



Disciplina: INE5411
Prof.: Marcelo Daniel Berejuck
Aluno: João Pedro Schmidt Cordeiro
Turma: 03208B

Semestre: 2023.1

Laboratório 1

1. Exercício 1

The screenshot shows a MIPS simulator interface. The top panel displays assembly code with instructions like `lui $t0, A`, `ori $t1, B`, `ori $t2, C`, `ori $t3, D`, `ori $t4, E`, `lw $t0, 0($t1)`, `addi $t1, $t0, 35`, and `lw $t0, 0($t4)`. The middle panel shows the data segment with values at various addresses. The right panel shows the registers, with `$t0` containing 268500992, `$t1` containing 268500996, `$t2` containing 268501000, `$t3` containing 268501004, `$t4` containing 268501008, `$t5` containing 268501012, `$t6` containing 268501016, `$t7` containing 268501020, `$t8` containing 268501024, `$t9` containing 268501028, `$t10` containing 268501032, `$t11` containing 268501036, `$t12` containing 268501040, `$t13` containing 268501044, `$t14` containing 268501048, `$t15` containing 268501052, `$t16` containing 268501056, `$t17` containing 268501060, `$t18` containing 268501064, `$t19` containing 268501068, `$t20` containing 268501072, `$t21` containing 268501076, `$t22` containing 268501080, `$t23` containing 268501084, `$t24` containing 268501088, `$t25` containing 268501092, `$t26` containing 268501096, `$t27` containing 268501100, `$t28` containing 268501104, `$t29` containing 268501108, `$t30` containing 268501112, `$t31` containing 268501116.

Conta com os valores alocados

$$a = 10 + 35$$

$$c = 15 - a + 40$$

$$c = 10$$

```
## Lab: 1
## Exercício: 1
## Nome: Joao Pedro Schmidt Cordeiro
## Matricula: 22100628
```

.data

```
A: .word 25
B: .word 10
C: .word 90
D: .word 15
E: .word 40
```

.text

main:

```
# Load all variable addresses
```

```

la    $s0, A
la    $s1, B
la    $s2, C
la    $s3, D
la    $s4, E

lw    $t0, 0($s1)
addi  $t1, $t0, 35

lw    $t0, 0($s4)
sub   $t2, $t0, $t1

lw    $t0, 0($s3)
add   $t3, $t2, $t0
sw    $t3, 0($s2)

```

2. Exercício 2

The screenshot shows the Mars MIPS simulator interface. The main window displays assembly code with addresses, codes, and comments. The registers window on the right shows the state of registers \$s0 through \$s4. The console at the bottom shows the execution output, including prompts for input and the final result.

Registers:

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	1
\$v1	3	0
\$a0	4	10
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	15
\$t1	9	0
\$t2	10	-5
\$t3	11	10
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	268500992
\$s1	17	268500996
\$s2	18	268501000
\$s3	19	268501004
\$s4	20	268501008
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194476
hi		0
lo		0

Console Output:

```

--- program is finished running (dropped off bottom) ---
Digite o valor de B: 10
Digite o valor de E: 40
Digite o valor de D: 15
0 resultado final eh: 10
--- program is finished running (dropped off bottom) ---

```

Conta com os valores alocados

$$a = 10 + 35$$

$$c = 15 - a + 40$$

$$c = 10$$

```

## Lab: 1
## Exercício: 2
## Nome: Joao Pedro Schmidt Cordeiro
## Matricula: 22100628

.data
    A: .word 0
    B: .word 0
    C: .word 0
    D: .word 0
    E: .word 0
    FRASE_B: .asciiz "Digite o valor de B: "
    FRASE_E: .asciiz "Digite o valor de E: "
    FRASE_D: .asciiz "Digite o valor de D: "
    FRASE_1: .asciiz "O resultado final eh: "

```

```

.text
main:
    ## Load all variable addresses
    la    $s0, A
    la    $s1, B
    la    $s2, C
    la    $s3, D
    la    $s4, E

    ## Primeira logica de ler o input
    li    $v0, 4                # Comando.
    la    $a0, FRASE_B          # Carrega string (endereço).
    syscall
    li    $v0, 5                # Comando para ler inteiro.
    syscall
    move   $t0, $v0              # Resultado salvo em $t0.

    addi   $t1, $t0, 35          # a = b + 35;

    ## Primeira logica de ler o input
    li    $v0, 4                # Comando.
    la    $a0, FRASE_E          # Carrega string (endereço).
    syscall
    li    $v0, 5                # Comando para ler inteiro.
    syscall
    move   $t0, $v0              # Resultado salvo em $t0.

```

```

sub    $t2, $t0, $t1          # $t2 = e - a

## Primeira logica de ler o input
li     $v0, 4                  # Comando.
la     $a0, FRASE_D            # Carrega string (endereço).
syscall

li     $v0, 5                  # Comando para ler inteiro.
syscall

move    $t0, $v0              # Resultado salvo em $t0.

add     $t3, $t2, $t0          # c = d + $t2
sw      $t3, 0($s2)

## Printando o resultado final
li     $v0, 4                  # Comando.
la     $a0, FRASE_1            # Carrega string (endereço).
syscall

lw      $t1, 0($s0)
li     $v0, 1                  # Comando.
move    $a0, $t3               # Carrega valor.
syscall

```