

**UNIVERSIDADE DO VALE DO ITAJAÍ  
CAMPUS ITAJAÍ  
ENGENHARIA DE COMPUTAÇÃO E CIÊNCIA DE  
COMPUTAÇÃO  
DISCIPLINA ARQUITETURA E ORGANIZAÇÃO DE  
COMPUTADORES  
PROFESSOR THIAGO FELSKI PEREIRA**

**RELATÓRIO AVALIAÇÃO 01 – PROGRAMAÇÃO EM  
LINGUAGEM DE MONTAGEM**

**Alunos: Eduardo Savian de Oliveira, Marcos  
Augusto Fehlauer Pereira, Yuri Rodrigues**

## Programa 01

### Código-fonte em C

```
#include <stdio.h>
#include <stdint.h>

#define i32 int32_t

int main(){
    i32 A[8], B[8], limit, tmp;
    while(1){
        printf("tamanho(1~8): ");
        scanf("%d", &limit);
        if(limit > 0 && limit <= 8)
            break;
        else
            printf("valor invalido\n");
    }

    for(i32 i=0; i < limit; i++){
        printf("A[%d] = ", i);
        scanf("%d", &A[i]);
    }

    for(i32 i=0; i < limit; i++){
        printf("B[%d] = ", i);
        scanf("%d", &B[i]);
    }

    for(i32 i=0; i < limit; i++){
        tmp = A[i];
        A[i] = B[i];
        B[i] = tmp;
    }

    for(i32 i=0; i < limit; i++) printf("A[%d] = %d\n", i, A[i]);

    for(i32 i=0; i < limit; i++) printf("B[%d] = %d\n", i, B[i]);

    return 0;
}
```

## Código-fonte em MIPS Assembly

```
#####  
# Disciplina: Arquitetura e Organização de Computadores  
# Atividade: Avaliação 01 – Programação em Linguagem de Montagem  
# Programa 01  
# Grupo: - Eduardo Savian de Oliveira  
#       - Marcos Augusto Fehlauser Pereira  
#       - Yuri Rodrigues
```

```
.data  
    ArrA: .word 0,0,0,0,0,0,0,0  
    ArrB: .word 0,0,0,0,0,0,0,0  
    msgLimite: .asciiz "Tamanho do vetor (1~8): "  
    msgA: .asciiz "A["  
    msgB: .asciiz "B["  
    msgSwitch: .asciiz "Trocando A e B!\n"  
    msgWarn: .asciiz "Valor invalido\n"  
    newline: .asciiz "\n"  
    cb_str: .asciiz "]" = "
```

```
.text
```

```
main:  
    addi $s4, $zero, 1      # limite = 1  
    la $a0, msgLimite      # prompt  
    addi $v0, $zero 4  
    syscall  
    addi $v0, $zero, 5      # syscall read integer  
    syscall  
    add $s4, $zero, $v0     # limite = <valor do usuario>  
    # avisar se o limite for invalido  
    bgt $s4, 8, warn  
    blt $s4, 1, warn  
    # Ler A  
    j readA  
returnA:  
    # Ler B  
    j readB  
returnB:  
    # Trocar  
    j switchAB  
returnSwitchAB:  
    # Imprimir  
    j printAB  
returnPrintAB:  
    j exit                  # fim do programa
```

# Ler array A

readA:

addi \$s0, \$zero, 0        # i = 0

loopReadA:

mul \$s1, \$s0, 4        # indice do array  
# prompt do usuario  
addi \$v0, \$zero, 4 # imprimir A[  
la \$a0, msgA  
syscall  
add \$a0, \$zero, \$s0 # imprime indice  
addi \$v0, \$zero, 1  
syscall  
addi \$v0, \$zero, 4 # imprimir ] =  
la \$a0, cb\_str  
syscall  
# ler valor  
addi \$v0, \$zero, 5 # syscall read integer  
syscall  
sw \$v0, ArrA(\$s1)        # A[idx] = int lido  
addi \$s0, \$s0, 1 # i++  
blt \$s0, \$s4, loopReadA # loop while i < \$s4  
  
j returnA

# Ler array B

readB:

addi \$s0, \$zero, 0        # i = 0

loopReadB:

mul \$s1, \$s0, 4        # indice do array  
# prompt do usuario  
addi \$v0, \$zero, 4 # imprimir B[  
la \$a0, msgB  
syscall  
add \$a0, \$zero, \$s0 # imprime indice  
addi \$v0, \$zero, 1  
syscall  
addi \$v0, \$zero, 4 # imprimir ] =  
la \$a0, cb\_str  
syscall  
# ler numero  
addi \$v0, \$zero, 5 # syscall read integer  
syscall  
sw \$v0, ArrB(\$s1)        # B[idx] = int lido  
addi \$s0, \$s0, 1 # i++  
blt \$s0, \$s4, loopReadB # loop while i < \$s4

exitReadB:

j returnB

# Trocar elementos de A e B

switchAB:

addi \$v0, \$zero, 4 # imprimir B[

la \$a0, msgSwitch # anunciar troca

syscall

addi \$s0, \$zero, 0 # i = 0

loopSwitchAB:

mul \$s1, \$s0, 4 # indice do array

lw \$s2, ArrA(\$s1) # tmp = A[idx]

lw \$s3, ArrB(\$s1) # tmp2 = B[idx]

sw \$s3, ArrA(\$s1) # A[idx] = tmp2

sw \$s2, ArrB(\$s1) # B[idx] = tmp

addi \$s0, \$s0, 1 # i++

blt \$s0, \$s4, loopSwitchAB # loop while i < \$s4

j returnSwitchAB

# Imprimir A e B

printAB:

# imprimir A

addi \$s0, \$zero, 0 # i = 0

j LA

LAret:

# imprimir B

addi \$s0, \$zero, 0 # i = 0

j LB

LBret:

j returnPrintAB

LA:

mul \$s1, \$s0, 4 # indice do array

addi \$v0, \$zero, 4 # imprimir A[

la \$a0, msgA

syscall

add \$a0, \$zero, \$s0 # imprime indice

addi \$v0, \$zero, 1

syscall

addi \$v0, \$zero, 4 # imprimir ] =

la \$a0, cb\_str

syscall

lw \$a0, ArrA(\$s1) # a = A[idx]

```

    addi $v0, $zero, 1 # print(a)
    syscall

    la $a0, newline
    addi $v0, $zero 4
    syscall

    addi $s0, $s0, 1 # i++
    blt $s0, $s4, LA      # loop while i < $s4
    j LAret

LB:
    mul $s1, $s0, 4      # indice do array
    addi $v0, $zero, 4 # imprimir B[
    la $a0, msgB
    syscall
    add $a0, $zero, $s0 # imprime indice
    addi $v0, $zero, 1
    syscall
    addi $v0, $zero, 4 # imprimir ] =
    la $a0, cb_str
    syscall
    lw $a0, ArrB($s1)    # a = B[idx]
    addi $v0, $zero, 1 # print(a)
    syscall

    la $a0, newline # nova linha
    addi $v0, $zero 4
    syscall

    addi $s0, $s0, 1 # i++
    blt $s0, $s4, LB      # loop while i < $s4
    j LBret

# aviso
warn:
    la $a0, msgWarn # imprimir aviso
    addi $v0, $zero 4
    syscall
    j main

exit:
    nop

```

## Capturas de tela da execução das entradas e saídas

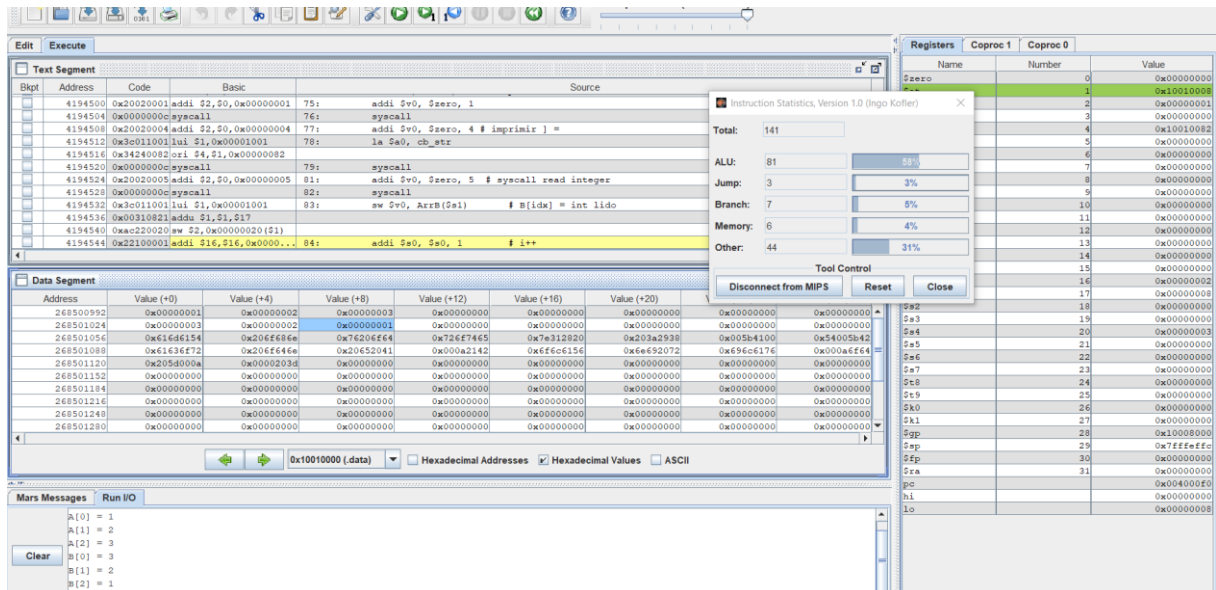
Nessa primeira imagem se tem a primeira parte do código, primeiro é inicializado o limite no registrador \$s4 como 1, em seguida é atribuído ao registrador \$a0 a

mensagem "Tamanho do vetor (1~8): "para informar qual é o limite (inferior, que é de uma unidade, e o superior, que é de oito unidades) em seguida é chamado o sistema, mudando o valor de \$v0 para 4, para que leia a mensagem e mostre para o usuário, logo em seguida se muda \$v0 para 5, para ler um número do tipo inteiro, e é chamado o sistema para ler o que for digitado, caso o número for maior que 8 ou menos que 1, irá ser imprimido a mensagem de aviso que o valor é invalido e irá pedir um novo valor ( consideremos o número 3 como tamanho de vetor, que está em \$s4).

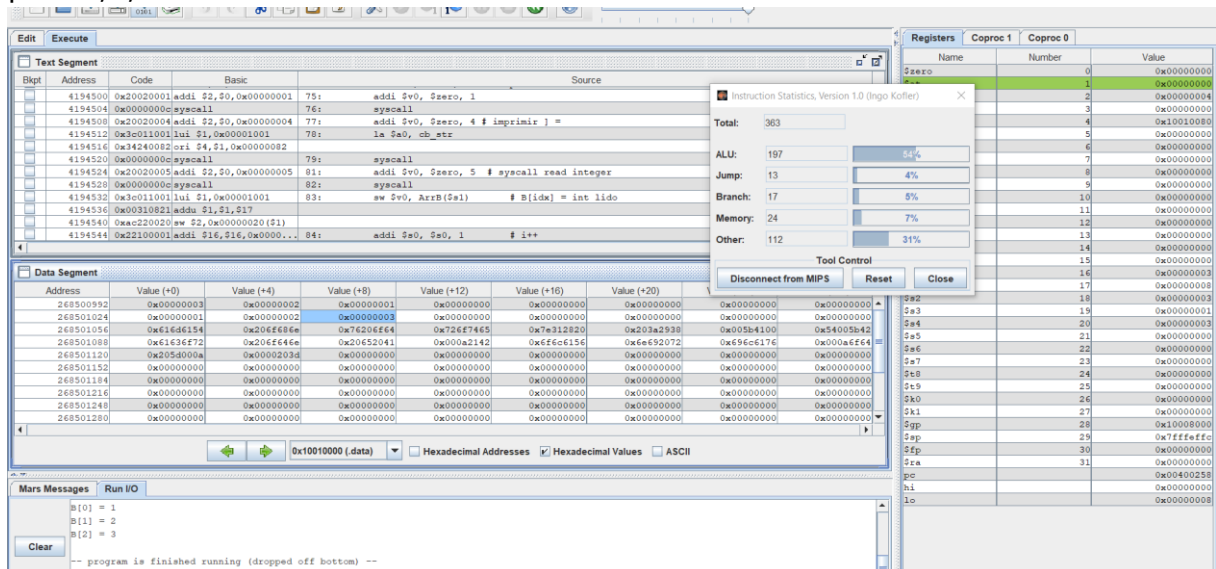
The screenshot displays the Mars MIPS simulator interface. The main window is divided into several sections:

- Text Segment:** A table showing assembly instructions with columns for Bkpt, Address, Code, Basic, and Source. The instructions include:
  - 23: `addi $s4, $zero, 1` # limite = 1
  - 24: `la $a0, msgLimite` # prompt
  - 25: `addi $v0, $zero, 4`
  - 26: `syscall`
  - 27: `addi $v0, $zero, 5` # syscall read integer
  - 28: `syscall`
  - 29: `add $s4, $zero, $v0` # limite = <valor do usuario>
  - 31: `bgt $s4, 8, warn`
  - 32: `blt $s4, 1, warn`
- Data Segment:** A table showing memory addresses and their corresponding values in hexadecimal. The values are mostly 0x00000000, with some non-zero values at specific addresses.
- Instruction Statistics:** A pop-up window titled "Instruction Statistics, Version 1.0 (Ingo Kofler)" showing the following statistics:
  - Total: 12
  - ALU: 8 (67%)
  - Jump: 0 (NaN)
  - Branch: 1 (8%)
  - Memory: 0 (NaN)
  - Other: 3 (25%)
- Mars Messages:** A section at the bottom showing the message "Tamanho do vetor (1~8): 8".

Depois se entra num laço de repetição que irá imprimir e receber os valores em determinada posição do vetor, via chamadas de sistema similar com o processo da inicialização com a diferença que terá um uma variável que irá incrementar e controlar a alocação e o armazenamento da memória do vetor até que seja feito o que foi solicitado, no caso alocar os números escolhidos pelo usuário em vetores com três posições, o Vetor A (endereço 0x1001000) será 1,2,3 e o Vetor B(endereço0x10010020) será 3,2,1.



Em seguida se entra no loop que irá trocar os números dos vetores A e B, esse loop vai armazenar os vetores nos registradores \$s2 e \$s3 respectivamente, para logo em seguida trocar, em processos de carregar(lw) e armazenar (lw) espaços do vetor, de lugar os números se baseando no índice dos vetores controlado pelo controlador do loop que vai até o número do tamanho do vetor, como se fossem fechas apontando onde o dado irá ser colocado. Vetor A foi de 1,2,3 para 3,2,1 e o Vetor B foi de 3,2,1 para 1,2,3.



Output de todos os processos para a conclusão do programa. Foi-se necessárias 363 ações para a completção do programa sendo grande maioria, 54%, de tipo aritmética ou lógico.



```

Tamanho do vetor (1~8): 3
A[0] = 1
A[1] = 2
A[2] = 3
B[0] = 3
B[1] = 2
B[2] = 1
Trocando A e B!
A[0] = 3
A[1] = 2
A[2] = 1
B[0] = 1
B[1] = 2
B[2] = 3

-- program is finished running (dropped off bottom) --

```

## Programa 02

### Código-fonte em C

```

#include <stdio.h>
#include <stdint.h>
#define i32 int32_t

int main() {
    i32 mask, days[16], p, d_select, s_select;
    while(1){
        mask = 0x1;
        printf("Dia(0~16): ");
        scanf("%d", &d_select);
        printf("Aluno(0~31): ");
        scanf("%d", &s_select);
        printf("Presenca(0/1): ");
        scanf("%d", &p);

        mask = mask << s_select;

        if(p)
            days[d_select] = days[d_select] | mask;
        else
            days[d_select] = days[d_select] & !mask;
    }
    return 0;
}

```

### Código-fonte em linguagem de montagem do MIPS

#####

# Disciplina: Arquitetura e Organização de Computadores

# Atividade: Avaliação 01 – Programação em Linguagem de Montagem

# Programa 02

# Grupo: - Eduardo Savian de Oliveira

# - Marcos Augusto Fehlaueir Pereira

# - Yuri Rodrigues

.data

ListaPresenca: 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,  
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,  
0xFFFFFFFF

msgAluno: .asciiz "Aluno(0~31):"

msgDia: .asciiz "Dia(0~15):"

msgPresenca: .asciiz "Presenca(0/1):"

.text

main:

addi \$s0, \$zero, 1 # mask = 0x00000001

addi \$s1, \$zero, 0 # dia = 0

addi \$s2, \$zero, 0 # aluno = 0

addi \$s3, \$zero, 0 # presenca = 0

loop:

addi \$s0, \$zero, 1 # mask = 0x00000001

# ler dia

addi \$v0, \$zero, 4

la \$a0, msgDia

syscall

addi \$v0, \$zero, 5

syscall

add \$s1, \$zero, \$v0

# ler aluno

addi \$v0, \$zero, 4

la \$a0, msgAluno

syscall

addi \$v0, \$zero, 5

syscall

add \$s2, \$zero, \$v0

# ler presenca

addi \$v0, \$zero, 4

la \$a0, msgPresenca

syscall

addi \$v0, \$zero, 5

syscall

add \$s3, \$zero, \$v0

```

sllv $s0, $s0, $s2 # mask = mask << aluno
mul $s4, $s1, 4 # calcular indice do array (dia * 4)
lw $s6, ListaPresenca($s4) # tmp = Array[idx]
beqz $s3, falta # if presenca == 0; marcarFalta tmp & !mask
or $s6, $s6, $s0 # else marcarPresenca {tmp | mask}
faltaReturn:
sw $s6, ListaPresenca($s4)
j loop

```

```
#inverte mascara, faz AND
nor $s0, $zero, $s0
and $s6, $s6, $s0
j faltaReturn
```

nop

Se é indicado a posição todas as posições em 0xffffffff indicando presença por padrão. Primeiro o usuário irá entrar em loop que vai imprimir e receber dados, via chamadas de sistema do tipo 4 e 5 em \$v0, para o dia escolhido, depois o número do aluno e se ele estava ou não presente.

26f501056	0x6f756c41	0x03028ff	0x3a293133	0x61694400	0x317e3028	0x003a2935	0x73657250	0x61636ae5
26f501088	0x312f3028	0x000003a29	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501120	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501152	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501184	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501216	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501248	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000
26f501280	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000	0x000000000

0x10010000 (.data) 
 ☐ Hexadecimal Addresses 
 ☒ Hexadecimal Values

---

### Mars Messages

Run I/O

```
Dia (0~15): 1
Aluno (0~31): 1
Presenca (0/1): 0
```

Clear

### Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 39

ALU: 26

Jump: 1

Branch: 1

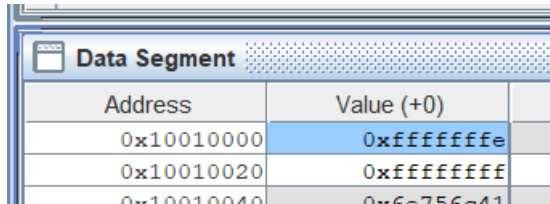
Memory: 2

Other: 9

Tool Control

Disconnect from MIPS Reset Close

Caso seja uma falta (0), o processo irá para uma instrução que fará uma máscara negada com NOR \$zero e depois aplicará um AND entre a máscara e o número, em caso de presença será feito um OR marcando assim a presença. Em ambos os casos o programa irá colocar no vetor dos dias na memória do aluno que ele levou uma falta. O loop continuará até se pare a aplicação ou seja cometido alguma ação que leve a um erro.



Address	Value (+0)
0x10010000	0xfffffffffe
0x10010020	0xffffffffff
0x10010040	0x5a756a41

Antes de ser falta era 0xffffffff, depois se tornou 0xfffffffffe, indicando que o 1º aluno faltou e os outros compareceram. Em um ciclo para atribuição da presença ou não foram usados 26 comandos, 67%, do tipo aritmética ou lógico e o restante de outros tipos além de um jump e de memória.