UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ACH2034 - ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

RELATÓRIO DO EXERCÍCIO-PROGRAMA (EP) DA DISCIPLINA

‘ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I’

DAVI BATISTA DE SOUZA

DIOGO DOS SANTOS DA ROCHA

Arquitetura do Conjunto de Instruções (ISA): A Arquitetura do Conjunto de Instruções (ISA) define o repertório de instruções de linguagem de máquina que um computador pode executar. A ISA estabelece a interface entre hardware e software, determinando as operações básicas que o processador pode realizar. Exemplos incluem adição, subtração e movimentação de dados.

Arquitetura MIPS: MIPS, que significa "Microprocessor without Interlocked Pipeline Stages", é uma arquitetura de load-store com 32 registradores de propósito geral e 32 registradores de ponto flutuante. Todas as instruções têm 32 bits de tamanho fixo, facilitando a compreensão e execução eficiente em pipelines.

Categorias de operações MIPS: As operações MIPS são divididas em transferência de dados, lógica e aritmética, controle e ponto flutuante. Cada categoria realiza funções específicas como movimentação de dados entre registradores e memória, cálculos aritméticos e lógicos, e controle de fluxo de execução.

Descrição do problema

Nosso problema consistia basicamente em somar dois bits, retornando o valor da soma e do "vai-um"

CÓDIGO EM C

```c
#include <stdio.h>

#include <stdlib.h>

void somabit(int b1, int b2, int* vaium, int* soma) {

  *soma = (b1 ^ b2) ^ *vaium;

  *vaium = (b1 & b2) | (b1 & *vaium) | (b2 & *vaium);

}

int main() {

  int b1 = 1, b2 = 1, vaium = 1, soma;

  somabit(b1, b2, &vaium, &soma);

  printf("Soma: %d\n", soma);

  printf("Vai-um: %d\n", vaium);

  return 0;

}
```

CÓDIGO EM ASSEMBLY

O código que foi desenvolvido em Assembly é composto por duas partes principais, sendo elas: “.data” e “.text”. A primeira parte abriga as variáveis que serão manipuladas com a ajuda dos registradores ofertados pelo MIPS. Já a segunda parte contém as rotinas que serão utilizadas para realizar a execução do programa.

Rótulo somabit: Este rótulo identifica a sequência de instruções que irá somar os bits “b1” e “b2”, depois comparar com “vaium” e por fim determinar o valor da soma.

Rótulo fim: O rótulo fim devolve ao usuário o valor das variáveis soma e vaium, e depois finaliza o programa

Rótulos “data” e “text”

```
67  .data
68  $b1: .word 1 # Declaração dos bits de b1 e b2
69  $b2: .word 0
70  $vaium: .word 1 # Declaração do vai-um e do resultado da soma
71  $soma: .word 0
72  $resultadoSoma: .asciiz "Resultado da soma: "
73  $resultadoVaiUm: .asciiz " Resultado do vai-um: "
```

```
1   .text
2   .globl main
3
4   main:
5       lw $t0, $b1 # Carrega o valor de b1 em t0
6       lw $t1, $b2 # Carrega o valor de b2 em t1
7       jal somabit # Chama a função somabit
8       j fim # Pula para o fim
```

Rótulos “somabit” e “fim”

```
10  fim:
11      la $a0, $resultadoSoma # Printando o resultado da soma
12      li $v0, 4
13      syscall
14
15      move $a0, $t7 # Transferindo o conteúdo do registrador t6 para a0
16      li $v0, 1
17      syscall
18
19      la $a0, $resultadoVaiUm # Carrega o endereço da string em a3
20      li $v0, 4
21      syscall
22
23      move $a0, $t6 # Printando o vai-um
24      li $v0, 1
25      syscall
26
27      li $v0, 10 # Termina o programa
28      syscall
```

```
30  somabit:
31      xor $t2, $a0, $a1 # Faz a operação XOR entre t0 e t1 e armazena em t2
32      addi $sp, $sp, -4 # Decrementa o ponteiro de pilha
33      sw $t2, 0($sp) # Armazena o valor de t2 na pilha
34      lw $t0, $vaium # Carrega o valor de vaium em t0
35      lw $v0, 0($sp) # Carrega o valor da pilha em v0
36      addi $sp, $sp, 4 # Incrementa o ponteiro de pilha
37      xor $t7, $t0, $v0 # Faz a operação XOR entre t0 e v0 e armazena em t1
38
39      lw $t0, $b1
40      lw $t2, $b2
41      and $t3, $t0, $t2 # Faz a operação AND entre t0 e t2 e armazena em t3
42      addi $sp, $sp, -4
43      sw $t3, 0($sp)
44
45      lw $t4, $vaium
46      and $t5, $t4, $t0 # Faz a operação AND entre t4 e t0 e armazena em t5
47      addi $sp, $sp, -4
48      sw $t5, 0($sp)
49
50      and $t6, $t2, $t4 # Faz a operação AND entre t2 e t4 e armazena em t6
51      addi $sp, $sp, -4
52      sw $t6, 0($sp)
53
54      lw $t0, 0($sp)
55      addi $sp, $sp, 4
56      lw $t1, 0($sp)
57      addi $sp, $sp, 4
58      lw $t2, 0($sp)
59      addi $sp, $sp, 4
60
61      or $t4, $t0, $t1
62      or $t6, $t4, $t2 # Faz a operação OR entre t4 e t2 e armazena em t5, SOMA ESTÁ EM T5
63
64      jr $ra
```