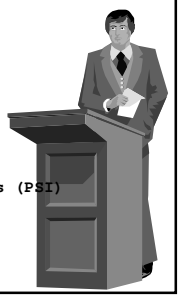
 1

Pilha de execução



Volnys Borges Bernal
volnys@lsi.usp.br

Departamento de Sistemas Eletrônicos (PSI)
Escola Politécnica da USP

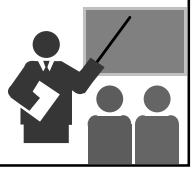
 2

Agenda

- Os problemas
- Pilha de execução
- Controle do endereço de retorno da função
- Passagem de argumentos da função
- Alocação de variáveis locais
- Controle do quadro da pilha de execução

 3

Os problemas

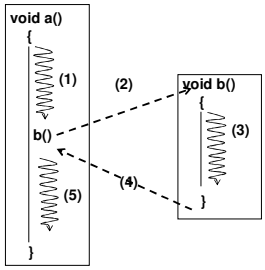


 4

Os problemas

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

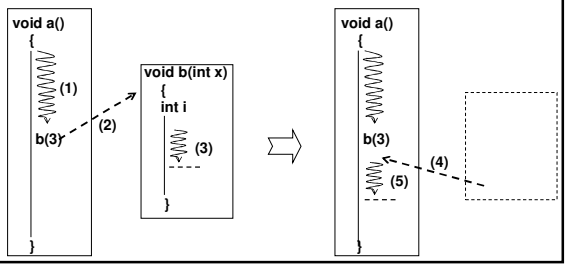


 5

Os problemas

2 – Variáveis locais

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

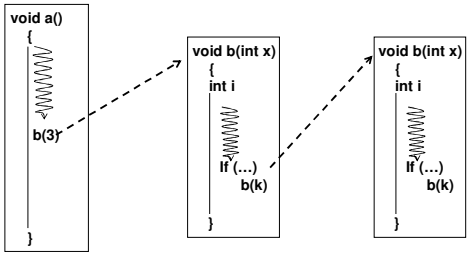


 6

Os problemas

3 – Instâncias independentes

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



 7

Pilha de execução

 8

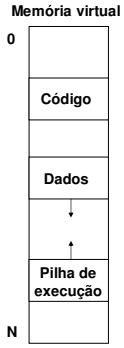
Pilha de execução

- **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**

 9

Pilha de execução

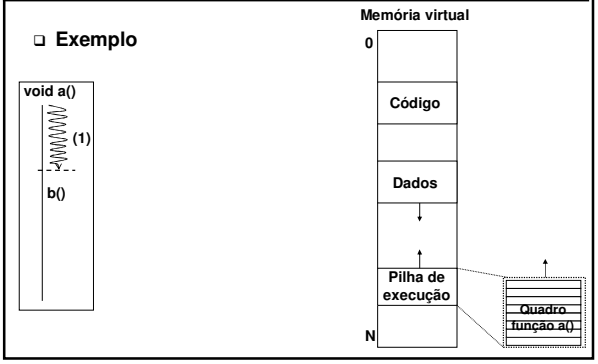
- **É reservado na memória virtual uma área exclusiva para a pilha de execução.**



 10

Pilha de execução

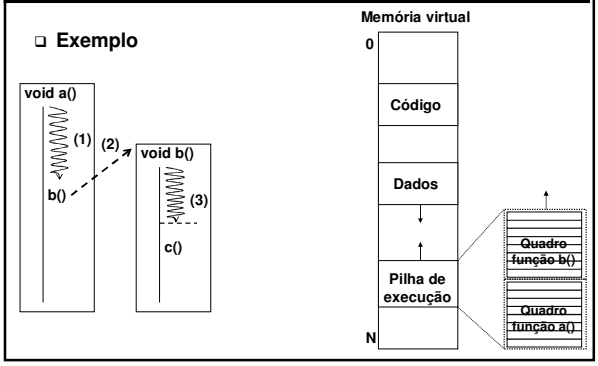
- **Exemplo**



 11

Pilha de execução

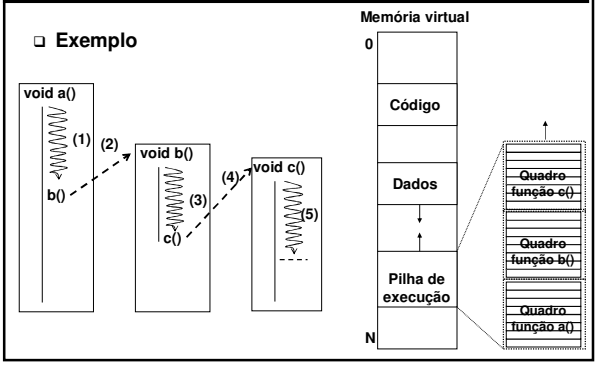
- **Exemplo**



 12

Pilha de execução

- **Exemplo**



 13

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

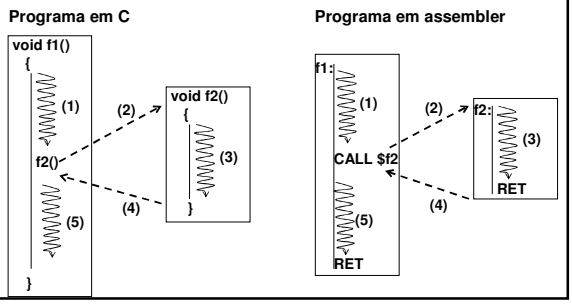
 14

Controle do endereço de retorno da função

 15

Controle do endereço de retorno

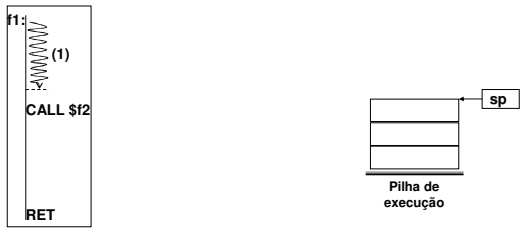
- **Realizado pelas instruções CALL e RET**



 16

Controle do endereço de retorno

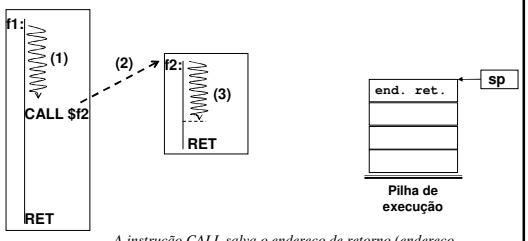
- **Exemplo de funcionamento das instruções CALL e RET**



 17

Controle do endereço de retorno

- **Exemplo de funcionamento das instruções CALL e RET**

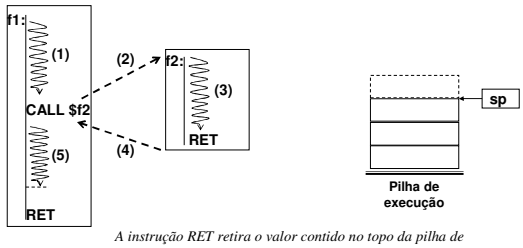


A instrução CALL salva o endereço de retorno (endereço da instrução após CALL) na pilha de execução e transfere o controle para f2.

 18

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.

 19

Passagem dos valores dos argumento da função

 20

Passagem de argumentos da função

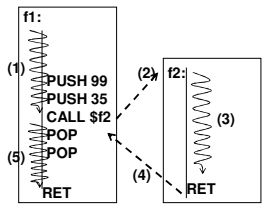
- A pilha de execução é utilizada, também, para passagem dos argumentos da função.
- Os valores dos argumentos são inseridos na pilha de execução antes da ativação da instrução CALL.

 21

Passagem de argumentos da função

- Exemplo

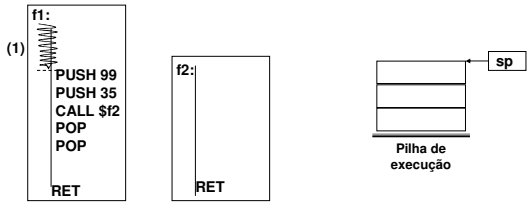
```
void f2(int a, int b)
     {
     ...
     }

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



 22

Passagem de argumentos da função

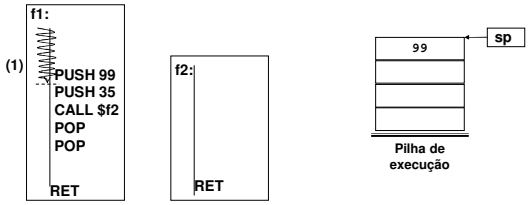
- Exemplo



 23

Passagem de argumentos da função

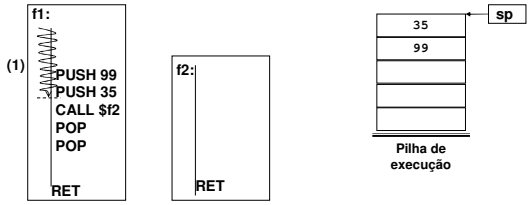
- Exemplo



 24

Passagem de argumentos da função

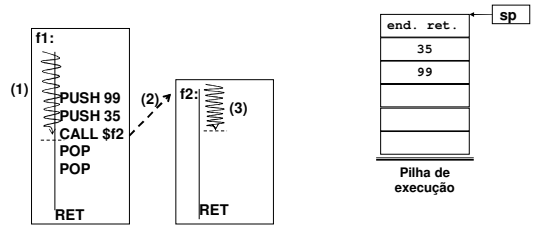
- Exemplo



 25

Passagem de argumentos da função

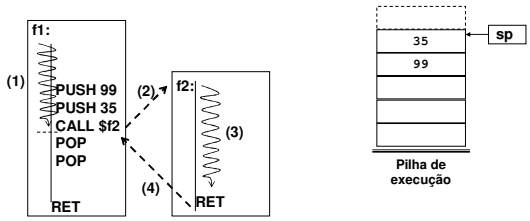
- Exemplo



 26

Passagem de argumentos da função

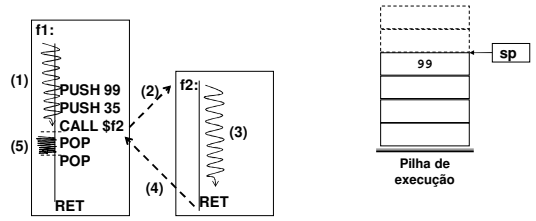
- Exemplo



 27

Passagem de argumentos da função

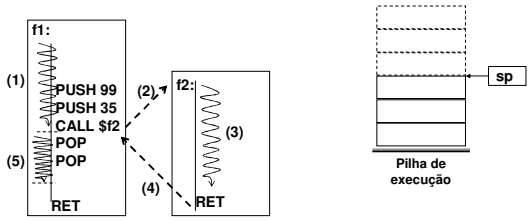
- Exemplo



 28

Passagem de argumentos da função

- Exemplo



 29

Alocação de variáveis locais

 30

Alocação de variáveis locais

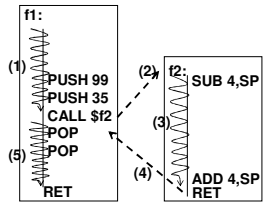
- Variáreis locais também são alocadas na pilha de execução.
- As variáveis locais de uma função ficam alocadas no quadro da respectiva função na pilha de execução

 31

Alocação de variáveis locais

- Exemplo

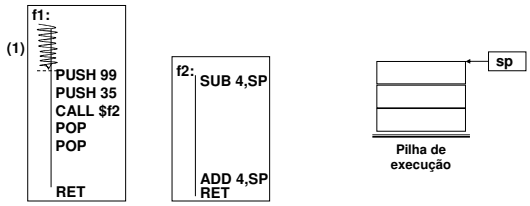
```
void f2(int a, int b)
     {
     int x;
     ...
     }

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



 32

Alocação de variáveis locais

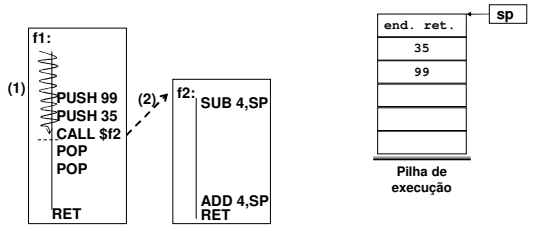
- Exemplo



 33

Alocação de variáveis locais

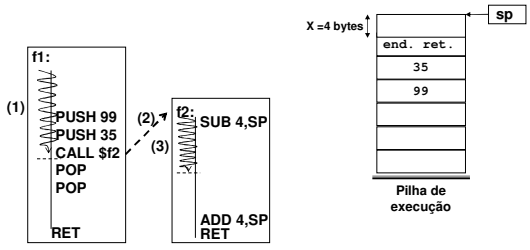
- Exemplo



 34

Alocação de variáveis locais

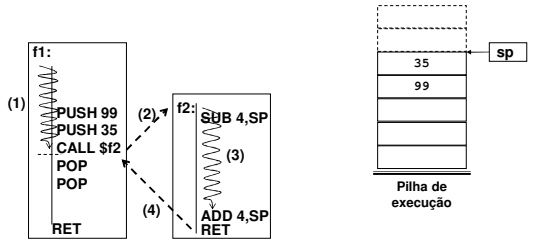
- Exemplo



 35

Alocação de variáveis locais

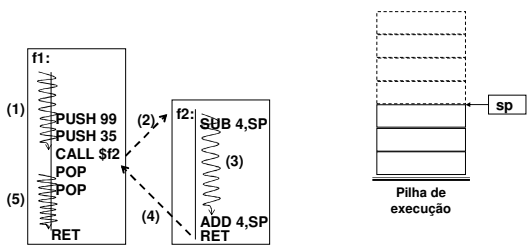
- Exemplo



 36

Alocação de variáveis locais

- Exemplo



 37

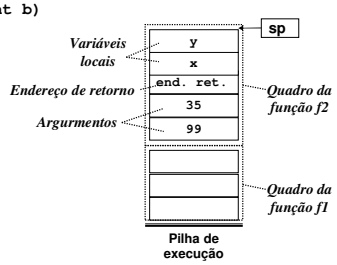
Controle do quadro da pilha de execução

 38

Controle do 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);
    ...
    }
```



 39

Exercício

 40

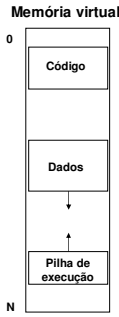
Exercício

(1) Mostre graficamente a evolução da área de dados e da pilha de execução decorrente da execução do programa a seguir:

```
int i;

int myfunction(int x)
  {
  x = x + 2;
  printf("%d\n",x);
  }

int main(int argc, char **argv)
  {
  i = i + 3;
  myfuncion(i);
  }
```

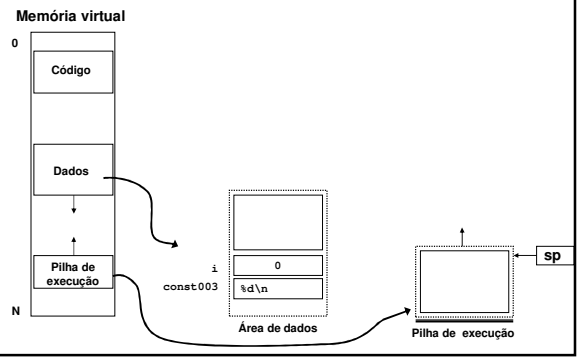
 41

Exercício



 42

Exercício



 43

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.

 44

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);
  }
```

 45

Controle do quadro da pilha de execução na arquitetura Intel Pentium

 46

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.

 47

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
      ```

 48

Controle do quadro da pilha de execução

Exemplo: Arquitetura Intel Pentium

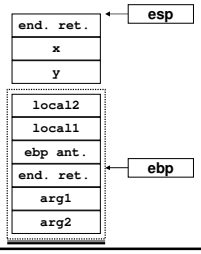
```
void troca (int *x, int *y)
    {
    int tmp;

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

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
```

 49

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
```

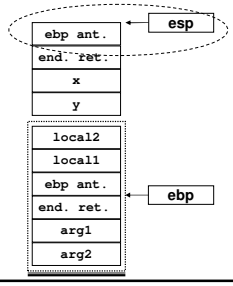


 50

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
```

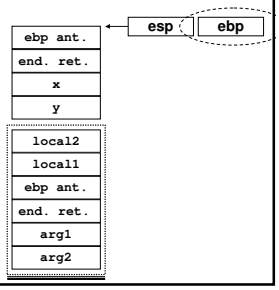


 51

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
```

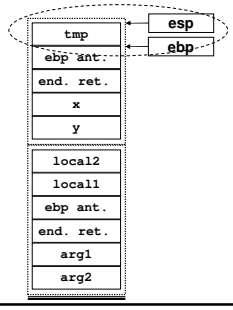


 52

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
```

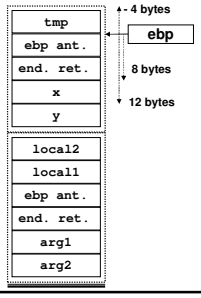


 53

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
```



 54

Exercício

 55

Exercício

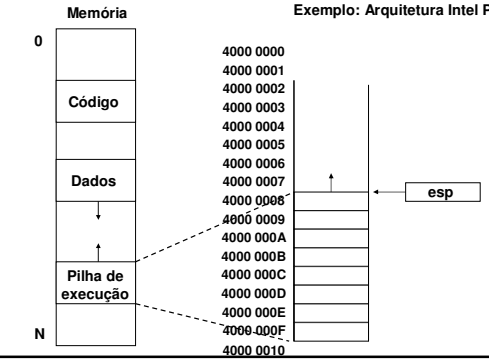
(3) Seja a configuração da pilha em um processador Intel Pentium (*little endian*) mostrada no próximo slide.

- **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)**

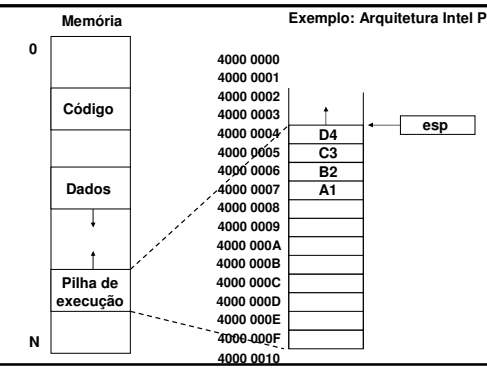
 56

Exercício



 57

Exercício



 58

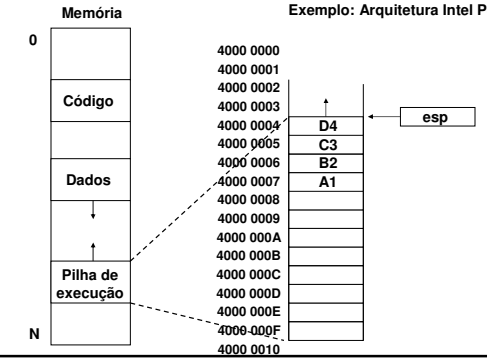
Exercício

(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)**

 59

Exercício



 60

Exercício