COMET IIのハードウェアーとマシン語

山本昌志*

2005年1月18日

1　これまでの復習と本日の内容

1.1　先週の復習

先週までは，CASL IIで取り扱うデータ，整数と文字のメモリーの格納の仕方を学習した．CASL IIで取り扱うデータは，整数と文字だけなので，これでデータの取り扱いは理解したはずである．実際のコンピューターはもっと複雑なデータを扱っているが，基本的にはこれまで学習した整数や文字の扱いと同じである．

コンピューターの内部では全ての情報をコード化して取り扱っていることを理解しなくてはならない．コンピューターでは，ビットパターンで全ての情報はコード化される．驚くことに，コンピューターが処理する対象は，ビットパターンのみなのである．複雑な情報処理をしているように見えるが，ビットパターンを変化させているだけである．

このビットパターンは，2進数と1対1の対応する．すなわち，整数で表現できることになる．コンピューターに取って都合の良いビットパターンは人間が紙を使って表現するのは不便なので，16進数の整数で表されることが多い．

これで，諸君は世界中のコンピューターでの情報の取り扱い方法を学んだことになる．情報をコード化する方法は，仕様書に書かれているので，それを見れば分かる．音や絵などさらに複雑な情報も，同じである．

1.2　本日の内容

本日は，教科書のp.15〜24について学習する．主な内容は，

- レジスターを中心としたCOMET IIのCPUについて
- 命令をメモリーに格納する方法について

である．

*国立秋田工業高等専門学校　電気工学科

2　レジスター

2.1　レジスターとは何か

コンピューターは，おもに計算をする CPU とプログラム (命令とデータ) を格納するメモリから構成される．CPU とメモリーの間で，命令やデータをやりとりして，プログラムを実行する．

計算を司る CPU の中にも，容量は小さいながらもレジスターと呼ばれる記憶装置がある．計算をするためにはある程度の記憶が必要である．教科書を読み理解するときの人間の動作を考えると分かりやすい．この場合，教科書がメモリーで脳が CPU に対応する．内容を理解するためには単語を一時的に記憶しているはずで，その記憶する場所がレジスターと考えればよい．

ここで，第一回の講義で述べたコンピューターの原理を示すチューリングマシーン (図 1) について，思い出して欲しい．明らかに，書き換え可能なテープがメモリー，オートマトンが CPU を表している．オートマトンの中にあり，内部状態を表すものがレジスターである．

> CPU 内にある小さな記憶装置がレジスターである．ここには CPU で計算するデータや計算結果を一時的に記憶する．

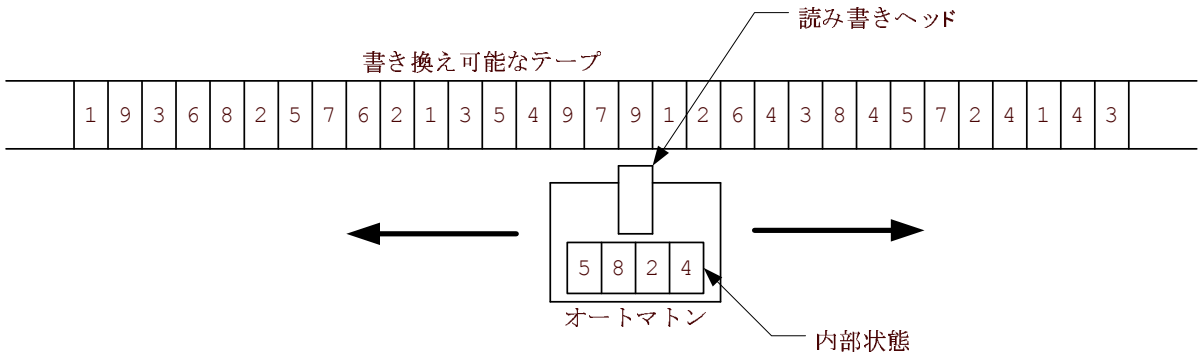

図 1: チューリング機械

2.2　コンピューターはどのようにプログラムを実行するか

以前述べたように，コンピューターのプログラムはデータと命令から構成される．この命令とデータは，実行時に主記憶装置 (メインメモリ) に格納される[1]．CPU と主記憶装置は，図 2 のような関係になっている．CPU は主記憶装置のアドレスを指定することにより，主記憶装置に格納されているデータを取り出す．そして，それはレジスタに記憶し，その中身に従い，処理する．処理された結果ももちろん，レジスタに記憶する．レジスタの中身を主記憶装置に戻すことにより，データの加工が完了する．

ここで，諸君が使っている実際のコンピューターでのプログラムの動作順序を示しておく．

1. 補助記憶装置 (ハードディスク等) からプログラムがメインメモリーにロードされる．この指令は，

[1] この格納の動作をロードと言う．

Operating System(OS) が出す.

2. メインメモリーに格納されたプログラムの指示に従い，CPU が動作する．その動作は，

   (a) CPU がメインメモリーから命令を取り出す．命令を取り出すアドレスは，CPU のプログラムレジスタに書かれている．

   (b) 取り出した命令は，CPU 内の命令デコーダーが内容を解析する．

   (c) 解析された命令は，論理演算装置 (ALU:arithematic logic unit) が処理 (計算) する．

   (d) 各種のレジスターに処理の結果が格納される．

   (e) プログラムレジスタの値を再設定する．

   (f) 以上の動作をプログラム終了まで繰り返す.

   である．

注意　CASL II の学習で 1 の補助記憶装置からメインメモリーへのロードは学習の範囲外である．以降の講義では，2 の動作について学習する．

このプログラムの実行方法からも，CPU の中にもデータを記憶する装置 (レジスター) が必要な理由が理解できるであろう．ところで，ここでプログラムレジスタという訳の分からないもが使われていることに気づく．これも教科書の読むときの動作を考えると分かりやすい．プログラムレジスタは，脳の中に記憶される教科書のページ番号だと思えばよい．どこまで読んだか憶えていて，そのページを読んだら次のページをめくるのである．

わざわざ CPU 内に作らなくても，メインメモリーの一部を使えば良いのでは，と考える人もいるであろう．それでもコンピューターは可能であるが，今よりも複雑になるだろう．また，CPU とメモリーのデータの間でのデータの交換回数が極端に増加して，動作が遅くなるに違いない．

レジスタもデータなどを蓄えるので，メインメモリー同様，記憶装置の一種である．しかし，以下のような違いがある．

- 主記憶装置
  - CPU とは独立である．
  - プログラム (命令とデータ) を格納する．
  - 記憶容量が大きい．COMET II の場合，1 ワード (16 ビット) のデータを 65536 個，記憶できる．
  - 番地を指定して目的のデータにアクセスする．
- レジスタ
  - CPU の構成部品のひとつである．
  - データを処理するための一時的な記憶場所である．また，処理結果も記憶する．
  - 記憶容量が小さい．COMET II のようなものだと，1 ワード (16 ビット) のデータを 20〜30 個程度で済むであろう．

– 名前を指定して，目的のデータにアクセスする．
– 現実の装置の場合，CPU のデータのアクセススピードは，レジスターの方がはるかに早い (C 言語ではレジスタを使ったプログラムができる)．

要するに主記憶装置は，いろいろなデータ (命令もデータの一種と考える) を蓄えるファイルキャビネットのようなものである．一方，レジスタは，実際に CPU がデータを加工するときに一時的に記憶する場所と考えれば良い．レジスターにある加工されたデータをメモリーにコピーすることにより，データ処理が完成するのである．

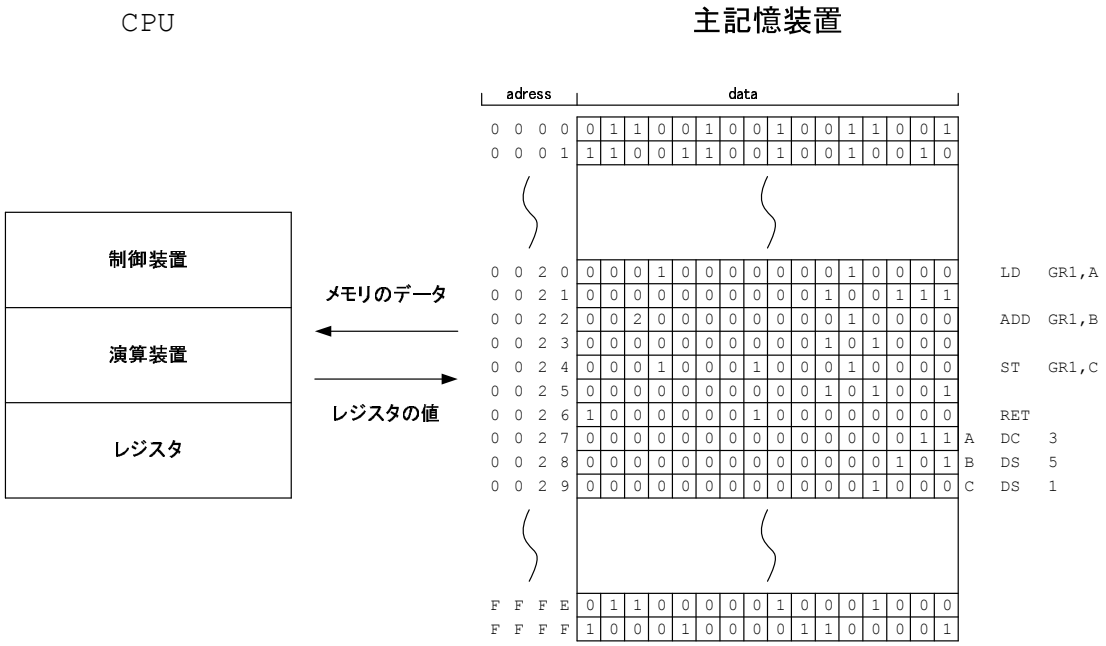

図 2: CPU と主記憶装置の関係

3 COMET II のレジスタ

図 2 を見て分かるように，COMET II は CPU と主記憶装置 (メインメモリー) から構成されている．大雑把に言うと，CPU は制御装置と演算装置，レジスターから構成される．これらのうち，プログラマが注意を払うべきものは，

- 主記憶装置
- レジスタ

だけである．今後アセンブラでプログラムを書くと，制御装置や演算装置について，あまり注意を払う必要が無いことが分かるだろう．

主記憶装置は，プログラム (命令とデータ) を記憶する装置でここでは特に説明しない．アセンブラのプログラムを書くと自然に理解できるであろう．レジスターについては，プログラムを記述する前にある程度理解する必要がある．COMET II のレジスタを表 1 にまとめる．以降，それぞれのレジスタについて，説明する．

表 1: CASL II のレジスタ

記号	語源	日本語	機能
GR	General Register	汎用レジスタ	計算等に用いる．また GR1〜GR7 は指標レジスタとしても使われる．
SP	Stack Pointer	スタックポインタ	スタック領域の最上段のアドレスを保持する．
PR	Program Register	プログラムレジスタ	次に実行する命令のアドレスを保持する
FR	Flag Register	フラグレジスタ	演算結果の状態を保持する

3.1 汎用レジスタ

これは，算術や論理，比較，シフト演算を実行するときに使う．GR0〜GR7 までの 8 個用意されている．あとは，教科書の通り．

- 汎用レジスタは，8 個用意されている．
- 汎用レジスタは，16 ビットである．メインメモリーのデータやアドレスのビット数とおなじである．

3.2 プログラムレジスタ

プログラムカウンターと呼ばれることもある．このレジスタの値は，プルグラムが次に実行する命令語の先頭番地を表す．したがって，

- CPU には必ず，1 個のプログラムレジスタが必要．
- プログラムレジスタは 16 ビットで，アドレスのビット数と同じ．

となる．

プログラムは，実行前に主記憶装置 (メインメモリー) に格納されている．プログラムレジスタ PR の値によって，プログラムを構成する命令を 1 つずつ取り出して，CPU は処理を行う．このような方式を逐次制御方式，あるいはプログラム内蔵方式 (stored program) と言う．

3.3 フラグレジスタ

Flag Regisuter のフラグとは，旗のことである．サッカーの試合では，プレーの状態により審判が旗を上げる．あれと同じ働きをする．コンピューターでは演算の結果により旗を上げる．

COMET II には，1 ビットのフラグレジスタが 3 個ある．演算結果によって，それらのレジスタの値がセットされる．セットされる内容は，教科書 P.18 の表 2.4 の通りである．主に，このレジスタは，実行順序を変更，分岐命令に使われる．

- フラグレジスタは，3 個あります．それで，計算結果の状態を表す．
- 各レジスタは旗の上げ下げなので，1 ビットである．

あとは教科書の説明通り．

3.4 スタックポインタ

メインメモリーの一部を CPU が専用の記憶領域として使うことがある．そのときのメインメモリーのアドレスを示す．したがって，

- 必要なスタックポインタは，1 個である．
- スタックポインタは 16 ビットで，アドレスのビット数と同じ．

となる．

これは，ここでは少し早すぎるので，実際に使うときに説明する．

3.5 指標レジスタ (index register)

これは，特殊なレジスタで，ハードウェアーは汎用レジスタが兼ねる．汎用レジスタのうち GR1〜GR7 を使う．GR0 を使わない理由，これはマシン語との関係で，後で述べる．

- 指標レジスタは，汎用レジスタの 7 個が使える．
- 指標レジスタは 16 ビットで，メインメモリーのデータのビット数と同じ．

教科書の図 2.5 の表現は分かりにくいので，具体例でその動作を示す．例えばクラス 40 人分の数学と英語と電子計算機のテストの点数がメモリに格納されているとする．それぞれの平均点を求める場合，指標レジスタを使うと便利である．このプログラムでは，それぞれの教科のクラスの合計点を計算するところが，重要である．指標レジスタを使う場合と使わない場合のフローチャートを図 3 に示す．

指標レジスタを使わないと，プログラムが大変である．一方，指標レジスタを使うと，基準点からのアドレスを加算して目的のデータにアクセスできる．加算する値を記憶するのが指標レジスターである．このように，基準アドレスに加算して目的のアドレス求める方法ををアドレス修飾と言う．

諸君は，これと同じプログラムテクニックを FORTRAN の授業で学んでいる．このアドレス修飾は，FORTRAN の配列と同じことを行っている．指標レジスタは，FORTRAN の配列の添え字の役割を果たし

ているのである．FORTRAN では分かりにくいのですが，C 言語の配列はまさにこれと同じことを行っている (実感できる)．

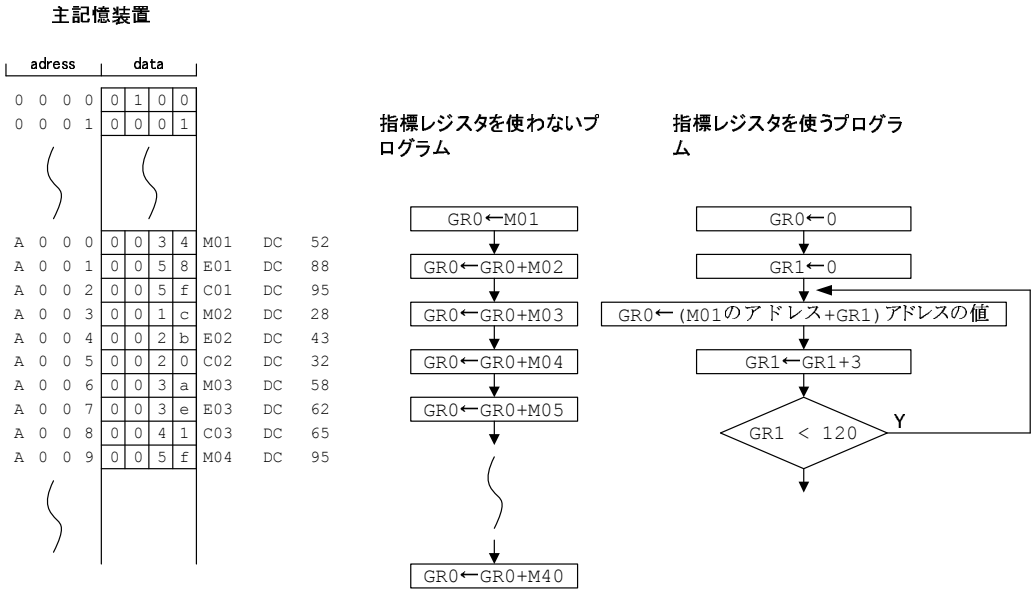

図 3: 指標レジスタを使った場合と使わない場合のプログラム．クラスの数学のテストの合計点を計算している．GR1 を指標レジスタとして使っている．

3.6 COMET II のハードウェアー

これまでの話から，COMET II のハードウェアーは図 4 のようになっていることが分かるだろう．

CPU の役割は、命令に従いデータの加工 (演算) を行うことである。その命令は、単純である。単純なことしかできないが、その処理速度は、信じられないくらい高速[2]である。CPU の回路は、2 年生の時に学習した論理回路 (組み合わせ回路、順序回路[3]) で構成されている。そこで、論理回路はどんなの入出力の論理でも可能であることを学習したはずである。それも、たった 3 つ (and, or,not) の回路の組み合わせで、できるから驚きである。このことから、どのような処理でも可能な回路ができることが分かる。

2 年生の時、or(論理和) と and(論理積)、not(否定) の回路がトランジスターで出来ることを学習した。ブール代数というソフトウェアーがトランジスターというハードウェアーで実現できるのである。コンピューターはまさにこれである。すなわち、トランジスターがビットパターン (命令とデータ) 応じた電圧を制御することにより、論理演算を行っている。要するに、いままで学習したビットパターンは、コンピューター内部では電圧のパターンに変換されて、トランジスターにより論理演算を行うのである。論理演算を行う装置は、以前学習した加算器のようなものである。

主記憶装置に格納されているデータは、16 ビットのただのビットパターンである。16 個の 0 と 1 の集ま

[2]例えば、Intel 社の Pentium の場合、3GHz で動作する。1 秒間に 30 億回、何かをするのである。
[3]順序回路は 4 年生で学習する。

りにすぎない。16 進数で書くと、4 桁の数字である。その 4 桁の 16 進数の数字が、整数や文字、あるいは命令を表したりする。CPU は、それらをどのように区別しているのであろうか?。そのからくりは?。それらを全く区別していないというのが答えである。ただ単に、プログラムレジスタ PR が示すアドレスの内容は命令と解釈するだけである。すごーく、単純である。後は、その命令に従い、主記憶装置の内容が命令になったり、整数になったり、文字になったりしているだけである。メモリーの内容を見ただけでは、それが示すものは、文字なのか整数なのか、命令なのかは分からないのである。

COMET II では、命令と処理すべきデータ (整数や文字) が同じところに、区別無く格納されている。世界中にある普通のコンピューターも同じようになっている[4]。このように、命令とデータ区別しないで、同じメモリーに格納するコンピューターをノイマンアーキテクチャーと言う。

通常のコンピューターは、とてつもないビットの操作をしていることが分かるであろう。それもひとつも間違えないで行うのは奇跡に等しいと思える。どのようにしているのであろうか?。

[4]命令とデータを区別して格納しているコンピューターをハーバードアーキテクチャーという．マイコンとして有名な PIC がハーバードアーキテクチャーである．

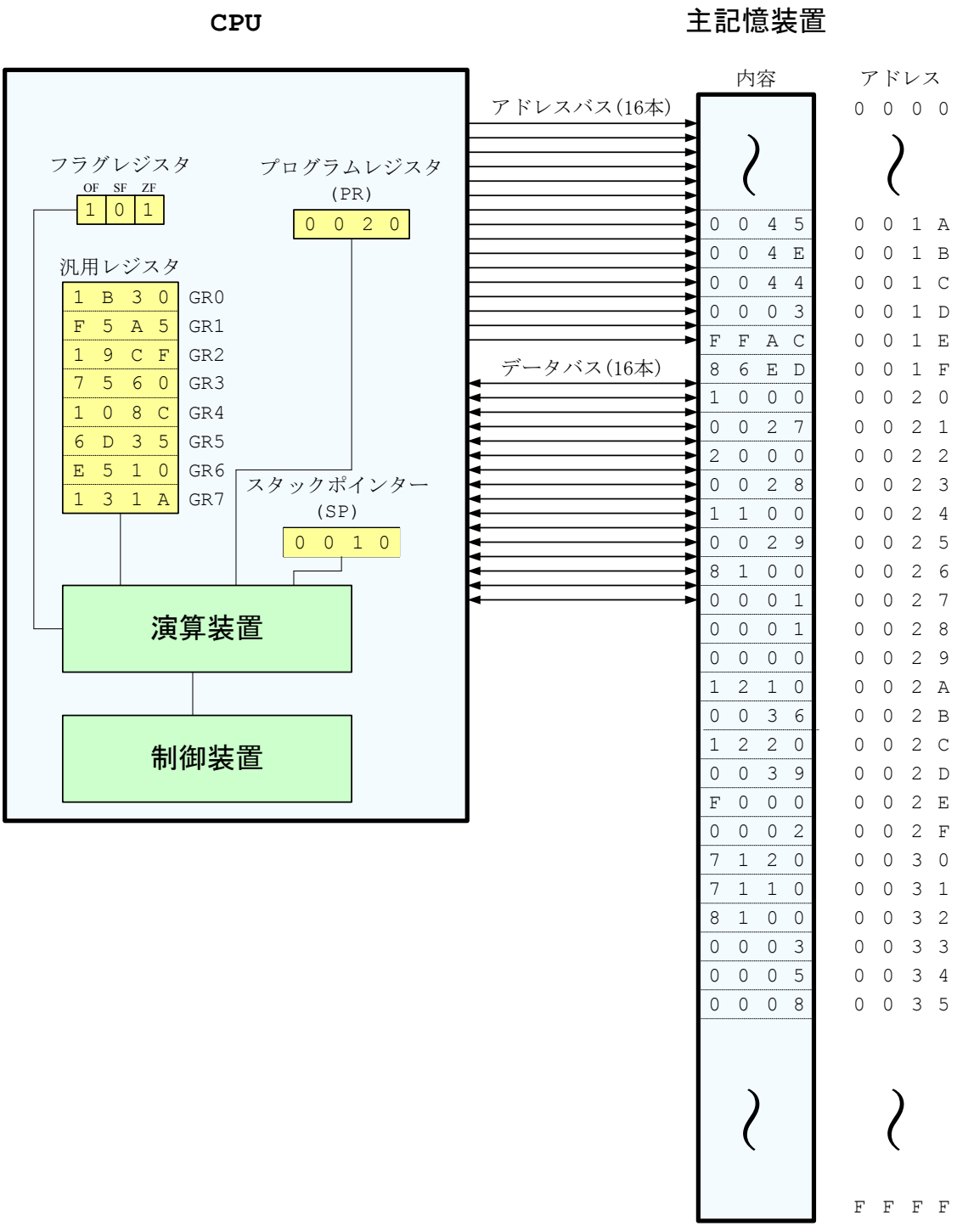

図 4: COMET II のハードウェアーとプログラムの状態

4　命令の表現方法

コンピューターのプログラムは，命令とデータから構成されるのは，以前に述べたとおりである．CASL II で扱うデータは整数と文字だけである．これらのデータをメモリーに格納する方法は，既に学習が済んでいる．ここでは，命令がどのようにメモリーに格納されるか学習する．

4.1　プログラム例

一般的なことは言わないで，実際の例で命令がデータに格納される様子を示す．リスト 1 に示したプログラムを用いて，メモリーに格納される様子を示す．その前に，このプログラムの動作を示した方がよいだろう．これは

- 3+5 を計算する

だけである．FORTRAN では `C=3+5`，C 言語では `c=3+5;` と書けばすむことを，アセンブラではこのようにいろいろ書かなくてはならない．理由は，後で説明する．

リスト 1: CASL II のプログラム例．3+5 を系算する

```
1  PGM      START
2           LD      GR1 , A
3           ADDA    GR1 , B
4           ST      GR1 , C
5           RET
6  A        DC      3
7  B        DC      5
8  C        DS      1
9           END
```

図 5 がこのプログラムの動作をフローチャートである．また，プログラムの各行の動作と命令/データの区別を表 2 に示しておく．

表 2: リスト 1 の各行の動作内容と命令/データの区別

行	文	機能	種類
1	`PGM START`	プログラムは，START から開始	
2	`LD GR1,A`	A の値をレジスタ GR1 に格納	命令
3	`ADDA GR1,B`	GR1 と B の値を加算して，GR1 に格納	命令
4	`ST GR1,C`	GR1 の値を C に格納	命令
5	`RET`	呼び出し元へ戻る	命令
6	`A DC 3`	値 $(3)_{10}$ を格納	データ
7	`B DC 5`	値 $(3)_{10}$ を格納	データ
8	`C DS 1`	1 ワード予約	データ
9	`END`	プログラムの終わりを示す	

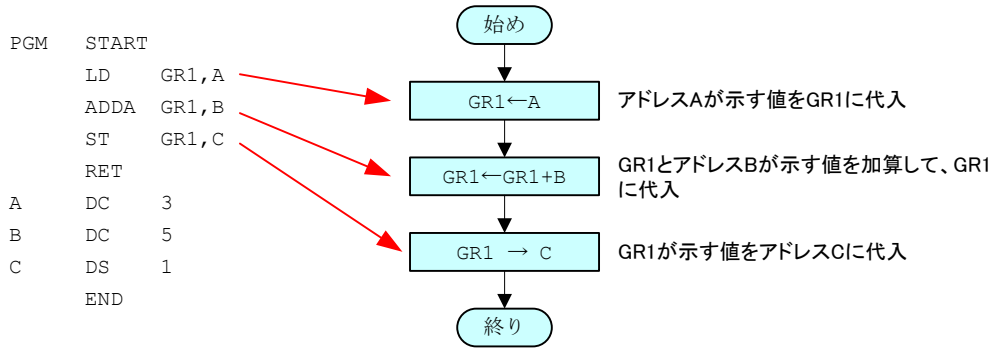

図 5: プログラムとフローチャート

4.2 アセンブラ言語を機械語に変換

リスト 1 のプログラムの大体の動作が分かったと思う．そこで，これをメモリーに格納できるように，ビットパターン (16 進整数) に変換しよう．この簡単なプログラムの命令の詳細は分からなくても良いが，命令もビットパターンに変換されることは理解しなくてはならない．

表 2 を見て分かるように，このプログラムは命令とデータからできている．ここで，命令を 16 進数に直す方法が分かれば，プログラムの全てをビットパターンに変換できる．このビットパターンこそ，コンピューターが唯一理解できるマシン語である．ここでは，人間の分かるアセンブラ言語からコンピューターが理解できるマシン語に変換する方法を学ぶのである．

4.2.1 マシン語変換表

文字の変換の仕方が，表 (教科書 p.13 JIS X0201) になっていたように，命令も表になっている．教科書の p.213 の命令語の構成に，全ての命令のビットパターンが書かれている．これから分かるように，CASL II には命令の数は 40 個程度しかない．

ただ，表の見方が，文字のコード表よりちょっと難しい．整数や文字は，16 ビットのビットパターンであったが，命令の場合は 16 ビットであったり，32 ビットだったりする．少し厄介であるが，慣れればたいしたことない．

それでは，実際の表の見方を示す．まずは，表の中央より右側に機械語命令 (アセンブラ言語) が書かれている．その左側がマシン語を表し，右側がその意味 (動作) を記述している．今は，動作はどうでもよいので，アセンブラの命令と機械語の対応を考える．たとえば，LD 命令を例にとる．表の機械語命令 LD を見ると，2 つあることに気が付く．それは，

```
LD   r,adr,x
LD   r1,r2
```

である．LD は分かるとして，それ以外 (オペランド) が分からない．詳しいことは，今後の学習に譲るとして，それを簡単にまとめると，次のようになる．

r	汎用レジスター	GR0〜GR7
r1	1 つの命令で 2 つの汎用レジスターを使うときの一方	GR0〜GR7
r2	もう一方の汎用レジスター	GR0〜GR7
adr	アドレスを示す.	レベル名が書かれることが多い.
x	アドレスをシフトするインデックスレジスタ.	GR1〜GR7

これで表の見方がわかった．アセンブラのプログラムをマシン語に変換できるようになった．

たとえば，ラベル A が $(A007)_{16}$ として，LD GR1,A,GR2 という命令は，

$$\texttt{LD GR1,A,GR2} \quad \Rightarrow \quad \begin{matrix}(1012)_{16}\\(A007)_{16}\end{matrix}$$

と変換される．また，LD GR1,GR2 という命令は，

$$\texttt{LD GR1,GR2} \quad \Rightarrow \quad (1412)_{16}$$

と変換される．これで，命令が 1 語の場合と 2 語の場合があることが分かるであろう．表を見て分かるように，2 語を使う命令場合，その 2 語目は必ず，アドレスとなっている．

これで，全て終わるのはまだ早い．賢い者は，LD GR1,A という命令の変換方法に疑問が湧くであろう．インデックスレジスターが無い場合である．これは，

$$\texttt{LD GR1,A} \quad \Rightarrow \quad \begin{matrix}(1010)_{16}\\(A007)_{16}\end{matrix}$$

と変換される．すなわち，命令を構成する 2 語の最初の 1 語の第 0〜3 ビットがゼロの場合，インデックスレジスターが無いと判断されるのである．もし，インデックスレジスターに GR0 が使えると，インデックスレジスターが無い場合と GR0 を使っている場合の区別ができなくなる．そのような理由から，インデックスレジスターに GR0 が使えないのである．ハードウェアー (CPU) がそうなっているからである．

4.2.2 ハンド アセンブル

準備が整ったので，リスト 1 のプログラムをマシン語 (ビットパターン) に変換する．これをビットパターンに変換したものが，教科書の p.17 の図 2.4 に書かれている．ただし，この表には間違いがあるので，注意が必要である．プログラムの最初の PGM START はアセンブラ命令と言って，機械語に変換されない．これについては来週の授業で説明する．したがって，最初に機械語に変換される命令は，LD GR1,A となる．その変換は，次のように行う．

1. LD という命令から，16 進数 4 桁の表示の最上位の桁は $(1)_{16}$ と分かる．
2. 次の桁は，LD には，$(0)_{16}$ か $(4)_{16}$ である．ここでは，LD r,adr,x のパターンとなっているので，次の桁は $(0)_{16}$ と分かる．
3. 次の桁は，汎用レジスターを示す．ここで使われている汎用レジスターは，GR1 なので，$(1)_{16}$ となる．
4. 次の桁は，インデックスレジスターを示す．インデックスレジスターは無いので，その桁は $(0)_{16}$ となる．

ここの命令は，LD r,adr,x のパターンであるので，命令語長は 2 語である．最初の 1 語は今示したとおり，$(1010)_{16}$ である．次の 1 語は，ラベル A のアドレスである．これは，プログラムが格納されるアドレスに依存する．ここでは，教科書 (p.17 の図 2.4) に沿って，$(A000)_{16}$ からプログラムは格納されるとすると，A のアドレスは $(A007)_{16}$ となる．これが第 2 語のビットパターンとなる．以上をまとめると，

命令	16 進数	2 進数
LD GR1,A	1010	0001000000010000
	A007	1010000000000111

となる．

このようにアセンブラー言語を人間が表を見ながら，マシン語に変換することをハンドアセンブル (ほとんど死語か?) と言う．これは単純作業なので，通常は，コンピューターの仕事である．ただし，コンピューターを学習する者にとっては，一度は経験しておきたいことである．

5 課題 (レポート)

[**問 1**] 本プリントの 10 ページのリスト 1 を参考にして，1+2+3 を計算するプログラムを作成せよ．

[**問 2**] 残りの命令

```
ADDA   GR1,B
ST     GR1,C
RET
```

をハンドアセンブルして，マシン語に直せ．結果は分かっているので，その過程をきちんと書くこと．

[**問 3**] 教科書の 17 ページの図 2.4 は間違っている．間違いを捜し，訂正せよ．

5.1 レポート提出要領

提出方法は，次の通りとする．

期限　11 月 25 日 (金)PM1:00 まで
用紙　A4
提出場所　山本研究室の入口のポスト
表紙　表紙を 1 枚つけて，以下の項目を分かりやすく記述すること．
　授業科目名「電子計算機」
　課題名「課題 5　マシン語」
　3E　学籍番号　氏名
　提出日
内容　問題の解答．