算術加算・減算

山本昌志*

2004 年 8 月 30 日

1　前回の復習と本日の学習

1.1　復習

1.1.1　アセンブラ命令

アセンブラ言語を機械語に変換するプログラムであるアセンブラーにその変換方法を指示するのがアセンブラ命令である。CASL II には、4 つのアセンブラ命令がある。

START	プログラムの先頭に、必ず書く必要がある。プログラムの実行開始番地を指示する。
END	プログラムの最後に、必ず書く必要がある。プログラムの記述の最後を示す。プログラムの実行の終了を示すものではない。
DC	プログラムで処理すべきデータを定義する。メモリーの初期値を与えると解釈してもよい。
DS	プログラムの実行に必要なメモリーを確保する。

プログラムは命令とデータから構成されると以前に述べたが、データ部を構成するために、DC と DS の命令がある。

1.1.2　機械語命令 (データ転送)

実際の CPU の動作を示す命令が機械語命令である。要するにこれは、CPU というハードウェアーができることを示している。プログラムを構成するデータと命令のうち、後者は機械語命令から構成される。

前回の授業では、データ転送に関する 3 の機械語命令を学習した。

LD	メインメモリーや他の汎用レジスターのデータ (内容) を汎用レジスターにコピーする。
ST	汎用レジスターのデータ (内容) をメインメモリーにコピーする。
LAD	メインメモリーの実効番地を汎用レジスターにコピーする。指標レジスターの使い方によっては、汎用レジスターの値を操作できる。

*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

本日は、CASL II の整数の演算である算術加算・減算について、説明する。学習のゴールは、以下の命令の動作を理解することである。

算術加算	ADDA	符号有り整数の足し算の演算である。
算術減算	SUBA	符号有り整数の引き算の演算である。

理解のポイントは、

- 加算するデータは 16 ビットで、それを符号付 2 進数として取り扱う。負の数は 2 の補数という表現が使われる。
- 計算結果、フラグレジスタがどのようになるかを考える。

である。

2　演算とは

2.1　コンピューターでの演算

コンピューターの仕事は、データの加工です。与えられたデータを、目的のデータに加工します。例えば、

- 整数の 5 と 8 のデータが与えられると、それを加工して和である 13 を表示する。
- CD-ROM のビット列のデータを加工して、音に変換する。
- 飛行機のスロットルレバーの角度とエンジンの回転数などのデータから、燃料噴射弁の角度を与えるデータを作る。

等です。どのような場合でも、図 1 のようになっている。この入出力データのことを情報と言い、それをプログラムの命令により処理を行い、出力データを作る。情報処理とはこのようなことを言う。

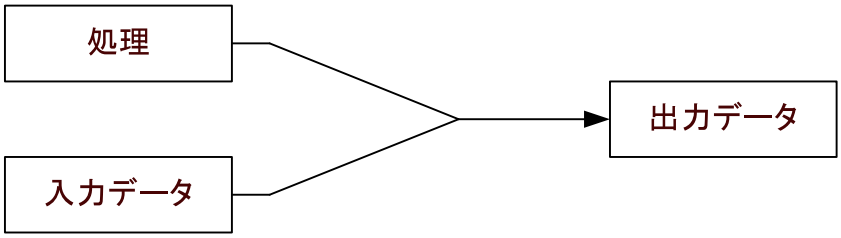

図 1: 情報処理

コンピューターの内部、とくにデータの処理を行う CPU では、入力と出力のデータはビット列である。CPU では命令に従い、ビット列の操作を行っている。このビット列を操作する処理のことを演算といい、プログラムでは演算命令がそれを行う。

数学でも演算と言う言葉を使うが、その内容は非常に似ている。例えば、数学との対比では、

- $\sin \pi/2$ を計算して、その結果として 1 を得る。この場合、入力データは $\pi/2$ で、演算は sin で、出力データは 1 である。

のように考えられる。

ポイント

コンピューターは、入力データのビット列から出力ビット列を作る処理を行っている。そのビット列の変換の処理を演算と言う。

2.2 CASL II の演算

先ほど述べたように、演算命令により、データのビットパターンを変化させる。変化させるデータは、

- 汎用レジスター (GR0〜7)
- メインメモリーの内容
- フラグレジスター (FR)

である。残りのプログラムレジスター (PR) とスタックポインター (SP) は演算命令により変化はしない。

CASL II で用意されている演算は、

- 算術・論理演算命令
- 比較演算命令
- シフト演算命令

である[1]。たったこれだけで、あらゆるデータ処理を行うのである。命令の動作については、今後、学習する。諸君は、数学的な準備は出来ているので、これらがそんなに難しく思う必要が無い。例えば、+ と言う数学の記号の代わりに、ADDA と書くことを覚えればよい。

2.3 整数演算

算術・論理演算命令は、整数の演算とブール代数の演算がある。それぞれは、

- 整数の演算
  - 算術加算 (ADDA)
  - 算術減算 (SUBA)
  - 論理加算 (ADDL)
  - 論理減算 (SUBL)

[1] 教科書の p.208 に全ての演算命令が書かれている

- ブール代数の演算
- 論理和 (OR)
- 論理積 (AND)
- 排他的論理和 (XOR)

と分けることができる。整数の演算のくせに、論理加算・減算というのも変な気がするが、仕様ではそうなっている。

本日は、算術加算・減算の演算を学習する。算術加算・減算と論理加算・減算の違いは、前者は処理する16ビットのデータを符号有り整数、後者は符号無し整数として取り扱うことである。

3 算術加算・減算

算術加算と減算は、データを16ビットの符号付整数として計算する。CASL IIでは加算と減算の命令が用意されているが乗除算は無い。

3.1 算術加算 (ADDA)

3.1.1 内容

命令語 ADDA: **ADD A**rithmetic (add:加える arithmetic:算術)

役割 符号有り整数の加算を行う命令

書式と内容

ラベル欄	命令コード欄	オペランド欄	
label	ADDA	r1,r2	r1←r1+r2
label	ADDA	r,*adr*[,x]	r←r+(*adr*+[x])$_{naiyou}$

FR 計算結果に応じて、変化する。

Flag	bit	計算結果	ビットパターン
OF	1	結果 <-32768 または 32767< 結果	結果が16ビットを超えたとき
	0	-32768≦ 結果 ≦32767	結果が16ビット以内
SF	1	結果が負の場合	第15ビットが1
	0	結果が正の場合	第15ビットが0
ZF	1	結果がゼロの場合	全てのビットが0
	0	結果がゼロ以外の場合	いずれかのビットが1

2個の16ビットの値を加算する命令である。16進数でも10進数でも正しく計算できる。ただし符号付で計算を行うので、最上位の第15ビットは符号ビットを表し、2の補数で表現される。

3.1.2 例

```
ADDA  GR0,GR1       ;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        ;GR0 ← GR0+5
```

3.2 算術減算 (SUBA)

3.2.1 内容

命令語　SUBA:　**SUB**tract **A**rithmetic　(subtract:引き算　arithmetic:算術)

役割　符号有り整数の減算 (引き算) を行う命令

書式と内容

ラベル欄	命令コード 欄	オペランド 欄	
label	SUBA	r1,r2	r1←r1-r2
label	SUBA	r,*adr*[,x]	r←r-(*adr*+[x])$_{内容}$

FR　計算結果に応じて、変化する。

Flag	bit	計算結果	ビットパターン
OF	1	結果 <-32768 または 32767< 結果	結果が 16 ビットを超えたとき
	0	-32768≦ 結果 ≦32767	結果が 16 ビット以内
SF	1	結果が負の場合	第 15 ビットが 1
	0	結果が正の場合	第 15 ビットが 0
ZF	1	結果がゼロの場合	全てのビットが 0
	0	結果がゼロ以外の場合	いづれかのビットが 1

二つのデータの引き算を行う。その他は、加算命令と同じ。

3.2.2 例

```
SUBA  GR0,GR1       ;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        ;GR0 ← GR0-5
```

4　乗除算はどうするの?

ここで、賢明な諸君は、4 則演算の残りの 2 つ、乗除算はどうしたのという疑問が湧くはずである。湧いて欲しい。

最近の CPU は、これら乗除算のハードウェアーが実装されており、それに対応する機械語命令がある。しかし、単純な COMET II には、そのハードウェアはなく、当然機械語命令も無い。そのため、ソフトウェアーでその仕組みを実現させなくてはならない。詳細については、後で学習するが、ちょっとだけ方法を示す。興味深い方法である。

4.1　乗算

4.1.1　算術加算を使う方法

例えば、10×3 を計算する場合、10+10+10 のように必要な回数だけ足し合わせて乗算を行う。このように算術加算を用いて乗算を行うことができる。ただし、この方法はもっとも原始的で効率の悪い方法である。人間がこの計算を行うのは非現実的であるが、単純作業を非常に高速で行うコンピューター向きの方法である。

この例でもわかるように加算ができれば、乗算はできるのである。

4.1.2　ビットシフトを使う方法

もう少し効率の良い方法は、ビットシフトを使う方法である。これは、小学生のときに学習をした筆算を用いた掛け算と同じである。たとえば、34×24 を計算する場合、筆算は $34 \times (2 \times 10^1 + 4 \times 10^0)$ と分解したはずである。そうして、次の手順でこの除算を行ったはずである。

1. 34×2 を計算し、1 桁ずらす (10 倍する)。
2. 34×4 を計算する。
3. 先の計算結果を合計する。この合計 816 が 34×24 の計算結果である。

同じことを 2 進数で行う。これがコンピューターによる乗算である。先ほどと同じ計算 (32×24) を行う。これを 2 進数で表現すると、

$$(100010)_2 \times (11000)_2 = (100010)_2 \times (1 \times 2^4 + 1 \times 2^3)$$

となる。これを先ほど同様の手順で計算する。

1. 掛け算は 1 倍なので計算する必要が無く、最初に $(100010)_2$ を 4 桁左にずらす (ビットシフト)。すると、$(1000100000)_2$ となる。
2. 次に $(100010)_2$ を 3 桁左にずらす。すると、$(100010000)_2$ となる。
3. 先の計算結果を合計すると、$(1100110000)_2$ となる。これは、10 進数の 816 である。

ここでは、ビットシフトと加算命令を使った。これらの命令が用意されていれば乗算ができることがわかるであろう。実際、CASL II にはビットシフトの命令は用意されている。

4.2 除算

4.2.1 算術減算を使う方法

10/3 の計算は 10 からから 3 を引いていきます。そして、負になったら計算は完了です。すると、商と余りが分かります。算術減算を用いて乗除算が出来ます。この例でもわかるように減算ができれば、除算はできるのである。この方法は効率が悪いので、もう少しましな方法を考える。小学生の時に学習した筆算のアルゴリズムを適用すれば効率は良くなる。計算は次のようにする。計算の準備として、10 と 3 を 2 進数で表す。

$$(10)_{10} = (1010)_2 \qquad (3)_{10} = (11)_2$$

次に示すように計算すれば、計算効率は上がるであろう。計算順序は、筆算での割り算と同じである。

1. $(1)_2$ から $(11)_2$ を減算したいが、負になるのでそれは不可とする。
2. $(10)_2$ から $(11)_2$ を減算したいが、これも負になるので不可とする。
3. $(101)_2$ から $(11)_2$ を減算する。それは可能で、結果は $(10)_2$ で、その桁に $(1)_2$ が立つ。
4. 先ほどの残りと次の桁を合わせた $(100)_2$ は減算可能である。減算の結果は $(1)_2$ で、その桁に $(1)_2$ が立つ。
5. これ以上桁がないので、計算は完了である。商は $(11)_2 = (3)_{10}$ 余りは $(1)_2 = (1)_{10}$ となる。

4.2.2 ビットシフトを使う方法

直ちに、ビットシフトが適用できるのは、割る数が 2^n になっている場合である。ただ、先ほどの筆算のアルゴリズムでもビットシフトは使ってはいる。

4.3 まとめ (乗除算)

ということで、加算と減算ができれば乗除算は可能である。さらに賢明な諸君は、次の疑問が湧くはずである。湧いて欲しい。三角関数や指数関数などは、どうやって計算しているのか?。三角関数や指数関数は、テイラー展開 (マクローリン展開) を使うと 4 則演算に分解できることを学習したはずである。したがって、4 則演算ができれば、それらの関数は計算可能である。高級言語のコンパイラーは、これらの関数を 4 則演算に変換して計算するようにしている。

5　課題

期限　　次回の授業まで
用紙　　A4 表紙付き
表紙　　授業科目名「電子計算機」
　　課題名「算術加算・減算」
　　3E 学籍番号　氏名
　　提出日

5.1　算術加算

次の計算を行うプログラムを作成し、プログラムの各実効段階のフラグレジスタの値を示せ。

問 1　　**10+123**
問 2　　**#0FFF+#0001**
問 3　　**#8001+32767**

5.2　算術減算

次の計算を行うプログラムを作成し、プログラムの各実効段階のフラグレジスタの値を示せ。

問 1　　**10-123**
問 2　　**#7FFF-#8FFE**
問 3　　**#8001-32767**