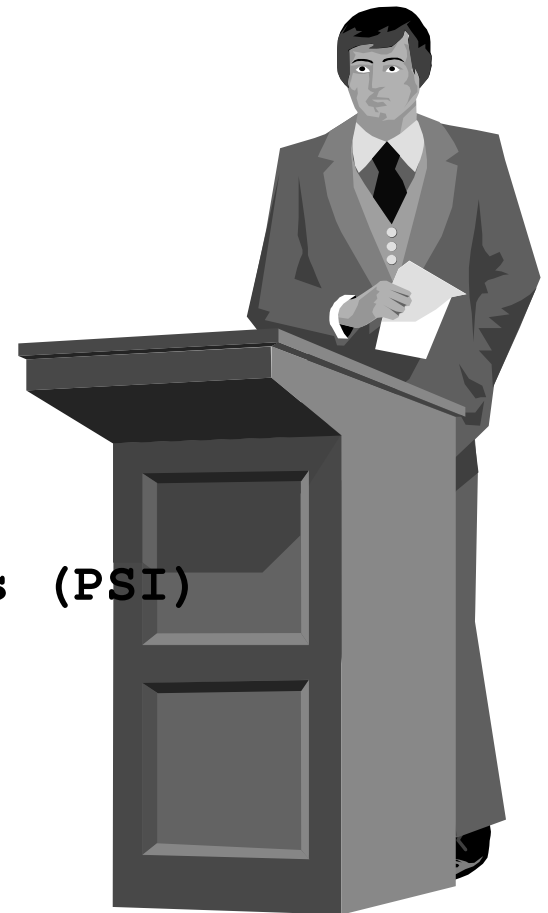

Pilha de execução

Volnys Borges Bernal

volnys@lsi.usp.br

Departamento de Sistemas Eletrônicos (PSI)

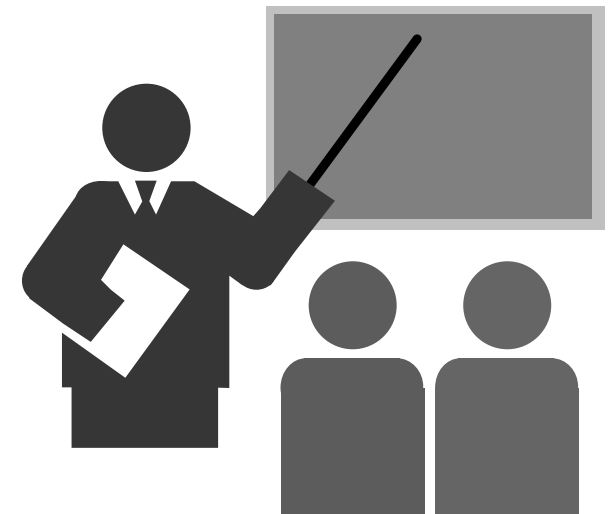
Escola Politécnica da USP



Agenda

- ❑ Os desafios da execução de subrotinas
- ❑ Pilha de execução
 - ❖ Controle do endereço de retorno da função
 - ❖ Passagem dos valores dos argumentos da função
 - ❖ Alocação de variáveis locais
- ❑ Quadro da pilha de execução

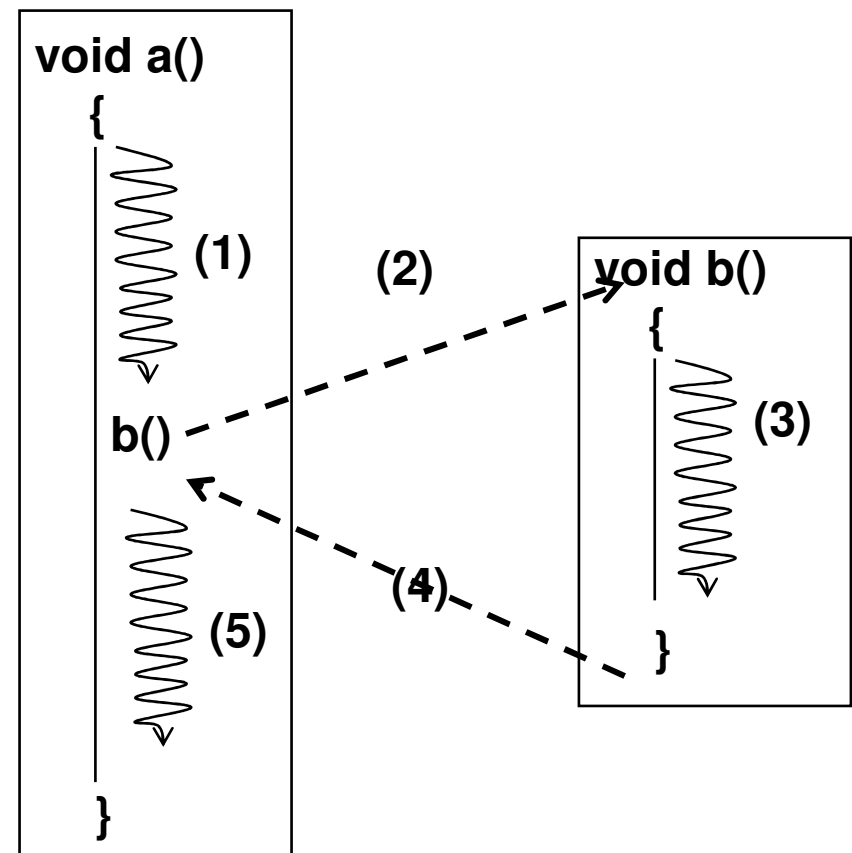
Os desafios da execução de subrotinas



Os desafios da execução de subrotinas

1 - Retorno de subrotina (função)

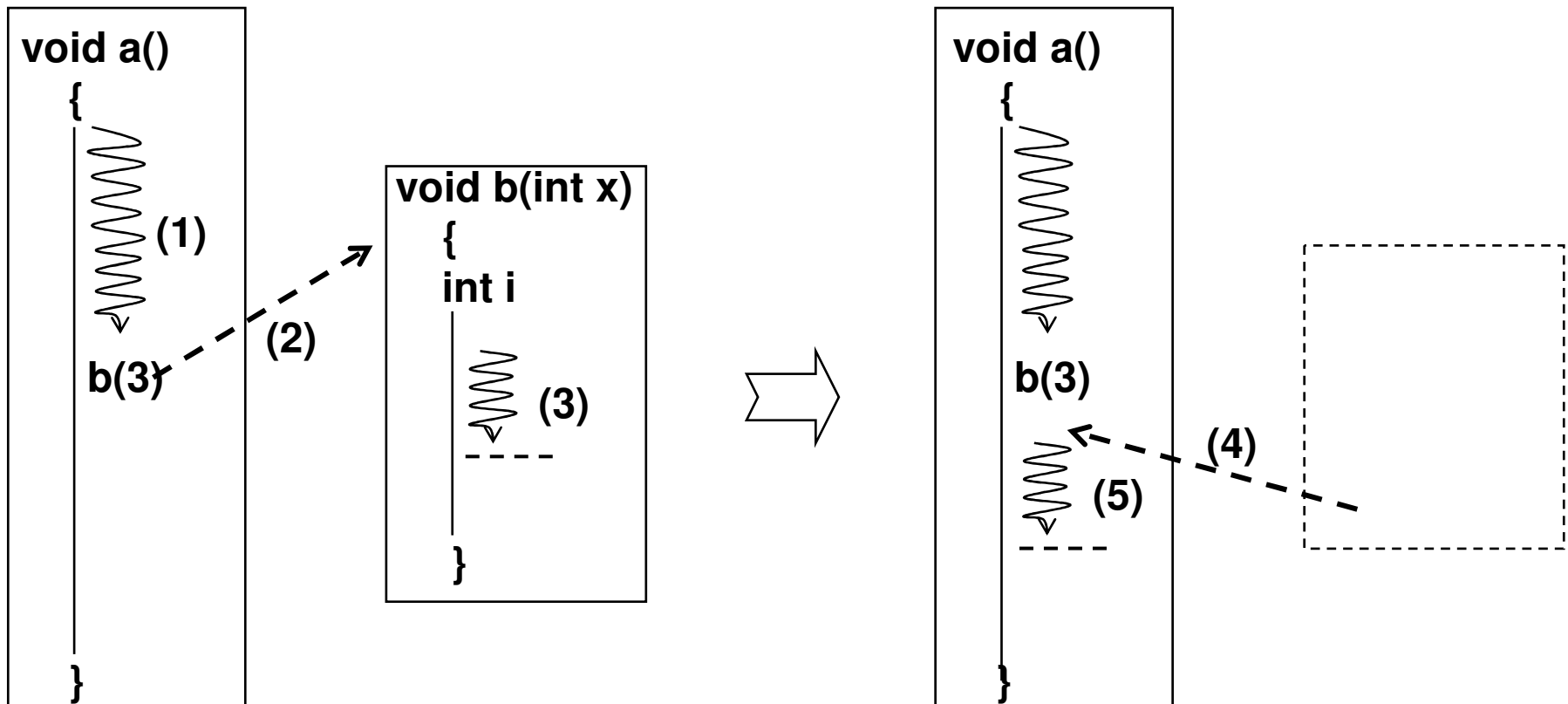
- ❖ Quando uma subrotina (função) é ativada, o controle da execução deve ser transferido para a instrução inicial da subrotina.
- ❖ Ao termino da subrotina, a próxima instrução a ser executada deve ser a instrução posterior à instrução que ativou a subrotina



Os desafios da execução de subrotinas

2 – Variáveis locais

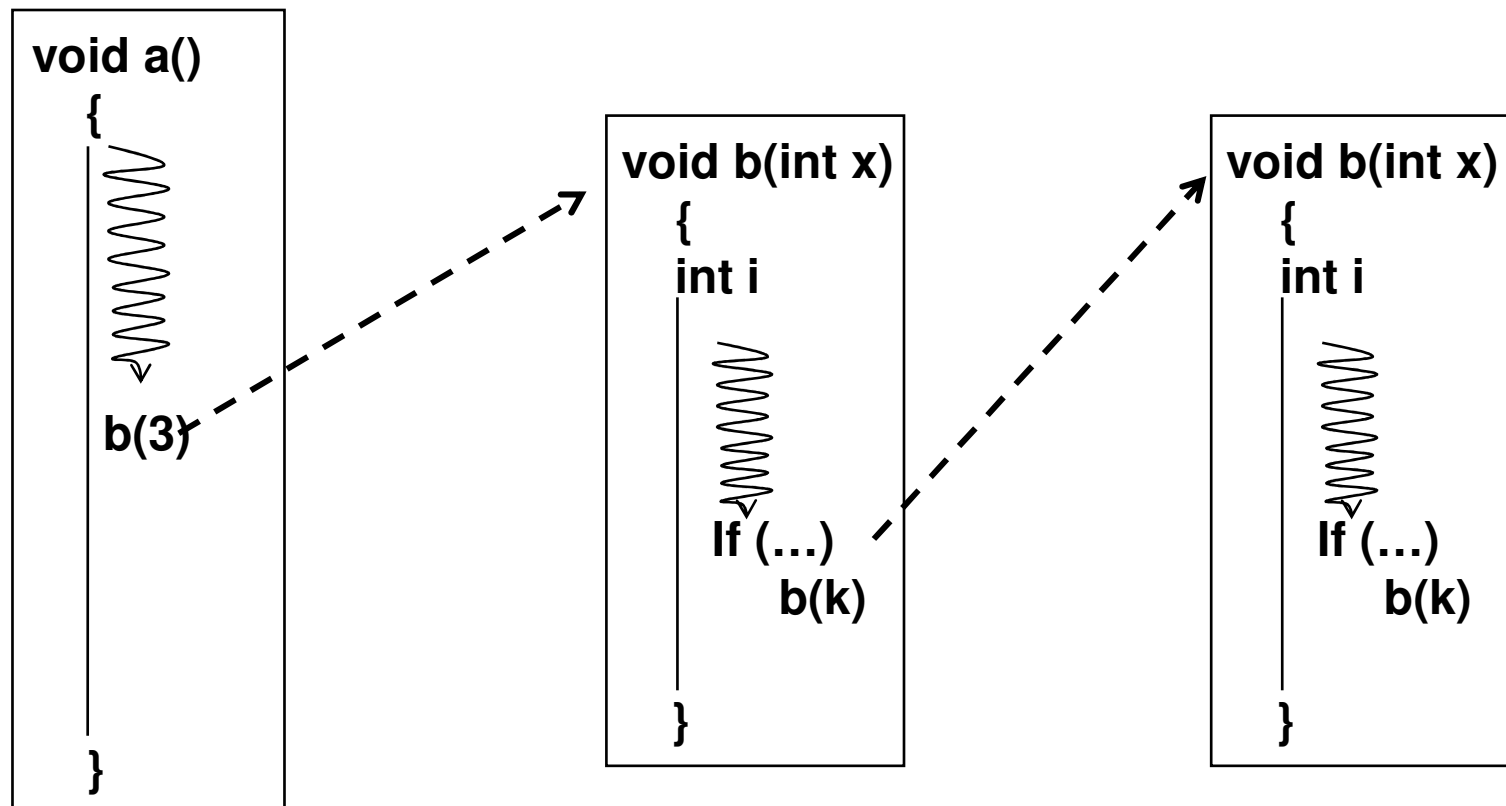
- ❖ As variáveis locais e parâmetros passados para a função devem existir somente enquanto durar a execução da função.



Os desafios da execução de subrotinas

3 – Instâncias independentes

- ❖ Cada instância de uma subrotina deve possuir suas próprias instâncias de variáveis locais e parâmetros.



Pilha de execução

Pilha de execução

❑ Finalidade

- ❖ Realizar o controle da execução de subrotinas de um programa:
 1. Controle do endereço de retorno da função;
 2. Passagem dos valores dos argumento da função;
 3. Alocação de variáveis locais à função.

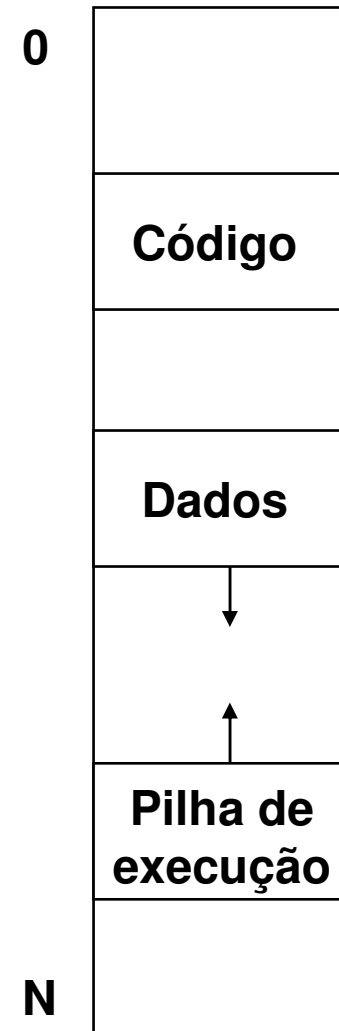
❑ Descrição

- ❖ Espaço de memória especialmente reservado para organização de uma pilha de quadros (frame) de ativação (ou registro de ativação).
- ❖ Cada quadro (frame) corresponde ao controle da ativação de uma função;
- ❖ Esta pilha de quadros é usada como memória auxiliar durante a execução do programa

Pilha de execução

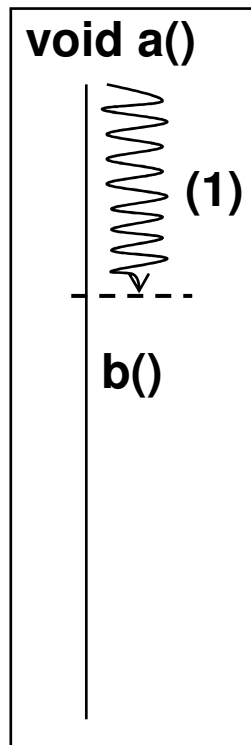
- ❑ Uma área exclusiva para a pilha de execução é reservada na memória virtual.
- ❑ A cada subrotina ativada, um “quadro” (frame) é criado na pilha de execução.

Memória virtual

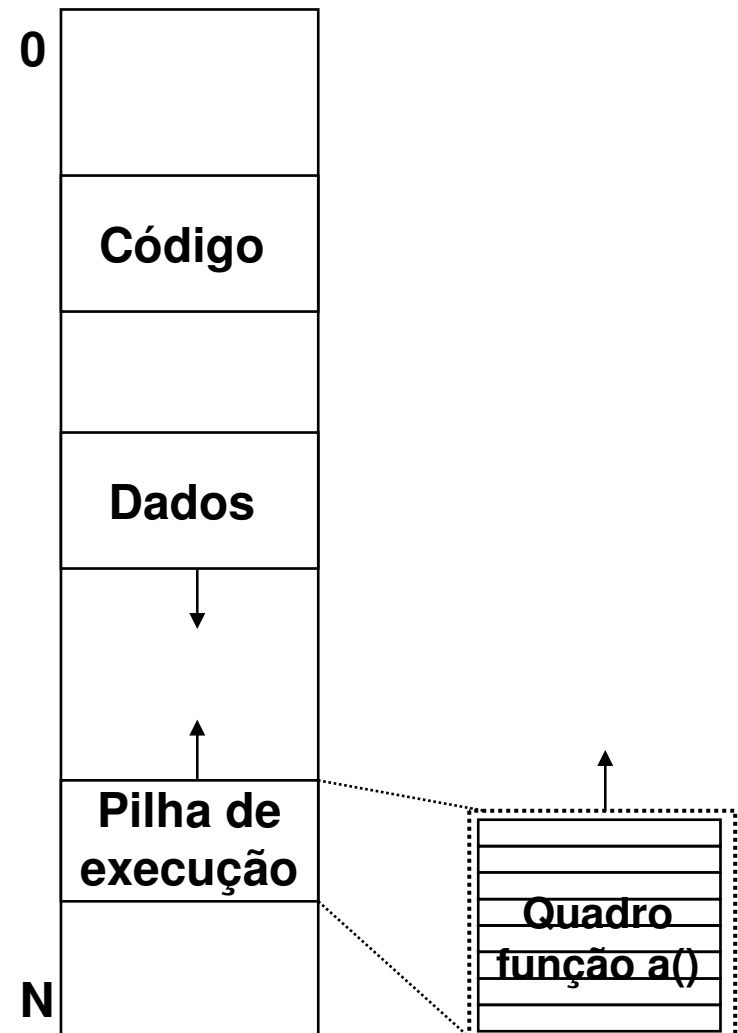


Pilha de execução

❑ Exemplo

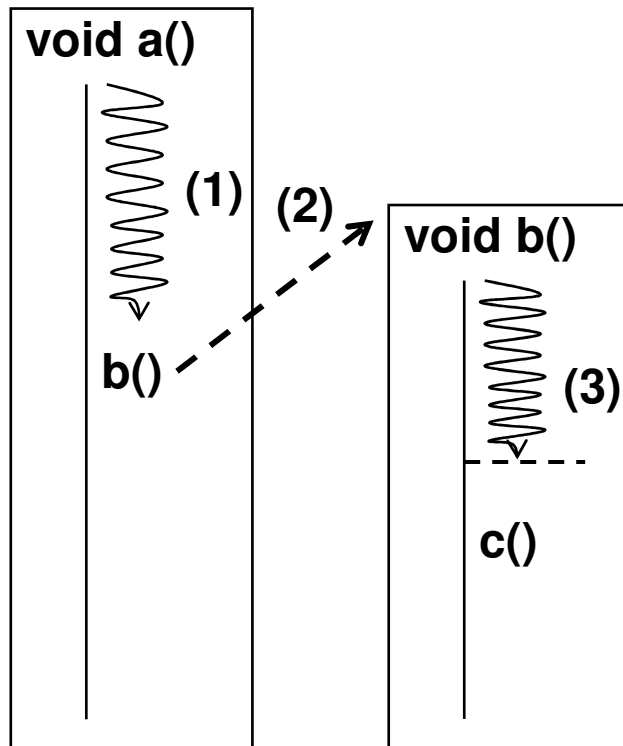


Memória virtual

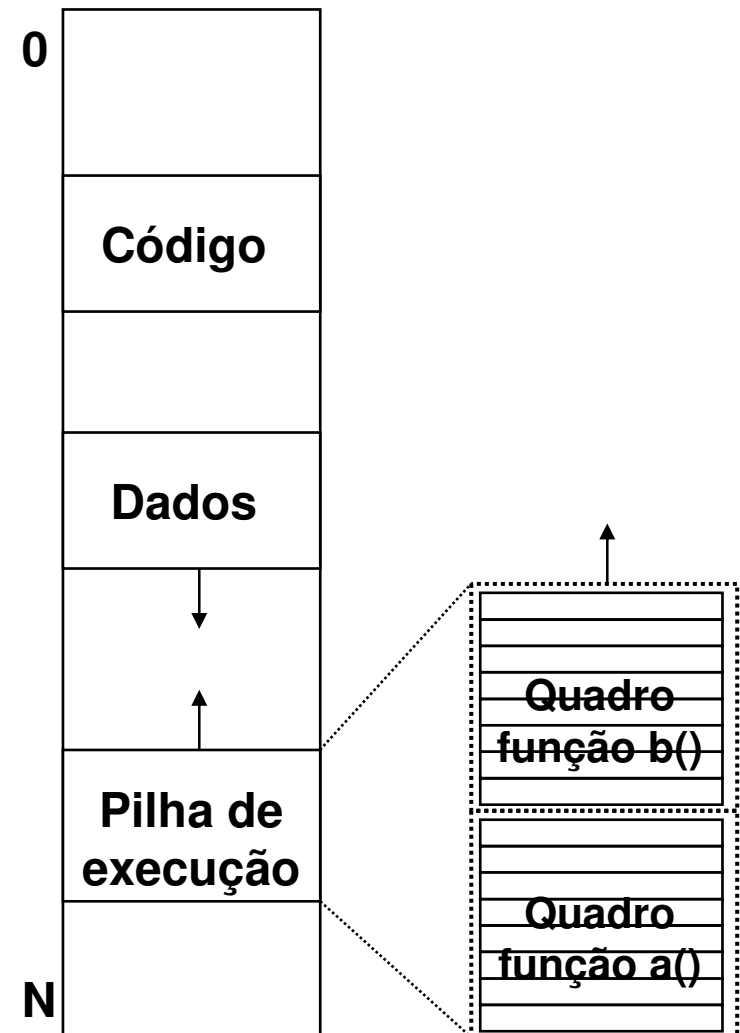


Pilha de execução

❑ Exemplo

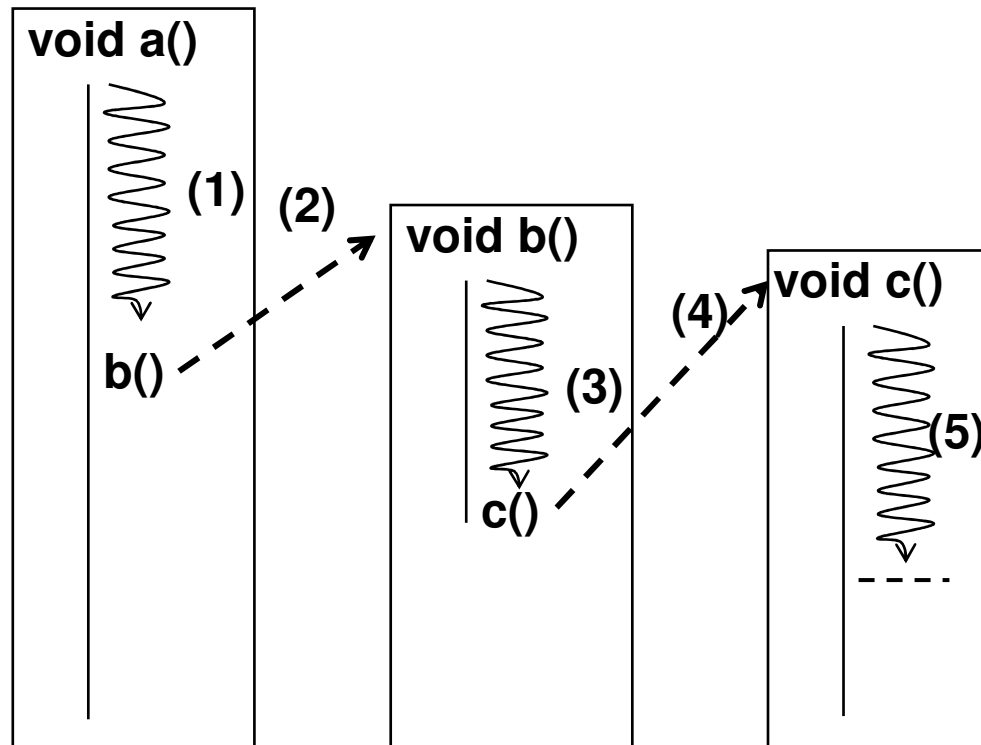


Memória virtual

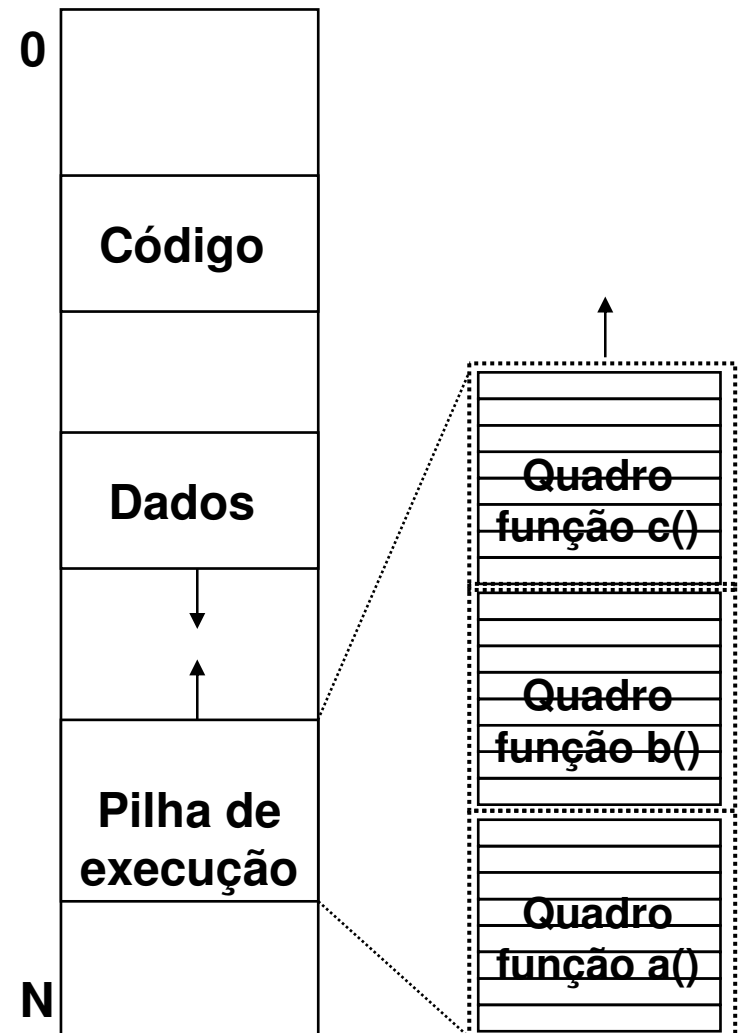


Pilha de execução

❑ Exemplo



Memória virtual



Pilha de execução

❑ Suporte pelo processador

❖ Instruções especiais para subrotinas:

- CALL – Chamada de subrotina:
 - Desvia o controle para uma subrotina salvando o endereço de retorno (endereço da próxima instrução) no topo da pilha.
- RET - Retorno de uma subrotina:
 - O controle do programa (PC) é transferido para o valor desempilhando do topo da pilha

❖ Registradores especiais para controle da pilha de execução:

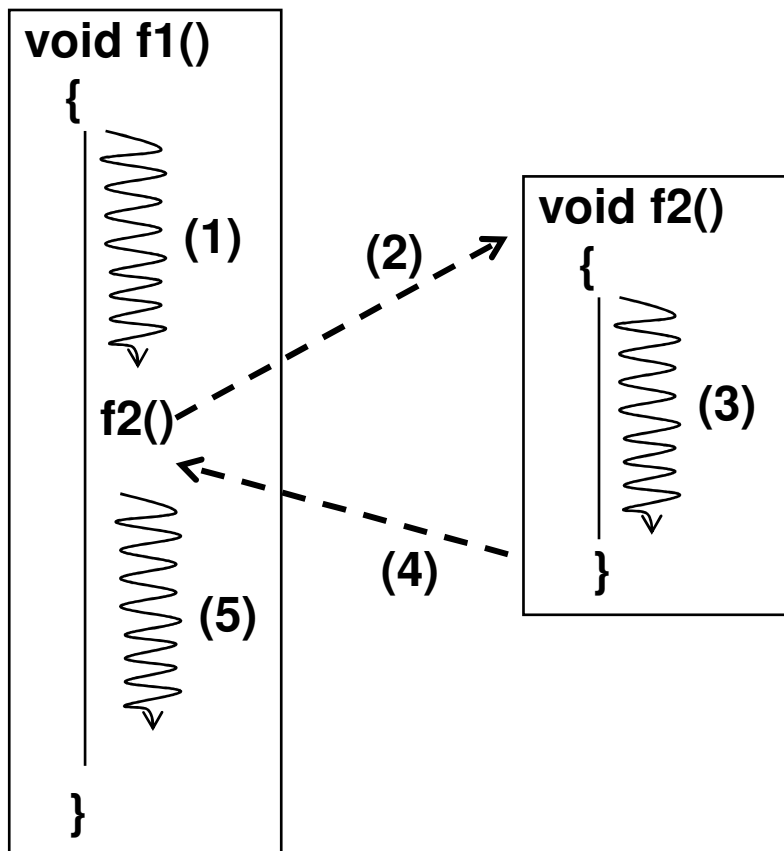
- SP (*stack pointer*): Contém o endereço do topo da pilha de execução

Controle do endereço de retorno da função

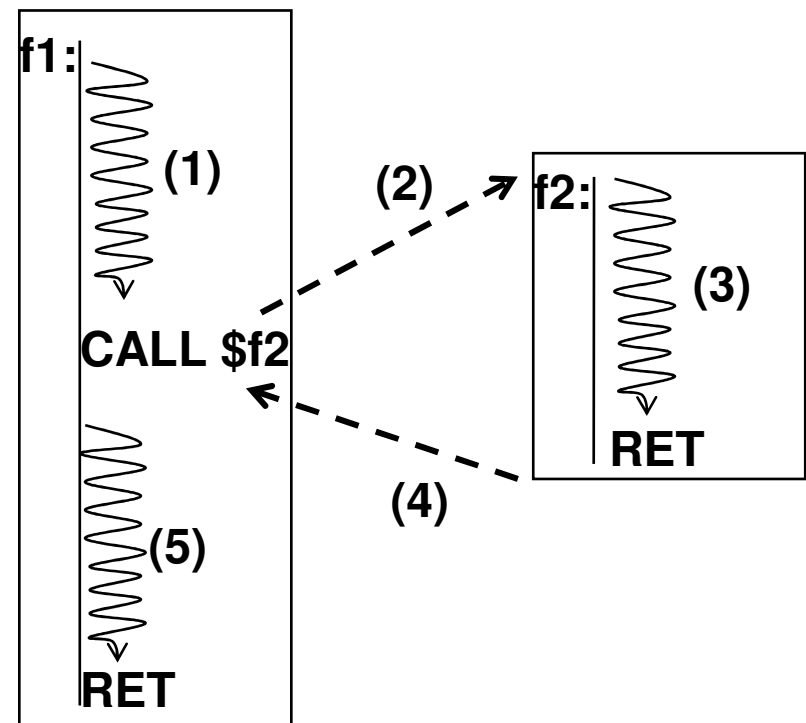
Controle do endereço de retorno

- ❑ Realizado pelas instruções **CALL** e **RET**

Programa em C

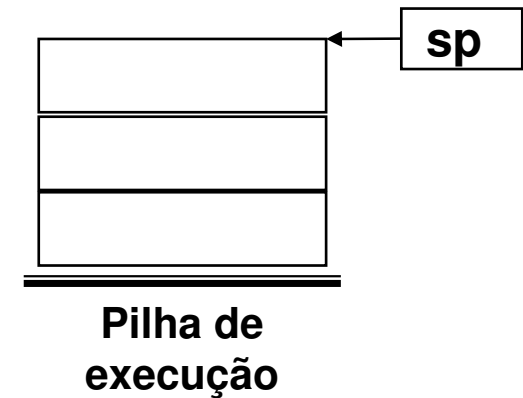
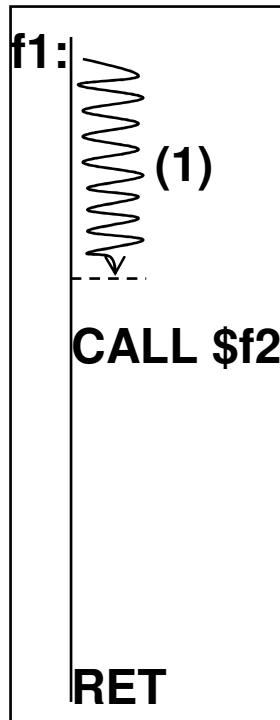


Programa em assembler



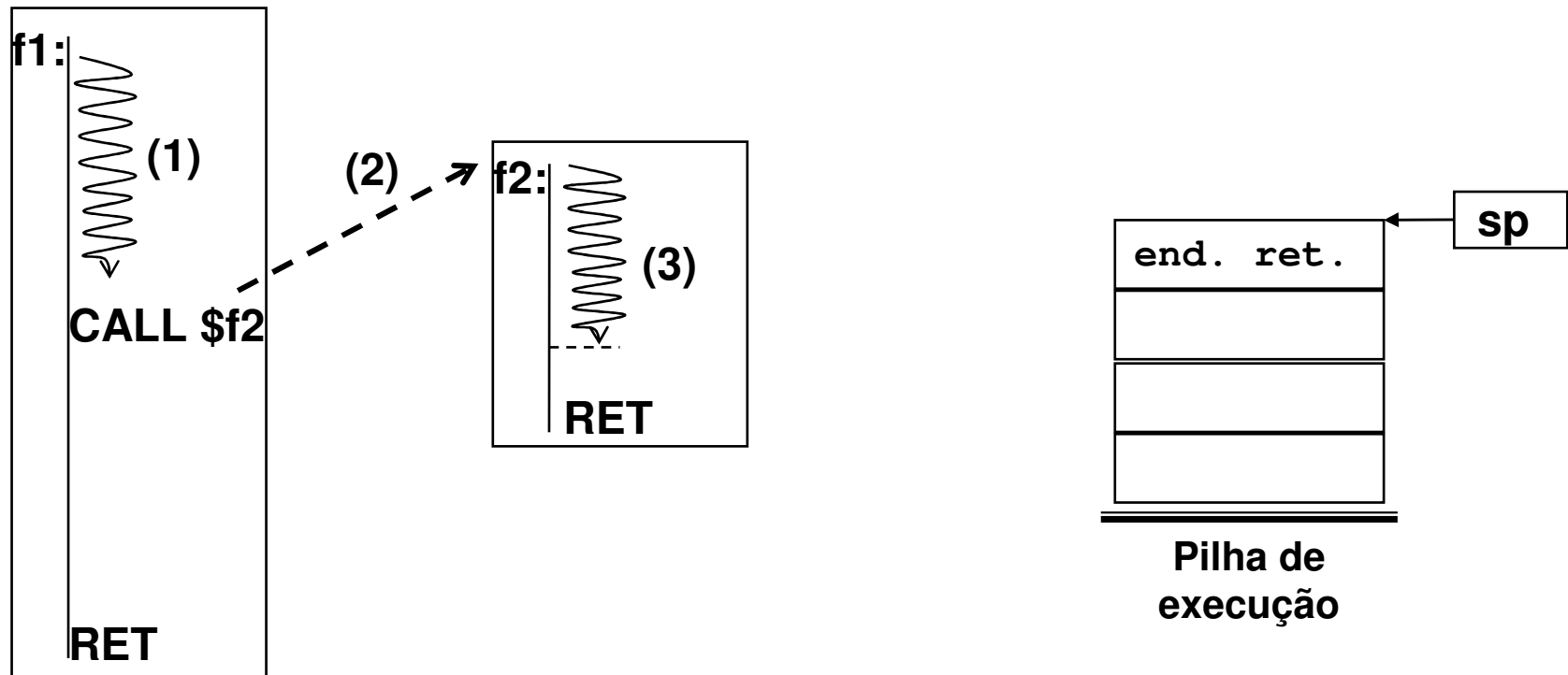
Controle do endereço de retorno

❑ Exemplo de funcionamento das instruções CALL e RET



Controle do endereço de retorno

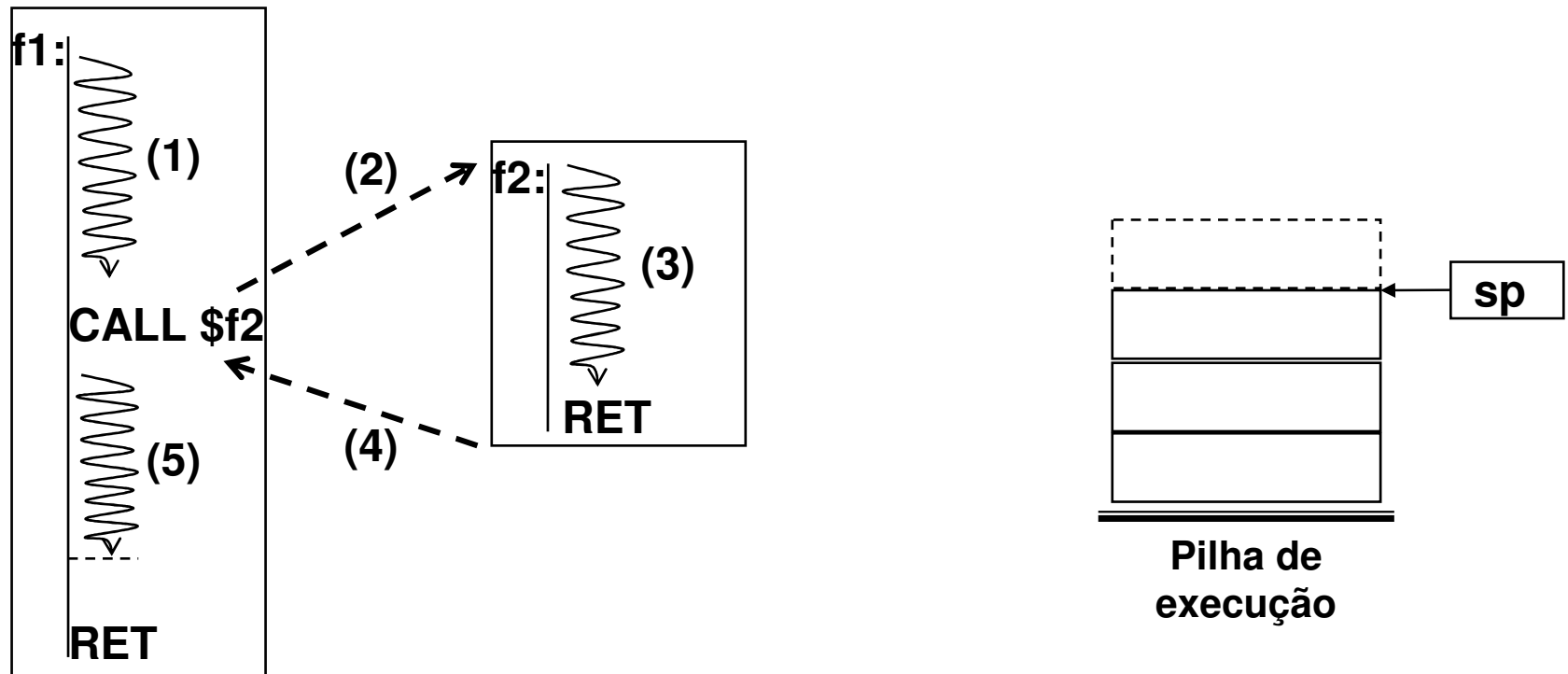
❑ Exemplo de funcionamento das instruções CALL e RET



A instrução CALL salva o endereço de retorno (endereço da instrução após CALL) na pilha de execução e transfere o controle para f2.

Controle do endereço de retorno

❑ Exemplo de funcionamento das instruções CALL e RET



A instrução RET retira o valor contido no topo da pilha de execução (endereço de retorno) e transfere o controle para o endereço representado por este valor.

Passagem dos valores dos argumento para a função

Passagem dos valores dos argumentos

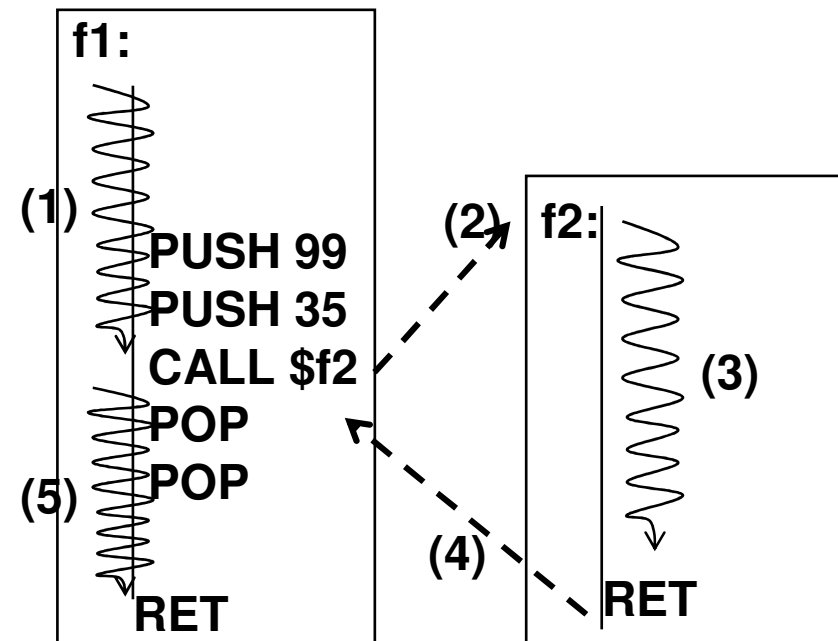
- ❑ A pilha de execução é utilizada, também, para passagem dos argumentos da função.**
- ❑ Os valores dos argumentos são inseridos na pilha de execução antes da ativação da instrução CALL.**

Passagem dos valores dos argumentos

❑ Exemplo

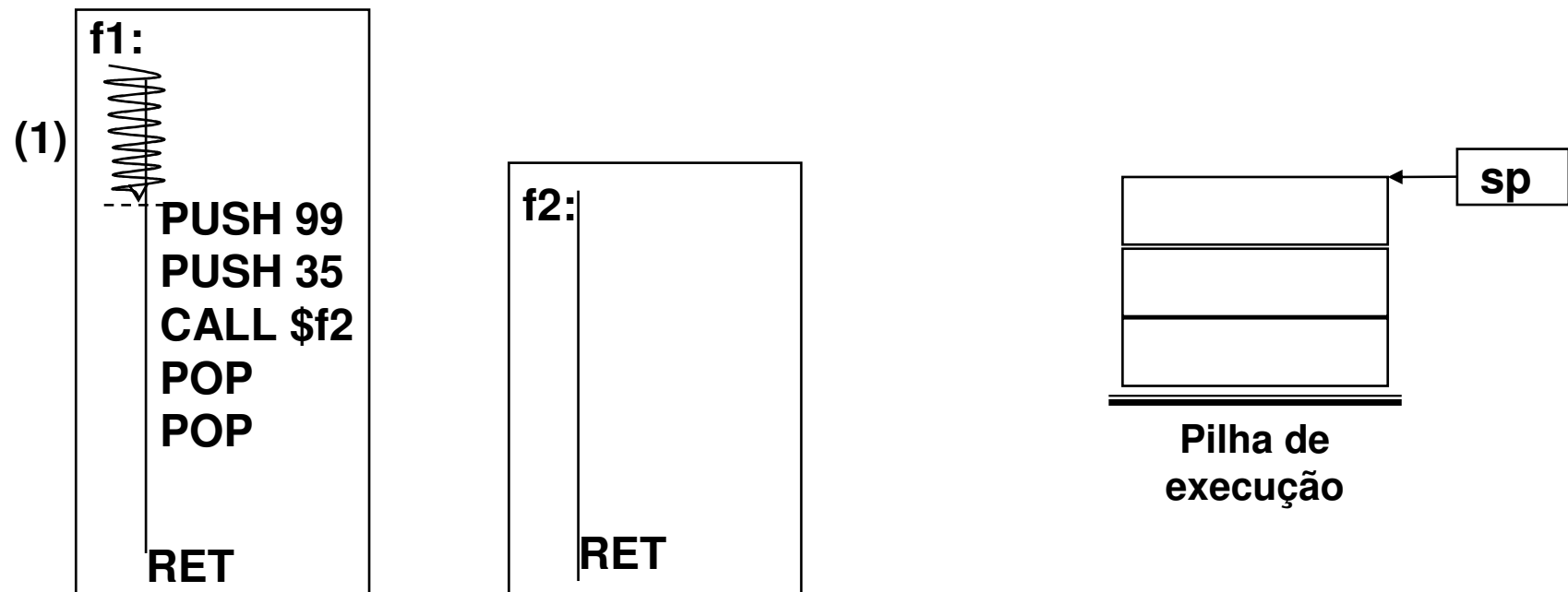
```
void f2(int a, int b)
{
    ...
}

int f1()
{
    ...
    f2(35, 99);
    ...
}
```



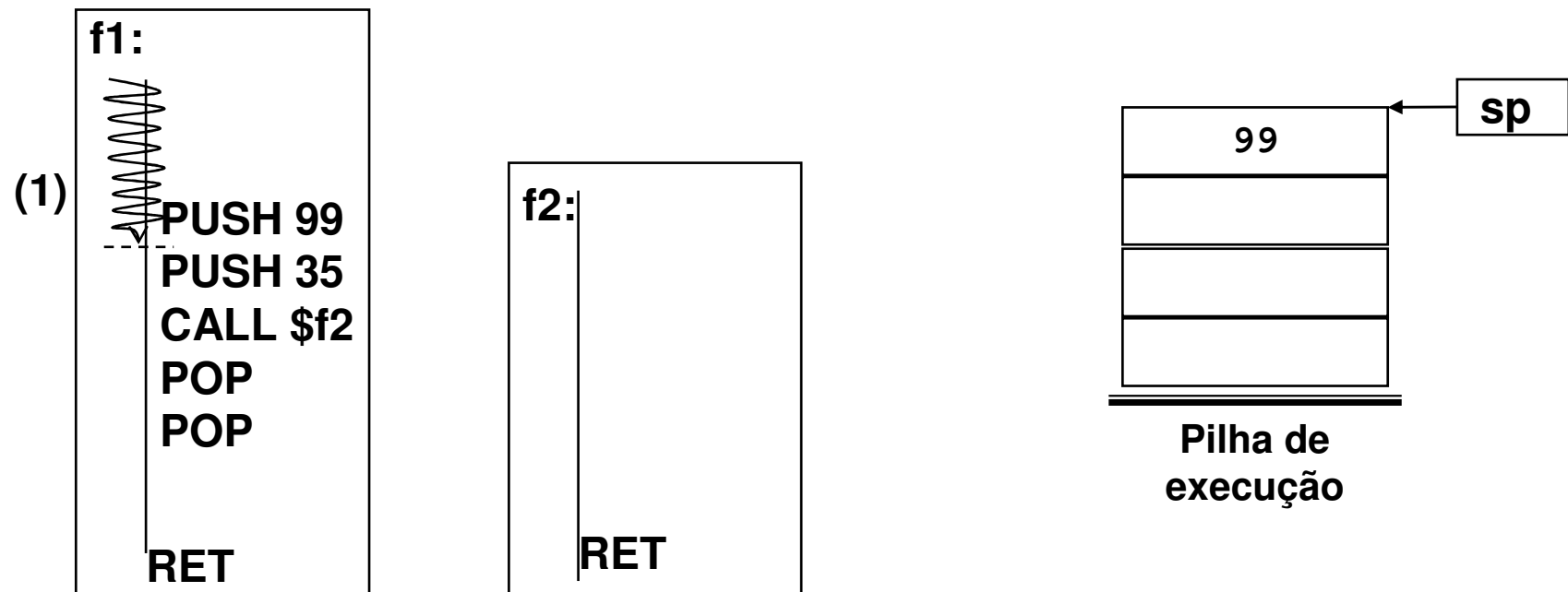
Passagem dos valores dos argumentos

□ Exemplo



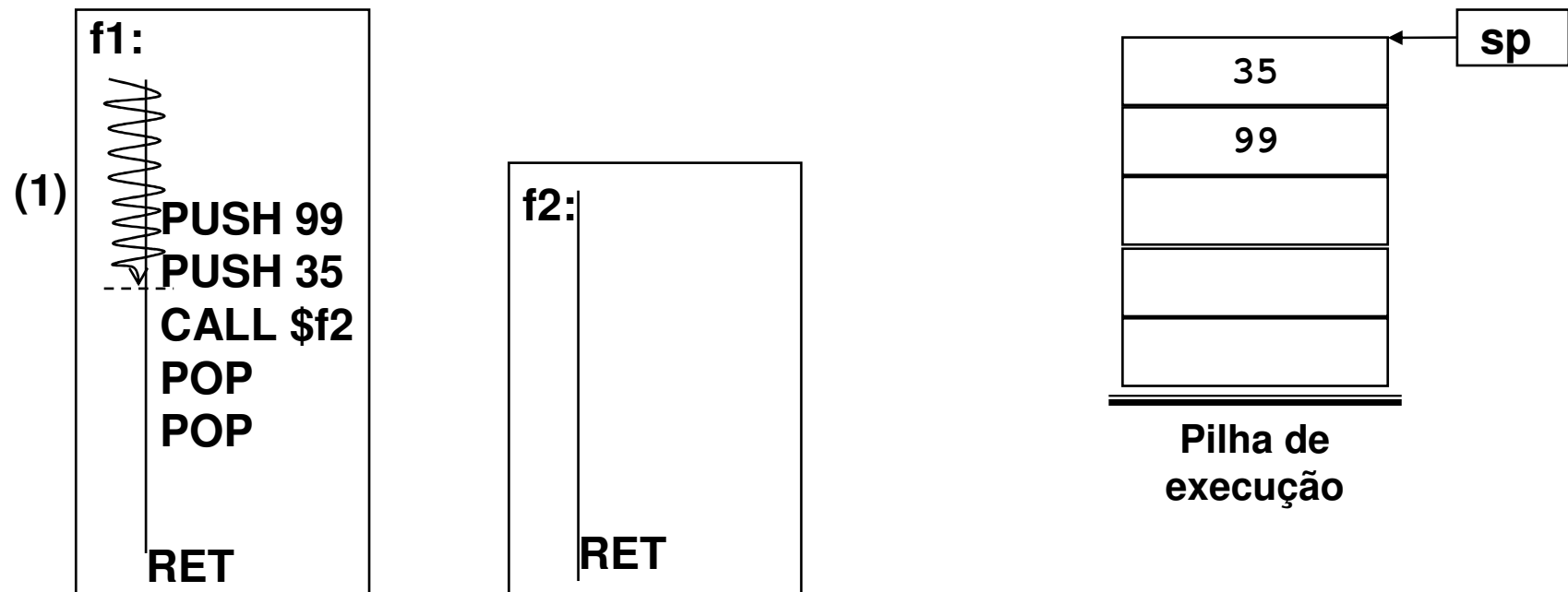
Passagem dos valores dos argumentos

❑ Exemplo



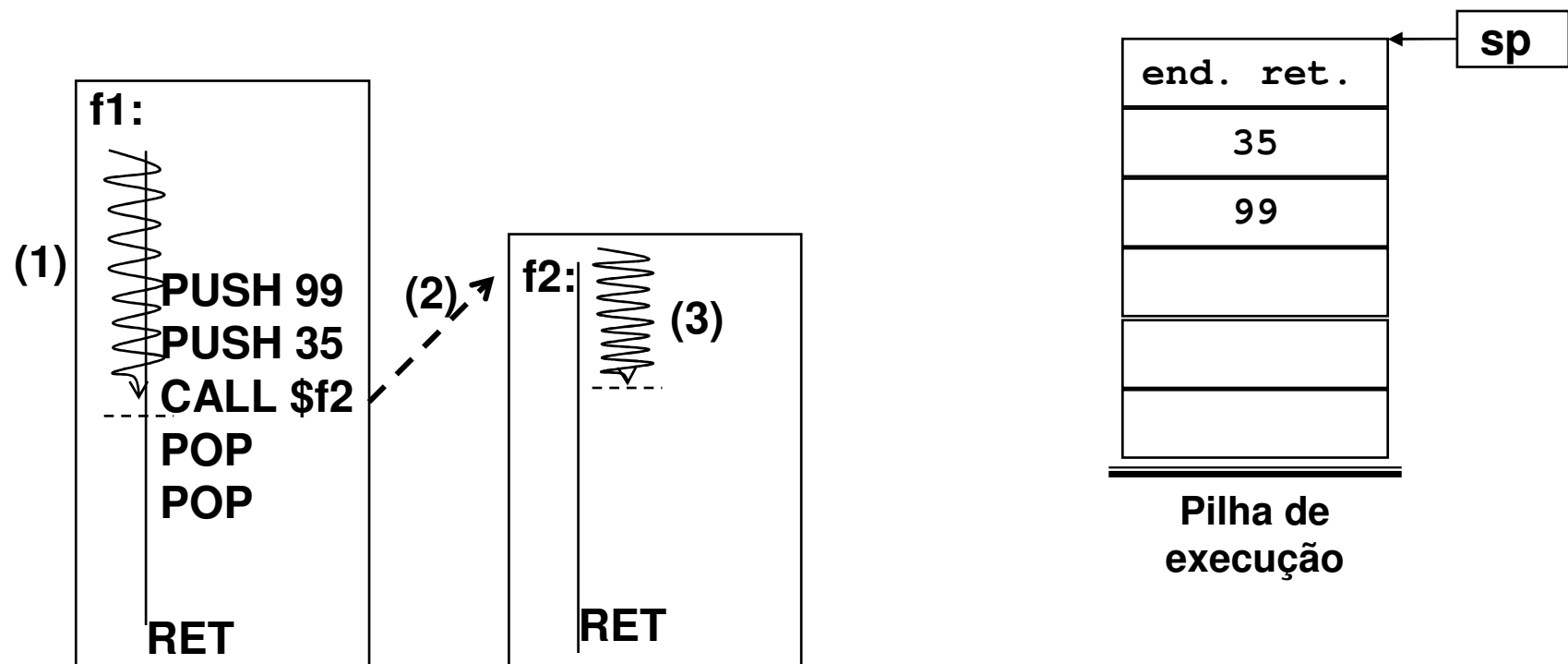
Passagem dos valores dos argumentos

❑ Exemplo



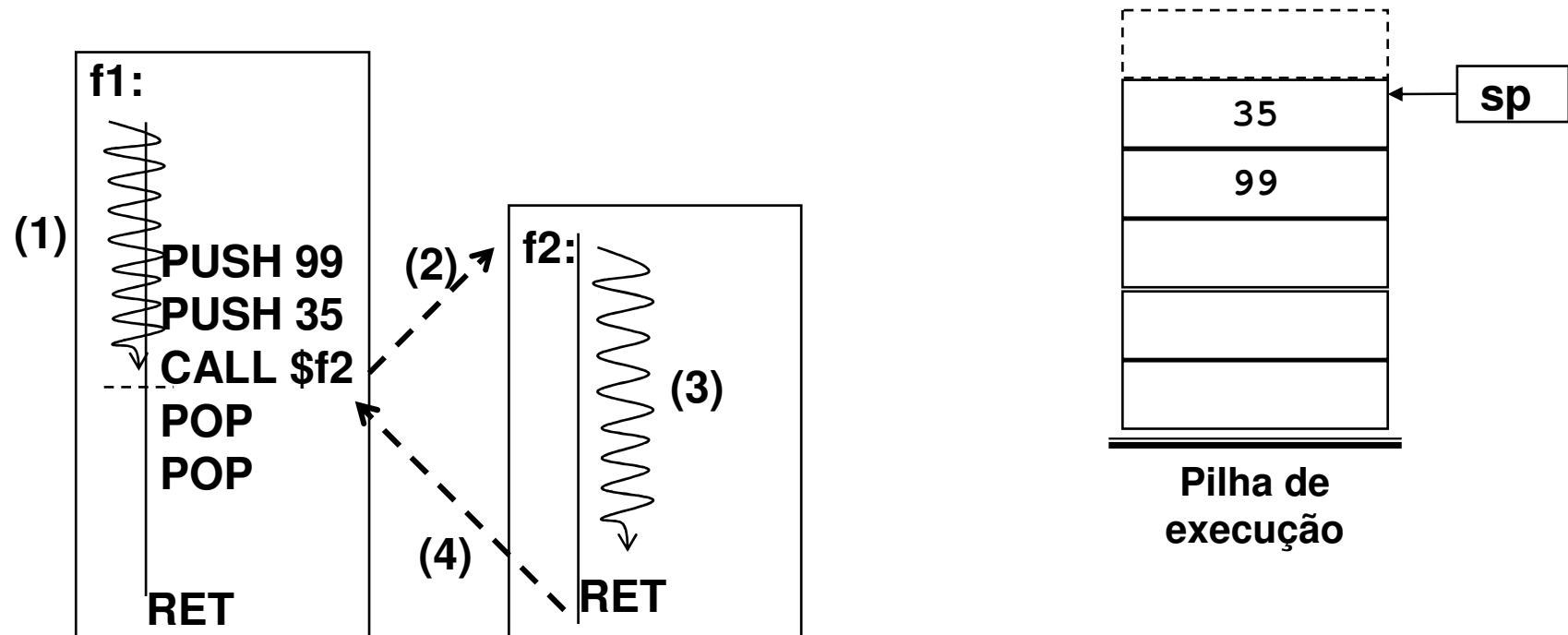
Passagem dos valores dos argumentos

□ Exemplo



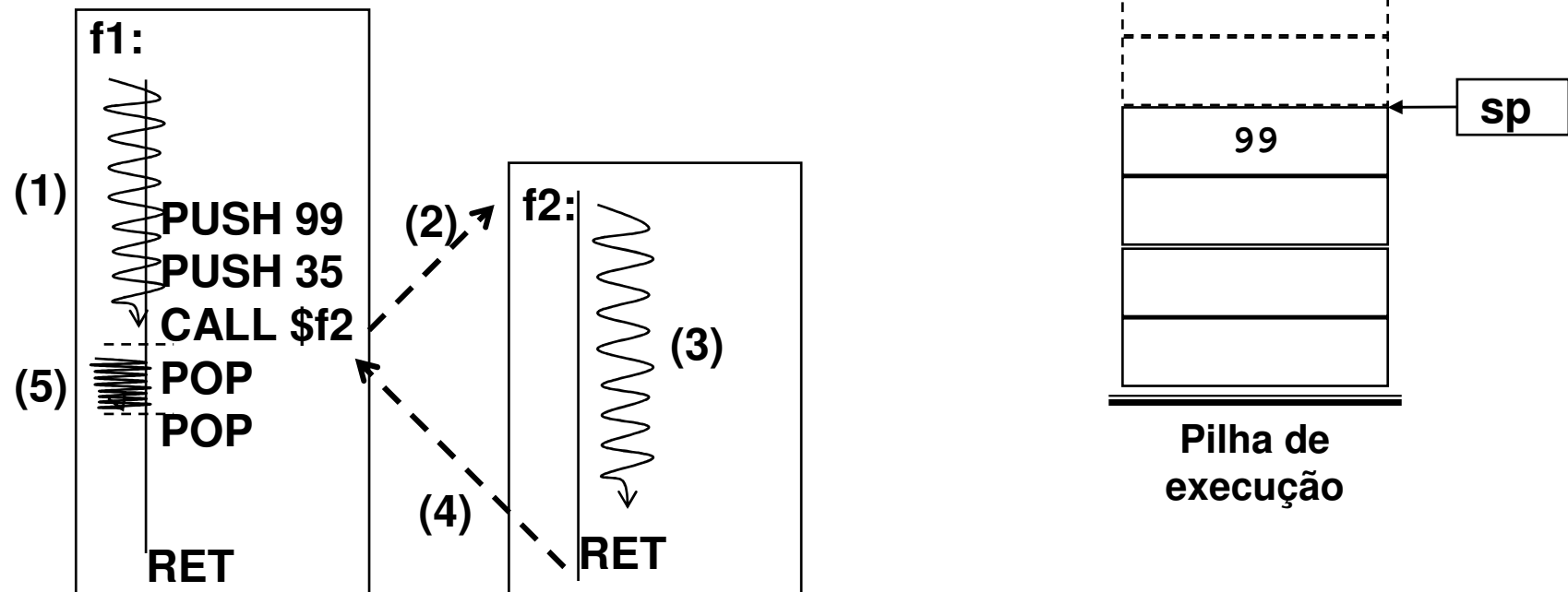
Passagem dos valores dos argumentos

□ Exemplo



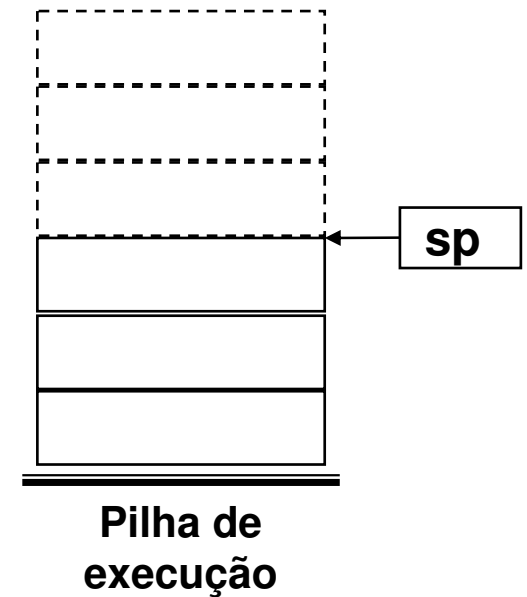
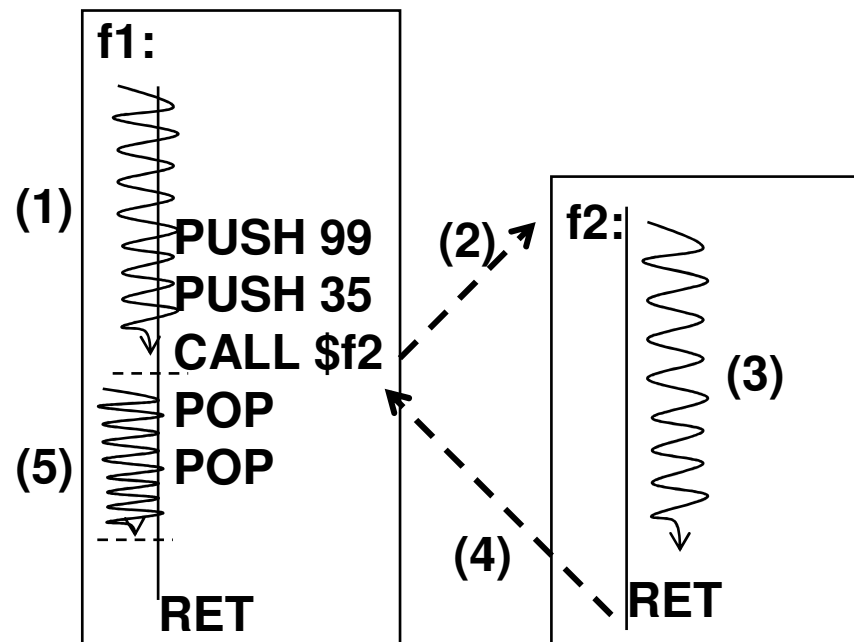
Passagem dos valores dos argumentos

□ Exemplo



Passagem dos valores dos argumentos

□ Exemplo



Alocação de variáveis locais

Alocação de variáveis locais

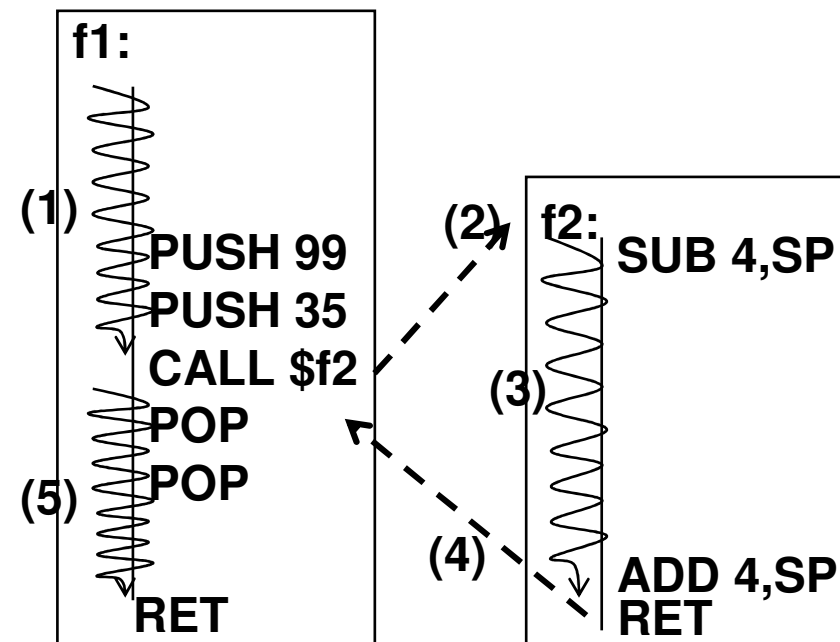
- ❑ Variáveis locais também são alocadas na pilha de execução.**
- ❑ As variáveis locais de uma função ficam alocadas no quadro da respectiva função na pilha de execução**

Alocação de variáveis locais

□ Exemplo

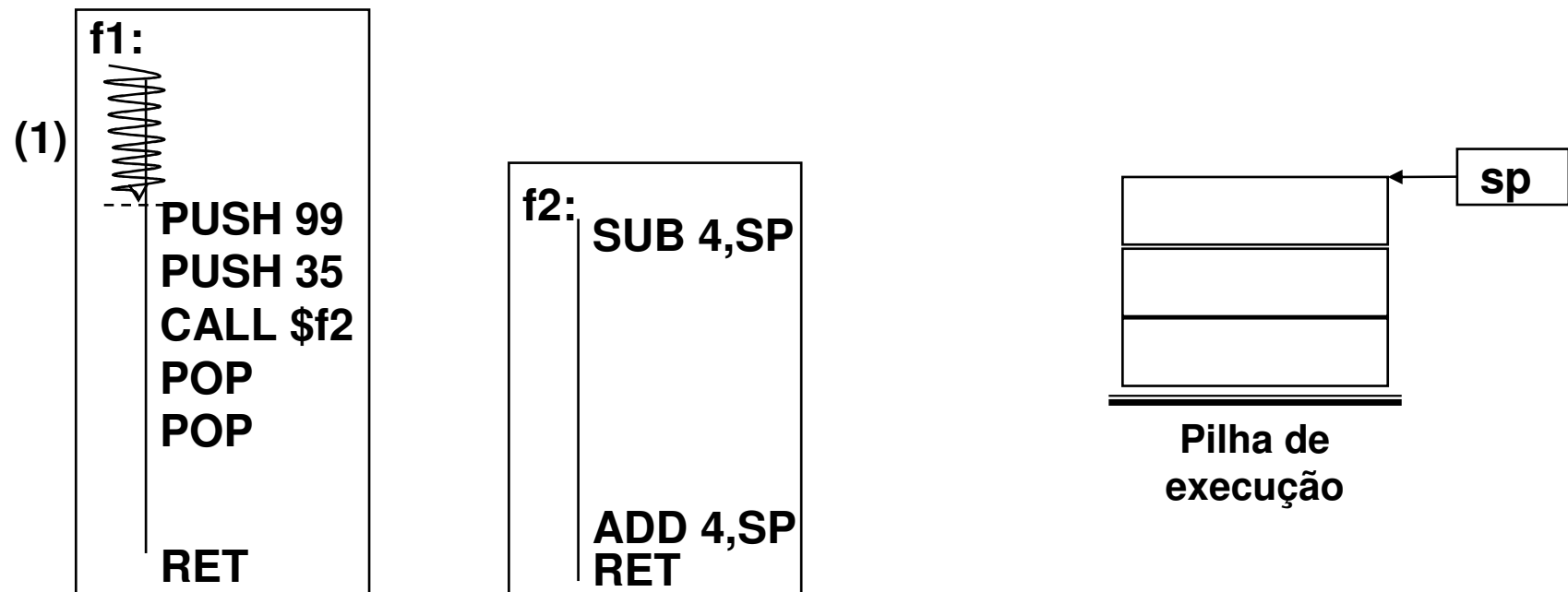
```
void f2(int a, int b)
{
    int x;
    ...
}

int f1()
{
    ...
    f2(35, 99);
    ...
}
```



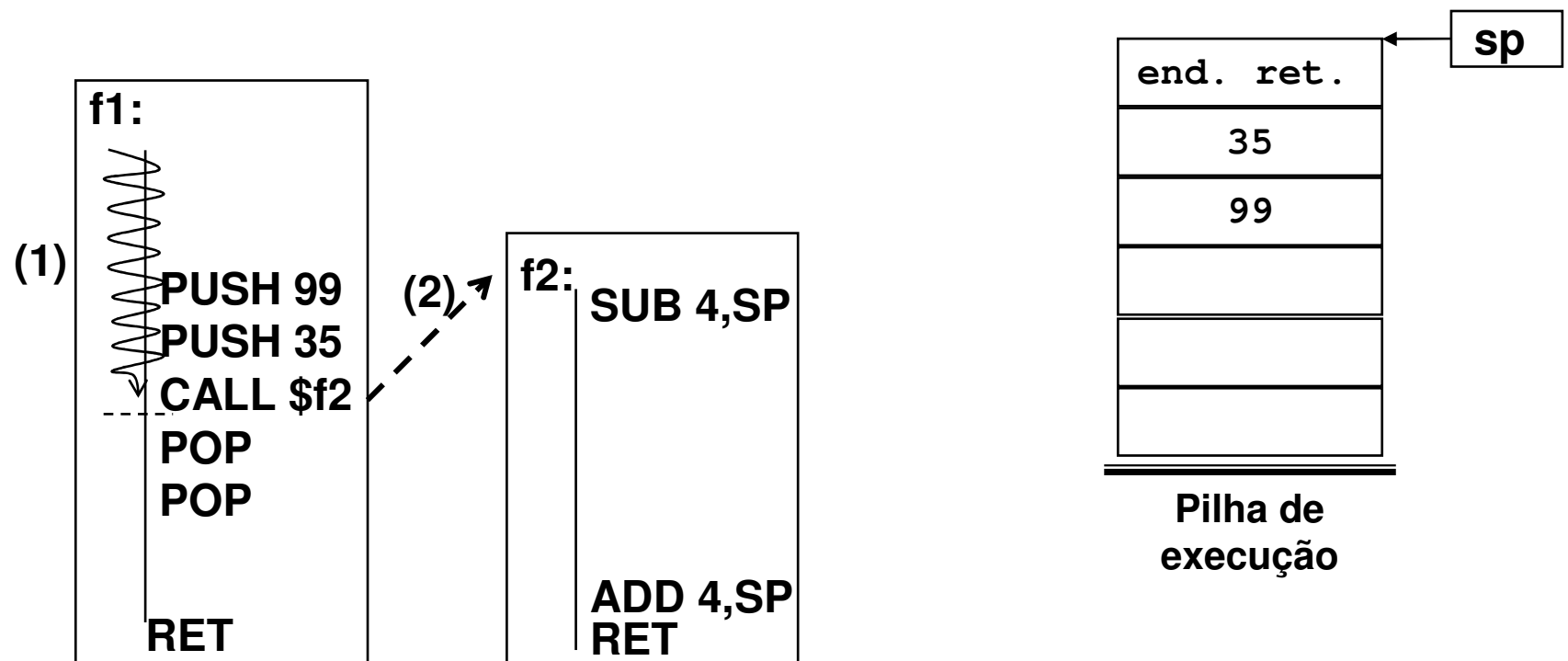
Alocação de variáveis locais

□ Exemplo



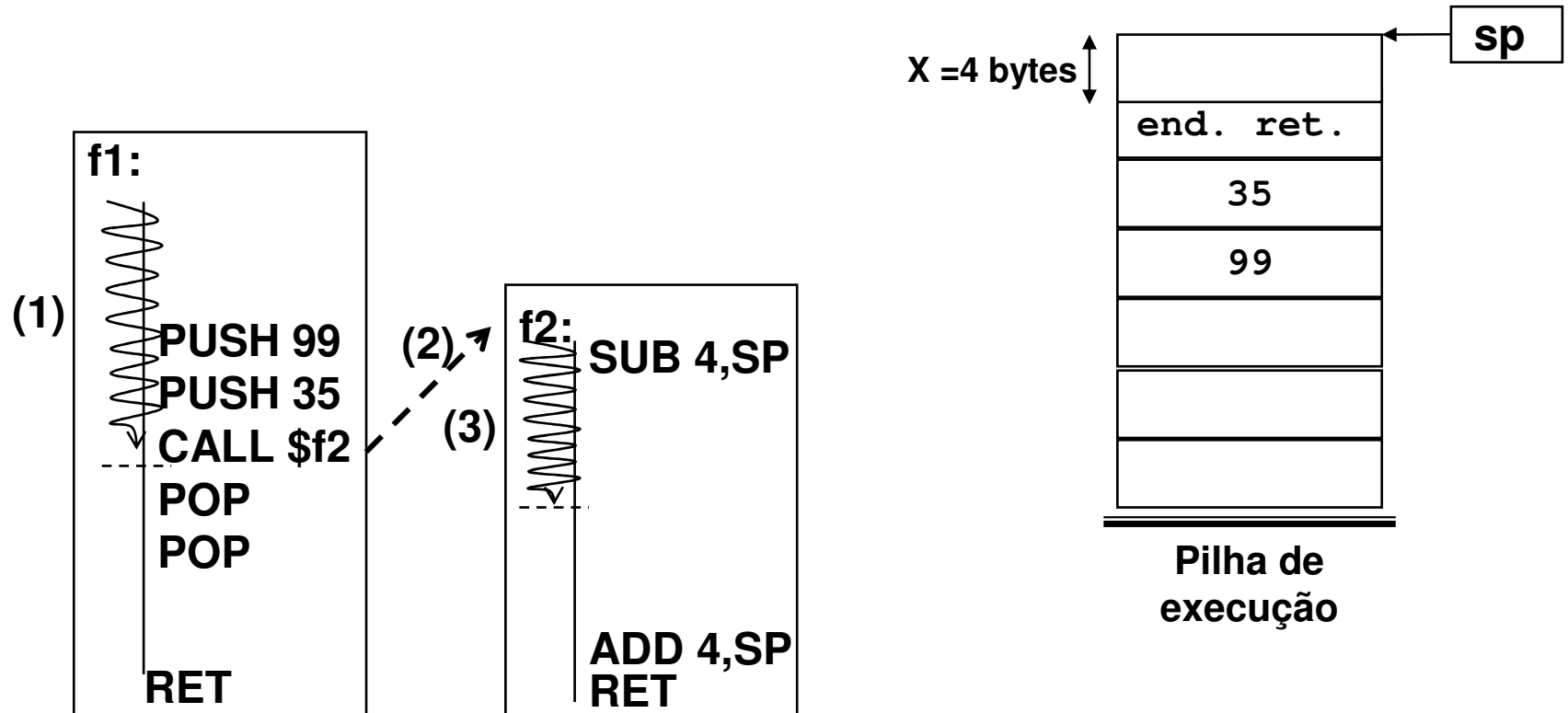
Alocação de variáveis locais

□ Exemplo



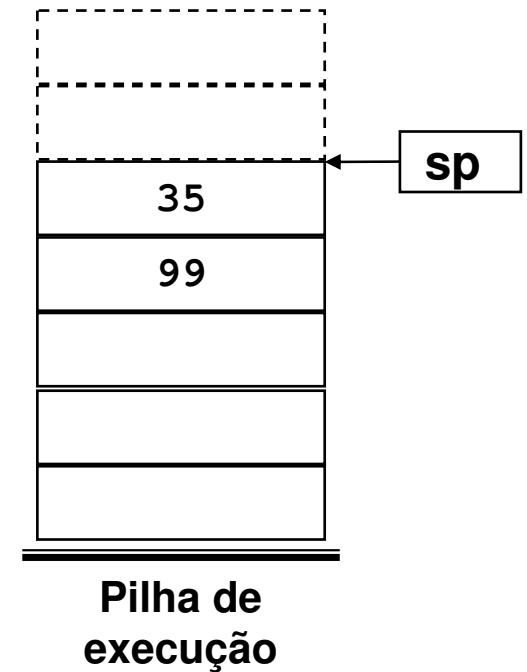
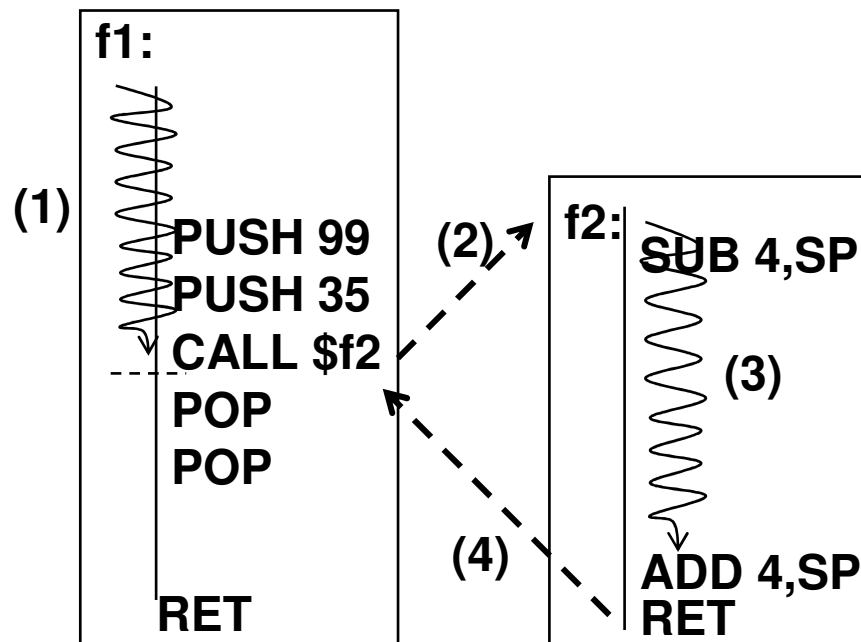
Alocação de variáveis locais

□ Exemplo



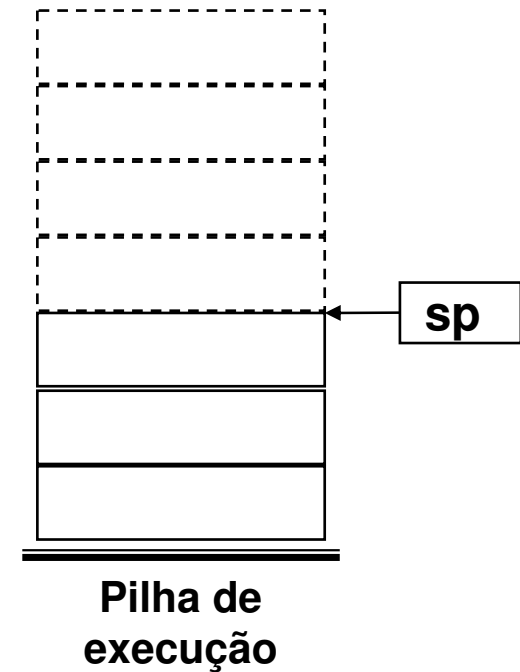
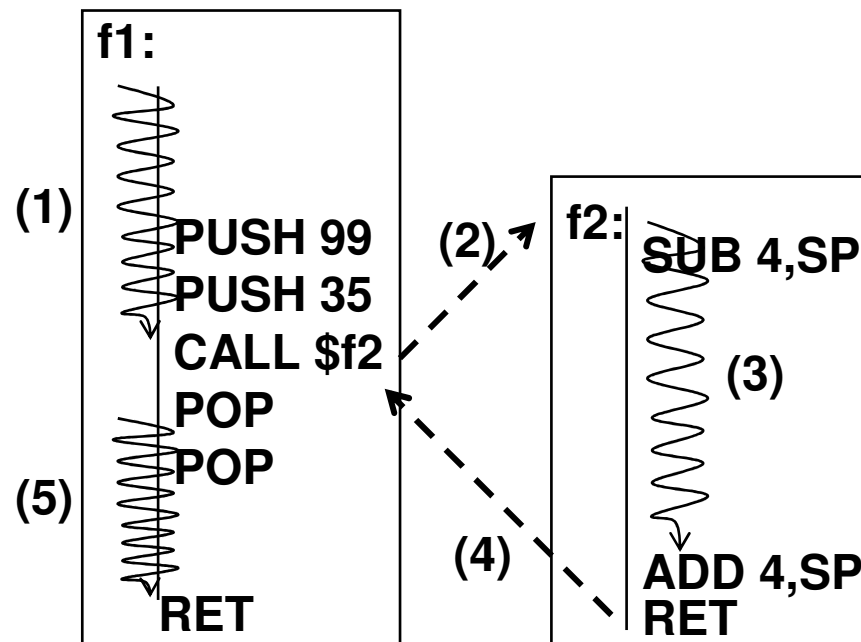
Alocação de variáveis locais

□ Exemplo



Alocação de variáveis locais

□ Exemplo



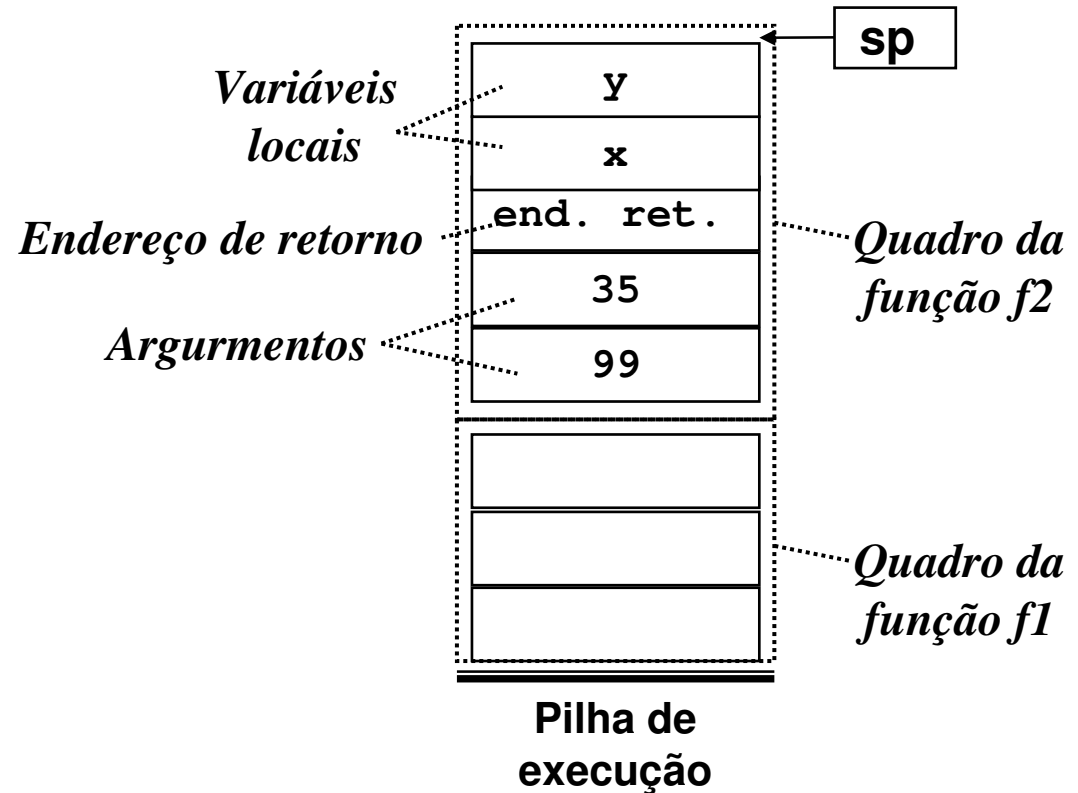
Quadro da pilha de execução

Quadro da pilha de execução

❑ Exemplo de quadro da pilha

```
void f2(int a, int b)
{
    int x;
    int y;
    ...
}
```

```
int f1()
{
    ...
    f2(35, 99);
    ...
}
```



Exercício

Exercício

- (1) Mostre graficamente a evolução da área de dados e da pilha de execução decorrente da execução do programa perímetro.c.**

Exercício

```
// Programa perimetro.c
// Calcula o perímetro de uma circunferencia

#include <stdio.h>

float      raio;
float      per;
const float pi = 3.1415912;

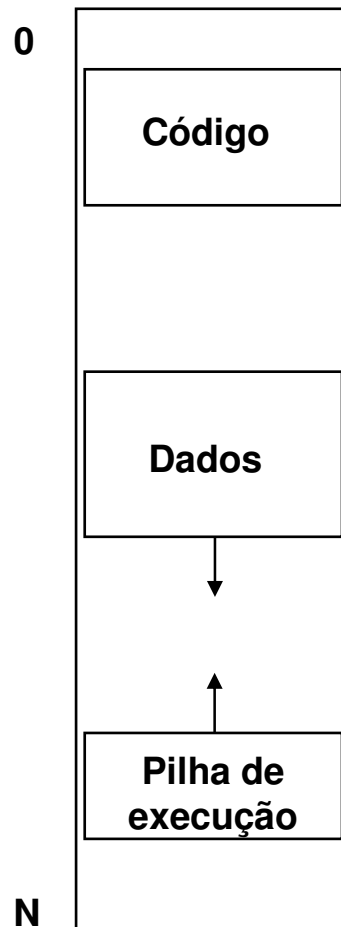
float perimetro(float r)
{
    float p; /* perimetro */

    p = 2 * pi * r;
    return(p);
}

int main()
{
    printf("Entre com o valor do raio: ");
    scanf("%f",&raio);
    per = perimetro(raio);
    printf("Perímetro: %3.2f \n", per);
}
```

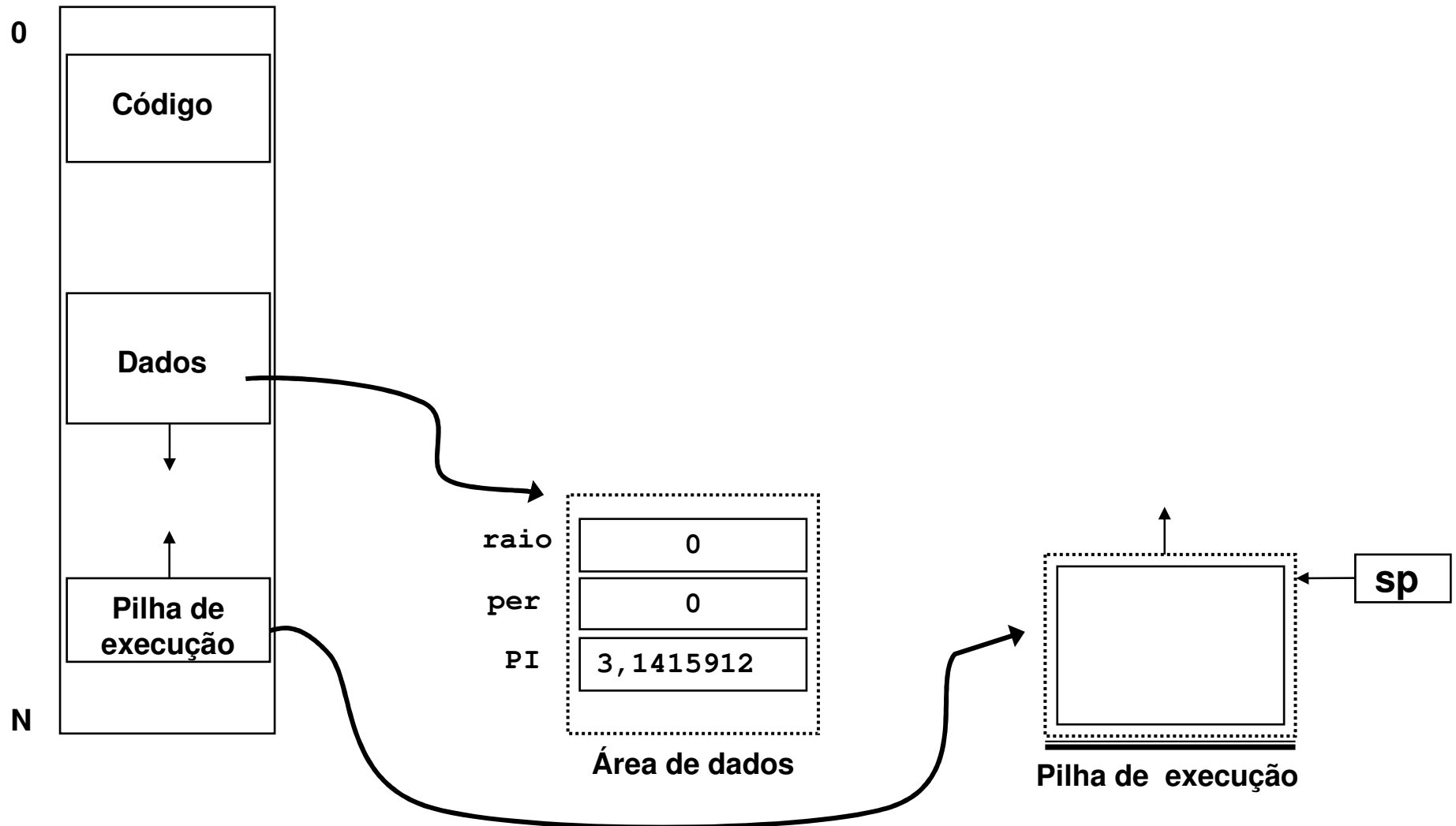
Exercício

Memória virtual



Exercício

Memória virtual



Exercício

(2) Mostre graficamente a evolução da área de dados e da pilha de execução decorrente da execução do programa “fatorial” para o cálculo de fatorial de 3.

Exercício

```
#include <stdio.h>
char versao[] = "2.1";
int n;
int resultado;

int fatorial (int x)
{
    int y;

    if (x <= 1)
        y = 1;
    else
        y = x * fatorial(x-1);
    return(y);
}

int main(int argc, char **argv)
{
    printf("Programa fatorial, versao %s \n", versao);
    printf("Entre com o valor: " );
    scanf("%d",&n);
    resultado = fatorial(n);
    printf("Resultado: %d \n",resultado);
}
```

Controle do quadro da pilha de execução na arquitetura Intel Pentium

Controle do quadro da pilha de execução

❑ Exemplo - Processador Intel Pentium

❖ Registradores especiais:

- Registrador ESP: contém o endereço do topo da pilha
- Registrador EBP: contém a base do quadro

❖ Sentido do crescimento da pilha

- Por motivos históricos, a pilha geralmente cresce em direção aos endereços menores de memória.
- Assim, para alocar espaço para um endereço na pilha, devemos subtrair 4 de ESP no Pentium (um endereço no Intel Pentium ocupa 4 bytes!). Para desalocar devemos somar 4 ao ESP.

Controle do quadro da pilha de execução

❑ Arquitetura Intel Pentium

❖ Registrador ebp

- O registrador ebp é utilizado como referência para o quadro de ativação corrente
- O código gerado por um compilador na execução de uma chamada de função começa com:

```
pushl %ebp  
movl  %esp, %ebp
```

- E termina com:

```
movl %ebp, %esp  
popl %ebp
```


Controle do quadro da pilha de execução

```
void troca (int *x, int *y)
{
    int tmp;

    tmp = *x;
    *x = *y;
    *y = tmp;
}
```

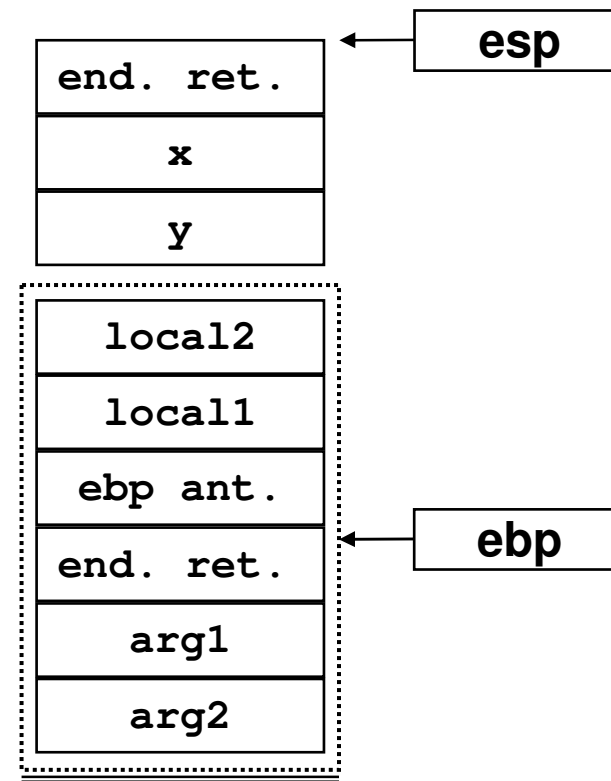
Exemplo: Arquitetura Intel Pentium

```
troca: push %ebp
       mov  %esp, %ebp
       sub  $4, %esp          /* reserva espaço na pilha para tmp */
       mov  8(%ebp), %eax      /* 1o parâmetro: endereço de x */
       mov  (%eax), %edx       /* pega valor de x */
       mov  %edx, -4(%ebp)     /* tmp = *x */
       mov  12(%ebp), %ebx     /* 2o parâmetro: endereço de y */
       mov  (%ebx), %edx       /* pega valor de y */
       mov  %edx, (%eax)       /* *x = *y */
       mov  -4(%ebp), %edx     /* leitura do valor de tmp */
       mov  %edx, (%ebx)       /* *y = tmp */
       mov  %ebp, %esp
       pop  %ebp
       ret
```

Controle do quadro da pilha de execução

Exemplo: Arquitetura Intel Pentium

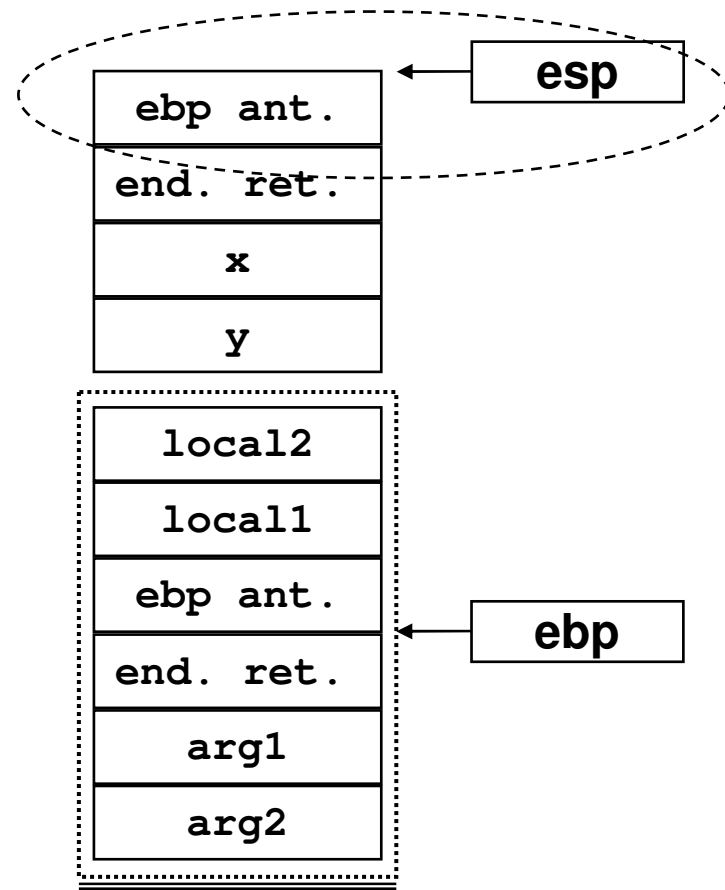
```
troca: push    %ebp
        mov     %esp, %ebp
        sub     $4, %esp
        mov     8(%ebp), %eax
        mov     (%eax), %edx
        mov     %edx, -4(%ebp)
        mov     12(%ebp), %ebx
        mov     (%ebx), %edx
        mov     %edx, (%eax)
        mov     -4(%ebp), %edx
        mov     %edx, (%ebx)
        mov     %ebp, %esp
        pop     %ebp
        ret
```



Controle do quadro da pilha de execução

Exemplo: Arquitetura Intel Pentium

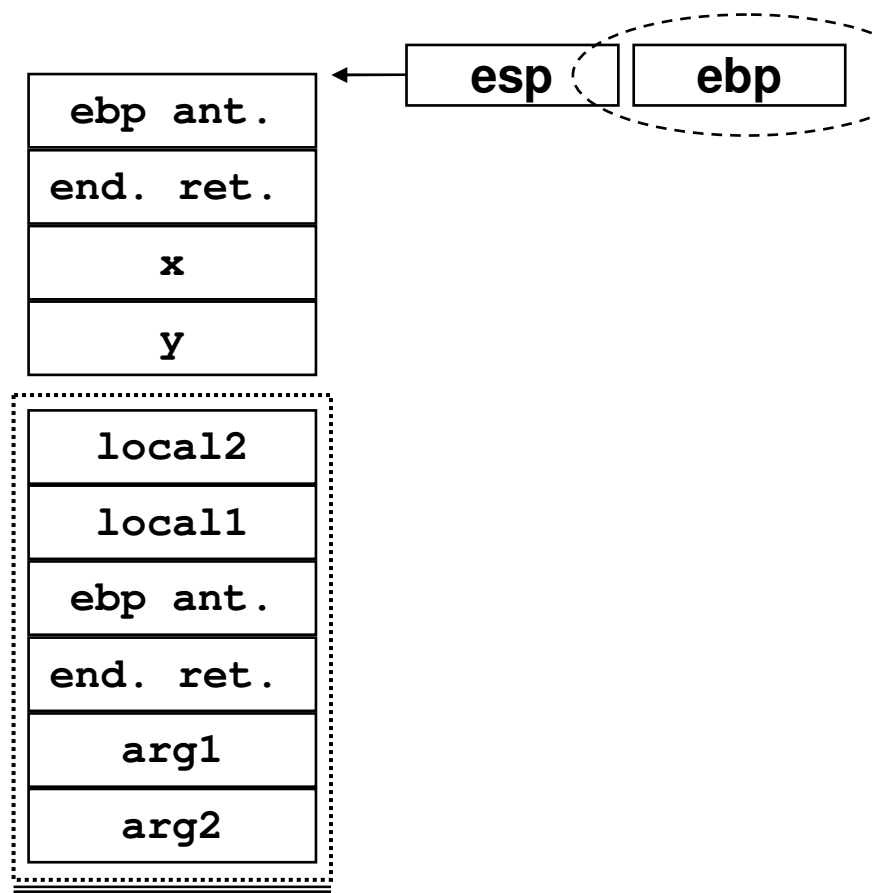
```
troca: push    %ebp
      mov     %esp, %ebp
      sub     $4, %esp
      mov     8(%ebp), %eax
      mov     (%eax), %edx
      mov     %edx, -4(%ebp)
      mov     12(%ebp), %ebx
      mov     (%ebx), %edx
      mov     %edx, (%eax)
      mov     -4(%ebp), %edx
      mov     %edx, (%ebx)
      mov     %ebp, %esp
      pop     %ebp
      ret
```



Controle do quadro da pilha de execução

Exemplo: Arquitetura Intel Pentium

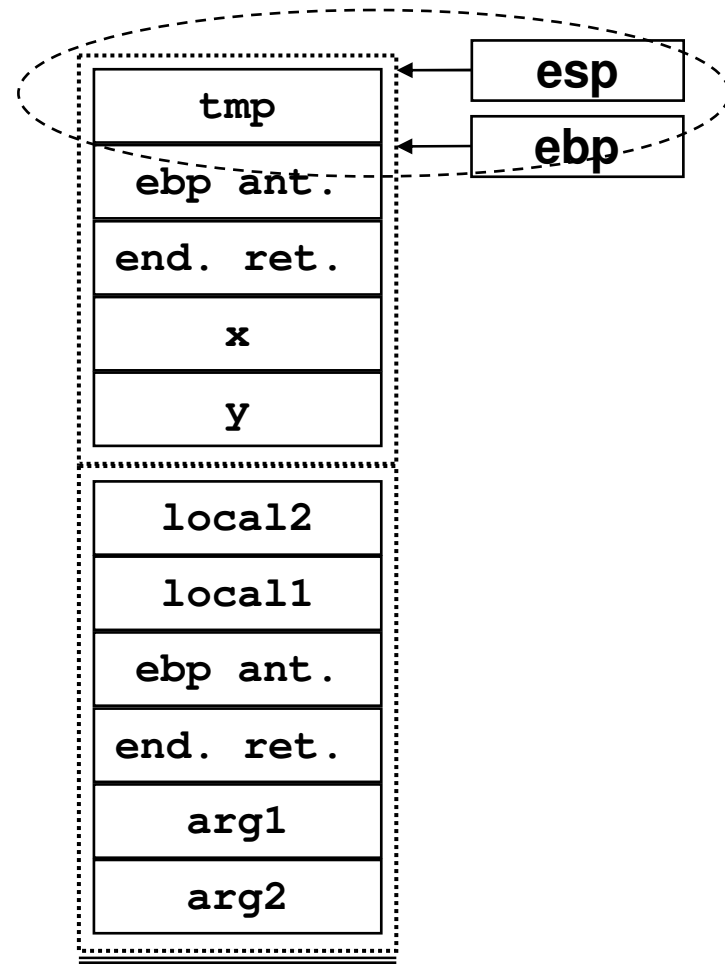
```
troca: push    %ebp
      mov     %esp, %ebp
      sub     $4, %esp
      mov     8(%ebp), %eax
      mov     (%eax), %edx
      mov     %edx, -4(%ebp)
      mov     12(%ebp), %ebx
      mov     (%ebx), %edx
      mov     %edx, (%eax)
      mov     -4(%ebp), %edx
      mov     %edx, (%ebx)
      mov     %ebp, %esp
      pop     %ebp
      ret
```



Controle do quadro da pilha de execução

Exemplo: Arquitetura Intel Pentium

```
troca: push    %ebp
       mov     %esp, %ebp
       sub     $4, %esp
       mov     8(%ebp), %eax
       mov     (%eax), %edx
       mov     %edx, -4(%ebp)
       mov     12(%ebp), %ebx
       mov     (%ebx), %edx
       mov     %edx, (%eax)
       mov     -4(%ebp), %edx
       mov     %edx, (%ebx)
       mov     %ebp, %esp
       pop     %ebp
       ret
```



Controle do quadro da pilha de execução

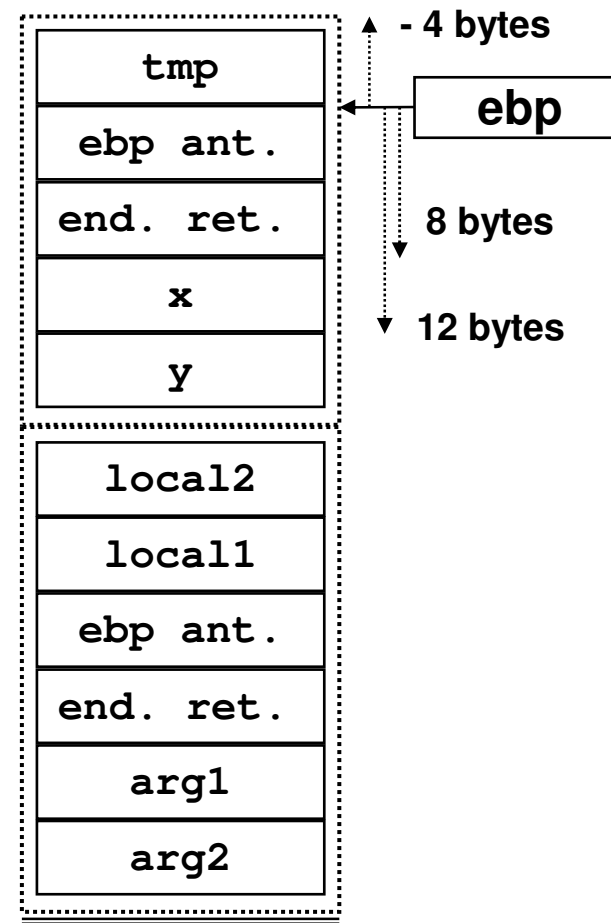
Exemplo: Arquitetura Intel Pentium

Endereços :

-4(%ebp) → variável local tmp

8(%ebp) → parâmetro x

12(%ebp) → parâmetro y



Exercício

Exercício

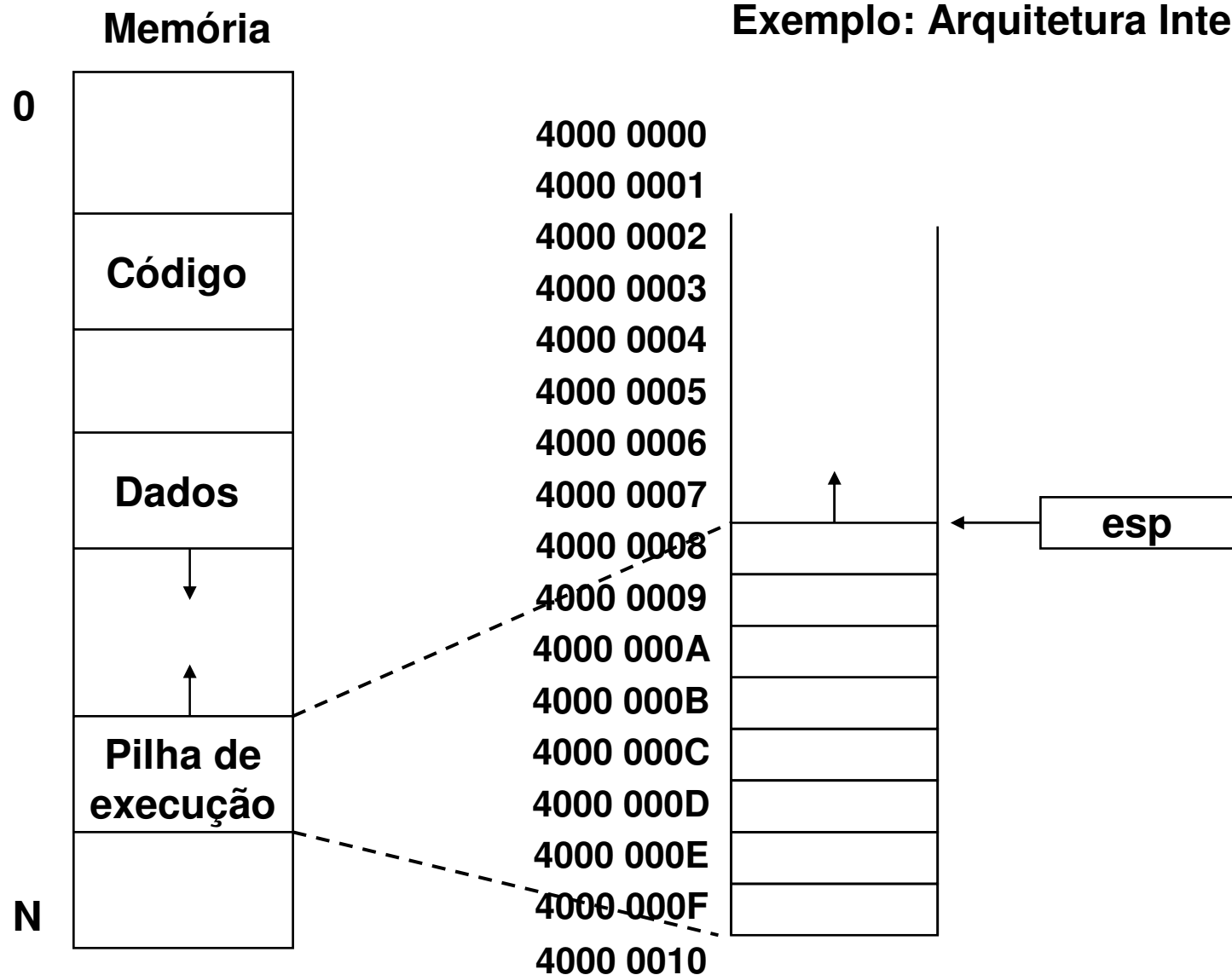
(3) Seja a configuração da pilha em um processador Intel Pentium (*little endian*) mostrada no próximo slide.

- ❖ **Suponha o valor do registrador `eax` = `A1B2C3D4` (hex)**
- ❖ **Qual é a configuração da pilha de execução após a execução da instrução a seguir?**

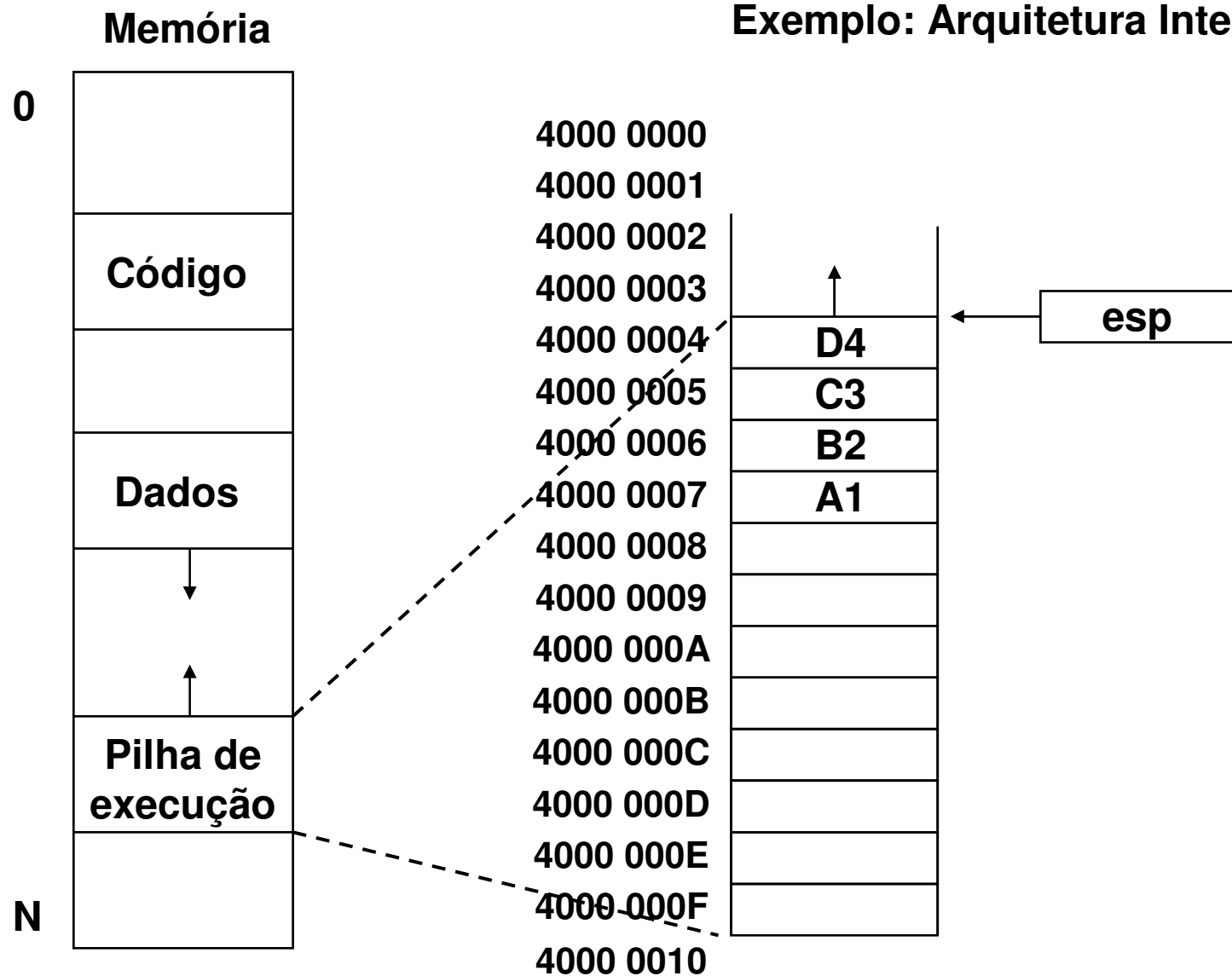
`pushl %eax`

(empilha os 4 bytes que representam o valor contido no registrador `eax`)

Exercício



Exercício



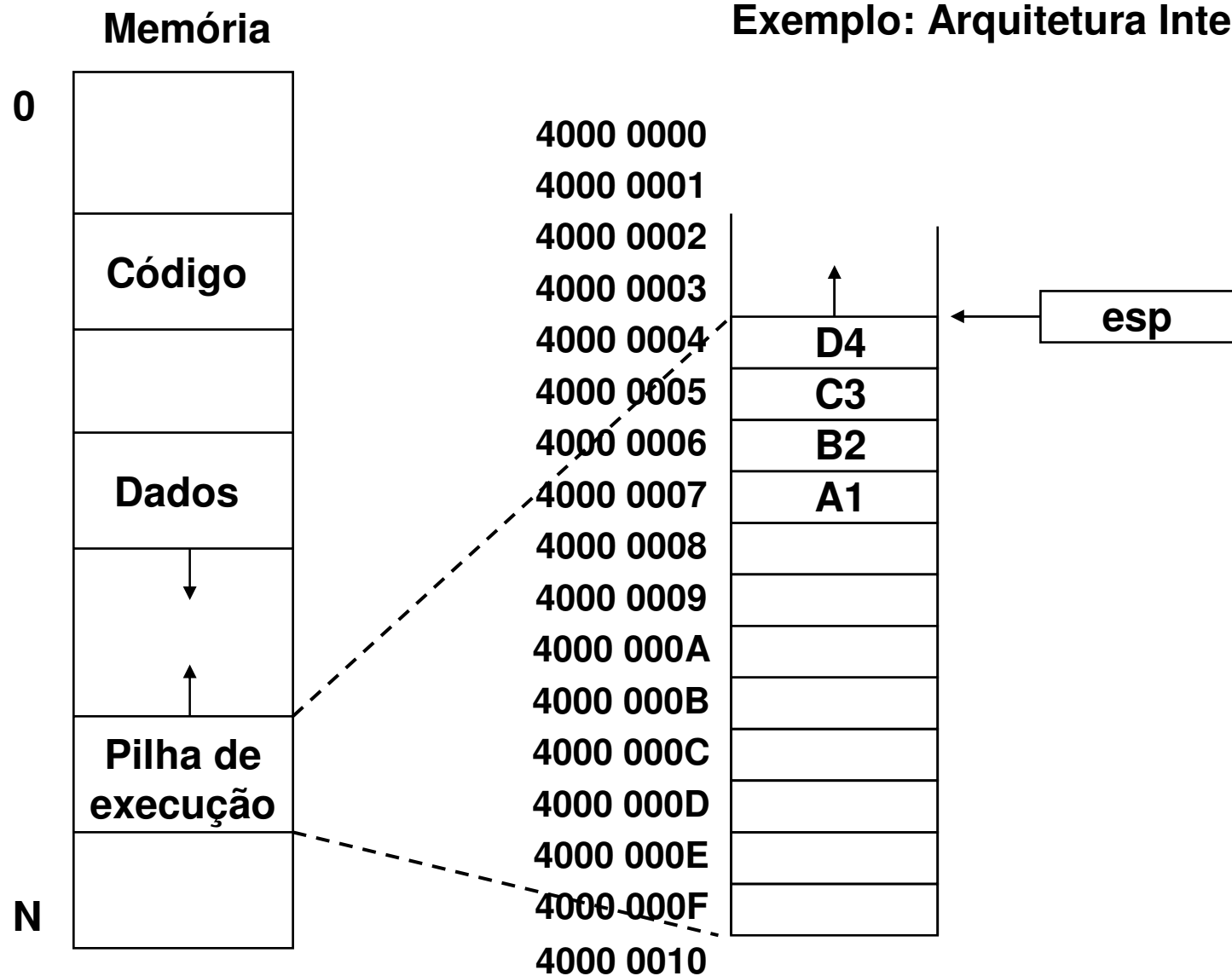
Exercício

(4) Em relação ao slide anterior, qual é a configuração da pilha de execução após a execução da seguinte instrução:

`popl %eax`

(desempilha 4 bytes do topo da pilha e armazena no registrador eax)

Exercício



Exercício

