

Microprocessadores

Hugo Marcondes
hugo.marcondes@ifsc.edu.br

Aula 08

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

Usa-se as chamadas de procedimentos quando a complexidade do código fica maior

Agenda



INSTITUTO FEDERAL
SANTA CATARINA

- Introdução a chamadas de procedimentos
 - **Caller vs. Callee**
 - Chamada de procedimentos básica
- Convenções de chamadas
- Pilha
 - Interação com a pilha
 - Organização e estrutura
- Ligação de Subrotinas

2 IFSC - Arquitetura de Computadores

O que é um procedimento



INSTITUTO FEDERAL
SANTA CATARINA

- **"Pedaço" de código reutilizável!**
 - Utilizado para fazer a mesma coisa em diferentes pontos do programa
 - Utilizado para organizar logicamente (decomposição)
- **Qual a diferença entre procedimento e função ?**
- **Procedimento podem chamar outros procedimentos !**
 - **Inclusive ele mesmo (recursão)**

3 IFSC - Microprocessadores

O que acontece ?



INSTITUTO FEDERAL
SANTA CATARINA

- Caller vs. Callee
 - **Caller - o código que chama o procedimento**
 - **Callee - o código que implementa o procedimento**

Chamada de Procedimento - Visão Geral

1. **caller chama callee**
 - caller para execução
 - controle é passado para o callee
2. **Callee executa**
3. **Callee retorna** para o caller
 - callee para execução
 - callee continua a execução do ponto onde a chamada foi realizada



4 IFSC - Microprocessadores

Chamando e retornando de um procedimento



- Para chamar um procedimento: **jal label**
 - **jal** - Jump and Link
 - 1. Armazena em **\$ra**, PC+4
salva o endereço da próxima instrução a ser executada pelo caller
 - 2. Armazena no PC o **label**
Desvia para o endereço da primeira instrução do callee
- Para retornar do procedimento: **jr \$ra**
 - **jr** - Jump to Register
 - Desvia de volta para a próxima instrução do caller

5 IFSC - Microprocessadores

PC -> program counter

Quando chamo Jump and Link meu endereço da onde eu estava ele fica armazenado no \$ra, e quando fizer jr (jump to register) \$ra volta para onde estava antes da função, que é onde está meu endereço de memória.

Argumentos e valores de retorno



- Por convenção
 - Os primeiros 4 argumentos em **\$a0 - \$a3**
 - Colocar o valor de retorno no **\$v0** e **\$v1**
- Visão simplificada !
 - Suporta apenas 4 argumentos (palavras)
 - Todo procedimento é final
 - Não chama nenhum outro procedimento

6 IFSC - Microprocessadores

Argumentos e valores de retorno



- Implemente o seguinte pseudo código

```
# Definição do procedimento
# Pseudo código:
# int sumOfSquares(int a, int b) {
#     return a*a + b*b;
# }
```

```
# Uso do procedimento
# Pseudo código:
# int main() {
#     int c;
#     c = sumOfSquares(3,5)
# }
```

7 IFSC - Microprocessadores

Exemplo



```
# Definição do procedimento
sumOfSquares:
    mul $t0, $a0, $a0 # tmp1 = a*a
    mul $t1, $a1, $a1 # tmp2 = b*b
    add $v0, $t0, $t1 # res = tmp1 + tmp2
    jr $ra

# Uso do procedimento
li $a0, 3 # (prepara parâmetro)
li $a1, 5 #
jal sumOfSquares # (chama procedimento)
move $t2, $v0 # (pega resultado)
```

8 IFSC - Microprocessadores

A pilha vai servir para armazenar valores e endereços de vários procedimentos.

É muito fácil pensar que se tiver 3 instruções e eu salvar \$ra o endereço e dps passar pro próximo e dps pro próximo eu vou modificar o \$ra para o do meio e quando voltar vai ficar eternamente em loop

