CASL IIのプログラム例(その6)

山本昌志*

2005年1月28日

1　復習と本日の内容

1.1　復習

先週と先々週の講義では、繰り返し処理とサブルーチンについて学習した。そこでは、データの中から、最大値を探索プログラムの例を示した。

1.2　本日の内容

今回は、以下の2点について、学習する。

- 通常、複数の同じ性質のデータは、ラベルとDC命令を用いて、1行で書かれる。その場合、ラベル名はデータの先頭アドレスを示す。最終アドレスを探す場合、結構面倒である。本日は、最終アドレスにラベル名を付ける方法を学習する。
- 数値データを文字データに変換する方法を学習する。

2　[例題9] ラベルを2重につける方法

教科書のList5-9のプログラムを例にして、ラベルを2重につける方法について説明する。

2.1　教科書の例

教科書の例題6(p.97)〜8(p.101)は、いずれも与えられたデータの最大値を求めるプログラムであった。これらの場合、最大値を求めたいデータの数列とその数が与えられていた。データ数を元に、数列を読みしと比較を繰り返すことにより最大値を探索した。ここでは、データ数が与えられていない場合のテクニックを学習する。

教科書のプログラムの内容は、

*独立行政法人　秋田工業高等専門学校　電気工学科

- ラベル DATA が示すアドレスからデータが格納されている。
- アセンブラ命令 DS を上手に使うことにより、データの終わりのアドレスは、ラベル LAST-1 で示している。
- アドレス DATA〜LAST-1 に格納されている数列を合計して、ラベル SUM に格納する。

である。このプログラム例で学習することは、最終データがあるアドレスにラベル名をつけることである。それは、プログラム中で示しているように

```
DATA  DC   1,5,6,8,9
LAST  DS   0       ; 数列の最終アドレス+1
```

とするのである。こうすると数列の先頭のアドレスは DATA で、最終アドレスは LAST-1 で示すことができる。

2.2 プログラムの構造とフローチャート

このプログラムのフローチャートを図 1 に示す。このプログラムを理解するために、ここで使われているレジスターやラベルの内容を表 3 に示しておく。プログラムの内容を理解するときには、変数が示す内容を考えるのが第一歩である。諸君も、プログラムの内容を調べるときには、変数の意味を調べることから始めよ。全て分からなくても良い。分かるものから、その意味をプログラム中に書け。

このプログラムは、そんなに難しくなく、最終アドレス間で次々に加算しているのが理解できるであろう。

表 1: 汎用レジスターとメモリの内容

GR0	加算途中および結果の合計値
GR1	データの最終アドレス+1(LAST)
GR2	読み込むデータのアドレス (LAST〜LAST-1)
DATA	加算する数列の先頭アドレス
LAST	加算する数列の最終アドレス
SUM	数列を加算した結果を格納するメモリーのアドレス

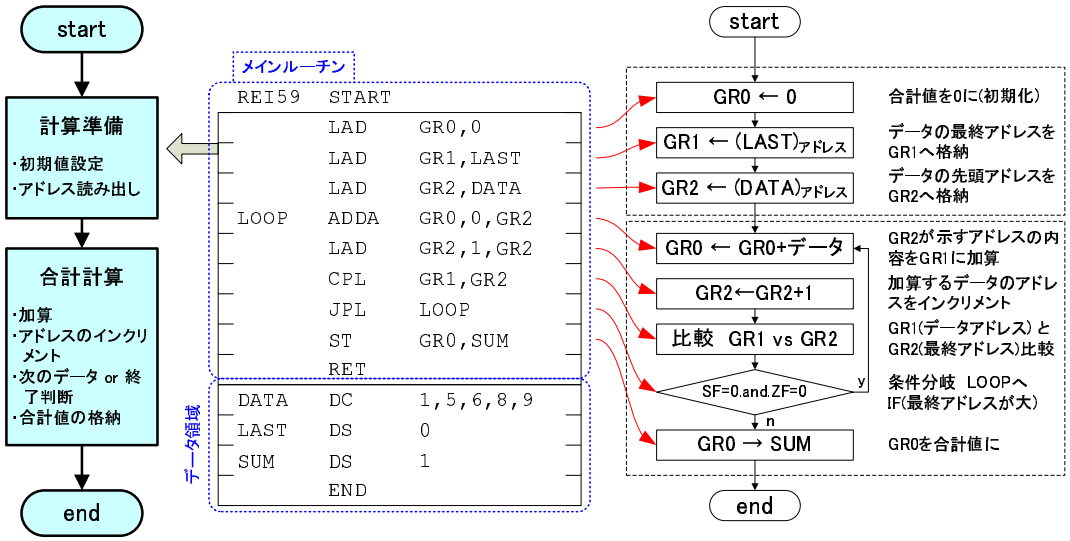


図 1: 教科書の List5-9 のプログラムの構造とフローチャート

3 [例題 10] 数値データを文字データに変換

教科書の List5-10 のプログラムを例にして、数値データを文字データに変換する方法を説明する。

3.1 教科書の例

教科書の例のプログラムは、メモリーに格納されている整数データを文字に変換して表示するものである。表示の約束は、以下の通りである。

- ラベル A に入っている数値 (最大 5 桁) を文字列に変換して、OUT 命令で表示する。
- 表示には 6 桁 (カラム) 用意して、第 1 桁は符号で負の場合のみ表示する。2〜6 桁は絶対値を表す。ただし、上位の桁が 0 の場合、スペースを入れる。

ラベル A の数値が最大 5 桁なのは、それを符号付き整数として取り扱うからである。その場合、1 ワードで表現できるのは、-32768〜32767 の範囲である。これが 5 桁で、符号を合わせると表示に 6 桁必要になる。実際には図のように、6 カラムで負号のみ左端に表示し、数値は右詰で表示する。

プログラムの作成方法と動作について、教科書に沿って説明する。

数値	表示					
-32768	-	3	2	7	6	8
-2693	-		2	6	9	3
-739	-			7	3	9
-12	-				1	2
-8	-					8
0						0
6						6
84					8	4
813				8	1	3
5173			5	1	7	3
32767		3	2	7	6	7

図 2: 教科書の List5-10 の数値を文字に変換

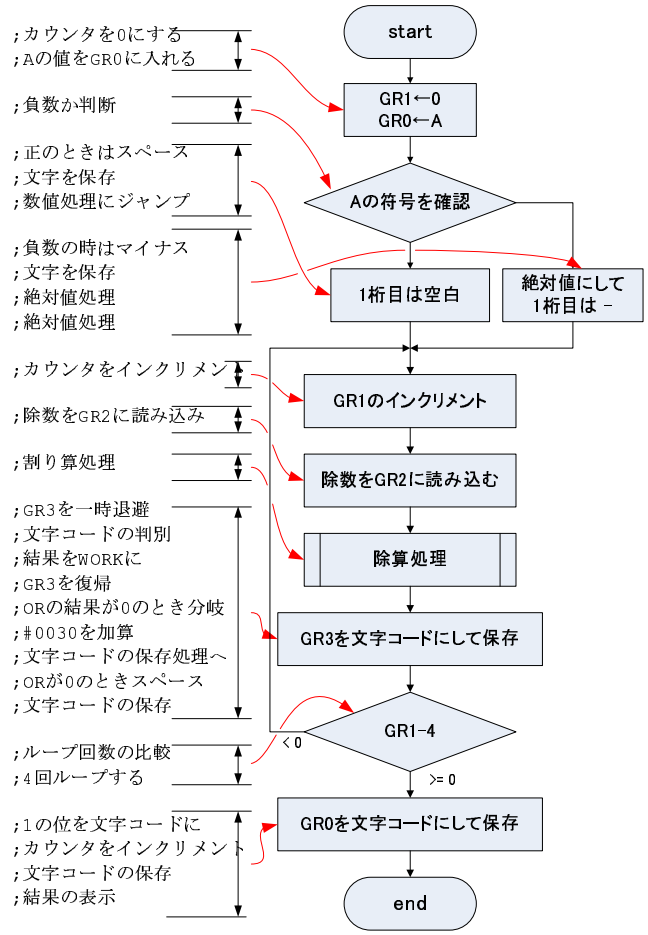
表 2: メインルーチンの汎用レジスターとメモリの内容

GR0	処理すべき数値 (処理する毎に桁が減少)
GR1	カウンター (処理する桁を示す)
GR2	除数
GR3	その桁の値 (整数)
C4	わり算が必要な桁数
BUFF	文字を格納するメモリーの先頭アドレス
MOJI	#0030。これを整数に足せば、その文字コードになる。
WORK	以前の桁のフラグ (0:全てゼロ それ以外:ゼロ以外が現れた)
WORK+1〜4	桁

表 3: サブルーチン DIV の汎用レジスターとメモリの内容。このサブルーチンでは GR0÷GR2 → 商 GR3 余り GR0 を計算している。

レジスター	実行前	実行後
GR0	被除数	余り
GR2	除数	除数 (変化無し)
GR3	不定	商

```
REI5A   START
        LAD     GR1,0           ;カウンタを0にする
        LD      GR0,A           ;Aの値をGR0に入れる
        JMI     FUSUU           ;負数か判断
        LD      GR3,=#0020      ;正のときはスペース
        ST      GR3,BUFF,GR1    ;文字を保存
        JUMP    KEISAN          ;数値処理にジャンプ
FUSUU   LD      GR3,='-'        ;負数の時はマイナス
        ST      GR3,BUFF,GR1    ;文字を保存
        LAD     GR0,0           ;絶対値処理
        SUBA    GR0,A           ;絶対値処理
KEISAN  LAD     GR1,1,GR1       ;カウンタをインクリメント
        LD      GR2,WORK,GR1    ;除数をGR2に読み込み
        CALL    DIV             ;割り算処理
        PUSH    0,GR3           ;GR3を一時退避
        OR      GR3,WORK        ;文字コードの判別
        ST      GR3,WORK        ;結果をWORKに
        POP     GR3             ;GR3を復帰
        JZE     SPACE           ;ORの結果が0のとき分岐
        OR      GR3,MOJI        ;#0030を加算
        JUMP    HOZON           ;文字コードの保存処理へ
SPACE   LD      GR3,=#0020      ;ORが0のときスペース
HOZON   ST      GR3,BUFF,GR1    ;文字コードの保存
        CPA     GR1,C4          ;ループ回数の比較
        JMI     KEISAN          ;4回ループする
        OR      GR0,MOJI        ;1の位を文字コードに
        LAD     GR1,1,GR1       ;カウンタをインクリメント
        ST      GR0,BUFF,GR1    ;文字コードの保存
        OUT     BUFF,LEN        ;結果の表示
        RET
A       DC      -2050
BUFF    DS      6               ;文字コードの保存領域
LEN     DC      6               ;文字コードの長さ
WORK    DC      0,10000,1000,100,10
MOJI    DC      #0030
C4      DC      4
        END
```



サブルーチンの処理

GR0÷GR2 → GR3余りGR0

```
DIV     START
        LAD     GR3,0           ;商を0にする
DLOOP   CPA     GR0,GR2         ;除数で引けるか判別
        JMI     FIN             ;引けなければFINへ
        SUBA    GR0,GR2         ;被除数-除数
        LAD     GR3,1,GR3       ;商をインクリメント
        JUMP    DLOOP           ;繰り返し処理へ
FIN     RET
        END
```

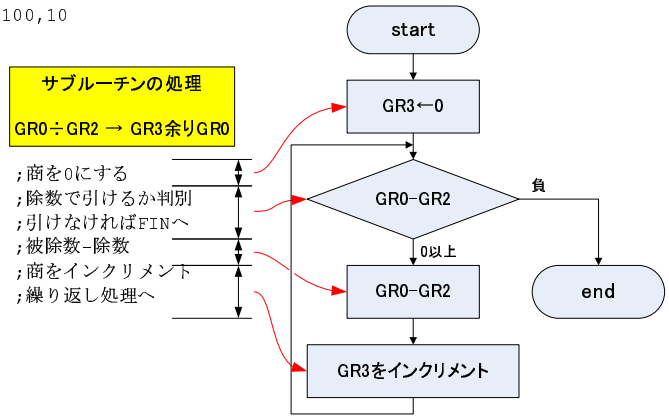


図 3: 教科書の List5-10 のプログラムとフローチャート