6. Subrotinas e Macros

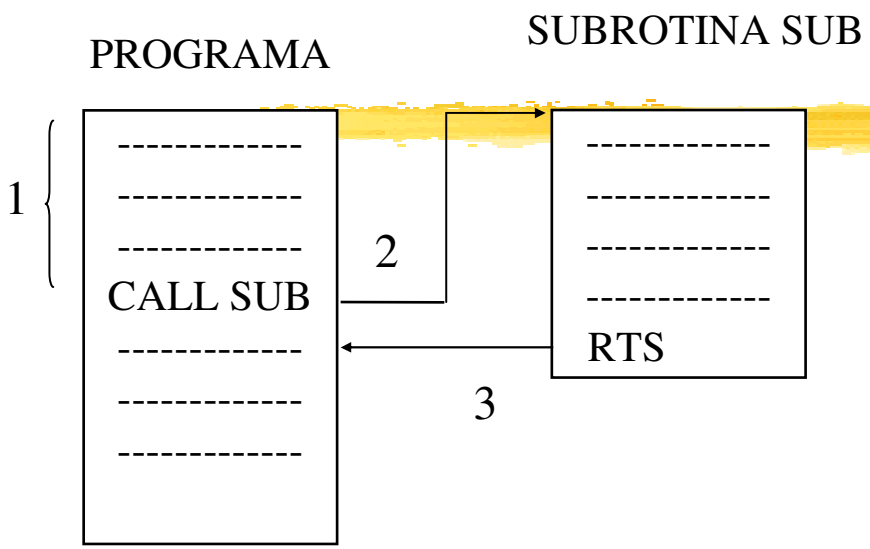
(capítulo 5 do livro texto)

Subrotinas e **macros** são formas alternativas para permitir que uma mesma sequência de instruções seja executada, em diferentes pontos de um programa, sobre diferentes conjuntos de dados.

Nas **subrotinas**, o controle de execução será desviado para a posição de memória onde a subrotina está armazenada e, após a execução da subrotina, o controle deverá ser retornado ao ponto imediatamente posterior ao da ocorrência do desvio.

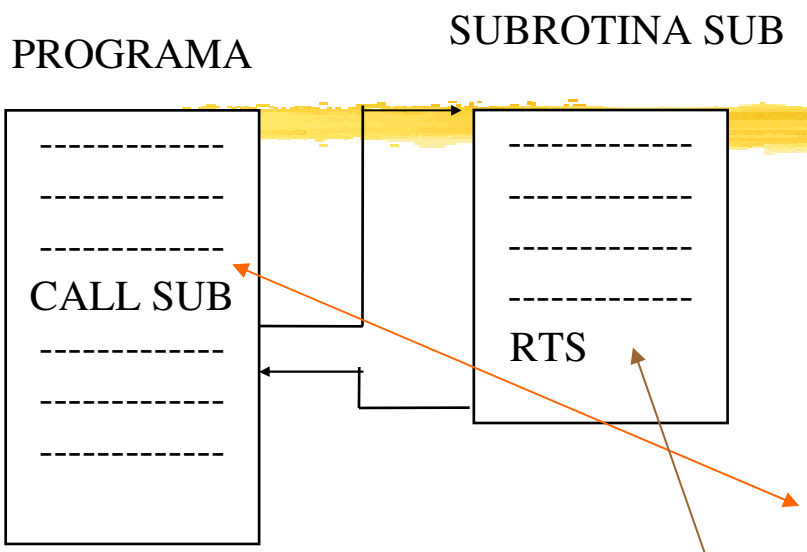
Macros, diferentemente das subrotinas, são blocos de programas que são incorporados ao código por um pré-processamento do tradutor. Desta forma, não existirá o "salto" durante a execução, já que o código fonte foi “expandido” no próprio programa original.

6.1 Subrotinas



1. programa sendo executado
2. desvio (CALL) para rotina
   - guardar endereço de retorno
   - passagem de parâmetros
3. retorno (RTS) para o programa
   - retorno ao endereço correto
   - passagem de parâmetros

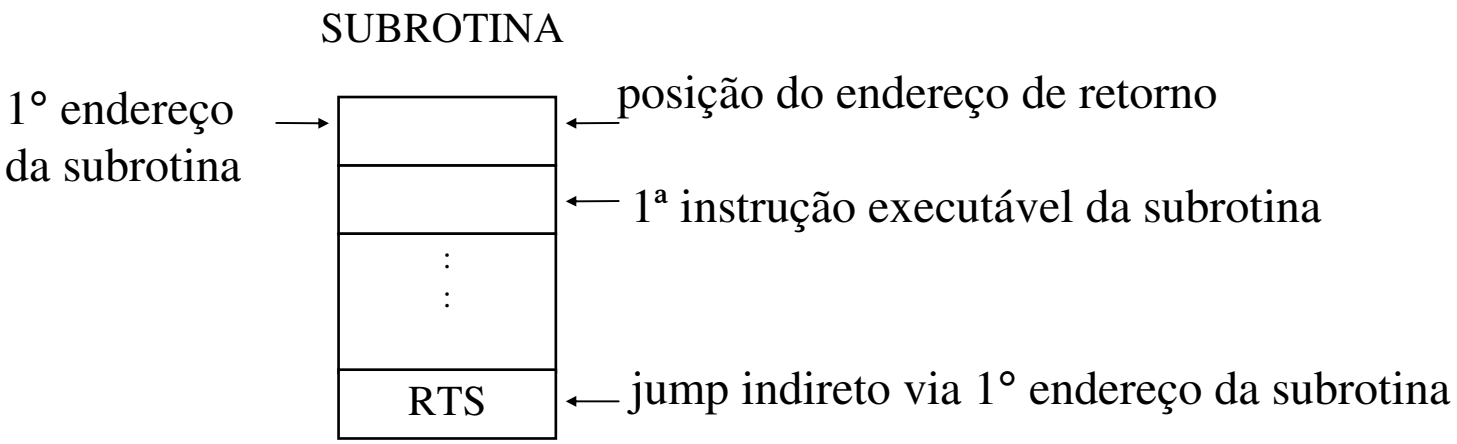
Subrotinas



1. programa sendo executado
2. desvio (CALL) para rotina
   - guardar endereço de retorno
   - passagem de parâmetros
3. retorno (RTS) para o programa
   - retorno ao endereço correto
   - passagem de parâmetros

Subrotinas – implementação
Opção 1

Endereço de retorno junto à subrotina



Subrotinas – implementação
Opção 1 – cont.

Exemplo em Assembly:

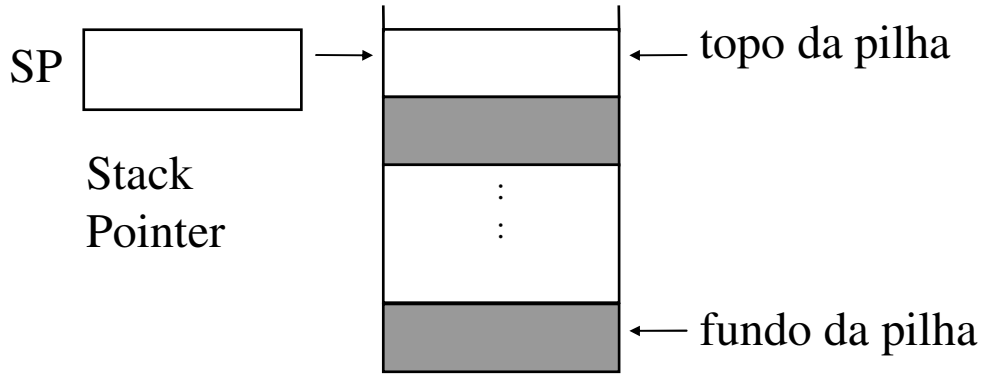
```
SUB:    DS        1          ;reserva 1 palavra para endereço
        ADD       .....
        MOVE      .....
        ............
        JUMP (SUB)           ;desvio indireto (modo absoluto indireto, por ex.)

Programa: ......
        CALL   SUB           ;SUB ← (PC) e PC ← endereço-de-SUB + 1
        ........
```

endereço seguinte à instrução CALL

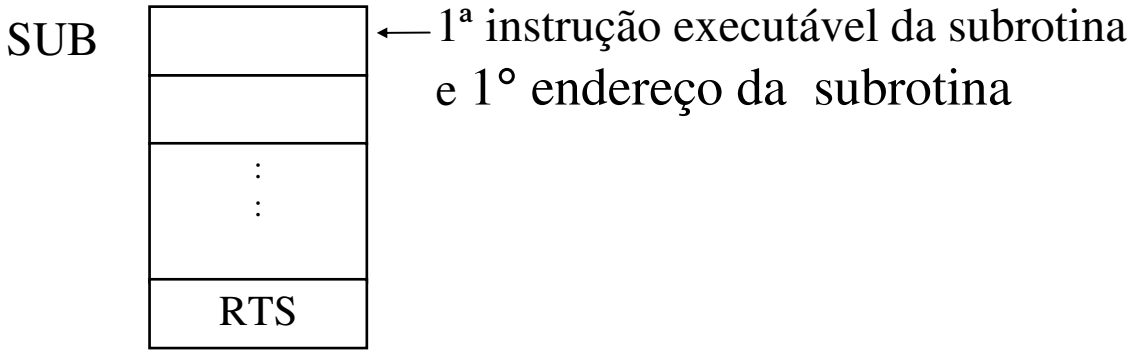
Subrotinas – implementação Opção 2

Endereço de retorno na pilha

LIFO - Last-In-First-Out



SUBROTINA



Subrotinas – implementação
Opção 2 – cont.

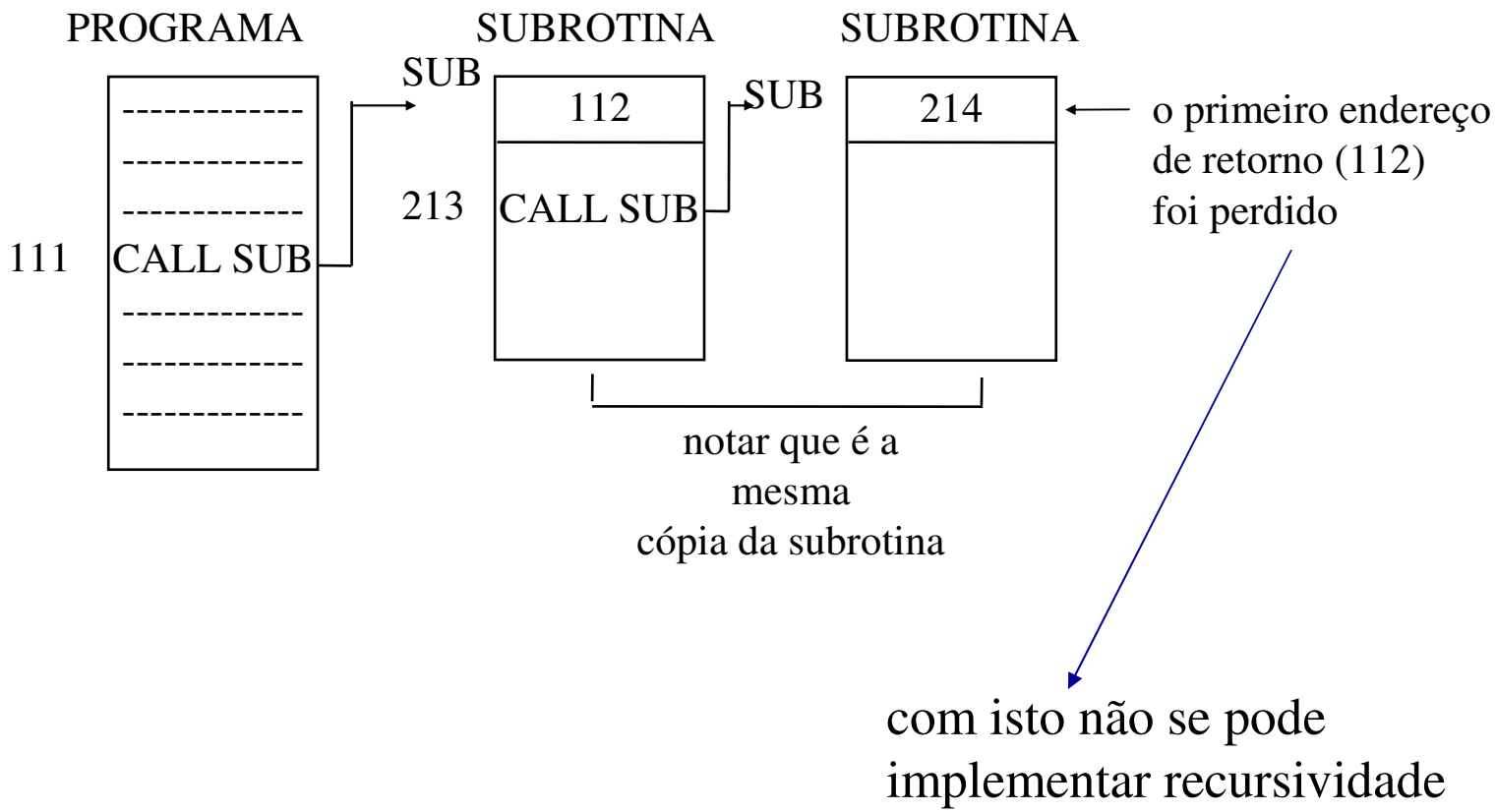
Exemplo em Assembly:

endereço seguinte à instrução CALL

```
Programa: .........
        CALL  SUB;    (SP) ← (PC)                  } PUSH  PC
                  ;    SP ← (SP) + 1
                  ;    PC ← "endereço de SUB"
          ........
SUB:    ......
        RTS       ;    SP ← (SP) -1                 } POP  PC
                  ;    PC ← ((SP))
```

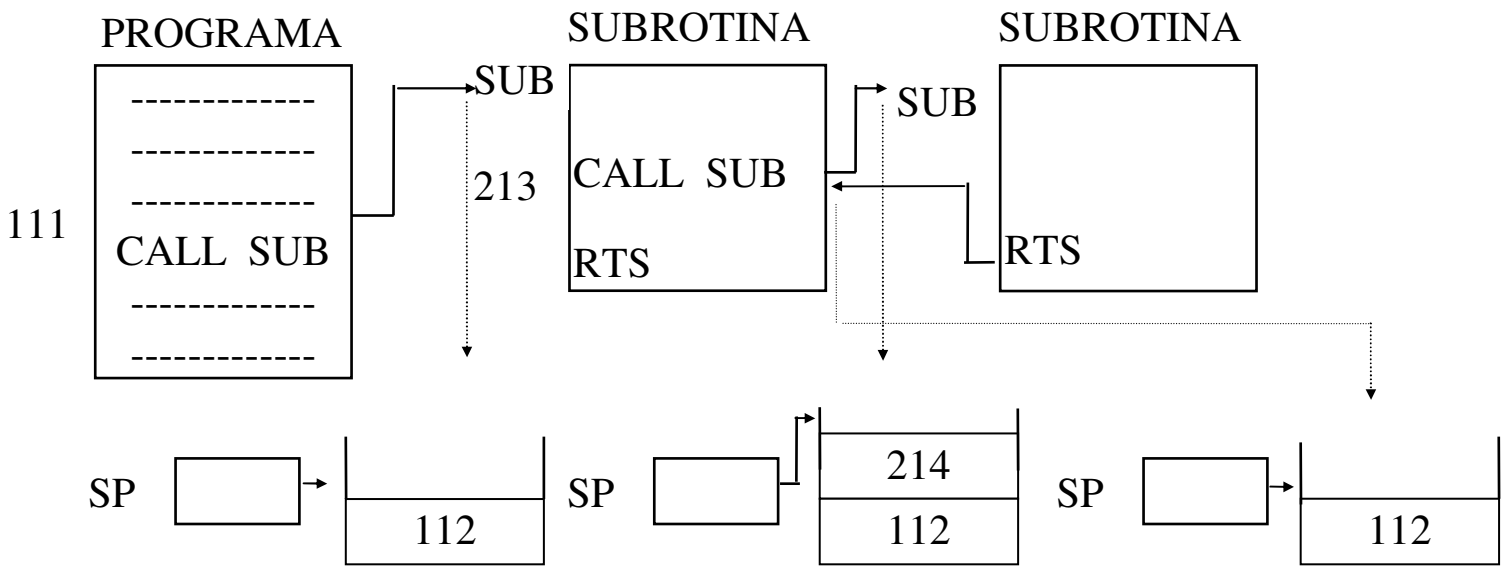
Recursividade em/com Subrotinas

- **opção com endereço de retorno junto à subrotina**



Recursividade em/com Subrotinas

- **opção com endereço de retorno na pilha**



6.2 Passagem de Parâmetros

Parâmetro Formal e Real

Parâmetro Formal: variável fictícia ("dummy variable") usada na definição da subrotina, sendo substituída por uma variável ou um valor, cada vez que a subrotina for chamada.

Parâmetro Real: variável ou valor que é usado como argumento na chamada da subrotina.

```
SUBROUTINE  SUB(X,Y)    ; declaração de SUB
```

X e Y são os **parâmetros formais**.

```
            :
            :
CALL  SUB(A,B)   ; chamada de SUB
            :
CALL  SUB(1,C)
            :
            :
```

A, B, C, e 1 são os **parâmetros reais**

Passagem de Parâmetros
. por valor

CONCEITO : passagem de parâmetros por valor - "call by value" (Pascal, Algol)

```
Function  F1 (x,y: real; max: integer): real
    ..... etc ......
```

parâmetros formais

```
Programa:  ..............
            a:= F1(w+z, 1.5, i)
            ..... etc ......
```

parâmetros reais

Na passagem de **parâmetros por valor** os valores dos **parâmetros reais** não são alterados.

Passagem de Parâmetros
. por endereço

CONCEITO : passagem de parâmetros por endereço (Fortran, Pascal)

```
PROCEDURE P1( VAR x, y : integer; VAR max : real )

        …… etc …….
```

parâmetros formais

```
Programa:  ..............

        call P1 (a, b, c)

        ..... etc ......
```

parâmetros reais
(devem ser definidos com o mesmo tipo)

Na passagem de **parâmetros por endereço** os valores dos **parâmetros reais** podem ser alterados, pois o **endereço** dos mesmos é compartilhado.

Passagem de Parâmetros
Exemplos em Assembly

Sintaxe de códigos mnemônicos

Para instruções de uma máquina (fictícia) de 2 endereços:

Formato geral:

rótulo: operação fonte, destino ; destino ← (fonte) operação (destino)

Exemplo:

ROTULO: ADD end1, end2 ;end2 ← (end1) + (end2),

Passagem de Parâmetros
Exemplos em Assembly – cont.

Casos a estudar:

Passagens de parâmetros por:

1. registrador da CPU
2. área associada à subrotina
3. área associada ao programa
4. área comum
5. pilha

Passagem de parâmetros em Assembly
1a. por registradores da CPU

Exemplo 1: Convenção para a passagem de parâmetros
- registrador R1 para o valor de entrada,
- registrador R2 para o valor do resultado.

Subrotina com passagem de parâmetros **por valor** em registradores da CPU

PROGRAMA

```
        MOVE  #50, SP
        --------------------
        MOVE DADO, R1
        CALL SUB
        MOVE R2, REST
        ---------------------
        STOP
DADO:   DW     5
REST:   DS     1
        --------------------
```

SUBROTINA

```
-------------
ADD  #3,R1
-------------
MOVE  R1,R2
-------------
RTS
```

Neste exemplo foi passado o *valor* calculado (valor de DADO permanece inalterado ao final).

Passagem de parâmetros em Assembly
1b. por registradores da CPU – cont.

Exemplo 2: Convenção para a passagem de parâmetros
- registrador R1 para o endereço do dado de entrada,
- registrador R2 para o endereço do resultado.

Subrotina com passagem de parâmetros **por endereço** em registradores da CPU

PROGRAMA

```
        MOVE  #50, SP
        --------------------
        MOVE #DADO, R1
        MOVE #REST,R2
        CALL SUB
        ---------------------
        STOP
DADO:   DW     5
REST:   DS     1
        --------------------
```

SUBROTINA

```
-------------
ADD  #3,(R1)
-------------
MOVE  (R1),(R2)
-------------
RTS
```

Neste exemplo foi passado o *endereço* de DADO. Então ao final o valor de DADO poderá estar alterado.

Passagem de parâmetros em Assembly
2. por Área associada à subrotina

Exemplo 3: Convenção para a passagem de parâmetros

- DIN1: posição para dados de entrada-1 (valor)
- DIN2: posição para dados de entrada-2 (valor)
- RES: posição onde SUB coloca o valor do resultado

PROGRAMA

```
        MOVE  #50, SP
        --------------------
        MOVE DAD1, DIN1
        MOVE DAD2, DIN2
        CALL SUB
        MOVE RES, RESU
        --------------------
        STOP
DAD1:   DW     5
DAD2:   DW    18
RESU:   DS     1
```

SUBROTINA

```
DIN1:   DS    1
DIN2:   DS    1
RES :   DS    1
SUB:    ----------
        ----------
        ADD  DIN1, DIN2
        ----------
        MOVE  R1, RES
        RTS
```

Neste exemplo foram passados os *valores* dos parâmetros para a subrotina.

Passagem de parâmetros em Assembly
3. por Área associada ao programa

PROGRAMA

```
        MOVE  #50, SP
        --------------------
        MOVE VAR1, DAD1
        MOVE VAR2, DAD2
        CALL SUB
DAD1:   DS    1
DAD2:   DS    1
RES1:   DS    1
        MOVE RES1, VAR5
        --------------------
        MOVE VAR3, DAD3
        MOVE VAR4, DAD4
        CALL SUB
DAD3:   DS    1
DAD4:   DS    1
RES2:   DS    1
        MOVE RES2, VAR6
        --------------------
```

```
VAR1:   DW    20
VAR2:   DW    67
VAR3:   DW    3
VAR4:   DW    13
VAR5:   DS    1
VAR6:   DS    1
```

SUBROTINA

```
SUB:    POP  R1
        MOVE (R1)+, R5
        MOVE (R1)+, R6
        -------------------
        -------------------
        ADD  R5, R6
        -------------------
        MOVE R6, (R1)+
        PUSH R1
        RTS
```

R1 ← topo da pilha

1° parâmetro em R5

2° parâmetro em R6

armazena o resultado no 3° parâmetro

o conteúdo de R1- endereço da próxima instrução - é colocado na pilha para o retorno

Passagem de parâmetros em Assembly
4. por Área comum de dados

Passagem de parâmetros em Assembly
4. por Área comum de dados – cont.

PROGRAMA

```
        MOVE  #50, SP
        ---------------------
        MOVE #AREA, R1
        MOVE DAD1, (R1)+
        MOVE DAD2, (R1)
        CALL SUB
        MOVE #AREA, R1
        INCR R1
        INCR R1
        MOVE (R1), RESU
        --------------------
        STOP
DAD1:   DW   5
DAD2:   DW   18
RESU:   DS   1
```

ÁREA COMUM DE DADOS

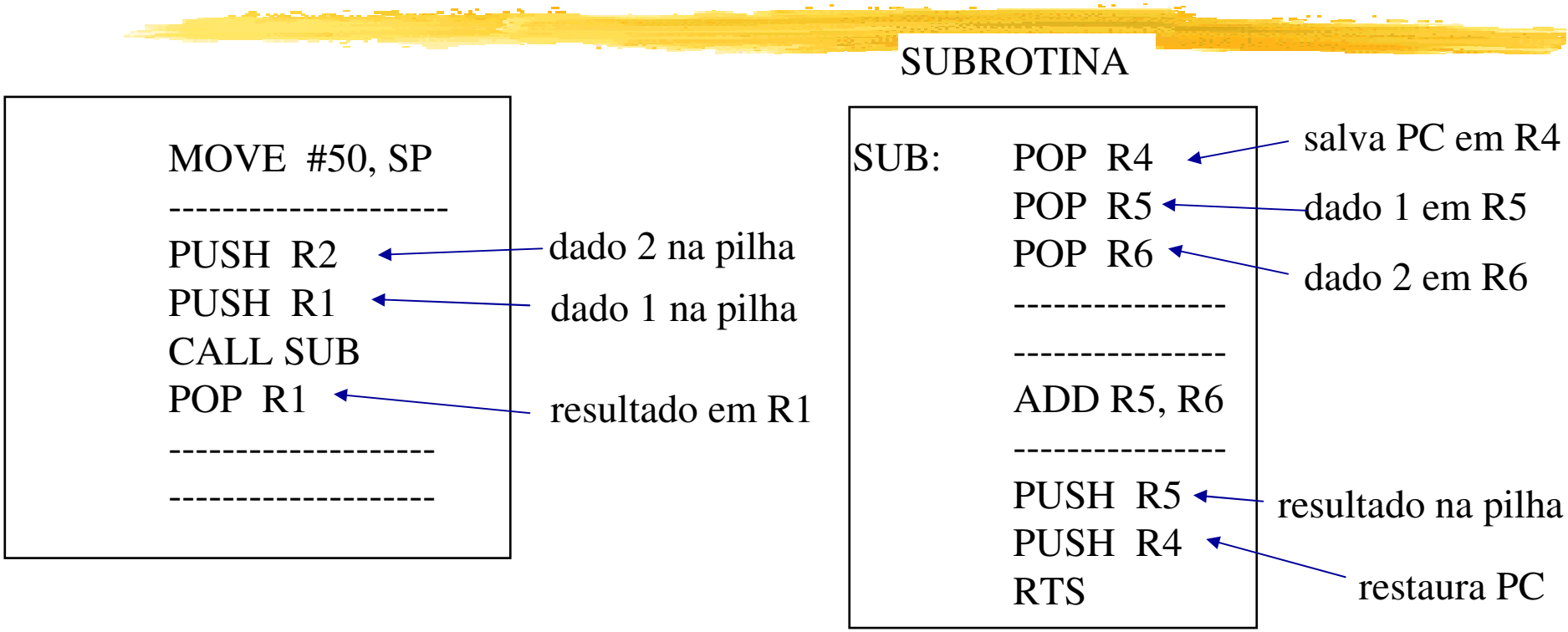
```
AREA:  DS  3
```

SUBROTINA

```
SUB:    ---------------------
        ---------------------
        MOVE #AREA, R1
        MOVE (R1)+, R2
        MOVE (R1)+, R3
        ---------------------
        ADD R2, R3
        ---------------------
        ---------------------
        MOVE R3, (R1)
        RTS
```

O endereço da área comum é conhecido tanto pela subrotina como pelo programa principal.

Passagem de parâmetros em Assembly
5. por Pilha

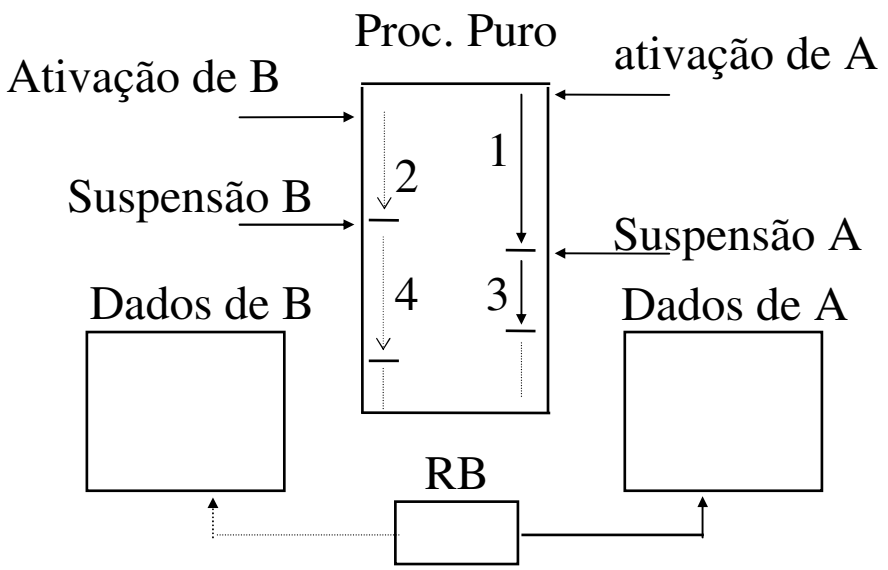


- o programa empilha os dados conforme convenção comum programa – subrotina
- variáveis locais da subrotina podem usar a pilha >> manipulação cuidadosa !

6.3 Código reentrante

O que queremos ? Subrotinas que possam ser interrompidas durante a sua execução – para execução em outro contexto – e então reiniciadas sem problemas.

Notemos que: subrotinas comuns só são **serialmente reusáveis**.



Conteúdo de RB
1. End. Dado A
2. End. Dado B
3. End. Dado A
4. End. Dado B

Notemos que na figura acima a sequência no tempo é: (1) → (2) → (3) → (4)

Código reentrante – cont.

Uma subrotina que não utiliza armazenagem temporária em seu próprio corpo e que não modifica a si própria, é um ***Procedimento Puro*** ou ***Código Reentrante***.

Os dados manipulados por um procedimento puro são referenciados de forma não direta, através de registradores (por exemplo, registrador base).

Exemplo de aplicação são os compiladores em um sistema multiusuário.

6.4 Exercícios sobre subrotinas

1. Resolva os exercícios (12), (13) e (15) do livro texto. Observe na errata correções no texto do exercício (12).