まとめ (CASL IIの命令)

山本昌志*

2004年11月15日

1　前回の復習と本日の学習

1.1　復習

前回は、少し駆け足だったけれども、以下の命令を学習した。

- 残りの機械語命令

  PUSH　スタック領域にデータを記憶させる。記憶させるデータは、*adr*[,x]で指定されるアドレスである。
  POP　スタック領域のデータを取りだして、指定のレジスターに格納する。
  CALL　サブルーチンの呼び出し命令である。
  RET　サブルーチンからその呼び出し元に戻る命令である。
  SVC　OSの機能を呼び出すときに使う。マクロ命令で使われており、CASL IIのプログラムで出会うことはほとんどない。
  NOP　何もしない命令である。ラベル名をつけて、そこへの分岐に使われることが多い。

- マクロ命令

  IN　あらかじめ割り当てられれた入力装置から、主記憶装置へ文字を入力する。通常のシミュレーターでは、入力装置としてキーボードが割り当てられている。
  OUT　あらかじめ割り当てられれた出力装置へ、主記憶装置に格納された文字列を出力する。。通常のシミュレーターでは、出力装置としてディスプレイが割り当てられている。
  RPUSH　汎用レジスターの内容をGR1,GR2,GR3,$\cdots$,GR7の順にスタック領域に格納する。
  RPOP　スタック領域の内容を順次取り出し、GR7,GR6,GR5,$\cdots$,GR1の順に汎用レジスターに格納する。順にスタック領域に格納する。

*国立秋田工業高等専門学校　電気工学科

1.2 本日の学習内容

前回までに、全ての CASL II の命令について、説明した。今後は、その命令を使って、実際のプログラムの作成テクニックを学習することになるが、その前に全ての命令の復習を行う。

どのような命令が有ったか思い出すことも重要であるが、コンピューターはこのような命令を実行するハードウェアーから出来上がっていることを理解して欲しい。命令は単純であるが、これを組み合わせて複雑なプログラムが出来上がる。COMET II のハードウェアーは単純なため、CASL II の命令もそれほど多くない。しかし、実際のコンピューターも似ており、基本は同じである。

コンピューターは思ったより単純なのである。CASL II の場合、機械語命令は 28 個しかない。この 28 個の命令通りにレジスターやメモリーを操作するハードウェアーが作れば、COMET II と言うコンピューターが出来上がる。ハードウェアーであるコンピューターは論理回路で作られている。これは 2 年生で学習したので、諸君はコンピューターを作ることができるのである。

2 命令の種類

CASL II の命令は、3 種類ある。プログラムを書く場合、これらがどのようになっているか、理解する必要がある。

2.1 アセンブラ命令

教科書の P.28〜P.35 で説明している非実行文と書かれているものである。アセンブラーという変換プログラムに対して、いろいろな指示を行う命令である。COMET II の CPU の動作の指示は行わない。したがって、この命令は機械語に変換されて特定のビットパターン (1 と 0 の組み合わせ) に変換されることはない。

CASL II には、次の 4 個のアセンブラ命令がある。

START	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
END	プログラムの終わりを明示
DC	定数を定義
DS	領域を確保

ただし、DC 命令は、それに引き続く値にビットパターンに変換される。DS 命令はビットパターンに変換されないが、必要な領域を確保する。この 2 つは、FORTRAN の変数宣言と同じような働きをする。実際のプログラムでは、データの値を定義することに使われる。

2.2 機械語命令

教科書の P.40〜P.82 で説明している。この命令は、COMET II の CPU の動作の指示を行う。そのため、この命令に対応した論理回路が、CPU の中に組み込まれている。これら命令は、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、そのパターンに従い、論理回路が動作する。

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ている。ビットパターンへの変換は、以前の授業で説明したハンドアセンブラーと同じことをする。

CASL II には、以下の 28 個の機械語命令 (教科書の P.203) があり、そんなに多くない。

LD, ST, LAD	データの移動
ADDA, SUBA, ADDL, SUBL	加算・減算
AND, OR, XOR	論理演算
SLA, SRA, SLL, SRL	シフト演算
CPA, CPL	比較演算
JPL, JMI, JNZ, JZE, JOV, JUMP	分岐処理
PUSH, POP	スタック操作
CALL, RET	サブルーチンへの移動と戻り
SVC, NOP	その他

2.3 マクロ命令

教科書の P.83〜P.86 で説明している。マクロ命令とは、特定の機能を果たす、いくつかの機械語命令の集まりに名前を付けたものである。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できる。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、便利である。サブルーチンみたいになっている。

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る。

CASL II には、以下の 4 個のマクロ命令がある (教科書の P.203)。

IN	入力装置 (キーボード) から、文字データを読み込む
OUT	出力装置 (ディスプレイ) に、文字データを書き込む
RPUSH	汎用レジスターの内容を、GR1, GR2, ···,GR7 の順でスタックに格納
RPOP	スタックの内容を GR7, GR6, ··· ,GR0 の順で汎用レジスターに格納

3 CASL II のプログラムの書き方

3.1 コーディングの約束

コーディングとはプログラムを記述する作業のことである。ここでは、CASL II のプログラムの書き方の約束を示す。

CASL II のプログラムの 1 行は、ラベル欄と命令コード欄、オペランド欄、注釈欄と機能毎に欄が分かれている。具体的には、図 1 のようにである。

各欄は FORTRAN のように桁数で分けられているわけではない。機能別の欄の区切りは、1 個以上の空白である。したがって、

- ラベル欄の先頭の空白は許されない。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る。

- 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ない。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る。できるだけ、そろえたほうが良い。

となる。


```
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
```

図 1: CASL II のプログラムの書き方

3.2 機能別の各欄の説明

3.2.1 ラベル欄

ラベルは、その記述する位置から FORTRAN の文番号にも似ている。あるいは、いままでの例でもわかるように、変数名の役割を果たしている。実際、プログラムでは、FORTRAN の文番号や変数名のような使われ方をする。実際には、CASL II では、それはアドレスを表す。そのアドレスは、それ引き続く命令に従い、次のように決まっている。

- 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る。FORTRAN の `GO TO` 文でその文番号に制御が移るのと同じである。
- `DC` 命令 の場合、ラベルは定数が格納されている領域 の先頭アドレスを示す。使い方は FORTRAN の変数名に似ているが、実態はアドレスである。
- `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表す。C 言語の配列の宣言と同じである。
- `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す。

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

教科書にも書かれている通り、ラベルの記述の約束は

- 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 文字以内で記述する。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) から始める。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る。
- 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ない。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るためである。

である。

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) する。また、プログラムの命令も主記憶装置に格納されている。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る。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らない。

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する。ありがたいものである。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ない。これは、コンパイラーが変数名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ているのである。

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らない。そんなに難しくない。

3.2.2 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、マクロ命令を書く。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ある。

- ラベルの後に 1 個以上の空白の後、命令コードを書く。
- 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、記述する。

3.2.3 オペランド 欄

この欄には、命令のオペランドを記述する。オペランド (operand：被演算子) とは、命令の対象となるアドレスやレジスタ、データのことである。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、文字、整数がオペランドとなる。その記述方法は、教科書に書かれているように、以下の通りである。

- 命令コードの後に 1 個以上の空白の後、オペランドを書く。
- 複数のオペランドは、カンマ","で区切って、連続して書く。途中に空白は入れない。

3.2.4 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る。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ない。プログラムの内容を分かりやすくするために書くことが多い。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る。

- 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、行全体が注釈となる。
- オペランドの後に 1 個以上の空白があれば、そこ以降も注釈となる。

4 命令一覧

4.1 アセンブラ命令

機能	書式	動作内容	フラグレジスタの変化
プログラム開始	START [実行開始番地]	プログラムの開始を示す。プログラムの最初に、必ず書かなくてはならない。	
プログラム終了	END	プログラムの終わりを示す。ラベルは使えない。プログラムの最後に、必ず書かなくてはならない。	
定数格納	DC n	10 進定数をラベルのアドレスに格納	
	DC #h	16 進定数をラベルのアドレスに格納	
	DC ’文字列’	文字列をラベルのアドレスから格納	
	DC ラベル名	ラベル名が示すアドレスを格納	
領域の確保	DS n	ラベル名で示すアドレスから n 語領域を確保	

注意

- アセンブラ命令ではフラグレジスタの値はセットされることはない。これは、アセンブラ命令はプログラム実行には動作しないためである。

4.2 機械語命令

機能	書式	動作内容	フラグレジスタの変化
ロード	LD r1,r2	レジスタ r2 の値をレジスタ r1 にコピー	コピーされた値に従い以下のようになる。 OF 0:常にゼロが設定される SF 1:負の時 (第 15 ビットが 1) 0:正の時 (第 15 ビットが 0) ZF 1:ゼロの時 (全てのビットが 0) 0:ゼロ以外
	LD r,adr[,x]	アドレス adr[,x] の主記憶の内容をレジスタ r にコピー	
ストア	ST r,adr[,x]	レジスタ r の内容を主記憶装置のアドレス adr[,x] にコピーする	変化なし
ロードアドレス	LAD r,adr[,x]	主記憶装置のアドレス値 adr[,x] をレジスタ r にコピーする。	変化なし
算術加算	ADDA r1,r2	レジスタ r1 と r2 の符号付き加算 r1←r1+r2	演算結果の値に従い以下のようになる。 OF 1:結果が-32768〜32767 の範囲外 0:範囲内 SF 1:負 (第 15 ビットが 1) 0:正 (第 15 ビットが 0) ZF 1:ゼロ (全てのビットが 0) 0:ゼロ以外
	ADDA r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容を符号付加算 r←r+adr[,x] の内容	
算術減算	SUBA r1,r2	レジスタ r1 と r2 の符号付き減算 r1←r1-r2	
	SUBA r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容を符号付減算 r←r-adr[,x] の内容	
論理加算	ADDL r1,r2	レジスタ r1 と r2 の符号無し加算 r1←r1+r2	演算結果の値に従い以下のようになる。 OF 1:結果が 0〜65535 の範囲外 0:範囲内 SF 1:第 15 ビットが 1) のとき 0:第 15 ビットが 0) のとき ZF 1:ゼロ (全てのビットが 0) 0:ゼロ以外
	ADDL r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容を符号無し加算 r←r+adr[,x] の内容	
論理減算	SUBL r1,r2	レジスタ r1 と r2 の符号無し減算 r1←r1-r2	
	SUBL r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容を符号無し減算 r←r-adr[,x] の内容	
論理積	AND r1,r2	レジスタ r1 と r2 のビット毎の論理積を計算。結果は r1 に格納。	演算結果の値に従い以下のようになる。 OF 0:常にゼロが設定される。 SF 1:第 15 ビットが 1) のとき 0:第 15 ビットが 0) のとき ZF 1:ゼロ (全てのビットが 0) 0:ゼロ以外
	AND r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容のビット毎の論理積を計算。結果は r1 に格納。	
論理和	OR r1,r2	レジスタ r1 と r2 のビット毎の論理和を計算。結果は r1 に格納。	
	OR r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容のビット毎の論理和を計算。結果は r1 に格納。	
排他的論理和	XOR r1,r2	レジスタ r1 と r2 のビット毎の排他的論理和を計算。結果は r1 に格納。	
	XOR r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) の内容のビット毎の排他的論理和を計算。結果は r1 に格納。	
算術比較	CPA r1,r2	レジスタ r1 と r2 を符号付き整数として比較を行う。比較の結果は、FR に設定。	2 つの整数の比較 (以下の演算) を行う。 r1-r2 r-adr[,x] の内容 OF 0:常にゼロが設定される。 SF 1:負 (第 15 ビットが 1) のとき) 0:正 (第 15 ビットが 0) のとき) ZF 1:等しい (全てのビットが 0) 0:等しくない
	CPA r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) を符号付き整数として比較。比較の結果は、FR に設定。	
論理比較	CPL r1,r2	レジスタ r1 と r2 を符号無し整数として比較。比較の結果は、FR に設定。	
	CPL r,adr[,x]	レジスタ r と主記憶装置 (アドレス adr[,x]) を符号無し整数として比較。比較の結果は、FR に設定。	

機能	書式	動作内容	フラグレジスタの変化
算術左シフト	SLA r,adr[,x]	レジスタ r の内容を符号ビットを除き、adr[,x] の番地分、各ビットを左へシフト。空いたビットには 0 が入る。	OF :最後に送り出されたビットの値 SF 1:負の時 (第 15 ビットが 1) 0:正の時 (第 15 ビットが 0) ZF 1:ゼロの時 (全てのビットが 0) 0:ゼロ以外
算術右シフト	SRA r,adr[,x]	レジスタ r の内容を符号ビットを除き、adr[,x] の番地分、各ビットを右へシフト。空いたビットには符号ビットと同じ値が入る。	
論理左シフト	SLL r,adr[,x]	レジスタ r の内容を、adr[,x] の番地分、各ビットを左へシフト。空いたビットには 0 が入る。	
論理右シフト	SRL r,adr[,x]	レジスタ r の内容を、adr[,x] の番地分、各ビットを左へシフト。空いたビットには 0 が入る。	
正分岐	JPL adr[,x]	フラグレジスタの SF と ZF の両方が 0 の時 (比較の結果、正)、adr[,x] のアドレスへ分岐 (実行が移動) する。	変化無し
負分岐	JMI adr[,x]	フラグレジスタの SF が 1 の時 (比較の結果、負)、adr[,x] のアドレスへ分岐 (実行が移動) する。	
非零分岐	JNZ adr[,x]	フラグレジスタの ZF が 0 の時 (比較の結果、等しくない)、adr[,x] のアドレスへ分岐 (実行が移動) する。	
零分岐	JZE adr[,x]	フラグレジスタの ZF が 1 の時 (比較の結果、等しい)、adr[,x] のアドレスへ分岐 (実行が移動) する。	
オーバーフロー分岐	JOV adr[,x]	フラグレジスタの OF が 1 の時 (オーバーフロー)、adr[,x] のアドレスへ分岐 (実行が移動) する。	
無条件分岐	JUMP adr[,x]	無条件に、adr[,x] のアドレスへ分岐 (実行が移動) する。	
プッシュ	PUSH adr[,x]	スタック領域に、adr[,x] のアドレスを格納する。	変化無し
ポップ	POP r	スタック領域からデータを取りだし、レジスタ r に格納	
コール	CALL adr[,x]	サブルーチンを呼び出す。adr[,x] に実行が移る。	変化無し
リターン	RET	サブルーチンから呼び出し元のルーチンへ実行が移る。	
スーパーバイザーコール	SVC adr[,x]	OS の機能を呼び出す。マクロ命令の IN や OUT で使われている。	不定。OS に依存する。
ノーオペレーション	NOP	なにも実行されない命令。	変化しない。

4.3 マクロ命令

機能	書式	動作内容	フラグレジスタの変化
入力命令	IN ラベル 1, ラベル 2	入力領域 (ラベル 1) に入力装置から文字データを入れる。入力文字長は、ラベル 2 に入る。	不定。OS に依存
出力命令	OUT ラベル 1, ラベル 2	出力領域 (ラベル 1) の文字データ、ラベル 2 が示す数だけを出力装置に送る。	不定。OS に依存
レジスターの待避	RPUSH	汎用レジスター内容を、GR1→GR7 の順序でスタック領域に格納。	不定。OS に依存
レジスターの復元	RPOP	スタック領域の内容を、GR7→GR1 の順序で汎用レジスタに格納。	不定。OS に依存