈に限らず、どこにある注釈でもアセンブラーは無視します。注釈はプログラマーのためのものであり、アセンブラーはいっさい関知しません。

2.4 主記憶装置の領域の確保(DS)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*DS*	*n*

プログラムの実行に必要な主記憶領域を確保します。確保する領域の大きさは、10 進数定数 n で指定します。n 語領域を確保します。確保された領域の先頭アドレスが、ラベルになります。

n はゼロ以上の整数です。ゼロとした場合は、領域の確保は行われませんが、ラベル名は有効です。

主記憶装置			プログラム			
adress	value		ラベル	命令コード	オペランド	注釈
			PGM	START	BEGIN	
0 0 2 0	1 0 1 0		BEGIN	LD	GR1,A	
0 0 2 1	0 0 2 7					
0 0 2 2	2 0 1 0			ADDA	GR1,B	;加算
0 0 2 3	0 0 2 8					
0 0 2 4	1 1 1 0			ST	GR1,C	
0 0 2 5	0 0 2 9					
0 0 2 6	8 1 0 0			RET		
0 0 2 7	0 0 0 3		A	DC	3	;3を格納
0 0 2 8	0 0 0 5		B	DC	5	;5を格納
0 0 2 9	0 0 0 8		C	DS	1	;1語の領域確保
				END		;プログラムの終了

図 1 加算プログラムと主記憶の内容。プログラム実行後の内容を示している。$(0029)_{16}$ 番地の内容が加算結果の $(0008)_{16}$ になっている。

3．プログラムの記述順序

気がついている人がいるかもしれませんが、プログラムの記述順序は、図 2 のようになります。START と RET の間に命令を書きます。RET と END にデータを書きます。START と RET の間に DC や DS 命令を書くと、そこのデータが命令と解釈されてしまいます。そのため、プログラムの実効命令の範囲である START と RET の間には、データを記述する DC や DS 命令は書きません。

この性質を悪用?して、DC を用いて、機械語命令を書くことができます。図 3 のプログラムは、図 1 のプログラムと全く同じ動作をします。ここまでくると、アセンブラではなくてマシン語ですが、面白いでしょう。

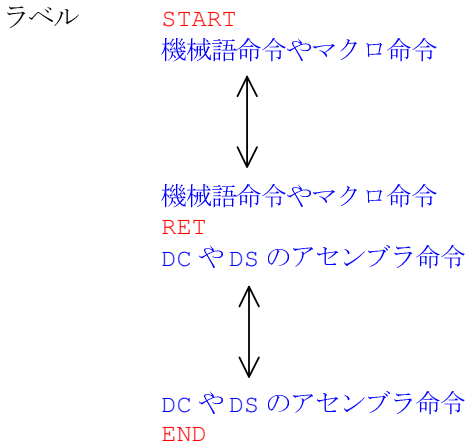

図 2 CASL II のプログラムの記述順序。

```
PGM         START
            DC          #1010
            DC          #0027
            DC          #2010
            DC          #0028
            DC          #1110
            DC          #0029
            DC          #8100
            DC          3
            DC          5
            DS          1
            END
```

図 3 図 1 のプログラムを DC 命令だけで記述。