4 条件による処理

4-1 処理の流れを変える準備
4-2 処理の流れを変える命令
4-3 処理の流れを複数に分ける

4-1 処理の流れを変える準備(0)

```
 1:PRG0401  START
 3:         LD GR1,=100
 4:         SUBL GR1,=200       ;OF=1.SF=1
 5:         LD GR1,=0           ;ZF=1
 6:         LAD GR1,300         ;FRの値は設定されない
 7:         LD GR1,=400
 9:         LAD GR1,10
10:         CPL GR1,=10         ;GR1と10を論理比較
11:         CPL GR1,=9          ;GR1と9を論理比較
12:         LAD GR1,11
13:         CPL GR1,GR2         ;GR1とGR2を論理比較
15:         LAD GR1,-10
16:         CPA GR1,=-10        ;GR1と-10を算術比較
17:         CPA GR1,=-9         ;GR1と-9を算術比較
18:         CPA GR1,=-11        ;GR1と-11を算術比較
20:         LAD GR1,10
21:         CPL GR1,=#FF00      ;GR1と#FF00を論理比較
22:         CPA GR1,=#FF00      ;GR1と#FF00を算術比較
24:         RET
25:         END
```


実行し、確認しましょう！！

【例題】実行結果をwebclassにて、解答せよ。

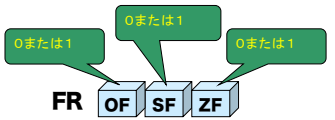
	汎用レジスタ		フラグレジスタ		
	GR0	GR1	OF	SF	ZF
4行目実行後					
10行目実行後					
16行目実行後					

復習

■ フラグレジスタとオーバフロー(1)

- 演算結果が範囲を超えてしまうことをオーバフローという.
- オーバフローの発生の有無を報知するためにオーバフローフラグ(OF)というレジスタが用意されている.

- COMET II システムには,汎用レジスタの他にフラグレジスタ(FR)と呼ばれるレジスタがある.
- FRは,演算結果のいろいろな状態を示し,演算命令などの命令が実行されるたびに,結果に応じた値が自動的に設定される.



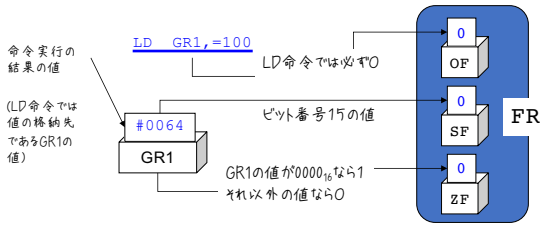
OF:「演算結果がオーバフロー」したら 1，しなければ 0
SF:「演算結果の符号ビットが1」なら 1，でなければ 0
ZF:「演算結果が0」なら 1，でなければ 0

4-1 処理の流れを変える準備(2)

■ フラグレジスタ

【3行目】 LD GR1,=100

- オーバフローフラグ(OF):オーバフローが発生すると1が設定されるフラグである.
  LD命令では,常に0が設定される.
- サインフラグ(SF):演算結果の符号ビットの値が設定されるフラグである.
  符号ビットの値に0が設定される.
- ゼロフラグ(ZF):演算結果がゼロの場合にのみ,ZFが1に設定される.
  0064_{16}は0ではないので,ZFには0が設定される.



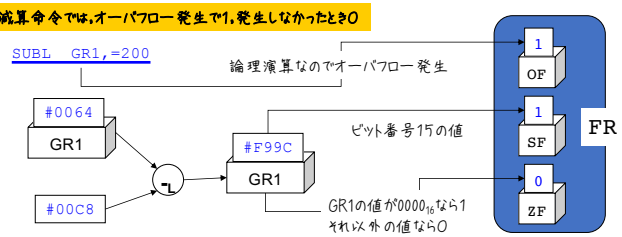
4-1 処理の流れを変える準備(3)

■ フラグレジスタ(2)

【4行目】 SUBL GR1,=200

- オーバフローフラグ(OF):論理減算は,符号なし数値に対する減算命令である.従って,
  0～65535を超え,オーバフローが発生したので1が設定される.
- サインフラグ(SF):演算結果$FF9C_{16}$のビット番号15の値1が設定される.
- ゼロフラグ(ZF):演算結果($FF9C_{16}$)は0000_{16}ではないので,ZFには0が設定される.

加減算命令では,オーバフロー発生で1,発生しなかったとき0



4-1 処理の流れを変える準備(4)

■ フラグレジスタ(3)

【6行目】 LAD GR1,300

- LAD命令がFRを設定しない命令だからである.LAD命令を実行しても,FRの値は変化しない.
- FRの値は6行目実行後の値のままであり,すべての命令がFRを変化させるわけではなく,LAD命令のようにFRの値を変化させない命令もある.

命令	書き方		命令の説明	FRの設定
	命令コード	オペランド		
ロード	LD	r1,r2	r1 ← (r2)	○
		r,adr[,x]	r ← (adr)	
ロードアドレス	LAD	r,adr[,x]	r ← 実行アドレス	−
論理減算	SUBL	r1,r2	r1 ← (r1) - L(r2)	○
		r,adr[,x]	r ← (r) - L(実効アドレス)	

4-1 処理の流れを変える準備(5)

■ フラグレジスタの特徴と機能

- FRに値を設定する命令(LD命令,SUBA命令など)と,設定しない命令(LAD命令など)がある.
- オーバフローフラグ(OF)は,加減算命令とシフト命令で,演算結果に応じて設定される.それ以外の命令では,常に0が設定される.
- サインフラグ(SF)には,ビット番号15(符号ビット)の値が設定される.結果の値を符号つき数値と見たとき,負の値かどうかを示すフラグである.
- ゼロフラグ(ZF)は,結果の値が0000_{16}かどうかを示すフラグである.結果が0のとき,ZFには1が設定される.

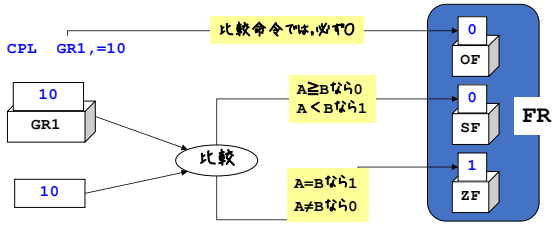
4-1 処理の流れを変える準備(6)

■ 比較命令(CPL)

【10行目】　CPL GR1,=10

- CPL命令は論理比較命令である.
- 論理比較命令ではオペランドで指定された2つの値の大小を符号なし数値として比較し,その結果に応じて,FRに値を設定する.

- CPL命令は以下のように設定される.
  - OF＝0　(比較命令では常に0が設定される)
  - SF＝0　(最初の値が大きいか等しい場合0が設定される)
  - ZF＝1　(同じ値の場合1が設定される)



4-1 処理の流れを変える準備(7)

■ 比較命令(CPA)

【16-18行目】　CPA GR1,=-10

- CPA命令は,算術比較命令であり,比較する数値を符号つき数値として扱う点,以外は,CPL命令(算術比較命令)と同じである.
- 各行とも,GR1の値-10とリテラルの値を比較している.

	比較する値		比較結果		
	GR1	リテラル	OF	SF	ZF
17行目	-10	-10	0	(-10) - (-10) = 0 ≧ 0 なので 0	1
18行目	-10	-9	0	(-10) - (-9) = -1 ＜ 0 なので 1	0
19行目	-10	-11	0	(-10) - (-11) = 1 ≧ 0 なので 0	0

4-1 処理の流れを変える準備(8)

■ CPL命令およびCPA命令の仕様

- 比較命令には符号なし数値として比較を行う論理比較命令と,符号つき数値として比較を行う算術比較命令がある.

命令	書き方		命令の説明
	命令コード	オペランド	
算術計算 ComPare Arithmetic	CPA	r1,r2 r,adr[,x]	(r1)と(r2),又は(r)と(実効アドレス)の算術比較又は論理比較を行う.比較結果によってFRに次の値を設定する
論理比較 ComPare Logical	CPL	r1,r2 r,adr[,x]	

比較結果	FRの値 SF	ZF
(r1) ＞ (r2) (r1) ＞ (実効アドレス)	0	0
(r1) ＝ (r2) (r1) ＝ (実効アドレス)	0	1
(r1) ＜ (r2) (r1) ＜ (実効アドレス)	1	0

4-2 処理の流れを変える命令 (0)

```
1:PRG0402   START
2:          RPUSH
3:;
4:          LAD GR1,10
5:          CPA GR1,=10      ;GR1は10と等しいか？
6:          JZE FIN          ;もし等しいならFINへジャンプ
7:          LD GR2,GR1       ;等しくないならGR2にコピー
8:FIN       RPOP
9:          RET
10:         END
```

4-2 処理の流れを変える命令(1)

■ 分岐命令とは

【4-5行目】

➡ 両者は等しいので,ZFのみが1に設定され,SFとOFには0が設定される.

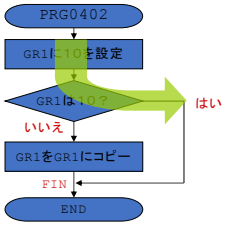
【6行目】 JZE FIN

➡ JZE命令(零分岐命令)である.

➡ JZEは,ZFが1のとき,指定されているラベル(ここではFIN)のある行にジャンプする命令である.

➡ このように,ある条件に応じて別の行に処理をジャンプ(分岐)させることを,条件分岐という.

```
4：          LAD GR1, 10
5：          CPA GR1, =10
6：          JZE FIN
7：          LD GR2, GR1
8：FIN       RPOP
```



153

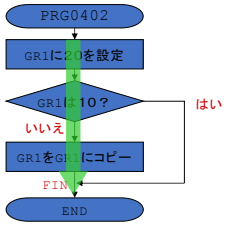
153

4-2 処理の流れを変える命令(2)

■ GR1の値が10以外の値であった場合

➡ ZFが0のままなので,JZE命令では分岐が起こらない.

```
1：PRG0402  START
2：         RPUSH
3：;
4：         LAD GR1, 20
5：         CPA GR1, =10
6：         JZE FIN
7：         LD GR2, GR1
8：FIN      RPOP
9：         RET
10：        END
```



154

154

4-2 処理の流れを変える命令(3)

■ その他の分岐命令

➡ 全部6種類がある

命令	書き方		命令の説明
	命令コード	オペランド	
正分岐 Jump on PLus	JPL	adr[,x]	FRによって,実効アドレスに分岐する.分岐しないときは,次の命令に進む.
負分岐 Jump on Minus	JMI	adr[,x]	
非零分岐 Jump on Non Zero	JNZ	adr[,x]	
零分岐 Jump on ZEro	JZE	adr[,x]	
オーバフロー分岐 Jump on OVerflow	JOV	adr[,x]	
無条件分岐 Unconditional JUMP	JUMP	adr[,x]	無条件に実効アドレスに分岐する.

命令	分岐するときのFRの値		
	OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

155

155

問題[4-2]

■ 二つの2桁の値を,各々,GR1とGR2に設定し(リテラル使用),CPA命令によりGA1とGR2を比較し,その結果が(値)が正であれば,'PLUS'を,負であれば,'MINUS'と表示するプログラムを作成しなさい。
なお、自分の学籍番号が奇数の場合は(1)の値を、偶数の場合は(2)の値を、選択する。

(1) GR1=20, GR2=10

(2) GR1=10, GR2=20

156

156

4-3 処理の流れを複数に分ける(0)

- ✓ プログラムPRG0403は，入力装置から入力された文字に応じたメッセージを，出力装置に出力するプログラムである．
- ✓ 文字'A'，文字'B'，文字'C'が入力されたら，それぞれ 'Apple'，'Blue'，'Clear'というメッセージを出力する．
- ✓ その他の文字の場合には'Other'と出力する．

4-3 処理の流れを複数に分ける(1)

```
 1:PRG0403  START
 4:          IN   DATA,LEN        ;文字を読み込む
 5:          LD   GR1,DATA
 6:          CPL  GR1,='A'        ;文字は'A'か?
 7:          JNZ  CHKB
 8:          OUT  MSGA,LENA
 9:          JUMP FIN
10:CHKB      CPL  GR1,='B'        ;文字は'B'か?
11:          JNZ  CHKC
12:          OUT  'MSGB'LENB
13:          JUMP  FIN
14:CHKC      CPL  GR1,='C'        ;文字は'C'か?
15:          JNZ  OTHER
16:          OUT MSGC,LENC
17:          JUMP  FIN
18:OTHER     OUT MSGO,LENC        ;その他の文字
19:FIN      RET
22:LEN       DC    1
23:DATA      DS    256
24:LENA      DC    5
25:MSGA      DC    'Apple'
26:LENB      DC    4
27:MSGB      DC    'BLUE'
28:LENC      DC    5
29:MSGC      DC    'Clear'
30:LENO      DC    5
31:MSGO      DC    'OTHER'
32:          END
```

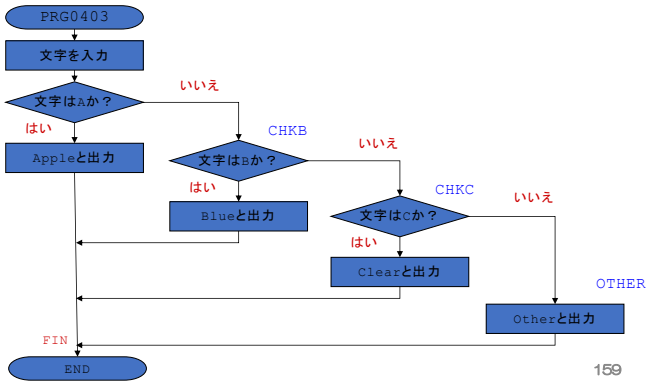
```
IN ? A
Apple

IN ? E
Other
```


4-3 処理の流れを複数に分ける(2)

■プログラムの流れ図

➡分岐命令を何段も置くことによって，処理の流れを何種類かに分けることができるようになる．



問題[4-3]

■GR1とGR2に，各々，16進数の7と16進数のAが格納されているとする．プログラム実行時に，以下の1文字をコマンドとして入力すると，それに応じた処理を行ってGR3に格納するプログラムを作成しなさい．

入力文字	処理
「M」	GR1とGR2の論理積をGR3に格納
「A」	GR1とGR2の論理和をGR3に格納
「X」	GR1とGR2の排他的論理和をGR3に格納
その他	GR1の値をGR3に格納

5 繰り返しの処理

5-1 決まった回数の繰り返し処理
5-2 ある条件になるまで繰り返す処理
5-3 判定の繰り返し処理

5-1 決まった回数の繰り返し処理(1)

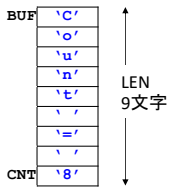
✓ 8から始めて1ずつ値を減らしながら1まで繰り返し出力するプログラムである

```
 1 PRG0501 START
 2:;       RPUSH
 3:;
 4:         LAD  GR1,8       ;GR1初期化
 5:LOOP     LD   GR2,='0'    ;繰り返し処理の始まり
 6:         ADDA GR2,GR1      ;数値8を文字'8'に変換
 7:         ST GR2,CNT        ;出力用バッファに追加
 8:         OUT  BUF,LEN      ;出力
 9:         SUBA GR1,=1       ;デクリメント
10:         JNZ LOOP          ;繰り返し処理を続けるかどうか判定
11:;
12:         RPOP
13:         RET
14:;
15:LEN      DC  9
16:BUF      DC 'Count='
17:CNT      DC ' '
18:         END
```


5-1 決まった回数の繰り返し処理 (3)

■ プログラムの動き(2)

【7行目】　ST　　GR2,CNT
【8行目】　OUT　　BUF,LEN



出力

```
Count = 8
```


5-1 決まった回数の繰り返し処理 (4)

■ プログラムの動き(3)

【9,10行目】　SUBA　GR1,=1

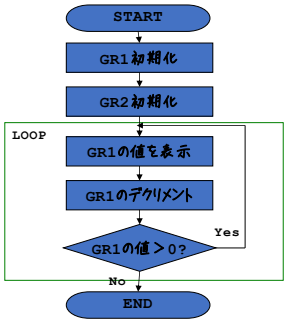
- GR1を1だけ減らし,その結果がゼロであるかを,ZFを使って10行目で判定している.
- 9行目のように,値を決まった数(たいていは1)ずつ減らしていくことを,デクリメントするという.
- 値を決まった数(たいていは1)ずつ増やしていく場合は,インクリメントするという.

	出力	GR1
ループ1回目	Count = 8	8 → 7
ループ2回目	Count = 7	7 → 6
ループ3回目	Count = 6	6 → 5
ループ8回目	Count = 1	1 → 0

5-1 決まった回数の繰り返し処理 (5)

■ プログラムの流れ図と後判定

- 後判定：このように繰り返し処理の最後に判定を行う
- 前判定：この他に繰り返し処理の最初で判定を行う



5-2 ある条件になるまで繰り返す処理(0)

✓ 繰り返し処理の方法について学ぶ.

✓ プログラムPRG0502は,1＋2＋3＋・・・と順に数を足していき,合計が100を超えるまで足し算を続けるプログラムである.

```
 1:PRG0502 START
 2:        RPUSH
 3:;
 4:        LAD     GR1,1         ; 足す数の初期化
 5:        LAD     GR2,0         ; 合計の初期化
 6:LOOP    ADDA    GR2,GR1       ; 繰り返し処理の開始
 7:        LAD     GR1,1,GR1     ; インクリメント
 8:        CPA     GR2,=100      ; 合計が100より小さいか？
 9:        JMI     LOOP          ; 繰り返し処理を続けるか判定(後判定)
10:;
11:        RPOP
12:        RET
13;
14:        END
```


5-2 ある条件になるまで繰り返す処理(1)

	GR0	GR1	GR2	OF	SF	ZF	
(a) 5行目実行後	0000	0001	0000	0	0	0	初期化
(b) 1回目の6行目実行後	0000	0001	0001	0	0	0	繰り返し処理1回目
(c) 1回目の7行目実行後	0000	0002	0001	0	0	0	
(d) 1回目の8行目実行後	0000	0002	0001	0	1	0	
(e) 2回目の6行目実行後	0000	0002	0003	0	0	0	繰り返し処理2回目
(f) 2回目の7行目実行後	0000	0003	0003	0	0	0	
(g) 2回目の8行目実行後	0000	0003	0003	0	1	0	
(h) 3回目の6行目実行後	0000	0003	0006	0	0	0	繰り返し処理3回目
(i) 3回目の7行目実行後	0000	0004	0006	0	0	0	
(j) 3回目の8行目実行後	0000	0004	0006	0	1	0	
(k) 13回目の6行目実行後	0000	000D	005B	0	0	0	繰り返し処理13回目
(l) 13回目の7行目実行後	0000	000E	005B	0	0	0	
(m) 13回目の8行目実行後	0000	000E	005B	0	1	0	
(n) 14回目の6行目実行後	0000	000E	0069	0	0	0	繰り返し処理14回目
(o) 14回目の7行目実行後	0000	000F	0069	0	0	0	
(p) 14回目の8行目実行後	0000	000F	0069	0	0	0	

5-2 ある条件になるまで繰り返す処理(2)

■ プログラムの動き

【6-9行目】 LOOP ADDA GR2,GR1

- 6行目が繰り返し処理の始まりである.
- 6行目では,ADDA命令により,GR2にGR1の値を加算している.つまり,足す数であるGR1の値を合計であるGR2に足している.
- 7行目では,LAD命令により,GR1の値に1を足した値を,GR1に再設定している.
- これは,つまりGR1をインクリメントすることを意味している.
- 8行目では,CPA命令によりGR2と100を比較している.
- 9行目では,GR2が100より小さかった場合,ラベルLOOPの6行目に戻る.
- こうして,GR2が100に達するまで,6～9行目の処理が繰り返される.

5-2 ある条件になるまで繰り返す処理(3)

■ 繰り返し処理の進行と終了条件

【6-9行目】

- 8行目では,GR2と100が比較され,GR2が100を超えたので,SFが0に設定される[実行のようす(p)].
- 9行目では,JMI命令でLOOPに分岐することなく,10行目にすすむ.そしてプログラムの実行が終了する.

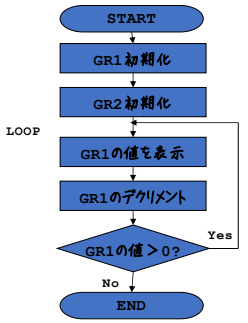
	GR1	GR2
ループ1回目	1 → 2	0+1=1
ループ2回目	2 → 3	0+1+2=3
ループ3回目	3 → 4	0+1+2+3=6
⋮	⋮	⋮
ループ14回目	3 → 4	0+1+2+3+…+15=105

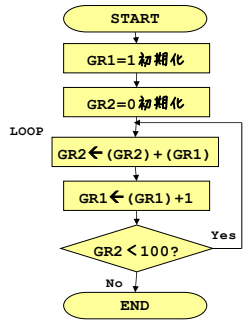
100を超えたのでループから抜ける

5-2 ある条件になるまで繰り返す処理(4)

■プログラムの流れ図

- 繰り返し回数で終了判定を行うのではなく,代わりに演算結果の値が判定に使われていることである.





問題[5-2]

■7の倍数のうち,100をはじめて超える数をGR1に求めるプログラムを作成しなさい(7, 14,21…と増やしていって,100を超えるところを見つける).

5-3 前判定の繰り返す処理(0)

✓ ここでは、前判定による繰り返し処理について学ぶ.

```
 1:PRG0503 START
 4:        LAD     GR1,ORG         ; もとの文字列を指すGR1を初期化
 5:        LAD     GR2,DST         ; 新しい文字列を指すGR2を初期化6:
 6:        LAD     GR3,0           ; 処理した文字数
 7:LOOP    CPA     GR3,ORGLEN      ; 繰り返し処理の開始文字列の最後まで来たか？
 8:        JZE     FIN             ; 繰り返し処理を続けるか判定(前判定)
 9:        LD      GR4,0,GR1       ; 文字を取ってくる
10:        CPA     GR4,='0'        ; 文字は'0'より小さいか？
11:        JMI     LOOPE
12:        CPA     GR4,='9'        ; 文字は'9'より大きいか？
13:        JPL     LOOPE
14:        ST      GR4,0,GR2       ; 新しい文字列に文字を追加
15:        LAD     GR2,1,GR2       ; 新しい文字列を指すGR2をインクリメント
16:LOOPE     LAD     GR1,1,GR1     ; もとの文字列を指すGR1をインクリメント
17:        ADDA    GR3,=1          ; 処理した文字数のインクリメント
18:        JUMP    LOOP
20:FIN     OUT     ORG,ORGLEN      ; もとの文字列を表示
21:        LAD     GR1,DST
22:        SUBL    GR2,GR1
23:        ST      GR2,DSTLEN      ; 新しい文字列の長さを求める
24:        OUT     DST,DSTLEN      ; 新しい文字列の表示
26:        RET
28:ORG     DC      '1A2BC34E5FG'
29:ORGLEN  DC      11
30:DST     DS      20
31:DSTLEN  DS      1
33:        END
```

```
1A2BC34E5FG
12345
```

5-3 前判定の繰り返す処理(1)

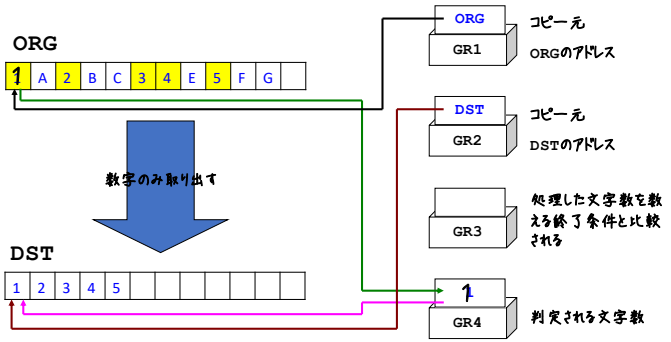
- プログラムPRG0503は，
  - 与えられた英数字文字列から，実行結果のように数字だけを取り出して表示するプログラムである．
  - 与えられた文字列はラベルORGで示された領域に格納されており，ここから図5.6のように数字だけを取り出してラベルDSTの領域に格納する．
  - この取り出した数字を表示する．

■ プログラムの動き(1)

- LAD命令によって，ラベルORGのアドレスをGR1に設定している．
- ラベルORGは28行目に定義されており，主記憶内に文字列'1A2BC34E5FG'(以降では，単に「もとの文字列」と呼ぶ)を格納している．
- つまりこの時点では，もとの文字列ORGの先頭の文字'1'のアドレスが，GR1に設定されている．
- この先，GR1は「もとの文字列の何番目の文字か」を指す．

5-3 前判定の繰り返す処理(3)

■ 図5.6 PRG0503の動作

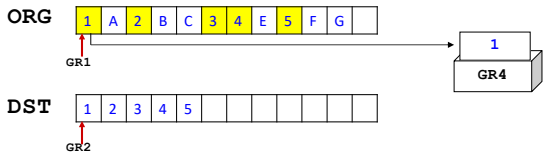


5-3 前判定の繰り返す処理(4)

■ プログラムの動き(2)

```
【7,8行目】   LOOP    CPA     GR3,ORGLEN
                      JZE     FIN
```

- 7行目は繰り返し処理の始まりであり，GR3の値がORGLEN(値11)に等しいかを調べる[実行のようす(b)]．
  - 等しい場合：ラベルFINに分岐し，処理を終了する
  - もとの文字列の最後まで処理が済んだということになる



5-3 前判定の繰り返す処理(5)

■ プログラムの動き(3)

```
【10-13行目】  CPA     GR4,='0'
               JMI     LOOPE
               CPA     GR4,='9'
               JPL     LOOPE
```

- GR4の値が'0'(文字コード0030_{16})より小さい場合，あるいは'9'(文字コード0039_{16})より大きい場合に，ラベルLOOPEに分岐する．
- 数字は，文字コード0030_{16}～0039_{16}のあいだにあり(図5.7参照)，それ以外の範囲には英字や記号が割り当てられている．
- つまり，数字以外の英字や記号の場合には14,15行目を実行しないようにしている．

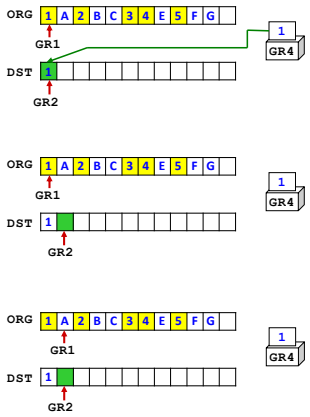
文字コード	文字	
$002E_{16}$	.	この範囲の文字を，10行目で除外する
$002F_{16}$	/	
0030_{16}	0	文字コードがこの範囲にあるときだけ，14,15行目を実行
0031_{16}	1	
0032_{16}	2	
0033_{16}	3	
0034_{16}	4	
0035_{16}	5	
0036_{16}	6	
0037_{16}	7	
0038_{16}	8	
0039_{16}	9	
:	:	
:	:	
:	:	
0041_{16}	A	この範囲の文字を，12行目で除外する
0042_{16}	B	

5-3 前判定の繰り返す処理(6)

■ プログラムの動き(4)

【14-18行目】

```
14:          ST      GR4,0,GR2
15:          LAD     GR2,1,GR2
16:LOOPE     LAD     GR1,1,GR1
17:          ADDA    GR3,=1
18:          JUMP    LOOP
```



【説明】

14:新しい文字列に文字を追加
15:新しい文字列を指すGR2をインクリメント
16:もとの文字列を指すGR1をインクリメント
17:処理した文字数のインクリメント
18:ラベルLOOPへ移動する

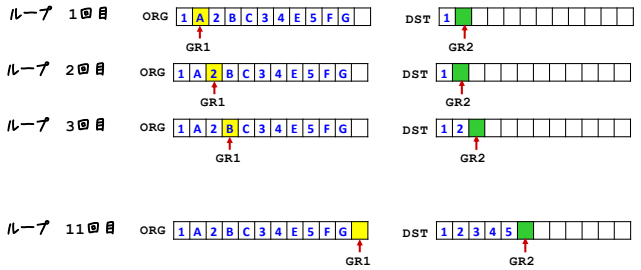
177

5-3 前判定の繰り返す処理(7)

■ プログラムの動き(5)

【7-18行目】

- これは18行目実行直前の主記憶のORG領域とDST領域の値を示したものである.
- 文字が数字のときだけ,ORGからDSTに文字がコピーされていく.



178

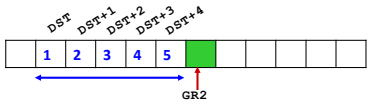
5-3 前判定の繰り返す処理(8)

■ プログラムの動き(6)

【21-24行目】

```
LAD     GR1,DST
SUBL    GR2,GR1
ST      GR2,DSTLEN      ; 新しい文字列の長さを求める
OUT     DST,DSTLEN      ; 新しい文字列の表示
```

- 新しい文字列がどのくらいの長さになるかは事前にはわからず,処理が終わって初めて,DST以降に格納された新しい文字列の長さを計算している.
- 21行目の実行の時点ではもう繰り返し処理が終わっており,GR2には新しい文字列の最後の文字のさらに1つ先のアドレスが格納されている.
- このアドレスからDSTのアドレスを引けば,ちょうど文字列の長さになるというわけである.
- 実行結果を見ると,'12345'と5文字分が表示されている.



新しい文字列の長さ = GR2の指すアドレス - DSTのアドレス

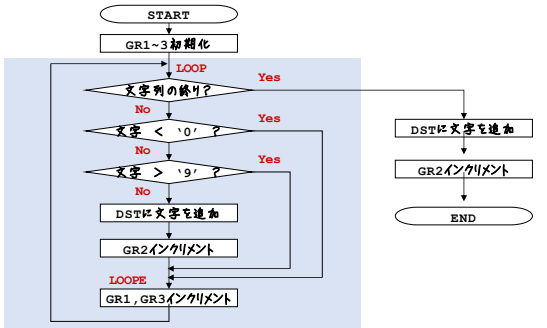
179

5-3 前判定の繰り返す処理(9)

■ 前判定による繰返し処理

【7行目】 LOOP CPA GR3,ORGLEN

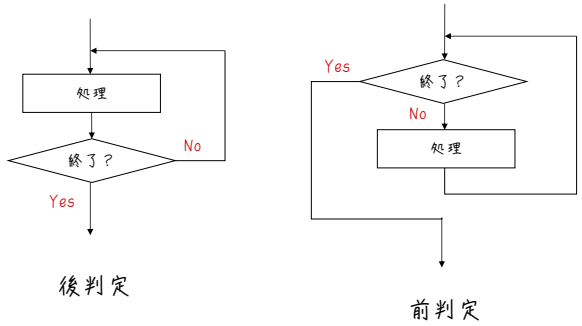
- ここでは,繰り返し処理の最初の部分で,繰り返し処理を続けるかどうかを判定し,そのあとで処理を行っている.
- このように,繰り返し処理の最初に判定を行うアルゴリズムを前判定という.



180

5-3 前判定の繰り返す処理(10)

■前判定と後判定



181

問題[5-3]

■8から始めて1ずつ値を減らしながら1まで繰り返し出力するプログラム(スライド22ページ目のPRG0501)とは逆に,GR1の値を1つずつ増やしていき,8まで表示したら止めるプログラムを,前判定のプログラムとして修正しなさい.

182

The END

183