機械語命令 (シフト・比較・分岐)

山本昌志*

2005 年 1 月 27 日

1　前回の復習と本日の学習

1.1　復習

今まで，学習した COMET II の命令は，次の通りである．これを思い出して，本日の学習内容と絡めて，理解を深める必要がある．

- アセンブラ命令

開始	START	プログラムの先頭を示し，入口名と実行開始番地を定義
終了	END	プログラムの終わりを明示
領域確保	DS	プログラムで使うメインメモリーを予約
定義	DC	メインメモリーの初期値設定

- 機械語命令
  - データ転送命令

データ転送	LD	メインメモリーやレジスターの内容を汎用レジスタに転送
データ転送	ST	汎用レジスタのデータをメインメモリーへ転送
アドレス転送	LAD	実効アドレスを汎用レジスターへ転送

  - 算術，論理演算

算術加算	ADDA	1 語のデータを符号付き整数と見なし，加算を行う.
論理加算	ADDL	1 語のデータを符号無し整数と見なし，加算を行う.
算術減算	SUBA	1 語のデータを符号付き整数と見なし，減算を行う.
算術減算	SUBL	1 語のデータを符号無し整数と見なし，減算を行う.
論理積	AND	1 語のデータのビット毎の論理積の演算を行う.
論理和	OR	1 語のデータのビット毎の論理和の演算を行う.
排他的論理和	XOR	1 語のデータのビット毎の排他的論理和の演算を行う.

*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

1.2.1 シフト命令

これは，乗算や除算を行うときに使う．CASL II のシフト命令は，次の通りである．

- レジスタの内容をシフトさせることにより，2^n 倍したり，2^{-n} 倍する．レジスターの内容は，符号付き整数として取り扱われる

算術左シフト	SLA	レジスタの内容を符号ビットを除き左にシフト．空きには 0 が入る．
算術右シフト	SRA	レジスタの内容を符号ビットを除き左にシフト．空きには符号ビットが入る

- レジスタの内容をシフトさせることにより，2^n 倍したり，2^{-n} 倍する．レジスターの内容は，符号なし整数として取り扱われる

論理左シフト	SLL	1 語全てを，左にシフト．空きには 0 が入る
論理右シフト	SRL	1 語全てを，右にシフト．空きには 0 が入る

1.2.2 比較とジャンプ命令

比較とジャンプ命令は，セットで使われることが多く，数の比較を行い，その結果を受けて，処理の実行を変える．FORTRAN や C 言語では，

```
IF(A.GT.B)GO TO 200          if(a>b)goto next_step;
```

と書かれる構文とにている．A.GT.B が比較命令で，IF と GO TO がジャンプ命令に相当する．
CASL II の比較とジャンプの命令は，次の通りである．

- 整数の大小の比較命令を学習する．比較の結果は，フラグレジスタ (FR) を設定することで示される．

算術比較	CPA	データは符号付き整数と見なし，比較を行う．
論理比較	CPL	データは符号無し整数と見なし，比較を行う．

- フラグレジスタ (FR) の値に基づいて，処理を分岐させる．

正分岐	JPL	SF と ZF がともに 0 の時，指定の実効アドレスに分岐
負分岐	JMI	SF が 1 の時，指定の実効アドレスに分岐
非零分岐	JNZ	ZF が 0 の時，指定の実効アドレスに分岐
零分岐	JZE	ZF が 1 の時，指定の実効アドレスに分岐
オーバーフロー分岐	JOV	OF が 1 の時，指定の実効アドレスに分岐
無条件分岐	JUMP	無条件に，指定の実効アドレスに分岐

2　シフト命令

2.1　シフト (基数 N の場合)

10 進数を 10 倍，100 倍，1000 倍，$\cdots$ するのは簡単である．10^n 倍するためには，左にゼロを n 個付ければ良い．これは，左シフトである．同様に 1/10 倍，1/100 倍，1/1000 倍，$\cdots$ するのは簡単である．10^{-n} 倍するためには，小数点の位置を n 個左に寄せれば良い．これは右シフトである．

16 進数の場合も同じである．たとえば，$(EF35)$ を $(16)_{10} = (10)_{16}$ 倍や $(16^2)_{10} = (10^2)_{16}$ 倍，$(16^{-1})_{10} = (10^{-1})_{16}$ 倍や $(16^{-2})_{10} = (10^{-2})_{16}$ 倍すると

$$\begin{aligned}
(EF35)_{16} \times (16^2)_{10} &= (EF35)_{16} \times (100)_{16} \\
&= (EF3500)_{16} \\
(EF35)_{16} \times (16)_{10} &= (EF35)_{16} \times (10)_{16} \\
&= (EF350)_{16} \\
(EF35)_{16} \times (16^{-1})_{10} &= (EF35)_{16} \times (0.1)_{16} \\
&= (EF3.5)_{16} \\
(EF35)_{16} \times (16^{-2})_{10} &= (EF35)_{16} \times (0.01)_{16} \\
&= (EF.35)_{16}
\end{aligned} \tag{1}$$

となる．やはり，右や左にシフトさせれば良い．

2 進数の場合も全く同じである．この場合，2^n の計算が簡単である．n が正の整数の場合，左に n ビットシフトさせる．一方，n が負の整数の場合，n の絶対値分，右にシフトさせる．

$$\begin{aligned}
(110011)_2 \times (2^2)_{10} &= (110011)_2 \times (100)_2 \\
&= (11001100)_2 \\
(110011)_2 \times (2)_{10} &= (110011)_2 \times (10)_2 \\
&= (1100110)_2 \\
(110011)_2 \times (2^{-1})_{10} &= (110011)_2 \times (0.1)_2 \\
&= (11001.1)_2 \\
(110011)_2 \times (2^{-2})_{10} &= (110011)_2 \times (0.01)_2 \\
&= (1100.11)_2
\end{aligned} \tag{2}$$

2.2　CASL II の場合

2.2.1　ビットシフト

CASL II で取り扱う 16 ビットの整数を 2^n 倍する事を考える．もし，その 16 ビットが正で有れば，それは簡単である．先に示したように，n ビット右や左にシフトさせれば良い．

問題は，符号付き整数で，第 15 ビットが 1 の負の場合である．これは，実例を示した方が分かりやすい．たとえば，$(-12)_{10}$ を 2 倍と 4 倍する事を考える．2 倍すると $(-24)_{10}$ で，4 倍すると $(-48)_{10}$ である．それぞれを，2 の補数で取り扱うと，となる．

$$(-12)_{10} \rightarrow (1111\ 1111\ 1111\ 0100)$$
$$(-24)_{10} \rightarrow (1111\ 1111\ 1110\ 1000)$$
$$(-48)_{10} \rightarrow (1111\ 1111\ 1101\ 0000)$$

従って，CASL II の符号付き 16 ビット整数の場合，2^n する場合は，左に n ビットシフトさせて，空いたビットに 0 を入れれば良い．

次に，1/2 倍と 1/4 倍する事を考える．すると

$$(-12)_{10} \rightarrow (1111\ 1111\ 1111\ 0100)$$
$$(-6)_{10} \rightarrow (1111\ 1111\ 1111\ 1010)$$
$$(-3)_{10} \rightarrow (1111\ 1111\ 1111\ 1101)$$

となる．この場合も右に n ビットシフトさせれば良いのであるが，空いたビットには 1 を入れなくてはならない．

ポイント

- 符号無し整数の場合
  - 2^n 倍する場合，左に n ビットシフトさせて，空いたビットに 0 を入れればよい．
  - 2^{-n} 倍する場合，右に n ビットシフトさせて，空いたビットに 0 を入れればよい．
- 符号付き整数の場合
  - 2^n 倍する場合，左に n ビットシフトさせて，空いたビットに 0 を入れればよい．
  - 2^{-n} 倍する場合，右に n ビットシフトさせて，空いたビットには符号ビットを入れればよい．

2.2.2 端数の処理

CASL II の整数をビットシフトを用いて 2 で割ったりすると，端数 (小数部) が生じる．この端数は，16 ビットを越えるので，無視される．ここで，商が切り上げなのか切り下げなのか，疑問が発生する．これについても，実際の整数で考える．

$(5)_{10}$ と $(-5)_{10}$ を 1 ビット右にシフトさせて，1/2 倍してみる．

$$
\begin{aligned}
(5)_{10} &\rightarrow (0000\ 0000\ 0000\ 0101) \\
&\quad 1\text{ビット右} \rightarrow (0000\ 0000\ 0000\ 0010) \rightarrow (2)_{10} \\
(-5)_{10} &\rightarrow (1111\ 1111\ 1111\ 1011) \\
&\quad 1\text{ビット右} \rightarrow (1111\ 1111\ 1111\ 1101) \rightarrow (-3)_{10}
\end{aligned}
$$

この結果から，以下のようにまとめることができる．

ポイント

- 正の整数の場合，端数は切り下げとなる．
- 負の整数の場合，端数は切り上げとなる．

2.3 算術シフト命令

算術シフト命令は，左シフト (SLA) と右シフト (SRA) の 2 つがある．いずれも，符号付き整数を取り扱い，前者は 2^n 倍，後者は 2^{-n} 倍する．算術シフト命令は，符号も考慮していることが重要である．

2.3.1 算術左シフト (SLA)

内容

命令語	SLA
語源	**S**hift **L**eft **A**rithmetic　(shift:移す　left:左　arithmetic:算術)
役割	レジスタの内容を n ビット左に移動させる．空いたビットには，0 が入る． これは，符号付き整数を 2^n 倍しているのと同じ．
書式	教科書 (p.59) の通り．第一オペランドは汎用レジスター．第二オペランドはアドレス．
機能	教科書 (p.59) の通り
フラグレジスタ	教科書 (p.59) の通り．

この命令は，符号付き整数を 2^n 倍する．従って，シフトにより空いたビットには，0 が入る．

使用例

```
SLA   GR0,2        ;GR0 の内容を 2 ビット左へシフト
SLA   GR0,0,GR1    ;GR0 の内容を GR1 の値，左へシフト
```

2.3.2 算術右シフト (SRA)

内容

命令語	SRA
語源	**S**hift **R**ight **A**rithmetic (shift:移す right:右 arithmetic:算術)
役割	レジスタの内容を n ビット右に移動させる．空いたビットには，符号ビットが入る．これは，符号付き整数を 2^{-n} 倍しているのと同じ．
書式	教科書 (p.62) の通り．第一オペランドは汎用レジスター．第二オペランドはアドレス．
機能	教科書 (p.62) の通り
フラグレジスタ	教科書 (p.62) の通り．

この命令は，符号付き整数を 2^{-n} 倍する．従って，シフトにより空いたビットには，符号ビットが入る．

使用例

```
SRA   GR0,2         ;GR0 の内容を 2 ビット右へシフト
SRA   GR0,0,GR1     ;GR0 の内容を GR1 の値，右へシフト
```

2.4 論理シフト命令

論理シフト命令は，左シフト (SLL) と右シフト (SRL) の 2 つがある．いずれも，符号無し整数を取り扱い，前者は 2^n 倍，後者は 2^{-n} 倍する．論理シフト命令は，符号は考慮していない．

2.4.1 論理左シフト (SLL)

内容

命令語	SLL
語源	**S**hift **L**eft **L**ogical (shift:移す left:左 logical:論理上の)
役割	レジスタの内容を n ビット左に移動させる．空いたビットには，0 が入る．これは，符号無し整数を 2^n 倍しているのと同じ．
書式	教科書 (p.65) の通り．第一オペランドは汎用レジスター．第二オペランドはアドレス．
機能	教科書 (p.65) の通り
フラグレジスタ	教科書 (p.65) の通り．

この命令は，符号無し整数を 2^n 倍する．従って，シフトにより空いたビットには，0 が入る．

使用例

```
SLL   GR0,2        ;GR0 の内容を 2 ビット左へシフト
SLL   GR0,0,GR1    ;GR0 の内容を GR1 の値，左へシフト
```

2.4.2 論理右シフト (SRL)

内容

命令語	SRL
語源	**S**hift **R**ight **L**ogical　(shift:移す　right:右　logical:論理上の)
役割	レジスタの内容を n ビット右に移動させる．空いたビットには，0 が入る．これは，符号付き整数を 2^{-n} 倍しているのと同じ．
書式	教科書 (p.67) の通り．第一オペランドは汎用レジスター．第二オペランドはアドレス．
機能	教科書 (p.67) の通り
フラグレジスタ	教科書 (p.67-68) の通り．

この命令は，符号無し整数を 2^{-n} 倍する．従って，シフトにより空いたビットには，符号ビットが入る．

使用例

```
SRL   GR0,2        ;GR0 の内容を 2 ビット右へシフト
SRL   GR0,0,GR1    ;GR0 の内容を GR1 の値，右へシフト
```

3 比較命令

3.1 算術比較 (CPA)

3.1.1 内容

命令語	CPA
語源	**C**om**P**are **A**rithmetic　(compare:比較する　arithmetic:算術)
役割	符号付き整数の比較を行う命令
書式	教科書 (p.56) の通り
機能	教科書 (p.56) の通り
フラグレジスタ	教科書 (p.56) の通り．

この命令は，符号付き整数の差の演算結果の状態がフラグレジスターに設定すると考えれば良い．たとえば，CPA GR1,GR2 の場合，GR1-GR2 の演算結果の正負，あるいはゼロか否か，オーバーフローが有るか無いかがフラグレジスタに設定される．ただし，オペランドである GR1 や GR2 の値は変わらない．

3.1.2 使用例

```
CPA    GR0,GR1        ;GR0-GR0 の状態をフラグレジスターに設定
CPA    GR0,A          ;GR0-(アドレス A の内容) の状態をフラグレジスターに設定
CPA    GR0,A,GR1      ;GR0-(アドレス [A+GR1] の内容) の状態をフラグレジスターに設定
CPA    GR0,=5         ;GR0-5 の状態をフラグレジスターに設定
```

教科書の例題を実行したときのメモリーとレジスターの内容を表 1 示す．1, 5, 6, 7 行はアセンブラ命令なので実行されない．そのため，メモリーやレジスタの値は空白としている．

表 1: 教科書 List4-11(p.56) の実行例.

行	プログラム			GR1	OF	SF	ZF	AA	BB
1	PGM	START							
2		LD	GR1,AA	2	0	0	0	2	-1
3		CPA	GR1,BB	2	0	0	0	2	-1
4		RET		2	0	0	0	2	-1
5	AA	DC	2						
6	BB	DC	-1						
7		END							

教科書の List4-11 はつまらない例題で，CPA コマンドも動作も分からないし，説明もおかしい．次のようなサンプルの方が良い．

表 2: CPA の実行例.

行	プログラム			GR1	OF	SF	ZF	AA	BB
1	PGM	START							
2		LD	GR1,AA	2	0	1	0	-2	-3
3		CPA	GR1,BB	2	0	0	0	-2	-3
4		RET		2	0	0	0	-2	-3
5	AA	DC	-2						
6	BB	DC	-3						
7		END							

3.2 論理比較 (CPL)

3.2.1 内容

命令語 CPL

語源 **C**om**P**are **L**ogical (compare:比較する logical:論理上の)

役割 符号無し整数の比較を行う命令

書式 教科書 (p.57) の通り

機能 教科書 (p.57) の通り

フラグレジスタ 教科書 (p.57) の通り.

この命令は，符号無し整数の差の演算結果の状態がフラグレジスターに設定すると考えれば良い．たとえば，CPA GR1,GR2 の場合，GR1-GR2 の演算結果の正負，あるいはゼロか否か，オーバーフローが有るか無いかがフラグレジスタに設定される．ただし，オペランドである GR1 や GR2 の値は変わらない．

3.2.2 使用例

```
CPL    GR0,GR1       ;GR0-GR0 の状態をフラグレジスターに設定
CPL    GR0,A         ;GR0-(アドレス A の内容) の状態をフラグレジスターに設定
CPL    GR0,A,GR1     ;GR0-(アドレス [A+GR1] の内容) の状態をフラグレジスターに設定
CPL    GR0,=5        ;GR0-5 の状態をフラグレジスターに設定
```

教科書の例題を実行したときのメモリーとレジスターの内容を表 3 示す．1, 5, 6, 7 行はアセンブラ命令なので実行されない．そのため，メモリーやレジスタの値は空白としている．

このサンプルプログラムで注意することは，符号付き整数でアドレス AA の値を決めているが，比較を行うときは符号無し整数としている．それらは，

アドレス	ビットパターン	符号無整数	符号有整数
AA	0000000000000010	$(2)_{10}$	$(2)_{10}$
BB	1111111111111111	$(65535)_{10}$	$(-1)_{10}$

となっている．

表 3: 教科書 List4-12(p.58) の実行例.

行	プログラム			GR1	OF	SF	ZF	AA	BB
1	PGM	START							
2		LD	GR1,AA	2	0	0	0	2	-1
3		CPL	GR1,BB	2	0	1	0	2	-1
4		RET		2	0	1	0	2	-1
5	AA	DC	2						
6	BB	DC	-1						
7		END							

4　ジャンプ命令

4.1　正分岐 (JPL)

4.1.1　内容

命令語	JPL
語源	**J**ump**PL**us　(jump:ジャンプ　plus:プラス (正))
役割	フラグレジスタ (FR) の SF(Sign Flag) が 0，かつ，ZF(Zero Flag) が 0 のとき，指定の実効アドレスに制御が移る．
書式	教科書 (p.71) の通り
機能	教科書 (p.71) の通り
フラグレジスタ	変化せず．

この命令は，これ以前の演算の結果の状態が，0 を含まない正の場合，指定の実効アドレスに分岐させる．通常は，比較命令とペアで使われることが多いが，そうでない場合もある．この命令を使う場合は，フラグレジスタの状態をよく考える必要がある．

4.1.2　使用例

```
JPL    A          ;SF=0 かつ ZF=0 の場合，アドレス a にジャンプ
JPL    A,GR1      ;SF=0 かつ ZF=0 の場合，アドレス [A+GR1] にジャンプ
```

教科書の例題 (p.71 List4-22) のプログラムを図 1 に示す．

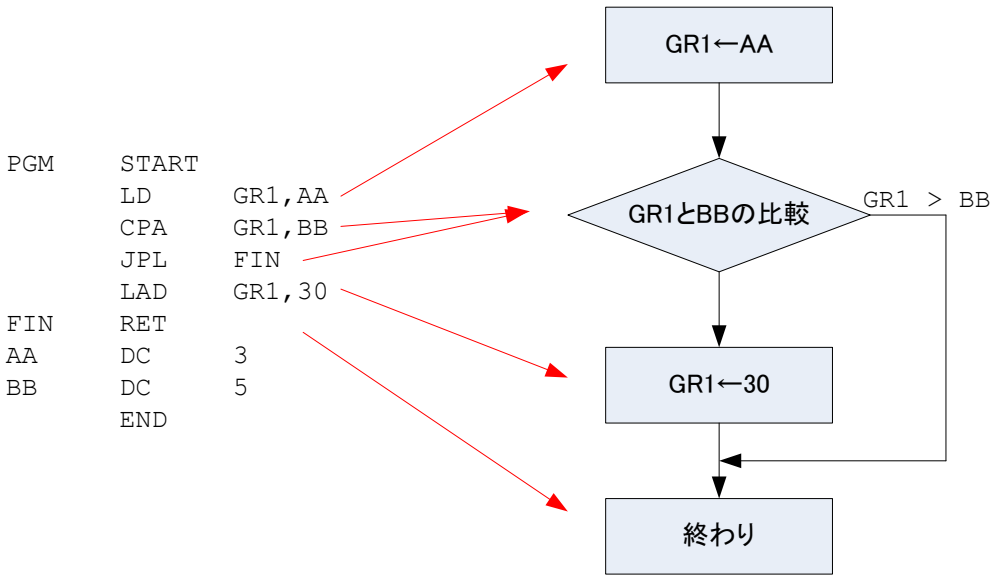

図 1: List4-22 のプログラムとフローチャート

4.2 負分岐 (JMI)

命令語 JMI
語源 **J**ump**MI**nus (jump:ジャンプ　minus:マイナス (負))
役割 フラグレジスタ (FR) の SF(Sign Flag) が 1 のとき，指定の実効アドレスに制御が移る．
書式 教科書 (p.72) の通り
機能 教科書 (p.72) の通り
フラグレジスタ 変化せず．

4.3 非零分岐 (JNZ)

命令語 JNZ
語源 **J**ump on **N**on **Z**ero (jump:ジャンプ　non:非　zero:ゼロ)
役割 フラグレジスタ (FR) の ZF(Zero Flag) が 0 のとき，指定の実効アドレスに制御が移る．
書式 教科書 (p.73) の通り
機能 教科書 (p.73) の通り
フラグレジスタ 変化せず．

4.4 零分岐 (JZE)

命令語 JZE
語源 **J**ump on **ZE**ro (jump:ジャンプ　zero:ゼロ)
役割 フラグレジスタ (FR) の ZF(Zero Flag) が 1 のとき，指定の実効アドレスに制御が移る．
書式 教科書 (p.73) の通り
機能 教科書 (p.74) の通り
フラグレジスタ 変化せず．

4.5 オーバーフロー分岐 (JOV)

命令語 JOV
語源 **J**ump on **OV**erflow (jump:ジャンプ　overflow:あふれる)
役割 フラグレジスタ (FR) の OF(Overflow Flag) が 1 のとき，指定の実効アドレスに制御が移る．
書式 教科書 (p.74) の通り
機能 教科書 (p.74) の通り
フラグレジスタ 変化せず．

4.6 無条件分岐 (JUMP)

命令語	JUMP
語源	unconditional **J**ump　　(jump:ジャンプ)
役割	フラグレジスタ (FR) に関係なく，強制的に指定の実効アドレスに制御を移す.
書式	教科書 (p.75) の通り
機能	教科書 (p.75) の通り
フラグレジスタ	変化せず.

5 課題

5.1 課題内容

課題を課すので，レポートとして提出すること．課題内容は，以下の通り.

[**問 1**] 整数演算，643/32+5 を計算する．以下の問いについて答えよ.

- CASL-II のプログラムを作成せよ.
- このプログラムを実行させた場合，計算結果である整数の値はいくらか?

[**問 2**] 教科書 p.75 の List4-23 のプログラムについて，以下の問いに答えよ.

- フローチャートを作成せよ．そして，ソースプログラムの各行とフローチャートの対応を示せ.
- 使っている汎用レジスタの値を各行が実行された後，どのようになるか示せ.
- 各行の実行後のフラグレジスタの値を示せ.
- 各行の実行後のアドレス AA と BB の値を示せ.

5.2 レポート 提出要領

期限　　2 月 3 日 (金) PM 1:00
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　機械語命令 (シフト・比較・分岐)」
　　3E　学籍番号　氏名
　　提出日
内容　　2 ページ以降に問いに対する答えを分かりやすく記述すること.