CASL IIの命令の種類とプログラムの書き方

山本昌志*

2005年11月25日

1　先週の復習と本日の学習

1.1　先週の復習

1.1.1　レジスター

COMET IIは主にCPUとメモリーから構成され，CPUはデータの処理を行い，メモリーはプログラム(命令とデータ)を格納する．CPUの中にも小さいながらも記憶装置があり，それをレジスターと言う．COMET IIには，次の4つのレジスターがある．

汎用レジスター(GR)　16ビットの記憶容量があり，8個(GR0〜GR7)用意されている．主に，データの処理(計算)に用いられ，プログラマーがもっとも多用するレジスターである．

フラグレジスター(FR)　1ビットのレジスターが3個用意(OF,SF,ZF)[1]されている．計算結果の状態を示す．OFは計算結果がメモリーに収まらなかったときに1(ビットが立つと言う)になる．SFは計算結果が負(第15ビットが1)になったとき，ZFは計算結果がゼロ(全てのビットが0)のとき1になる．

プログラムレジスター(PR)　コンピューターが次に実行する命令の先頭アドレスが格納されている．アドレスを格納するので記憶容量は16ビットである．CPUに中に1つある．

スタックポインター(SP)　スタックというデータ構造を使う場合，その最上段のアドレスを格納する．これについては，後の講義で詳しく説明する．

実際のプログラムを作成する場合，指標レジスターと言うものもつかわれ，それは汎用レジスターのGR1〜GR7が代用される．基準のアドレスからある値シフトして目的のアドレスを表すときの，オフセット値が指標レジスターに格納される．

ともあれ，プログラマーが覚えて置かなくてはならないのは，汎用レジスターとフラグレジスター，そして指標レジスターの使い方である．

*国立秋田工業高等専門学校　電気工学科

[1]それぞれ，Overflow Flag, Sign Flag, Zero Flag の略.

1.1.2 命令をマシン語にする方法

教科書の命令語の構成 (p.213) を使って，命令をマシン語 (0 と 1 のビットパターン，または 16 進数) に変換する方法を学習した．これは，課題が出来ていれば大丈夫．

1.2 本日の学習内容

本日から，本格的にアセンブラ言語 CASL-II の書き方 (文法およびアルゴリズム) の学習を行う．CASL II という特定のアセンブラ言語について学習することになるが，他のアセンブラ言語でも似ている．これをちゃんと習得しておけば，今後，他のアセンブラ言語を使うことになっても，短期間に理解できるであろう．

そこで，本日は，CASL-II の命令の種類とその書き方を学習する．本日のゴールは，

- ソースプログラムを機械語に翻訳するアセンブラーの仕事が分かる．
- CASL II の命令の種類が理解できる．
- CASL II のプログラムの書き方が理解できる．

である．

2 アセンブラーの仕事

本日の授業のテーマである「CASL II の命令の種類とプログラムの書き方」を理解するためには，アセンブラーの仕事を理解する必要がある．そのためには，ある程度プログラムの作成順序が分からなくてはなくてはならない．図 1 に, アセンブラ言語でのプログラムの作成手順を示す．作業の中心は，

1. テキストエディターにより，アセンブラ言語でソースプログラムを記述し，ソースファイルを作成する．
2. アセンブラーにより，ソースファイルを機械語の実行ファイルに変換する．

である．

> アセンブラーの役目は，人間が分かるアセンブラ言語で書かれたソースファイルをマシン語に変換することにより，実行ファイルを作る．

アセンブラーとはこのような仕事をするプログラムである．その様子を図 2 に示す[2]．何のことはない，先週の諸君の課題でアセンブラ言語をマシン語に直すのと同じで，アセンブラーはそれをやっているのである．

ただし，このプログラムは，**メモリーの**#0020 **からロード された場合**としているので注意が必要である．

[2]実際のアセンブラ言語ではもう少し複雑ことを行い，実行プログラムが作られる．

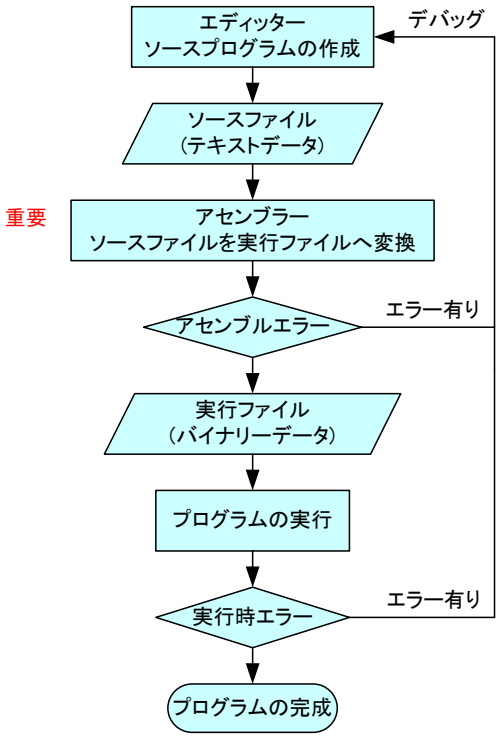

図 1: プログラム作成のフローチャート

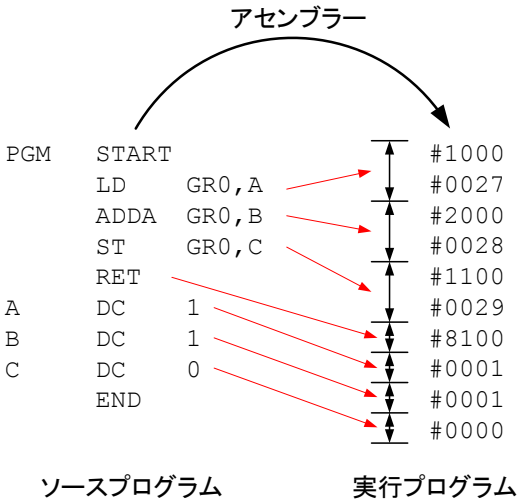

図 2: アセンブラ言語からマシン語への変換

3 CASL IIの命令の種類

CASL II の命令を大きき分けると，アセンブラ命令と機械語命令，マクロ命令の 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．命令の具体的な内容は，次回以降の授業で学習すが，ここでは簡単に概要を述べる．

3.1 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．プログラム実行時には，COMET II の CPU の動作の指示は行う命令ではない．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．

3.2 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 CCPU 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，先々週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 個の機械語命令がある (教科書の P.203)．

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

3.3 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．

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，$\cdots$,GR7 の順でスタックに格納
RPOP	スタックの内容を GR7，GR6，$\cdots$,GR1 の順で汎用レジスターに格納

4 プログラム例 (加算演算)

それでは，命令の種類と，それがどのようにメモリーに格納される調べる．そのためには，実際のプログラムで，それを見るのが良いであろう．次のような CASL II のプログラムを考えることにする．これで，CASL II の 3 種類の命令の違いを理解する．

- 加算をするプログラムである．内容は，
  - 加算をする 3 と 5 をメモリーに格納しておきます.
  - それを引き出して，加算を行います.
  - 加算の結果を，メモリーに戻します.
  - そして，最後に”END”と出力装置に表示します.

  である．

これを実現する CASL II のプログラムは，図 3 のようになる．合わせて，フローチャートも示しているので，内容を理解すること．

実行文ではないため，フローチャートには書かなかったが，命令コードの DC と DS には，

A DC 3	アドレス A に数値の 3 を格納
D DC 'END'	アドレス D に文字'E' を格納
	続いて，アドレス D+1 に文字'N' を格納
	最後に，アドレス D+2 に文字'D' を格納
C DS 1	アドレス C を先頭に，1 ワード分のメモリー領域を確保

の役割がある．これは，アセンブラーがメモリーの中身を決めたり，確保するために必要である．

それでは，この CASL II がアセンブラーでどのように機械語に変換されるか見る．シミュレーター WCASL-II[3]で変換すると，図 4 のようなマシン語になる．以下のことが重要である．

- アセンブラ命令は，マシン語に変換されていない．ただし，DC はデータに変換され，DS はメモリーが予約され適当な値が格納される．
- 機械語命令は，1 対 1 の対応でマシン語に変換される．対応については，前回のハンドアセンブルで，学習した通りである．
- マクロ命令は，複数のマシン語に変換される．変換されるマシン語は，OS やアセンブラーに依存する．したがって，マクロ命令のマシン語への変換については，ここでの学習の範囲外である．この変換を考えるためには，OS とアセンブラーの設計が必要である．

[3]COMET II を Windows 上で擬似的に動作させるプログラムである．

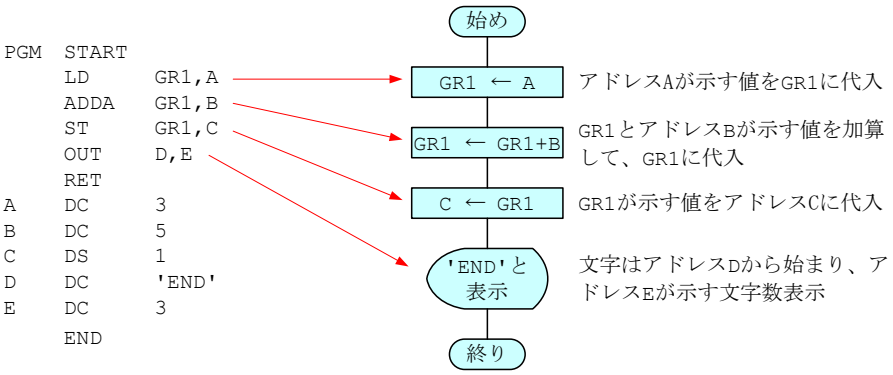

図 3: アセンブラ命令，機械語命令，マクロ命令があるプログラム例とフローチャート

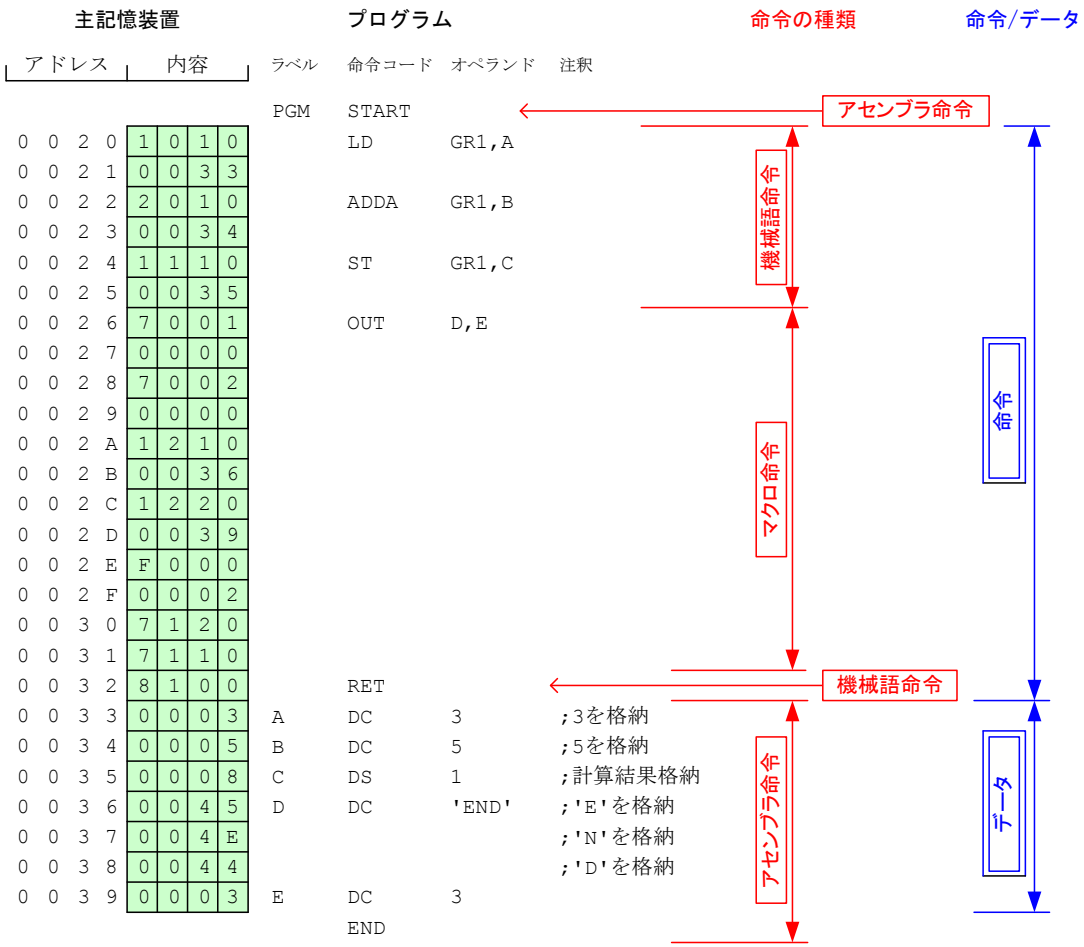

図 4: メモリーの内容と命令の種類

5　CASL IIのプログラムの書き方

5.1　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．本節では，CASL II のプログラムの書き方の約束を示す．FORTRAN でもプログラムの書き方があったように，CASL II でも約束がある．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) と役割分担がある．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る．例えば，先ほどのプログラムを例に取ると，図 5 のようになる．

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．

- 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．

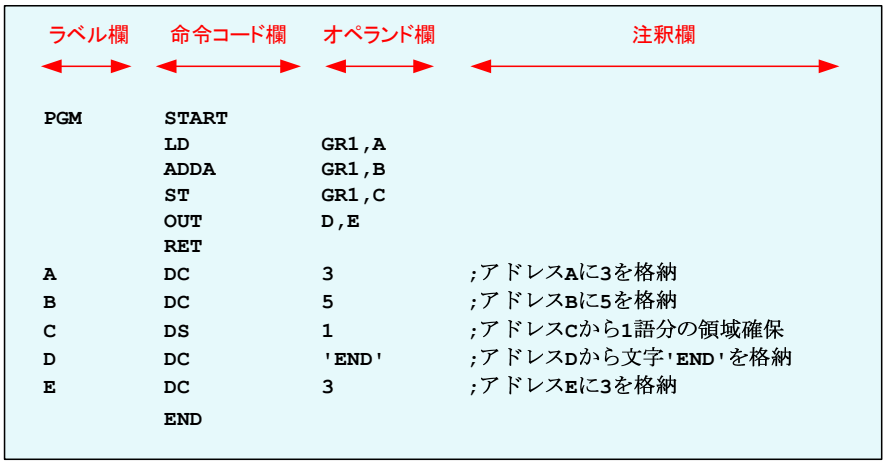

図 5: CASL II のプログラムの書き方

5.2　機能別の各欄の説明

5.2.1　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．実際のところ，マシン語の奥底では，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．
- `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以下の通りである．

- 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．

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．

5.2.2 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．

5.2.3 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．

5.2.4 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[4]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]コメントアウトすると言う