アセンブラ命令 (非実行文)

山本昌志*

2004 年 8 月 23 日

1　夏休み前の復習と本日の学習

1.1　復習

夏休み前に、コンピューターやアセンブラ言語に関しては、次のようなことを学習した。

- コンピューターの最も基本的な構成要素は、CPU とメモリーである。メモリーにプログラム (命令とデータ) が記憶され、その内容に従い、CPU が動作する。CPU は、論理回路で構成された、自動機械に過ぎない。
- コンピューターの中では、0 と 1 の信号が、猛烈な勢いで変化して、計算を行う。この 0 と 1 の信号が電圧に対応し、その動作にはブール代数を使って記述できる。
- どんなコンピューターでも、0 と 1 で書かれた機械語で動作する。コンピューターが理解できる言葉は、機械語のみである。
- 0 と 1 の羅列の機械語は、人間に分かりにくい。そこで、この 0 と 1 の組み合わせを、記号で表すことが考えられた。この記号をアセンブラ言語と言う。
- アセンブラ言語は、機械語に比べ、かなり人間に分かりやすい言語である。アセンブラ言語を機械語に変換するプログラムは、アセンブラー (assembler) と呼ばれる。アセンブラーを複雑にすることにより、いろいろな機能を持たすことは出来るが、CPU の各命令、メモリーの番地などの概念は、そのままアセンブラ言語に受け継がれた。アセンブラ言語と機械語の 1 つの命令は、明確に対応がつけられている。アセンブラ言語でプログラムを作成する場合、アルゴリズムは機械語のレベルで考えなくてはならない。
- CPU 毎に機械語が異なるように、アセンブラ言語が異なる。基本情報処理試験の場合、特定の CPU のアセンブラ言語の出題は好ましくないので、特別なアセンブラ言語、CASL II が考えられた。それが動作するハードウェアーが COMET II である。

CASL II に関しては、次のようなことを思い出せば、本日の授業は理解できるであろう。

*国立秋田工業高等専門学校　電気工学科

- CASL II のプログラムの書き方

ラベル欄	命令コード 欄	オペランド 欄	注釈欄
PGM	START	BEGIN	; プログラムの開始
BEGIN	LD	GR0,A	; アドレス A の値を GR1 にコピー
	ADDA	GR0,B	;GR0←GR0+B
	ST	GR0,C	;GR0 の値をアドレス C にコピー
	RET		;RETURN
A	DC	1	; アドレス A の値を 1 にする
B	DC	2	; アドレス B の値を 2 にする
C	DS	1	; アドレス C から 1 語予約する
	END		; プログラムの終わり

- CASL II の命令の種類

アセンブラ命令	機械語に変換するときに、アセンブラーに指示する命令。CPU が持っている命令ではないので、特定のビットパターンに変換されない。
機械語命令	CPU が持っている命令。CPU の命令に応じた特定のビットパターンに変換される。
マクロ命令	機械語命令を組み合わせたもの。

1.2 本日の学習内容

本日は、アセンブラ命令 (START, END, DC, DS) について、その役割と動作を学習する。

2 プログラムの開始と終了

2.1 プログラムの始まり (START)

書式

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

- プログラムの先頭は、START 命令で始まらなくてはならない。
- ラベルは、必要不可欠である。
- 実行開始番地は無くても良い。絶対番地を書くことは稀で、通常、最初に実行する命令が書かれている行のラベル名を書く。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令の役割は、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ことである。プログラムの先頭に必ず書く必要があり、これが無いと、どこからプログラムを実行するか分からない。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、最初の PR(プログラムレジスタ) の値になる。これが実行開始番地である。

次のプログラムで START 命令の役割を考える。

```
1  PGM   START
2  BEGIN LD    GR1,A
3        ADDA  GR1,B
4        ST    GR1,C
5        OUT   D,E
6        RET
7  A     DC    3       ; アドレス A に 3 を格納
8  B     DC    5       ; アドレス B に 5 を格納
9  C     DS    1       ; アドレス C から 1 語分の領域確保
10 D     DC    'END'   ; アドレス D から文字'END' を格納
11 E     DC    3       ; アドレス E に 3 を格納
12       END
```

図 1: 3+5 を計算して、END と表示するプログラム

このプログラムをアセンブラー[1]が機械語に変換すると、図 2 のようになる。START と END 命令はアセンブラ命令であるため、マシン語に変換されない。他のアセンブラ命令、DC や DS は、それが示すデータに変換される。

START 命令はマシン語に変換されないが、プログラマーはアセンブラーに、このプログラムはラベル BEGIN から実行すると言うことを伝えなくてはならない。ラベル BEGIN は、最初に実行する命令

```
LD GR1,A
```

の先頭番地が格納されているアドレスを示す。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 にセットされる。それを確実にするために、START 命令がある。この命令で、最初に実行させる命令を示すのである。CPU は、その番地に書かれた 0 と 1 の集まりは命令と解釈する。

START 命令など無くして、最初の行から実行するように約束することも出来るであろう。こうすると、いつも先頭の行から実行することになり、実際プログラムを書く場合不便なことがある。START 命令が有った方が、便利なのである。

もし、オペランド 欄に記述が無い場合、START 命令の次の行からプログラムの実行は開始することになっている。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ないが、絶対に必要である。ほかのプログラム、たとえば OS がプログラムの実行を指示する場合、このラベルが使われる。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る。したがって、図 1 のラベル PGM と BEGIN は同じアドレス#0020 である。

[1] 機械語に変換するプログラム

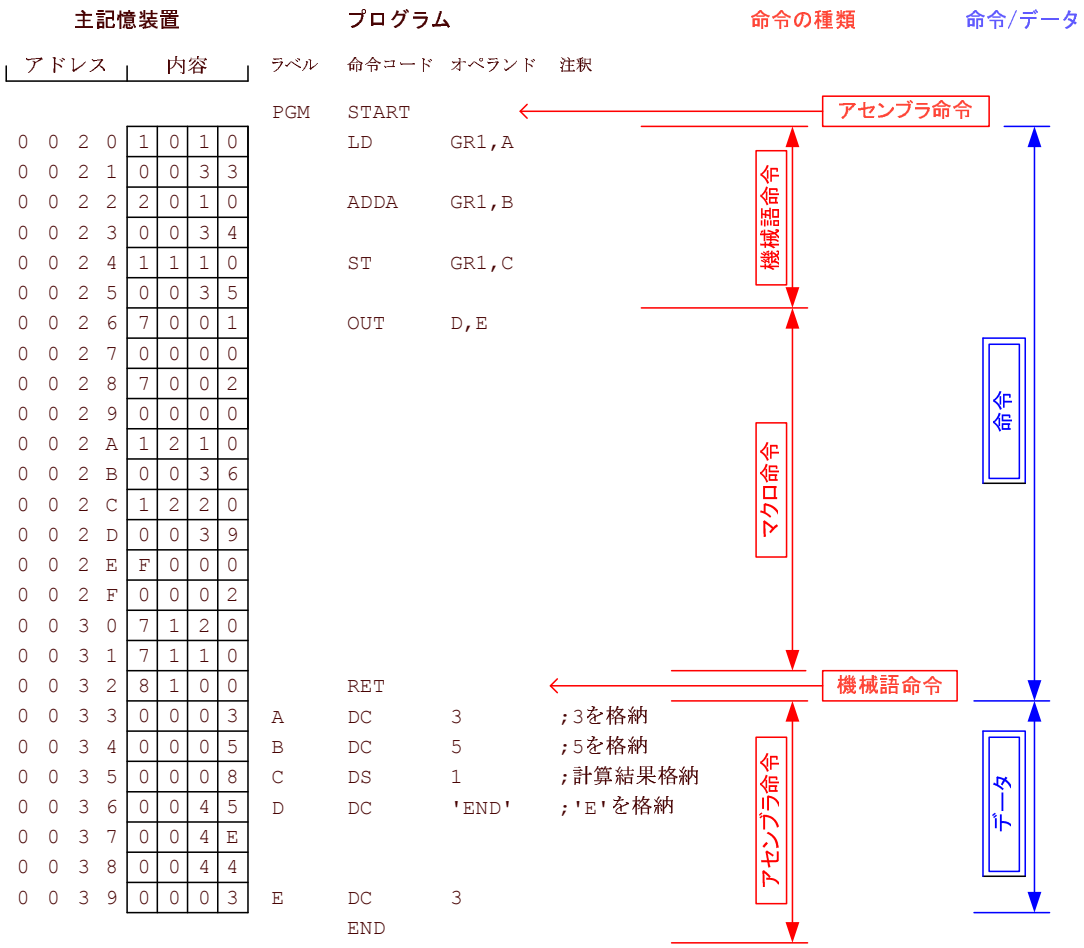

図 2: メモリーの内容と命令の種類

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 END 命令を書かなくてはならない。
- ラベル欄とオペランド欄は、書いてはならない。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令の役割は、プログラムの終わりをアセンブラーに対して知らせることである。プログラムの最後に必ず書く必要があり、これが無いと、どこでプログラムが終わったのか分からない。プログラムの実行が終わったところではなく、プログラマーが記述したソースコードの終わりを示す。

END 文はラベルをつけることはできない。この命令は、主記憶装置に格納されないので、対応するアドレスが無いからである。また、処理の対象となるオペランドもありませんので、書くことはできない。しかし、END 文の後に注釈は許される。アセンブラーは、これは無視する[2]。

3 定数の定義

以前から述べているように、プログラムは命令とデータから構成させれる。このうちデータを書き表すために、次から述べる DC が使われる。

3.1 10 進定数

3.1.1 内容

DC:Define Constant

書式

ラベル欄	命令コード欄	オペランド欄
[label]	DC	n

- ラベルは、無くても良い。その場合は、データが格納されるアドレスは、前の行の続きである。ラベルが無いと、目的のデータへのアクセスが複雑になるので、通常は付ける。
- n は、10 進数の整数である。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

役割は、10 進数整数のデータを定義することである。CASL の場合、整数は 16 ビットで表されるため、その範囲は-32768〜32767 までである。これを超えた場合、その下位 16 ビットが機械語に変換される。

[2] END 命令に限らず、どこにある注釈でもアセンブラーは無視する。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ない。

3.1.2 例

```
AA  DC  100
BB  DC  -3
```

3.2 16 進定数

3.2.1 内容

書式

ラベル欄	命令コード欄	オペランド欄
[label]	DC	#h

- ラベルは、無くても良い。その場合は、データが格納されるアドレスは、前の行の続きである。ラベルが無いと、目的のデータへのアクセスが複雑になるので、通常は付ける。
- h は、4 桁の 16 進数の正の整数である。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令の役割は、16 進数整数のデータを定義することである。CASL の場合、整数は 16 ビットで表されるため、その範囲は#0000〜#FFFF までである。

3.2.2 例

```
CC  DC  #0027
```

3.3 文字定数

3.3.1 内容

書式

ラベル欄	命令コード欄	オペランド欄
[label]	DC	’文字列’

- ラベルは、無くても良い。その場合は、データが格納されるアドレスは、前の行の続きである。ラベルが無いと、目的のデータへのアクセスが複雑になるので、通常は付ける。
- 文字列は、JIS X 0201 で決められた文字から構成する。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令の役割は、文字列のデータを定義することである。CASL では 1 文字を 16 ビット (1 ワード) で表現するが、JIS X 0201 では 8 ビットで表現する。このことより、上位 8 ビットは 0 とし、下位 8 ビットで表現することになっている。以前学習したとおりである。

アポストロフィ「'」をデータとして使いたい場合は、それを 2 つ続けて「''」のように書く。そうすると、1ヶのアポストロフィがデータとして定義される。

ラベルは、第一文字目のデータが格納されたアドレスを示す。

3.3.2 例

```
DD  DC  'AT<&'
```

3.4 アドレス定数

3.4.1 内容

書式

ラベル欄	命令コード欄	オペランド欄
[label]	DC	ラベル名

- ラベルは、無くても良い。その場合は、データが格納されるアドレスは、前の行の続きである。ラベルが無いと、目的のデータへのアクセスが複雑になるので、通常は付ける。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令により、記号番地であるラベル名をアドレスの絶対番地メモリーに格納できる。指定されたラベル名の絶対番地がメモリーに格納される。

3.4.2 例

```
FF  DC  GG
GG  DC  2
```

3.5 複数の定数を記述

3.5.1 内容

書式

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

- カンマで区切って、いくつでも定数を書くことができる。

3.5.2 例

```
HH  DC  10,20,'AB',#FFFF
```

4 領域の確保

4.1 メモリー領域の確保

領域の確保の命令で、プログラムの実行に必要なメモリーの領域を確保する。計算途中や結果を貯めるために、必要な記憶領域を確保するのである。

4.1.1 内容

DS:Define Storage

書式

ラベル欄	命令コード 欄	オペランド 欄
[label]	DS	n

- ラベルは、無くても良い。その場合は、予約されるアドレスは、前の行の続きである。ラベルが無いと、目的のメモリー領域へのアクセスが複雑になるので、通常は付ける。
- n は、10 進数の整数である $(0 \leqq n)$。n 語分の領域を確保する。
- ラベル名は、確保された領域の先頭のアドレスである。
- 非実行文なので、フラグレジスタは変化しない (関係ない)。

この命令の役割は、プログラムの実行に必要なメモリーの領域を確保することである。

4.1.2 例

```
AA  DS  0
BB  DC  'AB'
CC  DS  2
```