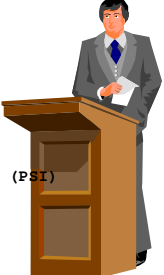


© 2005-2015 Volnys Bernal 1

## Pilha de execução

Volnys Borges Bernal  
volnys@lsi.usp.br

Departamento de Sistemas Eletrônicos (PSE)  
Escola Politécnica da USP




© 2005-2015 Volnys Bernal 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

© 2005-2015 Volnys Bernal 3

## Os problemas



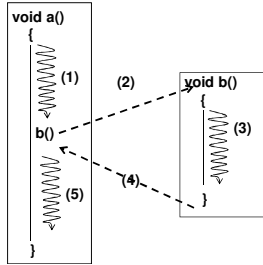
© 2005-2015 Volnys Bernal 4

## 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 término da subrotina, a próxima instrução a ser executada deve ser a instrução posterior à instrução que ativou a subrotina.

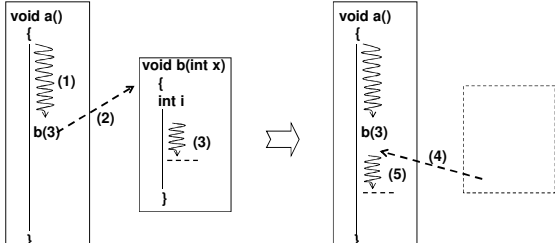


© 2005-2015 Volnys Bernal 5

## Os problemas

### 2 – Variáveis locais

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

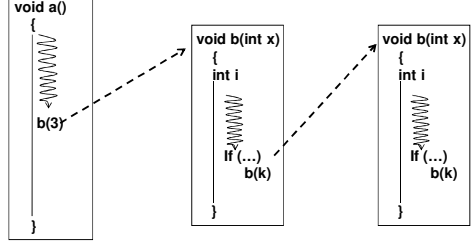


© 2005-2015 Volnys Bernal 6

## Os problemas

### 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:

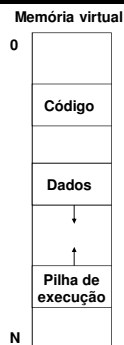
1. Controle do endereço de retorno da função;
2. Passagem dos valores dos argumento da função;
3. Alocação de variáveis locais à função.

### 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

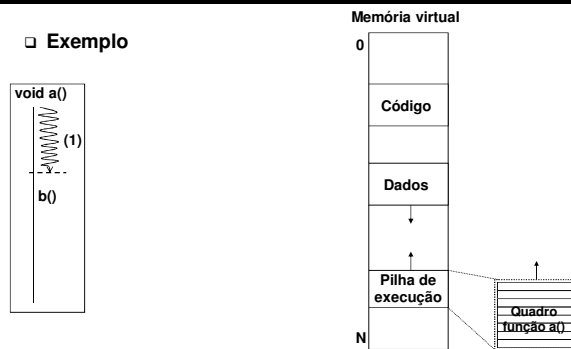
## Pilha de execução

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



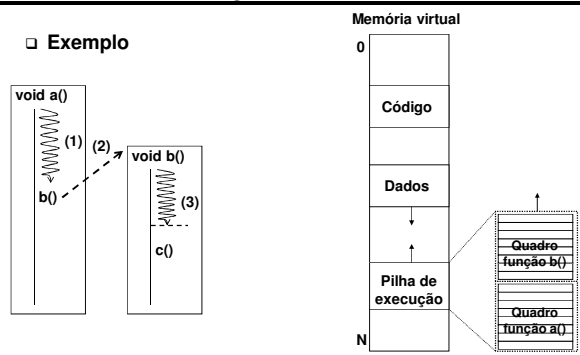
## Pilha de execução

### Exemplo



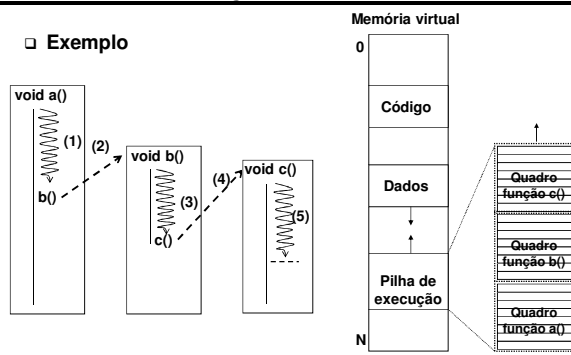
## Pilha de execução

### Exemplo



## Pilha de execução

### Exemplo



© 2005-2015 Volnys Bernal 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

© 2005-2015 Volnys Bernal 14

## Controle do endereço de retorno da função

© 2005-2015 Volnys Bernal 15

## Controle do endereço de retorno

- ❑ Realizado pelas instruções CALL e RET

Programa em C

```
void f1()
{
  // (1)
  f2();
  // (5)
}
```

Programa em assembler

```
f1:
  // (1)
  CALL $f2
  // (5)
  RET

f2:
  // (3)
  RET
```

© 2005-2015 Volnys Bernal 16

## Controle do endereço de retorno

- ❑ Exemplo de funcionamento das instruções CALL e RET

© 2005-2015 Volnys Bernal 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.

© 2005-2015 Volnys Bernal 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.

## Passagem dos valores dos argumento da função

## Passagem de argumentos da função

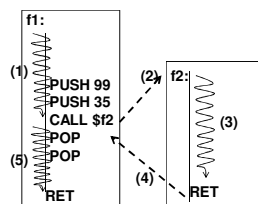
- ❑ A pilha de execução é utilizada, também, para passagem dos argumentos da função.
- ❑ Os valores dos argumentos são inseridos na pilha de execução antes da ativação da instrução CALL.

## Passagem de argumentos da função

### Exemplo

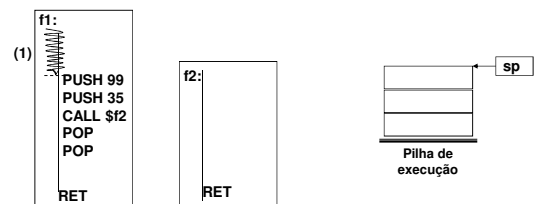
```
void f2(int a, int b)
{
    ...
}

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



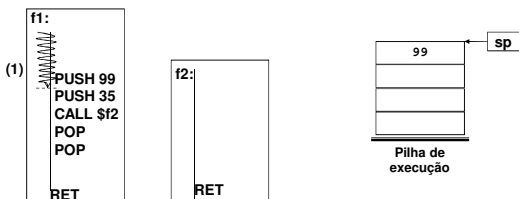
## Passagem de argumentos da função

### Exemplo



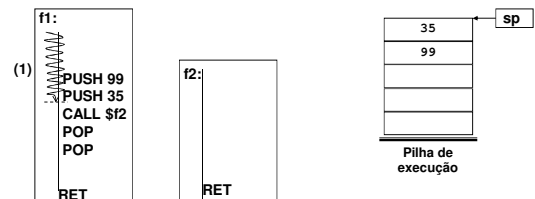
## Passagem de argumentos da função

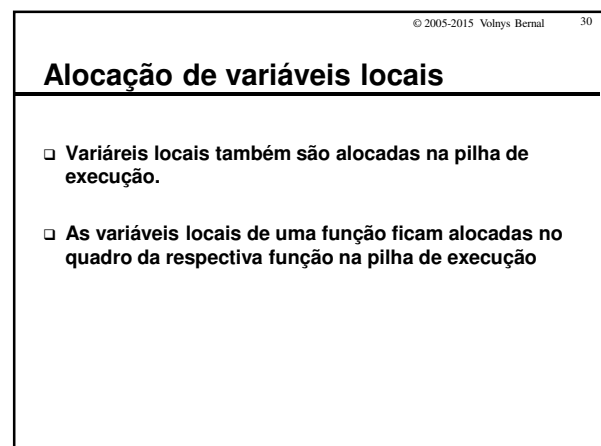
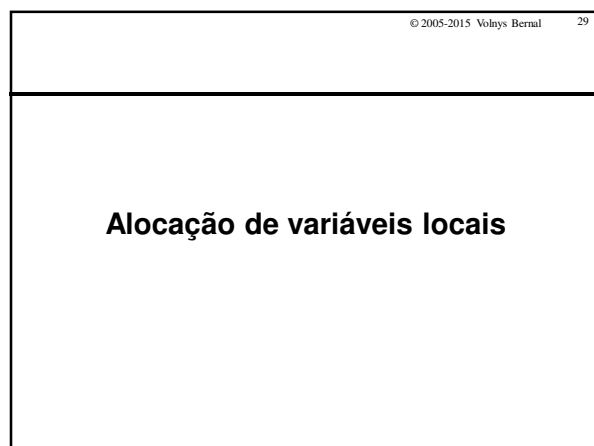
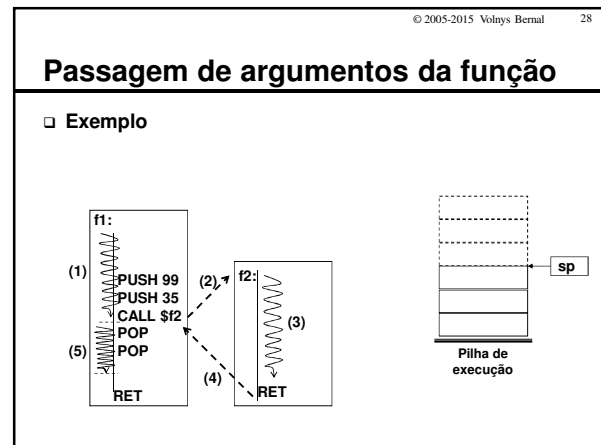
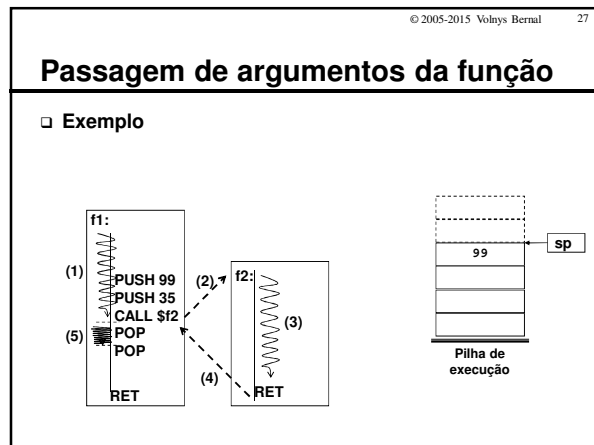
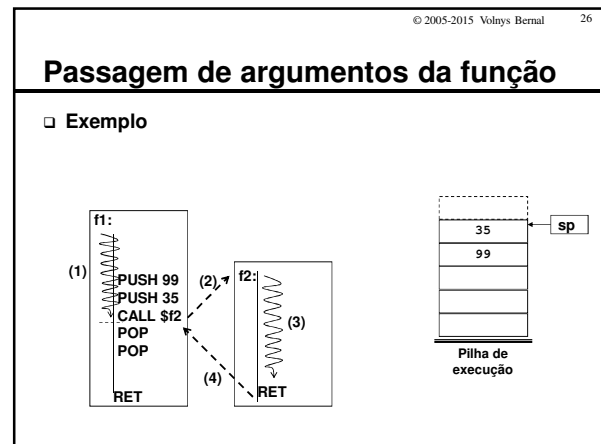
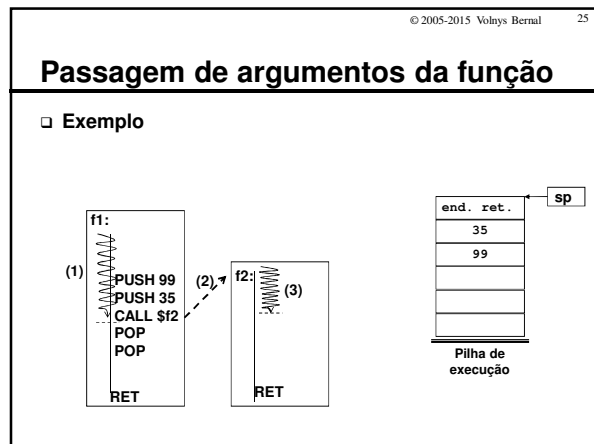
### Exemplo



## Passagem de argumentos da função

### Exemplo





© 2005-2015 Volnys Bernal 31

### Alocação de variáveis locais

❑ Exemplo

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

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

© 2005-2015 Volnys Bernal 32

### Alocação de variáveis locais

❑ Exemplo

© 2005-2015 Volnys Bernal 33

### Alocação de variáveis locais

❑ Exemplo

© 2005-2015 Volnys Bernal 34

### Alocação de variáveis locais

❑ Exemplo

© 2005-2015 Volnys Bernal 35

### Alocação de variáveis locais

❑ Exemplo

© 2005-2015 Volnys Bernal 36

### Alocação de variáveis locais

❑ Exemplo

© 2005-2015 Volnys Bernal37

### Controle do quadro da pilha de execução

© 2005-2015 Volnys Bernal38

### 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);
    ...
}
```

© 2005-2015 Volnys Bernal39

### Exercício

© 2005-2015 Volnys Bernal40

### 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;
    myfunction(i);
}
```

© 2005-2015 Volnys Bernal41

### Exercício

Memória virtual

© 2005-2015 Volnys Bernal42

### 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
```



© 2005-2015 Volnys Bernal49

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)  
pop %ebp  
ret

esp ← end. ret.

x

y

local2

local1

ebp ant.

end. ret.

arg1

arg2

ebp ←

© 2005-2015 Volnys Bernal50

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)  
pop %ebp  
ret

esp ← ebp ant.

end. ret.

x

y

local2

local1

ebp ant.

end. ret.

arg1

arg2

ebp ←

© 2005-2015 Volnys Bernal51

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

esp ← ebp

ebp ant.

end. ret.

x

y

local2

local1

ebp ant.

end. ret.

arg1

arg2

© 2005-2015 Volnys Bernal52

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

esp ← tmp

ebp ant.

end. ret.

x

y

local2

local1

ebp ant.

end. ret.

arg1

arg2

© 2005-2015 Volnys Bernal53

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

tmp

ebp ant.

end. ret.

x

y

local2

local1

ebp ant.

end. ret.

arg1

arg2

← 4 bytes

← 8 bytes

← 12 bytes

© 2005-2015 Volnys Bernal54

Exercício

© 2005-2015 Volnys Bernal55

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`)

© 2005-2015 Volnys Bernal56

Exercício

Memória

Exemplo: Arquitetura Intel Pentium

0

Código

Dados

Pilha de execução

N

4000 0000

4000 0001

4000 0002

4000 0003

4000 0004

4000 0005

4000 0006

4000 0007

4000 0008

4000 0009

4000 000A

4000 000B

4000 000C

4000 000D

4000 000E

4000 000F

4000 0010

↑

esp

© 2005-2015 Volnys Bernal57

Exercício

Memória

Exemplo: Arquitetura Intel Pentium

0

Código

Dados

Pilha de execução

N

4000 0000

4000 0001

4000 0002

4000 0003

4000 0004

4000 0005

4000 0006

4000 0007

4000 0008

4000 0009

4000 000A

4000 000B

4000 000C

4000 000D

4000 000E

4000 000F

4000 0010

↑

D4

C3

B2

A1

esp

© 2005-2015 Volnys Bernal58

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`)

© 2005-2015 Volnys Bernal59

Exercício

Memória

Exemplo: Arquitetura Intel Pentium

0

Código

Dados

Pilha de execução

N

4000 0000

4000 0001

4000 0002

4000 0003

4000 0004

4000 0005

4000 0006

4000 0007

4000 0008

4000 0009

4000 000A

4000 000B

4000 000C

4000 000D

4000 000E

4000 000F

4000 0010

↑

D4

C3

B2

A1

esp

© 2005-2015 Volnys Bernal60

Exercício

Memória

Exemplo: Arquitetura Intel Pentium

0

Código

Dados

Pilha de execução

N

4000 0000

4000 0001

4000 0002

4000 0003

4000 0004

4000 0005

4000 0006

4000 0007

4000 0008

4000 0009

4000 000A

4000 000B

4000 000C

4000 000D

4000 000E

4000 000F

4000 0010

↑

esp