Relatório - Laboratório 05
Disciplina: INE5411 – Organização de Computadores I
Integrantes: Rodrigo Martins dos Santos e Lucas Pastre de Souza.

Introdução:

O presente relatório busca mostrar o funcionamento dos exercícios 1, 2 e 3 referentes à quinta atividade de laboratório da disciplina Organização e Arquitetura de Computadores do curso de Ciências da Computação na Universidade Federal de Santa Catarina (UFSC). Esses exercícios abordam os conceitos de procedimentos e BHT (Branch History Table) em Assembly.

Exercício 1:

O exercício 1 implementa um programa em Assembly MIPS que calcula o fatorial de um número n via input de teclado sem o uso de procedimentos e mostra o resultado no terminal do MARS, como demonstra o código abaixo:

```
.data
        n:.word 0

        msg: .asciiz "Digite um valor para n: "
        msgR: .asciiz "Fatorial de n: "

.text
        li $v0, 4
        la $a0, msg
        syscall

        li $v0, 5
        syscall
        sw $v0, n

        li $s0, 1
        lw $s1, n
        li $s2, 1

        beq $s1, $zero, fim

        loop:
                beq $s1, $s0, fim
                mul $s2, $s2, $s1
                addi $s1, $s1, -1

                j loop

        fim:
                sw $s2, n

                li $v0, 4
                la $a0, msgR
                syscall

                li $v0, 1
                move $a0, $s2
                syscall

                li $v0, 10
                syscall
```

Exercício 2:

O exercício 2 assim como o exercício 1 visa a implementação de um programa que calcule o fatorial de um número n, mas desta vez utilizando procedimentos em Assembly MIPS, como demonstra o código abaixo:

```
1   .data
2           n:  .word 0
3
4           msg:  .asciiz "Digite um valor para n: "
5           msgR:  .asciiz "Fatorial de n: "
6
7   .text
8
9   main:
10
11          li $v0, 4
12          la $a0, msg
13          syscall
14
15          li $v0, 5
16          syscall
17          move $a0, $v0
18          sw $a0, n
19
20          jal Fatorial
21
22          sw $v0, n
23
24          li $v0, 4
25          la $a0, msgR
26          syscall
27
28          li $v0, 1
29          lw $a0, n
30          syscall
31
```

```
32
33          li $v0, 10
34          syscall
35
36  Fatorial:
37
38  # Caso base: se n <= 1, retorna 1
39
40          addi $t0, $zero, 1
41          ble $a0, $t0, return1
42
43  # Caso recursivo: n * Fatorial(n - 1)
44
45          addi $sp, $sp, -8           # ajusta a pilha para salvar $ra
46          sw $a0, 0($sp)              # salva o valor de $ra na pilha
47          sw $ra, 4($sp)              # salva o valor de $a0 na pilha
48
49          addi $a0, $a0, -1           # decrementa n
50
51          jal Fatorial                # chama Fatorial(n - 1)
52
53          lw $a0, 0($sp)              # restaura o valor de $ra da pilha
54          lw $ra, 4($sp)              # restaura o valor de $a0 da pilha
55          addi $sp, $sp, 8            # ajusta a pilha de volta
56
57          mul $v0, $a0, $v0           # multiplica n pelo resultado da recursão
58          jr $ra                      # retorna ao chamador
59
60          return1:
61                  addi $v0, $zero, 1
62                  jr $ra                      # retorna ao chamador
```

Exercício 3:

O exercício 3 visa executar os 2 exercícios anteriores no BHT Simulator do MARS e analisar qual dos códigos obteve o melhor desempenho, segue abaixo a análise:

Em ambos os testes tanto para BHT de 1 Bit quanto para BHT de 2 bits os resultados foram relativamente próximos, porém o exercício 1 apresenta uma leve vantagem pois em loops iterativos, o branch predictor tem mais facilidade para prever corretamente quando o loop terminará, especialmente se ele segue um padrão fixo. A versão iterativa tende a ter uma melhor taxa de acertos na previsão de branch, principalmente com um BHT de 2 bits, pois o padrão de branches é mais constante.

Algo que não acontece no exercício 2 pois em uma chamada recursiva, cada chamada ao próprio método “Fatorial” cria uma nova previsão de branch, onde o controle é transferido para o endereço de retorno, o que dificulta a previsão do branch predictor.