電子計算機 **3E**

3E　2003.9.22

試験の範囲

・教科書の P.25～P.50 までです。
・あわせてハンドアセンブルも範囲とします。

本日の授業のテーマ

今までの学習内容をまとめます。

(1) CASL　II 命令の種類
(2) ハンドアセンブル
(3) プログラムの書き方
(4) アセンブラ命令
　・プログラムの開始(START)
　・プログラムの終了(END)
　・定数の定義(DC)
　・メモリー領域確保(DS)
(5) 機械語命令の書き方とアドレス
(6) フラグレジスタ
(7) 機械語命令(データ転送命令)
　・ロード(LD)
　・ストア(ST)
　・ロードアドレス(LAD)
(8) 機械語命令(算術演算と論理演算)
　・算術加算(ADDA)
　・算術減算(SUBA)
　・論理加算(ADDL)
　・論理減算(SUBL)
(9) プログラム

本日の授業のゴールは、以下のとおり。

・アセンブラ命令と機械語命令、マクロ命令の役割が理解できる。そして、0 と 1 のマシン語への対応が分かる。
・CASL II のソースを 0 と 1 の機械語に変換できる。
・アセンブラのプログラムの書き方が理解できる。
・アセンブラ命令(START, END, DC, DS)の動作が理解できる。
・機械語命令の書き方が理解できる。アドレスの意味がわかり、プログラム中での使用方法がわかる。
・データ転送命令の使い方が分かる。これに伴うフラグレジスタの動作が分かる。
・算術演算と論理演算の違いが分かり、それぞれの命令が使える。フラグレジスタの動作が分かる。

1 CASL II の命令の種類

- CASL II の命令は、①アセンブラ命令②機械語命令③マクロ命令に分けられる。

- アセンブラ命令は、アセンブラーという変換プログラム(CASL II→マシン語)に対して、指示を行う命令です。COMET II の CPU の動作の指示は行いません。したがって、この命令は 1 と 0 の組み合わせの機械語に変換されません。

START	プログラムの先頭を定義。実行開始番地を定義等
END	プログラムの終わりを明示
DC	領域を確保
DS	定数を定義

- 機械語命令は、COMET II の CPU の動作の指示を行います。機械語命令は、アセンブラーにより特定のビットパターンの機械語に変換されてます。実行時には、そのビットパターンが主記憶装置に格納されます。

ロード、ストア	LD ST LAD
算術、論理演算	ADDA ADDL SUBA SUBL 等
リターン	RET
その他いろいろ	

- マクロ命令とは、特定の機能を果たすためのいくつかの命令の集まりに名前を付けたものです。この名前を指定するだけで、これらの命令の集まりが実行されます。これにより、頻繁に使われる定形的な命令群をマクロ命令にすることにより、同じようなプログラムをいちいち書くことを省くことができます。多くの命令から構成されるため、アセンブラーにより変換されるビットパターンは非常に多くなります。

  IN, OUT 等

- これまで学習した命令(START, END, DC, DS, LD, ST, LAD, ADDA, ADDL, SUBA, SUBL)と RET は、それぞれどの種類に属するか憶えましょう。

2 ハンドアセンブル

- 教科書 P.213 の表を用いて、0 と 1 のビットパターンのマシン語に変換出来ます。この表は問題に書きますので憶える必要はありません。ハンドアセンブルが出来るようになってください。

ソースプログラム			メインメモリー アドレス	内容(16 進数)	ソースとの対応	
PGM	START					
	LD	GR1,A	0020	1010	LD	GR1,A
	SUBA	GR1,B	0021	0027		
	ST	GR1,C	0022	2110	SUBA	GR1,B
	RET		0023	0028		
A	DC	285	0024	1110	ST	GR1,C
B	DC	353	0025	0029		
C	DS	1	0026	8100	RET	
	END		0027	011D	285	
			0028	0161	353	
			0029	FFBC	-68	

3 プログラムの書き方

- CASL II のプログラムは、以下のようにかかれます。図 1 参照

  - 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。
  - 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。


```
PGM     START
        LD      GR1,A
        ADDA    GR1,B
        ST      GR1,C
        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
        END
```

図 1 CASL II のプログラム

- ラベル欄に関しては、以下の約束があります。

  - 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。
  - 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。
  - 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。

- ラベルはアドレスをあらわします。そのアドレスは、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。1 語の場合はそのアドレス。
  - DC 命令の場合、ラベルは定数が格納されている領域の先頭アドレスです。
  - DS 命令の場合、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の先頭アドレスを示します。
  - IN や OUT のマクロ命令の場合は、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。

- オペランド欄には、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。

  - 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。

・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。プログラムの実行に何ら影響を与えません。

  ・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。
  ・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。

・プログラムは、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。

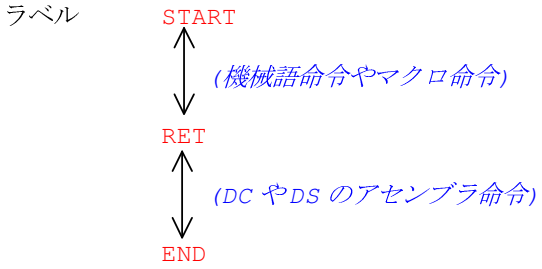


図 2 CASL II のプログラムの記述順序。

4 アセンブラ命令

4.1 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します。

・プログラムの先頭に必ず書く必要があります。この START 命令の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。

・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されます。

・START 命令のラベルは、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。

4.2 プログラムの終了(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。

・アセンブラ命令である END 文は、主記憶装置に格納されないのラベルをつけることはできません。処理の対象となるオペランドもありません。

4.3 定数の定義(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]*

・主記憶装置の領域を確保して、そこに定数を格納します。実態はメモリーの領域を確保して初期値を設定しているだけです。

・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。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。

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

4.4 メモリー領域確保(DS)

書式

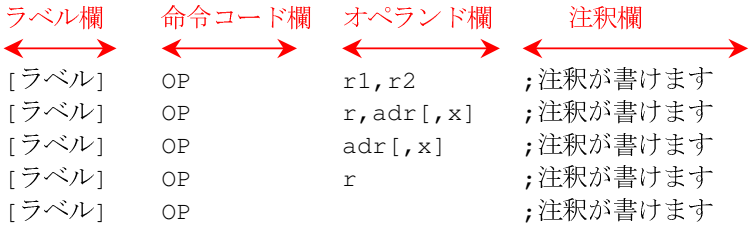
ラベル欄	命令コード欄	オペランド欄
[ラベル]	*DS*	*n*

・プログラムの実行に必要な主記憶装置の n 語領域を確保します。領域の大きさは、10 進数定数 n で指定します。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。

5 機械語命令の書き方とアドレス

・機械語命令の書き方は、オペランドが異なる以下の 5 種類に分類できます。r と r1, r2 は汎用レジスタを表します。adr はアドレスです。x は指標レジスタです。[ ]は、その中は省略可能をということを示しています。




```
[ラベル]   OP        r1,r2          ;注釈が書けます
[ラベル]   OP        r,adr[,x]      ;注釈が書けます
[ラベル]   OP        adr[,x]        ;注釈が書けます
[ラベル]   OP        r              ;注釈が書けます
[ラベル]   OP                       ;注釈が書けます
```

・指標レジスタは、基準点のアドレスにある値を加算してデータにアクセスするときに使います。オペランド欄に、

```
adr,x
```

と書きます。adr が基準点のアドレスで、x が指標レジスタです。実際にデータが操作される実アドレスは、adr+x です。指標レジスタ x は、汎用レジスタの GR1～GR7 を使います。

- 機械語命令の 1 行は、1 あるいは 2 語(ワード)のマシン語に変換されれ、メインメモリーに格納されます。その 1 あるいは 2 ワードで、命令の種類と対象であるオペランドを示す 0 と 1 のビットパターンのマシン語になるのです。

- COMET II のメインメモリーのアドレスは、16 ビットです。従って、アドレスの範囲は、0～65535(#0000～#FFFF)番地です。

- アドレスはつぎに示す 3 通りの方法で記述できます。最初の 2 つの 10 進数と 16 進数を使う場合、絶対アドレスを指定することになります。

10 進定数	10 進数の定数を用います。内容は、教科書に書かれている通りです。
16 進定数	16 進数の定数を用います。16 進数であることを表すために、先頭に#を付けます。
アドレス定数	ラベル名を指定します。アセンブラーにより、ラベル名がアドレスに変換されます。

- アドレス adr の指定は、10 進定数と 16 進定数、アドレス定数の他にリテラルでもそれを指定できます。リテラルを用いたアドレスは、10 進定数や 16 進定数、あるいは文字定数の前に‘=’の記号をつけます。

リテラルを使った場合

```
      :
      :
      LD   GR1,=2
      LD   GR2,=#FFFF
      LD   GR3,='A'
      LD   GR4,='ABC',GR1
      :
      :
```

左のリテラルを使ったプログラムと以下のプログラムは同じマシン語になる。

```
      :
      :
      LD   GR1,C1
      LD   GR2,C2
      LD   GR3,C3
      LD   GR4,C4,GR1
      :
      :
C1    DC   2
C2    DC   #FFFF
C3    DC   'A'
C4    DC   'ABC'
      END
```

6 フラグレジスタ

- フラグレジスタは、OF(Over Flag)と SF(Sign Flag)、ZF(Zero Flag)の 3 個のビットからなる。演算結果によって、それらのレジスタの値がセットされます。これ以外、例えば LD 命令の場合もフラグレジスタがセットされますので気をつけてください。

表 2 フラグレジスタの動作

Flag	bit	bit が設定される条件
OF	1	• 算術演算の結果が-32768～32767 に収まらなかったとき • 論理演算の結果が 0～65535 に収まらなかったとき • シフト演算では、最後に送り出されたビットが 1 のとき
	0	• 上記以外
SF	1	• 演算結果が負(ビット番号 15 が 1)のとき
	0	• それ以外(ビット番号 15 が 0)のとき
ZF	1	• 演算結果がゼロ(全てのビットが 0)のとき
	0	• 上記以外

・これまでの学習の範囲では、LD, ADDA, SUBA, ADDL, SUBL の機械語命令で、フラグレジスタがセットされます。

6 機械語命令(データ転送命令)

6.1 ロード(LD)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*LD*	*r1,r2*
[ラベル]	*LD*	*r,adr[,x]*

・メインメモリーから汎用レジスタ、あるいは汎用レジスター間で内容(データ)をコピーします。フラグレジスタの値は変化します。

・以下のような使い方があります。

```
LD    GR0,GR1      GR1 の内容を GR0 にコピー
LD    GR0,A        アドレス(A)の内容を GR0 にコピー
LD    GR0,A,GR1    アドレス(A+GR1)の内容を GR0 にコピー
LD    GR1,GR1      GR1 の内容を GR1 ピー。フラグレジスタを見ることにより、GR1 が正負、ゼロの判断が出来る。
```

6.2 ストア(ST)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*ST*	*r,adr[,x]*

・汎用レジスタの内容をメインメモリの指定番地にコピーします。フラグレジスタの値は変化しません。

・以下のような使い方があります。

```
ST    GR0,A        GR0 の内容をアドレス(A)にコピー
ST    GR0,A,GR1    GR0 の内容をアドレス(A+GR1)にコピー
```

6.3 ロードアドレス(LAD)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*LAD*	*r,adr[,x]*

・実行アドレスを汎用レジスタにコピーします。また、実行アドレス(adr,[x])の指定に10進定数や16進定数を指定することにより、直接、汎用レジスタに値を格納することができます。

・以下のような使い方があります。

```
LAD   GR0,A          Aが示すアドレスをGR0にコピー
LAD   GR0,A,GR1      (A+GR1)が示すアドレスをGR0にコピー
LAD   GR1,3          GR1に定数3を格納
LAD   GR1,-1         GR1に定数-1を格納
LAD   GR1,1,GR1      GR1の内容を加算(+1) GR1←1+GR1
LAD   GR2,-3,GR2     GR2の内容を減算(-3) GR2←3+GR2
LAD   GR2,0,GR3      GR3の内容をGR2にコピー GR3←0+GR3
```

7 機械語命令(算術演算と論理演算)

7.1 算術加算(ADDA)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*ADDA*	*r1,r2*
[ラベル]	*ADDA*	*r,adr[,x]*

・レジスタ間、あるいはレジスタと指定したアドレスの主記憶装置の内容の加算をします。演算は符号付です(2の補数表現、第15ビットが1の時負)。フラグレジスタは変化します(表2参照)。

・以下のような使い方があります。

```
ADDA  GR0,GR1        レジスタ同士の加算。GR0←GR0+GR1
ADDA  GR0,A          GR0←GR0+(アドレスAの内容)
ADDA  GR0,A,GR1      GR0←GR0+(アドレスA+GR1の内容)
ADDA  GR0,=5         リテラルを用いての加算。GR0←GR0+5
```

7.2 算術減算(SUBA)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*SUBA*	*r1,r2*
[ラベル]	*SUBA*	*r,adr[,x]*

・レジスタ間、あるいはレジスタと指定したアドレスの主記憶装置の内容の減算をします。演算は符号付です(2の補数表現、第15ビットが1の時負)。フラグレジスタは変化します(表2参照)。

・以下のような使い方があります。

```
SUBA  GR0,GR1        レジスタ同士の減算。GR0←GR0-GR1
SUBA  GR0,A          GR0←GR0-(アドレスAの内容)
SUBA  GR0,A,GR1      GR0←GR0-(アドレスA+GR1の内容)
SUBA  GR0,=5         リテラルを用いての減算。GR0←GR0-5
```

7.3 論理加算(ADDL)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*ADDL*	*r1,r2*
[ラベル]	*ADDL*	*r,adr[,x]*

- レジスタ間、あるいはレジスタと指定したアドレスの主記憶装置の内容の加算をします。演算は符号無しです(正の整数、2の補数ではなく16ビットの正の整数)。フラグレジスタは変化します(表2参照)。

- 以下のような使い方があります。

```
ADDL   GR0,GR1        レジスタ同士の加算。GR0←GR0+GR1
ADDL   GR0,A          GR0←GR0+(アドレスAの内容)
ADDL   GR0,A,GR1      GR0←GR0+(アドレスA+GR1の内容)
ADDL   GR0,=5         リテラルを用いての加算。GR0←GR0+5
```

7.4 論理減算(SUBL)

書式

ラベル欄	命令コード欄	オペランド欄
[ラベル]	*SUBL*	*r1,r2*
[ラベル]	*SUBL*	*r,adr[,x]*

- レジスタ間、あるいはレジスタと指定したアドレスの主記憶装置の内容の減算をします。演算は符号無しです(正の整数、2の補数ではなく16ビットの正の整数)。フラグレジスタは変化します(表2参照)。

- 以下のような使い方があります。符号無しに注意。

```
SUBL   GR0,GR1        レジスタ同士の減算。GR0←GR0-GR1
SUBL   GR0,A          GR0←GR0-(アドレスAの内容)
SUBL   GR0,A,GR1      GR0←GR0-(アドレスA+GR1の内容)
SUBL   GR0,=5         リテラルを用いての減算。GR0←GR0-5
```

8 プログラム

- これまでの学習を総合して、以下のようなプログラムは書けるはずです。

  - メインメモリーに285と353を記憶させる。
  - 285-353の算術減算をさせて、その結果をメインメモリーに格納する。

```
PGM    START
       LD          GR1,A
       SUBA        GR1,B
       ST          GR1,C
       RET
A      DC          285
B      DC          353
C      DS          1
       END
```