CASL IIのプログラム例(その3)

山本昌志*

2006年2月17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 4]〜[例題 7] を学習した．

- **[例題 4] 論理演算とアドレス修飾**
  - アドレスをずらす (オフセット) ために，アドレス修飾が使われる．アドレスと汎用レジスターの GR1〜GR7 をカンマで区切れはよい．(例) `LD  GR1,DATA,GR2`
- **[例題 5] シフト演算**
  - データを左に1ビットシフトさせると2倍される．右に1ビットさせると 1/2 にできる．
  - これを込み合わせて，かけ算や割り算が可能である．演算に使われる数を 2^n の和に分解するのがコツである．ただし，n は負の数も含む．
- **[例題 6] 繰り返し処理**
  - ジャンプ命令とそれを制御するフラグレジスターを使って，繰り返し処理ができる．
- **[例題 7] 繰り返し処理とサブルーチン**
  - サブルーチンはプログラムの部品である．
  - CALL 命令でサブルーチンを呼び出し，RET 命令で呼び出し元へ戻る．

1.2　本日の内容

本日は教科書の p101-114 の以下の内容について学習する．

- **[例題 8] アドレスの受け渡し**
  - 完璧なサブルーチンを目指し，サブルーチンへのデータの渡し方を学習する．

*独立行政法人　秋田工業高等専門学校　電気工学科

- **[例題 9] ラベルを 2 重に付ける方法**
  - DS 命令を使って，同じアドレスに 2 つのラベル名を付ける方法を学習する．
- **[例題 10] 数値データを文字データに変換**
  - メモリ中の整数値のデータを文字のデータとして，メモリーに格納する方法を学習する．これは数値を標準出力 (ディスプレイ) に表示するときの必須テクニックである．

2 [例題 8] アドレスの受け渡し

2.1 サブルーチンを実現するために必要なこと

プログラムの規模が大きくなると，その動作の内容が分かりにくくなる．また，似たような処理が増えてくる．そのため，プログラムを機能毎に分割することが行われている．機能毎に分割した，専用のプログラムを利用するのである．このことにより

- 同じような機能のプログラムは 1 回だけ書いて，それをプログラムのいろいろな場所から呼び出して何回もう．
- プログラムが機能毎に分かれているので，ソースが分かりやすくなる．

というようなメリットが生じる．この機能毎に分割したプログラムの単位をサブルーチンと言う．いわゆるプログラムの部品みたいなものである．一般には，サブルーチンは簡単な機能を持つ単純な構造にする．複雑な動作をするものが一つあるよりも，単純なものを組み合わせて複雑な動作をさせる方が簡単である．

実際のプログラムでは，図 1 の様な構造となる．一番左の START から END のラインがメインルーチンである．そして，処理 A から処理 D のラインが，それぞれサブルーチンとなっている．このようにサブルーチンが集まり，プログラムができあがる．実際の動作は，呼び出されたサブルーチンが順次動作し，プログラムが実行される．

サブルーチンはプログラムの部品であると先程述べた．それが部品となるためには，他のプログラムにも転用可能である必要がある．ボルトは機械要素の部品である．これは，自動車にも使えるし，飛行機，冷蔵庫などあらゆるものに使える．自動車にしか使えないボルトとなると用途が限られ，部品としては役割が小さくなる．部品として有用になるためには，汎用性が重要となる．サブルーチンも同じで，これが有用となるためにはどのプログラムでも使えるようにしなくてはならない．

このような観点から，教科書 [1] の [例題 7] のプログラムを見る．そのプログラムはリスト 1 の通りで，最大値を検索サブルーチンを使ったプログラムとなっている．9〜20 行がサブルーチンである．このサブルーチンの部分を他のプログラムで使おうとすると，DATA と MAX の部分を書き直す必要がある．プログラム毎に書き直す必要があるので汎用性があるとは言い難い．あまり良いサブルーチンとは言えない．

サブルーチンのプログラムを書き換えることなく，他のプログラムでも使えるようにしなくてはならない．そのためには，重要なデータ，ここでは DATA や KOSUU，MAX を引数として受け渡しすればよい．実際のプログラム方法は，教科書の List5-8 で学習する．

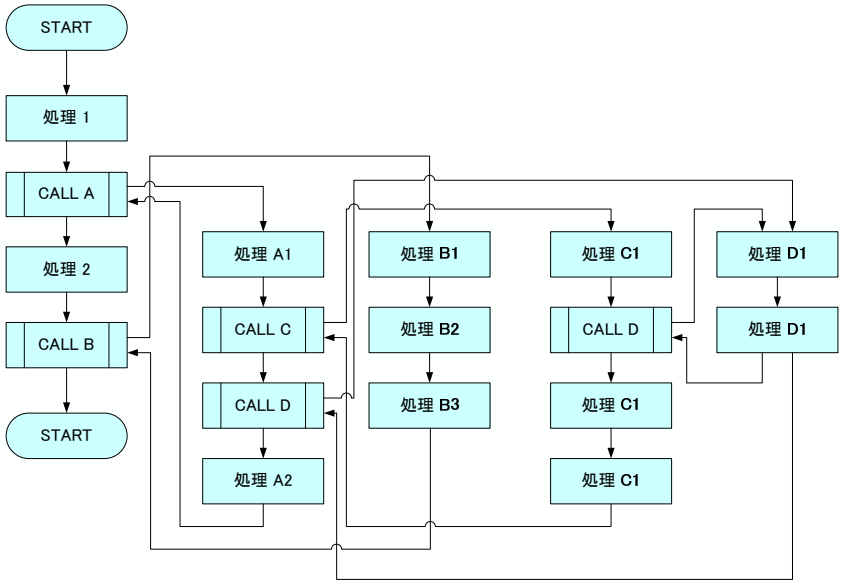

図 1: サブルーチンを使ったプログラムの構造

リスト 1: サブルーチンを使ったプログラム例

```
1  REI57     START
2            LD      GR1, KOSUU          ;データ数をGR1に入れる
3            CALL    SAIDAI              ;サブルーチンの呼び出し
4            RET
5  DATA      DC      10,15,8,20,7
6  KOSUU     DC      5
7  MAX       DS      1
8  ;サブルーチン化した部分
9  SAIDAI    LAD     GR1,-1,GR1
10           LAD     GR2,0               ;カウンタを0に
11           LD      GR0,DATA            ;先頭データ読み込み
12           ST      GR0,MAX             ;暫定最大値の保存
13 LOOP      LAD     GR2,1,GR2           ;カウンタのインクリメント
14           LD      GR0,DATA,GR2        ;データの読み込み
15           CPA     GR0,MAX             ;最大値と比較
16           JMI     SKIP                ;GR0が小さいときSKIP
17           ST      GR0,MAX             ;最大値保存
18 SKIP      CPA     GR1,GR2             ;データ数とカウンタを比較
19           JPL     LOOP                ;カウンタが小さいときLOOP
20           RET
21           END
```

2.2 データの渡し方

教科書の p.102 にちょっと分かりにくいプログラムが載せてある．C 言語が分かる人はここで何が言いたいか分かるが，諸君は学習していないのであまりにも唐突に思えるだろう．分からない人は，気にしないで

も良い．

サブルーチンにデータを渡す方法は，2 種類ある．**アドレス渡し**と**値渡し**である．図 2 のようにメモリーにデータがあるとする．アドレス#0042 のデータ#1234 をサブルーチンで処理する場合，それを渡す必要がある．この場合，

アドレス渡し 処理するデータのアドレス#0042 をサブルーチンに渡す．この場合，サブルーチンはアドレスを知ることが出きるので，#0042 番地の内容を書き換えることができる．

値渡し 処理するデータの値#1234 をサブルーチンに渡す．この場合，サブルーチンはアドレスが分からないので，#0042 番地の内容を書き換えることができない．

である．いずれにしても，サブルーチンは，処理すべきデータ#1234 を知ることができるが，アドレス#0042 の内容を書き換えることができるか否かが異なる．

どちらの方法を使うかは，処理の内容により異なる．諸君が経験を積めば，どちらを使うべきか分かるだろう．ここでは，そこまで踏み込まないことにする．

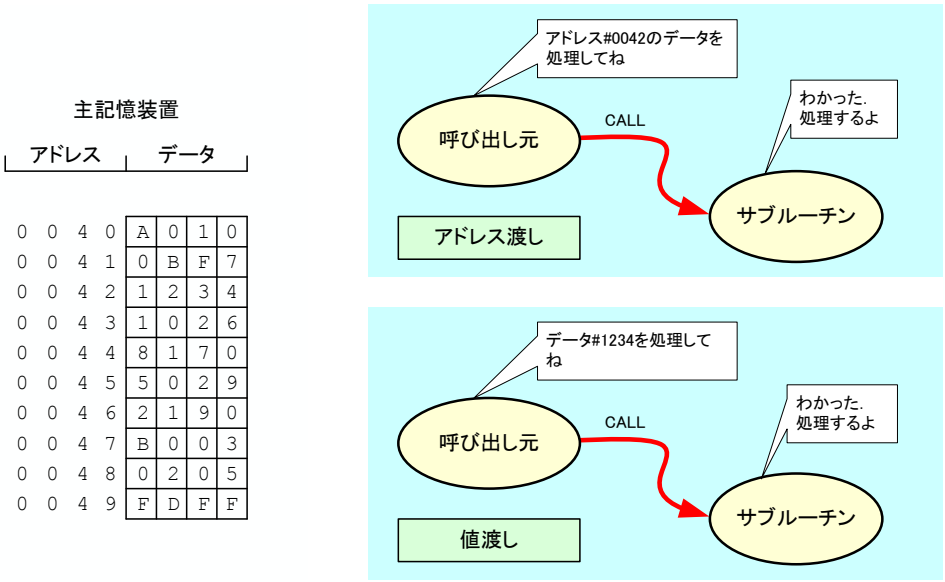

図 2: サブルーチンへ処理すべき内容を渡す方法

2.3 教科書の例

教科書の [例題 8] は，[例題 6] や [例題 7] 同様，最大値データの最大値を求めるプログラムである．これらのプログラムで解く問題は，

- ラベル DATA が示すアドレス以降に，整数のデータが格納されている．
- 格納されているデータ数は，ラベル KOSUU が示すアドレスに格納されている．

- 格納されている整数データの最大値を捜し，ラベル MAX が示すアドレスに格納する．

である．同じ問題を解いているが，それぞれプログラムの方法が異なっており，少しずつプログラムが以下のように進化している．

[**例題 6**] メインルーチンだけで最大値を求めているため，プログラムの処理が分かりにくいプログラムとなっている．

[**例題 7**] サブルーチンを使って分かりやすいプログラムになっているが，サブルーチンの汎用性が無い．

[**例題 8**] サブルーチンが汎用性があり，他のプログラムでも修正無しで使えるようになっている．

すでに，[例題 7] までの学習は完了している．ここでは，[例題 7] から [例題 8] への進化の過程を学習する．ここでの進化は，最大値を捜す関数の独立性を高め，サブルーチンを書き換えることなく，他のプログラムに移植可能にしたことである．それを可能にするためには，サブルーチン内では，DATA や KOSUU，MAX という文字を書かないことである．しかし，これらはサブルーチンでの処理上，極めて重要なデータとなっているので，そのデータの内容はサブルーチンに伝える必要がある．このような場合，引数を使うのが常套手段である．教科書の List 5-8 では，次節に示すように，汎用レジスターを使ってデータの受け渡しをしている．このようにすると，呼び出し側でレジスターに入れる変数を変えるだけで，サブルーチンはどのようなルーチンからでも呼び出しができるようになる．

汎用的なサブルーチンは，どんなプログラムからでも呼び出しができるようにしなくてはならない．[例題 10] の List5-10 では，割り算のサブルーチンが使われており，これは頻繁に使われるので，レジスターを使ってデータを渡している．どんな呼び出し元であれ，レジスターの値を変えるだけで，サブルーチンに処理を依頼できる．

2.4 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 3 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 4 に示しておく．

汎用レジスターの GR1 と GR2，GR3 を使って，サブルーチンで処理に必要なデータ DATA や KOSUU，MAX を送っている．こうすると，他のルーチンからもデータの先頭アドレスと個数，最大値を格納するアドレスを送れば，このサブルーチンが処理できる．ほかのルーチンでのラベル名が異なっても，このサブルーチンが使えるのである．

サブルーチンでは，GR1〜GR5 を使って処理をしている．サブルーチンが終了するときには，メインルーチンで使っているこれらのレジスターを元に戻さなくてはならない．サブルーチンがメインルーチンの動作に関与する可能性が生じるからである．そのために，次のような動作が必要である．

- サブルーチンでは，最大値を求める処理に先立って，レジスターの値をスタック領域に待避させる．
- 最大値を求める処理が終了した後，メインルーチンに戻る前に，スタック領域に待避させたレジスターの値を元に戻す．

これらのことを，PUSH と POP 命令を使って，処理している．

表 1: 汎用レジスターとメモリの内容

	メインルーチン	サブルーチン
DATA	データの先頭アドレス	未使用
KOSUU	データ数が格納されたアドレス	未使用
MAX	最大値を格納するアドレス	未使用
GR1	データ数	データ数-1
GR2	データの先頭アドレス	比較するデータのアドレス
GR3	最大値を格納するアドレス	最大値を格納するアドレス
GR4	未使用	カウンター (比較済みデータ数-1)
GR5	未使用	比較するデータ

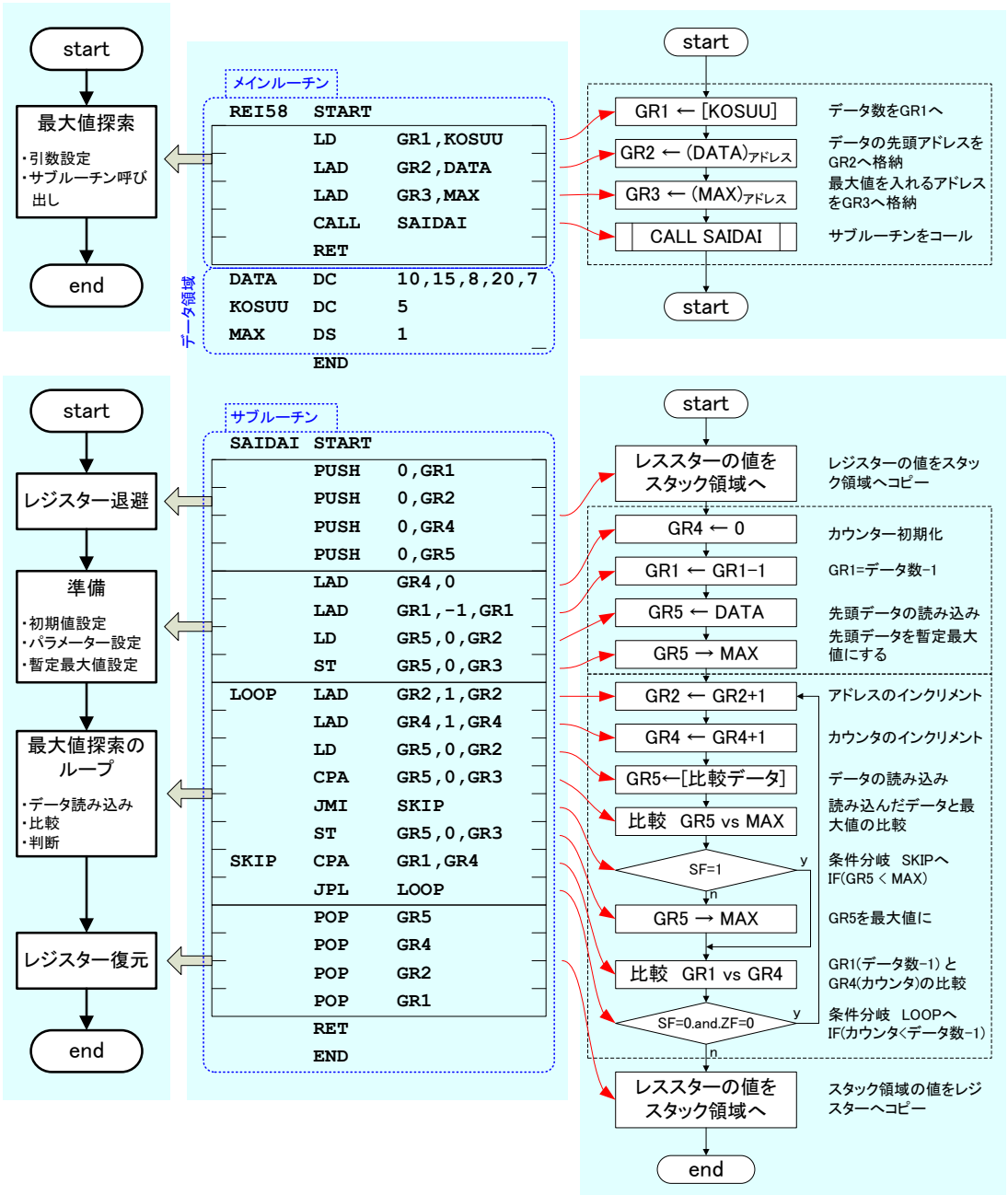

図 3: 教科書の List 5-9 のプログラムの構造とフローチャート

3　[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．

3.1　教科書の例

教科書の [例題 6](p.97)〜[例題 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，

- 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．

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 4 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 4 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 2: 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

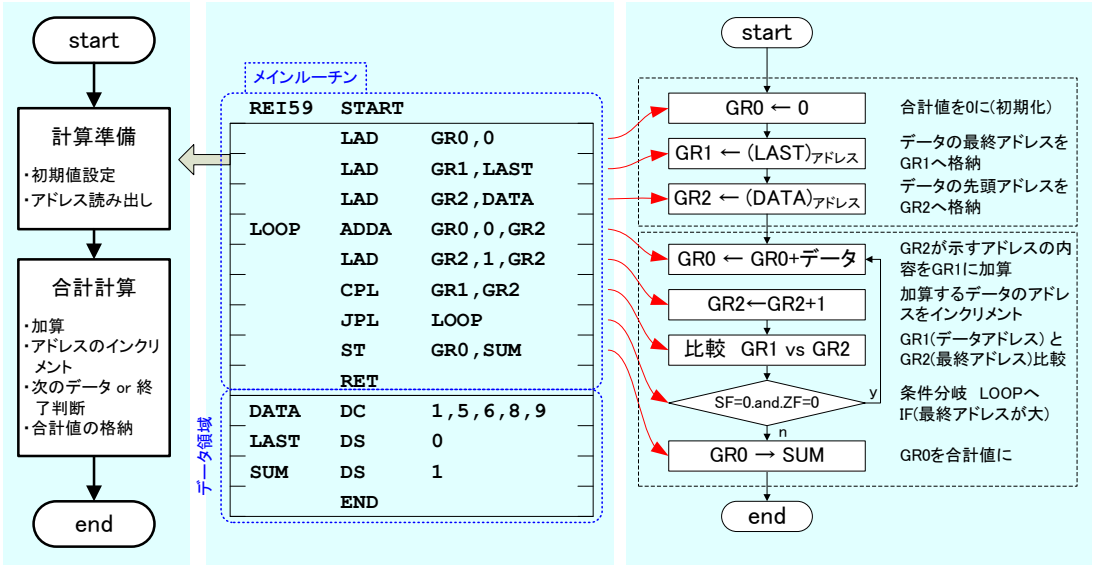

図 4: 教科書の List5-9 のプログラムの構造とフローチャート

4　[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．

4.1　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．

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 5: 教科書の List5-10 の数値を文字に変換

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．

表 3: 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 4: 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

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．

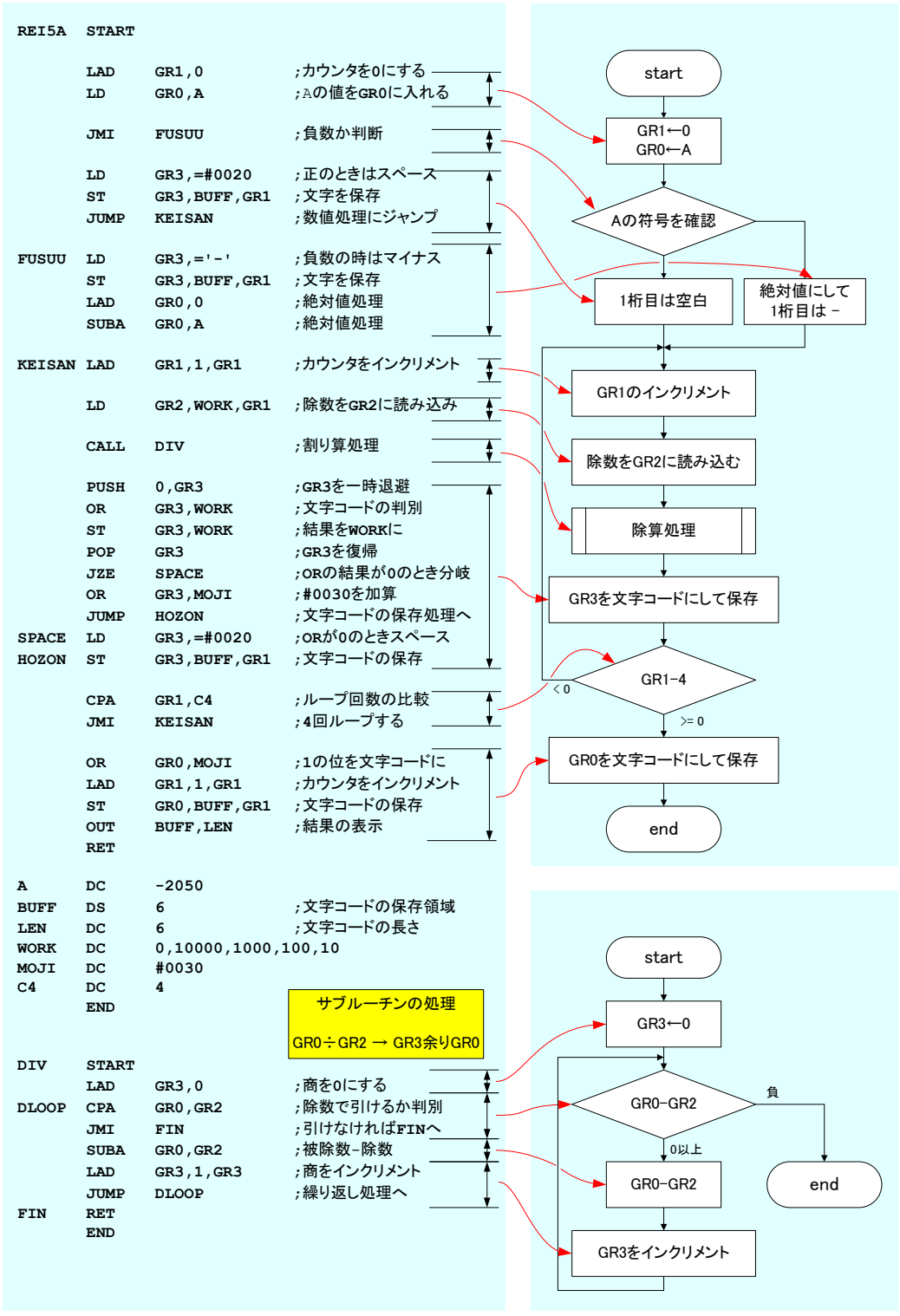

図 6: 教科書の List5-10 のプログラムとフローチャート