

# INE5607 – Organização e Arquitetura de Computadores

Linguagem de Montagem e de Máquina

## Aula 12: Suporte à chamada de procedimentos

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br



# Sumário

- Procedimentos
- A pilha
- Procedimento folha compilado
- Procedimentos aninhados
- Considerações finais

# PROCEDIMENTOS

# Procedimentos

- Procedimentos
  - Sinônimos: rotinas, sub-rotinas, funções, métodos...
  - Efetuam uma tarefa específica
    - Baseada em parâmetros passados
  - Exemplo:

```
int soma_dois(int a, int b){  
    int c = 0;  
    c = a + b;  
    return c;  
}
```

# Procedimentos

- Vantagens
  - Estruturação do programa
  - Reutilização de código
  - Isolamento do resto do programa
    - Entrada: **argumentos** ou parâmetros
    - Saída: **valores** retornados

# Procedimentos

- Etapas de uma chamada de procedimento
  1. Colocar os argumentos num lugar onde o procedimento possa acessá-los
  2. Transferir o controle para o procedimento
  3. Adquirir os recursos de armazenamento necessários para executar o procedimento
  - 4. Efetuar a tarefa desejada**
  5. Colocar o resultado num lugar onde o programa chamador possa acessá-lo
  6. Retornar o controle ao ponto de origem

# Procedimentos

- Suporte para procedimentos
  - Registradores
    - **\$a0-\$a3**: para armazenar **argumentos**
    - **\$v0-\$v1**: para armazenar **valores de retorno**
    - **\$ra**: para armazenar o **endereço de retorno**  
(*return address*)

# Procedimentos

- Suporte para procedimentos
  - Instruções
    - **jal EndereçoDoProcedimento**
      - Salva em **\$ra** o endereço da próxima instrução ( $\$ra = PC+4$ )
      - Desvia para o endereço ( $PC = \text{Endereço}$ )
      - **Programa armazenado!**
    - **jr \$ra**
      - Retorna à rotina chamadora ( $PC = \$ra$ )



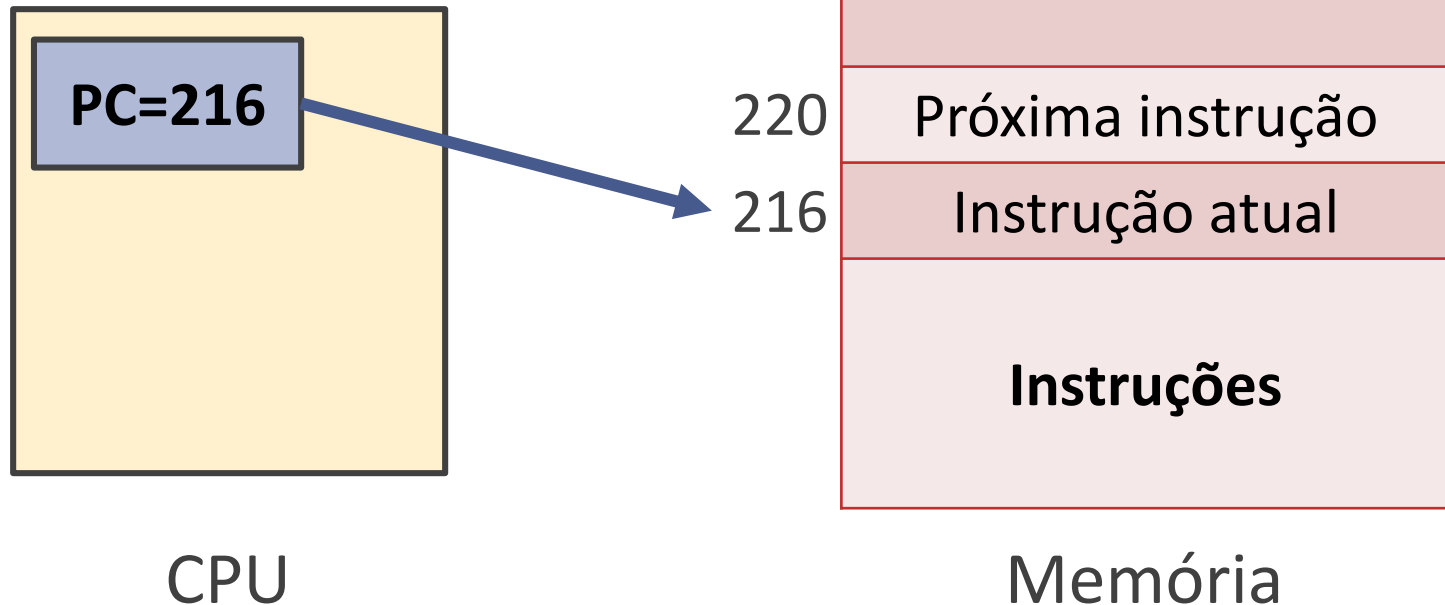
# Procedimentos

- **PC: contador de programa**

- *Program Counter*

- Registrador

- Endereço de instrução



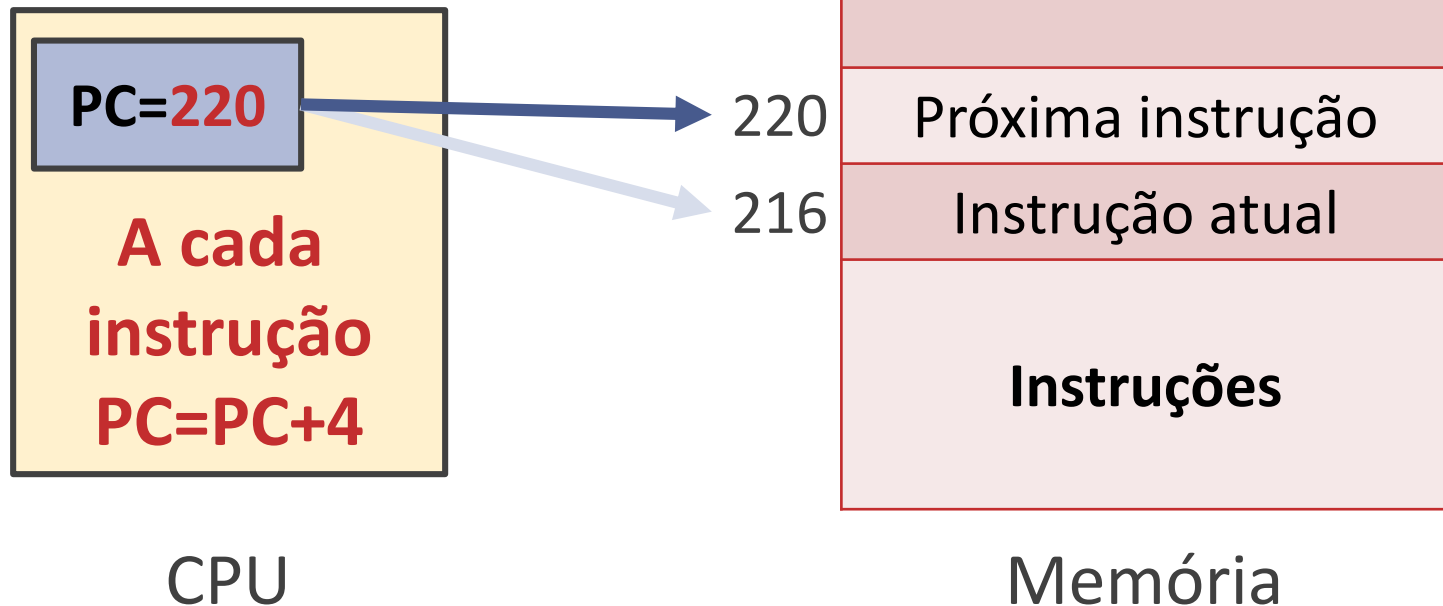
# Procedimentos

- **PC: contador de programa**

- *Program Counter*

- Registrador

- Endereço de instrução



# Procedimentos

- Antes de invocar o procedimento
  - **Escrever na memória** o conteúdo dos registradores a serem utilizados pelo procedimento chamado
    - **Salvamento do contexto** do procedimento chamador
- Depois de executar o procedimento
  - **Restaurar os valores originais** nos registradores
    - **Restauração do contexto** do procedimento chamador

# A PILHA

# A pilha

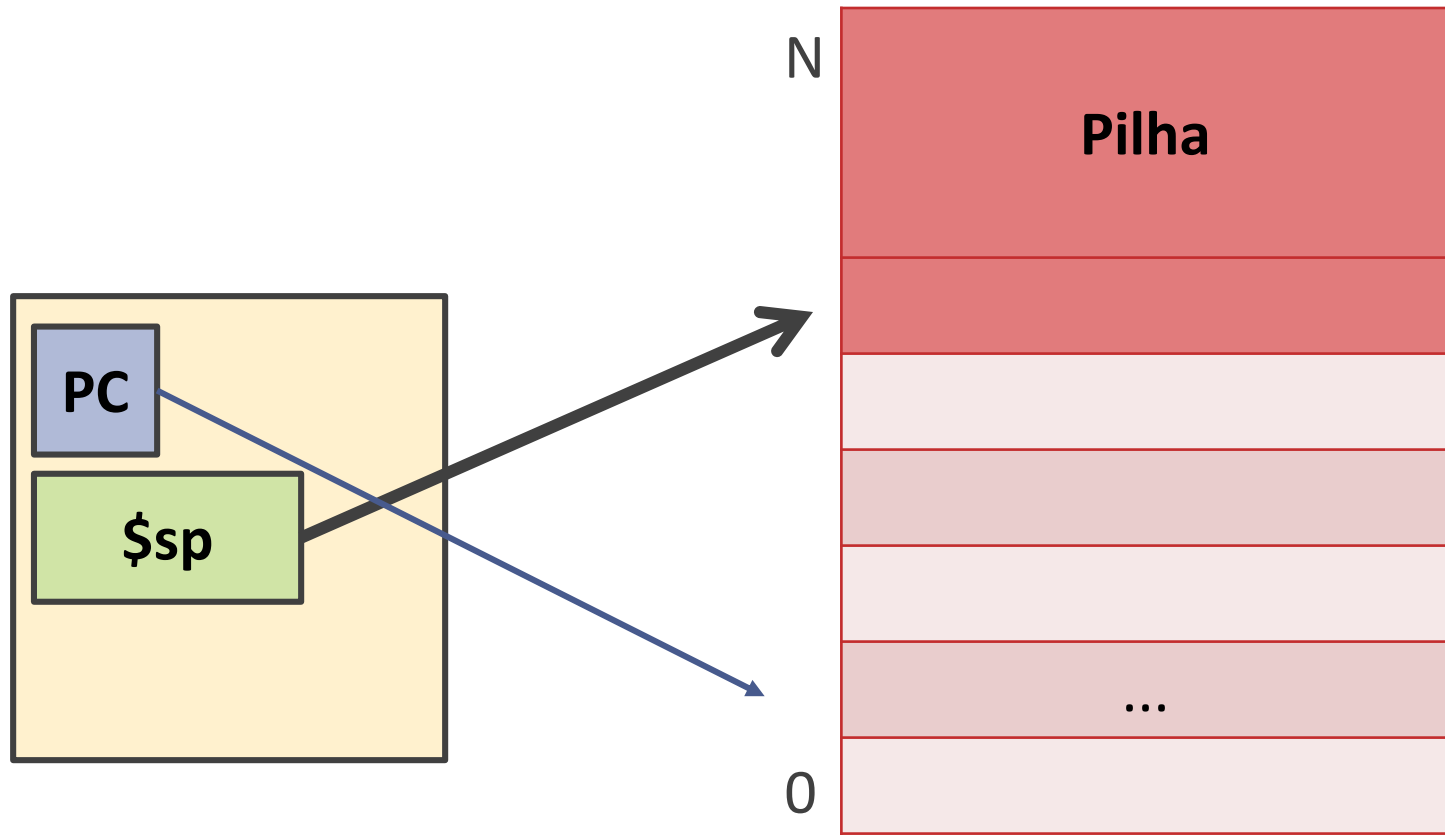
- **Pilha**
  - Estrutura de dados simples
    - Em memória
  - LIFO: *Last in, first out*
- Ponteiro para o topo da pilha
  - **\$sp: stack pointer** (ponteiro da pilha)
- Operações
  - *Push*: insere na pilha
  - *Pop*: remove da pilha

# A pilha

- Organização da pilha em memória
  - Cresce de cima para baixo
    - **Maior endereço para menor endereço**
    - *Push*: decrementa registrador
    - *Pop*: incrementa registrador

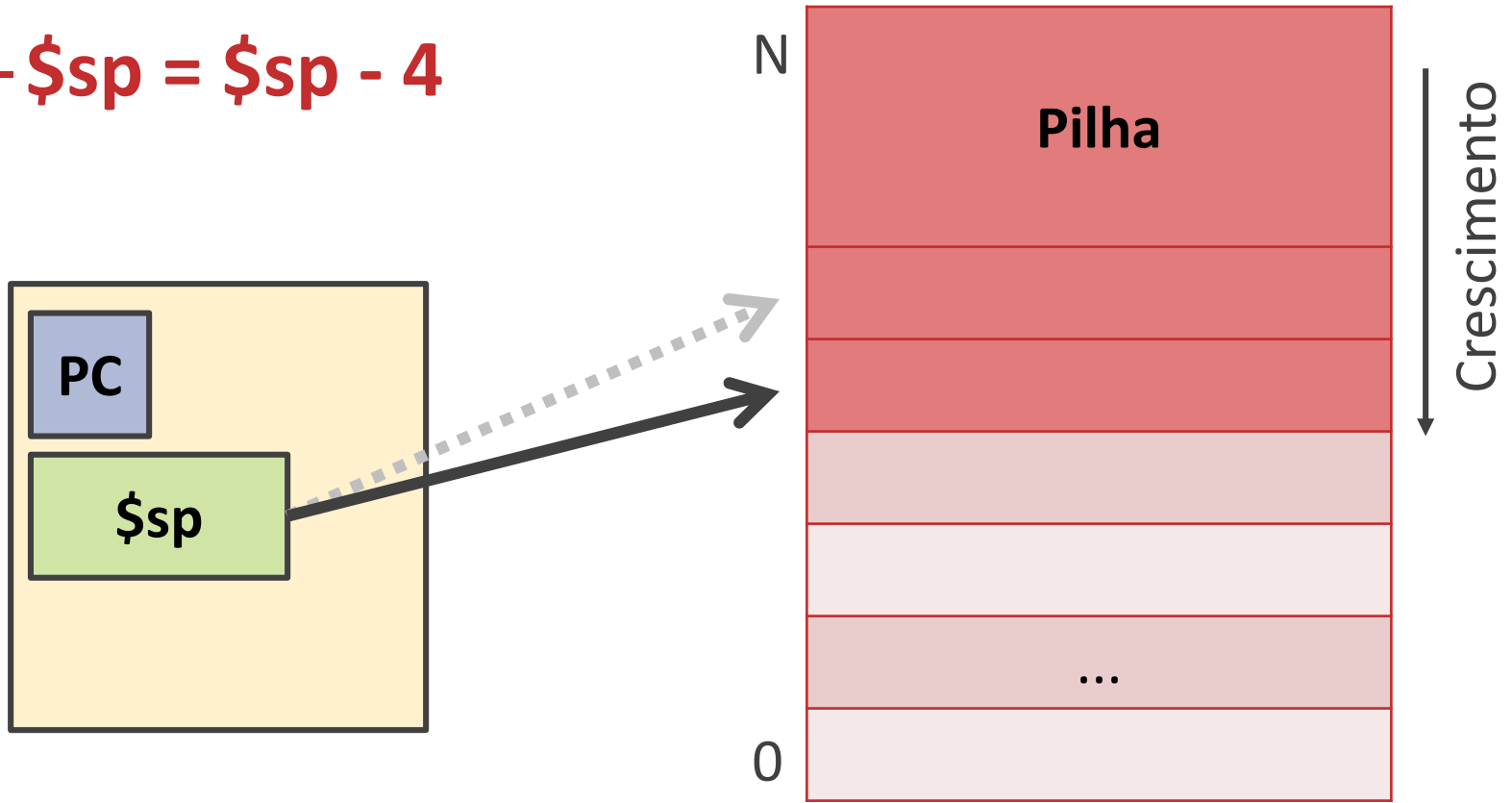
# A pilha

- Organização da pilha em memória



# A pilha

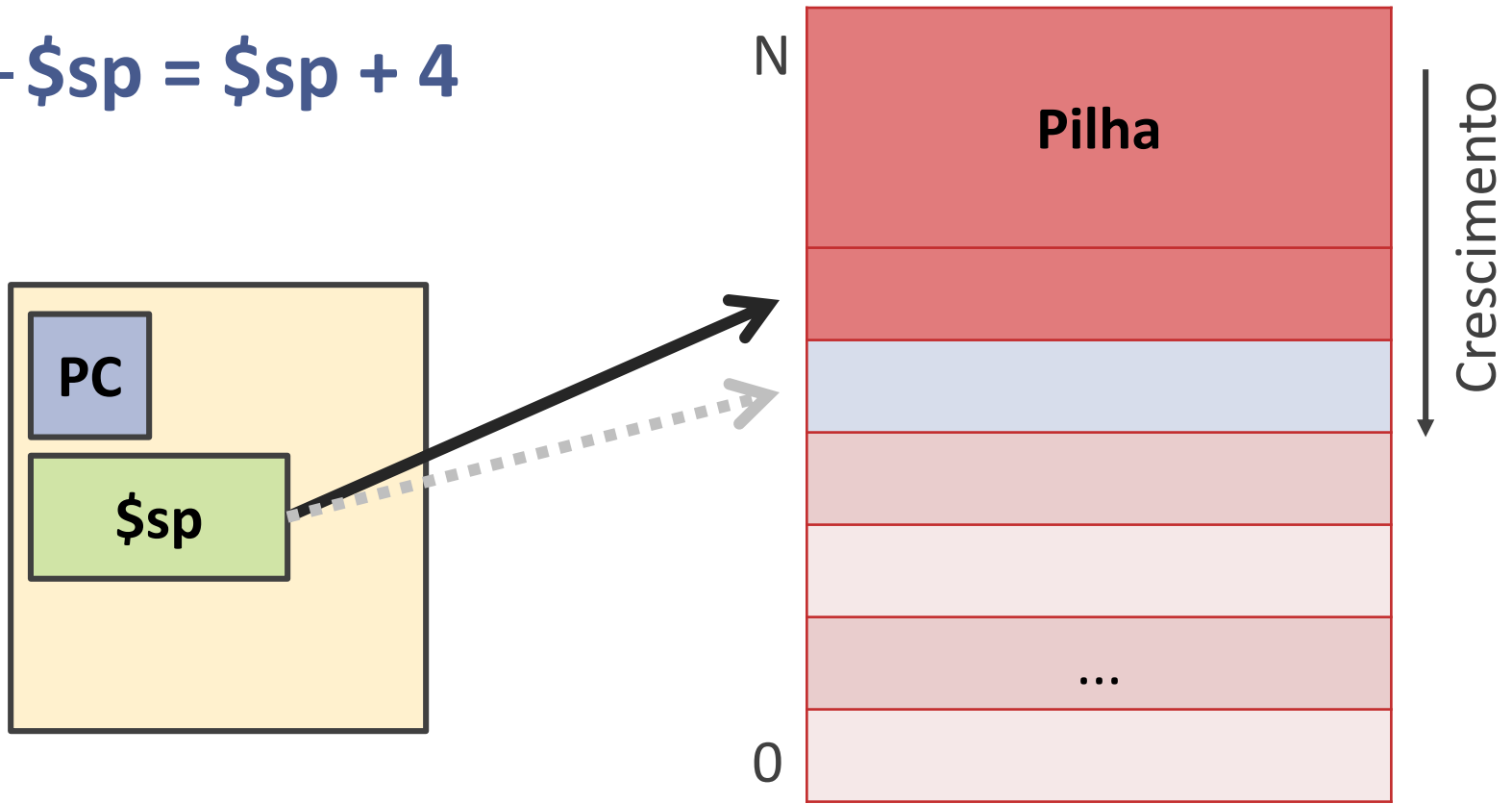
- Organização da pilha em memória
  - Push: decrementa  $\$sp$
  - $\$sp = \$sp - 4$





# A pilha

- Organização da pilha em memória
  - Pop: incrementa  $\$sp$
  - $\$sp = \$sp + 4$



# A pilha

- Exemplo armazenamento

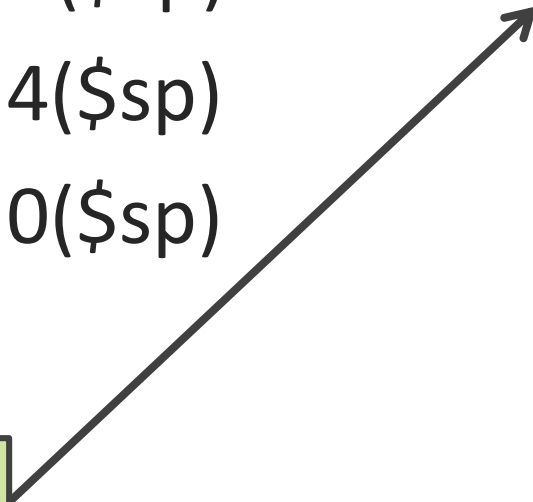
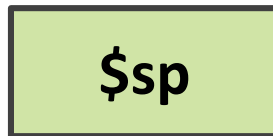
- addi \$sp, \$sp, -16

- sw **\$s0**, 12(\$sp)

- sw **\$s1**, 8(\$sp)

- sw **\$t4**, 4(\$sp)

- sw **\$t5**, 0(\$sp)



# A pilha

- Exemplo armazenamento

- addi \$sp, \$sp, -16

- sw **\$s0**, 12(\$sp)

- sw **\$s1**, 8(\$sp)

- sw **\$t4**, 4(\$sp)

- sw **\$t5**, 0(\$sp)

\$sp



# A pilha

- Exemplo armazenamento

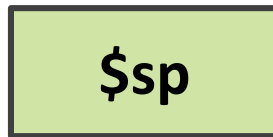
- addi \$sp, \$sp, -16

- sw **\$s0**, 12(\$sp)

- sw **\$s1**, 8(\$sp)

- sw **\$t4**, 4(\$sp)

- sw **\$t5**, 0(\$sp)



# A pilha

- Exemplo recuperação

- lw **\$t5**, 0(\$sp)
- lw **\$t4**, 4(\$sp)
- lw **\$s1**, 8(\$sp)
- lw **\$s0**, 12(\$sp)
- addi \$sp, \$sp, 16

\$sp



# A pilha

- Exemplo recuperação

- lw **\$t5**, 0(\$sp)
- lw **\$t4**, 4(\$sp)
- lw **\$s1**, 8(\$sp)
- lw **\$s0**, 12(\$sp)
- addi \$sp, \$sp, 16

\$sp

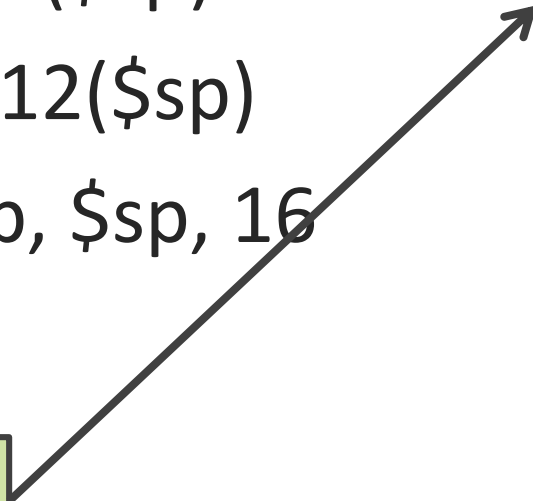


# A pilha

- Exemplo recuperação

- lw **\$t5**, 0(\$sp)
- lw **\$t4**, 4(\$sp)
- lw **\$s1**, 8(\$sp)
- lw **\$s0**, 12(\$sp)
- addi \$sp, \$sp, 16

**\$sp**



# PROCEDIMENTO FOLHA COMPILADO



# Procedimento folha compilado

- Exemplo de procedimento folha
  - **Folha** = não chama outros procedimentos

```
int exemplo_folha(int g, int h, int i, int j){  
    int f;  
    f = (g + h) - (i + j);  
    return f;  
}
```

# Procedimento folha compilado

```
int exemplo_folha(int g, int h, int i, int j){  
    int f;  
    f = (g + h) - (i + j);  
    return f;  
}
```

(g,h,i,j) -> (\$a0,\$a1,\$a2,\$a3)

f -> \$s0

# Procedimento folha compilado

- **Corpo do procedimento**

—  $f = (g + h) - (i + j)$

add \$t0, \$a0, \$a1

# \$t0 = g + h

add \$t1, \$a2, \$a3

# \$t1 = i + j

sub \$s0, \$t0, \$t1

# f = \$t0 - \$t1

# Procedimento folha compilado

- Salvando o contexto do procedimento chamador

```
addi $sp, $sp, -12    # amplia pilha p/ 3 itens
```

```
sw $t1, 8($sp)        # salva registrador $t1
```

```
sw $t0, 4($sp)        # salva registrador $t0
```

```
sw $s0, 0($sp)        # salva registrador $s0
```

- Copiando o valor de retorno (f)

```
add $v0, $s0, $zero    #retorna f
```

# Procedimento folha compilado

- Restaurando os registradores salvos

lw **\$s0**, 0(\$sp)      # restaura \$s0

lw **\$t0**, 4(\$sp)      # restaura \$t0

lw **\$t1**, 8(\$sp)      # restaura \$t1

addi \$sp, \$sp, 12    # “remove” 3 itens

- Retornando ao procedimento chamador  
jr \$ra

# Procedimento folha compilado

- exemplo\_folha:

```
addi $sp, $sp, -12
sw  $t1, 8($sp)
sw  $t0, 4($sp)
sw  $s0, 0($sp)
add $t0, $a0, $a1
add $t1, $a2, $a3
sub $s0, $t0, $t1
add $v0, $s0, $zero
lw  $s0, 0($sp)
lw  $t0, 4($sp)
lw  $t1, 8($sp)
addi $sp, $sp, 12
jr  $ra
```


## Rotina chamadora:


Escreve em \$aX  
jal exemplo\_folha  
Lê de \$v0

# Procedimento folha compilado

- exemplo\_folha:

addi \$sp, \$sp, -12

sw \$t1, 8(\$sp) 

sw \$t0, 4(\$sp) 

sw \$s0, 0(\$sp)


add \$t0, \$a0, \$a1


add \$t1, \$a2, \$a3

sub \$s0, \$t0, \$t1

add \$v0, \$s0, \$zero

lw \$s0, 0(\$sp)

lw \$t0, 4(\$sp) 

lw \$t1, 8(\$sp) 

addi \$sp, \$sp, 12

jr \$ra

Rotina chamadora:

Escreve em \$aX  
jal exemplo\_folha  
Lê de \$v0

**\$t0-\$t9: não são preservados**  
**\$s0-\$s7: preservados**

# PROCEDIMENTOS ANINHADOS



# Procedimentos aninhados

- Procedimento folha
  - Não chamam outros procedimentos
- Procedimentos aninhados
  - Exemplo: “**main**” chama “**A**” que chama “**B**”
  - “**main**” chama “**A**(3)”
    - \$a0 = 3; jal **A**
  - “**A**” chama “**B**(7)”
    - \$a0 = 7; jal **B**
    - Argumento de “**A**” modificado por “**B**”
    - Endereço de retorno a “**main**” é perdido

# Procedimentos aninhados

- Solução: preservar registradores na pilha
  - Rotina **chamadora** salva na pilha
    - Registradores de argumento (**\$a0-\$a3**)
    - Registradores temporários (**\$t0-\$t9**)
  - Rotina **chamada** salva na pilha
    - Registradores usados em seu corpo (**\$s0-\$s7**)
    - Registrador de endereço de retorno (**\$ra**)
  - No **retorno**
    - **Registradores são restaurados**
    - **\$sp é reajustado** apropriadamente

# Procedimentos aninhados

- Exemplo: fatorial com recursão

```
int fact (int n) {  
    if (n < 1) return (1);  
    else return (n * fact(n-1));  
}
```

# Procedimentos aninhados

- Exemplo: fatorial com recursão
  - Considerando  $n$  em  $\$a0$
  - Armazenar  $\$a0$  e  $\$ra$
  - Testar  $n < 1$ 
    - Sim: retorna 1
    - Não: chama a si própria (fact)
  - Restaura o contexto do procedimento chamador
  - Retorna o valor do produto

# Procedimentos aninhados

- Armazenar \$a0 e \$ra
  - addi \$sp, \$sp, -8
  - sw \$ra, 4(\$sp)
  - sw \$a0, 0(\$sp)

# Procedimentos aninhados

- Testar  $n < 1$ 
  - `slti $t0, $a0, 1`                      #  $t0 = 1$  se  $n < 1$
  - `beq $t0, $zero, Maior`    # se  $n \geq 1$ , desvia

# Procedimentos aninhados

- Sim: retorna 1
  - addi \$v0, \$zero, 1
  - addi \$sp, \$sp, 8
  - jr \$ra
  - O que acontece com o que estava na pilha?

# Procedimentos aninhados

- Não: chama a si própria (fact)
  - Maior:
  - `addi $a0, $a0, -1`      `#argumento = n-1`
  - `jal fact`



# Procedimentos aninhados

- Restaura o contexto do procedimento chamador
  - lw \$a0, 0(\$sp)
  - lw \$ra, 4(\$sp)
  - addi \$sp, \$sp, 8

# Procedimentos aninhados

- Retorna o valor do produto
  - mul \$v0, \$a0, \$v0
    - Façam de conta que existe esse mul ;)
  - jr \$ra

# CONSIDERAÇÕES FINAIS

# Considerações finais

- Procedimentos
  - Etapas e suporte
- A pilha
  - O que come, como se reproduz
- Procedimentos folha
  - Responsabilidades
- Procedimentos aninhados
  - Responsabilidades adicionais

# Considerações finais

- Próxima aula
  - Tradução e inicialização de um programa

# INE5607 – Organização e Arquitetura de Computadores

Linguagem de Montagem e de Máquina

## Aula 12: Suporte à chamada de procedimentos

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

