CASL IIのプログラム例(その2)

山本昌志*

2006年2月14日

1　前回の復習と本日の学習内容

1.1　復習

前回の講義では，教科書 [1] の第5章の **CASL IIプログラム例**の [例題1]〜[例題3] を学習した．

- **[例題1] 加算**
  - 演算を行う場合，汎用レジスターを使用しなくてはならない．
  - メモリーの操作が必要である．
- **[例題2] 加算と条件分岐**
  - 条件により実行する文が異なる処理を分岐という．
  - CASL IIの場合，ジャンプ命令を使って分岐を行う．フラグレジスターの値によりジャンプ先が変わる．
  - フラグレジスターの値は，比較命令または演算により設定される．
- **[例題3] マスク処理と条件分岐**
  - 特定のビットを取り出すためのビットパターンをマスクという．
  - 以下のようにすれば，特定のビットパターンになっているか，否かを調べることができる．
    1. マスクと AND 演算を行い，調べたい場所ののビットパターンを取り出す．
    2. CPL 命令により，ビットパターンの比較を行う．結果は，フラグレジスターの ZF に設定される．

1.2　本日の内容

本日は教科書の p93-101 の以下の内容について学習する．

- **[例題4] 論理演算とアドレス修飾**
  - 指標レジスターを用いたアドレス修飾の方法を学習する．

*国立秋田工業高等専門学校　電気工学科

- **[例題 5] シフト 演算**
  - シフト 演算を用いた効率の良いかけ算と割り算の方法を学習する.
- **[例題 6] 繰り返し処理**
  - 繰り返し処理 (ループ文) のプログラム方法を学習する.
- **[例題 7] 繰り返し処理とサブルーチン**
  - サブルーチンのプログラム方法を学習する.

2 [例題 4] 論理演算とアドレス修飾

教科書の List 5-4 のプログラムを例にして，論理演算とアドレス修飾について説明する．以下のことが，ここでの学習の重要なポイントである.

- アドレス修飾の使い方
- カウンターとインクリメントのプログラムの記述方法.

2.1 論理演算

2.1.1 教科書の例

教科書のプログラムは,

- ラベル A,B に#0030,#009F が格納されている.
- ラベル ANS から，3 語このプログラムで確保されている.
- ANS から確保された 3 語の領域に，A AND B と A OR B，A XOR B の演算結果を格納せよ.

と言う問題を解く，プログラムである.

このようなプログラムを作成するために必要なことは,

- データ領域
  - 演算の対象データ (#0030,#009F) をラベル (A,B) を指定してメモリーに書き込む.
  - 演算結果を書き込む領域をラベル (ANS) を指定して，確保する.
- 命令領域
  - 演算対象データをレジスターにコピー
  - 演算の実行
  - 計算結果の格納

である.

2.1.2 アドレス修飾

大まかなプログラムの流れは，分かった．また，論理演算も説明することもないだろう．演算対象のデータのそれぞれのビット毎の論理和 (OR) と論理積 (AND)，排他的論理和 (XOR) を計算しているだけである．

プログラムの命令領域とデータ領域は，図 1 のようになるだろう．プログラムの書き方によっては，こうならないこともあるが，通常はこのようになる．

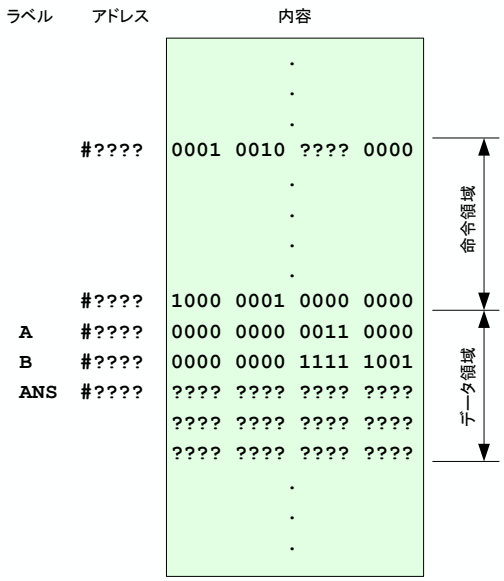

図 1: 教科書の List5-4 のプログラムを実行する場合のメモリ構造．図中の?は値はあるが，不明を示している．

この場合，プログラムのデータ領域にアクセスする事を考える．ラベル A や B は簡単で，ラベル名を示せば良い．ラベル名はアドレスを示すからである．問題は，結果を格納する領域である．このアドレスは，3 つ続いて確保されているが，先頭だけ ANS とラベル名がある．残りの 2 つの表し方である．これらのアドレスは，ANS+1 と ANS+2 である．ANS のアドレスにオフセットの値を加算するのである．

プログラムで使うメモリーのアドレスは，ANS+オフセットで，オフセットは，0,1,2 とすれば良い．論理和の結果を ANS+0，論理積の結果を ANS+1，論理和の結果を ANS+2 に格納する．プログラムでは，オフセットの 0,1,2 を GR2 に入れておき，

```
ST      GR1,ANS,GR2
```

と書く．演算の結果 (GR1) の値が，ANS にオフセット値 (GR2) を加えたアドレスに格納される．

ここで，使っている GR2 のように，1 つずつ値が増加するものをカウンターと呼ぶことがある．これを使うためには，

- カウンターの初期化．ここでは，GR2 をゼロに設定する．

– CASL では，LAD GR2,0

- カウンターのインクリメント．カウンターの値を 1 増加させる．
  – CASL では，LAD GR2,1,GR2

とする．このテクニックは，重要である．内容をよく理解する必要がある．

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 2 に示す..

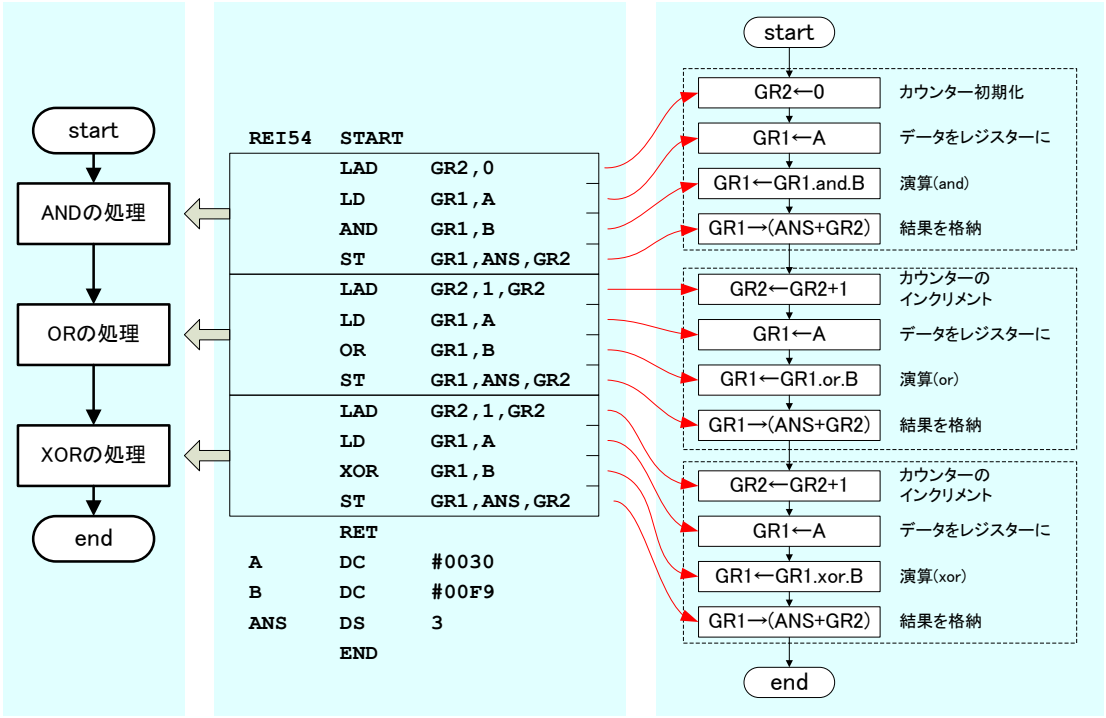

図 2: 教科書の List5-4 のプログラムの構造とフローチャート

3 [例題 5] シフト 演算

教科書の List 5-5 のプログラムを例にして，シフト演算を使ったかけ算と割り算の方法を学習する．以下のことが，ここでの学習の重要なポイントである．

- データを 2^n 倍するためには，データのビットをシフトさせることにより可能である．
- これを上手に使うと任意のかけ算と割り算ができる．

3.1 積の演算

積 (かけざん) の演算を行うとき，シフト命令を使えば効率の良いプログラムができる．ビットシフトを用いると積の演算ができる理由は以前述べているが，忘れた人もいるので，もう一度，説明する．

シフト命令を使った積の演算は，小学生のときに学習をした筆算の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，次の手順でこの計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．

シフトと加算命令でかけ算ができることが分かったはずである．

今回の問題のように分数 (少数) の場合でも，

$$\begin{aligned} 0.75 &= \frac{1}{2} + \frac{1}{4} \\ &= \left(2^{-1}\right) + \left(2^{-2}\right) \end{aligned} \tag{1}$$

と分解する．右に 1 ビットシフトさせたものと，右に 2 ビットシフトさせたものを加算すれば良い．

教科書のように

$$0.75 = 1 - \left(2^{-2}\right) \tag{2}$$

と分解するのは一般的ではない．

3.2 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 3 に示す..

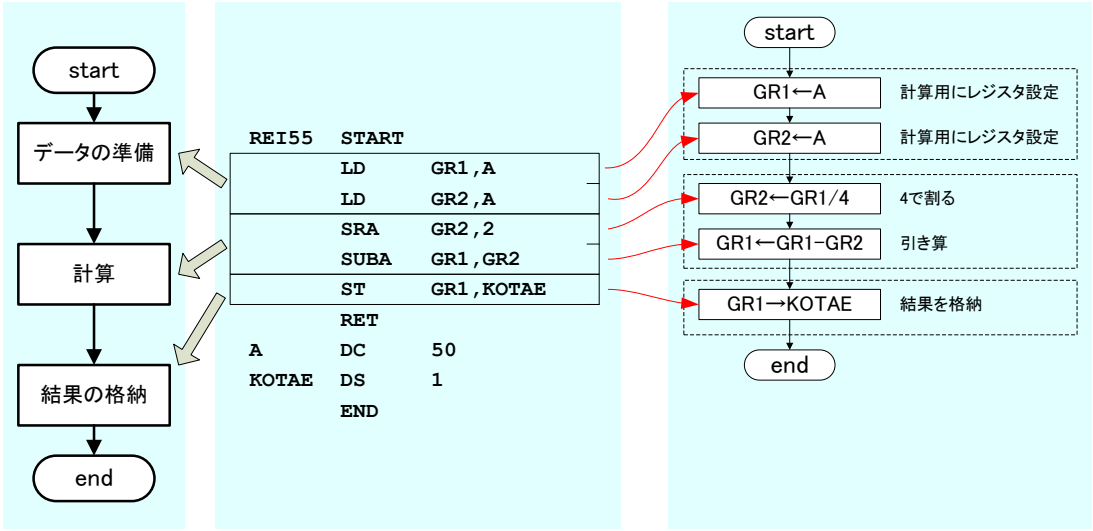

図 3: 教科書の List5-5 のプログラムの構造とフローチャート

4 [例題 6] 繰り返し処理

教科書の List5-6 のプログラムを例にして，繰り返し処理 (ループ) について説明する．以下のことが，ここでの学習の重要なポイントである．

- 比較とジャンプ命令を使うことにより，繰り返し処理が出きる．

4.1 プログラムの基本構造

高級言語のプログラムは，

順次 プログラムの命令は上から下へ実行される．

選択 制御式により実行される分が選択される．これは，FORTRAN や C 言語の if 文のことである．

繰り返し ループあるいは反復とも呼ばれ，同じ命令を繰り返す．FORTRAN では DO 文，C 言語では for 文などである．

の基本構造からなる．いままで，順次は気が付かないで使っていたが，なにも考えることはない．例題 [3] で示した条件分岐 (比較+ジャンプ) が選択の構造である．本日の例題 [6] では，繰り返し文を学習する．

4.2 教科書の例

教科書のプログラムは，

- ラベル DATA から，ラベル KOSUU が示す語数の整数のデータが格納されている．
- このデータの最大値を探し出し，それをラベル MAX が示す領域に格納する．

という問題を解く，プログラムである．このプログラムは，教科書の p.98 の List5-6 に示されている．まず最初に，

- このプログラムの命令とデータの領域の区別

を考える．これは，さんざんやったので理解できているものとする．

このプログラムの核となる部分は，最大値を探すアルゴリズムである．教科書の例では，それは，次のようなアルゴリズムとなっている．

1. 最初に読み込むデータ (アドレス [data]) を暫定最大値とする．
2. それ以降は繰り返し処理.
   (a) 次のデータと最大値を比較する
   (b) 個数分のデータの比較が済んでいなければ，元 (次データ処理) に戻る

である．最大値を探すアルゴリズムには，次々にデータを最大値と比較する処理が必要である．ここに繰り返し処理が使われる．

4.3 繰り返し処理

高級言語では繰り返し専用の命令が用意されているが，アセンブラ言語にはない．そのため，条件分岐を使い繰り返しを行うことにする．アセンブラ言語では，以前学習したように，条件分岐は比較命令 (CPA,CPL) とジャンプ命令 (JMI,JNZ,JZE,JUMP,JPL,JOV) を上手に使って，繰り返し処理を行うことになる．フローチャートで書くと，図 4 の様な構造である．

このような繰り返し構造を実現するためには，一度実行した命令に戻る必要がある．そのために，フローチャートの上へ分岐 (ジャンプ) するのである．このままだと，無限ループに陥るので，そこから抜けるための機構も必要である．パラメーターの値に従いループを続けるか，そこから抜けるかを決める．それは，分岐 (比較とジャンプ命令) で実現できる．

今回の問題であれば，データの個数分だけ繰り返せばよい．そのために，カウンターを用いて，データ数のカウントをしている．これはまた，指標レジスターにも使える．

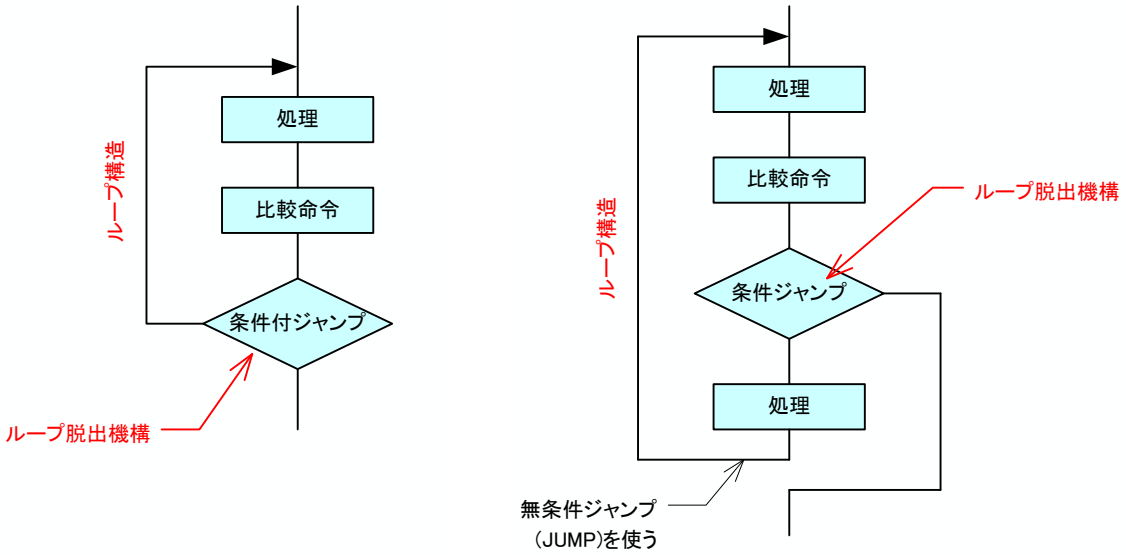

図 4: 分岐命令を使った繰り返し構造

4.4 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 5 に示す．ループ構造になっているのが分かるだろう．それについての説明の前に，データを取り扱うレジスターやラベルの内容を表 1 に示しておく．

表 1: 汎用レジスターとメモリの内容

GR0	読み込んだデータ (比較すべき対象) を入れる．
GR1	データ数から 1 引いた値．指標レジスタの最大値．
GR2	データのカウンタ．0 から始まり，指標レジスタとしてつかう．
DATA	調べるデータの先頭アドレス．
KOSUU	調べるデータ数が書かれているアドレス．
MAX	調べたデータの最大値を入れるアドレス．

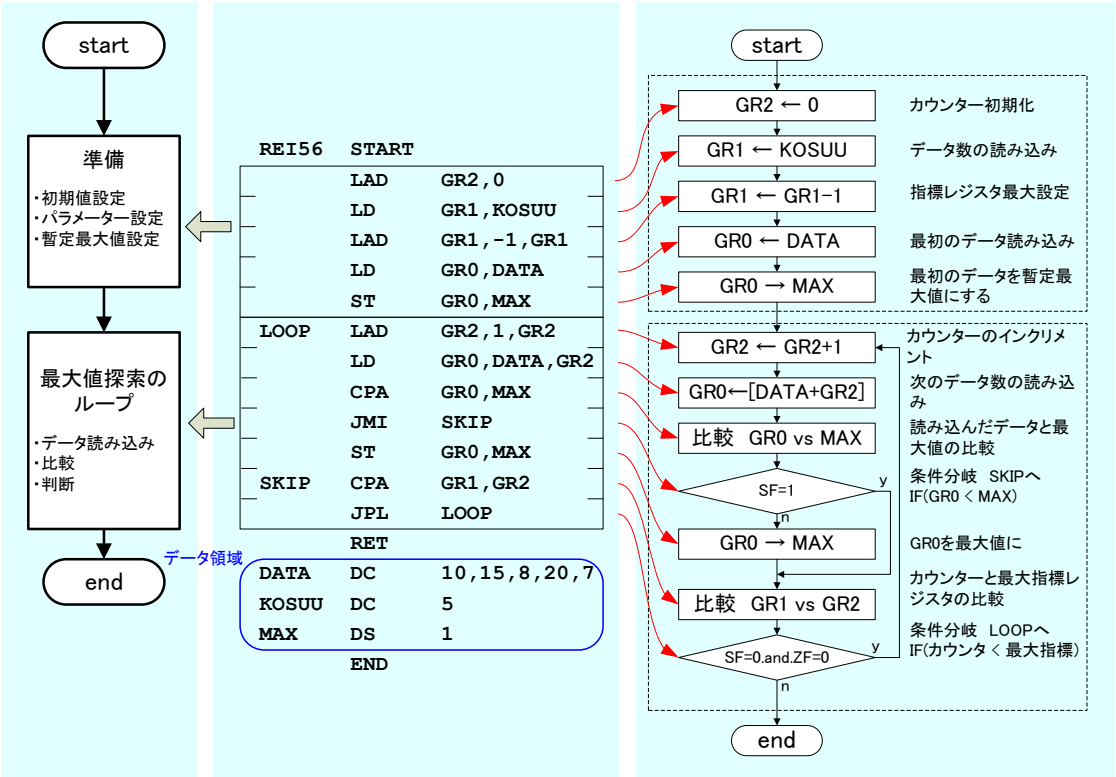

図 5: 教科書の List5-6 のプログラムの構造とフローチャート

5 [例題 7] 繰り返し処理とサブルーチン

教科書の List5-7 のプログラムを例にして，サブルーチンについて説明する．以下のことが，ここでの学習の重要なポイントである．

- サブルーチンの呼び出しと呼び出し元に戻る方法を理解する．
- サブルーチンを使うとプログラムが部品化でき，内容が分かりやすくなる．

5.1 サブルーチンが必要な理由

長いプログラムを，先に示した基本構造だけで作成することは不可能である．技術的には可能であるが，何が書いてあるか全く分からないプログラムになってしまい．メンテナンスが不可能である．そのため，プログラムを機能毎に細かく分割して，分かりやすくする方法がとられる．この機能毎に分割されたプログラムをサブルーチンという．FORTRAN では，`SUBROUTINE` とか `FUNCTION` というものがそれに当たる．ここでは，このサブルーチンを CASL II で実装する方法をである．

5.2 教科書の例

教科書の [例題 6] のプログラムの動作内容は，[例題 5] と全く同じである．ただし，最大値を探索する部分をサブルーチンにして，プログラムの内容を分かりやすくしている．

5.3 サブルーチン

プログラムは，分かりやすく書かなくてはならない．分かりにくいプログラムはメンテナンスが大変である．ここでは，最大値を探す機能をサブルーチンとして分割している．

実際，サブルーチンを作成するとにもっとも気にかけることは，データの受け渡しである[1]．メインルーチンからサブルーチンに，ある処理を依頼するのであるが，そのためにはデータが必要である．メイン → サブ，メイン ← サブと 2 通りある．高級言語ではいろいろな方法があるが，CASL II では汎用レジスターを使うのが一般的である．

例題のプログラムを例にすると，

- メインルーチンがサブルーチンに依頼している仕事の内容は，データの最大値を探すことである．
- そのために，メインルーチンはサブルーチンに，GR1 を用いて，データの個数を渡している．

5.4 プログラムの構造とフローチャート

データを取り扱うレジスターやラベルの内容を表 1 と同じである．また，プログラムのフローチャートを図 6 に示す..

[1] 受け渡しのデータのことを引数と言う．呼び出し側が渡すデータを実引数，呼び出された側が受け取るデータを仮引数と言う

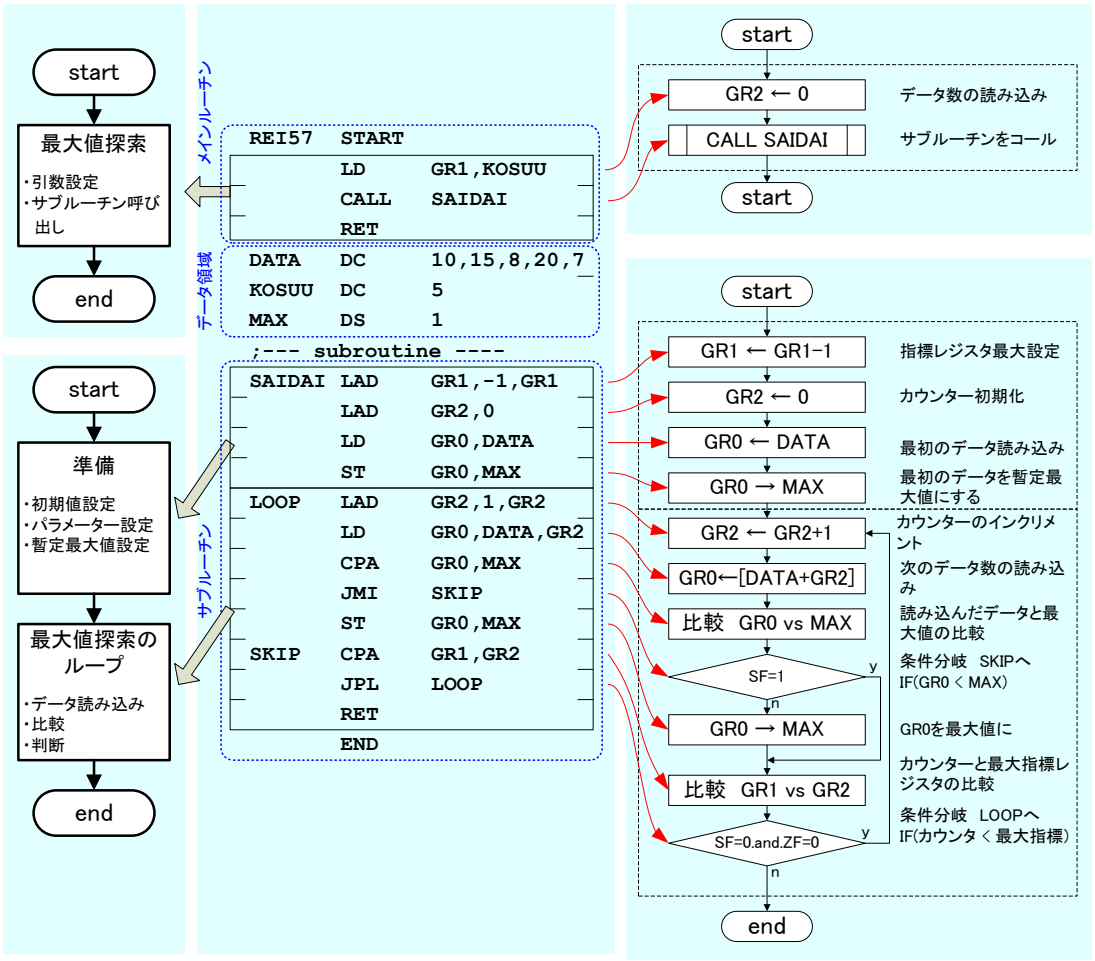

図 6: 教科書の List5-7 のプログラムの構造とフローチャート

6 課題

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．

6.1 問題

最初の 3 問はアドレス修飾とカウンターに関する問題である．次の 3 問はシフト演算，引き続き繰り返しとサブルーチンに関する問いである．

[**問 1**] データの格納 (I)

– データ領域を 3 ワード確保する．

– 確保された領域に，アドレス修飾を利用して，1, 2, 3 と整数を格納する．

[**問 2**] データの格納 (II)

– データ領域を 3 ワード確保する．
– 確保された領域に，アドレス修飾を利用して，2, 4, 6 と整数を格納する．

[**問 3**] データの格納 (III)

– データ領域を 100 ワード確保する．
– 確保された領域に，アドレス修飾を利用して，2, 4, 6,$\cdots$,200 と整数を格納する．ヒント：ジャンプ命令を上手に使うこと．

[**問 4**] データを 8 倍

– ラベル名 DATA が示すメモリーの領域に $(00FF)_{16}$ の値を格納する．
– シフト命令を利用して，この値を 8 倍する．
– 8 倍された値は，ラベル名 KEKKA が示す領域に格納する．

[**問 5**] データを 1/16 倍

– ラベル名 DATA が示すメモリーの領域に $(30000)_{10}$ の値を格納する．
– シフト命令を利用して，この値を 1/16 倍にする．
– 1/16 倍された値は，ラベル名 KEKKA が示す領域に格納する．

[**問 6**] データを 5.75 倍

– ラベル名 DATA が示すメモリーの領域に $(100)_{10}$ の値を格納する．
– シフト命令を利用して，この値を 5.75 倍にする．
– 5.75 倍された値は，ラベル名 KEKKA が示す領域に格納する．

[**問 7**] 1〜1000 までの和を計算するプログラムの作成

– 1〜1000 までの加算は，サブルーチンで実行すること．そして，繰り返し構造を用いた加算であること．
– 加算結果は，メモリーの適当な場所に格納すること．

6.2 レポート提出要領

提出方法は，次の通りとする．

期限　2 月 21 日 (火) PM 5:00
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　プログラム練習 (その 2)」
　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．

参考文献

[1] 東田幸樹, 山本芳人, 広瀬啓雄. アセンブラ言語 CASL II. 工学図書 (株), 2002 年.