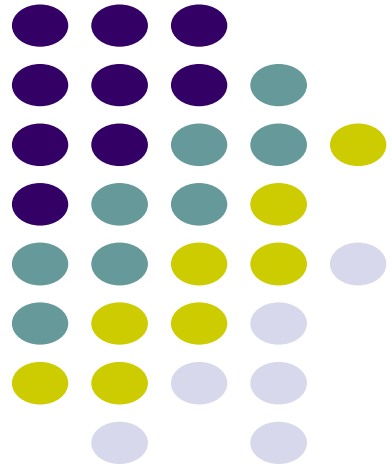
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

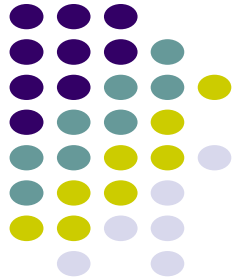
Arquitetura e Organização de Computadores

Instruções: a linguagem do computador
Parte IV



Prof. Sílvio Fernandes

Exemplo de cálculo de Fatorial



- Calcula o fatorial de um número armazenado em **$4** duas vezes.
- O resultado fica em **$2**.
  - O valor de **$4** é setado para 5 e em seguida o fatorial de 5 é calculado, deixando o resultado, 120, em **$2**.
  - Em seguida **$4** recebe o valor 9 e um novo cálculo, agora para 9!, é realizado e armazenado em **$2** (valor 362880)
- Finalmente os dois fatoriais são somados

Exemplo de cálculo de Fatorial

Valor de entrada: $4
Resultado: $2

```
main:     addi $4,$0, 5
fat1:     addi $8, $0, 1
lab1:     mul $8, $8, $4
          addi $4, $4, -1
          bne $4, $0, lab1
fimFat1:  add $2, $8, $0
          add $3, $2, $0
          addi $4, $0, 9
fat2:     addi $8, $0, 1
lab2:     mul $8, $8, $4
          addi $4, $4, -1
          bne $4, $0, lab2
fimFat2:  add $2, $8, $0
          add $3, $3, $2
```

Cálculo do Fatorial

Cálculo do Fatorial

Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

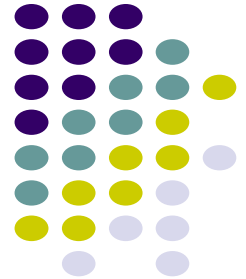
Suporte a procedimentos no hardware do computador

- Procedimento ou função é um recurso bastante empregado em linguagens de alto nível para modularizar o código.
- Podemos ver um procedimento como um espião, pois um espião:
  - Sai com um plano secreto, adquire recursos, realiza a tarefa, cobre seus rastros e depois retorna ao ponto de origem com o resultado desejado.
- De forma similar funcionam os procedimentos, vamos ver como isso funciona.

Suporte a procedimentos no hardware do computador

1. Colocar parâmetros em um lugar onde o procedimento possa acessá-los
2. Transferir o controle para o procedimento
3. Adquirir os recursos de armazenamento necessário para o procedimento
4. Realizar a tarefa desejada
5. Colocar o valor de retorno em um local onde o programa que o chamou possa acessá-lo
6. Retornar o controle para o ponto de origem, pois um procedimento pode ser chamado de vários pontos em um programa

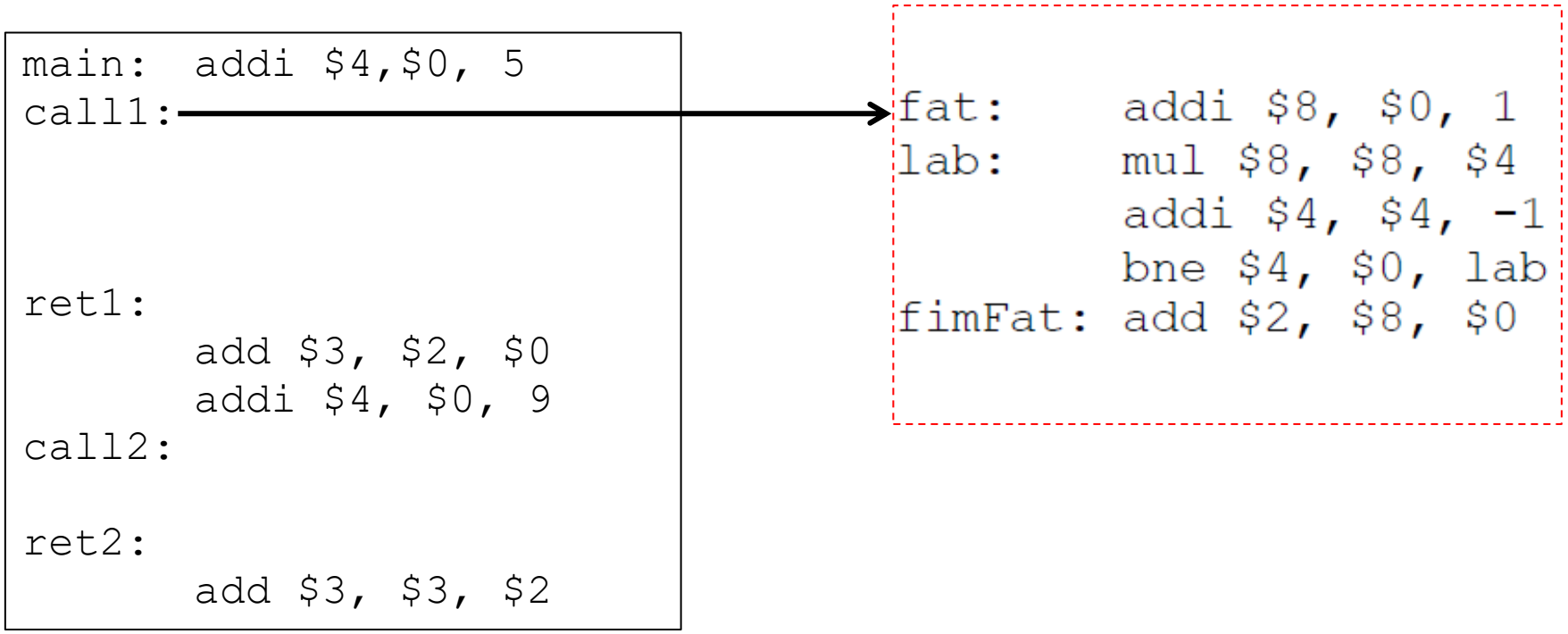
Suporte a procedimentos no hardware do computador



```
main:  addi $4,$0, 5
call1:




ret1:
       add $3, $2, $0
       addi $4, $0, 9
call2:

ret2:
       add $3, $3, $2
```

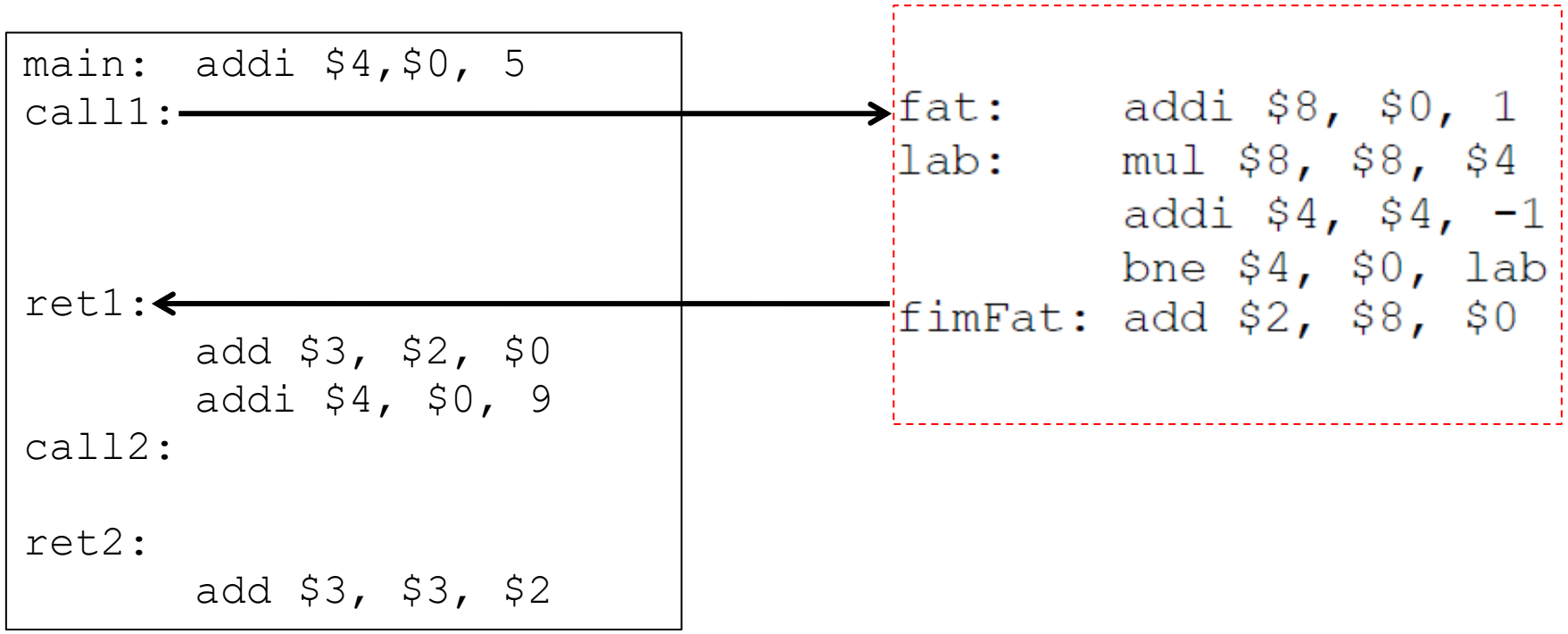
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



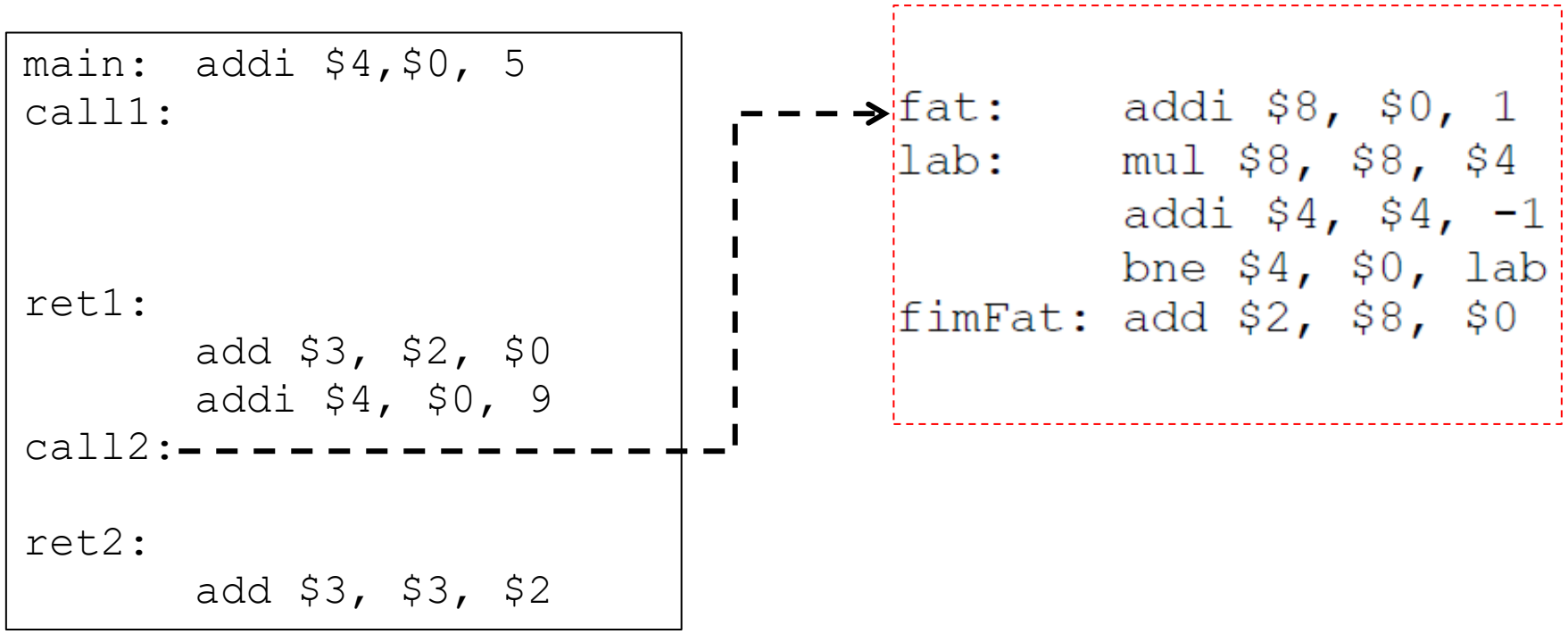
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



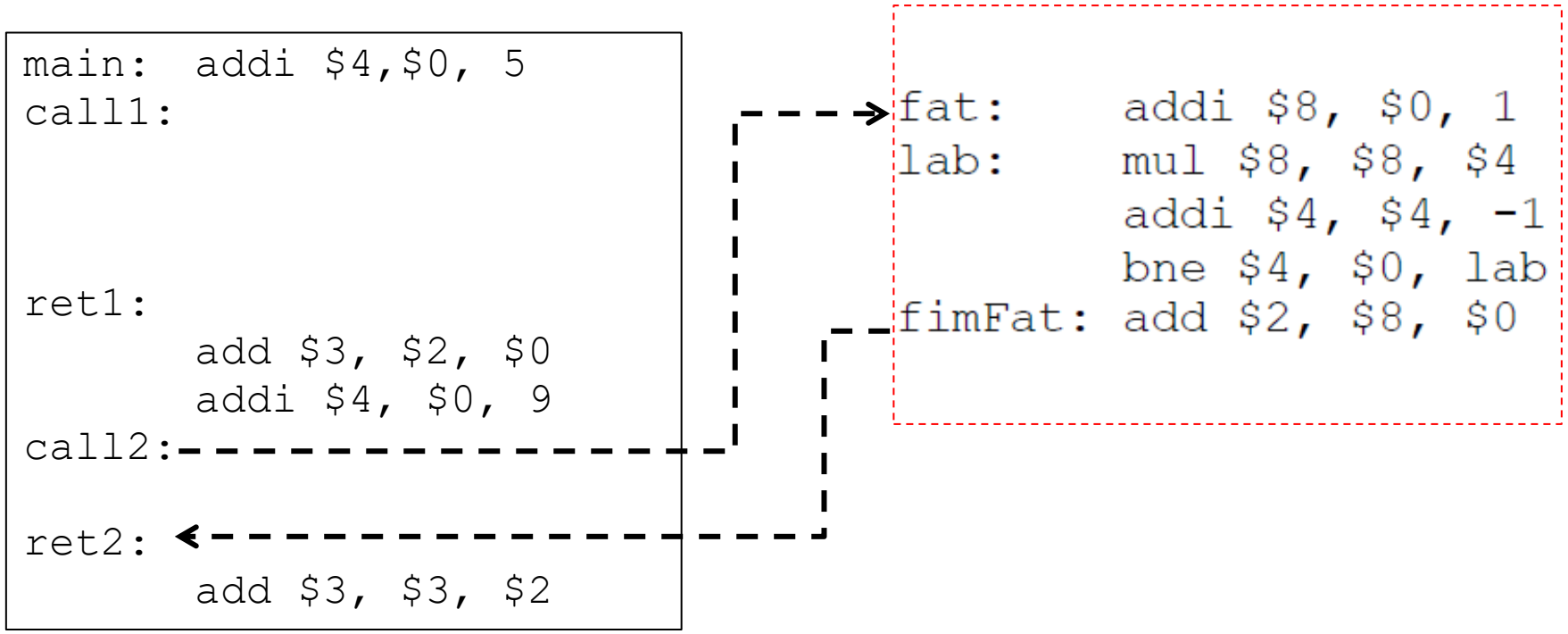
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



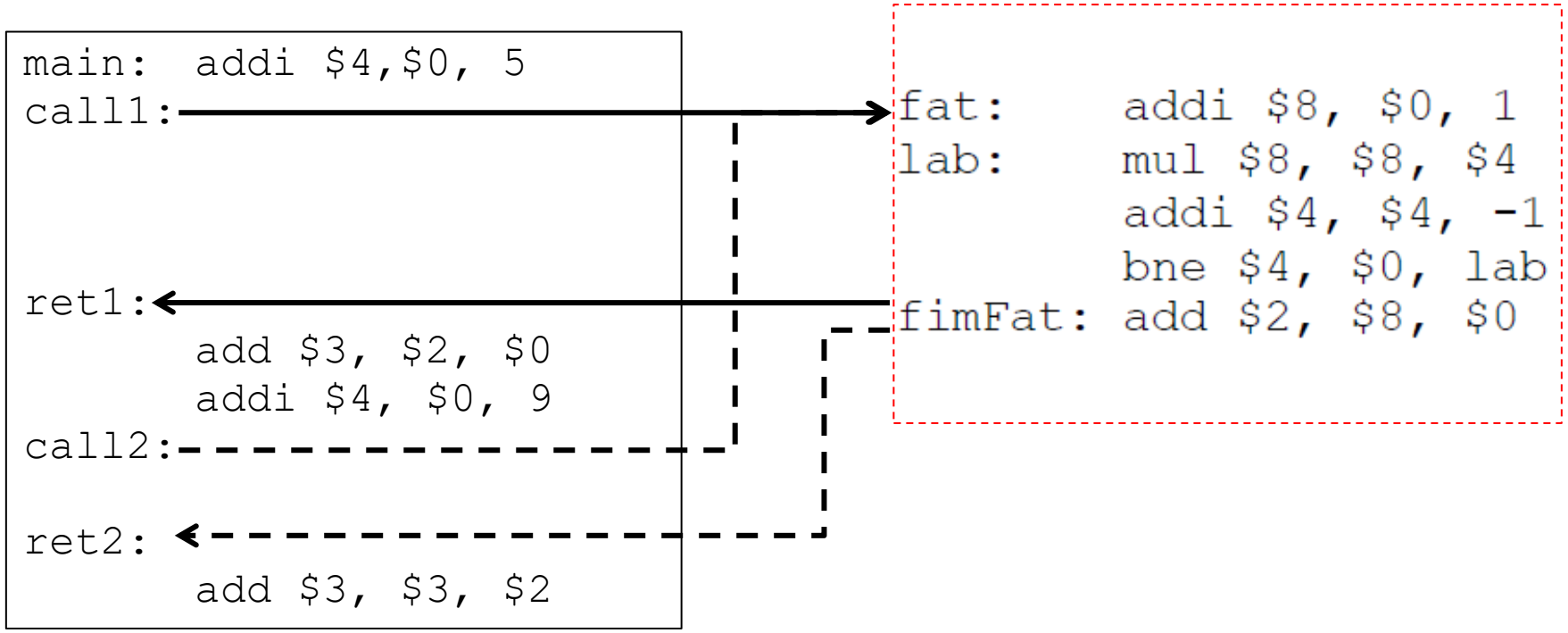
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



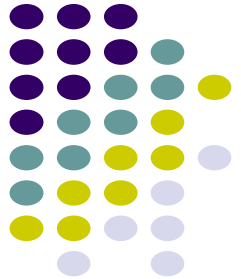
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



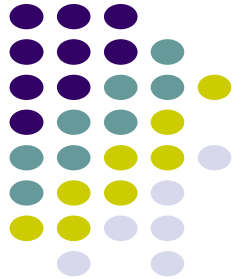
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



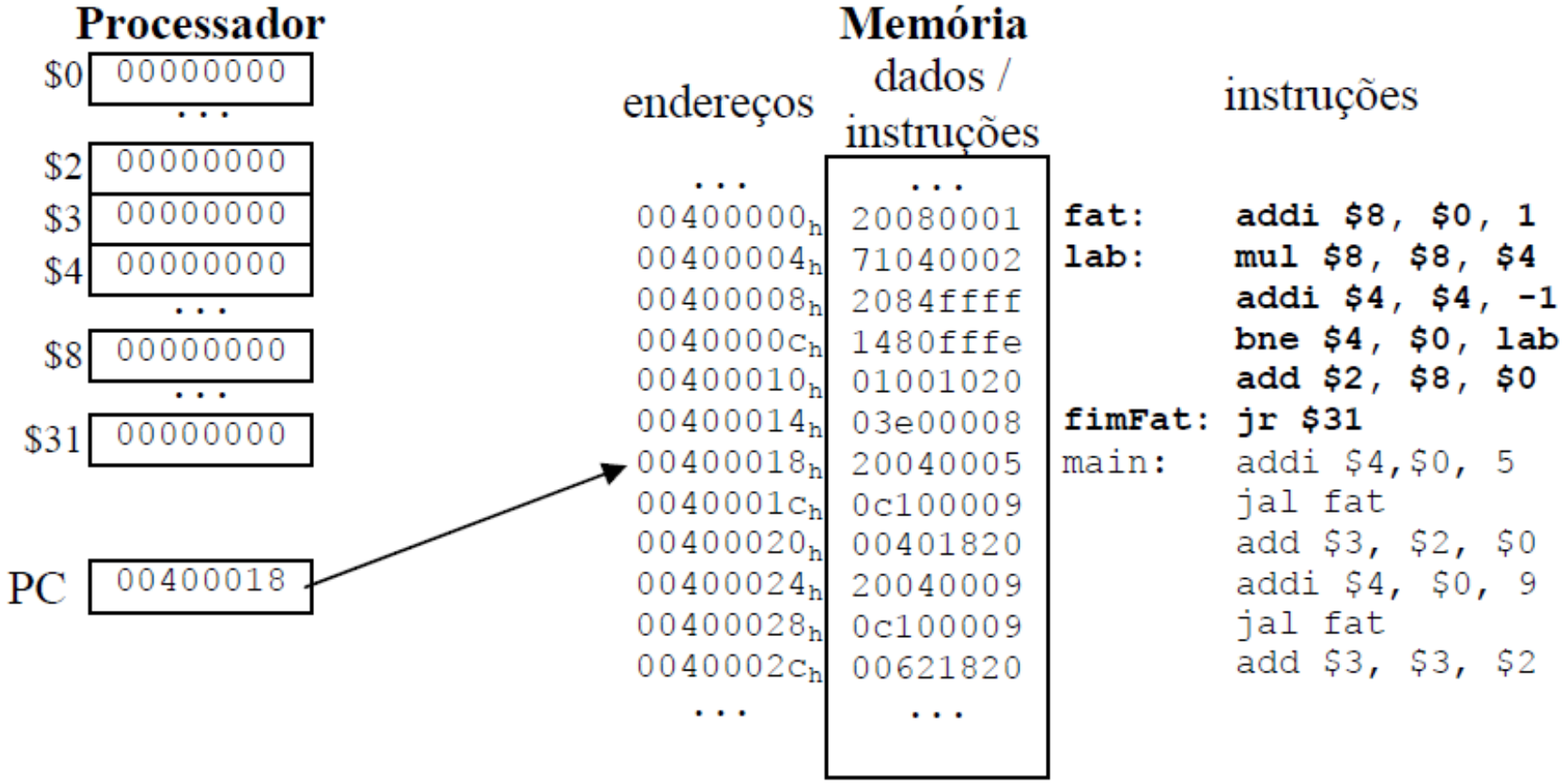
- Além de alocar esses registradores, o assembly do MIPS inclui uma instrução apenas para tratamento de funções/procedimentos.
- Esta instrução desvia para um endereço e simultaneamente salva o endereço da instrução seguinte no registrador **$ra**.
- Esta instrução se chama **jal** (***jump-and-link***). Sua sintaxe é a seguinte:
  - jal EndereçoProcedimento

Suporte a procedimentos no hardware do computador



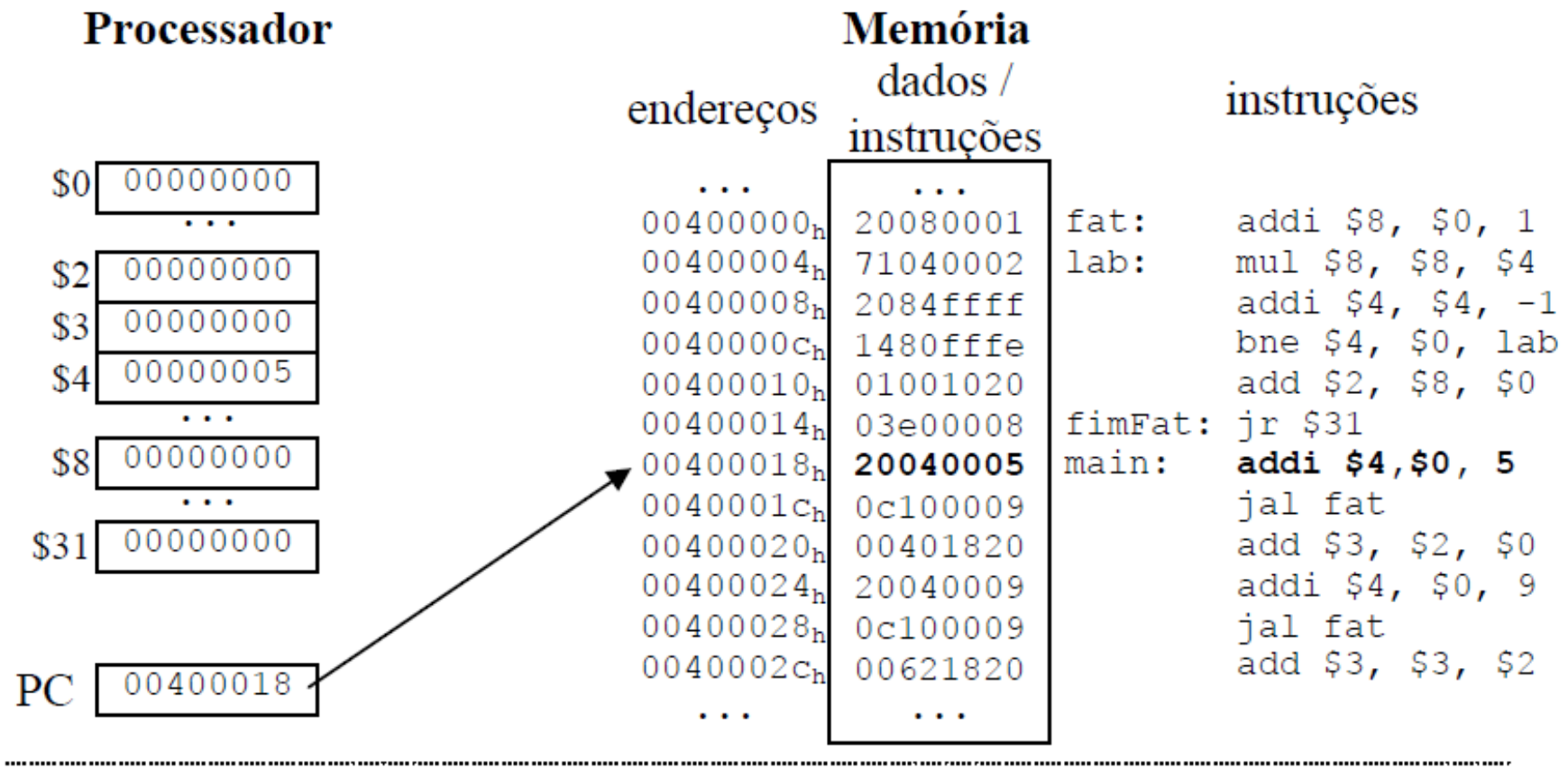
- Existe também um outro registrador de propósito específico denominado PC (program counter) ou **contador de programa**
- Este registrador armazena o endereço da instrução atual sendo executada.
- Então qual endereço salvo por jal no registrador $ra?
  - $PC + 4;
- O MIPS ainda utiliza a instrução *jump register* (jr) para desvio incondicional para o endereço armazenado em um registrador
  - jr $ra

Suporte a procedimentos no hardware do computador



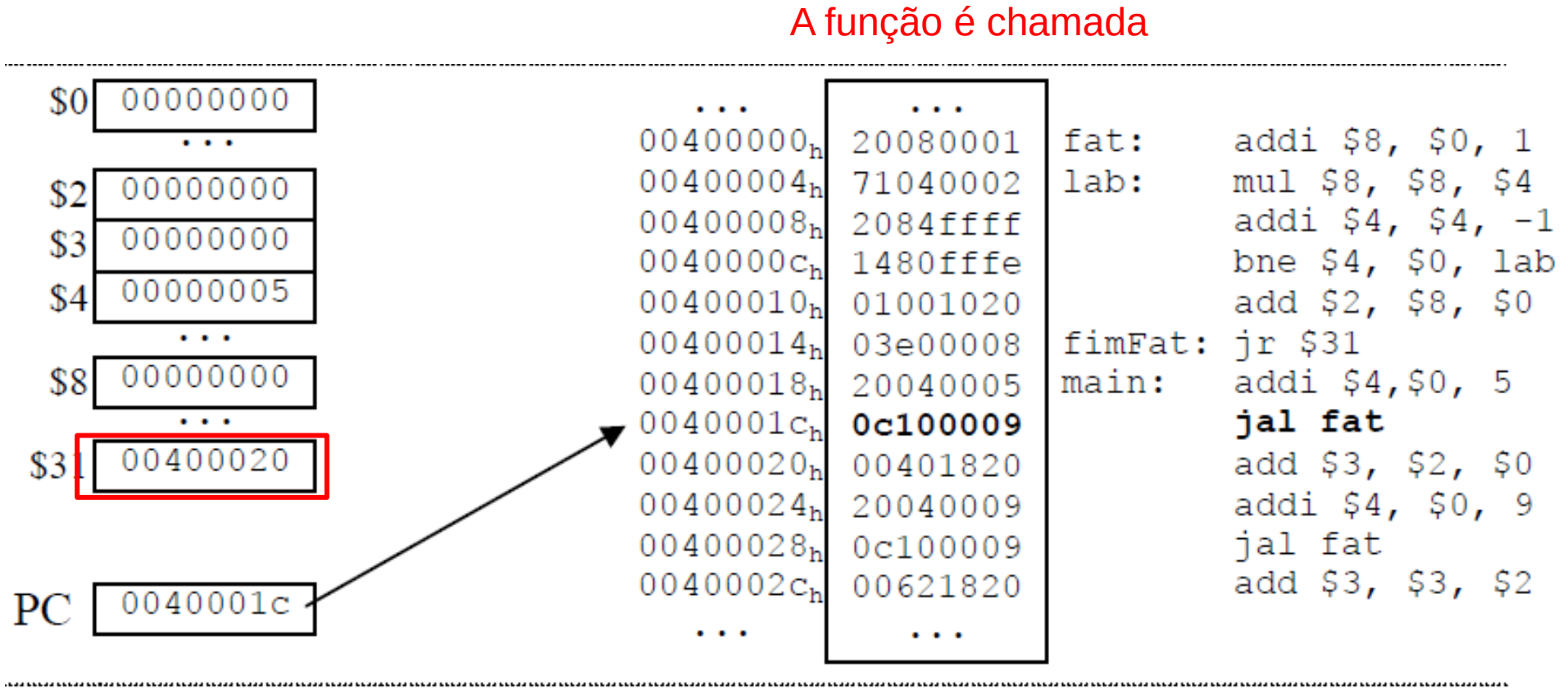
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



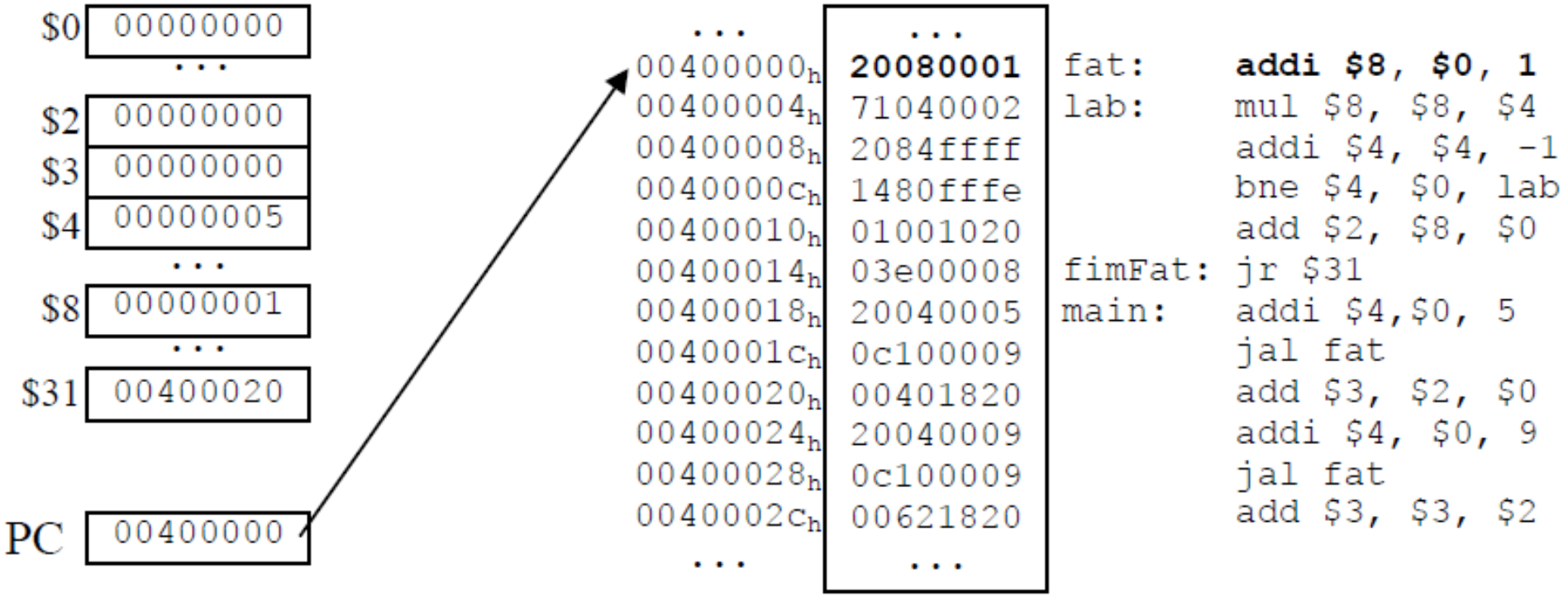
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



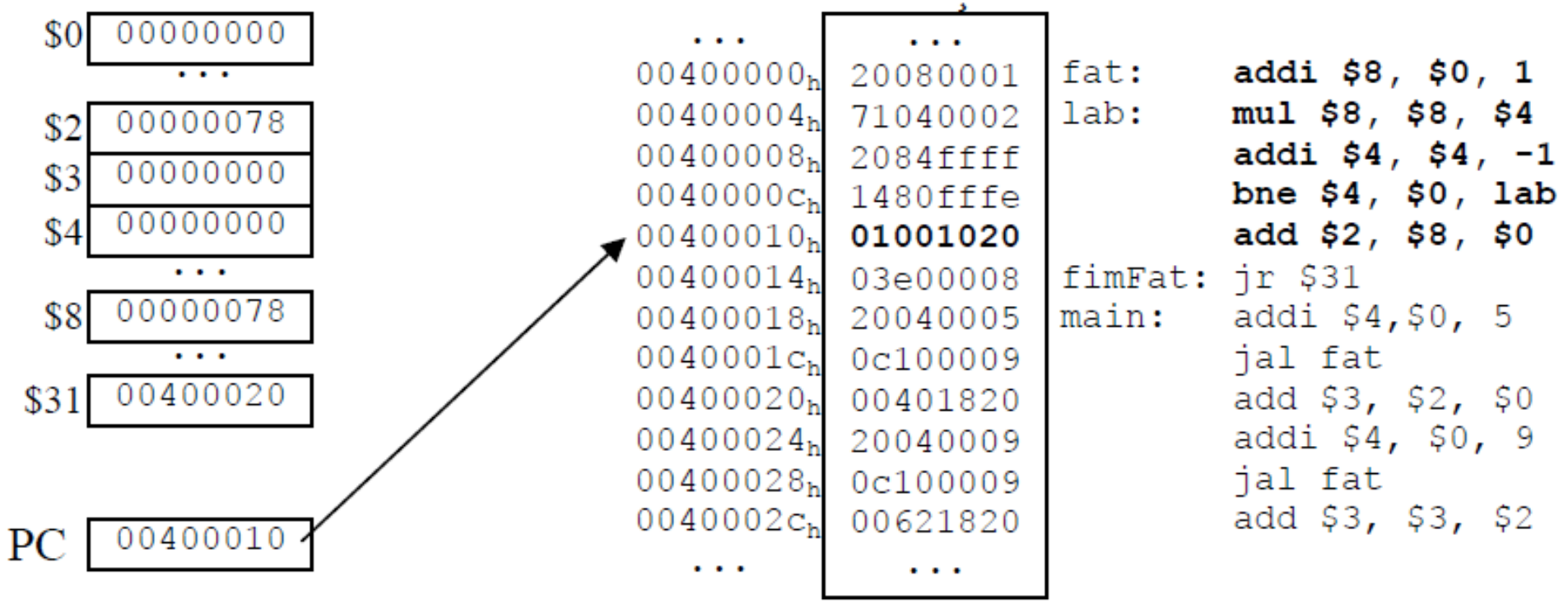
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



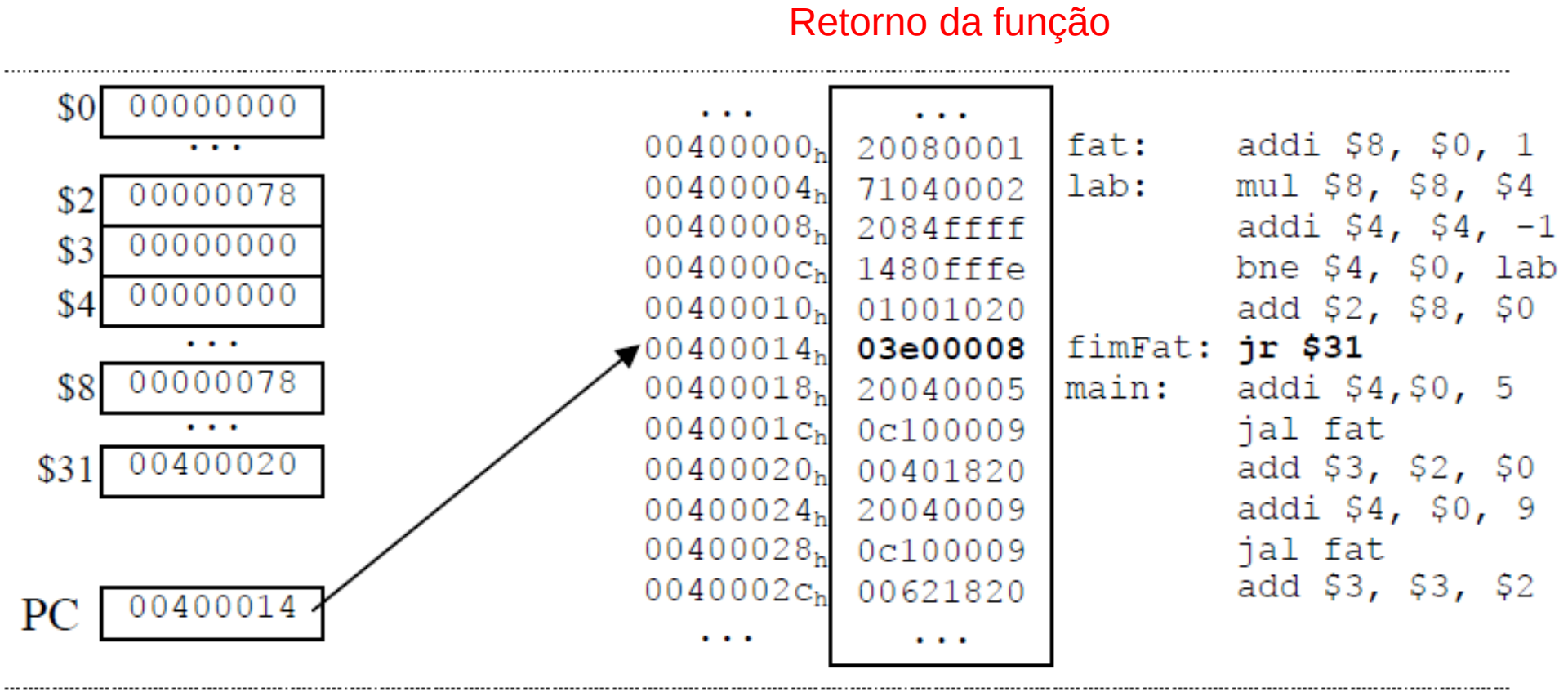
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



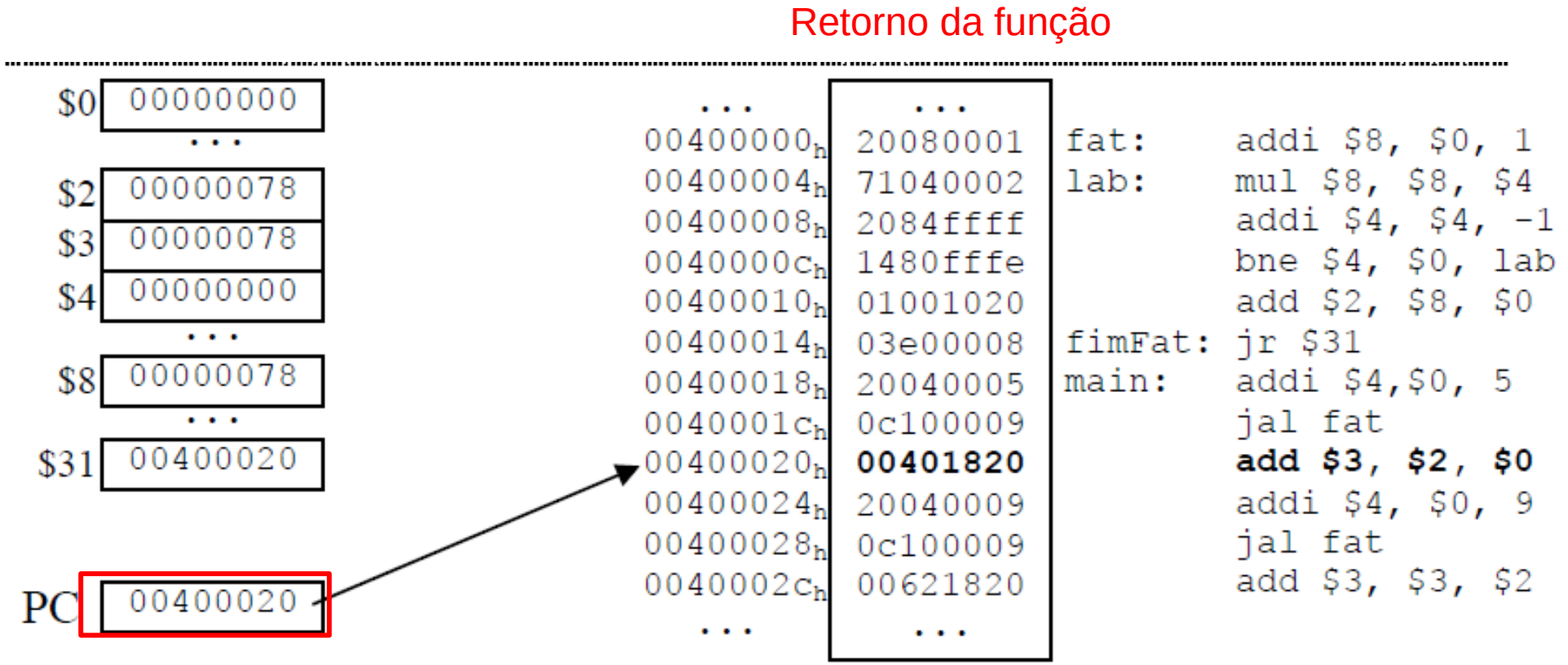
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



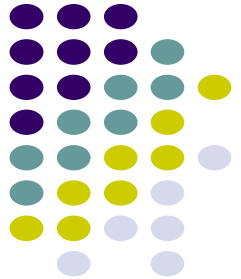
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



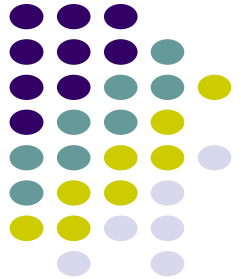
Fonte: WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005

Suporte a procedimentos no hardware do computador



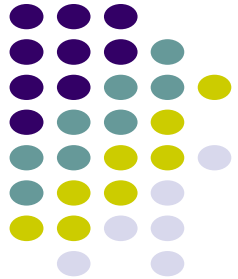
- Na chamada de uma função, alguns registradores do MIPS são reservados para propósito especial, são eles:
  - **$a0 - $a3**: quatro registradores de argumento, para passar parâmetros
  - **$v0-$v1**: dois registradores de valor, para valores de retorno
  - **$ra**: um registrador de endereço de retorno, para retornar ao ponto de origem.

Suporte a procedimentos no hardware do computador



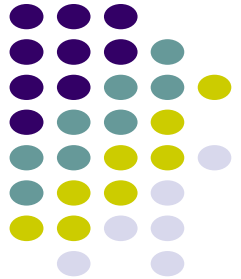
- Então, quando ocorre uma chamada de função:
  - 1. O programa que chama (***caller)***, coloca os valores de parâmetro em $a0 - $a3.
  - 2. Em seguida, ***caller***, utiliza jal X para desviar o procedimento para o procedimento X (o procedimento chamado é denominado ***callee***).
  - 3. O ***callee***, então, realiza os cálculos, coloca os resultados em $v0-$v1.
  - 4. Em seguida o ***callee*** retorna o controle para o ***caller*** usando jr $ra.

Suporte a procedimentos no hardware do computador



- E se precisarmos de mais argumentos além dos 4 registradores para argumentos e os dois para valores de retorno?
- O que o ***callee*** deve fazer?

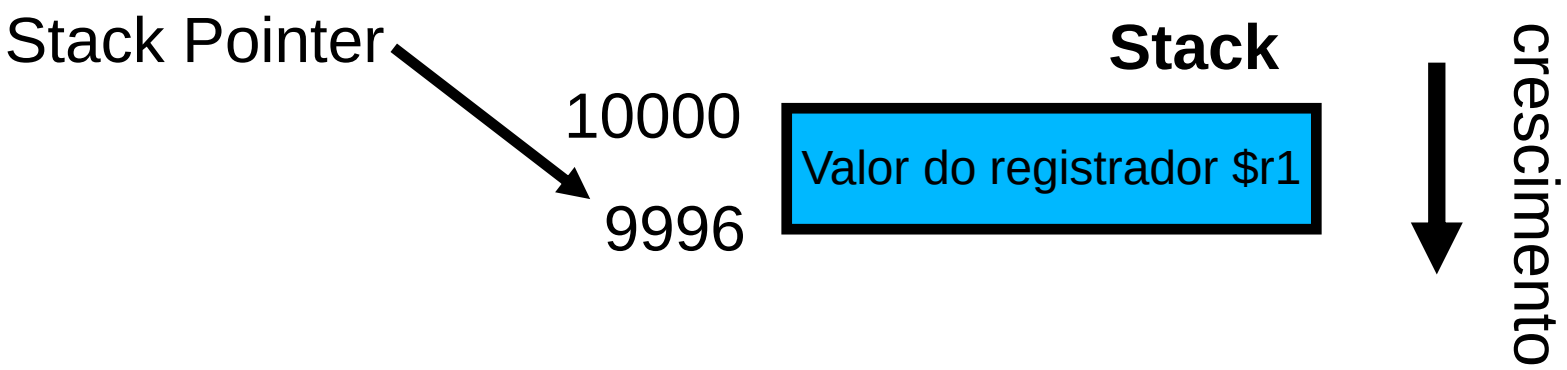
Suporte a procedimentos no hardware do computador



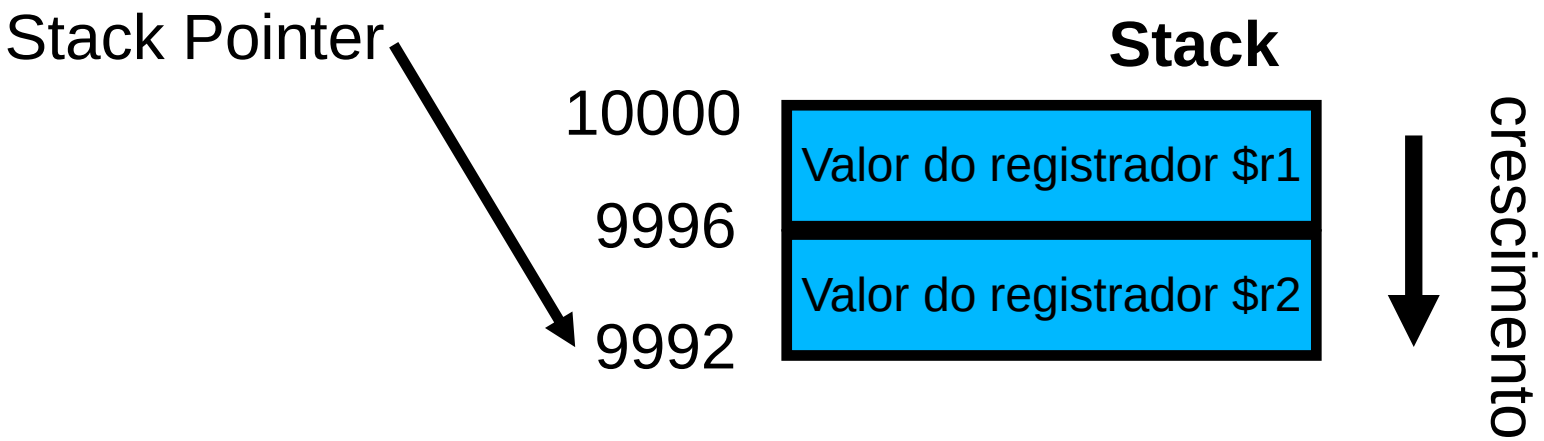
- O ***callee*** executa um processo denominado ***spilling registers***.
  - A idéia básica é armazenar em memória valores que serão necessários posteriormente para a execução do programa.
  - A estrutura de dados utilizada para fazer este armazenamento é uma pilha.
  - Para controle desta pilha, o MIPS possui um registrador especial denominado stack pointer ($sp).
    - O stack pointer sempre aponta para o último endreço alocado mais recentemente.
    - Colocar dados na pilha é *push*
    - Remover dados da pilha é *pop*

Suporte a procedimentos no hardware do computador

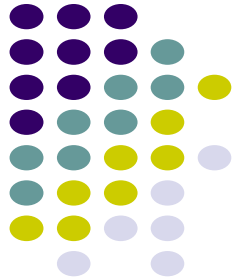
Suporte a procedimentos no hardware do computador



Suporte a procedimentos no hardware do computador



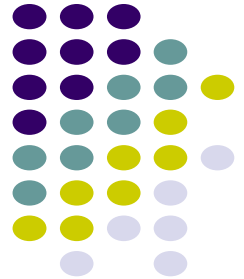
Suporte a procedimentos no hardware do computador



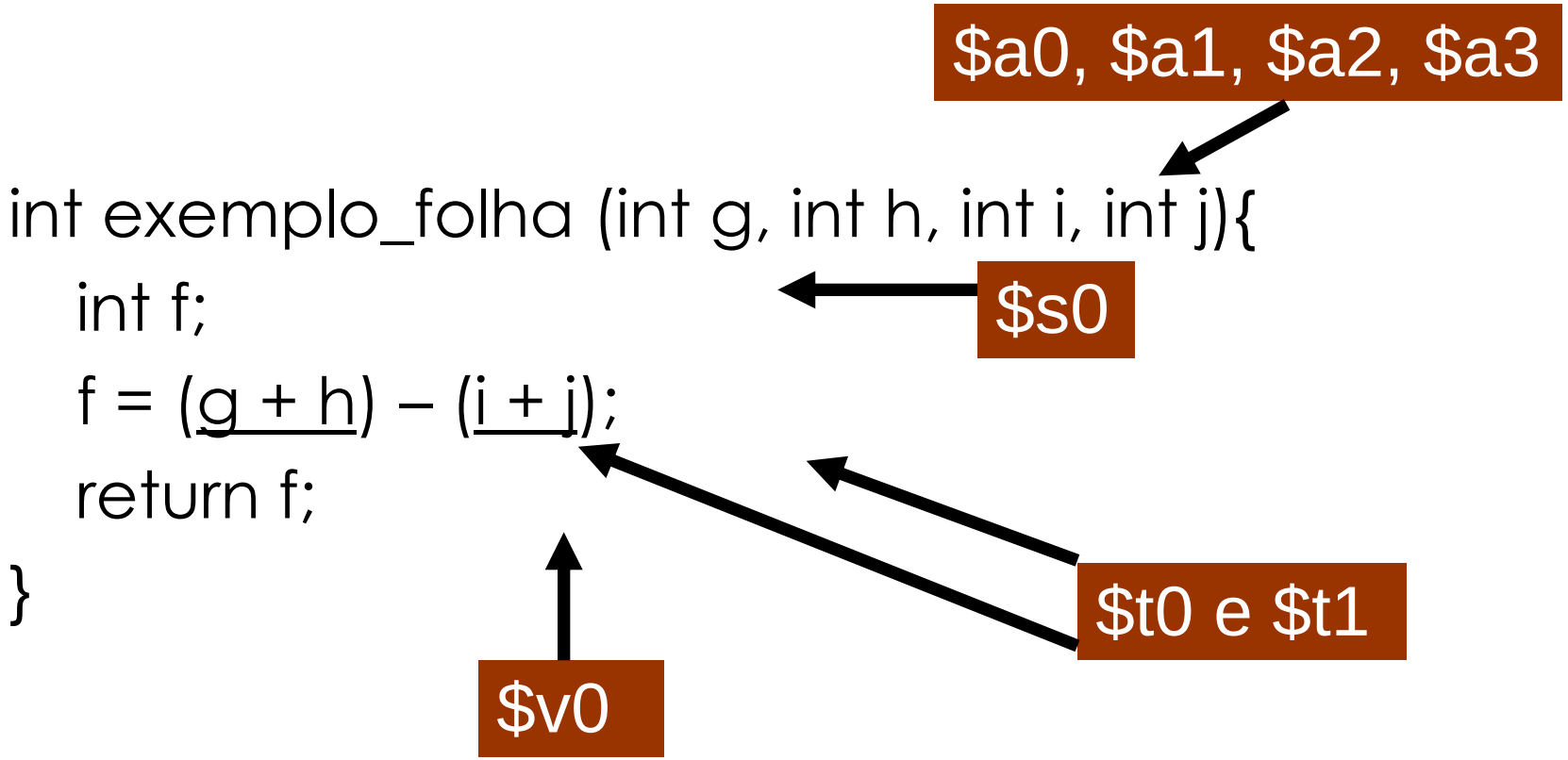
- Qual seria o código assembly do seguinte procedimento C?

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

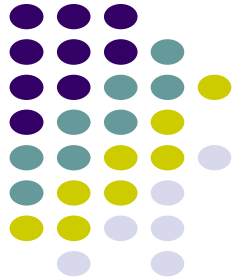
Suporte a procedimentos no hardware do computador



- Fazendo o mapeamento dos elementos da função para registradores, teremos:



Suporte a procedimentos no hardware do computador



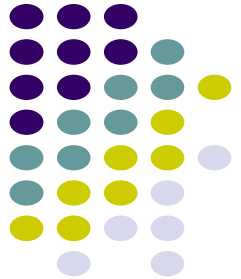
- Percebam a existência de registradores conflitantes, ou seja, que já podem estar sendo utilizados ($s0, $t0 e $t1).

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

$s0

$t0 e $t1

Suporte a procedimentos no hardware do computador

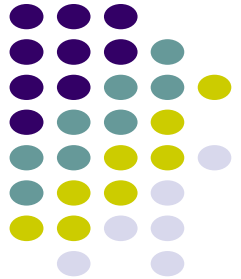


Alguém chama jal exemplo_folha

```
#Liberando registradores
exemplo_folha:
addi $sp, $sp, -12   # ajusta a pilha criando
                     #espaço para três itens
```


```
sw $t1, 8($sp)   #salva registradores $t1, $t0 e $s0
sw $t0, 4($sp)   # para ser usado depois
sw $s0, 0($sp)
```

Suporte a procedimentos no hardware do computador

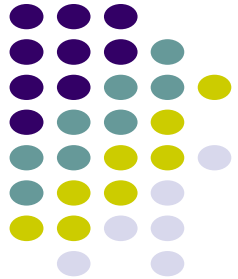


```
add $t0, $a0, $a1    # $t0 = g+h
add $t1, $a2, $a3    # $t1 = i + j
sub $s0, $t0, $t1    # $s0 = (g+h) – (i + j), ou seja, f
```

```
#Copiando o valor de f para ser retornado
add $v0, $s0, $zero
```

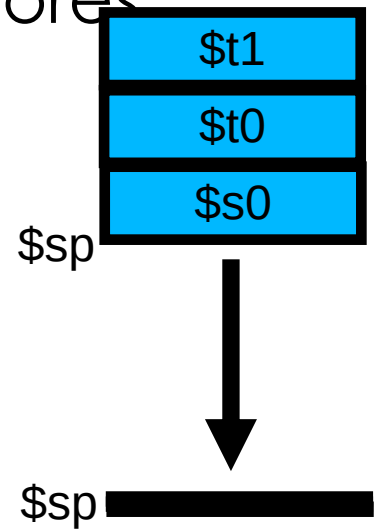
```
int exemplo_folha (int g, int h, int i, int j){
    int f;
    f = (g + h) – (i + j);
    return f;
}
```

Suporte a procedimentos no hardware do computador



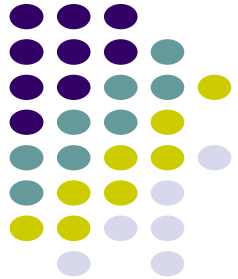
#Restaurando os três valores antigos dos registradores

```
lw $s0, 0($sp)
lw $t0, 4($sp)
lw $t1, 8($sp)
addi $sp, $sp, 12
```



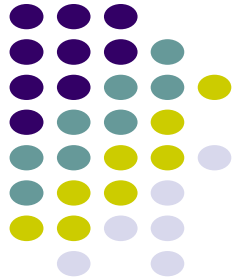

```
# E finalmente, o procedimento termina
jr $ra     # Desvia de volta à rotina que chamou
```

Suporte a procedimentos no hardware do computador



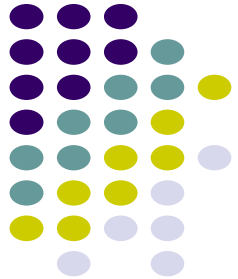
- No exemplo anterior, salvamos os valores dos registradores $t0 e $t1. Porém existem casos em que o valor de um registrador pode ser descartado sem problemas.
- Por causa disso, o MIPS separa 18 dos registradores em dois grupos:
  - **$t0-$t9**: 10 registradores ***temporários*** que não são preservados pelo procedimento chamado (callee) em uma chamada de procedimento.
  - **$s0-$s7**: 8 registradores ***Salvos*** que precisam ser preservados em uma chamada de procedimento (se forem usados, o procedimento chamado os salva e restaura).

O que é preservado e o que não é em chamada de procedimentos



Preserved	Not preserved
Saved registers: `$s0–$s7`	Temporary registers: `$t0–$t9`
Stack pointer register: `$sp`	Argument registers: `$a0–$a3`
Return address register: `$ra`	Return value registers: `$v0–$v1`
Stack above the stack pointer	Stack below the stack pointer

Registradores do MIPS

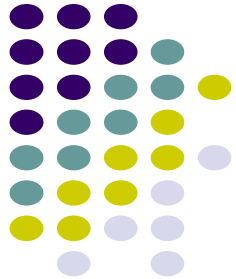


Name	Register number	Usage	Preserved on call?
$zero	0	The constant value 0	n.a.
$v0–$v1	2–3	Values for results and expression evaluation	no
$a0–$a3	4–7	Arguments	no
$t0–$t7	8–15	Temporaries	no
$s0–$s7	16–23	Saved	yes
$t8–$t9	24–25	More temporaries	no
$gp	28	Global pointer	yes
$sp	29	Stack pointer	yes
$fp	30	Frame pointer	yes
$ra	31	Return address	yes

Assembly do MIPS

Categoria	Instrução	Exemplo	Significado	Comentário
Aritmética	Add	add $s1, $s2, $s3	$s1 = $s2+$s3	3 operandos; dados registradores
	Subtract	sub $s1, $s2, $s3	$s1 = $s2-$s3	3 operandos; dados registradores
	Add Immediate	dddi $s1, $s2, 100	$s1 = $s2+100	Usada para somar constantes
Lógica	And	and $s1, $s2, $s3	$s1 = $s2 & $s3	3 operandos em reg; AND bit a bit
	Or	or $s1, $s2, $s3	$s1 = $s2 | $s3	3 operandos em reg; OR bit a bit
	Nor	nor $s1, $s2, $s3	$s1 = ~($s2 | $s3)	3 operandos em reg; NOR bit a bit
	And Immediate	andi $s1, $s2, 100	$s1 = $s2 & 100	AND bit a bit entre reg. e constante
	Or Immediate	ori $s1, $s2, 100	$s1 = $s2 | 100	OR bit a bit entre reg. e constante
	Shift left logical	sll $s1, $s2, 10	$s1 = $s2 << 10	Deslocamento à esquerda por const.
	Shift rigth logical	srl $s1, $s2, 10	$s1 = $s2 >> 10	Deslocamento à direita por const.
Transferência de dados	Load word	lw $s1, 100($s2)	$s1 = Mem[$s2+100]	Dados da mem. para o registrador
	Store Word	sw $s1, 100($s2)	Mem[$s2+100] = $s1	Dados do registrador para a mem.

Assembly do MIPS



Categoria	Instrução	Exemplo	Significado	Comentário
Desvio Condicional	Branch on equal	beq $s1, $s2, L	If($s1 == $s2) go to L	Testa igualdade e desvia
	Branch on not equal	bne $s1, $s2, L	If($s1 != $s2) go to L	Testa desigualdade e desvia
	Set on less than	stl $s1, $s2, $s3	If($s2 < $s3) $s1 = 1; Else $s1 = 0	Compara menor que
Desvio incondicional	Set on less than immediate	stli $s1, $s2, 100	If($s2 < 100) $s1 = 1; Else $s1 = 0	Compara menor que imediato
	Jump	j L	go to L	Desvia para endereço de destino
	Jump regiter	jr $ra	go to $ra	Para retorno de procedimento
	Jump and link	jal L	$ra = PC+4 go to L	Para chamada de procedimento

Questões

- Escrever um programa em assembly que tenha:
  - Uma função que soma 2 valores e retorna o resultado da soma
  - Um vetor com 10 valores
  - Um laço de repetição que chame a função passando um valor do vetor e o valor acumulado da chamada anterior
- Imprima o resultado final da soma

Questões

- Escrever um programa em assembly que chama uma função que calcula a soma de 4 variáveis que representam 4 notas (n1, n2, n3 e n4).
  - A função deve somar as 3 primeiras notas, considerando-a como nota parcial (NP)
  - Caso a NP seja superior ou igual a 21 deve imprimir a nota e informar que o aluno foi aprovado
  - Caso a NP seja abaixo de 21, a NP deve ser dividida por 2 (use Shift para direita) e o resultado da divisão somado a n4 e verificar as seguintes condições da nota final (NF)
    - Se for maior ou igual a 15 imprima a nota e informe que o aluno foi aprovado em 4ª. Prova
    - Caso contrário imprima a nota e informe que o aluno foi reprovado

Questões

- Escrever um programa em assembly que utilize a função da questão anterior para calcular a média de 5 alunos.

Referências

- PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005.
- WANDERLEY NETTO, Bráulio. **Arquitetura de Computadores**: A visão do software. Natal: Editora do CEFET-RN, 2005
- Notas de aula do Prof. André Luis Meneses Silva