UNIVERSIDADE DO VALE DO ITAJAÍ
CAMPUS KOBRASOL – SÃO JOSÉ
CIÊNCIA DA COMPUTAÇÃO

Arquitetura de Computadores

Avaliação 01 – Programação em linguagem de montagem

Professor: Douglas Rossi de Melo
Aluno: Matheus Henrique Schaly

Data de entrega: 15/09/2017

C++ 1

```
// Disciplina: Arquitetura e Organização de Computadores
// Atividade: Avaliação 01 – Programação em C++
// Programa 01
// Grupo: - Matheus Henrique Schaly

#include <iostream>

using namespace std;

int main()
{
    int vector1[8], vector2[8];
    int numElementos;
    while (true) {
        cout << "Enter with array's size (max. = 8):\n";
        cin >> numElementos;
        if (numElementos > 0 && numElementos < 9) {
            break;
        }
        cout << "Invalid value.\n";
    }
    for (int i = 0; i < numElementos; i++) {
        cout << "Vector1[" << i << "] = ";
        cin >> vector1[i];
    }
    for (int i = 0; i < numElementos; i++) {
        cout << "Vector2[" << i << "] = ";
        cin >> vector2[i];
    }
    int temp;
    for (int i = 0; i < numElementos; i++) {
        temp = vector1[i];
        vector1[i] = vector2[i];
        vector2[i] = temp;
    }
    cout << "\n";
    for (int i = 0; i < numElementos; i++) {
        cout << "Vetor1[" << i << "] = " << vector1[i] << "\n";
    }
    for (int i = 0; i < numElementos; i++) {
        cout << "Vetor2[" << i << "] = " << vector2[i] << "\n";
    }
}
```

Assembly MIPS 1

```
# Disciplina: Arquitetura e Organização de Computadores
# Atividade: Avaliação 01 – Programação em Linguagem de Montagem
# Programa 01
# Grupo: - Matheus Henrique Schaly

.data

        #Creates RAM variables
        vector1:        .word  0, 0, 0, 0, 0, 0, 0, 0  #Inicializing vector 1
        vector2:        .word  0, 0, 0, 0, 0, 0, 0, 0  #Inicializing vector 2
        mensagem1:.asciiz "Enter with array's size (max. = 8):\n"     #Message prompting vectors size
        mensagem2:.asciiz "Invalid value.\n"     #Message warning invalid value
        mensagem3:.asciiz "Vector1["
        mensagem4:.asciiz "] = "
        mensagem5:.asciiz "Vector2["
        mensagem6:.asciiz "\n"              #Jump line

.text

        #Creates while's bounds
        addi    $t1,    $zero, 0            #Store 0 at t1 (while's lower bound)
        addi    $t2,    $zero, 9            #Store 9 at t2 (while's upper bound)

        #Prompt arrays' size
while:
        #Prints mensagem1
        li      $v0,    4                   #Command to print a text
        la      $a0,    mensagem1           #Load address of mensagem1 to a0
        syscall                             #Do it

        #Reads integer
        li      $v0,    5                   #Read an integer and store it in v0
        syscall                             #Do it

        #Stores integer
        move  $t0,      $v0                 #Move to t0 (arrays' size) the integer in v0

        #While's first if condition
        bgt     $t0     $t1     secondCondition     #Branch to secondCondition if t0 (arrays' size) is greater than t1 (while's lower bound)

        # Prints mensagem2
        li      $v0,    4                   #Command to print a text
        la      $a0,    mensagem2           #Load address of mensagem2 to a0
        syscall                             #Do it

        #Restarts while loop
        j while                                 #Jump to while

        #While's second if condition
secondCondition:
        blt     $t0     $t2     exit            #Branch to exit if t0 (arrays' size) is less than t2
(while's lower bound)

        # Prints mensagem2
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem2               #Load address of mensagem2 to a0
        syscall                                 #Do it

        #Restarts while
        j       while                           #Jump to while
exit:

        #Loads array1's base address
        la      $t2,    vector1                 #Load array1's base address to t2 (array1's base
address)

        #Loads array2's base address
        la      $t5,    vector2                 #Load array2's base address to t5 (array2's base
address)

        #Creates loop's index
        addi    $t1,    $zero, 0                #Add zero and 0 and store it in t1 (loop's index)

        #First for loop to gather array1's values
for1:
        #Loop's if condition
        beq     $t1,    $t0,    for1Exit        #Branch to for1Exit if t1 (loop's index) is equal to
t0 (arrays' size)

        #Prints mensagem3
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem3               #Load address of mensagem3 to a0
        syscall                                 #Do it

        #Prints index
        li      $v0,    1                       #Command to print a integer
        move    $a0,    $t1                     #Move t1 (loop index) to a0
        syscall                                 #Do it

        #Prints mensagem4
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem4               #Load address of mensagem4 to a0
        syscall                                 #Do it

        #Reads integer
        li      $v0,    5                       #Read an integer and store it in v0 (it now has the
input)
        syscall                                 #Do it
```

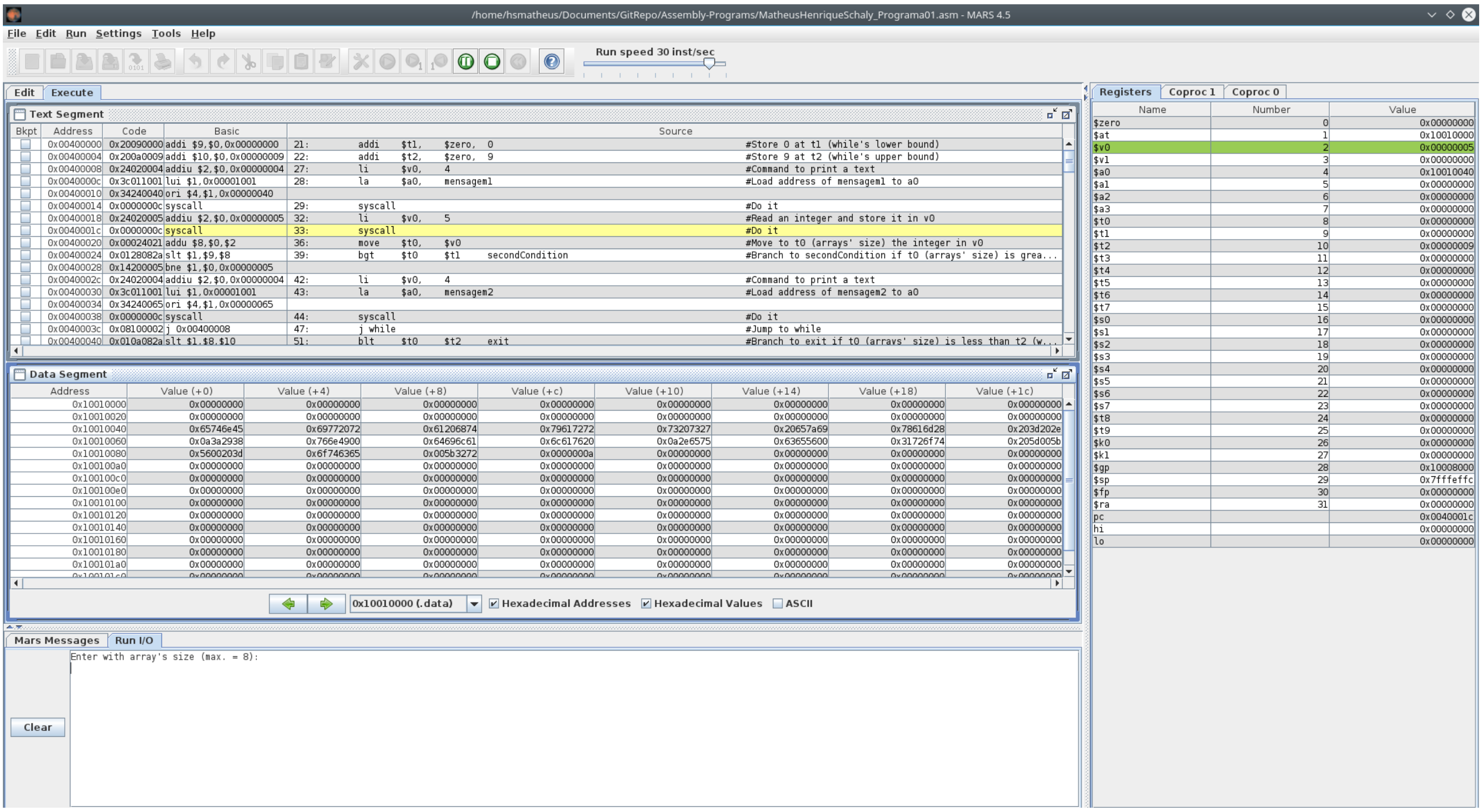
```
	#Calculates the array1's addres to store the integer
	sll	$t3,	$t1,	2		#Multiply t1 (loop's index) by 4 and put the result into t3 (bytes to be moved from array's base address)
	add	$t3,	$t3,	$t2		#Add t2 (array's base) and t3 (bytes to be moved from array's base address) and put it back into t3 (array's fully calculated address)

	#Stores the input in array
	sw	$v0,	($t3)			#Store word from v0 (that has the imput) in t3 (array's fully calculated address)

	#Increases loop's index
	addi	$t1,	$t1,	1		#Increase t1 (loop's index) by 1

	#Restarts for loop
	j	for1				#Jump to for1

	#Exits first for loop
for1Exit:

	#Resets loop's index to 0
	addi	$t1,	$zero,	0		#Add zero and 0 and store it in t1 (loop's index)

	#Second for loop to gather array2's values
for2:
	#Loop's if condition
	beq	$t1,	$t0,	for2Exit	#Branch to for2Exit if t1 (loop's index) is equal to t0 (arrays' size)

	#Prints mensagem3
	li	$v0,	4			#Command to print a text
	la	$a0,	mensagem5		#Load address of mensagem3 to a0
	syscall					#Do it

	#Prints index
	li	$v0,	1			#Command to print a integer
	move	$a0,	$t1			#Move t1 (loop index) to a0
	syscall					#Do it

	#Prints mensagem4
	li	$v0,	4			#Command to print a text
	la	$a0,	mensagem4		#Load address of mensagem4 to a0
	syscall					#Do it

	#Reads integer
	li	$v0,	5			#Read an integer and store it in v0 (it now has the input)
	syscall					#Do it

	#Calculates the array2's addres to store the integer
	sll	$t3,	$t1,	2		#Multiply t1 (loop's index) by 4 and put the result into t3 (bytes to be moved from array's base address)
```

```
        add     $t3,    $t3,    $t5             #Add t5 (array's base) and t3 (bytes to be moved
from array's base address) and put it back into t3 (array's fully calculated address)

        #Stores the input in array
        sw      $v0,    ($t3)                   #Store word from v0 (that has the imput) in t3
(array's fully calculated address)

        #Increases loop's index
        addi    $t1,    $t1,    1               #Increase t1 (loop's index) by 1

        #Restarts for loop
        j       for2                            #Jump to for2

        #Exits second for loop
for2Exit:

        #Resets loop's index to 0
        addi    $t1,    $zero,  0               #Add zero and 0 and store it in t1 (loop's index)

        #Third for loop to swap arrays' values
for3:
        #Loop's if condition
        beq     $t1,    $t0,    for3Exit        #Branch to for3Exit if t1 (loop's index) is equal to
t0 (arrays' size)

        #Calculates the array1's addres to load an integer
        sll     $t3,    $t1,    2               #Multiply t1 (loop's index) by 4 and put the result
into t3 (bytes to be moved from array's base address)
        add     $t3,    $t3,    $t2             #Add t2 (array1's base) and t3 (bytes to be moved
from array1's base address) and put it back into t3 (array1's fully calculated address)

        #Loads the input from array1
        lw      $t4,    ($t3)                   #Load word from t3 (array1's fully calculated
address) to t4 (array1's value)

        #Calculates the array2's addres to load an integer
        sll     $t6,    $t1,    2               #Multiply t1 (loop's index) by 4 and put the result
into t6 (bytes to be moved from array's base address)
        add     $t6,    $t6,    $t5             #Add t5 (array2's base) and t6 (bytes to be moved
from array2's base address) and put it back into t6 (array2's fully calculated address)

        #Loads the input from array2
        lw      $t7,    ($t6)                   #Load word from t6 (array2's fully calculated
address) to t7 (array2's value)

        #Stores array1's value in array2's
        sw      $t4,    ($t6)                   #Store word from t4 (array1's value) in t6 (array2's
fully calculated address)

        #Stores array2's value in array1's
        sw      $t7,    ($t3)                   #Store word from t7 (array2's value) in t3 (array1's
fully calculated address)
```

```
        #Increases loop's index
        addi    $t1,    $t1,    1               #Increase t1 (loop's index) by 1

        #Restarts for loop
        j       for3                            #Jump to for3

        #Exits third for loop
for3Exit:

        #Prints mensagem6
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem6               #Load address of mensagem6 to a0
        syscall                                 #Do it

        #Resets loop's index to 0
        addi    $t1,    $zero, 0                #Add zero and 0 and store it in t1 (loop's index)

        #Forth for loop to print array1's values
for4:
        #Loop's if condition
        beq     $t1,    $t0,    for4Exit        #Branch to for4Exit if t1 (loop's index) is equal to
t0 (arrays' size)

        #Prints mensagem3
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem3               #Load address of mensagem3 to a0
        syscall                                 #Do it

        #Prints index
        li      $v0,    1                       #Command to print a integer
        move    $a0,    $t1                     #Move $t1 (loop index) to a0
        syscall                                 #Do it

        #Prints mensagem4
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem4               #Load address of mensagem4 to a0
        syscall                                 #Do it

        #Calculates the array1's addres to load an integer
        sll     $t3,    $t1,    2               #Multiply t1 (loop's index) by 4 and put the result
into t3 (bytes to be moved from array's base address)
        add     $t3,    $t3,    $t2             #Add t2 (array1's base) and t3 (bytes to be moved
from array's base address) and put it back into t3 (array's fully calculated address)

        #Loads the input from array1
        lw      $t4,    ($t3)                   #Load word from t3 (array1's fully calculated
address) to t4 (array1's value)

        #Prints array1 at index t1
        li      $v0,    1                       #Command to print a integer
        move    $a0,    $t4                     #Move t4 (the value) to a0
        syscall                       #Do it

        #Prints mensagem6
        li     $v0,  4                #Command to print a text
        la     $a0,  mensagem6        #Load address of mensagem6 to a0
        syscall                       #Do it

        #Increases loop's index
        addi   $t1,  $t1,  1          #Increase t1 (loop's index) by 1

        #Restarts for loop
        j      for4                   #Jump to for4

        #Exits forth for loop
for4Exit:

        #Resets loop's index to 0
        addi   $t1,  $zero, 0         #Add zero and 0 and store it in t1 (loop's index)

        #Fifth for loop to print array2's values
for5:
        #Loop's if condition
        beq    $t1,  $t0,  for5Exit   #Branch to for5Exit if t1 (loop's index) is equal to
t0 (arrays' size)

        #Prints mensagem3
        li     $v0,  4                #Command to print a text
        la     $a0,  mensagem5        #Load address of mensagem5 to a0
        syscall                       #Do it

        #Prints index
        li     $v0,  1                #Command to print a integer
        move   $a0,  $t1              #Move $t1 (loop index) to a0
        syscall                       #Do it

        #Prints mensagem4
        li     $v0,  4                #Command to print a text
        la     $a0,  mensagem4        #Load address of mensagem4 to a0
        syscall                       #Do it

        #Calculates the array2's addres to load an integer
        sll    $t3,  $t1,  2          #Multiply t1 (loop's index) by 4 and put the result
into t3 (bytes to be moved from array's base address)
        add    $t3,  $t3,  $t5        #Add t5 (array2's base) and t3 (bytes to be moved
from array's base address) and put it back into t3 (array's fully calculated address)

        #Loads the input from array2
        lw     $t4,  ($t3)            #Load word from t3 (array1's fully calculated
address) to t4 (array1's value)

        #Prints array2 at index t1
        li     $v0,  1                #Command to print a integer
        move    $a0,    $t4                     #Move t4 (the value) to a0
        syscall                                 #Do it

        #Prints mensagem6
        li      $v0,    4                       #Command to print a text
        la      $a0,    mensagem6               #Load address of mensagem6 to a0
        syscall                                 #Do it

        #Increases loop's index
        addi    $t1,    $t1,    1               #Increase t1 (loop's index) by 1

        #Restarts for loop
        j       for5                            #Jump to for5

        #Exits fifth for loop
for5Exit:
```

Capturas de Tela 1:

Pergunta o tamanho do vetor:


/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa01.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Edit
Execute
Text Segment
Data Segment
Registers
Coproc 1
Coproc 0
Mars Messages
Run I/O
Clear

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x20090000	addi $9,$0,0x00000000	21: addi $t1, $zero, 0 #Store 0 at t1 (while's lower bound)
☐	0x00400004	0x200a0009	addi $10,$0,0x00000009	22: addi $t2, $zero, 9 #Store 9 at t2 (while's upper bound)
☐	0x00400008	0x24020004	addiu $2,$0,0x00000004	27: li $v0, 4 #Command to print a text
☐	0x0040000c	0x3c011001	lui $1,0x00001001	28: la $a0, mensagem1 #Load address of mensagem1 to a0
☐	0x00400010	0x34240040	ori $4,$1,0x00000040	
☐	0x00400014	0x0000000c	syscall	29: syscall #Do it
☐	0x00400018	0x24020005	addiu $2,$0,0x00000005	32: li $v0, 5 #Read an integer and store it in v0
☐	0x0040001c	0x0000000c	syscall	33: syscall #Do it
☐	0x00400020	0x00024021	addu $8,$0,$2	36: move $t0, $v0 #Move to t0 (arrays' size) the integer in v0
☐	0x00400024	0x0128082a	slt $1,$9,$8	39: bgt $t0 $t1 secondCondition #Branch to secondCondition if t0 (arrays' size) is grea...
☐	0x00400028	0x14200005	bne $1,$0,0x00000005	
☐	0x0040002c	0x24020004	addiu $2,$0,0x00000004	42: li $v0, 4 #Command to print a text
☐	0x00400030	0x3c011001	lui $1,0x00001001	43: la $a0, mensagem2 #Load address of mensagem2 to a0
☐	0x00400034	0x34240065	ori $4,$1,0x00000065	
☐	0x00400038	0x0000000c	syscall	44: syscall #Do it
☐	0x0040003c	0x08100002	j 0x00400008	47: j while #Jump to while
☐	0x00400040	0x010a082a	slt $1,$8,$10	51: blt $t0 $t2 exit #Branch to exit if t0 (arrays' size) is less than t2 (w...

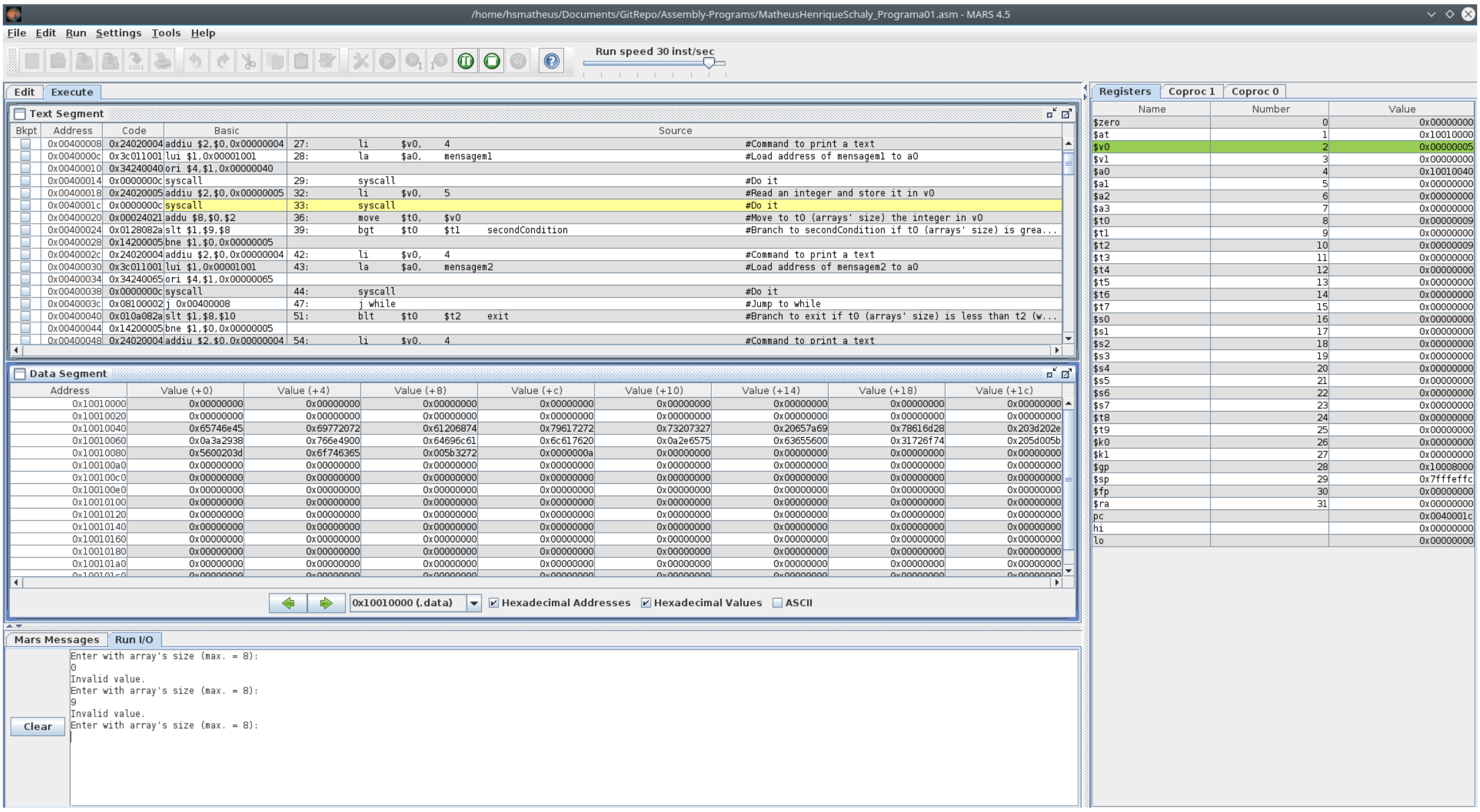
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x65746e45	0x69772072	0x61206874	0x79617272	0x73207327	0x20657a69	0x78616d28	0x203d202e
0x10010060	0x0a3a2938	0x766e4900	0x64696c61	0x6c617620	0x0a2e6575	0x63655600	0x31726f74	0x205d005b
0x10010080	0x5600203d	0x6f746365	0x005b3272	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010040
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000009
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040001c
hi		0x00000000
lo		0x00000000

```
Enter with array's size (max. = 8):
```

Possíveis valores inválidos:


/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa01.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Edit
Execute
Text Segment
Data Segment
Registers
Coproc 1
Coproc 0
Mars Messages
Run I/O
Clear

Bkpt	Address	Code	Basic	Source
☐	0x00400008	0x24020004	addiu $2,$0,0x00000004	27: li $v0, 4 #Command to print a text
☐	0x0040000c	0x3c011001	lui $1,0x00001001	28: la $a0, mensagem1 #Load address of mensagem1 to a0
☐	0x00400010	0x34240040	ori $4,$1,0x00000040	
☐	0x00400014	0x0000000c	syscall	29: syscall #Do it
☐	0x00400018	0x24020005	addiu $2,$0,0x00000005	32: li $v0, 5 #Read an integer and store it in v0
☐	0x0040001c	0x0000000c	syscall	33: syscall #Do it
☐	0x00400020	0x00024021	addu $8,$0,$2	36: move $t0, $v0 #Move to t0 (arrays' size) the integer in v0
☐	0x00400024	0x0128082a	slt $1,$9,$8	39: bgt $t0 $t1 secondCondition #Branch to secondCondition if t0 (arrays' size) is grea...
☐	0x00400028	0x14200005	bne $1,$0,0x00000005	
☐	0x0040002c	0x24020004	addiu $2,$0,0x00000004	42: li $v0, 4 #Command to print a text
☐	0x00400030	0x3c011001	lui $1,0x00001001	43: la $a0, mensagem2 #Load address of mensagem2 to a0
☐	0x00400034	0x34240065	ori $4,$1,0x00000065	
☐	0x00400038	0x0000000c	syscall	44: syscall #Do it
☐	0x0040003c	0x08100002	j 0x00400008	47: j while #Jump to while
☐	0x00400040	0x010a082a	slt $1,$8,$10	51: blt $t0 $t2 exit #Branch to exit if t0 (arrays' size) is less than t2 (w...
☐	0x00400044	0x14200005	bne $1,$0,0x00000005	
☐	0x00400048	0x24020004	addiu $2,$0,0x00000004	54: li $v0, 4 #Command to print a text

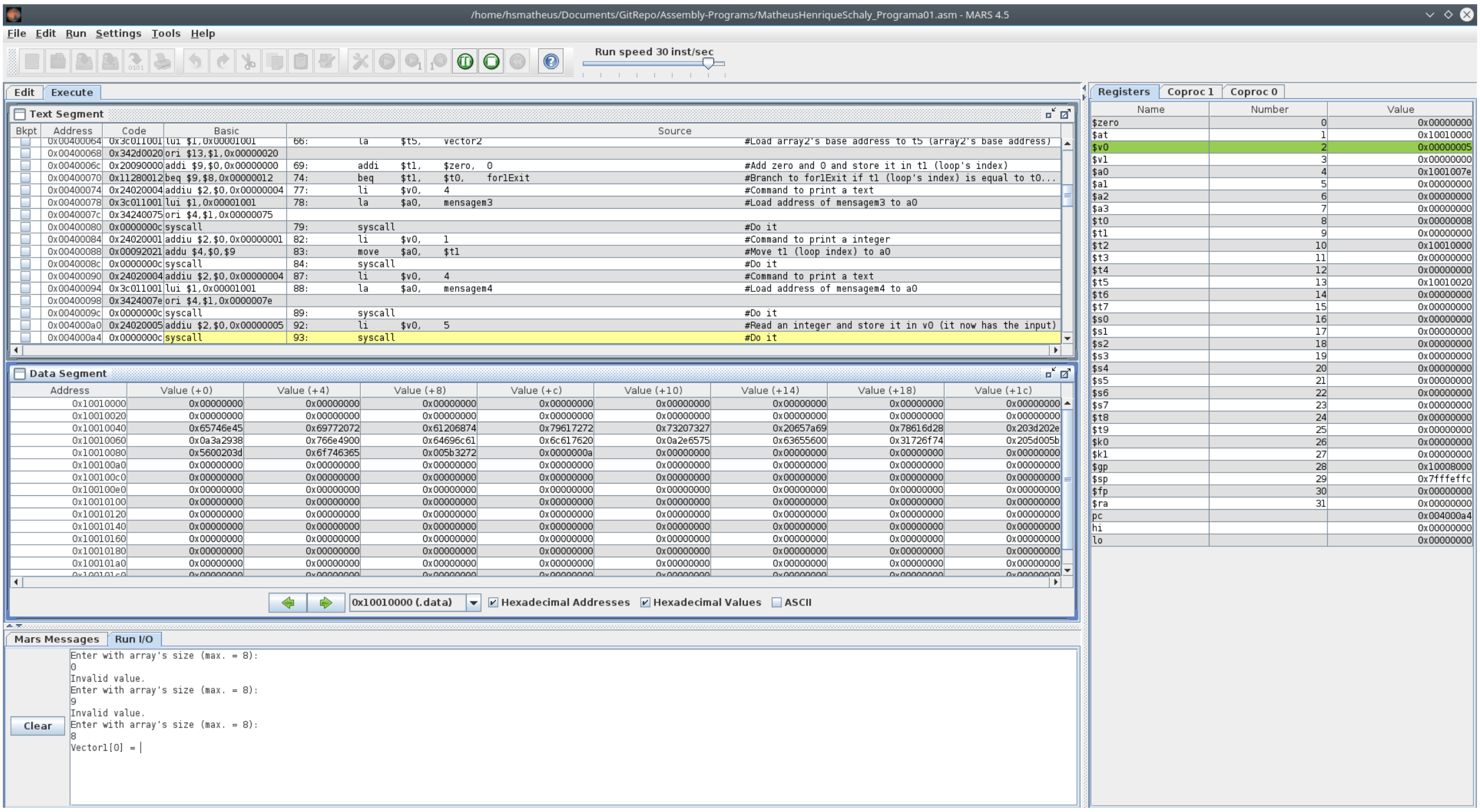
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x65746e45	0x69772072	0x61206874	0x79617272	0x73207327	0x20657a69	0x78616d28	0x203d202e
0x10010060	0x0a3a2938	0x766e4900	0x64696c61	0x6c617620	0x0a2e6575	0x63655600	0x31726f74	0x205d005b
0x10010080	0x5600203d	0x6f746365	0x005b3272	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010040
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000009
$t1	9	0x00000000
$t2	10	0x00000009
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040001c
hi		0x00000000
lo		0x00000000

```
Enter with array's size (max. = 8):
0
Invalid value.
Enter with array's size (max. = 8):
9
Invalid value.
Enter with array's size (max. = 8):
```

Entrada de um valor correto:



Bkpt	Address	Code	Basic	Source
☐	0x00400064	0x3c011001	lui $1,0x00001001	66: la $t5, vector2 #Load array2's base address to t5 (array2's base address)
☐	0x00400068	0x342d0020	ori $13,$1,0x00000020	
☐	0x0040006c	0x20090000	addi $9,$0,0x00000000	69: addi $t1, $zero, 0 #Add zero and 0 and store it in t1 (loop's index)
☐	0x00400070	0x11280012	beq $9,$8,0x00000012	74: beq $t1, $t0, for1Exit #Branch to for1Exit if t1 (loop's index) is equal to t0...
☐	0x00400074	0x24020004	addiu $2,$0,0x00000004	77: li $v0, 4 #Command to print a text
☐	0x00400078	0x3c011001	lui $1,0x00001001	78: la $a0, mensagem3 #Load address of mensagem3 to a0
☐	0x0040007c	0x34240075	ori $4,$1,0x00000075	
☐	0x00400080	0x0000000c	syscall	79: syscall #Do it
☐	0x00400084	0x24020001	addiu $2,$0,0x00000001	82: li $v0, 1 #Command to print a integer
☐	0x00400088	0x00092021	addu $4,$0,$9	83: move $a0, $t1 #Move t1 (loop index) to a0
☐	0x0040008c	0x0000000c	syscall	84: syscall #Do it
☐	0x00400090	0x24020004	addiu $2,$0,0x00000004	87: li $v0, 4 #Command to print a text
☐	0x00400094	0x3c011001	lui $1,0x00001001	88: la $a0, mensagem4 #Load address of mensagem4 to a0
☐	0x00400098	0x3424007e	ori $4,$1,0x0000007e	
☐	0x0040009c	0x0000000c	syscall	89: syscall #Do it
☐	0x004000a0	0x24020005	addiu $2,$0,0x00000005	92: li $v0, 5 #Read an integer and store it in v0 (it now has the input)
☐	0x004000a4	0x0000000c	syscall	93: syscall #Do it

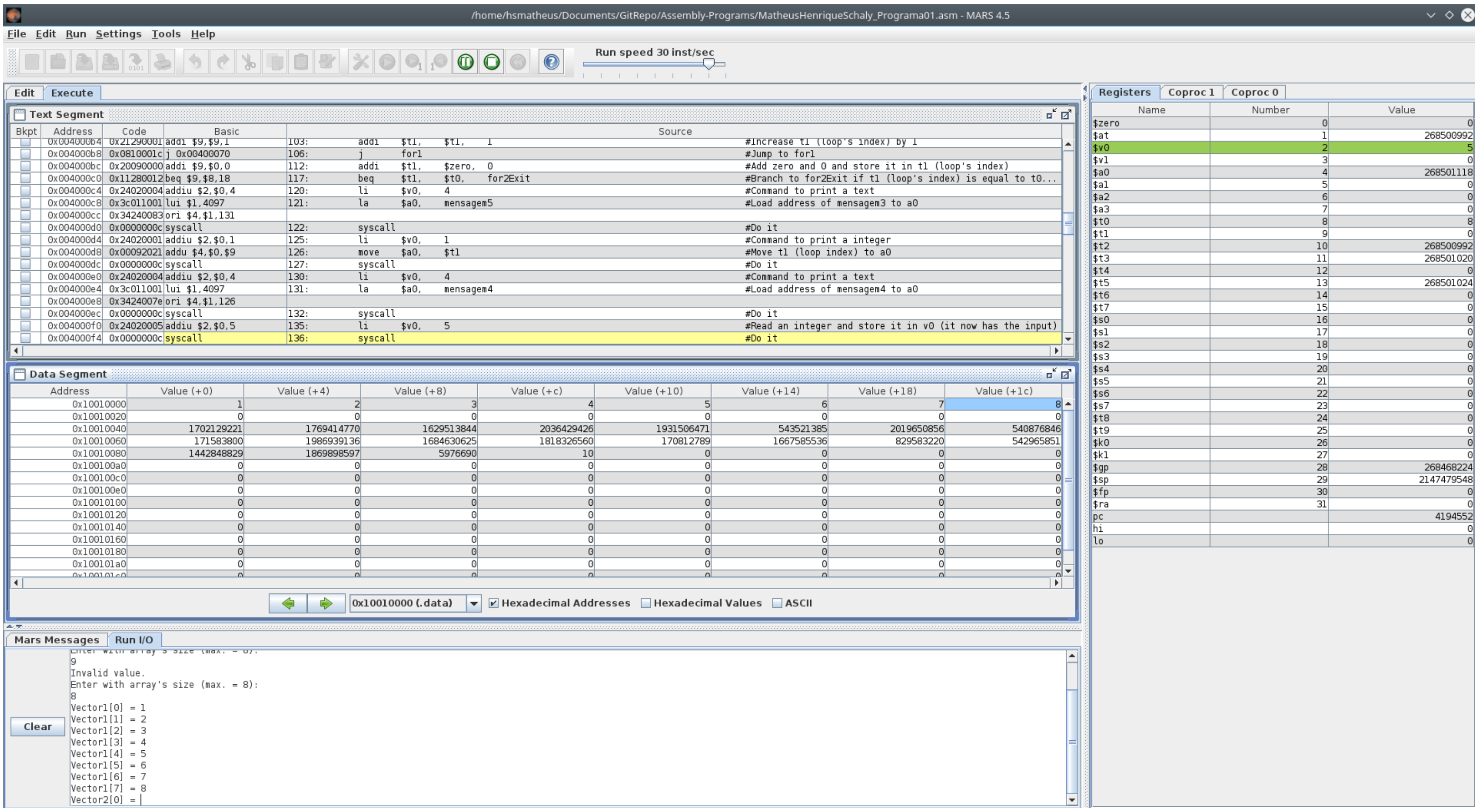
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x65746e45	0x69772072	0x61206874	0x79617272	0x73207327	0x20657a69	0x78616d28	0x203d202e
0x10010060	0x0a3a2938	0x766e4900	0x64696c61	0x6c617620	0x0a2e6575	0x63655600	0x31726f74	0x205d005b
0x10010080	0x5600203d	0x6f746365	0x005b3272	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

```
Enter with array's size (max. = 8):
0
Invalid value.
Enter with array's size (max. = 8):
9
Invalid value.
Enter with array's size (max. = 8):
8
Vector1[0] = |
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x1001007e
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000008
$t1	9	0x00000000
$t2	10	0x10010000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x10010020
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x004000a4
hi		0x00000000
lo		0x00000000

Entrada de valores no vetor1:



Bkpt	Address	Code	Basic	Source
☐	0x004000b4	0x21290001	addi $9,$9,1	103: addi $t1, $t1, 1 #Increase t1 (loop's index) by 1
☐	0x004000b8	0x0810001c	j 0x00400070	106: j for1 #Jump to for1
☐	0x004000bc	0x20090000	addi $9,$0,0	112: addi $t1, $zero, 0 #Add zero and 0 and store it in t1 (loop's index)
☐	0x004000c0	0x11280012	beq $9,$8,18	117: beq $t1, $t0, for2Exit #Branch to for2Exit if t1 (loop's index) is equal to t0...
☐	0x004000c4	0x24020004	addiu $2,$0,4	120: li $v0, 4 #Command to print a text
☐	0x004000c8	0x3c011001	lui $1,4097	121: la $a0, mensagem5 #Load address of mensagem3 to a0
☐	0x004000cc	0x34240083	ori $4,$1,131	
☐	0x004000d0	0x0000000c	syscall	122: syscall #Do it
☐	0x004000d4	0x24020001	addiu $2,$0,1	125: li $v0, 1 #Command to print a integer
☐	0x004000d8	0x00092021	addu $4,$0,$9	126: move $a0, $t1 #Move t1 (loop index) to a0
☐	0x004000dc	0x0000000c	syscall	127: syscall #Do it
☐	0x004000e0	0x24020004	addiu $2,$0,4	130: li $v0, 4 #Command to print a text
☐	0x004000e4	0x3c011001	lui $1,4097	131: la $a0, mensagem4 #Load address of mensagem4 to a0
☐	0x004000e8	0x3424007e	ori $4,$1,126	
☐	0x004000ec	0x0000000c	syscall	132: syscall #Do it
☐	0x004000f0	0x24020005	addiu $2,$0,5	135: li $v0, 5 #Read an integer and store it in v0 (it now has the input)
☐	0x004000f4	0x0000000c	syscall	136: syscall #Do it

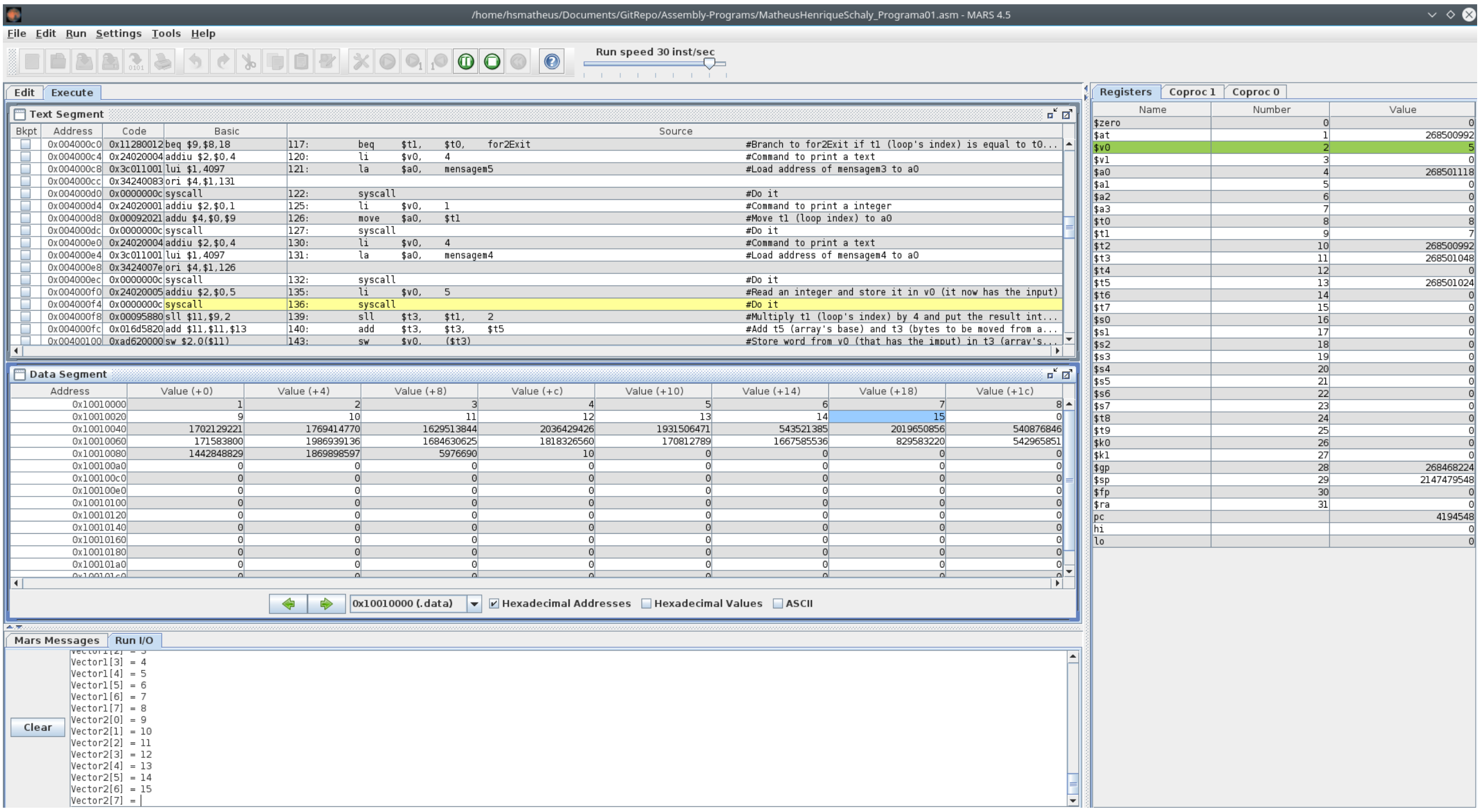
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	6	7	8
0x10010020	0	0	0	0	0	0	0	0
0x10010040	1702129221	1769414770	1629513844	2036429426	1931506471	543521385	2019650856	540876846
0x10010060	171583800	1986939136	1684630625	1818326560	170812789	1667585536	829583220	542965851
0x10010080	1442848829	1869898597	5976690	10	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

```
Enter with array's size (max. = 8):
9
Invalid value.
Enter with array's size (max. = 8):
8
Vector1[0] = 1
Vector1[1] = 2
Vector1[2] = 3
Vector1[3] = 4
Vector1[4] = 5
Vector1[5] = 6
Vector1[6] = 7
Vector1[7] = 8
Vector2[0] = |
```

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	5
$v1	3	0
$a0	4	268501118
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	0
$t2	10	268500992
$t3	11	268501020
$t4	12	0
$t5	13	268501024
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194552
hi		0
lo		0

Entrada de valores no vetor2:



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa01.asm - MARS 4.5

Text Segment

```
Bkpt Address     Code       Basic              Source
□    0x004000c0  0x11280012 beq $9,$8,18       117:   beq     $t1,   $t0,   for2Exit   #Branch to for2Exit if t1 (loop's index) is equal to t0...
□    0x004000c4  0x24020004 addiu $2,$0,4      120:   li      $v0,   4                 #Command to print a text
□    0x004000c8  0x3c011001 lui $1,4097        121:   la      $a0,   mensagem5         #Load address of mensagem3 to a0
□    0x004000cc  0x34240083 ori $4,$1,131
□    0x004000d0  0x0000000c syscall            122:   syscall                          #Do it
□    0x004000d4  0x24020001 addiu $2,$0,1      125:   li      $v0,   1                 #Command to print a integer
□    0x004000d8  0x00092021 addu $4,$0,$9      126:   move    $a0,   $t1               #Move t1 (loop index) to a0
□    0x004000dc  0x0000000c syscall            127:   syscall                          #Do it
□    0x004000e0  0x24020004 addiu $2,$0,4      130:   li      $v0,   4                 #Command to print a text
□    0x004000e4  0x3c011001 lui $1,4097        131:   la      $a0,   mensagem4         #Load address of mensagem4 to a0
□    0x004000e8  0x3424007e ori $4,$1,126
□    0x004000ec  0x0000000c syscall            132:   syscall                          #Do it
□    0x004000f0  0x24020005 addiu $2,$0,5      135:   li      $v0,   5                 #Read an integer and store it in v0 (it now has the input)
□    0x004000f4  0x0000000c syscall            136:   syscall                          #Do it
□    0x004000f8  0x00095880 sll $11,$9,2       139:   sll     $t3,   $t1,   2          #Multiply t1 (loop's index) by 4 and put the result int...
□    0x004000fc  0x016d5820 add $11,$11,$13    140:   add     $t3,   $t3,   $t5        #Add t5 (array's base) and t3 (bytes to be moved from a...
□    0x00400100  0xad620000 sw $2,0($11)       143:   sw      $v0,   ($t3)             #Store word from v0 (that has the imput) in t3 (array's...
```

Data Segment

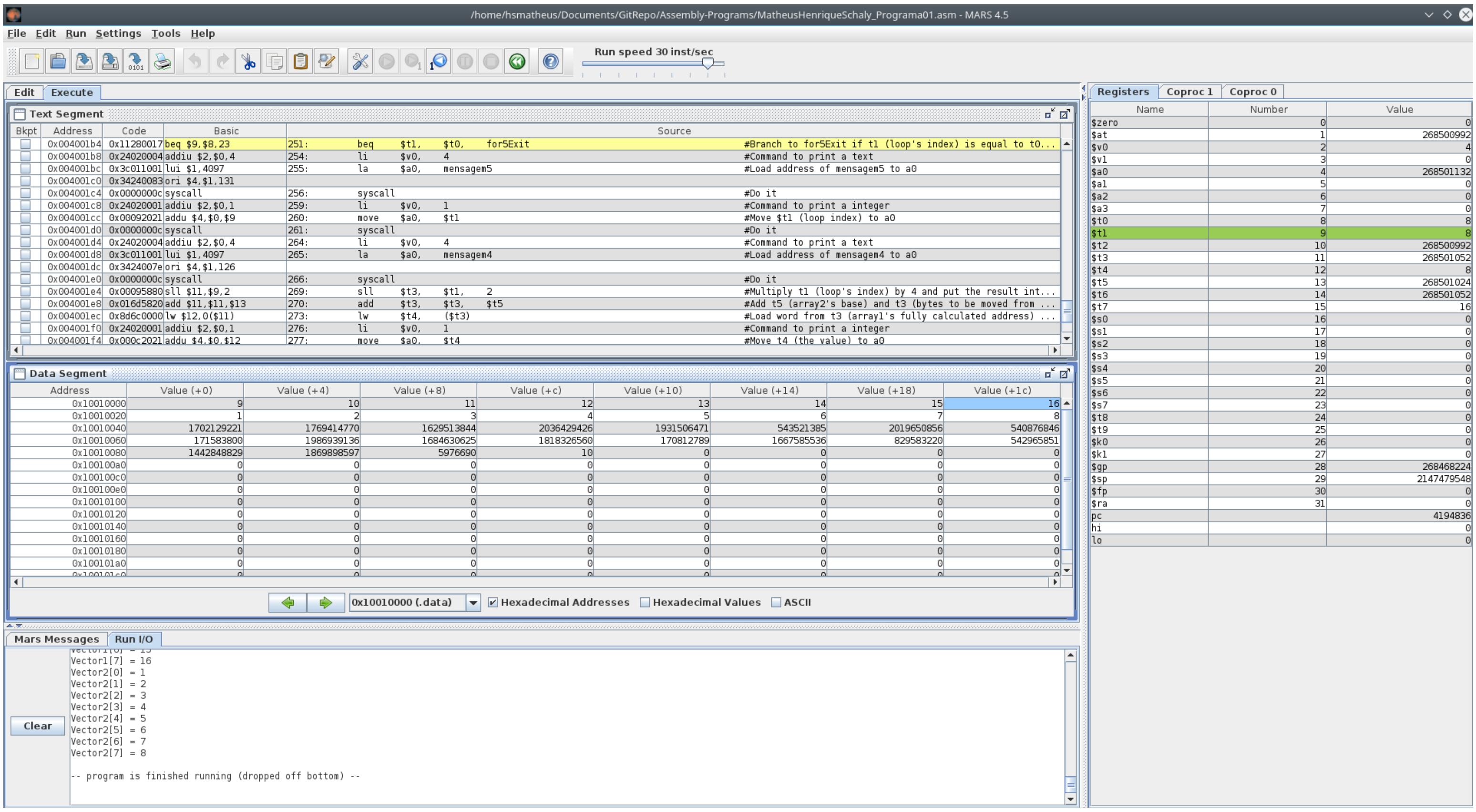
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	6	7	8
0x10010020	9	10	11	12	13	14	15	0
0x10010040	1702129221	1769414770	1629513844	2036429426	1931506471	543521385	2019650856	540876846
0x10010060	171583800	1986939136	1684630625	1818326560	170812789	1667585536	829583220	542965851
0x10010080	1442848829	1869898597	5976690	10	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	5
$v1	3	0
$a0	4	268501118
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	7
$t2	10	268500992
$t3	11	268501048
$t4	12	0
$t5	13	268501024
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194548
hi		0
lo		0

```
Vector1[2] = 3
Vector1[3] = 4
Vector1[4] = 5
Vector1[5] = 6
Vector1[6] = 7
Vector1[7] = 8
Vector2[0] = 9
Vector2[1] = 10
Vector2[2] = 11
Vector2[3] = 12
Vector2[4] = 13
Vector2[5] = 14
Vector2[6] = 15
Vector2[7] = 
```

Resultado final, com valores já trocados:



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa01.asm - MARS 4.5

Text Segment

```
Bkpt Address     Code       Basic              Source
□    0x004001b4  0x11280017 beq $9,$8,23       251:   beq     $t1,   $t0,   for5Exit   #Branch to for5Exit if t1 (loop's index) is equal to t0...
□    0x004001b8  0x24020004 addiu $2,$0,4      254:   li      $v0,   4                 #Command to print a text
□    0x004001bc  0x3c011001 lui $1,4097        255:   la      $a0,   mensagem5         #Load address of mensagem5 to a0
□    0x004001c0  0x34240083 ori $4,$1,131
□    0x004001c4  0x0000000c syscall            256:   syscall                          #Do it
□    0x004001c8  0x24020001 addiu $2,$0,1      259:   li      $v0,   1                 #Command to print a integer
□    0x004001cc  0x00092021 addu $4,$0,$9      260:   move    $a0,   $t1               #Move $t1 (loop index) to a0
□    0x004001d0  0x0000000c syscall            261:   syscall                          #Do it
□    0x004001d4  0x24020004 addiu $2,$0,4      264:   li      $v0,   4                 #Command to print a text
□    0x004001d8  0x3c011001 lui $1,4097        265:   la      $a0,   mensagem4         #Load address of mensagem4 to a0
□    0x004001dc  0x3424007e ori $4,$1,126
□    0x004001e0  0x0000000c syscall            266:   syscall                          #Do it
□    0x004001e4  0x00095880 sll $11,$9,2       269:   sll     $t3,   $t1,   2          #Multiply t1 (loop's index) by 4 and put the result int...
□    0x004001e8  0x016d5820 add $11,$11,$13    270:   add     $t3,   $t3,   $t5        #Add t5 (array2's base) and t3 (bytes to be moved from ...
□    0x004001ec  0x8d6c0000 lw $12,0($11)      273:   lw      $t4,   ($t3)             #Load word from t3 (array1's fully calculated address) ...
□    0x004001f0  0x24020001 addiu $2,$0,1      276:   li      $v0,   1                 #Command to print a integer
□    0x004001f4  0x000c2021 addu $4,$0,$12     277:   move    $a0,   $t4               #Move t4 (the value) to a0
```

Data Segment

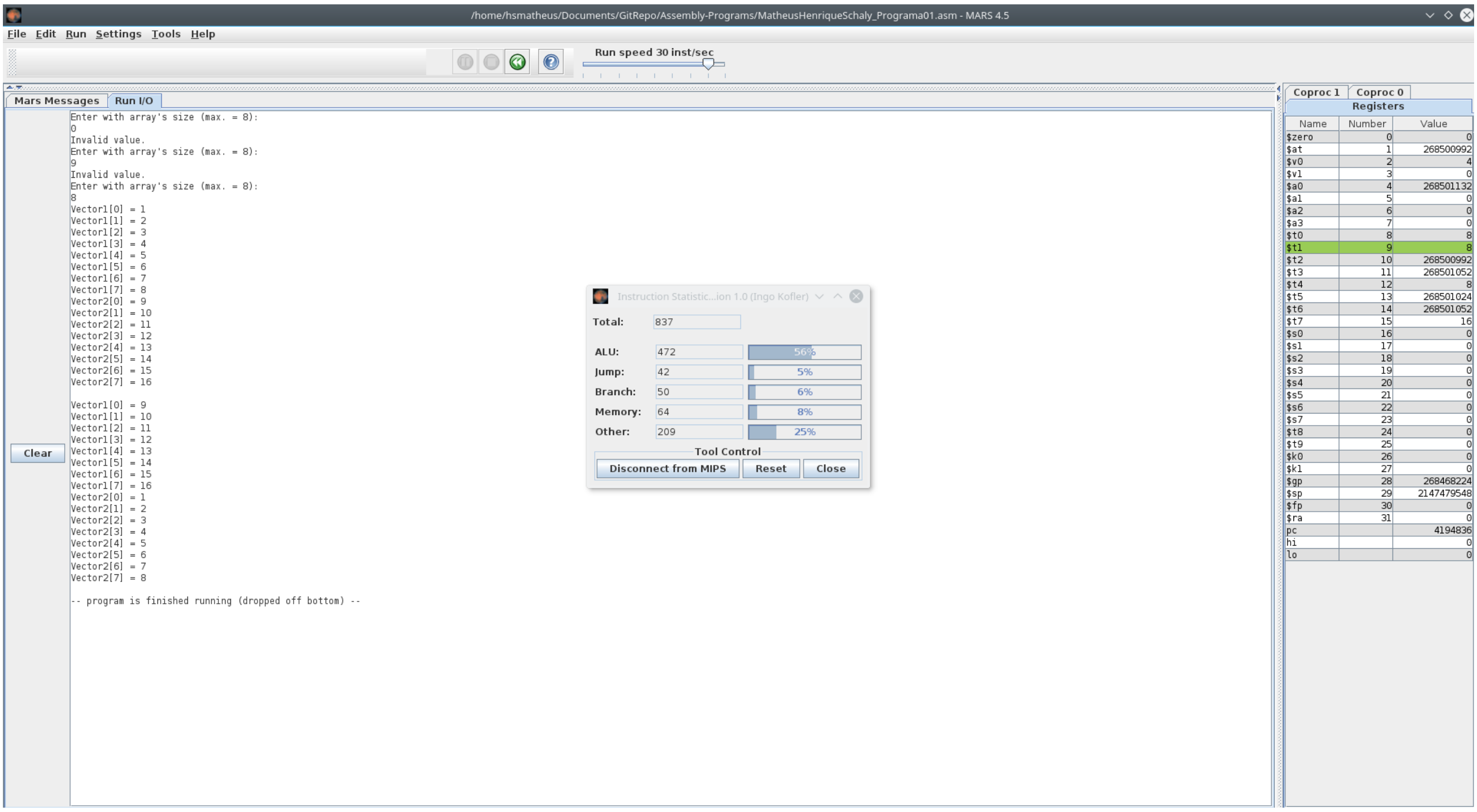
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	9	10	11	12	13	14	15	16
0x10010020	1	2	3	4	5	6	7	8
0x10010040	1702129221	1769414770	1629513844	2036429426	1931506471	543521385	2019650856	540876846
0x10010060	171583800	1986939136	1684630625	1818326560	170812789	1667585536	829583220	542965851
0x10010080	1442848829	1869898597	5976690	10	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	4
$v1	3	0
$a0	4	268501132
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	8
$t2	10	268500992
$t3	11	268501052
$t4	12	8
$t5	13	268501024
$t6	14	268501052
$t7	15	16
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194836
hi		0
lo		0

```
Vector1[6] = 15
Vector1[7] = 16
Vector2[0] = 1
Vector2[1] = 2
Vector2[2] = 3
Vector2[3] = 4
Vector2[4] = 5
Vector2[5] = 6
Vector2[6] = 7
Vector2[7] = 8

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

Quadro de análise das instruções:



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa01.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Mars Messages
Run I/O
Clear
Instruction Statistic...ion 1.0 (Ingo Kofler)
Tool Control
Disconnect from MIPS
Reset
Close
Coproc 1
Coproc 0
Registers

```
Enter with array's size (max. = 8):
0
Invalid value.
Enter with array's size (max. = 8):
9
Invalid value.
Enter with array's size (max. = 8):
8
Vector1[0] = 1
Vector1[1] = 2
Vector1[2] = 3
Vector1[3] = 4
Vector1[4] = 5
Vector1[5] = 6
Vector1[6] = 7
Vector1[7] = 8
Vector2[0] = 9
Vector2[1] = 10
Vector2[2] = 11
Vector2[3] = 12
Vector2[4] = 13
Vector2[5] = 14
Vector2[6] = 15
Vector2[7] = 16

Vector1[0] = 9
Vector1[1] = 10
Vector1[2] = 11
Vector1[3] = 12
Vector1[4] = 13
Vector1[5] = 14
Vector1[6] = 15
Vector1[7] = 16
Vector2[0] = 1
Vector2[1] = 2
Vector2[2] = 3
Vector2[3] = 4
Vector2[4] = 5
Vector2[5] = 6
Vector2[6] = 7
Vector2[7] = 8

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

Total:	837	
ALU:	472	56%
Jump:	42	5%
Branch:	50	6%
Memory:	64	8%
Other:	209	25%

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	4
$v1	3	0
$a0	4	268501132
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	8
$t2	10	268500992
$t3	11	268501052
$t4	12	8
$t5	13	268501024
$t6	14	268501052
$t7	15	16
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194836
hi		0
lo		0

C++ 2

```
// Disciplina: Arquitetura e Organização de Computadores
// Atividade: Avaliação 01 – Programação em C++
// Programa 02
// Grupo: - Matheus Henrique Schaly

#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
    int record[16][32], student, myClass, presence;

    for (int i = 0; i < 16; i++) {
        for (int j = 0; j < 32; j++) {
            record[i][j] = 1;
        }
    }

    while (true) {
        do {
            cout << "Enter class' number (0 to 15): ";
            cin >> myClass;
        } while (myClass < 0 || myClass > 15);
        do {
            cout << "Enter student's number (0 to 31): ";
            cin >> student;
        } while (student < 0 || student > 31);
        do {
            cout << "Enter register's type (presence = 1; absence = 0): ";
            cin >> presence;
        } while (presence < 0 || presence > 1);
        record[myClass][student] = presence;
    }
}
```

Assembly MIPS 2

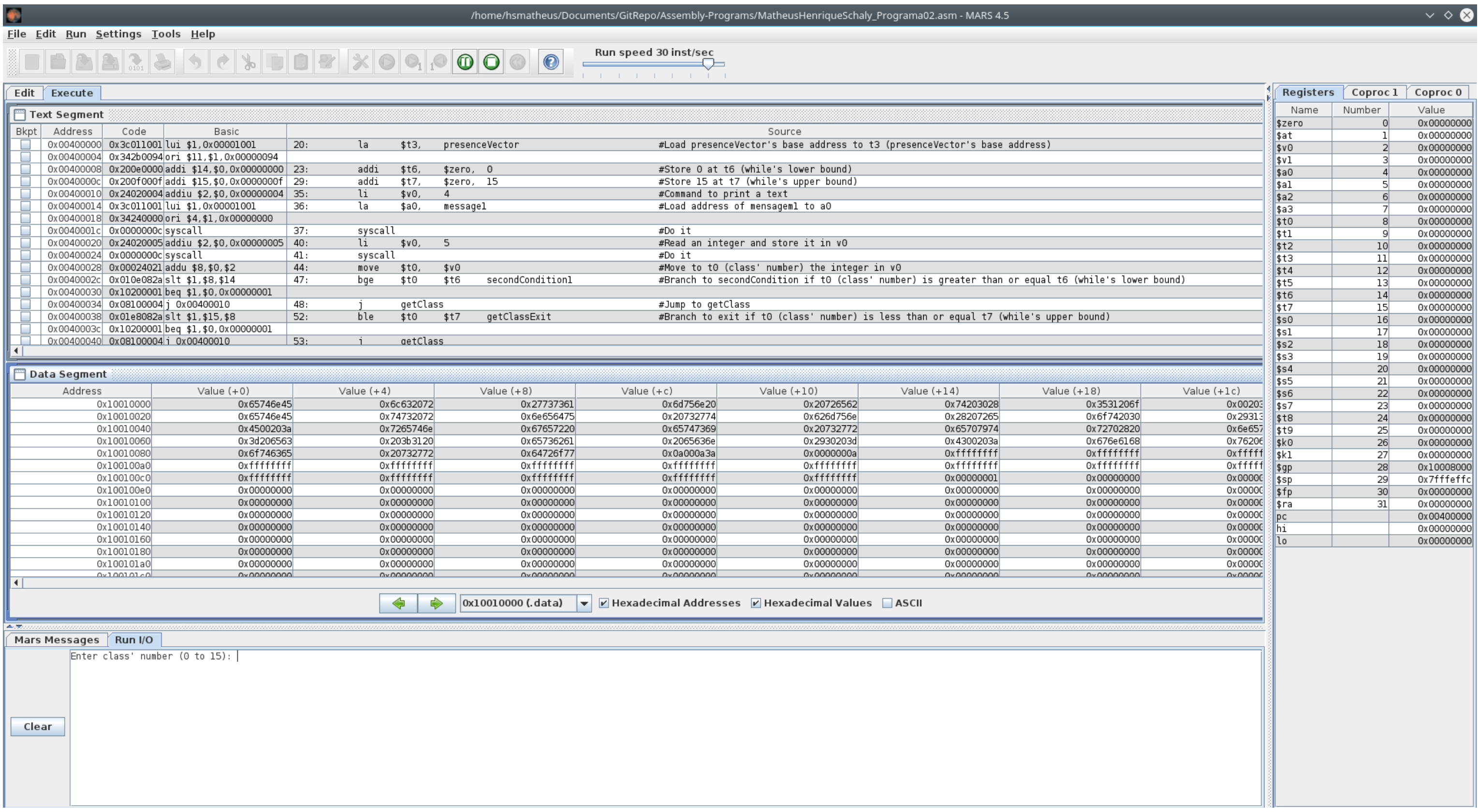
```
# Disciplina: Arquitetura e Organização de Computadores
# Atividade: Avaliação 01 – Programação em Linguagem de Montagem
# Programa 02
# Grupo: - Matheus Henrique Schaly

#Data stored in RAM
.data
        #Creates RAM variables
        message1:   .asciiz     "Enter class' number (0 to 15): "
        message2:   .asciiz     "Enter student's number (0 to 31): "
        message3:   .asciiz     "Enter register's type (presence = 1; absence = 0): "
        message4:   .asciiz     "Changed vector's word:\n"
        message5:   .asciiz     "\n\n"
        presenceVector:     .word       0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF            #Create a vector with 16 elements, each of them has 32 bits, that are all set to 1
        mask:       .word       1           #Mask with 32 bits, with only its last bit set to 1

.text

        #Loads array1's base address
        la      $t3,    presenceVector              #Load presenceVector's base address to t3 (presenceVector's base address)

        #Creates while's lower bound
        addi    $t6,    $zero, 0                    #Store 0 at t6 (while's lower bound)

        #Inifinite loop
start:

        #Creates/Resets while's upper bound
        addi    $t7,    $zero, 15                   #Store 15 at t7 (while's upper bound)

        #Do while loop to get class' number
getClass:

        #Prints a text
        li      $v0,    4                           #Command to print a text
        la      $a0,    message1                    #Load address of mensagem1 to a0
        syscall                                     #Do it

        #Reads integer
        li      $v0,    5                           #Read an integer and store it in v0
        syscall

        #Stores integer
        move    $t0,    $v0                         #Move to t0 (class' number) the integer in v0

        #While's first if condition
        bge     $t0     $t6     secondCondition1    #Branch to secondCondition if t0 (class' number) is greater than or equal t6 (while's lower bound)
        j       getClass                            #Jump to getClass
```

```
	#While's second if condition
secondCondition1:
	ble	$t0	$t7	getClassExit		#Branch to exit if t0 (class' number) is less than or equal t7 (while's upper bound)
	j	getClass

getClassExit:

	#Readjusts while's upper bound
	addi	$t7,	$zero, 31			#Store 31 at t7 (while's upper bound)

	#Do while loop to get student's number
getStudent:

	#Prints a text
	li	$v0,	4				#Command to print a text
	la	$a0,	message2			#Load address of mensagem2 to a0
	syscall						#Do it

	#Reads integer
	li	$v0,	5				#Read an integer and store it in v0
	syscall						#Do it

	#Stores integer
	move	$t1,	$v0				#Move to t1 (student's number) the integer in v0

	#While's first if condition
	bge	$t1,	$t6	secondCondition2	#Branch to secondCondition if t1 (student's number) is greater than or equal t6 (while's lower bound)
	j	getStudent				#Jump to getStudent

	#While's second if condition
secondCondition2:
	ble	$t1,	$t7	getStudentExit		#Branch to exit if t1 (student's number) is less than or equal t7 (while's upper bound)
	j	getStudent

getStudentExit:

	#Readjusts while's upper bound
	addi	$t7,	$zero, 1			#Store 31 at t7 (while's upper bound)

	#Do while loop to get student's presence
getPresence:

	#Prints a text
	li	$v0,	4				#Command to print a text
	la	$a0,	message3			#Load address of mensagem3 to a0
	syscall						#Do it

	#Reads integer
	li	$v0,	5				#Read an integer and store it in v0
	syscall

	#Stores integer
	move	$t2,	$v0				#Move to t2 (student's presence) the integer in v0

	#While's first if condition
	bge	$t2,	$t6	secondCondition3	#Branch to secondCondition if t2 (student's presence) is greater
than or equal t6 (while's lower bound)
	j	getPresence				#Jump to getPresence

	#While's second if condition
secondCondition3:
	ble	$t2,	$t7	getPresenceExit		#Branch to exit if t2 (student's presence) is less than or equal t7
(while's upper bound)
	j	getPresence

getPresenceExit:

	#A - Calculates the presenceVector's address to be changed
	sll	$t4,	$t0,	2			#Multiply t0 (class' number) by 4 and put the result into t4 (bytes
to be moved from presenceVector's base address)
	add	$t4,	$t4,	$t3			#Add t3 (presenceVector's base) and t4 (bytes to be moved from
presenceVector's base address) and put it back into t4 (presenceVector's fully calculated address)

	#Loads the input from presenceVector
	lw	$t5,	($t4)				#Load word from t4 (presenceVector's fully calculated address) to
t5 (presenceVector's value)

	#B - Calculates how many mask's bits will be moved
	lw	$t7,	mask				#Load address of mask to t7 (original mask)
	sllv	$t7,	$t7,	$t1			#Multiply t7 (original mask) by t1 (student's number) and put the
result into t7 (mask bits moved for t1 times)

	#C - If condition to check if student is or not present
	beq	$t2,	$t6,	registerAusence		#Branch to registerAusence if t2 (student's presence) is equal to
t6 (while's lower bound (which is 0))

	#Register a presence

	#OR mask and presenceVector's single position
	or	$t5,	$t5,	$t7			#OR t7 (changed mask) and t5 (presenceVector's value) then put
the result into t5 (presenceVector's changed value)

	#Stores presenceVector's changed value back
	sw	$t5,	($t4)				#Store word from t5 (presenceVector's changed value) in t4
(presenceVector's fully calculated address)

	#Restarts the whole loop
	j	print					#Jump to start

	#Register a ausence
registerAusence:

	#XOR mask and 32 bits (all set to 1)
	xori	$t7,	$t7,	0xFFFFFFFF		#XOR (exclusive OR) t7 (changed mask) and 32 bits (all set to 1)
then put the result into t7 (XORed mask)
```

```
        #AND XORed mask and presenceVector's single position
        and     $t5,    $t7,    $t5                 #OR t7 (XORed mask) and t5 (presenceVector's value) then put the result into t5 (presenceVector's changed value)

        #Stores presenceVector's changed value back
        sw      $t5,    ($t4)                       #Store word from t5 (presenceVector's changed value) in t4 (presenceVector's fully calculated address)

        #Prints the value that has returned to presenceVector and new lines
print:

        #Prints mensagem4
        li      $v0,    4                           #Command to print a text
        la      $a0,    message4                    #Load address of message4 to a0
        syscall                                     #Do it

        #Prints the value that has returned to presenceVector
        la      $v0,    35                          #Command to print a integer as binary
        la      $a0,    ($t5)                       #Load word of t5 (presenceVector's value) to a0
        syscall                                     #Do it

        #Prints mensagem5
        li      $v0,    4                           #Command to print a text
        la      $a0,    message5                    #Load address of message5 to a0
        syscall                                     #Do it

        #Restarts the whole loop
        j       start                               #Jump to start
```

Capturas de Tela 2:

Inicializando programa:



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic		Source
☐	0x00400000	0x3c011001	lui $1,0x00001001	20:	la $t3, presenceVector #Load presenceVector's base address to t3 (presenceVector's base address)
☐	0x00400004	0x342b0094	ori $11,$1,0x00000094		
☐	0x00400008	0x200e0000	addi $14,$0,0x00000000	23:	addi $t6, $zero, 0 #Store 0 at t6 (while's lower bound)
☐	0x0040000c	0x200f000f	addi $15,$0,0x0000000f	29:	addi $t7, $zero, 15 #Store 15 at t7 (while's upper bound)
☐	0x00400010	0x24020004	addiu $2,$0,0x00000004	35:	li $v0, 4 #Command to print a text
☐	0x00400014	0x3c011001	lui $1,0x00001001	36:	la $a0, message1 #Load address of mensagem1 to a0
☐	0x00400018	0x34240000	ori $4,$1,0x00000000		
☐	0x0040001c	0x0000000c	syscall	37:	syscall #Do it
☐	0x00400020	0x24020005	addiu $2,$0,0x00000005	40:	li $v0, 5 #Read an integer and store it in v0
☐	0x00400024	0x0000000c	syscall	41:	syscall #Do it
☐	0x00400028	0x00024021	addu $8,$0,$2	44:	move $t0, $v0 #Move to t0 (class' number) the integer in v0
☐	0x0040002c	0x010e082a	slt $1,$8,$14	47:	bge $t0 $t6 secondCondition1 #Branch to secondCondition if t0 (class' number) is greater than or equal t6 (while's lower bound)
☐	0x00400030	0x10200001	beq $1,$0,0x00000001		
☐	0x00400034	0x08100004	j 0x00400010	48:	j getClass #Jump to getClass
☐	0x00400038	0x01e8082a	slt $1,$15,$8	52:	ble $t0 $t7 getClassExit #Branch to exit if t0 (class' number) is less than or equal t7 (while's upper bound)
☐	0x0040003c	0x10200001	beq $1,$0,0x00000001		
☐	0x00400040	0x08100004	j 0x00400010	53:	j getClass

Data Segment

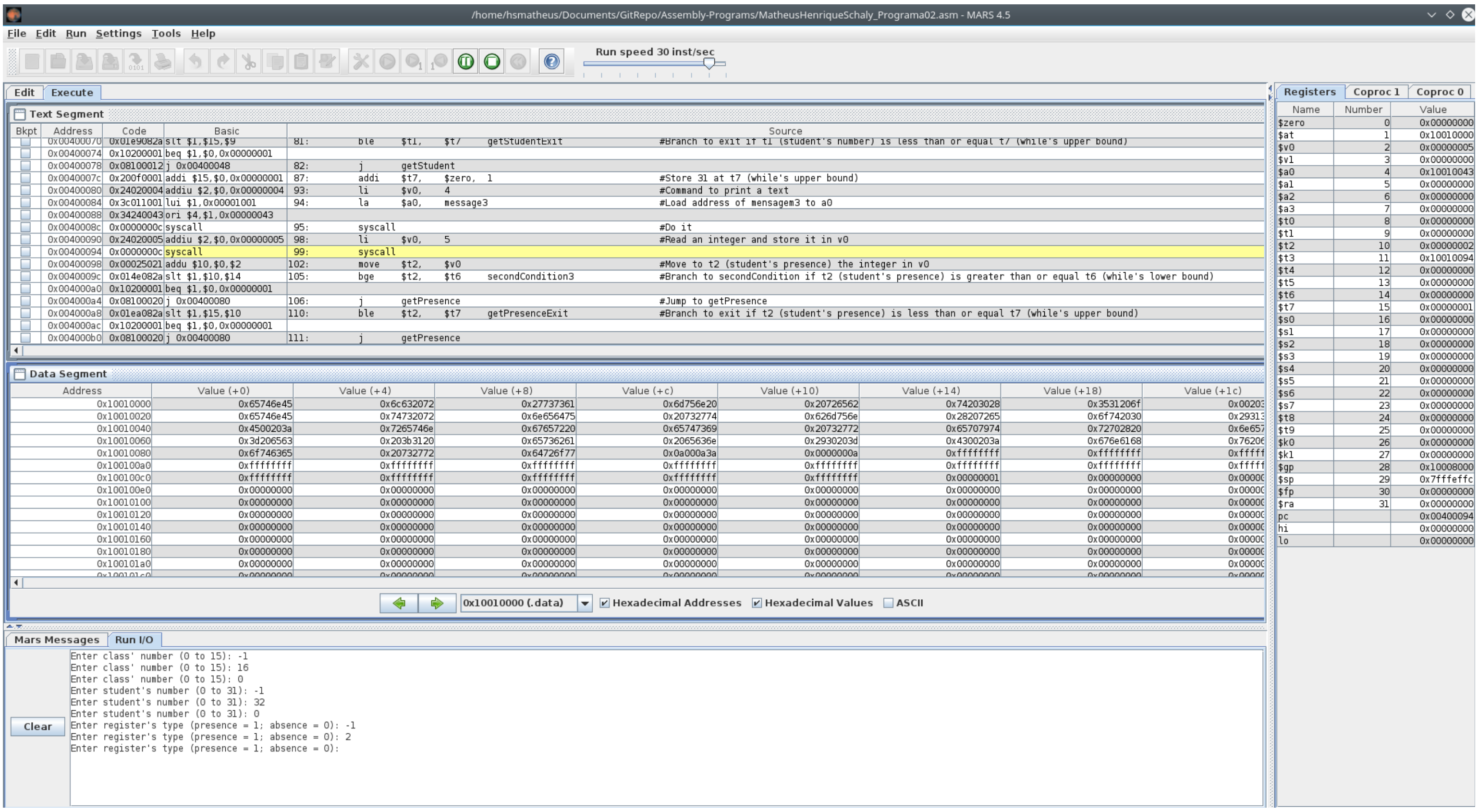
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x6c632072	0x27737361	0x6d756e20	0x20726562	0x74203028	0x3531206f	0x00203
0x10010020	0x65746e45	0x74732072	0x6e656475	0x20732774	0x626d756e	0x28207265	0x6f742030	0x29313
0x10010040	0x4500203a	0x7265746e	0x67657220	0x65747369	0x20732772	0x65707974	0x72702820	0x6e657
0x10010060	0x3d206563	0x203b3120	0x65736261	0x2065636e	0x2930203d	0x4300203a	0x676e6168	0x76206
0x10010080	0x6f746365	0x20732772	0x64726f77	0x0a000a3a	0x0000000a	0xffffffff	0xffffffff	0xffff
0x100100a0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffff
0x100100c0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0x00000001	0x00000000	0x0000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

```
Enter class' number (0 to 15): |
```

Possíveis valores inválidos e em seguida entrada correta:



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic		Source
☐	0x00400070	0x01e9082a	slt $1,$15,$9	81:	ble $t1, $t7 getStudentExit #Branch to exit if t1 (student's number) is less than or equal t7 (while's upper bound)
☐	0x00400074	0x10200001	beq $1,$0,0x00000001		
☐	0x00400078	0x08100012	j 0x00400048	82:	j getStudent
☐	0x0040007c	0x200f0001	addi $15,$0,0x00000001	87:	addi $t7, $zero, 1 #Store 31 at t7 (while's upper bound)
☐	0x00400080	0x24020004	addiu $2,$0,0x00000004	93:	li $v0, 4 #Command to print a text
☐	0x00400084	0x3c011001	lui $1,0x00001001	94:	la $a0, message3 #Load address of mensagem3 to a0
☐	0x00400088	0x34240043	ori $4,$1,0x00000043		
☐	0x0040008c	0x0000000c	syscall	95:	syscall #Do it
☐	0x00400090	0x24020005	addiu $2,$0,0x00000005	98:	li $v0, 5 #Read an integer and store it in v0
☐	0x00400094	0x0000000c	syscall	99:	syscall
☐	0x00400098	0x00025021	addu $10,$0,$2	102:	move $t2, $v0 #Move to t2 (student's presence) the integer in v0
☐	0x0040009c	0x014e082a	slt $1,$10,$14	105:	bge $t2, $t6 secondCondition3 #Branch to secondCondition if t2 (student's presence) is greater than or equal t6 (while's lower bound)
☐	0x004000a0	0x10200001	beq $1,$0,0x00000001		
☐	0x004000a4	0x08100020	j 0x00400080	106:	j getPresence #Jump to getPresence
☐	0x004000a8	0x01ea082a	slt $1,$15,$10	110:	ble $t2, $t7 getPresenceExit #Branch to exit if t2 (student's presence) is less than or equal t7 (while's upper bound)
☐	0x004000ac	0x10200001	beq $1,$0,0x00000001		
☐	0x004000b0	0x08100020	j 0x00400080	111:	j getPresence

Data Segment

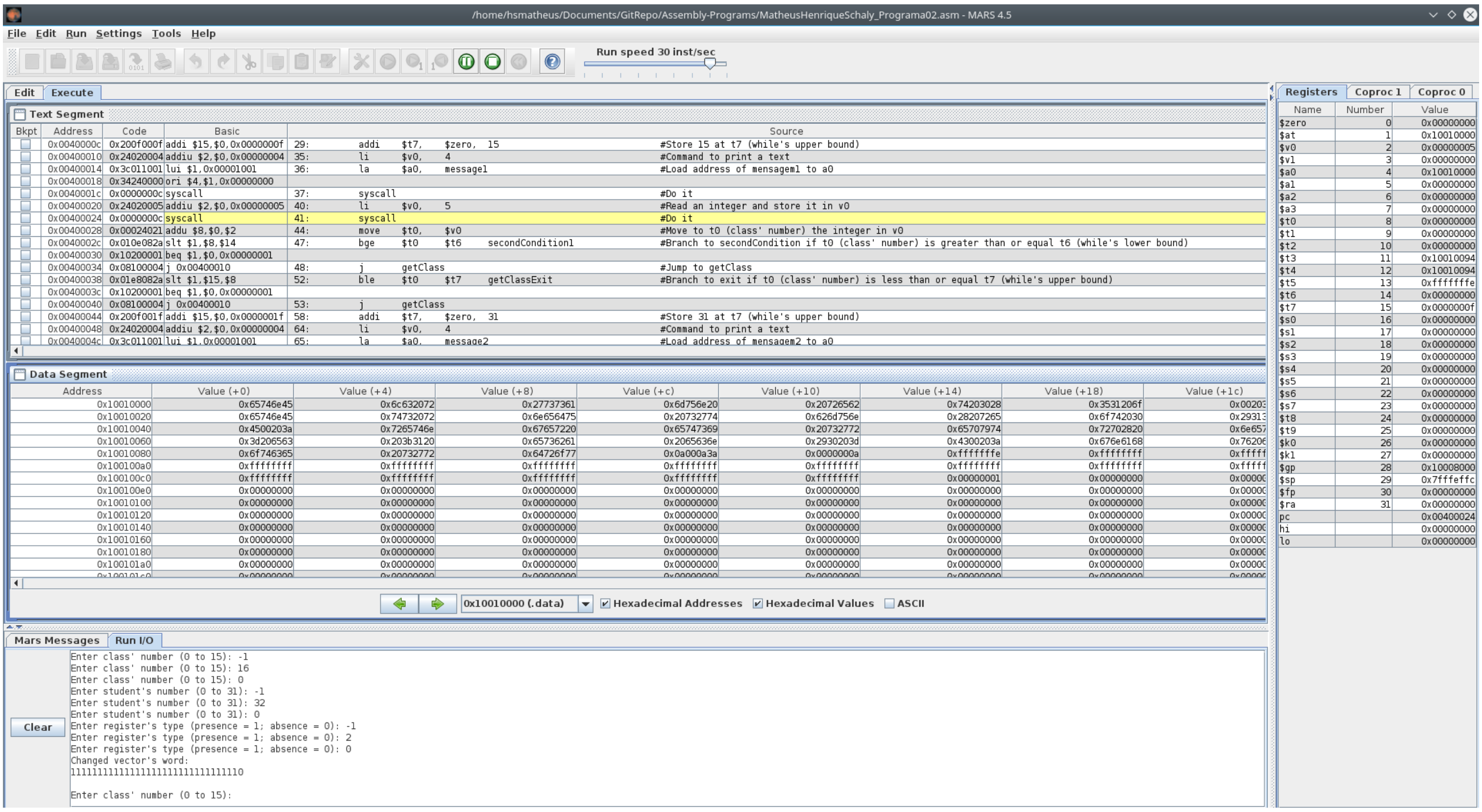
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x6c632072	0x27737361	0x6d756e20	0x20726562	0x74203028	0x3531206f	0x00203
0x10010020	0x65746e45	0x74732072	0x6e656475	0x20732774	0x626d756e	0x28207265	0x6f742030	0x29313
0x10010040	0x4500203a	0x7265746e	0x67657220	0x65747369	0x20732772	0x65707974	0x72702820	0x6e657
0x10010060	0x3d206563	0x203b3120	0x65736261	0x2065636e	0x2930203d	0x4300203a	0x676e6168	0x76206
0x10010080	0x6f746365	0x20732772	0x64726f77	0x0a000a3a	0x0000000a	0xffffffff	0xffffffff	0xffff
0x100100a0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffff
0x100100c0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0x00000001	0x00000000	0x0000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010043
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000002
$t3	11	0x10010094
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000001
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400094
hi		0x00000000
lo		0x00000000

```
Enter class' number (0 to 15): -1
Enter class' number (0 to 15): 16
Enter class' number (0 to 15): 0
Enter student's number (0 to 31): -1
Enter student's number (0 to 31): 32
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): -1
Enter register's type (presence = 1; absence = 0): 2
Enter register's type (presence = 1; absence = 0):
```

Após entrada de valores, o valor da coluna Value(+14) e linha 0x10010080 (que simboliza o primeiro elemento do vetor) passa de 0xffffffff para 0xfffffffe, ou seja, um de seus bits foi alterado. Neste caso o primeiro bit (índice 0) foi alterado, como é demonstrado mais explicitamente na saída do programa, onde o primeiro bit (índice 0) passa de 1 para 0:


/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

Bkpt	Address	Code	Basic		Source
□	0x0040000c	0x200f000f	addi $15,$0,0x0000000f	29: addi $t7, $zero, 15	#Store 15 at t7 (while's upper bound)
□	0x00400010	0x24020004	addiu $2,$0,0x00000004	35: li $v0, 4	#Command to print a text
□	0x00400014	0x3c011001	lui $1,0x00001001	36: la $a0, message1	#Load address of mensagem1 to a0
□	0x00400018	0x34240000	ori $4,$1,0x00000000		
□	0x0040001c	0x0000000c	syscall	37: syscall	#Do it
□	0x00400020	0x24020005	addiu $2,$0,0x00000005	40: li $v0, 5	#Read an integer and store it in v0
□	0x00400024	0x0000000c	syscall	41: syscall	#Do it
□	0x00400028	0x00024021	addu $8,$0,$2	44: move $t0, $v0	#Move to t0 (class' number) the integer in v0
□	0x0040002c	0x010e082a	slt $1,$8,$14	47: bge $t0 $t6 secondCondition1	#Branch to secondCondition if t0 (class' number) is greater than or equal t6 (while's lower bound)
□	0x00400030	0x10200001	beq $1,$0,0x00000001		
□	0x00400034	0x08100004	j 0x00400010	48: j getClass	#Jump to getClass
□	0x00400038	0x01e8082a	slt $1,$15,$8	52: ble $t0 $t7 getClassExit	#Branch to exit if t0 (class' number) is less than or equal t7 (while's upper bound)
□	0x0040003c	0x10200001	beq $1,$0,0x00000001		
□	0x00400040	0x08100004	j 0x00400010	53: j getClass	
□	0x00400044	0x200f001f	addi $15,$0,0x0000001f	58: addi $t7, $zero, 31	#Store 31 at t7 (while's upper bound)
□	0x00400048	0x24020004	addiu $2,$0,0x00000004	64: li $v0, 4	#Command to print a text
□	0x0040004c	0x3c011001	lui $1,0x00001001	65: la $a0, message2	#Load address of mensagem2 to a0

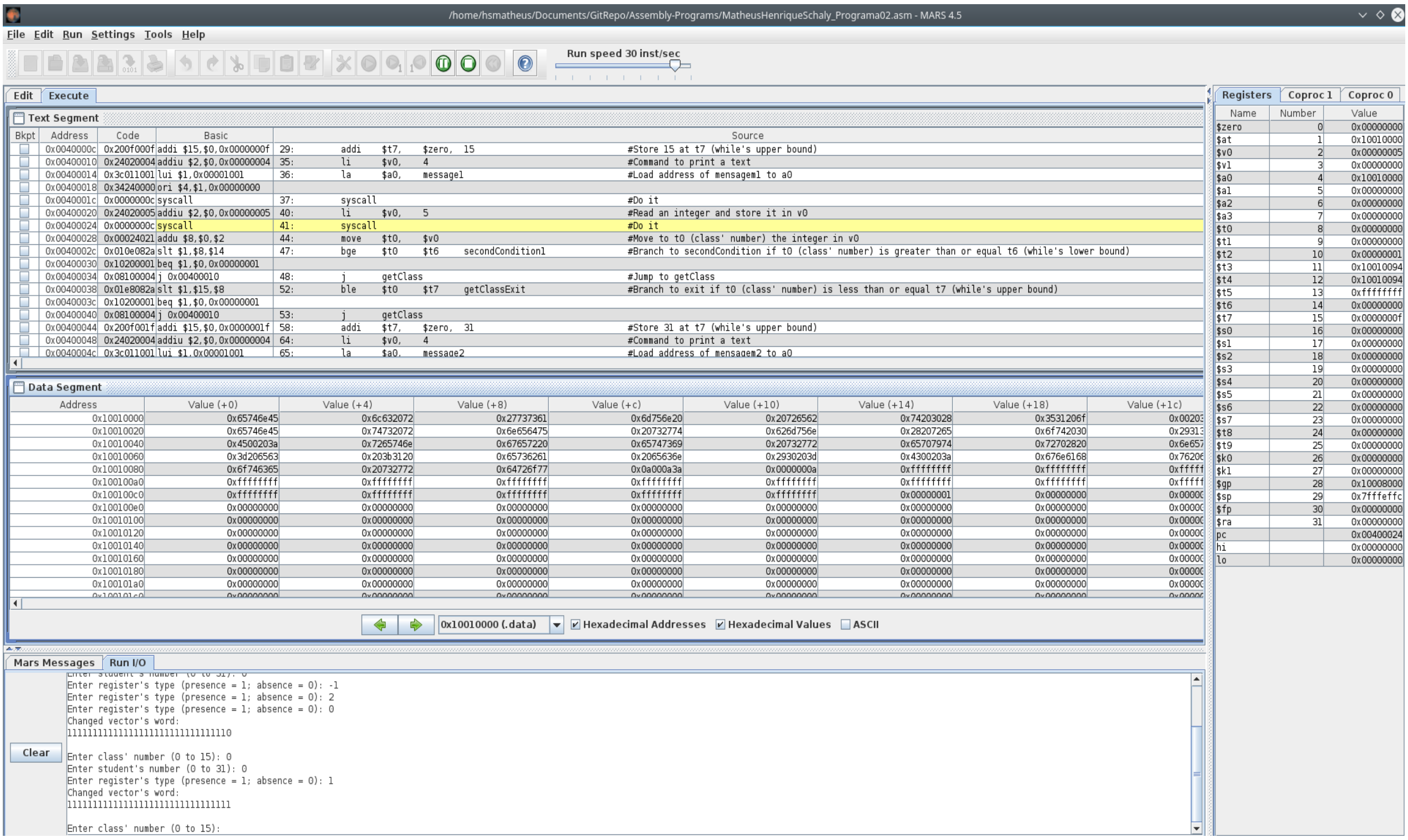
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x6c632072	0x27737361	0x6d756e20	0x20726562	0x74203028	0x3531206f	0x00203
0x10010020	0x65746e45	0x74732072	0x6e656475	0x20732774	0x626d756e	0x28207265	0x6f742030	0x29313
0x10010040	0x4500203a	0x7265746e	0x67657220	0x65747369	0x20732772	0x65707974	0x72702820	0x6e657
0x10010060	0x3d206563	0x203b3120	0x65736261	0x2065636e	0x2930203d	0x4300203a	0x676e6168	0x76206
0x10010080	0x6f746365	0x20732772	0x64726f77	0x0a000a3a	0x0000000a	0xfffffffe	0xffffffff	0xfffff
0x100100a0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xfffff
0x100100c0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0x00000001	0x00000000	0x0000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x10010094
$t4	12	0x10010094
$t5	13	0xfffffffe
$t6	14	0x00000000
$t7	15	0x0000000f
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

```
Enter class' number (0 to 15): -1
Enter class' number (0 to 15): 16
Enter class' number (0 to 15): 0
Enter student's number (0 to 31): -1
Enter student's number (0 to 31): 32
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): -1
Enter register's type (presence = 1; absence = 0): 2
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15):
```

Retornando o valor do primeiro elemento do vetor para 0xffffffff, acompanhado pela demonstração binária na saída do programa:


/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

Bkpt	Address	Code	Basic		Source
□	0x0040000c	0x200f000f	addi $15,$0,0x0000000f	29: addi $t7, $zero, 15	#Store 15 at t7 (while's upper bound)
□	0x00400010	0x24020004	addiu $2,$0,0x00000004	35: li $v0, 4	#Command to print a text
□	0x00400014	0x3c011001	lui $1,0x00001001	36: la $a0, message1	#Load address of mensagem1 to a0
□	0x00400018	0x34240000	ori $4,$1,0x00000000		
□	0x0040001c	0x0000000c	syscall	37: syscall	#Do it
□	0x00400020	0x24020005	addiu $2,$0,0x00000005	40: li $v0, 5	#Read an integer and store it in v0
□	0x00400024	0x0000000c	syscall	41: syscall	#Do it
□	0x00400028	0x00024021	addu $8,$0,$2	44: move $t0, $v0	#Move to t0 (class' number) the integer in v0
□	0x0040002c	0x010e082a	slt $1,$8,$14	47: bge $t0 $t6 secondCondition1	#Branch to secondCondition if t0 (class' number) is greater than or equal t6 (while's lower bound)
□	0x00400030	0x10200001	beq $1,$0,0x00000001		
□	0x00400034	0x08100004	j 0x00400010	48: j getClass	#Jump to getClass
□	0x00400038	0x01e8082a	slt $1,$15,$8	52: ble $t0 $t7 getClassExit	#Branch to exit if t0 (class' number) is less than or equal t7 (while's upper bound)
□	0x0040003c	0x10200001	beq $1,$0,0x00000001		
□	0x00400040	0x08100004	j 0x00400010	53: j getClass	
□	0x00400044	0x200f001f	addi $15,$0,0x0000001f	58: addi $t7, $zero, 31	#Store 31 at t7 (while's upper bound)
□	0x00400048	0x24020004	addiu $2,$0,0x00000004	64: li $v0, 4	#Command to print a text
□	0x0040004c	0x3c011001	lui $1,0x00001001	65: la $a0, message2	#Load address of mensagem2 to a0

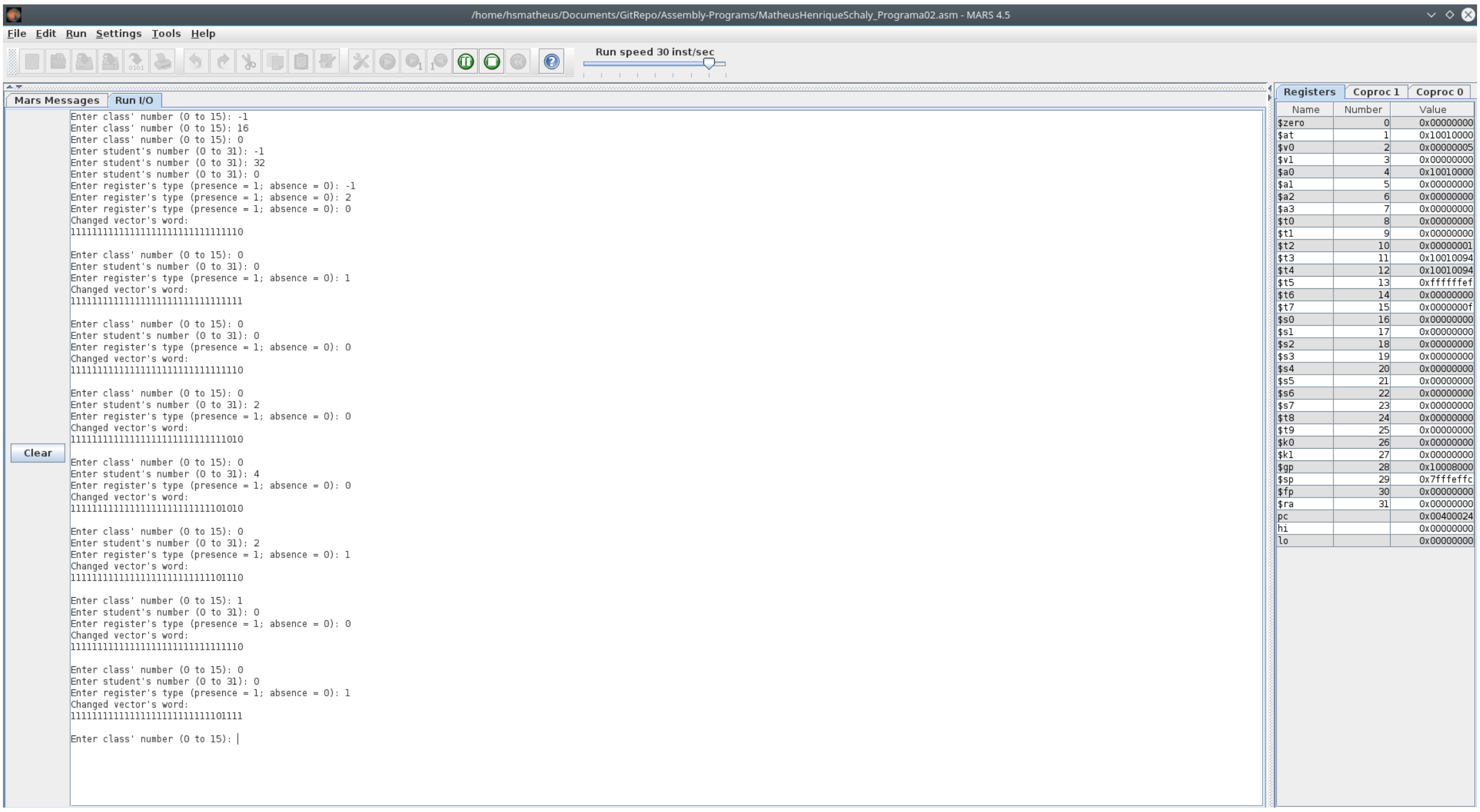
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x6c632072	0x27737361	0x6d756e20	0x20726562	0x74203028	0x3531206f	0x00203
0x10010020	0x65746e45	0x74732072	0x6e656475	0x20732774	0x626d756e	0x28207265	0x6f742030	0x29313
0x10010040	0x4500203a	0x7265746e	0x67657220	0x65747369	0x20732772	0x65707974	0x72702820	0x6e657
0x10010060	0x3d206563	0x203b3120	0x65736261	0x2065636e	0x2930203d	0x4300203a	0x676e6168	0x76206
0x10010080	0x6f746365	0x20732772	0x64726f77	0x0a000a3a	0x0000000a	0xffffffff	0xffffffff	0xfffff
0x100100a0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xfffff
0x100100c0	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0xffffffff	0x00000001	0x00000000	0x0000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000001
$t3	11	0x10010094
$t4	12	0x10010094
$t5	13	0xffffffff
$t6	14	0x00000000
$t7	15	0x0000000f
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

```
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): -1
Enter register's type (presence = 1; absence = 0): 2
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 1
Changed vector's word:
11111111111111111111111111111111

Enter class' number (0 to 15):
```

Exemplos do programa em execução:



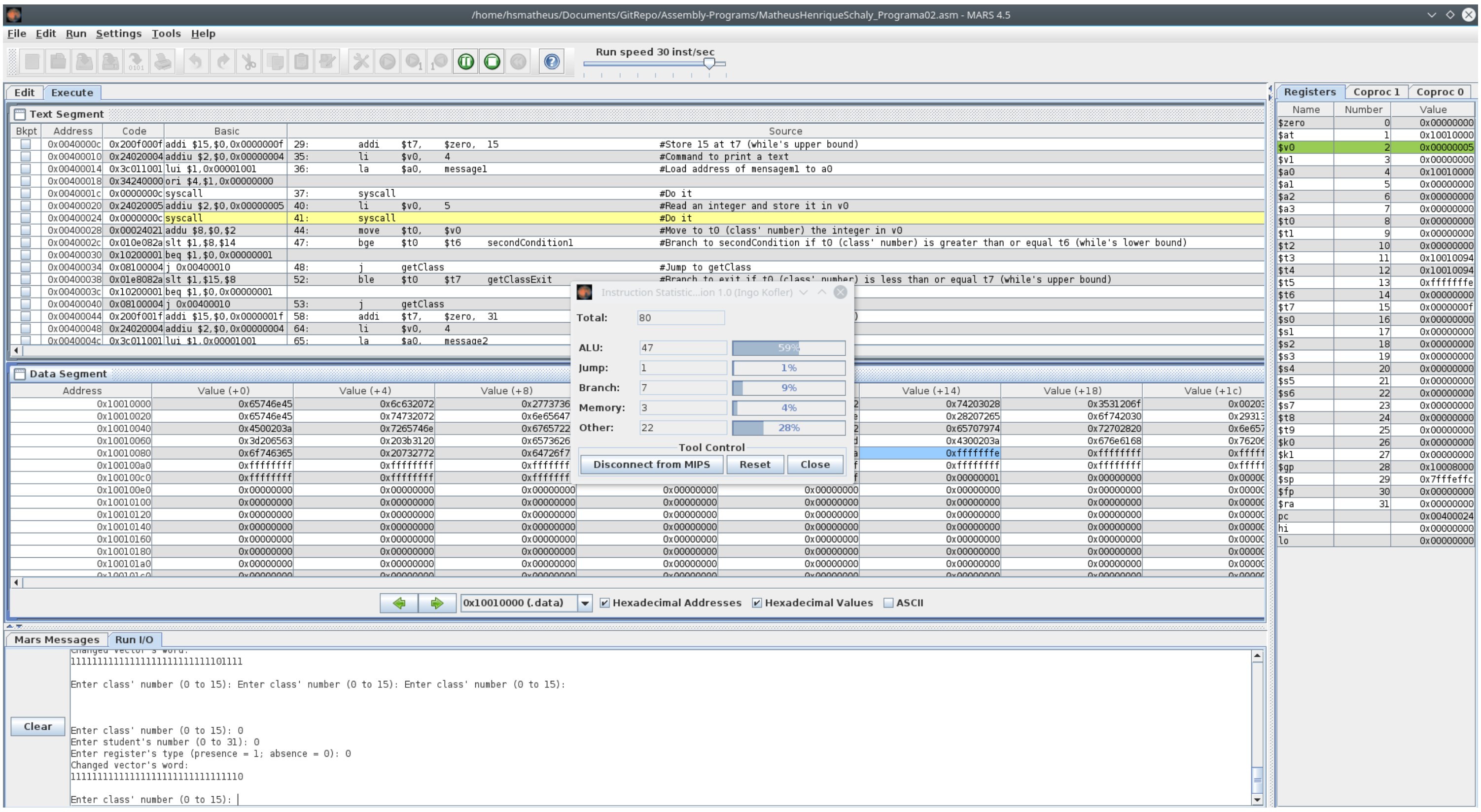
/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Mars Messages
Run I/O
Clear

```
Enter class' number (0 to 15): -1
Enter class' number (0 to 15): 16
Enter class' number (0 to 15): 0
Enter student's number (0 to 31): -1
Enter student's number (0 to 31): 32
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): -1
Enter register's type (presence = 1; absence = 0): 2
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 1
Changed vector's word:
11111111111111111111111111111111

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 2
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111010

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 4
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111101010

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 2
Enter register's type (presence = 1; absence = 0): 1
Changed vector's word:
11111111111111111111111111101110

Enter class' number (0 to 15): 1
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 1
Changed vector's word:
11111111111111111111111111101111

Enter class' number (0 to 15): |
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000001
$t3	11	0x10010094
$t4	12	0x10010094
$t5	13	0xffffffef
$t6	14	0x00000000
$t7	15	0x0000000f
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

Quadro de análise das instruções (apenas um loop):



/home/hsmatheus/Documents/GitRepo/Assembly-Programs/MatheusHenriqueSchaly_Programa02.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed 30 inst/sec
Edit
Execute
Text Segment
Data Segment
Instruction Statistic...ion 1.0 (Ingo Kofler)
Tool Control
Disconnect from MIPS
Reset
Close
Mars Messages
Run I/O
Clear

Text Segment

Bkpt	Address	Code	Basic		Source
□	0x0040000c	0x200f000f	addi $15,$0,0x0000000f	29: addi $t7, $zero, 15	#Store 15 at t7 (while's upper bound)
□	0x00400010	0x24020004	addiu $2,$0,0x00000004	35: li $v0, 4	#Command to print a text
□	0x00400014	0x3c011001	lui $1,0x00001001	36: la $a0, message1	#Load address of mensagem1 to a0
□	0x00400018	0x34240000	ori $4,$1,0x00000000		
□	0x0040001c	0x0000000c	syscall	37: syscall	#Do it
□	0x00400020	0x24020005	addiu $2,$0,0x00000005	40: li $v0, 5	#Read an integer and store it in v0
□	0x00400024	0x0000000c	syscall	41: syscall	#Do it
□	0x00400028	0x00024021	addu $8,$0,$2	44: move $t0, $v0	#Move to t0 (class' number) the integer in v0
□	0x0040002c	0x010e082a	slt $1,$8,$14	47: bge $t0 $t6 secondCondition1	#Branch to secondCondition if t0 (class' number) is greater than or equal t6 (while's lower bound)
□	0x00400030	0x10200001	beq $1,$0,0x00000001		
□	0x00400034	0x08100004	j 0x00400010	48: j getClass	#Jump to getClass
□	0x00400038	0x01e8082a	slt $1,$15,$8	52: ble $t0 $t7 getClassExit	#Branch to exit if t0 (class' number) is less than or equal t7 (while's upper bound)
□	0x0040003c	0x10200001	beq $1,$0,0x00000001		
□	0x00400040	0x08100004	j 0x00400010	53: j getClass	
□	0x00400044	0x200f001f	addi $15,$0,0x0000001f	58: addi $t7, $zero, 31	)
□	0x00400048	0x24020004	addiu $2,$0,0x00000004	64: li $v0, 4	
□	0x0040004c	0x3c011001	lui $1,0x00001001	65: la $a0, message2	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x6c632072	0x2773736	[illegible]	[illegible]	0x74203028	0x3531206f	0x00203
0x10010020	0x65746e45	0x74732072	0x6e65647	[illegible]	[illegible]	0x28207265	0x6f742030	0x29313
0x10010040	0x4500203a	0x7265746e	0x6765722	[illegible]	[illegible]	0x65707974	0x72702820	0x6e657
0x10010060	0x3d206563	0x203b3120	0x6573626	[illegible]	[illegible]	0x4300203a	0x676e6168	0x76206
0x10010080	0x6f746365	0x20732772	0x64726f7	[illegible]	[illegible]	0xfffffffe	0xffffffff	0xffff
0x100100a0	0xffffffff	0xffffffff	0xffffff	[illegible]	[illegible]	0xffffffff	0xffffffff	0xffff
0x100100c0	0xffffffff	0xffffffff	0xffffff	[illegible]	[illegible]	0x00000001	0x00000000	0x0000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Total:	80	
ALU:	47	59%
Jump:	1	1%
Branch:	7	9%
Memory:	3	4%
Other:	22	28%

```
Changed vector's word:
11111111111111111111111111101111

Enter class' number (0 to 15): Enter class' number (0 to 15): Enter class' number (0 to 15):



Enter class' number (0 to 15): 0
Enter student's number (0 to 31): 0
Enter register's type (presence = 1; absence = 0): 0
Changed vector's word:
11111111111111111111111111111110

Enter class' number (0 to 15): |
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000005
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x10010094
$t4	12	0x10010094
$t5	13	0xfffffffe
$t6	14	0x00000000
$t7	15	0x0000000f
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000