プロセッサと機械語

2年次前期　（第10回）

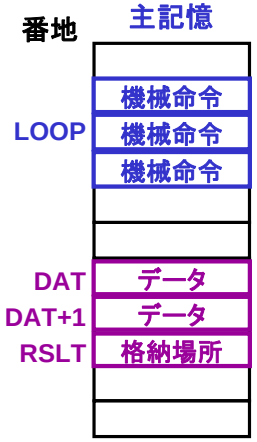
中島 克人

情報メディア学科
nakajima@im.dendai.ac.jp

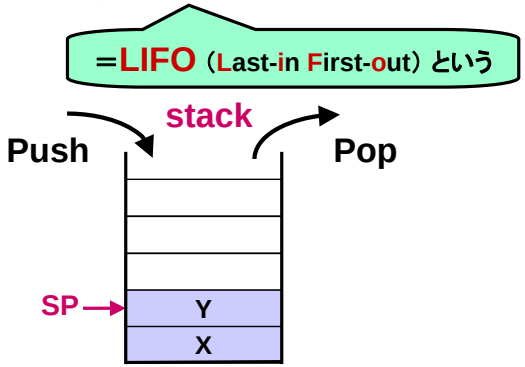
スタックとサブルーチン

データの格納場所

- メモリ（主記憶）
  - 番地の付いた1次元配列
  - ランダム（どこでも）アクセス



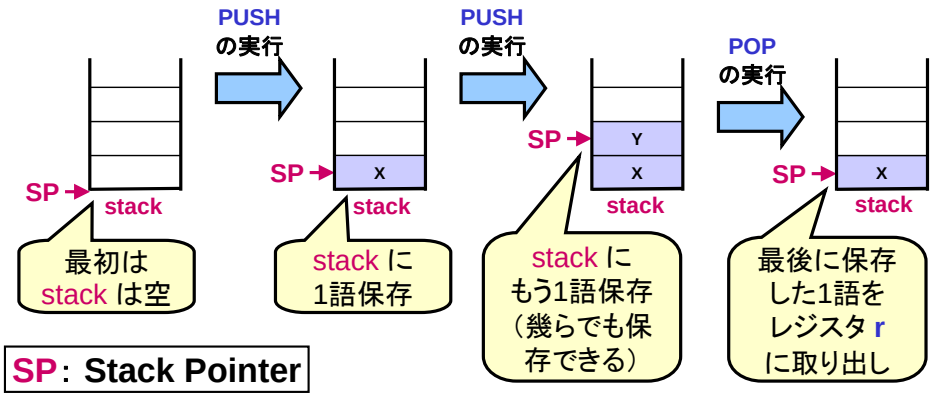
- スタック（stack）
  - 先頭（1番上）だけがアクセスできる（1次元領域）
  - 後入れ先出し



SP（Stack Pointer）：どこが1番上かを示す
… プログラマは余り意識しない

COMET Ⅱマシンの機械命令

- PUSH と POPによる stack操作
  - PUSH adr[,xr] ... 実効番地（adr[,xr]）（X,Y等とする）をstackの先頭に push
  - POP gr ... stackの先頭の内容を レジスタ gr に pop



SP：Stack Pointer

COMET Ⅱマシンの機械命令

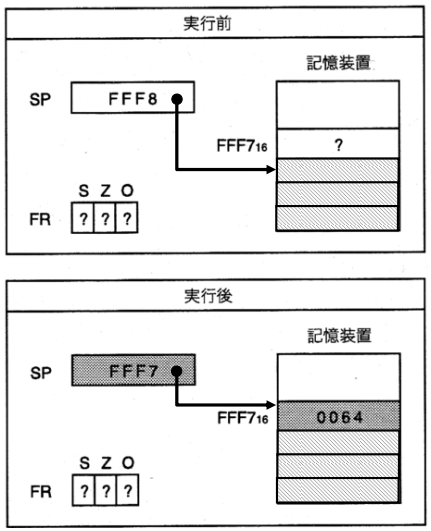
（教科書 p.26）

- プッシュ（push）命令

[書き方] PUSH　adr[,xr]

[機能]　<SP>－1 → SP;
　　e → <SP>

[実行例]　PUSH　100

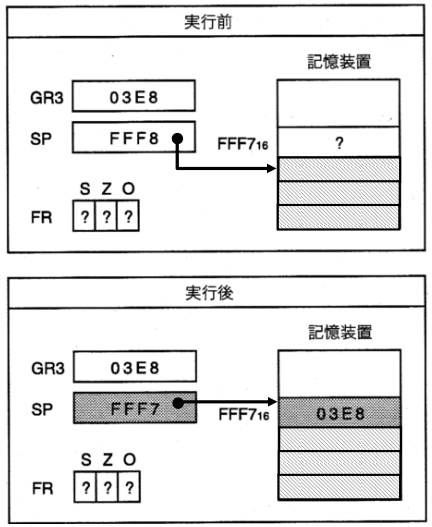


COMET Ⅱマシンの機械命令

（教科書 p.26）

- プッシュ（push）命令

[実行例]　PUSH　0,GR3



COMET Ⅱマシンの機械命令

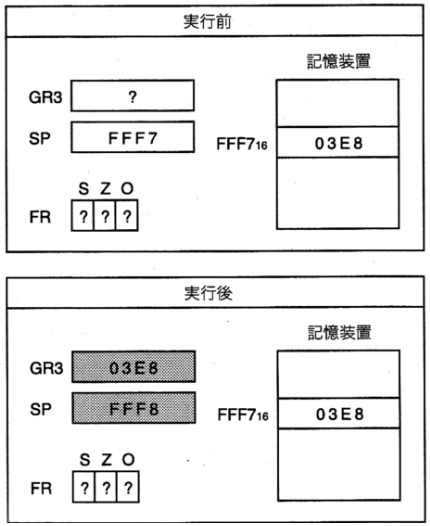
（教科書 p.27）

- ポップ（pop）命令

[書き方] POP　gr

[機能]　<<SP>> → GRn;
　　<SP>＋1 → SP

[実行例]　POP　GR3



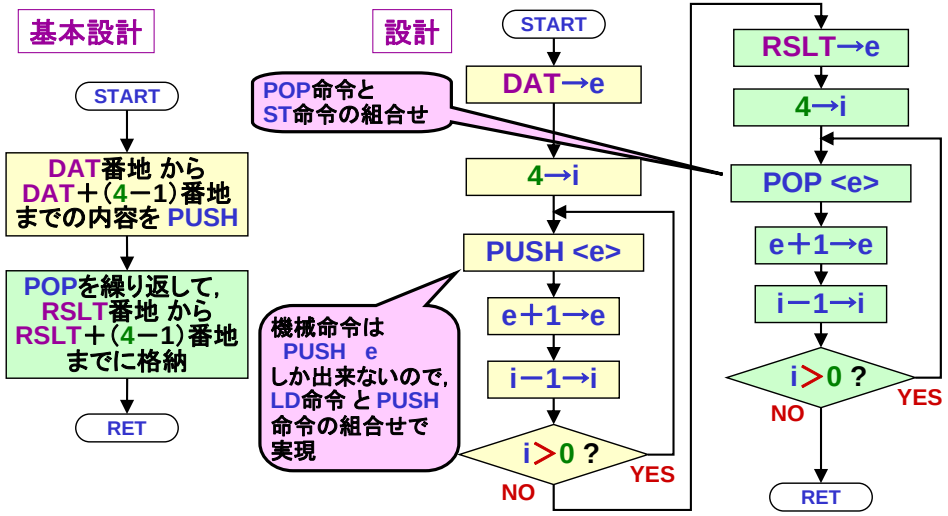
アセンブリ言語CASLⅡ

- 例題 RVGR ... GR1,GR2,GR3,GR4の内容をスタックを用いてそれぞれ，（順番が逆順になるように並べ替えて）GR4,GR3,GR2,GR1に格納するプログラムとプログラムを作成せよ

```
RVGR    START
        LAD   GR1,1     ; テストのための値の準備で何でも良い（以下同様）
        LAD   GR2,2
        LAD   GR3,3
        LAD   GR4,4
;ここから
        PUSH  0,GR1     ; 0＋<GR1> 即ち，GR1の内容がPUSHされる（以下同様）
        PUSH  0,GR2
        PUSH  0,GR3
        PUSH  0,GR4
        POP   GR1       ; 最後にPUSHした GR4の内容をGR1に（以下同様）
        POP   GR2
        POP   GR3
        POP   GR4
        RET
        END
```

アセンブリ言語CASL Ⅱ

- 例題 REV … DAT番地から連続する 4語の内容を，順番が逆順になるように並べ替えて，RSLT番地から連続する 4語に格納するプログラムのフローチャートとプログラムを作成せよ．



アセンブリ言語CASL Ⅱ

- 例題 REV … DAT番地から連続する 4語の内容を，順番が逆順になるように並べ替えて，RSLT番地から連続する 4語に格納するプログラムのフローチャートとプログラムを作成せよ

```
REV     START
        LAD   GR1,DAT   ; DAT→e (<GR1>)
        LAD   GR2,4     ; ループカウンタ i=4 に初期化
        LAD   GR3,1     ; ループカウンタの減算値1
LOOP1   LD    GR4,0,GR1 ; 次と2つの命令で PUSH <e> (GR0は使えない！)
        PUSH  0,GR4     ;
        LAD   GR1,1,GR1 ; アドレスの加算 e＋1→e (GR0は使えない！)
        SUBL  GR2,GR3   ; ループカウンタのデクリメント i－1→i
        JPL   LOOP1     ; i＞0 ならば LOOP1に戻り繰り返し
        LAD   GR1,RSLT  ; RSLT→e ; ここから後半だが，<GR3>はそのまま使用
        LAD   GR2,4     ; ループカウンタ i=4 に初期化
LOOP2   POP   GR4       ; POP
        ST    GR4,0,GR1 ; <e>に格納
        LAD   GR1,1,GR1 ; アドレスの加算 e＋1→e
        SUBL  GR2,GR3   ; ループカウンタのデクリメント i－1→i
        JPL   LOOP2     ; i＞0 ならば LOOP2に戻り繰り返し
        RET
DAT     DC  100,200,300,400
RSLT    DS  4
        END
```


COMET Ⅱマシンの機械命令

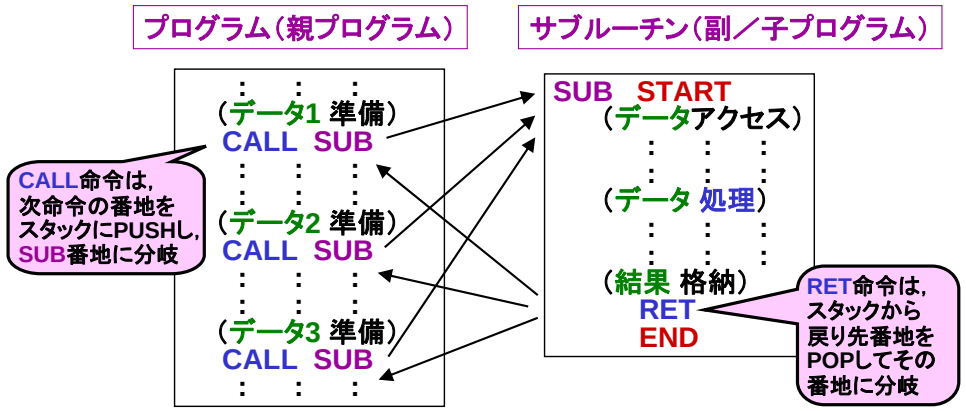
（教科書 p.27）

- コール（call subroutine）命令
  [書き方] CALL adr[,xr]
  [機能] <SP>－1 → SP;
  <PR> → <SP>;
  e → PR
- スーパーバイザコール（supervisor call）命令
  [書き方] SVC adr[,xr]
  [機能] 略
- リターン（return from subroutine）命令
  [書き方] RET
  [機能] <<SP>> → t;
  <SP>＋1 → SP;
  <t> → PR
- ノーオペレーション（no operation）命令
  [書き方] NOP
  [機能] （何もしない）

サブルーチンの構成（1）

（教科書 p.53）

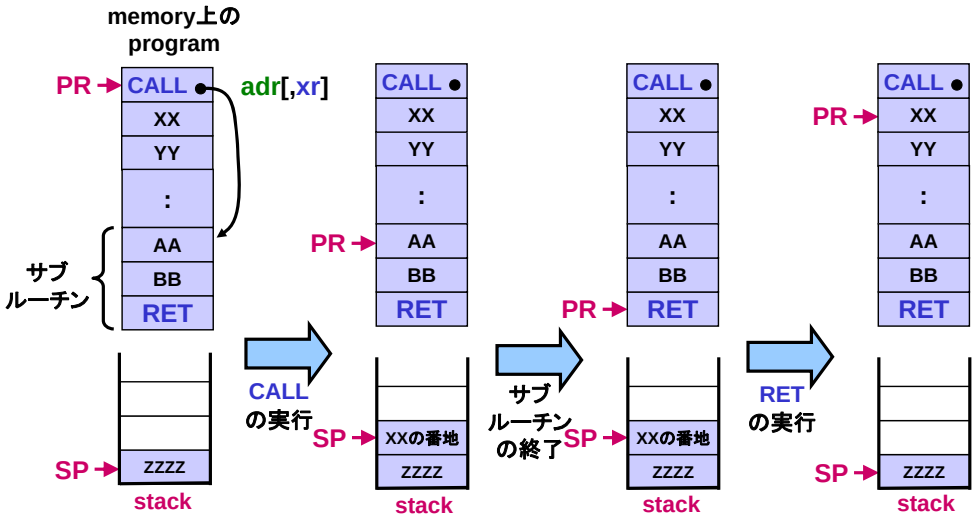
- データは異なるが同じ処理を何度も繰り返す場合，データを受け取って処理を行う部分をサブルーチンとする



- データをサブルーチンに渡し（アクセスできるようにし），結果データを呼び出し元のプログラムに返す（アクセスできるようにする）
  ⇒ これらのデータを引数（parameter／argument）と言う

COMETⅡマシンの機械命令

- CALL adr[,xr] と RET によるサブルーチンの実現

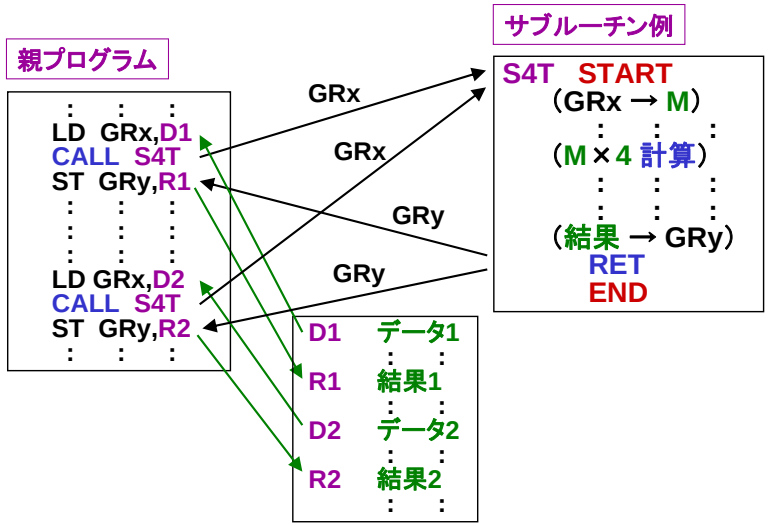


stack を戻り番地の保存に使用
（注：実際には汎用レジスタの内容等の退避も必要）

サブルーチンの構成（2）

（教科書 p.54）

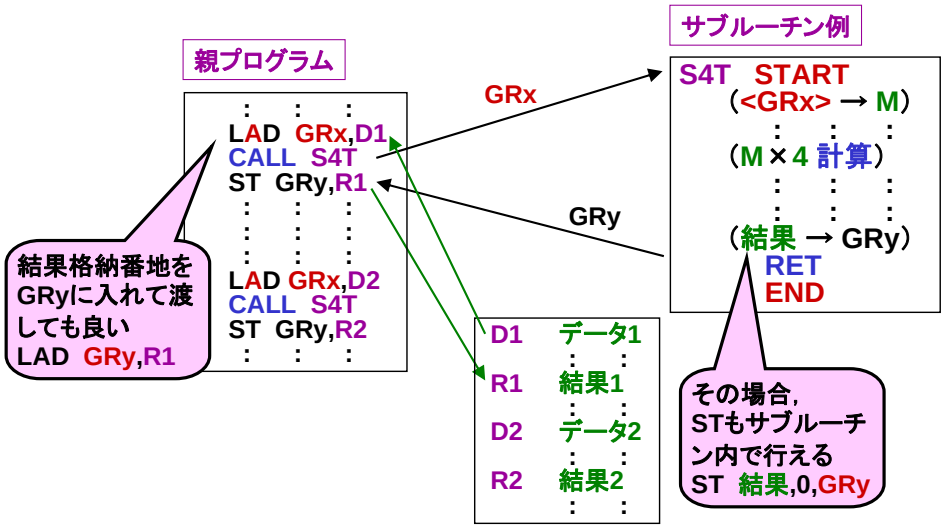
- 引数の受渡しは汎用レジスタを用いる
  - サブルーチンは呼び出し元のプログラムで使用しているラベルは知らないので汎用レジスタに（入力）データを入れる



サブルーチンの構成（3）

（教科書 p.54）

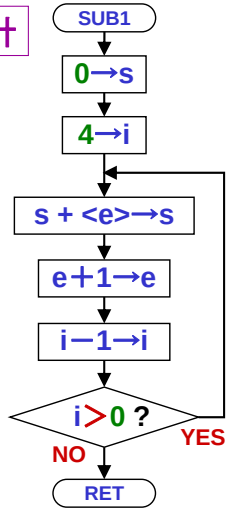
- 引数の受渡しは汎用レジスタを用いる
  - 汎用レジスタに親プログラムで使用している番地を入れても良い



アセンブリ言語CASLⅡ

- 演習 SUB1 ... <GR1>番地（e＝GR1に格納されている番地）から連続する4語の内容を加算し，GR0に格納する副プログラム（サブルーチン）を作成せよ．

設計



; 加算結果 s＝0 に初期化
; ループカウンタ i＝4 に初期化
; メモリ（e番地）から1語ロードし，s に加算
; アドレス（e番地）の加算 e+1→e
; ループカウンタのデクリメント i−1→i
; i>0 ならば 繰り返し

アセンブリ言語CASL Ⅱ

- 演習 SUB1 ... <GR1>番地から連続する4語の内容を加算し，GR0に格納する副プログラム（サブルーチン）を作成せよ．

実装

```
SUB1   START
       XOR    GR0,GR0    ; 加算結果を収納するGR0を 0 に初期化
              ┌┐         ; ループカウンタを <N>＝4 に初期化
              └┘         ; ループカウンタの減算値1
LOOP          ┌┐         ; メモリ（e＝<GR1>）から1語ロードし，加算
              └┘
       LAD    GR1,1,GR1  ; アドレスの加算 <GR1>＋1→GR1
       SUBL   GR2,GR3    ; ループカウンタのデクリメント <GR2>－1→GR2
       JPL    LOOP       ; <GR1>＞0 ならば LOOPに戻り繰り返し
       RET               ; 加算結果は GR0 にあるので，そのままリターン
N      DC     4          ; 指定ループ回数
       END
```

◆ このサブルーチンは入力が GR1（番地情報），出力が GR0（加算結果）であるが，主プログラムでは，これを呼び出した後，レジスタ等にどういう変化が起こるか？

アセンブリ言語CASL Ⅱ

- 演習 SUB2 ... SUB1をGRの値を変更しないように改良せよ．

```
SUB2   START
       PUSH   0,GR1      ; GR1 の内容をスタックに退避
              ┌┐         ; GR2 の内容をスタックに退避
              └┘         ; GR3 の内容をスタックに退避
              ┌┐         ; 加算結果を収納するGR0を 0 に初期化
              └┘         ; ループカウンタを <N>＝4 に初期化
              ┌┐         ; ループカウンタの減算値1
LOOP          └┘         ; メモリ（e＝<GR1>）から1語ロードし，加算
       LAD    GR1,1,GR1  ; アドレスの加算 <GR1>＋1→GR1
       SUBL   GR2,GR3    ; ループカウンタのデクリメント <GR2>－1→GR2
       JPL    LOOP       ; <GR1>＞0 ならば LOOPに戻り繰り返し
              ┌┐         ; GR3 の内容をスタックから復元
              └┘         ; GR2 の内容をスタックから復元
                         ; GR1 の内容をスタックから復元
       RET               ; 加算結果は GR0 にあるので，そのままリターン
N      DC     4          ; 指定ループ回数
       END
```

◆ GR0, GR4～GR7, FR はスタックに退避していないが，何故か？

アセンブリ言語CASL Ⅱ

- 演習 CAL ... 左下の主プログラムにおいて，SUB1の呼び出しによってGR1,GR2,GR3の内容が壊れても良いように改良せよ．

改良前

```
CAL  START
       :
       :
       :
     LAD  GR1,DAT ; SUB1の入力
                  ;（GR1＝e）の準備
     CALL  SUB1   ; ループカウンタを
                  ; <N>＝4 に初期化
       :
       :
       :
       :
DAT  DS  4
```

改良後

```
CAL  START
       :
       ┌┐
       └┘
     LAD  GR1,DAT ; SUB1の入力
                  ;（GR1＝e）の準備
     CALL  SUB1   ; ループカウンタを
                  ; <N>＝4 に初期化
       ┌┐
       └┘
     （LAD  GR1,DAT）; 必要なら
       :
DAT  DS  4
```

◆ 改良後の主プログラムから SUB1 に代えて SUB2 を呼び出したらどうなるか？SUB2 からリターンした直後のスタックの様子を図示せよ．

CASL Ⅱのマクロ命令

（教科書 p.31）

- レジスタプッシュ（rpush）命令
  [書き方] RPUSH
  [説明] GR の内容を，GR1, ... ,GR7 の順序でスタックにプッシュする．
- レジスタポップ（rpop）命令
  [書き方] RPOP
  [説明] スタックからポップし，GR7, ... ,GR1 の順序で GR に格納する．

- 演習 SUB2 の解答内の PUSH，POP命令に代えて，RPUSH，RPOP命令を用いる場合の長所・短所を考察せよ．

実効番地と即値について（確認）

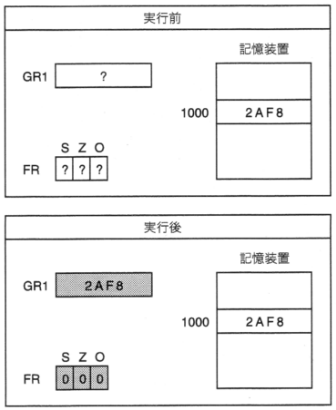
- 命令によって 実効番地 adr[,xr] の使われ方が異なる事に注意

実効番地e＝アドレス+ <xr>

load命令　LD gr, adr [,xr]

[機能]　<e> → GRn; 0→OF;

[実行例]　LD　GR1,1000

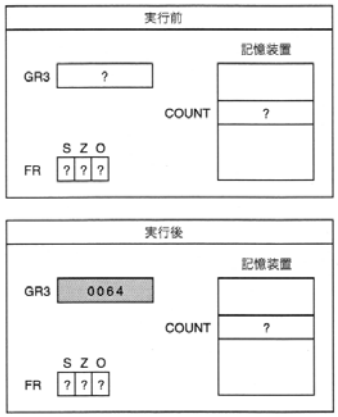


実効番地e＝即値+ <xr>

load address命令 LAD gr, adr [,xr]

[機能]　e → GRn

[実行例]　LAD　GR3,COUNT



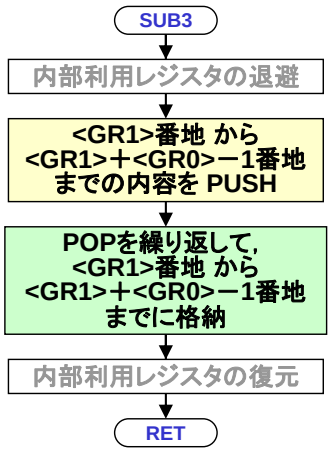
即値を取る他の命令

SLA gr, val [,xr]
SRA gr, val [,xr]
SLL gr, val [,xr]
SRL gr, val [,xr]
PUSH val [,xr]

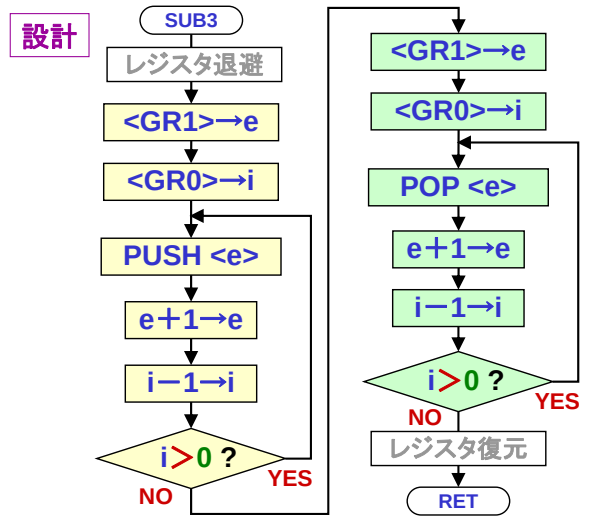
アセンブリ言語CASL Ⅱ

- 演習 SUB3 ... <GR1>番地から連続する<GR0>語の内容を，順番が逆順になるように並べ替える副プログラムのフローチャートとプログラムを作成せよ．（GR0,GR1以外のGRは非破壊で）

基本設計



設計



アセンブリ言語CASL Ⅱ

- 演習 SUB3 ... <GR1>番地から連続する<GR0>語の内容を，順番が逆順になるように並べ替える副プログラムのフローチャートとプログラムを作成せよ．（GR0,GR1以外のGRは非破壊で）

```
SUB3    START
        RPUSH
        LD      GR2,GR1    ; <GR1>→e
        LD      GR3,GR0    ; ループカウンタ i=<GR0> に初期化
        LAD     GR4,1      ; ループカウンタの減算値1
LOOP1   [          ]       ; 次と2つの命令で PUSH <e>
        PUSH    0,GR5      ;
        [          ]       ; アドレスの加算 e＋1→e
        SUBL    GR3,GR4    ; ループカウンタのデクリメント i－1→i
        JPL     LOOP1      ; i＞0 ならば LOOP1に戻り繰り返し
    ; ここから後半だが，e=<GR1>，i=<GR0>，<GR4>=1 はそのまま使用
LOOP2   POP     GR5        ; POP
        [          ]       ; <e>に格納
        LAD     GR1,1,GR1  ; アドレスの加算 e＋1→e
        SUBL    GR0,GR4    ; ループカウンタのデクリメント i－1→i
        JPL     LOOP2      ; i＞0 ならば LOOP2に戻り繰り返し
        RPOP
        RET
        END
```


アセンブリ言語CASL Ⅱ

（教科書 p.53,54）

- サブルーチンの引数（復習）
  - 演習SUB1～3において，GR0 や GR1 は親プログラムとサブルーチンとの間で（直接または間接的に）データのやりとりを行うために用いられている．サブルーチンとのやり取りデータを引数（パラメータやアーギュメントとも）という．入力引数と出力引数がある．
  - サブルーチンの共用のため，通常固定したメモリ領域を引数とはしない
  - GR自身に入出力引数値を入れる場合と，GRに入力引数値の入った（出力引数値の格納される）メモリ領域への番地を入れる場合がある