Aula 07 - Funções em Assembly e implementação de condicionais - 16/12/22

Como implementar >, ≥, < e ≤ ?

Contamos com as instruções:

- slt (rs < rt?) então passa 1 se rs < rt. Caso rs !< rt, passa 0 para o terceiro registrador.
- beq (desvia se rs == rt) beq rs, rt, label
- bne (desvia se rs ≠ rt) bne rs, rt, label

Desvio se < (digamos $s0 < $s1):

```
slt $t0, $s0, $s1
bne $t0, $zero, menor
```

Desvio se > (digamos $s0 > $s1):

```
slt $t0, $s1, $s0 ;$t0 = 1 se $s1 < $s0
bne $t0, $zero, maior ;$t0 = 0 $s1 >= $s0
```

Desvio se (set on less then) ≥ (digamos $s0 ≥ $s1):

```
slt $t0, $s0, $s1 ;$t0 = não existe (0) <=> $s0 !< $s1 -> $s0 >= $s1
beq $t0, $zero, maiorIgual
```

Desvio se ≤ ($s0 ≤ $s1):

```
slt $t0, $s1, $s0
beq $t0, $zero, menorIgual
```

Procedimentos

- **caller**: rotina que chama um procedimento.
- **callee**: procedimento chamado
- **Program Counter (PC)**: registrador especial que contém o endereço de memória da instrução que está sendo executada.

Para chamar um procedimento:

1. Coloque os parâmetros nos registradores adequados **(Opcional) → $a0 - $a3**.
2. Desvie a execução para o procedimento. **(Obrigatório)**
3. Ajuste o armazenamento no procedimento. **(Opcional)**
4. Execute o procedimento.
5. Salve o resultado no registrador adequado **(Opcional) → ($v0 - $v1) → o ideial é usar $v0 porque com $v1 ele faz mais de 1 retorno.**
6. Restaure os valores dos registradores $s (que foram salvos no Passo 3). **(Opcional)**
7. Retorne ao caller.

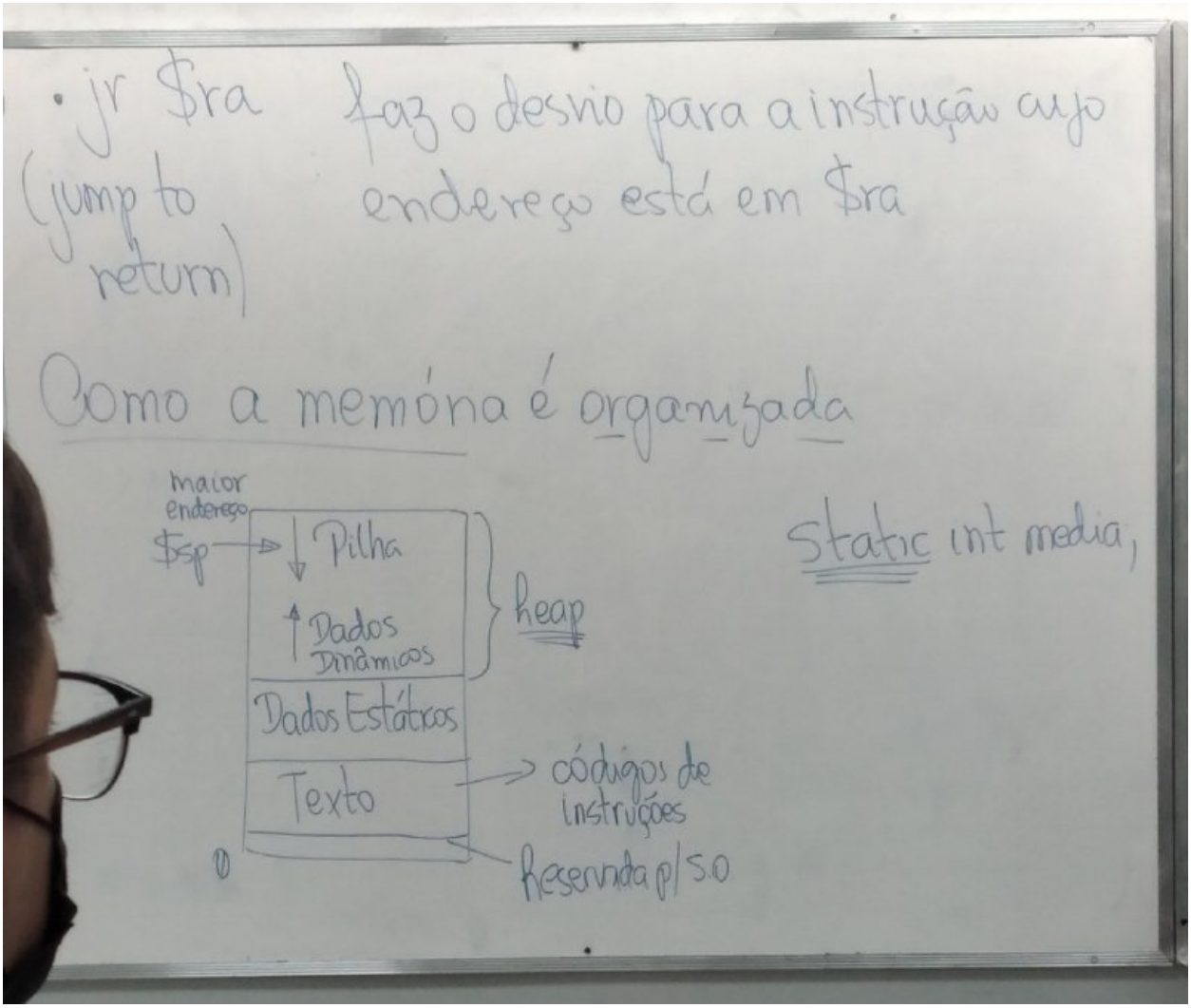
Para fazer os desvios, temos 2 instruções:

- **jal (jump-and-link) label:** faz o desvio para a instrução rotulada por label e salva o endereço da próxima instrução no registrador $ra.
- **jr (jum to return) $ra**: faz o desvio para a instrução cujo endereço está em $ra.

Como a memória é organizada

- **Variáveis Estáticas**: São declaradas e ocupam um espaço específico da memória (Dados Estátiticos), mesmo quando o procedimento terminar, a memória vai preservar o valor da variável no endereço no qual ela estava alocada.

- **$sp = Stack Pointer.**



Para usar a pilha:

1. Aloque espaço na pilha
    - Subtrai do registrador $sp a quantidade, em bytes, que deseja-se alocar.
2. Salve os dados.
    - Usando sw, sabendo que o end base é $sp.

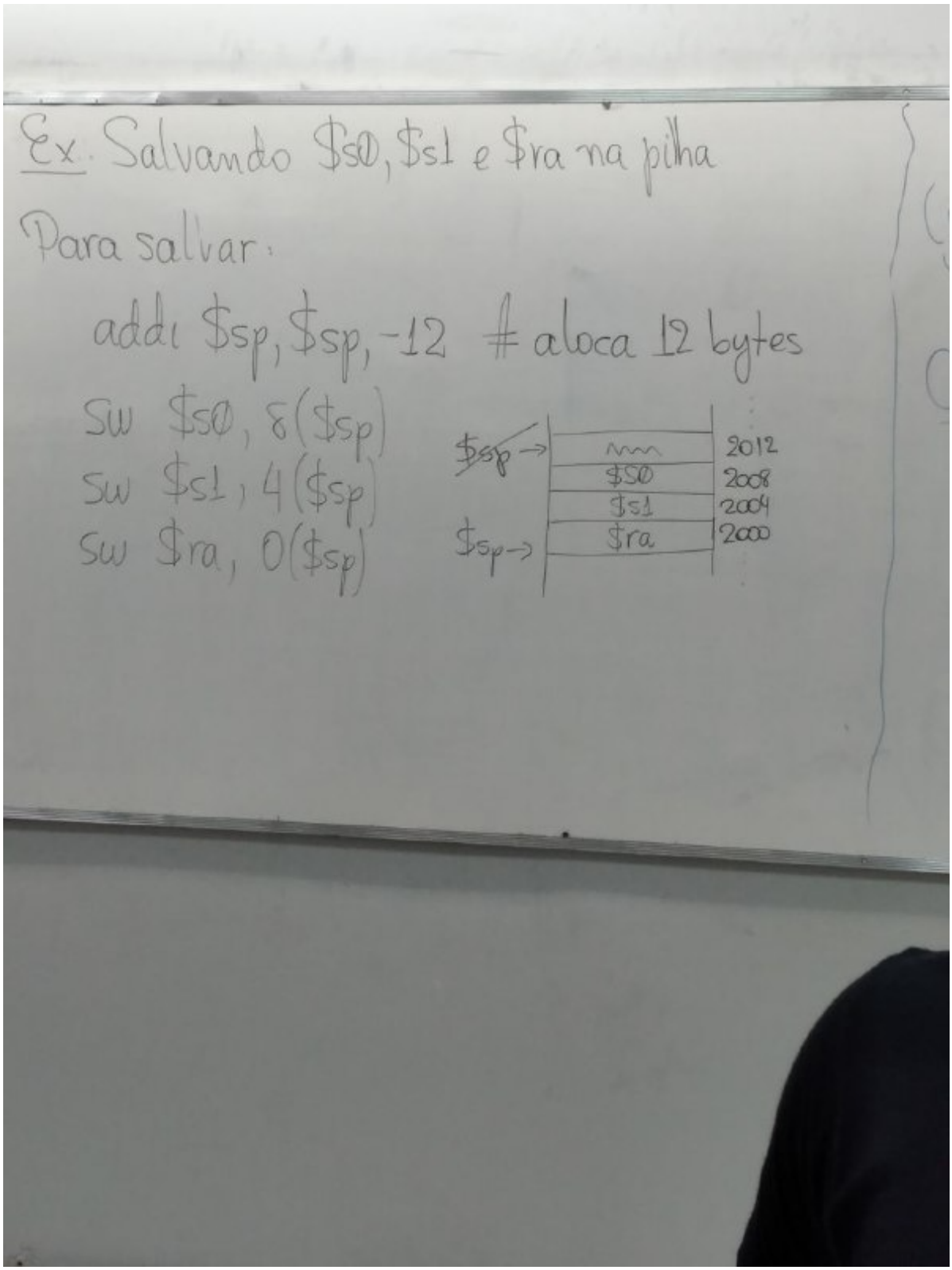
3. Restauras os dados.
    - Usando lw

4. Desalocar o espaço na pilha.
    - Soma ao registrador $sp a quantidade em bytes, alocada em 1.

Ex.: Salvando $s0, $s1 e $ra na pilha.

Para salvar:

```
addi $sp, $sp, -12 ;aloca 12 bytes
sw $s0, 8($sp)
sw $s1, 4($sp)
sw $ra, 0($sp)
; operações acima: SALVA
```



Para restaurar:

```
lw $s0, 8($sp)
lw $s1, 4($sp)
lw $ra, 0($sp)
; operações acima: RESTAURA
addi $sp, $sp, 12 ;desaloca
```