Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Bacharelado em Ciência da Computação
Engenharia de Computação

Arquitetura e Organização de Computadores I

Prática

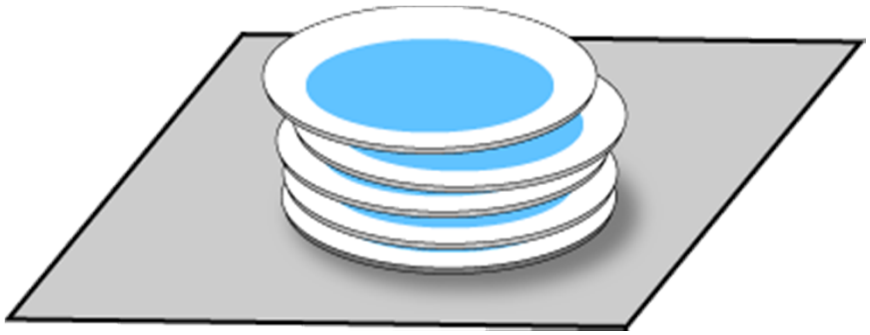
Aula 9

Pilhas e Pilha de Execução

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

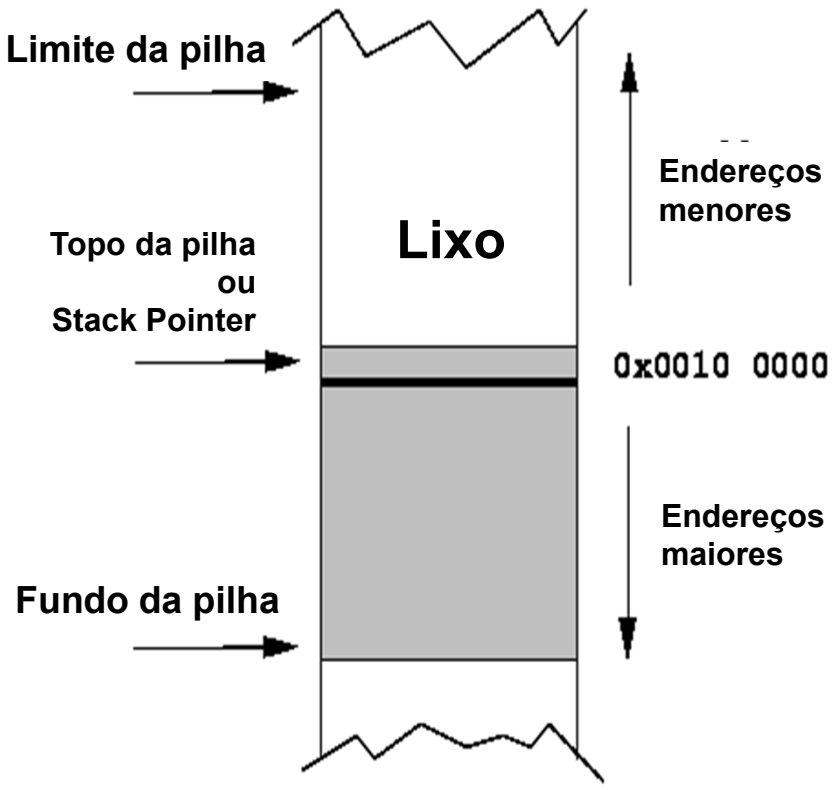
Pilhas

- Estrutura de dados clássica;
- Análoga a uma pilha de pratos;
- **Coloca**-se um prato (dados) no **topo** da pilha;
- **Retira**-se o prato que estiver no **topo** da pilha.



Pilhas

- No MIPS, a pilha de execução é uma **pilha invertida**;
- O **topo** da pilha é a sua palavra de **menor endereço**;
- Por **convenção**, o topo da pilha é sempre indicado pelo registrador **$sp**



Pilhas

Operação push

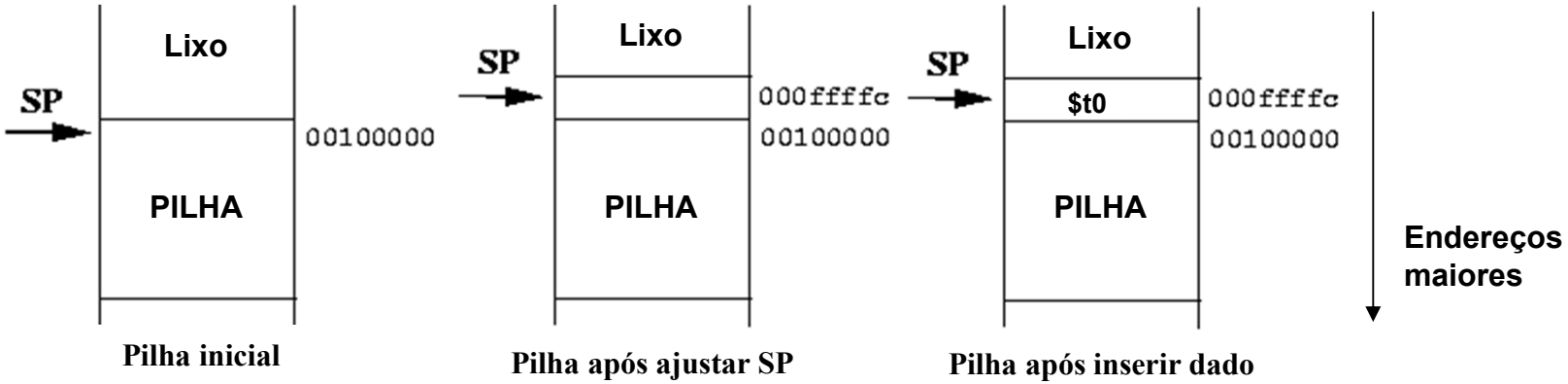
- **Armazena** uma palavra no **topo** da pilha;
- Atualiza $sp:

  $sp ← $sp - 4
- Escreve a palavra a ser armazenada no endereço indicado por $sp

```
addi $sp,$sp,-4          # aponta para o endereço do novo item,
sw   $t0,($sp)     # armazena o conteúdo de $t0 no novo topo
```

Pilhas

Operação push




```
addi $sp,$sp,-4          # aponta para o endereço do novo item,
sw   $t0,($sp)     # armazena o conteúdo de $t0 no novo topo
```

Pilhas

Operação pop

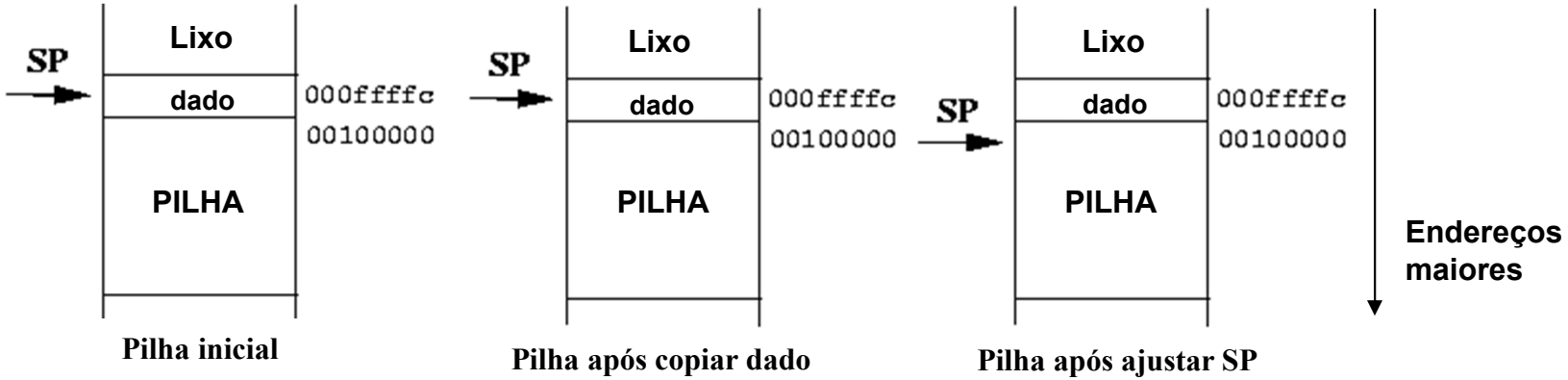
- **Retira** uma palavra do **topo** da pilha;
- Lê a palavra do endereço indicado por $sp e salva em um registrador
- Atualiza $sp:

  $sp ← $sp + 4

```
lw   $t0,($sp)      #   Copia o item para $t0
addi $sp,$sp,4              #   Atualiza topo
```

Pilhas

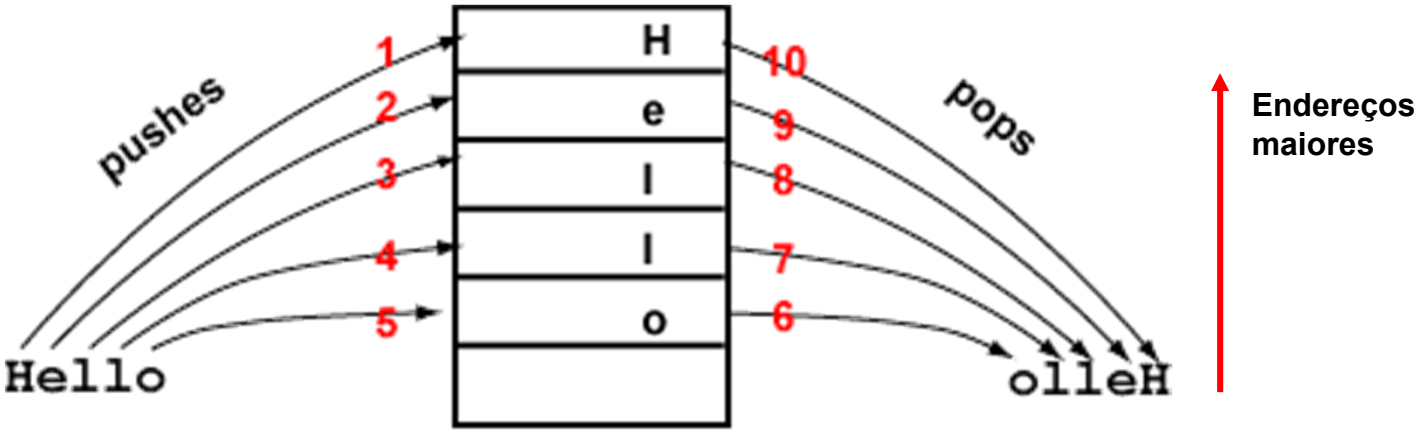
Operação pop




```
lw   $t0,($sp)      #  Copia o item para $t0
addi $sp,$sp,4             #  Atualiza topo
```

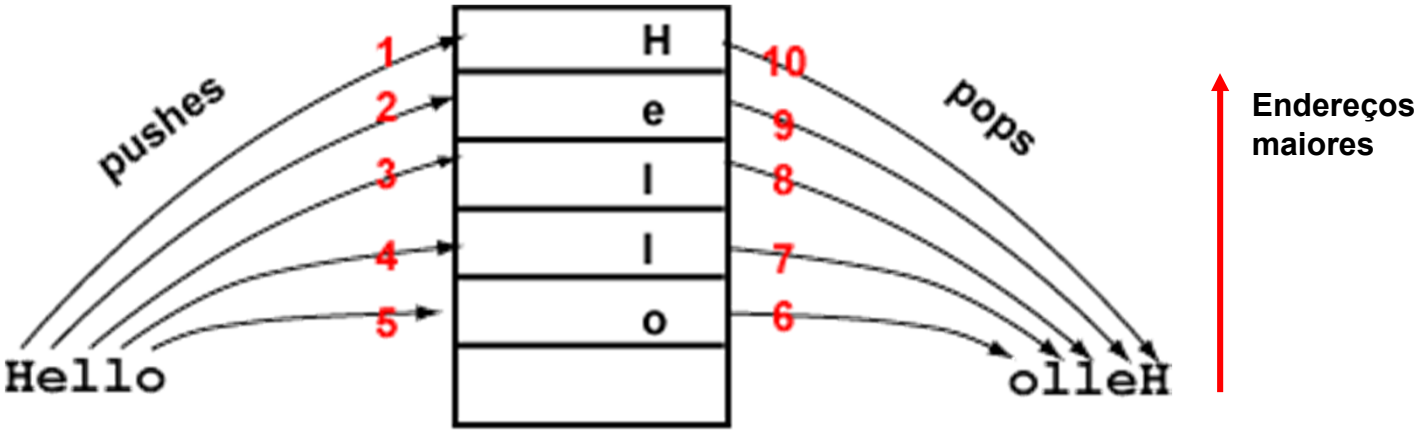
Pilhas

Exemplo: inverter uma *string*



Pilhas

Exemplo: inverter uma *string*

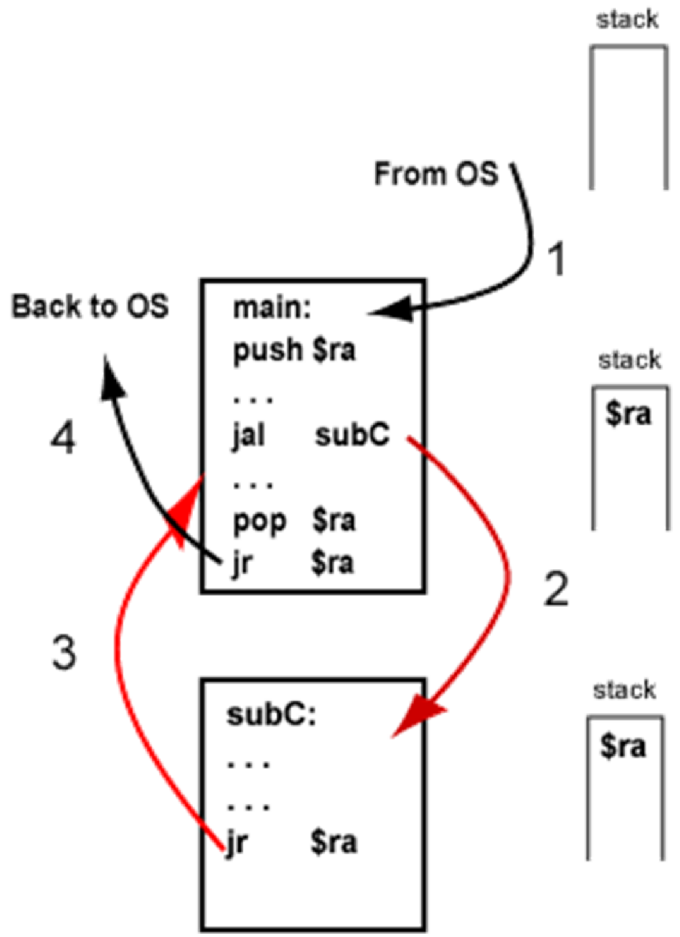


Exercício:

Leia uma *string* com comprimento máximo de 30 caracteres e a inverta utilizando a pilha do MIPS.

Pilha de Execução

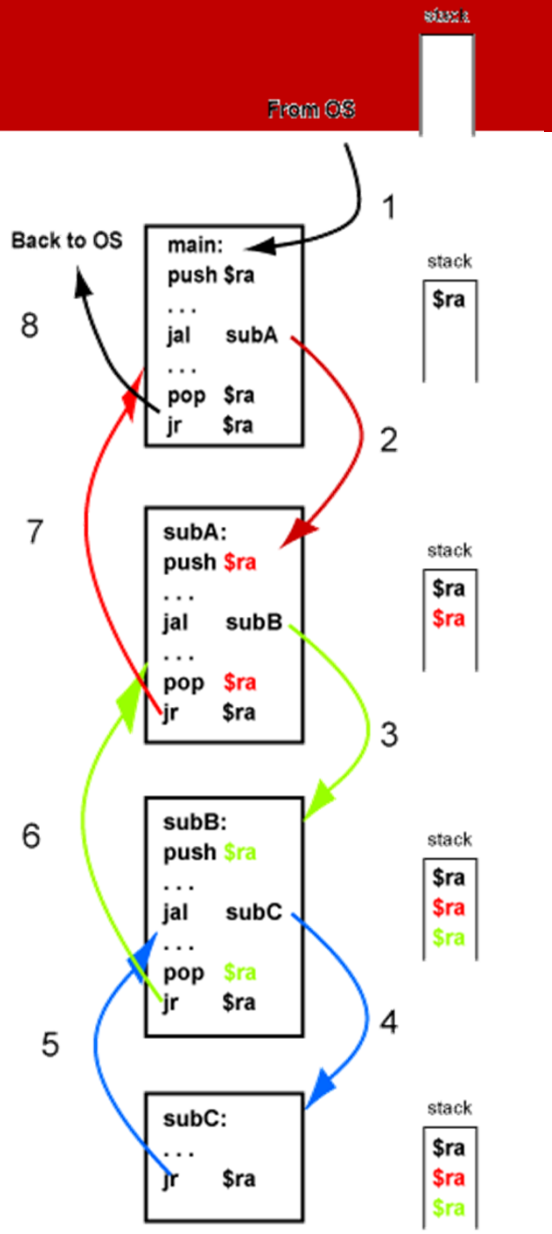
- O **endereço de retorno** de uma subrotina encontra-se sempre em **$ra**;
- Para que uma subrotina possa chamar outra, **$ra precisa ser salvo**;
- A **pilha de execução** é normalmente utilizada para salvar esses valores, já que o número de registradores é limitado.



Pilha de Execução

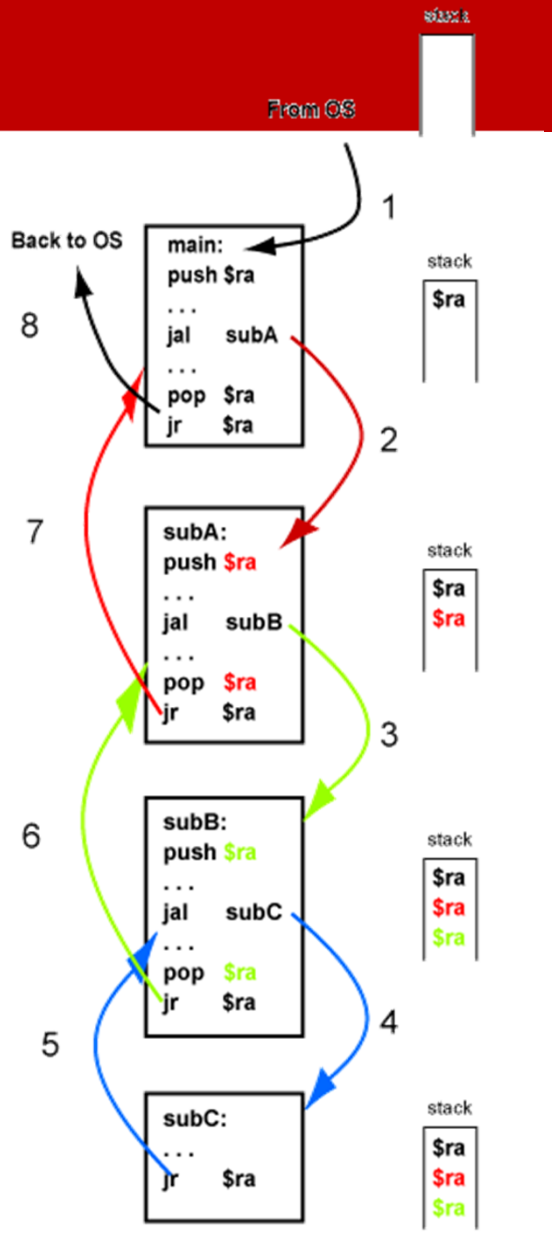
- Cada subrotina que, em seu código chama outra subrotina, **empilha o valor de $ra** em seu ponto de entrada;

- Antes da subrotina retornar, **$ra deve ser desempilhado**, deixando a pilha da mesma forma em que estava anteriormente.



Pilha de Execução

- Além de $ra, **qualquer** outro registrador pode ser salvo na pilha;
- Basta que a subrotina **desempilhe** todos os valores que tenha empilhado antes de sair;
- Registradores salvos **$s0-$s7** podem ser reutilizados se os valores originais forem salvos na pilha e recuperados ao fim da subrotina.



Pilha de Execução

Guia para utilização da pilha de execução

- Chamada da subrotina (executado pelo *chamador*):
  1. Empilhar registradores **$t0-$t9** que precisem ser salvos, se existirem. A subrotina pode alterá-los.
  2. Colocar argumentos em **$a0-$a3**.
  3. Chamar a subrotina usando **jal**.

- Prólogo da subrotina (executado no início da subrotina):
  4. Se a subrotina chama outras subrotinas, salvar **$ra** na pilha.
  5. Salvar na pilha os registradores **$s0-$s7** que a subrotina venha a alterar.

Pilha de Execução

Guia para utilização da pilha de execução

- Corpo da subrotina:
  6. A subrotina pode alterar qualquer registrador **T** ou **A**, ou qualquer registrador **S** que tenha sido salvo no passo 5.
  7. Se a subrotina chama outra subrotina, ela cumpre os passos deste guia.

- Epílogo da subrotina (executado ao final da subrotina, antes de retornar):
  8. Colocar valores de retorno em **$v0**-**$v1**.
  9. Desempilhar (ordem inversa) quaisquer registradores **$s0**-**$s7** que tenham sido salvos no passo 5.
  10. Se **$ra** foi empilhado no passo 4, desempilhá-lo.
  11. Retornar ao chamador usando **jr $ra**.

Pilha de Execução

Guia para utilização da pilha de execução

- Recuperando o controle (executado pelo chamador após a subrotina):

12. Desempilhar (ordem inversa) quaisquer registradores **$t0-$t9** que tenham sido salvos no passo 1.

Observe que este modelo suporta recursividade!