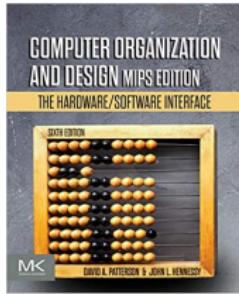


Aula 05 – Linguagem Assembly (Continuação)

Prof. Dr. Clodoaldo A. de Moraes Lima

Material baseado no livro “Patterson, David A., Hennessy, J. L. - Computer Organization And Design: The Hardware/Software Interface”



Uso de Registradores em Função/Proc.

- Suponha uma função/procedimento que necessite de quatro parâmetros.
- Ao final da execução os dados utilizados por tais registradores necessitam ser restaurados aos originais.
 - Analogia do espião: cobertura dos rastros.
- A estrutura ideal para o gerenciamento dos registradores é uma pilha (stack).
- Uma pilha mantém sempre um apontador o endereço mais recentemente alocado.

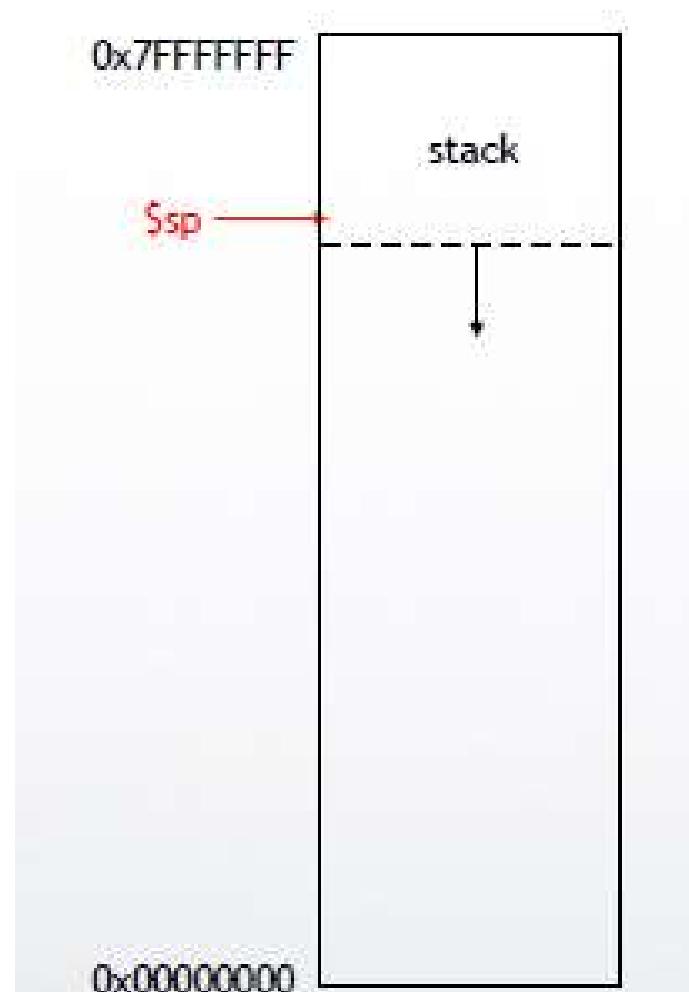
Uso de Registradores em Função/Proc.

- O apontador de pilha (stack pointer - \$sp) é ajustado para cada palavra do registrador colocado ou retirado da pilha.
- Nomenclatura própria:
 - **push**: Adição de um novo item a pilha;
 - **pop**: Remoção de um item da pilha.
- Por convenção: Stack cresce de endereços de memória mais altos para os mais baixos.
 - Adição de items se dá pela subtração do \$sp.

Suporte a Procedimentos e Funções

Uso de Registradores em Função/Proc.

- push e pop deve ser feito pelo programador.



Suporte a Procedimentos e Funções

Uso de Registradores em Função/Proc.

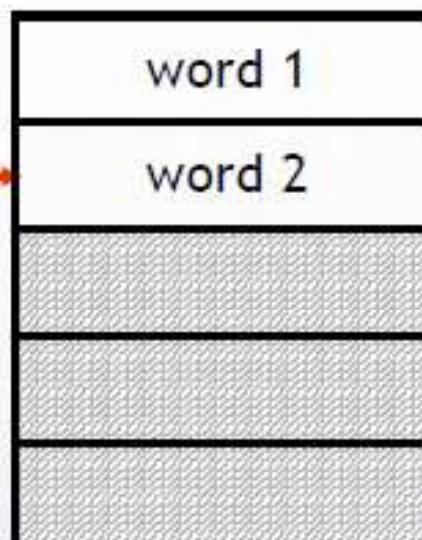
- Para colocar elementos na pilha (**push**)
 - Mover o **stack pointer (\$sp)** para baixo
- Exemplo

```
subi $sp, $sp, 8  
sw $t1, 4($sp)  
sw $t2, 0($sp)
```

\$sp →

ou:

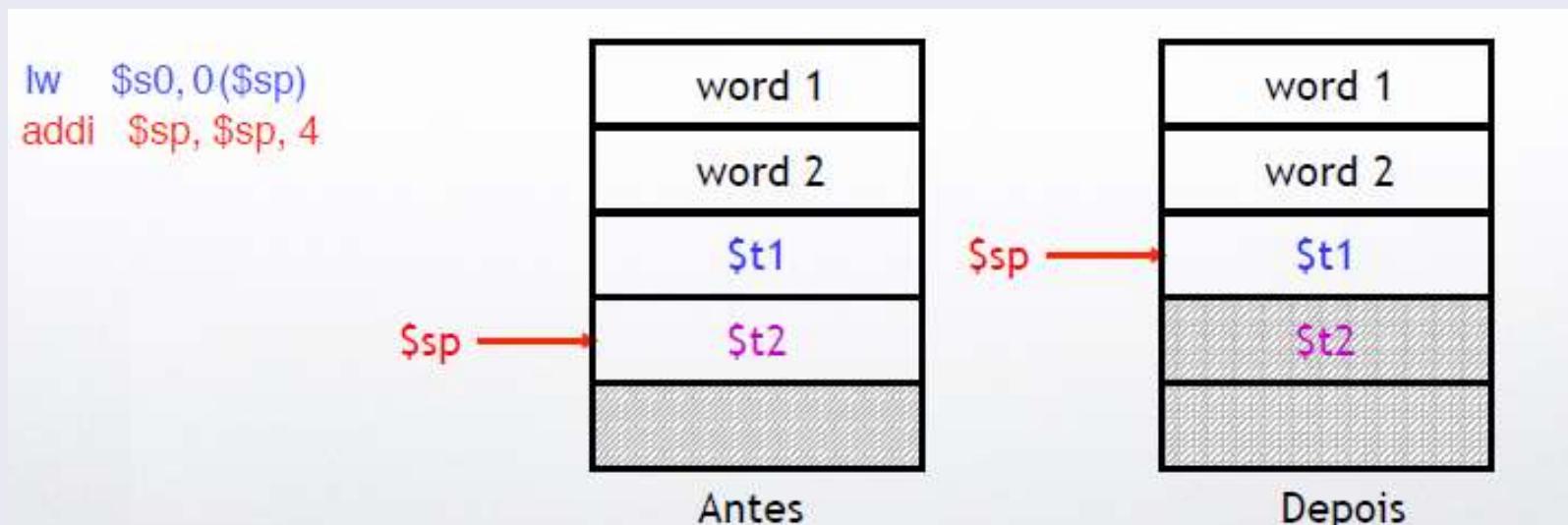
```
sw $t1, -4($sp)  
sw $t2, -8($sp)  
sub $sp, $sp, 8
```



Suporte a Procedimentos e Funções

Uso de Registradores em Função/Proc.

- Para colocar elementos na pilha (**push**)
 - Mover o **stack pointer (\$sp)** para cima
- Exemplo



Uso de Registradores em Função/Proc.

- Exemplo 12

- Traduzir a seguinte função em C para assembly MIPS (sem chamada):

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

- As variáveis g,h, i e j correspondem respectivamente aos registradores \$a0, \$a1, \$a2 e \$a3.
- A variável f está associada ao registrador \$s0

Uso de Registradores em Função/Proc.

- Resposta

- Passo 1: Salvamento dos registradores utilizados pelo procedimento.
 - Instrução interna ao procedimento:
 $f = (g + h) - (i + j);$
add t0, g, h # variável temporária t0 contendo g+h
add t1, i, j # variável temporária t0 contendo i+j
sub f, t0, t1 # pega t0 - t1, que é (g+h) -(i+j)
 - Uso de dois registradores temporários (\$t0 e \$t1).
 - "Apagar os rastros": Valores dos temporários podem já estar sendo utilizados.

Uso de Registradores em Função/Proc.

- Resposta

- Passo 1: Salvamento dos registradores utilizados pelo procedimento.
 - Necessário guardar valores anteriores dos registradores a serem utilizados.
 - Guardar valores = Colocá-los na pilha para posterior recuperação.
 - Entretanto, previamente é necessário alocar espaço para 3 registradores (a serem utilizados) na pilha.

Uso de Registradores em Função/Proc.

- Resposta

- Passo 1: Salvamento dos registradores utilizados pelo procedimento.
 - Criando o espaço necessário para três registradores na pilha (12 bytes):

`addi $sp, $sp, -12 #Ajuste o ponteiro para abrir espaço para 3 itens.`

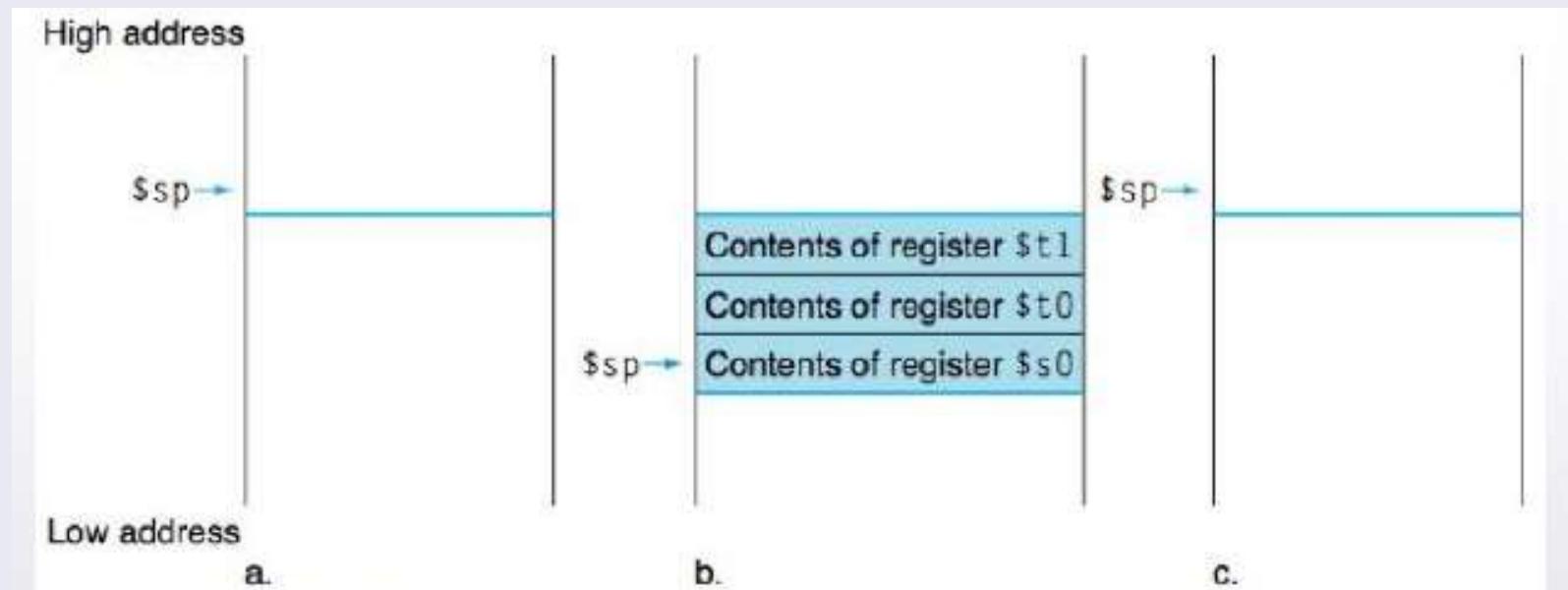
`sw $t1, 8($sp) # salva o registrador $t1`

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

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

Uso de Registradores em Função/Proc.

- Resposta
 - Passo 1: Salvamento dos registradores utilizados pelo procedimento.



Uso de Registradores em Função/Proc.

- Resposta

- Passo 2: Fazer o cálculo da soma expressão:

`add $t0, $a0, $a1 # register $t0 contains g + h`

`add $t1, $a2, $a3 # register $t1 contains i + j`

`sub $s0, $t0, $t1 # f = $t0 - $t1, que é (g + h) - (i + j)`

- Passo 3: Copiar o valor de retorno para o registrador de retorno (return register)

`add $v0, $s0, $zero # retorno f ($v0 = $s0 + 0)`

Uso de Registradores em Função/Proc.

- Resposta

- Passo 4: Antes de retornar, fazer a restauração dos três valores previamente salvos na pilha (pop):

`lw $s0,0($sp) # recupera registrador $s0 para quem chamou`

`lw $t0,4($sp) # recupera registrador $t0 para quem chamou`

`lw $t1,8($sp) # recupera registrador $t1 para quem chamou`

`addi $sp, $sp, 12) # ajusta o ponteiro para deletar 3 items`

- Passo 5: Retornar o fluxo de execução para o endereço de retorno:

`jr $ra # jump back to calling routine`

Uso de Registradores em Função/Proc.

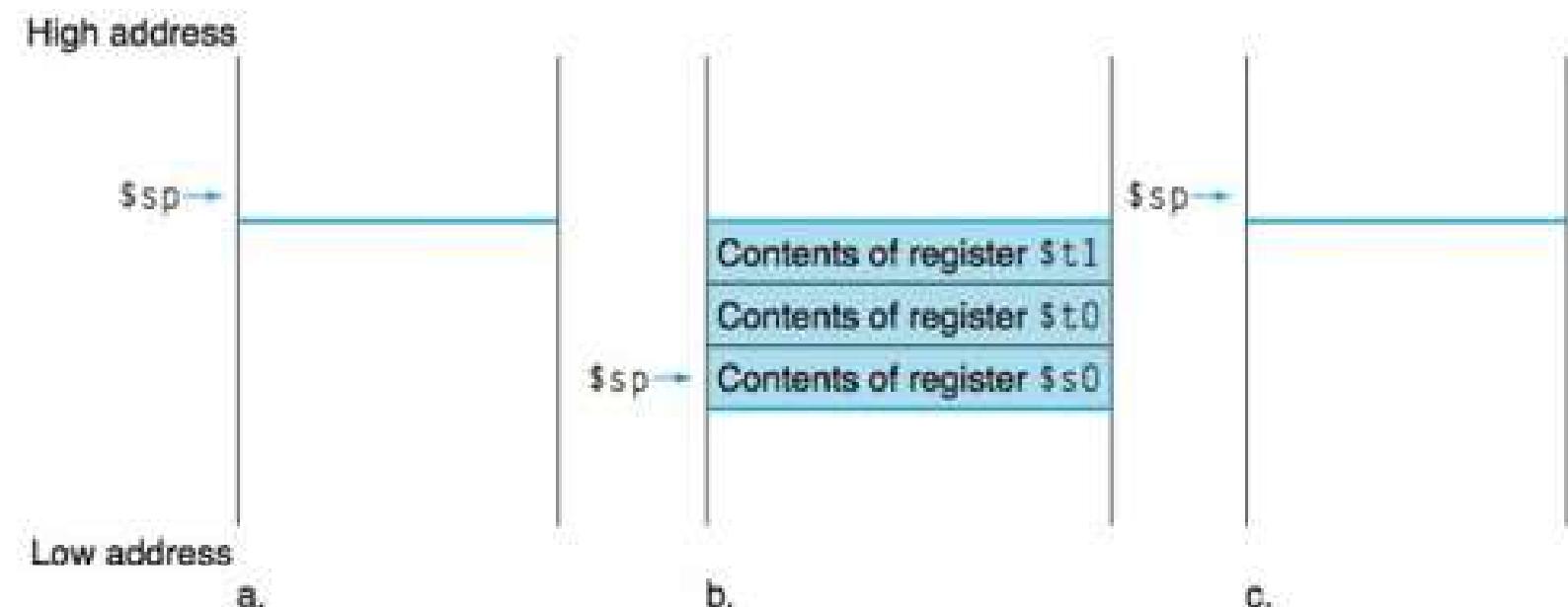
- Argumentos são passados nos registradores \$a0 a \$a3.
- Limite de 4 argumentos por função.
- O que fazer se uma função possuir > 4 argumentos? Como fazer uso de mais registradores?
 - Inicialmente faz-se necessário um melhor entendimento do funcionamento da pilha.

Tratamento e Manutenção dos Dados na Pilha

- Linguagem C possui duas classes de armazenamento de variáveis:
 - Automática: Variáveis locais a uma função. São descartadas quando o procedimento chega ao seu final.
 - Estática: Variáveis que mantém sua existência mesmo entre chamadas de diferentes procedimentos.
- Variáveis em C declaradas fora de qualquer procedimento são consideradas estáticas. O restante automáticas.
- Acesso a informações estáticas: \$gp (global pointer).

Tratamento e Manutenção dos Dados na Pilha

- Necessário preservar o conteúdo anterior da pilha, garantindo que a função chamadora conseguirá recuperar os mesmos dados após a chamada da função.
- Os valores na pilha acima de \$sp são preservados simplesmente fazendo com que a função chamada não tenha acesso aos mesmos.



Tratamento e Manutenção dos Dados na Pilha

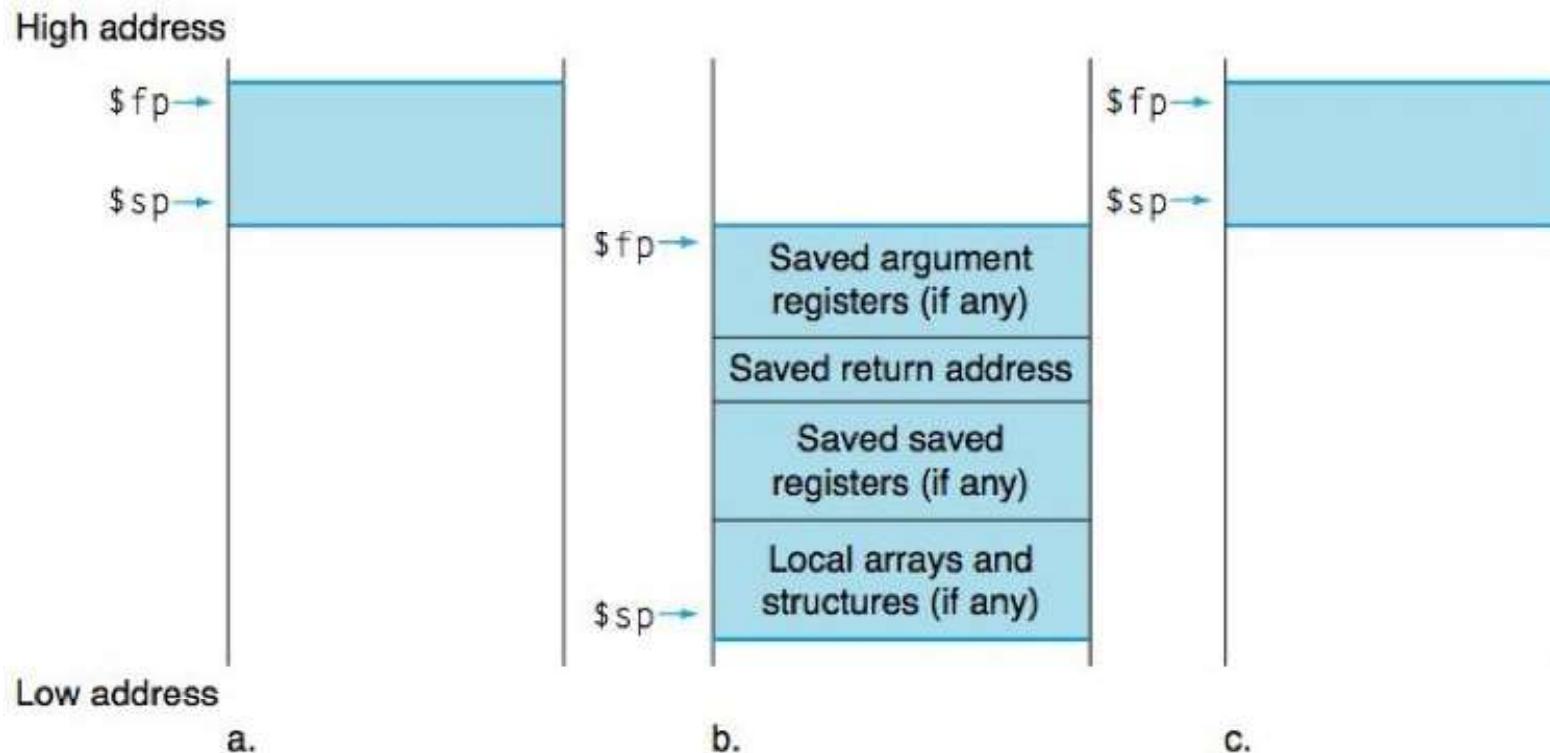
- \$sp é preservado com a soma do mesmo valor que foi subtraído do mesmo anteriormente.
- Outros registradores são preservados salvando-os na pilha e restaurando-os de volta.

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

Alocação de Novos Dados na Pilha

- Pilha é utilizada também para armazenar variáveis que são locais a um procedimento mas não cabem nos registradores.
- O segmento de pilha que contém os registradores e variáveis locais de um procedimento.
 - frame de procedimento (procedure frame); ou
 - registro de ativação (activation record).
- Registrador frame pointer (\$fp) aponta para a primeira palavra de um frame de procedimento.

Alocação de Novos Dados na Pilha



Estados da pilha antes(a), durante(b) e depois(c) exec. procedimento

Alocação de Novos Dados na Pilha

- Quando a função é chamada:
 - $\$fp = \sp
 - Pilha é ajustada de forma a conter espaço suficiente para todos os registradores e variáveis em memória.
- Quando função é finalizada:
 - $\$sp = \fp
- Se não existem variáveis locais na pilha dentro de um procedimento, então o compilador economiza tempo pois não precisa salvar e restaurar os valores em $\$fp$.

Alocação de Novos Dados na Pilha

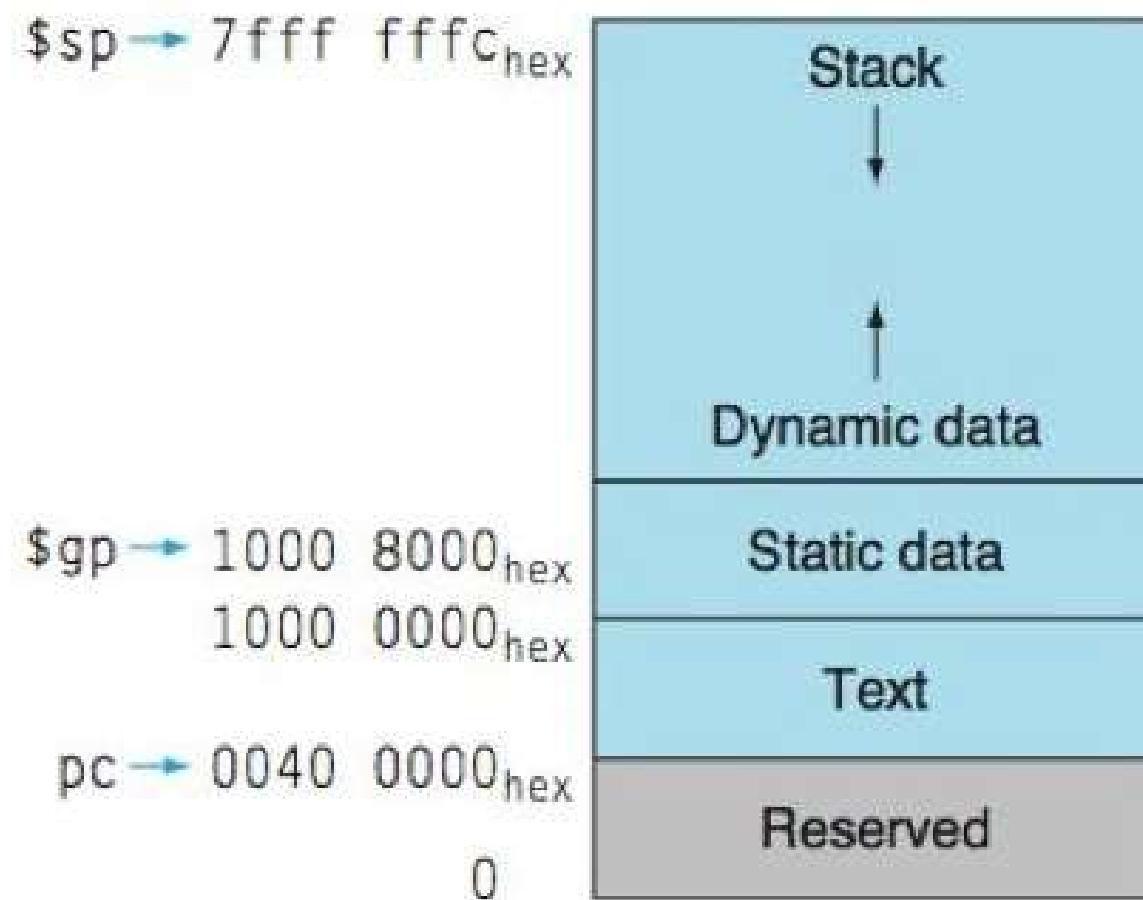
- O que fazer se uma função possuir > 4 argumentos? Como fazer uso de mais registradores?
 - Utilizar a pilha para salvamento dos argumentos restantes.
- Pilha auxilia a aumentar flexibilidade no uso de registradores.
- Então, é necessário compreender como alocar novos dados na memória (heap).

Alocação de Novos Dados na Pilha

- Além de variáveis automáticas, programadores fazem uso também de variáveis estáticas e estruturas dinâmicas.
- Pilha inicia endereçamento em endereços de memória mais altos.
- Na parte mais baixa:
 - Espaço reservado;
 - Código do MIPS (texto);
 - Dados estáticos; e
 - Dados dinâmicos.

Suporte a Procedimentos e Funções

Alocação de Novos Dados na Pilha



Alocação de Novos Dados na Pilha

- A parte mais baixa da memória é o segmento de texto
- Depois o segmento de dados estáticos (constantes, globais, estáticas)
- Depois a heap (segmento para estruturas de dados que mudam ao longo da execução do programa, ex., com uso de malloc em C e new em Java)
- O ponteiro global \$gp é alocado no meio do segmento de dados

Alocação de Novos Dados na Pilha

- Linguagem C aloca e libera espaços na pilha através das funções malloc() e free().
- Esquecer de liberar espaços alocados na pilha pode levar ao problema conhecido como vazamento de memória (memory leak).
- Memory leaks podem levar o SO a interromper seu funcionamento.
- Liberar memória muito cedo pode levar ao problema de “ponteiros pendentes” (dangling pointers).

Alocação de Novos Dados na Pilha

- Convenção de uso dos registradores para o 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

Alocação de Novos Dados na Pilha

- O que fazer se uma função possuir > 4 argumentos? Como fazer uso de mais registradores?
 - Convenção em MIPS é colocar os parâmetros excedentes na pilha logo acima do \$fp (frame pointer).
 - Procedimento/função espera que os primeiros quatro parâmetros estejam em \$a0 até \$a3 e o resto em memória.
 - Argumentos são endereçáveis através do \$fp.
 - Convenientes pois os offsets dos argumentos serão sempre iguais.

Alocação de Novos Dados na Pilha

- Exemplo 13:
 - Traduzir a seguinte função em C para assembly MIPS:

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

int main() {
    int res = leaf_example(10,20,30,40,50);
    return res;
}
```

Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5º. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 =

0xFFFFFFFF

\$sp →

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

stack



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$sp →

stack

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30    % Atribuição de a2
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1            % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40    % Atribuição de argumentos
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1            % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

0xFFFFFFFF

0x00000000

stack

\$sp →



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra

```

Resposta:

`$a0 = 10`

\$a | = 20

\$a2 = 30

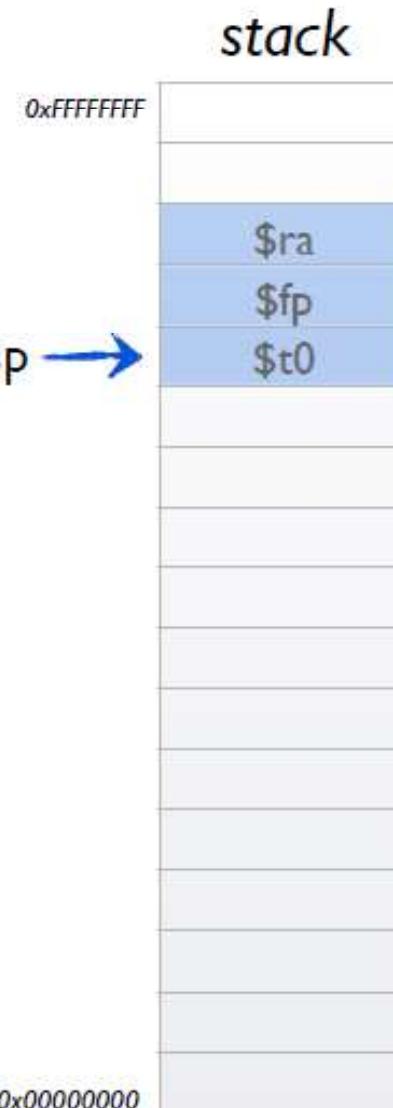
`$a3 = 40`

\$s0 =

$\$s | =$

$\S\$2 =$

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

0xFFFFFFFF

0x00000000

stack

\$ra

\$fp

\$t0



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

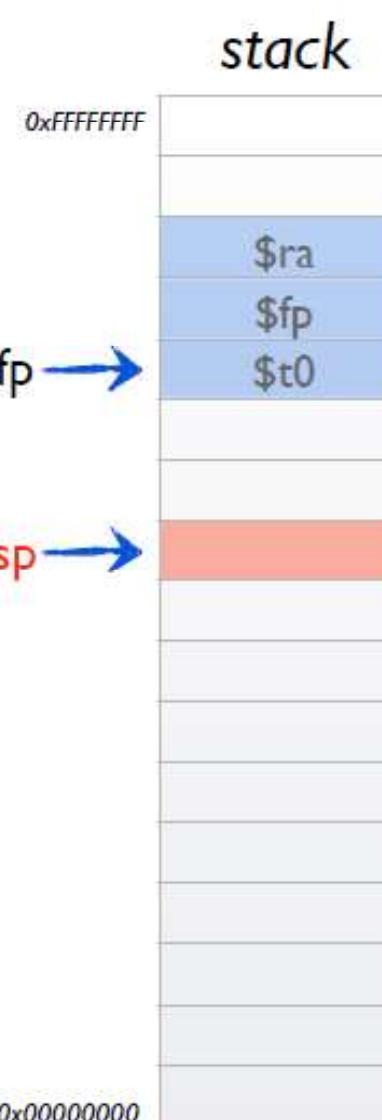
\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

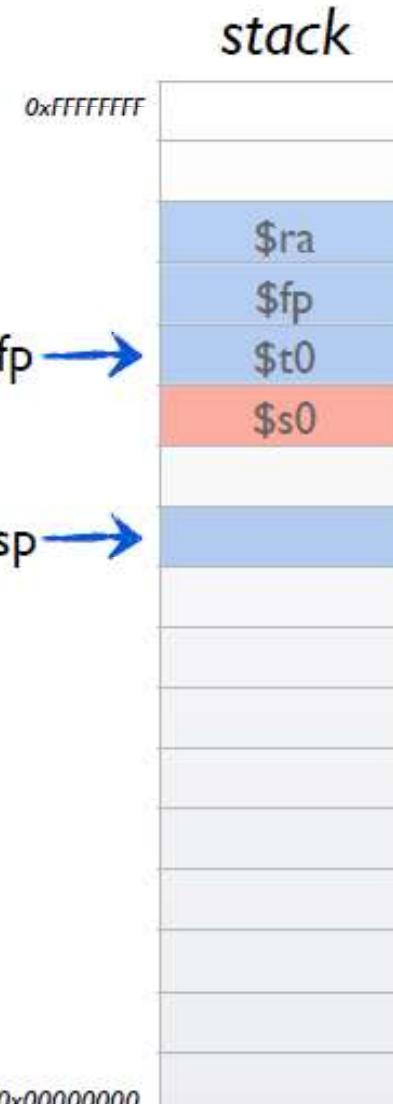
\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5º. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

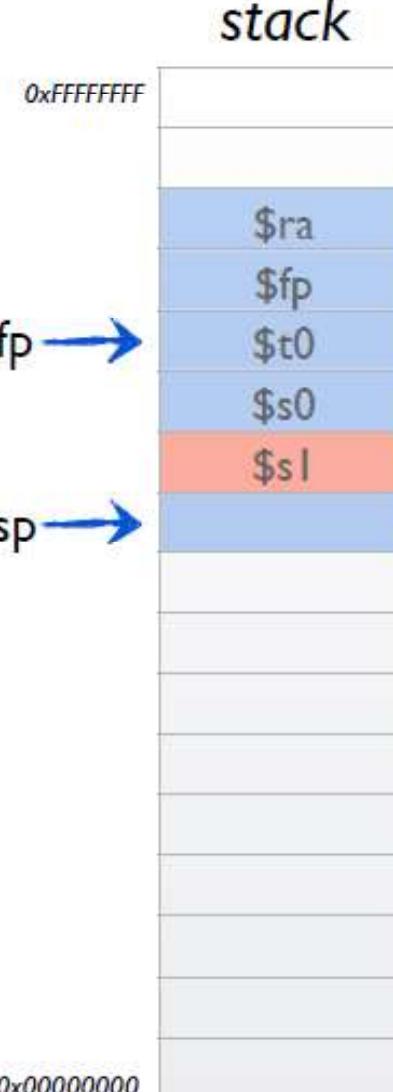
\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

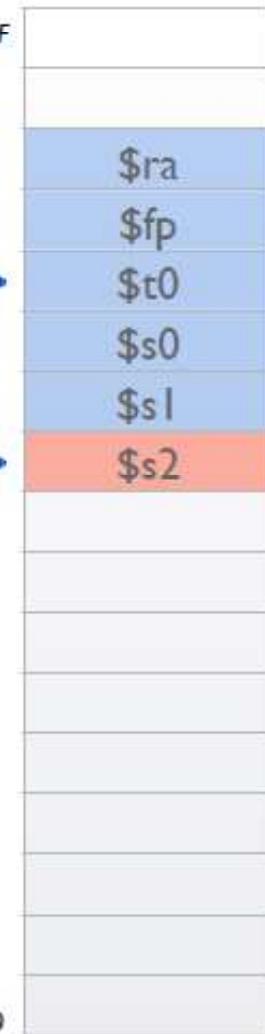
0xFFFFFFFF

\$fp →

\$sp →

0x00000000

stack



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)        % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 =

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)        % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

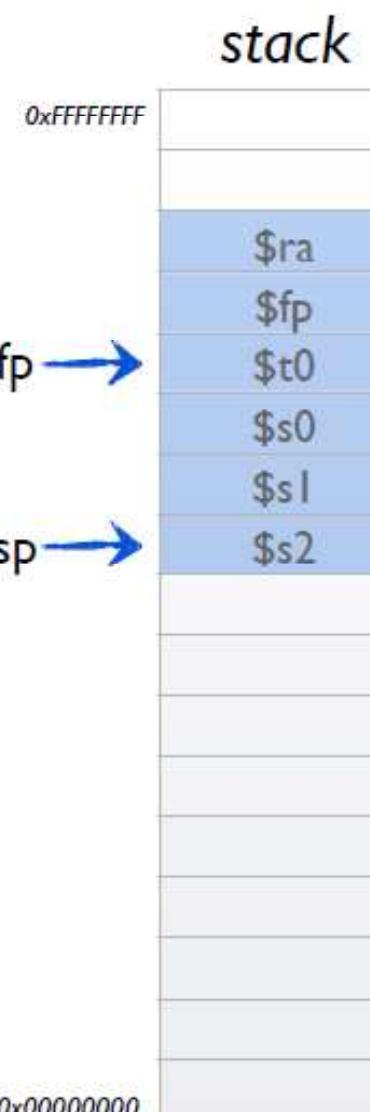
\$a3 = 40

\$s0 = 50

\$s1 = 30

\$s2 =

\$t0 = 50



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 = 30

\$s2 = 70

\$t0 = 50

0xFFFFFFFF

0x00000000

stack

\$ra

\$fp

\$t0

\$s0

\$s1

\$s2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1              % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 = -40

\$s2 = 70

\$t0 = 50

0xFFFFFFFF

0x00000000

stack

\$ra

\$fp

\$t0

\$s0

\$s1

\$s2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5º. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1             % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 = -90

\$s2 = 70

\$t0 = 50

0xFFFFFFFF

\$fp →

\$sp →

0x00000000

stack

\$ra

\$fp

\$t0

\$s0

\$s1

\$s2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1             % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 = -90

\$s2 = 70

\$t0 = 50

\$v0 = -90

stack

0xFFFFFFFF

\$ra

\$fp

\$t0

\$s0

\$s1

\$s2

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando $t0. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 = -90

\$s2 =

\$t0 = 50

\$v0 = -90

stack

0xFFFFFFFF

\$ra

\$fp

\$t0

\$s0

\$s1

\$s2

\$fp →

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 = 50

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1            % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra

```

Resposta:

`$a0 = 10`

$\$a = 20$

\$a2 = 30

$$\$a3 = 40$$

\$50 =

§§ | =

852

\$t0 = 50

\$y_0 = -90



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando $t0. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1            % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

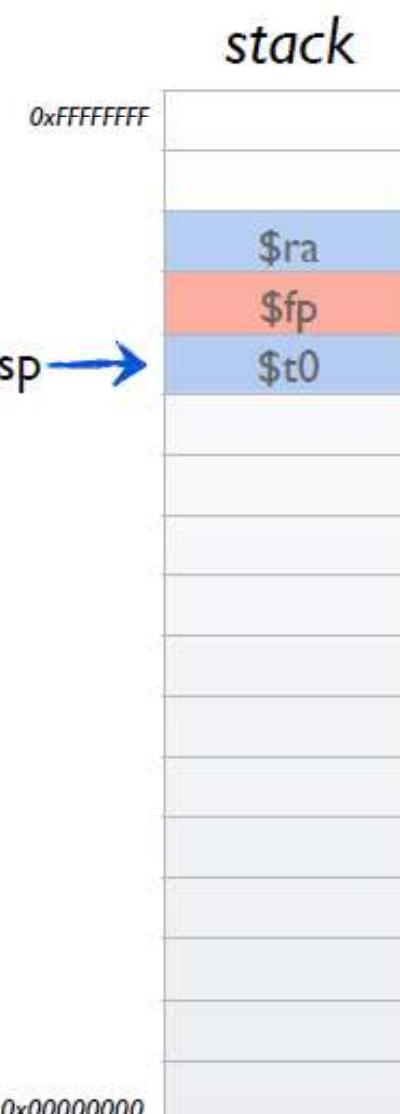
\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

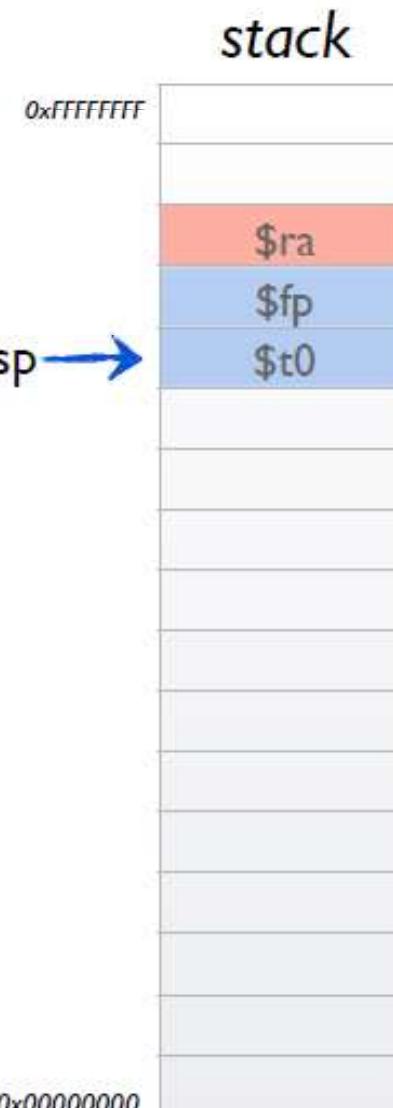
\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10      %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12        % Abrindo espaço na pilha
    sw $ra,8($sp)          % Salvando $ra
    sw $fp,4($sp)          % Salvando $fp
    sw $t0,0($sp)          % Salvando 5o. arg.
    move $fp,$sp            % $fp = $sp

    jal leaf_example        %chamada ao proced.

    lw $fp,4($sp)           % Restaurando $fp
    lw $ra,8($sp)           % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                  % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12        %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)           % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1             % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,10    %Atribuições
    addi $a1,$zero,20
    addi $a2,$zero,30
    addi $a3,$zero,40
    addi $t0,$zero,50

    subi $sp,$sp,12      % Abrindo espaço na pilha
    sw $ra,8($sp)        % Salvando $ra
    sw $fp,4($sp)        % Salvando $fp
    sw $t0,0($sp)        % Salvando 5o. arg.
    move $fp,$sp          % $fp = $sp

    jal leaf_example     %chamada ao proced.

    lw $fp,4($sp)         % Restaurando $fp
    lw $ra,8($sp)         % Restaurando $ra
    addi $sp,$sp,12

    jr $ra                % Finalizando programa

.globl leaf_example
leaf_example:
    subi $sp,$sp,12      %Abrindo espaço na pilha
    sw $s0,8($sp)
    sw $s1,4($sp)
    sw $s2,0($sp)

    lw $s0,0($fp)         % Fazendo os calculos
    add $s1,$a0,$a1
    add $s2,$a2,$a3
    sub $s1,$s1,$s2
    sub $s1,$s1,$s0

    move $v0,$s1           % Copiando valor de retorno
    lw $s2,0($sp)
    lw $s1,4($sp)
    lw $s0,8($sp)
    addi $sp,$sp,12
    jr $ra
```

Resposta:

\$a0 = 10

\$a1 = 20

\$a2 = 30

\$a3 = 40

\$s0 =

\$s1 =

\$s2 =

\$t0 = 50

\$v0 = -90



Alocação de Novos Dados na Pilha

- Importante: O que é e o que não é preservado entre chamadas?

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

Alocação de Novos Dados na Pilha

- Exercícios: Transforme o seguinte código em C para a linguagem Assembly MIPS:

```
int calculaDelta(int a, int b, int c) {
    return ((b*4) - 4*a*c);
}

int sqrt(int val) {
    return 1;
}

int calculaRaiz(int a, int b, int c, int c1, int c2, int c3) {
    return ((b*-1)+sqrt(calculaDelta(a,b,c)));
}

int main() {
    return calculaRaiz(1,2,3,4,5,6);
}
```

Funções recursivas

- Procedimentos/funções que não chamam outros procedimentos são conhecidos como folhas.
- Procedimentos podem também chamar outros procedimentos.
- Ainda, procedimentos podem invocar novas instâncias deles mesmos: Procedimentos/funções recursivos(as).
- Para tais casos faz-se necessário tomar certos cuidados quando invocando novos procedimentos/funções.

Funções recursivas

- Exemplo 14: Suponha que a função main de um dado programa invoque a função A com o argumento de valor 3 passado no registrador \$a0 através de jal A. Suponha ainda que o procedimento A chame o procedimento B passando o argumento de valor 7 no mesmo registrador (\$a0) através de jal B.
 - Problema 1: Função A ainda não terminou sua execução e o mesmo registrador \$a0 é utilizado também por B.
 - Problema 2: Conflito no endereço de retorno em \$ra, uma vez que agora o endereço de retorno é o de B (e não mais o de A).
 - Como prevenir tais problemas?

Funções recursivas

- Problema 1: Função A ainda não terminou sua execução e o mesmo registrador \$a0 é utilizado também por B.
- Problema 2: Conflito no endereço de retorno em \$ra, uma vez que agora o endereço de retorno é o de B (e não mais o de A).
- Como prevenir tais problemas?

Solução

- Colocar na pilha os registradores que necessitam ser preservados para posterior uso após a chamada da função/procedimento.
- No retorno, os registradores previamente salvos necessitam ser restaurados da memória.

Funções recursivas

- Exemplo 15:
 - Traduzir a função a seguir para a linguagem MIPS Assembly:

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

int main()
{
    int a = 2;
    return fact(a);
}
```

Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 =

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =

stack

0xFFFFFFFF

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =

stack

0xFFFFFFFF

\$sp →

0x0000000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        #salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq   $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)          #restaura arg.
    lw    $fp, 4($sp)          #restaura fp
    lw    $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul   $v0,$a0,$v0           #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =

stack

0xFFFFFFFF

\$ra

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp) // $fp = 0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        #salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi    $v0,$zero,1       #return 1
    addi    $sp,$sp,12        #retira 3 itens pilha
    jr     $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$fp,\$sp →

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq   $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)        #restaura arg.
    lw    $fp, 4($sp)        #restaura fp
    lw    $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr      $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr      $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$fp →

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi    $v0,$zero,1       #return 1
    addi    $sp,$sp,12        #retira 3 itens pilha
    jr      $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr      $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 2

\$a | =

\$a2 =

\$a3 =

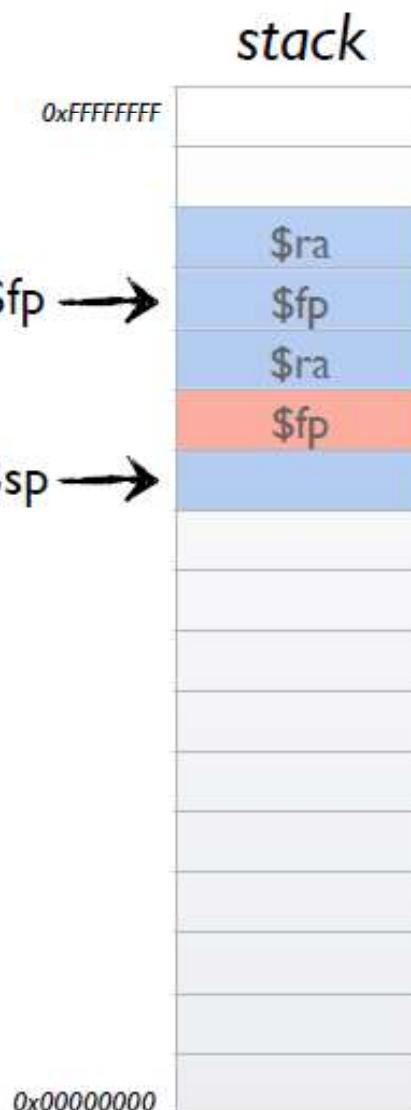
\$s0 =

$\$s | =$

§s2 =

\$t0 =

\$y_0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

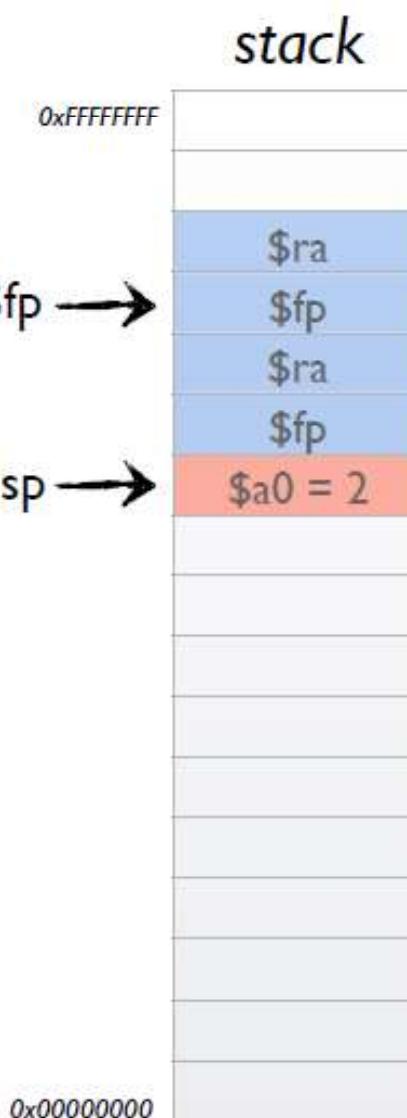
\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        $salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

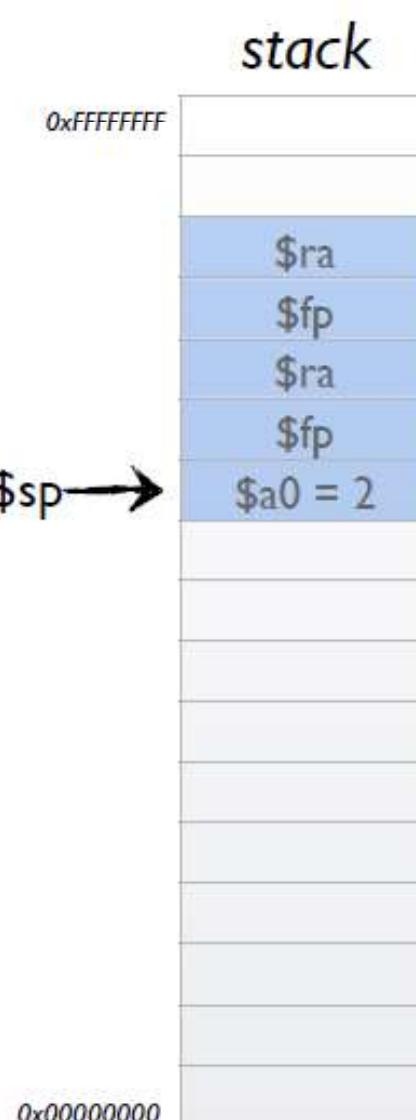
\$s0 =

\$s1 =

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq   $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)        #restaura arg.
    lw    $fp, 4($sp)        #restaura fp
    lw    $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

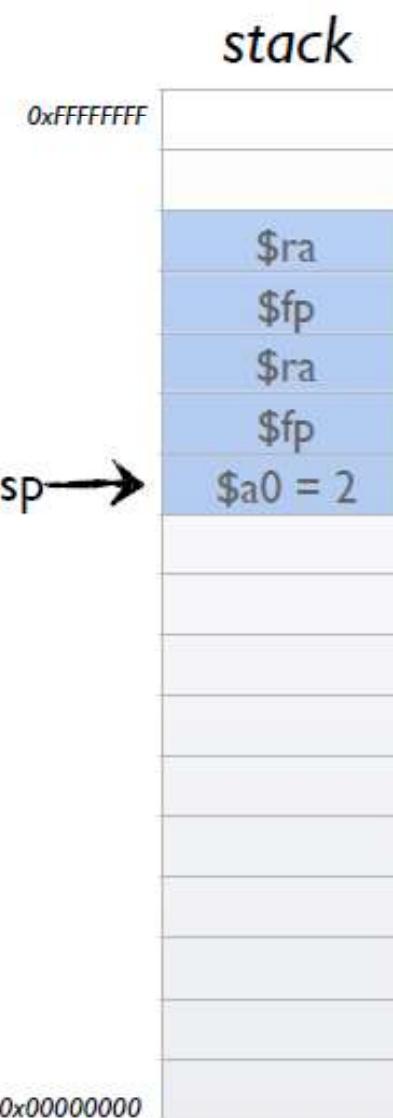
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        $salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)          #restaura arg.
    lw    $fp, 4($sp)          #restaura fp
    lw    $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

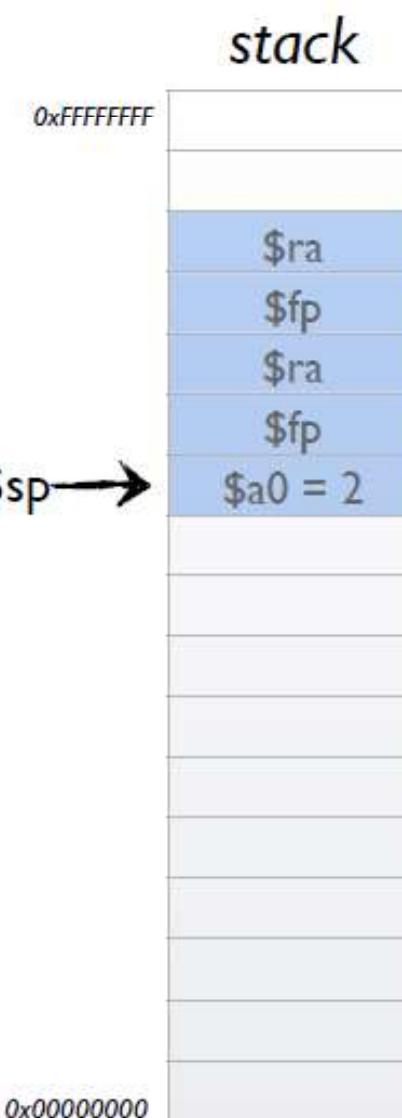
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi    $v0,$zero,1        #return 1
    addi    $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)       #restaura arg.
    lw      $fp, 4($sp)       #restaura fp
    lw      $ra, 8($sp)       #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = |

\$a1 =

\$a2 =

\$a3 =

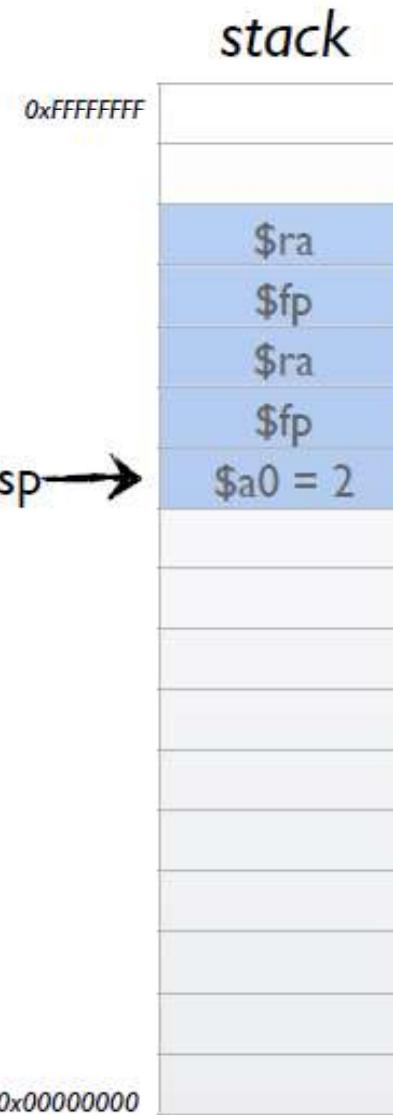
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr      $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal      fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr      $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

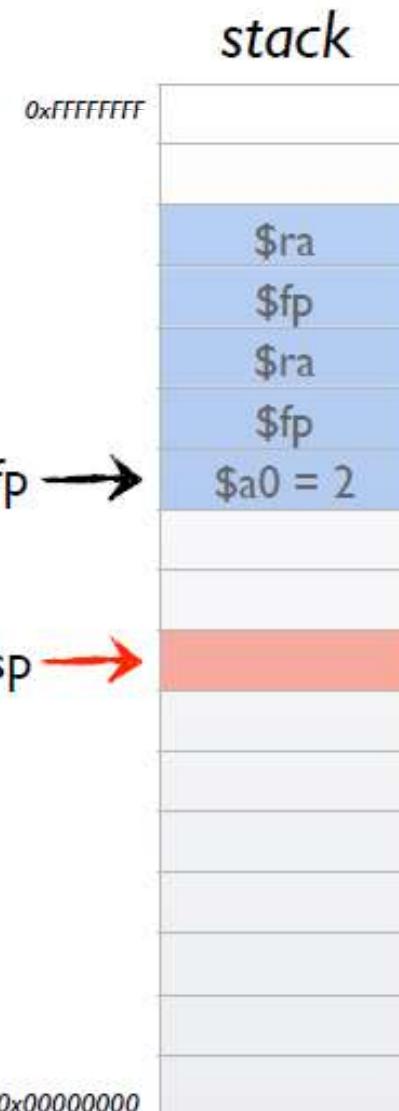
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

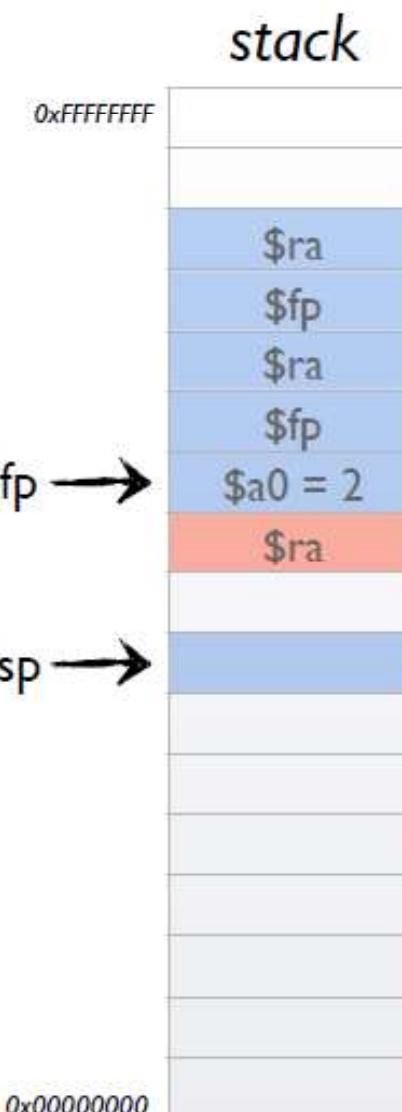
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw     $ra, 8($sp)      #salvo end. retorno
    sw     $fp, 4($sp)      $salvo fp
    sw     $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1     #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)        #restaura arg.
    lw    $fp, 4($sp)        #restaura fp
    lw    $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

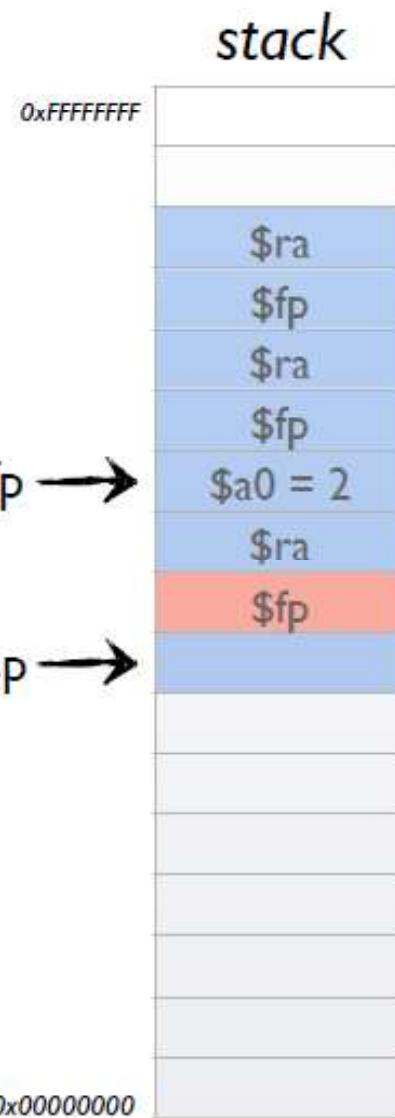
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi    $v0,$zero,1          #return 1
    addi    $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi    $a0,$a0,-1          #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)       #restaura arg.
    lw      $fp, 4($sp)       #restaura fp
    lw      $ra, 8($sp)       #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 1

\$a | =

\$a2 =

\$a3 =

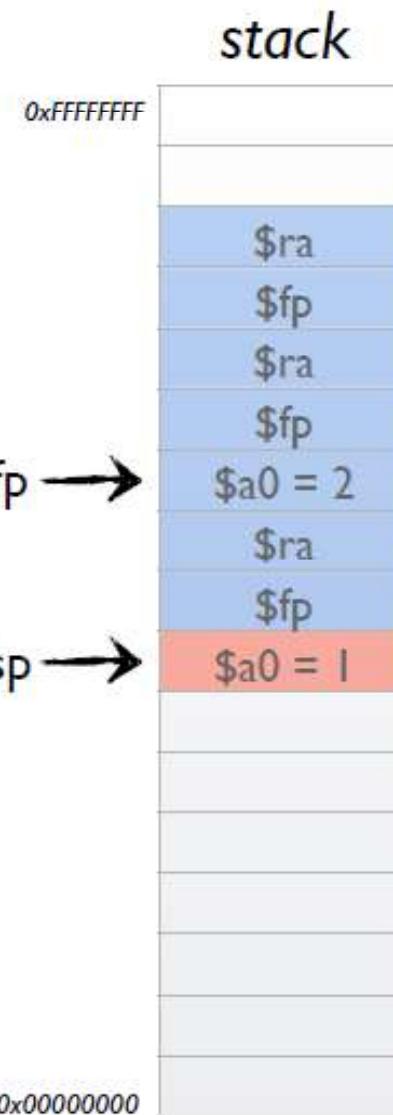
\$s0 =

$\$s| =$

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

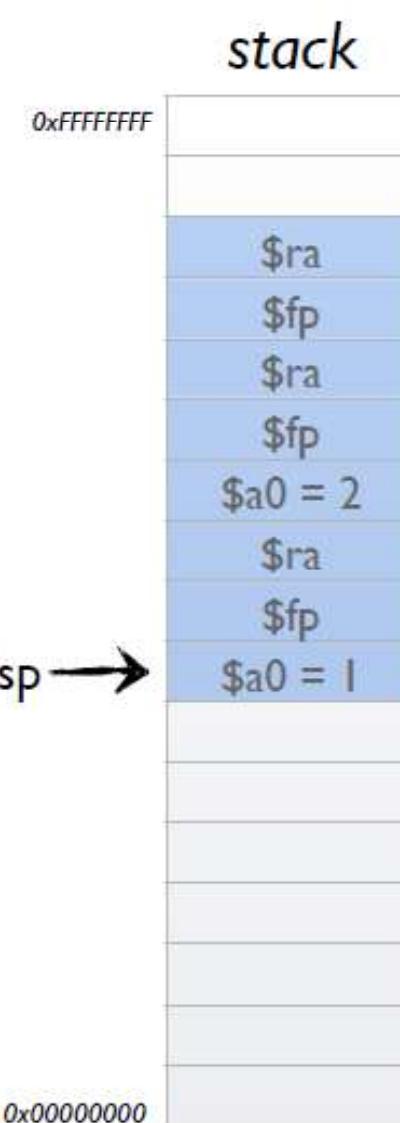
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq   $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1       #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

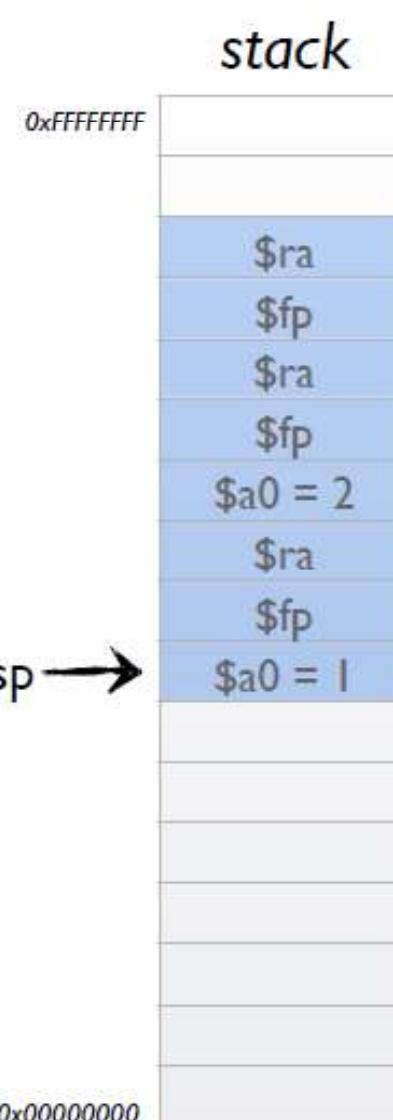
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq   $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)       #restaura arg.
    lw     $fp, 4($sp)       #restaura fp
    lw     $ra, 8($sp)       #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

0x00000000

\$fp,\$sp →



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12          #return n*fact(n-1)
    mul    $v0,$a0,$v0
    jr     $ra

```

Resposta:

`$a0 = 0`

\$a | =

\$a2 =

\$a3 =

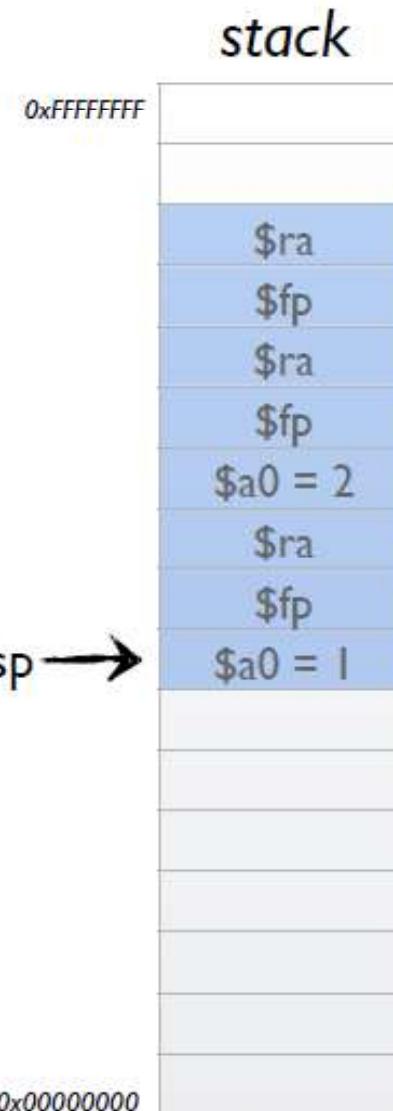
\$s0 =

$\$S| =$

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi $sp, $sp, -12    #espaço na pilha
    sw $ra, 8($sp)      #salvo end. retorno
    sw $fp, 4($sp)      $salvo fp
    sw $a0, 0($sp)      #salvo argumento
    move $fp,$sp
    slti $t0,$a0,1        #testa se n < 1
    beq $t0,$zero,L1      #se n>=1, vai p/ L1
    addi $v0,$zero,1       #return 1
    addi $sp,$sp,12        #retira 3 itens pilha
    jr $ra

L1:
    addi $a0,$a0,-1        #n >= 1: --argumento
    jal fact
    lw $a0, 0($sp)      #restaura arg.
    lw $fp, 4($sp)      #restaura fp
    lw $ra, 8($sp)      #restaura end. ret.
    addi $sp,$sp,12
    mul $v0,$a0,$v0         #return n*fact(n-1)
    jr $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

\$fp →

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     $salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi    $v0,$zero,1       #return 1
    addi    $sp,$sp,12        #retira 3 itens pilha
    jr      $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr      $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

\$ra

\$fp →

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

\$ra

\$fp

\$fp →

\$sp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

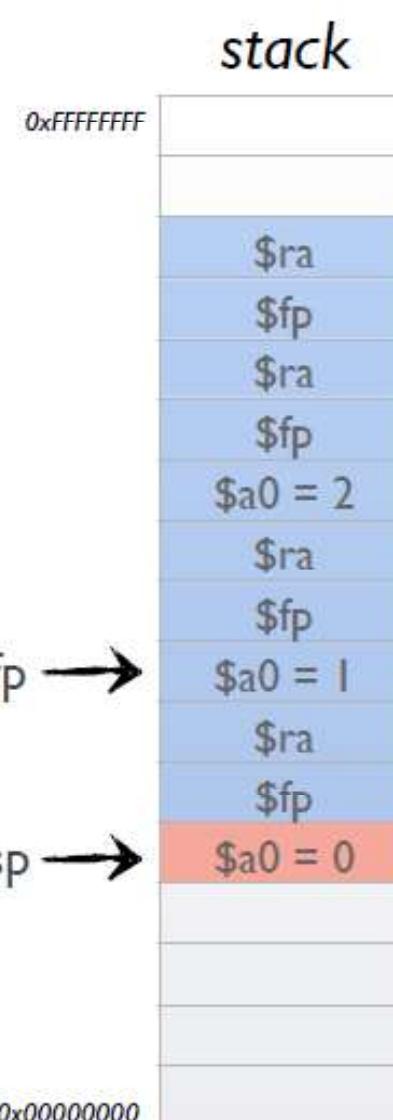
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12        #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

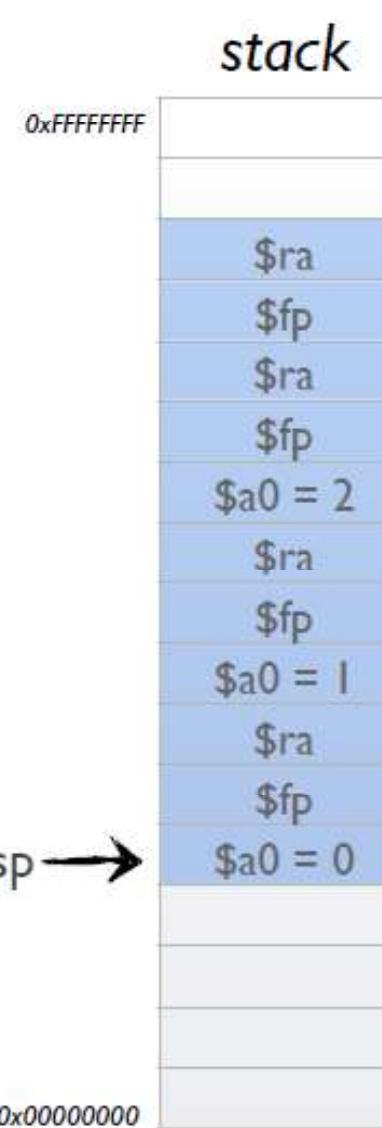
\$s0 =

\$s1 =

\$s2 =

\$t0 = 0

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      #salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1        #testa se n < 1
    beq   $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

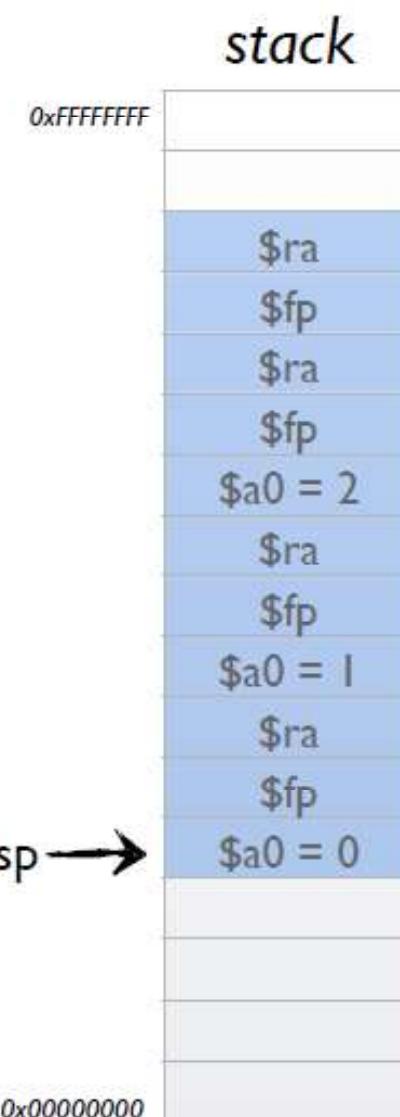
\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)      #salvo end. retorno
    sw      $fp, 4($sp)      $salvo fp
    sw      $a0, 0($sp)      #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1        #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

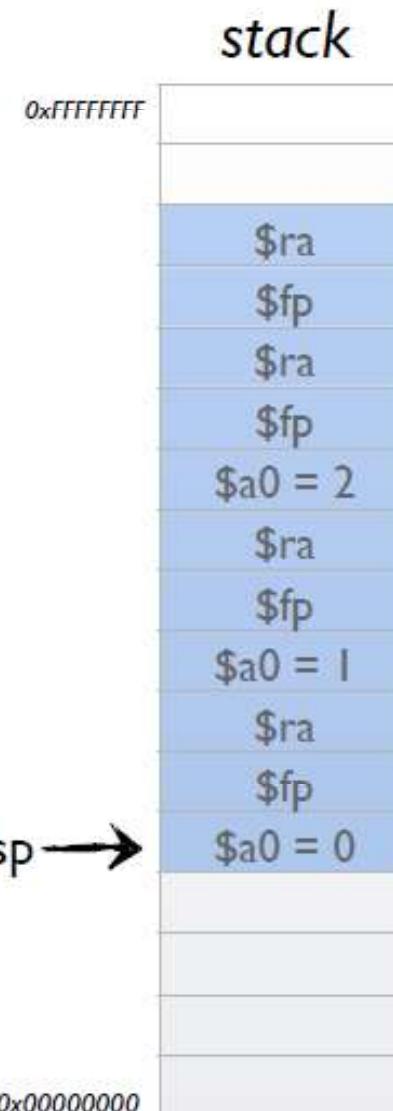
\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 =



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)         #restaura arg.
    lw     $fp, 4($sp)         #restaura fp
    lw     $ra, 8($sp)         #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 0

\$a | =

\$a2 =

\$a3 =

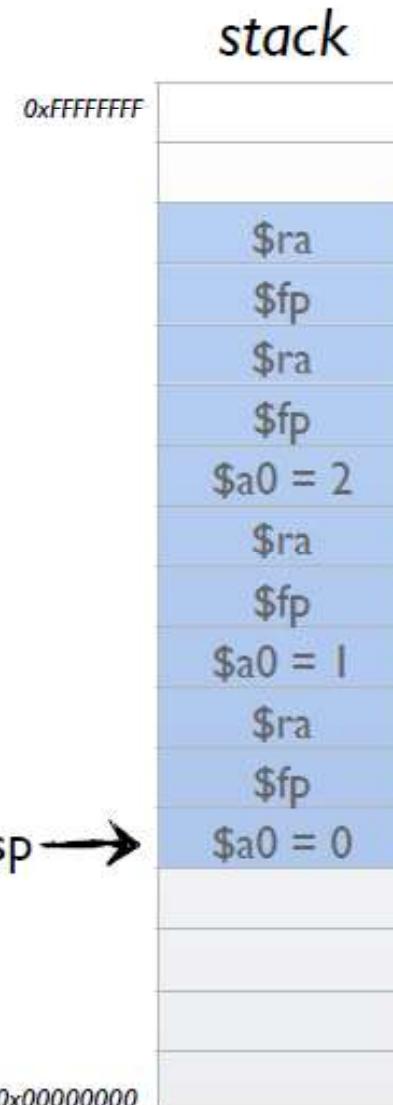
\$s0 =

\$S| =

§§2 =

\$t0 =

\$y0 =



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0           #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 0

\$a1 =

\$a2 =

\$a3 =

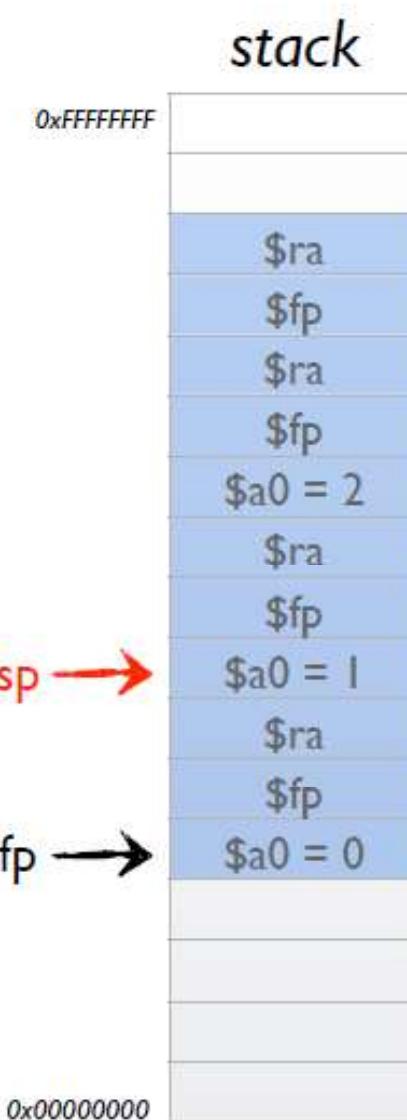
\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = |



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)         #restaura arg.
    lw     $fp, 4($sp)         #restaura fp
    lw     $ra, 8($sp)         #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr    $ra

```

Resposta:

\$a0 = 0

\$a | =

\$a2 =

\$a3 =

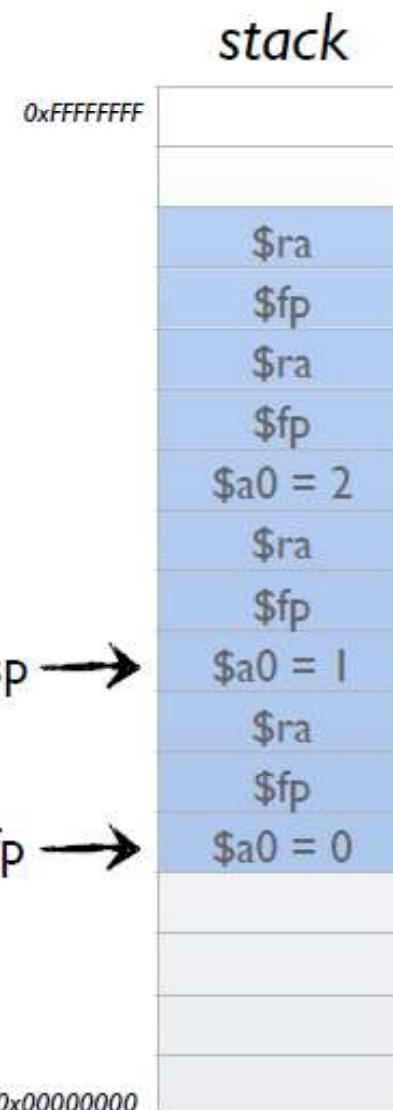
\$s0 =

$\$s | =$

\$s2 =

\$t0 =

\$v0 =



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)         #restaura arg.
    lw     $fp, 4($sp)         #restaura fp
    lw     $ra, 8($sp)         #restaura end. ret.
    addi  $sp,$sp,12
    mul   $v0,$a0,$v0           #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = |

\$a | =

\$a2 =

\$a3 =

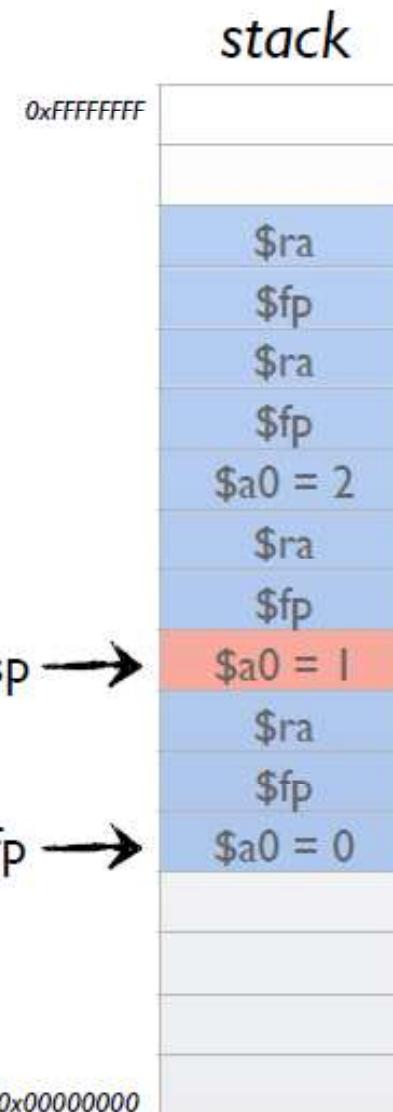
\$s0 =

$\$s| =$

§ 57 =

\$t0 = 1

\$v0 = |



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1         #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0        #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 1

\$a1 =

\$a2 =

\$a3 =

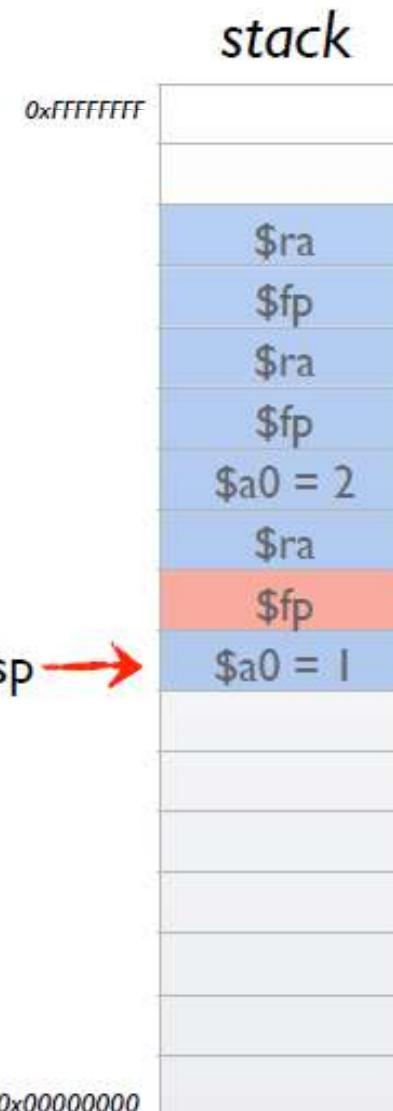
\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 1



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi    $v0,$zero,1        #return 1
    addi    $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)       #restaura arg.
    lw      $fp, 4($sp)       #restaura fp
    lw      $ra, 8($sp)       #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = |

\$a1 =

\$a2 =

\$a3 =

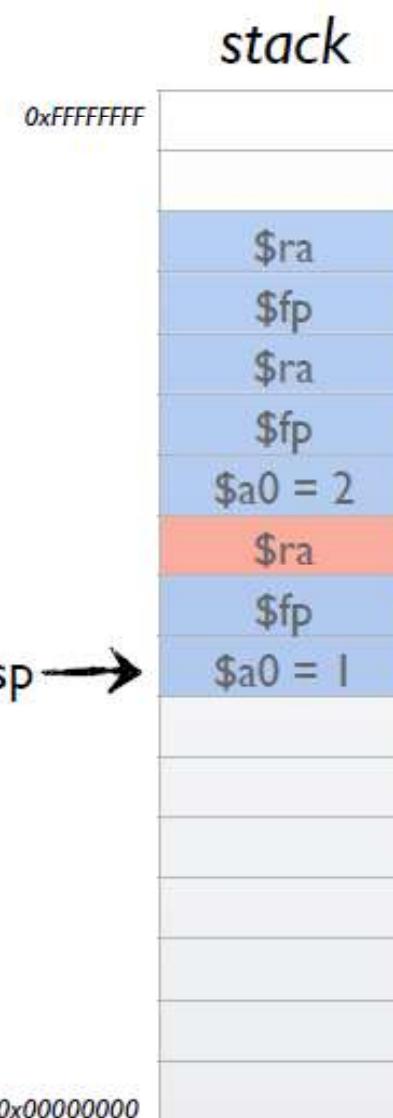
\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = |



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq   $t0,$zero,L1         #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12           #return
    mul    $v0,$a0,$v0           #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = |

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = |

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$ra

\$fp

\$a0 = 1

\$sp →

\$fp →

0x00000000



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)         #restaura arg.
    lw     $fp, 4($sp)         #restaura fp
    lw     $ra, 8($sp)         #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 1

\$a | =

\$a2 =

\$a3 =

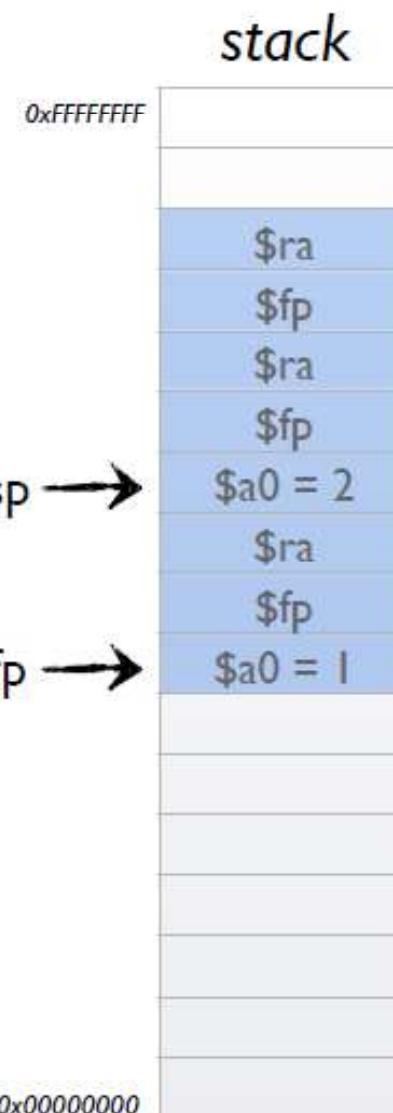
\$s0 =

$\$s| =$

§52 =

\$t0 = 1

`$v0 =` |



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1         #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi  $sp,$sp,12
    mul   $v0,$a0,$v0            #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 1

\$a | =

\$a2 =

\$a3 =

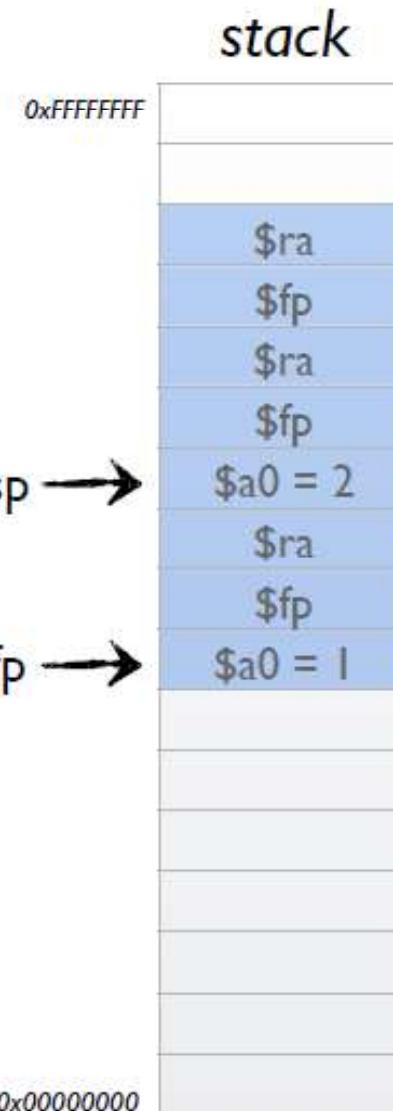
\$s0 =

$\$s | =$

\$s2 =

\$t0 =

\$v0 = |



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        $salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

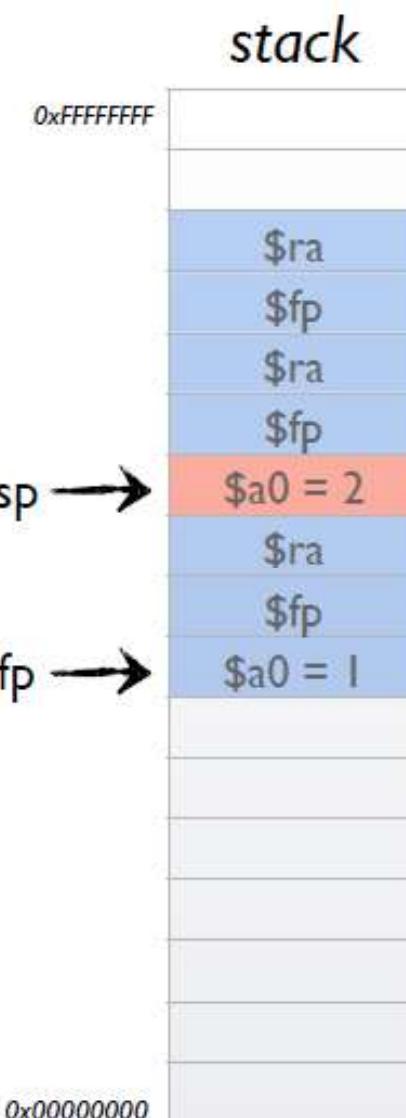
\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 1



Suporte a Procedimentos e Funções

```

.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1           #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)         #restaura arg.
    lw     $fp, 4($sp)         #restaura fp
    lw     $ra, 8($sp)         #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra

```

Resposta:

\$a0 = 2

\$a | =

\$a2 =

\$a3 =

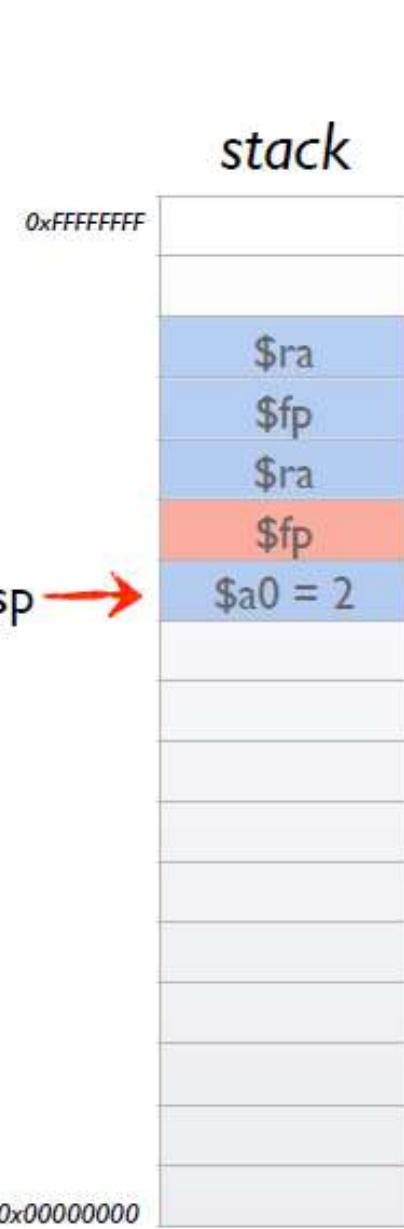
\$s0 =

$\$s| =$

§52 =

\$t0 =

\$v0 = 1



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12    #espaço na pilha
    sw      $ra, 8($sp)     #salvo end. retorno
    sw      $fp, 4($sp)     #$salvo fp
    sw      $a0, 0($sp)     #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1        #testa se n < 1
    beq    $t0,$zero,L1      #se n>=1, vai p/ L1
    addi    $v0,$zero,1       #return 1
    addi    $sp,$sp,12        #retira 3 itens pilha
    jr      $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)     #restaura arg.
    lw      $fp, 4($sp)     #restaura fp
    lw      $ra, 8($sp)     #restaura end. ret.
    addi    $sp,$sp,12
    mul     $v0,$a0,$v0       #return n*fact(n-1)
    jr      $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

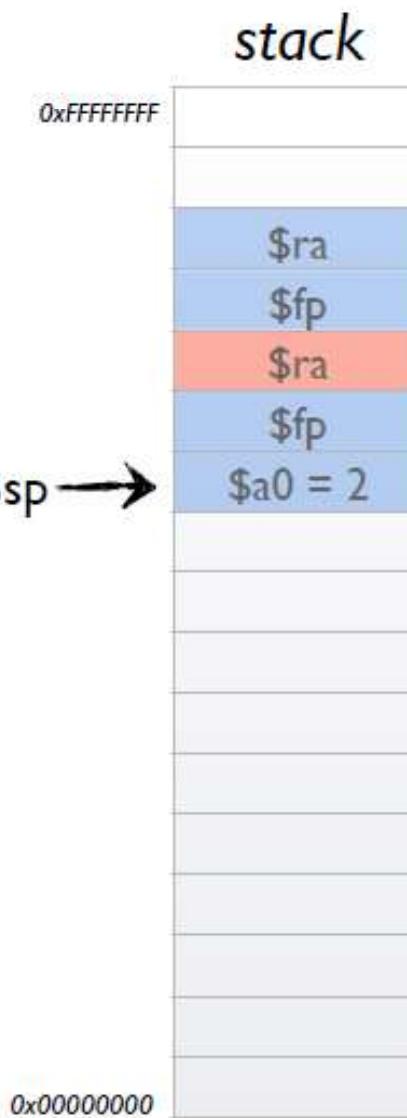
\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = |



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        #salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)          #restaura arg.
    lw    $fp, 4($sp)          #restaura fp
    lw    $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12          #return n*fact(n-1)
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = |

stack

0xFFFFFFFF

\$ra

\$fp

\$ra

\$fp

\$a0 = 2

\$sp →

\$fp →

0x00000000



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        #salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

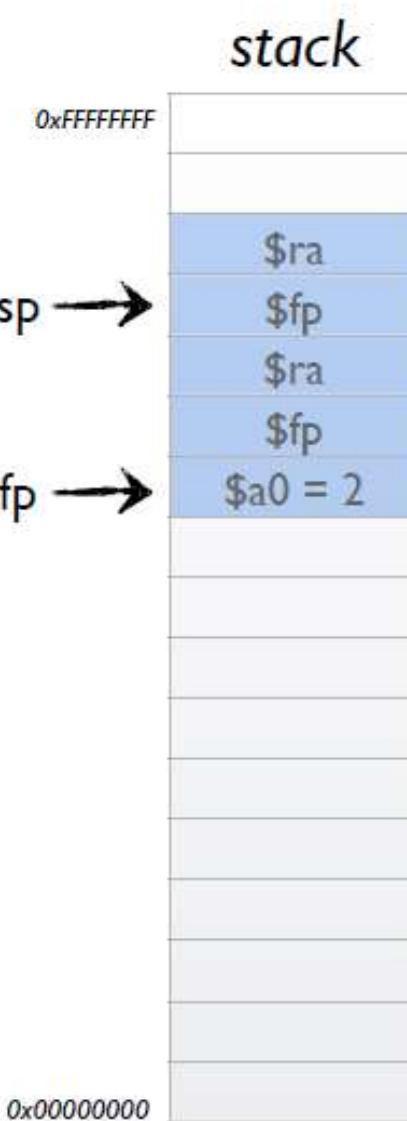
\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi    $v0,$zero,1        #return 1
    addi    $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi    $a0,$a0,-1        #n >= 1: --argumento
    jal     fact
    lw      $a0, 0($sp)       #restaura arg.
    lw      $fp, 4($sp)       #restaura fp
    lw      $ra, 8($sp)       #restaura end. ret.
    addi    $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

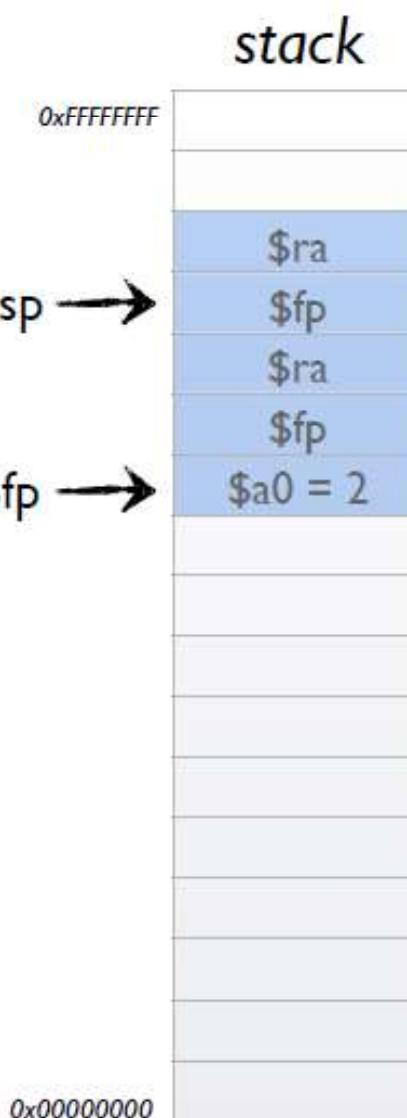
\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp) lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)       #salvo end. retorno
    sw      $fp, 4($sp)       $salvo fp
    sw      $a0, 0($sp)       #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1         #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1        #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1         #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$fp,\$sp →

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        $salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)        #restaura arg.
    lw     $fp, 4($sp)        #restaura fp
    lw     $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$fp,\$sp →

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2

0xFFFFFFFF

0x00000000

stack

\$ra

\$fp



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        $salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12         #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)        #restaura arg.
    lw    $fp, 4($sp)        #restaura fp
    lw    $ra, 8($sp)        #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw      $ra, 8($sp)        #salvo end. retorno
    sw      $fp, 4($sp)        $salvo fp
    sw      $a0, 0($sp)        #salvo argumento
    move    $fp,$sp
    slti    $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1        #se n>=1, vai p/ L1
    addi   $v0,$zero,1          #return 1
    addi   $sp,$sp,12           #retira 3 itens pilha
    jr     $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw     $a0, 0($sp)          #restaura arg.
    lw     $fp, 4($sp)          #restaura fp
    lw     $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr     $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = 1

\$v0 = 2



Suporte a Procedimentos e Funções

```
.globl main
main:
    addi $a0,$zero,2
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $fp,0($sp)
    move $fp,$sp
    jal fact
    lw $fp,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

.globl fact
fact:
    addi    $sp, $sp, -12      #espaço na pilha
    sw     $ra, 8($sp)        #salvo end. retorno
    sw     $fp, 4($sp)        #salvo fp
    sw     $a0, 0($sp)        #salvo argumento
    move   $fp,$sp
    slti   $t0,$a0,1          #testa se n < 1
    beq    $t0,$zero,L1       #se n>=1, vai p/ L1
    addi   $v0,$zero,1         #return 1
    addi   $sp,$sp,12          #retira 3 itens pilha
    jr    $ra

L1:
    addi   $a0,$a0,-1          #n >= 1: --argumento
    jal    fact
    lw    $a0, 0($sp)          #restaura arg.
    lw    $fp, 4($sp)          #restaura fp
    lw    $ra, 8($sp)          #restaura end. ret.
    addi   $sp,$sp,12
    mul    $v0,$a0,$v0          #return n*fact(n-1)
    jr    $ra
```

Resposta:

\$a0 = 2

\$a1 =

\$a2 =

\$a3 =

\$s0 =

\$s1 =

\$s2 =

\$t0 = |

\$v0 = 2

stack

0xFFFFFFFF

\$sp →

0x00000000



Introdução

- Diferença entre as instruções compreendidas por humanos e computadores.
- Instruções são mantidas e processadas pelo computador como uma série de sinais eletrônicos (altos e baixos) - código binário.
- Cada pedaço da instrução pode ser considerado individualmente como um número.
- A concatenação dos números forma a instrução como um todo.

Introdução

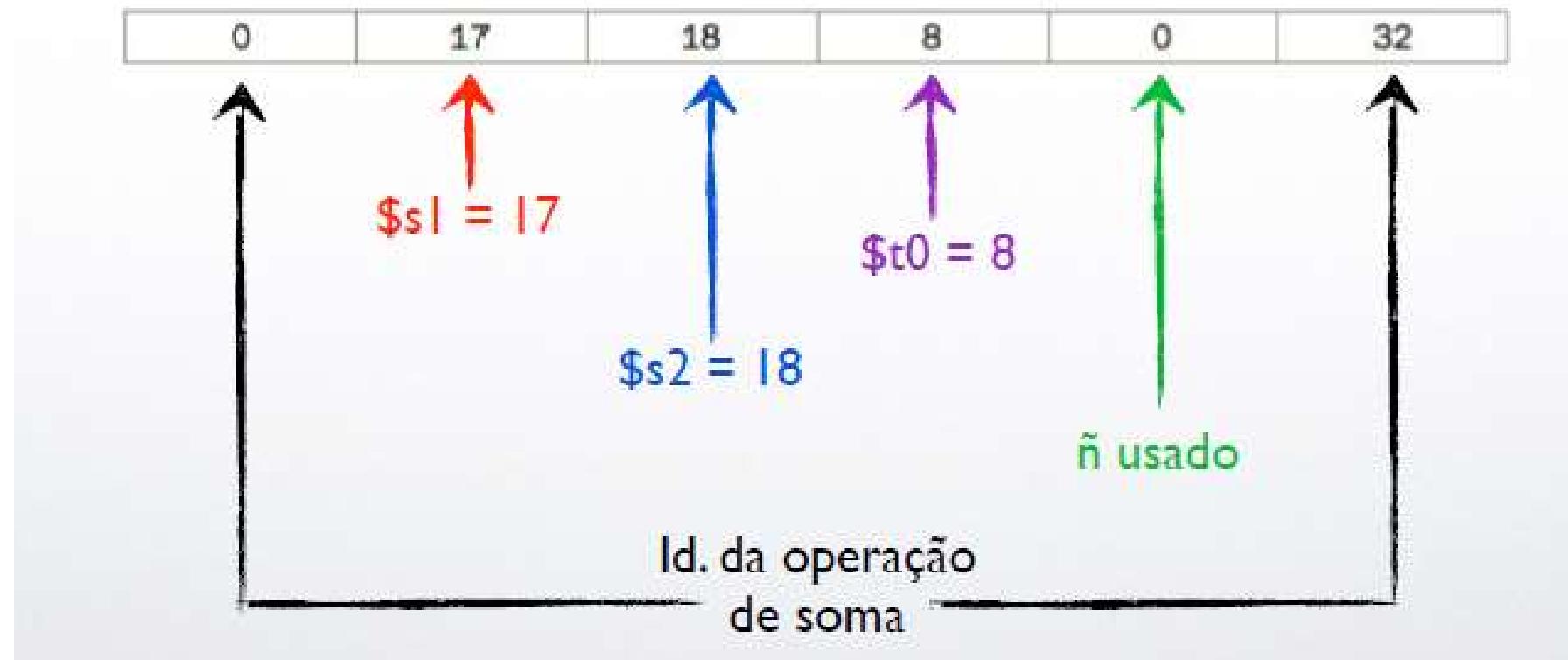
- Importante: Registradores também são mapeados em números.
 - Em MIPS, os registradores \$s0 a \$s7 são mapeados nos números 16 a 23.
 - Em MIPS, os registradores \$t0 a \$t7 são mapeados nos números 8 a 15.
- Exemplo: Traduzir para a linguagem real do MIPS a instrução:

add \$t0, \$t1, \$t2

Representação de Instruções no Computador

Introdução

- Representação decimal



Representação de Instruções no Computador

Introdução

- Cada um dos segmentos é chamado de "campo" (field)
- Representação binária

add \$t0,\$s1,\$s2

000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- Todas as instruções em MIPS possuem tamanho de 32 bits
- O layout da instrução é chamado de formato da instrução

Representação de Instruções no Computador

Introdução

- Cada campo do formato de instrução possui um nome específico.



- op: Operação básica da instrução. Denota a operação a ser executada pela instrução.
- rs: Primeiro operando-fonte.
- rt: Segundo operando-fonte.
- rd: Registrador de destino da operação
- shamt: shift amount. Utilizado em operações de shift de bits
- funct: Seleciona uma variante específica da operação declarada no campo op

Introdução

- Exemplo anterior funciona muito bem para instruções aritméticas.
Mas e quanto às outras?
- Por exemplo, um campo com 5 bits pode representar um máximo de 32 valores.
- O compromisso encontrado pelos projetistas do MIPS foi manter todas as instruções com um mesmo tamanho.
- Para atender a todos os tipos de instrução, vários formatos foram criados.

Representação de Instruções no Computador

Introdução

- Formato R (R-type): Exemplo anterior.



- Formato I (I-type): Aplicado a instruções de transferência de dados e operações com valores imediatos:



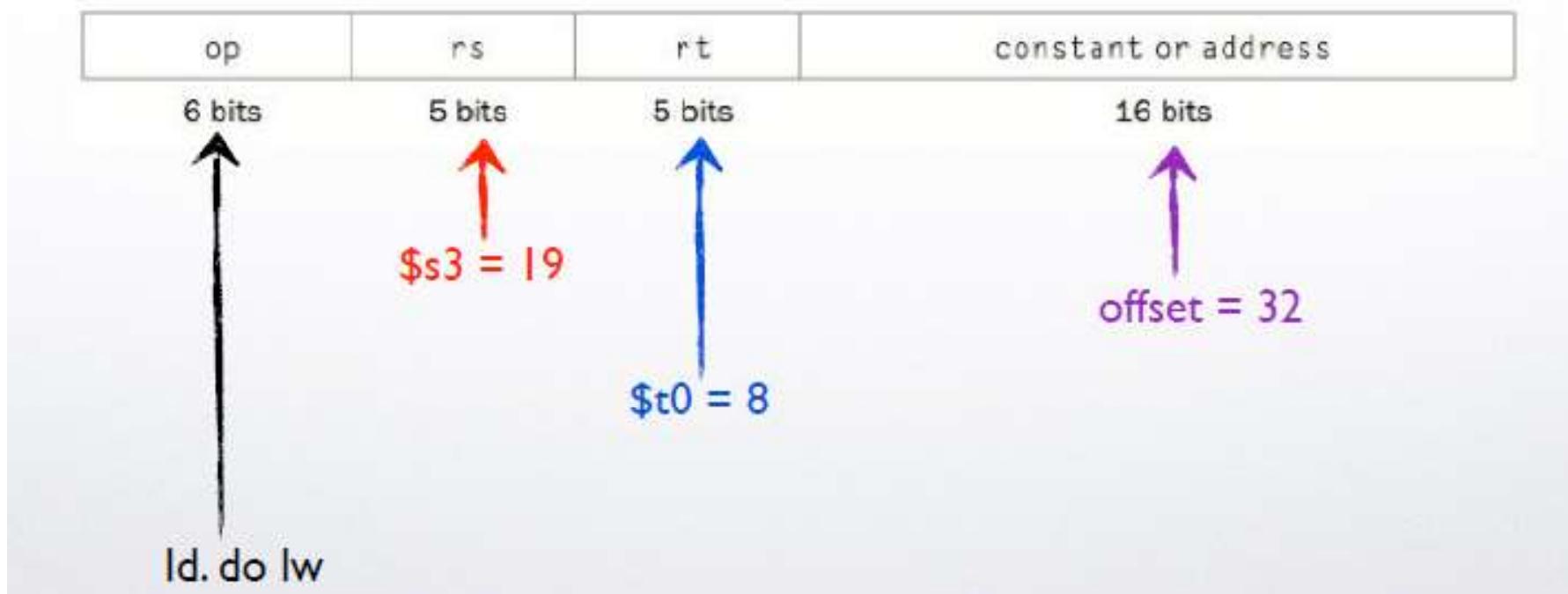
- Com um campo de endereço maior (16 bits) é possível se endereçar uma quantidade maior de valores.

Representação de Instruções no Computador

Introdução

- Exemplo:

lw \$t0,32(\$s3) # Temporary reg \$t0 gets A[8]



Introdução

- Complexidade é reduzida através da manutenção dos formatos similares uns aos outros.
- Formatos são diferenciados uns dos outros à partir dos valores presentes no primeiro campo (op).
- Restante da informação pode ser parseada com base no primeiro campo.

Representação de Instruções no Computador

Introdução

- Exemplos dos possíveis formatos para algumas instruções:

Instruction	Format	op	rs	rt	rd	shamt	funct	address
add	R	0	reg	reg	reg	0	32_{ten}	n.a.
sub (subtract)	R	0	reg	reg	reg	0	34_{ten}	n.a.
add immediate	I	8_{ten}	reg	reg	n.a.	n.a.	n.a.	constant
lw (load word)	I	35_{ten}	reg	reg	n.a.	n.a.	n.a.	address
sw (store word)	I	43_{ten}	reg	reg	n.a.	n.a.	n.a.	address

Introdução

- Exemplo 16:

- Traduzir as seguintes instruções em linguagem C / MIPS assembly para linguagem de máquina:

- Considerando \$t1 como endereço base de A[] e h como \$s2

$A[300] = h + A[300];$

- Equivale a:

`lw $t0, 1200, ($t1) # Temporary reg $t0 gets A[300]`

`add $t0, $s2, $t0 # Temporary reg $t0 gets h + A[300]`

`aw $t0, 1200, ($t1) # Stores h + A[300] back into A[300]`

Introdução

- Exemplo 16:

`lw $t0, 1200, ($t1) # Temporary reg $t0 gets A[300]`

`add $t0, $s2, $t0 # Temporary reg $t0 gets h + A[300]`

`aw $t0, 1200, ($t1) # Stores h + A[300] back into A[300]`

- Traduzindo

- op de lw = 35, reg.\$t1 = 9, reg.\$t0 = 8, offset = $300 \times 4 = 1200$
- op de add = 35, reg.\$t1 = 9, reg.\$t0 = 8, offset = $300 \times 4 = 1200$
- op de sw = 43

op	rs	rt	rd	address/ shamt	funct
35	9	8		1200	
0	18	8	8	0	32
43	9	8		1200	

Introdução

- Exemplo 16:

- Equivalente em binário

100011	01001	01000	0000 0100 1011 0000
000000	10010	01000	01000 00000 100000
101011	01001	01000	0000 0100 1011 0000

- Diferença entre as instruções lw e sw é 1 bit (em destaque).
 - Similaridade entre as instruções facilitam o projeto de hardware.

Representação de Instruções no Computador

Introdução

- Formatos de instruções do tipo R e I estão relacionados às instruções aritméticas e memória.
- E as instruções de desvio condicional? (beq, bne, etc)
- Formato de instruções I também se aplicam as operações de desvio condicional:



Representação de Instruções no Computador

Introdução



- $\text{opcode} = 4$.
- Campo de deslocamento utilizado para calcular o endereço alvo.
- Se conteúdo do registrador cujo endereço está no campo **rs** for igual ao conteúdo do registrador cujo endereço está em **rt**, então salta para a posição designada.

Representação de Instruções no Computador

Introdução

- Existe as instruções de desvio incondicional? (j, jr, etc)



- opcode = 2.
- Campo de endereço utilizado para calcular o endereço-alvo.
- Salto para o endereço descrito no último campo.
- Chamadas de instruções do tipo J.

Endereçamento MIPS para imediatos de 32-bits e endereços

- Manter as instruções em tamanho de 32-bits simplifica o projeto de hardware.
- Porém, algumas vezes é conveniente ter uma constante ou endereço em 32-bits.
- Frequentemente endereços e campos imediatos podem ser endereçados utilizando-se um espaço de 16-bits.
- Mas... E quando são maiores?

Endereçamento MIPS para imediatos de 32-bits e endereços

- Conjunto de instruções MIPS inclui a instrução lui (load upper immediate).
 - Permite que os 16-bits mais significativos sejam carregados em um registrador; e
 - Instrução subsequente carrega os 16-bits menos significativos da constante.
- Exemplo 17:
 - Qual é o código assembly MIPS para carregar a seguinte constante (de 32-bits) em um registrador \$s0?

Endereçamento MIPS para imediatos de 32-bits e endereços

- Exemplo 17:

- Qual é o código assembly MIPS para carregar a seguinte constante (de 32-bits) em um registrador \$s0?

```
0000 0000 0011 1101 0000 1001 0000 0000
```

- Passo 1: Carregar os primeiros 16 bits (61 em decimal):

```
0000 0000 0011 1101 0000 1001 0000 0000
```

```
lui $s0, 61    # 61 decimal = 0000 0000 0011 1101 binary
```

- Valor do registrador \$s0 após o comando acima:

```
0000 0000 0011 1101 0000 0000 0000 0000
```

Representação de Instruções no Computador

Endereçamento MIPS para imediatos de 32-bits e endereços

- Exemplo 17:

- Passo 2: Inserir os 16 bits restantes (2304 em decimal):

```
0000 0000 0011 1101 0000 1001 0000 0000
```

```
ori $s0, $s0, 2304 # 2304 decimal = 0000 1001 0000 0000
```

- Valor do registrador \$s0 após o comando acima:

```
0000 0000 0011 1101 0000 1001 0000 0000
```

Endereçamento MIPS para imediatos de 32-bits e endereços

- Exemplo 17:

- Versão em linguagem de máquina para a instrução **lui \$t0, 255**
 - \$t0 é o registrador 8

001111	00000	01000	0000 0000 1111 1111
--------	-------	-------	---------------------

- Conteúdo do registrador \$t0 após a execução de **lui \$t0, 255**

0000 0000 1111 1111	0000 0000 0000 0000
---------------------	---------------------

Endereçamento MIPS para imediatos de 32-bits e endereços

- O tamanho dos endereços de memória (32-bits) em instruções sw e lw também podem ser um problema.
- Se o trabalho de carregamento de tais endereços fica a cargo do assembly MIPS, então algum registrador temporário precisa ser utilizado.
- Registrador \$at é utilizado.
- Registrador \$at é reservado especificamente para tal uso pelo assembler.

Endereçamento em jumps e branches

- Instruções de jump em MIPS utilizam o formato mais simples: tipo J.
- 6 bits para o campo de operação (opcode) e o restante (26 bits) para o endereço em si.
- A instrução: j 10000, pode ser organizada da seguinte maneira no formato J:



Endereçamento em jumps e branches

- Diferentemente das instruções do tipo J, os branches precisam especificar dois campos a mais no tipo I.



- Dessa forma, endereços não podem ter tamanho superior a 2^{16} .
- Problema: Espaço de endereçamento de 16-bits é muito pequeno

Endereçamento em jumps e branches

- Solução: Especificar um registrador que sempre será adicionado ao endereço de branch.
- A instrução de branch calcularia então o endereço-alvo à partir da seguinte fórmula:
 $\text{Contador de programa (PC)} = \text{Registrador base} + \text{Endereço de branch}$
- Permite endereços com tamanhos superiores (2^{32}).
- Problema: Qual registrador utilizar como base ?

Endereçamento em jumps e branches

- Solução:

- Branches (if) tendem a fazer o salto para endereços próximos.
- Registrador PC (program counter) armazena a posição corrente de execução do programa.
- Logo, para saltos de até $\pm 2^{15}$ posições adiante, o registrador \$pc é utilizado.
- Esta forma de endereçamento é conhecida como endereçamento relativo ao PC (PC-relative addressing).

Endereçamento em jumps e branches

- Todas as instruções em MIPS são de tamanho 4 bytes (32 bits).
- Importante: MIPS aumenta a distância possível de salto fazendo com que o endereçamento relativo ao PC se refira ao número de palavras (words) para a próxima instrução a ser executada (e não ao número de bytes).
- Um campo de 16-bits pode então acessar endereços 4x mais distantes em contrapartida ao endereçamento relativo a bytes.

Endereçamento em jumps e branches

- Similarmente, nas instruções de jump o campo de endereço é relativo ao tamanho de uma palavra (word) e não aos bytes.
- O campo de 26-bits pode então representar um endereço de até 28-bits.
- Importante: De onde vem os 4-bits restantes?
 - A instrução jump do MIPS altera somente os 28 bits menos significativos.
 - Carregador e linkador evitam colocar o código a uma distância maior que 256MB (64 milhões de instruções).

Endereçamento em jumps e branches

- Exemplo 18:

- Traduzir o código em MIPS assembly abaixo para a linguagem de máquina considerando que os endereços iniciam em 80000:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8		0	
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8		0	
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8	0		
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8		0	
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit:
```

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8		0	
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					

Endereçamento em jumps e branches

- Exemplo 18:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
```

Exit:

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8		0	
80012	5	8	21		2	
80016	8	19	19		1	
80020	2			20000		
80024	...					