



**UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
INE-DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
INE5411 - ORGANIZAÇÃO DE COMPUTADORES I**

**PEDRO TAGLIALENHA (22203674)
ERIK ORSOLIN DE PAULA (22102195)**

RELATÓRIO LABORATÓRIO 1

**FLORIANÓPOLIS
2023**

1- Introdução

Assembly é uma linguagem de programação de baixo nível que se assemelha à linguagem de máquina. Cada instrução em Assembly corresponde a uma instrução para o processador, tornando a programação muito próxima do hardware. Neste relatório, abordaremos as instruções de Assembly MIPS usadas para realizar operações de alto nível em duas questões do Laboratório 01.

2- Funções de Assembly Usadas

- **lw**: Carrega uma palavra (32 bits) da memória para um registrador.
- **sw**: Armazena uma palavra de um registrador na memória.
- **addi**: Adiciona um valor imediato a um registrador.
- **sub**: Subtrai o valor de dois registradores e armazena o resultado em um terceiro registrador.
- **add**: Adiciona o valor de dois registradores e armazena o resultado em um terceiro registrador.
- **li**: Carrega um valor imediato em um registrador.
- **syscall**: Chama uma função do sistema.

3- Códigos desenvolvidos

Código 1:

```
# Questão 1
.data

A: .word 5
B: .word 5
C: .word 5
D: .word 5
E: .word 5

.text

lw $s0, A      # registrador $s0 recebe o conteúdo de A
lw $t0, B      # registrador temporário $t0 recebe o conteúdo de B

addi $s0, $t0, 35  # registrador $s0 recebe o conteúdo de B + 35 ( A = B + 35), logo
$s0 deve ter o valor 40

lw $s1, C      # registrador $s1 recebe o conteúdo de C
lw $t1, E      # registrador temporário $t1 recebe o conteúdo de E
lw $t2, D      # registrador temporário $t2 recebe o conteúdo de D

sub $t3, $t2, $s0 # registrador temporário $t3 recebe a subtração dos conteúdos dos
registradores $t2 e $s0 (D - A), logo $t3 deve ter o valor -35
add $s1, $t1, $t3 # o registrador $s1 recebe a subtração dos conteúdos dos registradores
$t2 e $t3 (D - A + E), logo $s1 deve ter o valor -30
sw $s1, C      # gravando o resultado na variável C
```

Fonte: Elaborado por autores(2023)

- A memória de dados é inicializada com as variáveis A, B, C, D e E contendo o valor 5.
- \$s0 e \$s1 são registradores de armazenamento que serão usados para armazenar as

variáveis A e C, respectivamente.

- \$t0, \$t1, \$t2, \$t3 são registradores temporários usados para operações intermediárias.

Código 2:

```
.data
    A: .word 5
    B: .word 0    # Inicialmente, B será 0
    C: .word 5
    D: .word 5
    E: .word 5

.text
    .globl main

main:

    # Lendo o valor de B do usuário
    li    $v0, 5        # Código do syscall para ler inteiro
    syscall
    sw    $v0, B        # Armazenando o valor lido na variável B

    lw    $s0, A        # $s0 recebe o conteúdo de A
    lw    $t0, B        # $t0 recebe o conteúdo de B

    addi   $s0, $t0, 35 # $s0 recebe o conteúdo de B + 35 (A = B + 35), $s0
deve ter o valor 40

    lw    $s1, C        # $s1 recebe o conteúdo de C
    lw    $t1, E        # $t1 recebe o conteúdo de E
    lw    $t2, D        # $t2 recebe o conteúdo de D

    sub    $t3, $t2, $s0 # $t3 recebe a subtração dos conteúdos dos
registradores $t2 e $s0 (D - A), $t3 deve ter o valor -35
    add    $s1, $t1, $t3 # $s1 recebe a soma dos conteúdos dos registradores
$t1 e $t3 (E + D - A), $s1 deve ter o valor -30

    sw    $s1, C        # Gravando o resultado na variável C

    lw    $a0, C        # Carregando o valor de C para o registrador $a0
    li    $v0, 1        # Código do syscall para imprimir um inteiro
    syscall             # Realizando a chamada de sistema para imprimir o
valor em $a0

    # Terminando o programa
    li    $v0, 10       # Código do syscall para sair do programa
    syscall             # Realizando a chamada de sistema para terminar o
programa
```

Fonte: Elaborado por autores(2023)

Nesta variação, o valor de B é fornecido pelo usuário via teclado usando o syscall com código 5. O restante do programa é semelhante à Questão 1, exceto que agora B é dinâmico.

4- saídas dos códigos:

Figura 1: Saída do código 1

The screenshot displays the MARS 4.5 assembly simulator. The main window shows the assembly code for a program. The code includes instructions for loading values into registers, performing arithmetic operations, and using the syscall instruction to interact with the operating system. The Registers panel on the right shows the current state of the registers, with values like 0x00000000 for \$zero and 0x00000001 for \$at. The Data Segment panel at the bottom shows the memory layout, with addresses and values for various data segments. The Mars Messages panel at the bottom left shows the output of the program, indicating that it has finished running.

Fonte: Elaborado por autores(2023)

Para mostrar o print foram adicionados as seguintes linhas para realizar o print:

```
lw    $a0, C      # carrega o valor de C para o registrador $a0
li    $v0, 1       # código do syscall para imprimir um inteiro
```

syscall

chama o syscall para imprimir o valor em \$a0

Figura 2: Saída do código 2

The screenshot displays the MARS 4.5 IDE interface. The main window shows assembly code for a program that reads integers from memory and prints them. The code is organized into a 'Text Segment' with instructions like `addiu $2,$0,0x00000005`, `syscall`, `lui $1,0x00001001`, `sw $2,0x00000004($1)`, `lw $s0,A`, `lw $t0,B`, `addi $s0,$t0,35`, `lw $s1,C`, `lw $t1,E`, `lw $t2,D`, `sub $t3,$t2,$s0`, `add $s1,$t1,$t3`, `sw $s1,C`, `lw $a0,C`, `addiu $2,$0,0x00000001`, `syscall`, and `addiu $2,$0,0x0000000a`. Comments in Portuguese explain each step, such as 'Código do syscall para ler inteiro', 'Armazenando o valor lido na variável', 'recebe o conteúdo de A', etc.

On the right, the 'Registers' panel shows the state of various registers. Notable values include `$a0` at 0x00000002, `$a1` at 0x00000000, `$a2` at 0x00000000, `$a3` at 0x00000000, `$t0` at 0x00000005, `$t1` at 0x00000005, `$t2` at 0x00000005, `$t3` at 0xfffffdd, `$t4` at 0x00000000, `$t5` at 0x00000000, `$t6` at 0x00000000, `$t7` at 0x00000000, `$s0` at 0x00000028, `$s1` at 0xffffffe2, `$s2` at 0x00000000, `$s3` at 0x00000000, `$s4` at 0x00000000, `$s5` at 0x00000000, `$s6` at 0x00000000, `$s7` at 0x00000000, `$t8` at 0x00000000, `$t9` at 0x00000000, `$k0` at 0x00000000, `$k1` at 0x00000000, `$gp` at 0x10008000, `$sp` at 0x7fffffc0, `$fp` at 0x00000000, `$ra` at 0x00000000, `pc` at 0x00400064, `hi` at 0x00000000, and `lo` at 0x00000000.

At the bottom, the 'Mars Messages' panel shows the execution output: '0', '-25', '-- program is finished running --', 'Reset: reset completed.', '5', '-30', and '-- program is finished running --'. A 'Clear' button is visible next to the messages.

Fonte: Elaborado por autores(2023)

5- Conclusão

Os programas em Assembly aqui descritos são exemplos diretos de como as operações de alto nível podem ser realizadas em Assembly. Eles mostram o uso de instruções para manipulação de dados, aritmética e chamadas de sistema.