UNIVALI - Universidade do Vale do Itajaí

Ciência da Computação

Arquitetura de Computadores 1

Professor: Douglas Rossi de Melo

Aluno: Mauricio Macário de Farias Junior

Avaliação M1
Programação em linguagem de montagem

Data de entrega: 15/09/2017

Programa 01

Versão do código em C++:

```
#include <iostream>

using namespace std;

int main()
{
   int Vetor_A[8], Vetor_B[8], tam, aux, i;

   cout << "Entre com o tamanho dos vetores (max. = 8): ";
   do
   {
      cin >> tam;
      if (tam<1 || tam>8)
      {
         cout << "Entrada invalida!!" << endl;
      }
   }
   while (tam<1 || tam>8);

   for (i=0;i<tam;i++)
   {
      cout << "Vetor_A[" << i << "]: ";
      cin >> Vetor_A[i];
   }

   for (int i=0;i<tam;i++)
   {
      cout << "Vetor_B[" << i << "]: ";
      cin >> Vetor_B[i];
   }

   for (int i=0;i<tam;i++)
   {
      aux = Vetor_A[i];
      Vetor_A[i] = Vetor_B[i];
      Vetor_B[i] = aux;
   }

   for (int i=0;i<tam;i++)
   {
      cout << "Vetor_A[" << i << "]: " << Vetor_A[i] << endl;
   }

   for (int i=0;i<tam;i++)
```

```
    {
        cout << "Vetor_B[" << i << "]: " << Vetor_B[i] << endl;
    }
    return 0;
}
```

Versão do Código em Assembly:

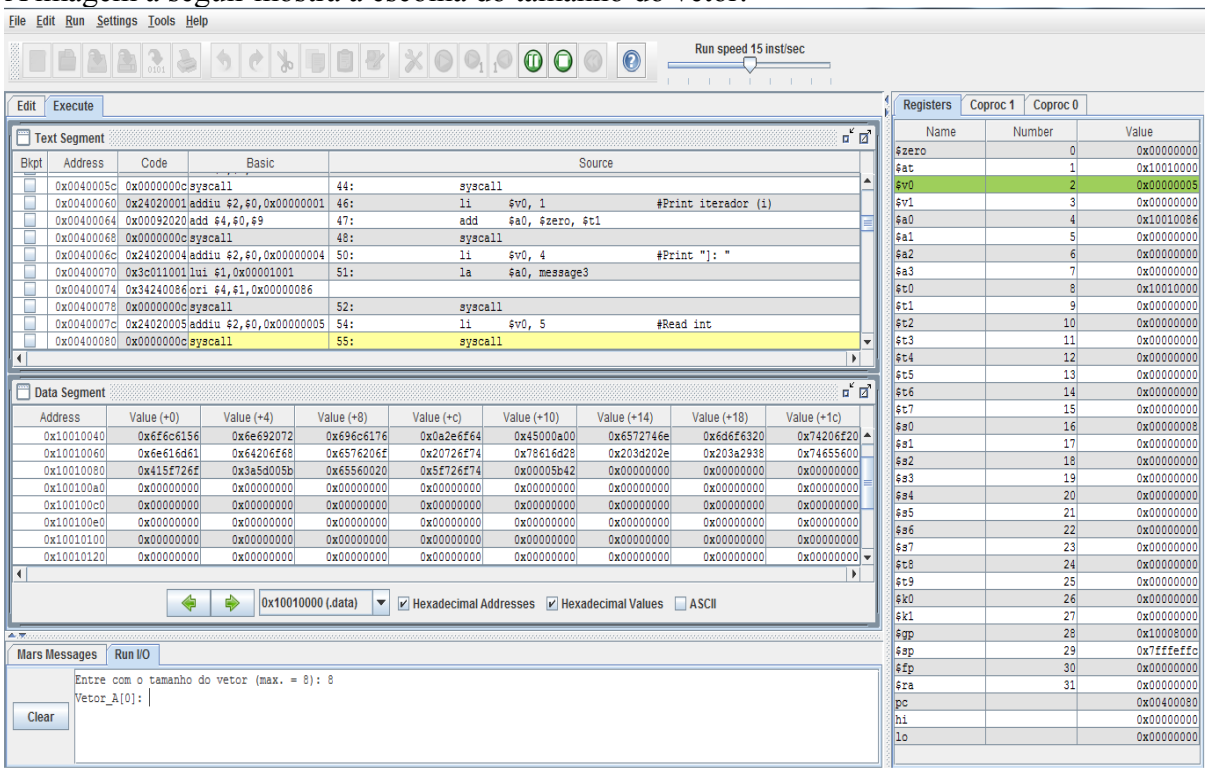
```
# Disciplina: Arquitetura e Organizacao de Computadores
# Atividade: Avaliacao 01 - Programacao em Linguagem de Montagem
# Programa 01
# Grupo: - Mauricio

.data   ######################      DADOS      ######################

Vetor_A:        .word 0,0,0,0,0,0,0,0
Vetor_B:        .word 0,0,0,0,0,0,0,0

invalid_message: .asciiz "Valor invalido.\n"
quebra_linha:    .asciiz "\n"
message1:        .asciiz "Entre com o tamanho do vetor (max. = 8): "
message2:        .asciiz "Vetor_A["
message3:        .asciiz "]: "
message4:        .asciiz "Vetor_B["

.text   ######################      CODIGO      ######################

                j       main                    #Inicia programa no main

invalid:        li      $v0, 4                  #Print "Valor invalido.\n"
                la      $a0, invalid_message
                syscall

main:           li      $v0, 4                  #Print "Entre com o tamanho do vetor (max. = 8): "
                la      $a0, message1
                syscall

                li      $v0, 5                  #Read int
                syscall
                blt     $v0, 1, invalid         #Desvia para invalid se for menor que 1
                bgt     $v0, 8, invalid         #Desvia para invalid se for maior que 8
                add     $s0, $zero, $v0         #Salva tamanho do vetor lido para o registrador $s0

                la      $t0, Vetor_A            #Pega endereço base do Vetor A
                li      $t1, 0                  #Inicia iterador (i)

for:            li      $v0, 4                  #Print "Vetor_A["
                la      $a0, message2
                syscall
```

```
        li      $v0, 1              #Print iterador (i)
        add     $a0, $zero, $t1
        syscall

        li      $v0, 4              #Print "]: "
        la      $a0, message3
        syscall

        li      $v0, 5              #Read int
        syscall

        sw      $v0, ($t0)          #Salva valor lido no vetor

        addi    $t0, $t0, 4         #Incrementa endereço
        addi    $t1, $t1, 1         #Incrementa iterador
        blt     $t1, $s0, for       #Se iterador não chegou no valor volta no loop
        la      $t0, Vetor_B        #Pega endereço base do Vetor B
        li      $t1, 0              #Inicia iterador (i)

for1:   li      $v0, 4              #Print "Vetor_B["
        la      $a0, message4
        syscall

        li      $v0, 1              #Print iterador (i)
        add     $a0, $zero, $t1
        syscall

        li      $v0, 4              #Print "]: "
        la      $a0, message3
        syscall

        li      $v0, 5              #Read int
        syscall

        sw      $v0, ($t0)

        addi    $t0, $t0, 4         #Incrementa endereço
        addi    $t1, $t1, 1         #Incrementa iterador
        blt     $t1, $s0, for1      #Se iterador não chegou no valor volta no loop

        la      $t0, Vetor_A        #Pega endereço base do Vetor A
        la      $t1, Vetor_B        #Pega endereço base do Vetor B
        li      $t2, 0              #Inicializa iterador com 0

for2:   lw      $t3, ($t0)          # $t3 = Vetor_A[i]
        lw      $t4, ($t1)          # $t4 = Vetor_B[i]
        sw      $t3, ($t1)          # Vetor_B[i] = $t3
        sw      $t4, ($t0)          # Vetor_A[i] = $t4
```

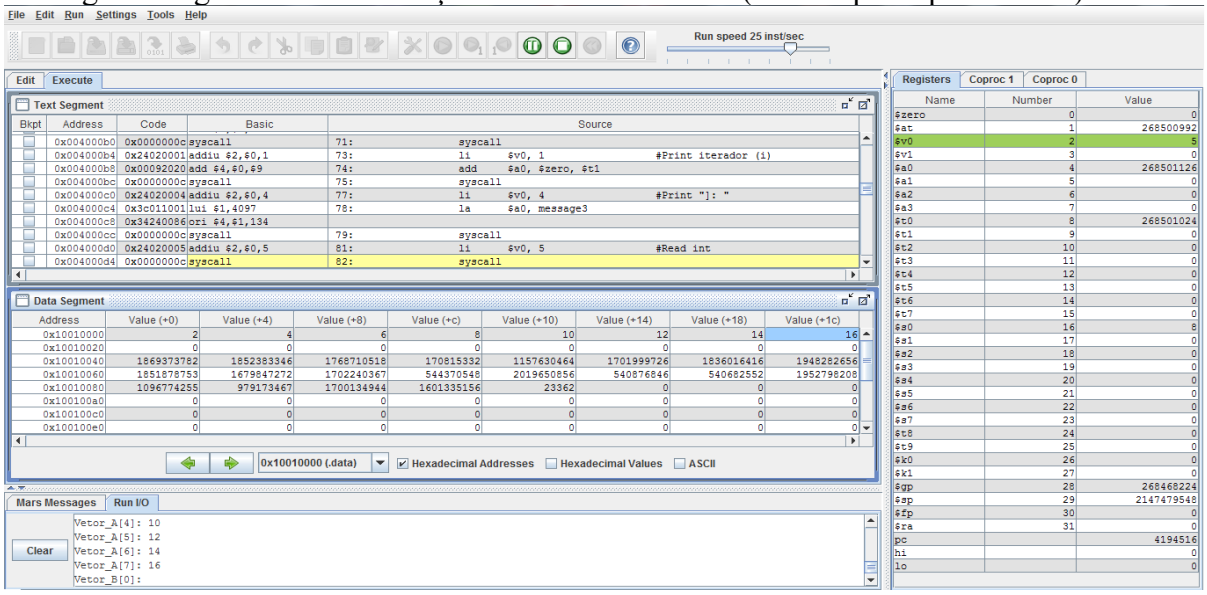
```
        addi    $t0, $t0, 4             #Incrementa endereço do Vetor A
        addi    $t1, $t1, 4             #Incrementa endereço do Vetor B
        addi    $t2, $t2, 1             #Incrementa iterador

        blt     $t2, $s0, for2          #Se iterador não chegou no valor volta no loop
        la      $t0, Vetor_A            #Pega endereço base do Vetor A
        li      $t1, 0                  #Inicia iterador (i)

for3:   li      $v0, 4                  #Print "Vetor_A["
        la      $a0, message2
        syscall

        li      $v0, 1                  #Print iterador (i)
        add     $a0, $zero, $t1
        syscall

        li      $v0, 4                  #Print "]: "
        la      $a0, message3
        syscall

        li      $v0, 1                  #Print Vetor_A[i]
        lw      $a0, ($t0)
        syscall

        li      $v0, 4                  #Print "\n"
        la      $a0, quebra_linha
        syscall

        addi    $t0, $t0, 4             #Incrementa endereço
        addi    $t1, $t1, 1             #Incrementa iterador
        blt     $t1, $s0, for3          #Se iterador não chegou no valor volta no loop
        la      $t0, Vetor_B            #Pega endereço base do Vetor A
        li      $t1, 0                  #Inicia iterador (i)

for4:   li      $v0, 4                  #Print "Vetor_B["
        la      $a0, message4
        syscall

        li      $v0, 1                  #Print iterador (i)
        add     $a0, $zero, $t1
        syscall

        li      $v0, 4                  #Print "]: "
        la      $a0, message3
        syscall

        li      $v0, 1                  #Print Vetor_B[i]
        lw      $a0, ($t0)
        syscall
```

```
li      $v0, 4                  #Print "\n"
la      $a0, quebra_linha
syscall

addi    $t0, $t0, 4             #Incrementa endereço
addi    $t1, $t1, 1             #Incrementa iterador
blt     $t1, $s0, for4          #Se iterador não chegou no valor volta no loop
```

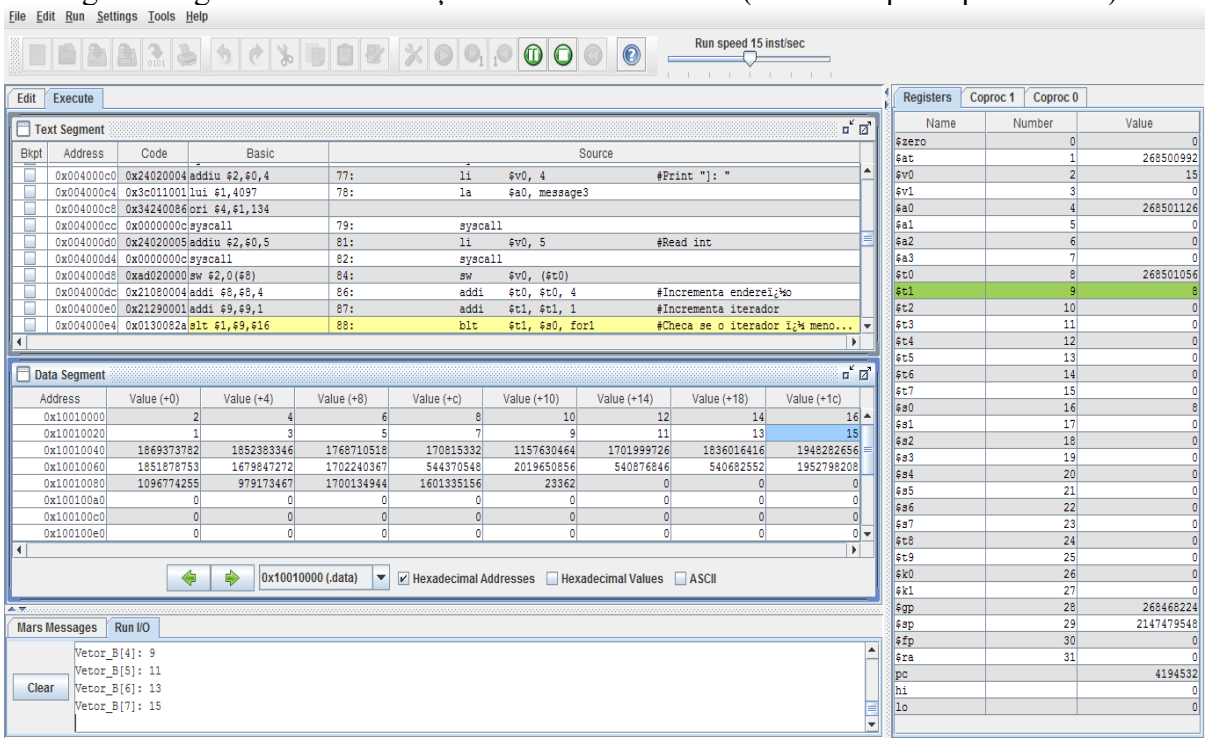
A imagem a seguir mostra a escolha do tamanho do vetor:



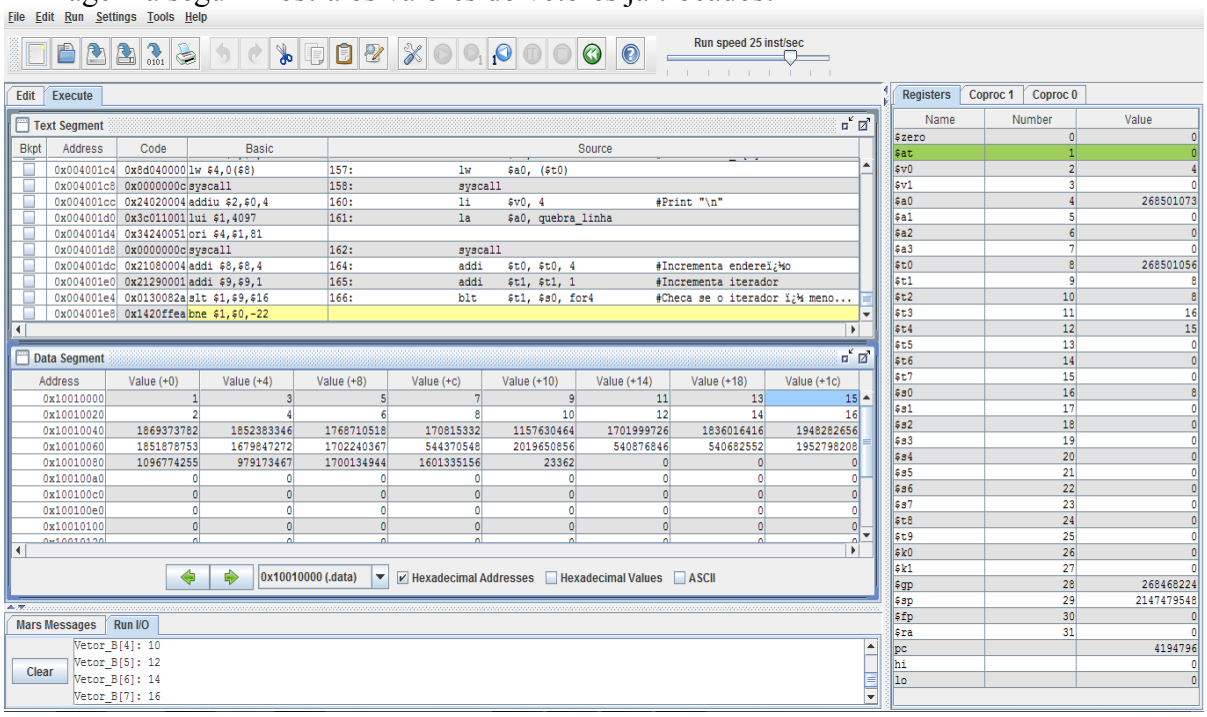
A imagem a seguir mostra a inserção de dados no vetor A(valores pares para o teste):



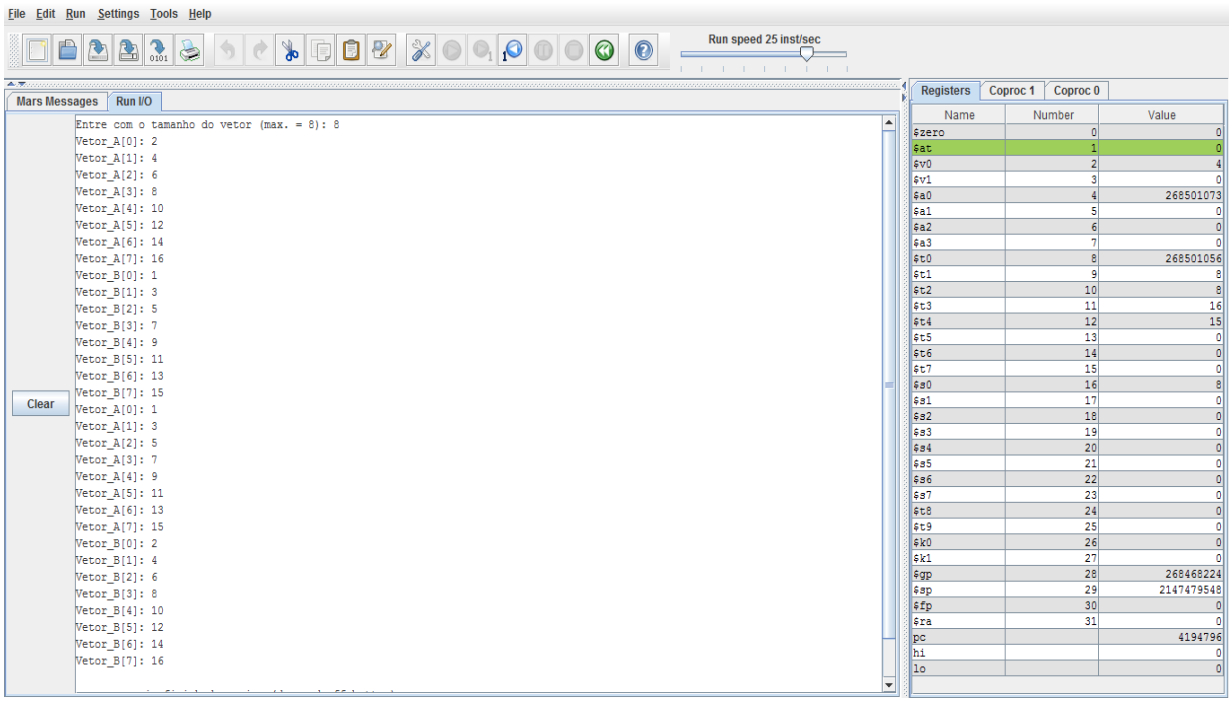
A imagem a seguir mostra a inserção de dados do vetor B(valores impares para o teste):



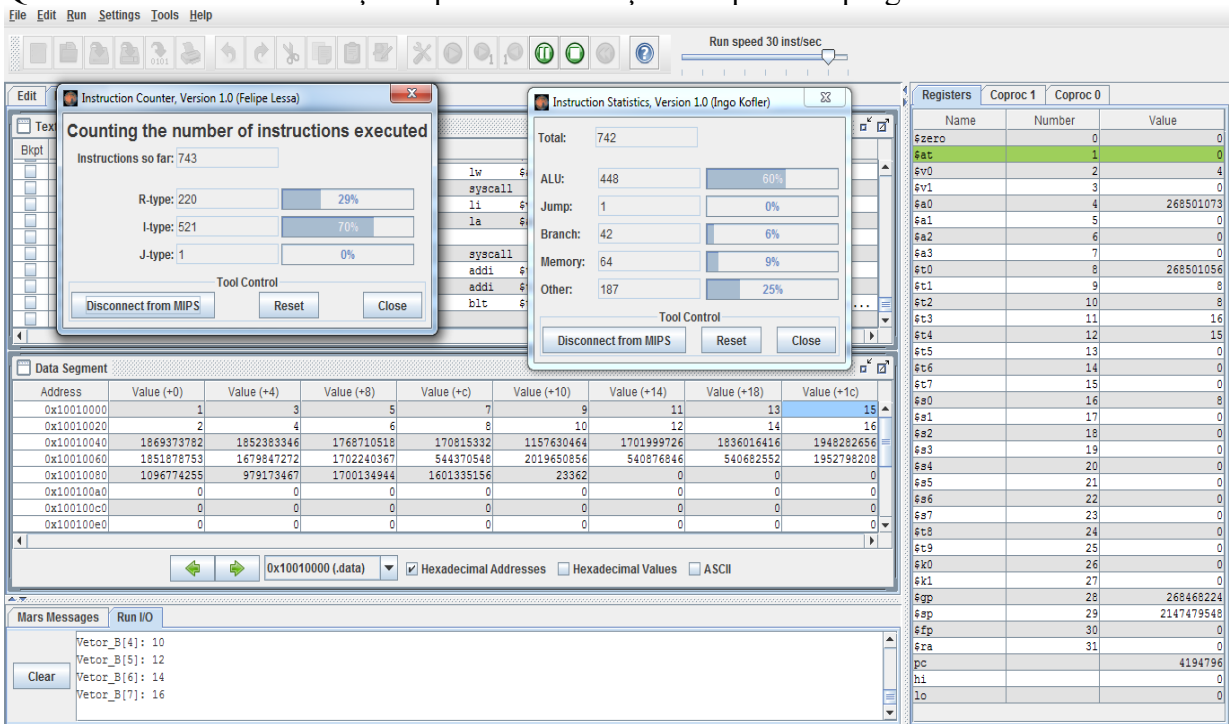
A imagem a seguir mostra os valores do vetores ja trocados:



A imagem a seguir mostra as entradas de dados e a seguir a impressão dos valores invertidos:



Quadro de analise de instruções apos uma execução completa do programa:



Programa 02

Versão do código em C++:

```
#include <iostream>

using namespace std;

int main()
{
   int presenca[16][32];         //Matriz simbolizando presença dos alunos
   int aula;                     //Dia que será escolhido pelo usuario
   int aluno;                    //Aluno que será escolhido pelo usuario
   int registro;                 //Tipo de registro escolhido pelo usuario; presença = 1,
ausência = 0
   while(true)
   {
      do                         //Le qual aula o usuario deseja
      {
         cout << "Entre com o numero da aula (de 0 a 15): ";
         cin >> aula;
      }
      while (aula < 0 || aula> 15);

      do                   //Le qual aluno o usuario deseja
      {
         cout << "Entre com o ńumero do aluno (de 0 a 31): ";
         cin >> aluno;
      }
      while(aluno < 0 || aluno > 31);

      do                   //Le o tipo de registro escolhido
      {
         cout << "Entre com o tipo do registro (presenca = 1;ausencia = 0): ";
         cin >> registro;
      }
      while (registro < 0 || registro > 1);

      presenca[aula][aluno] = registro; //Atribui Falta=0 ou Presença=1 para o aluno na
aula escolhida
   }
   return 0;
}
```

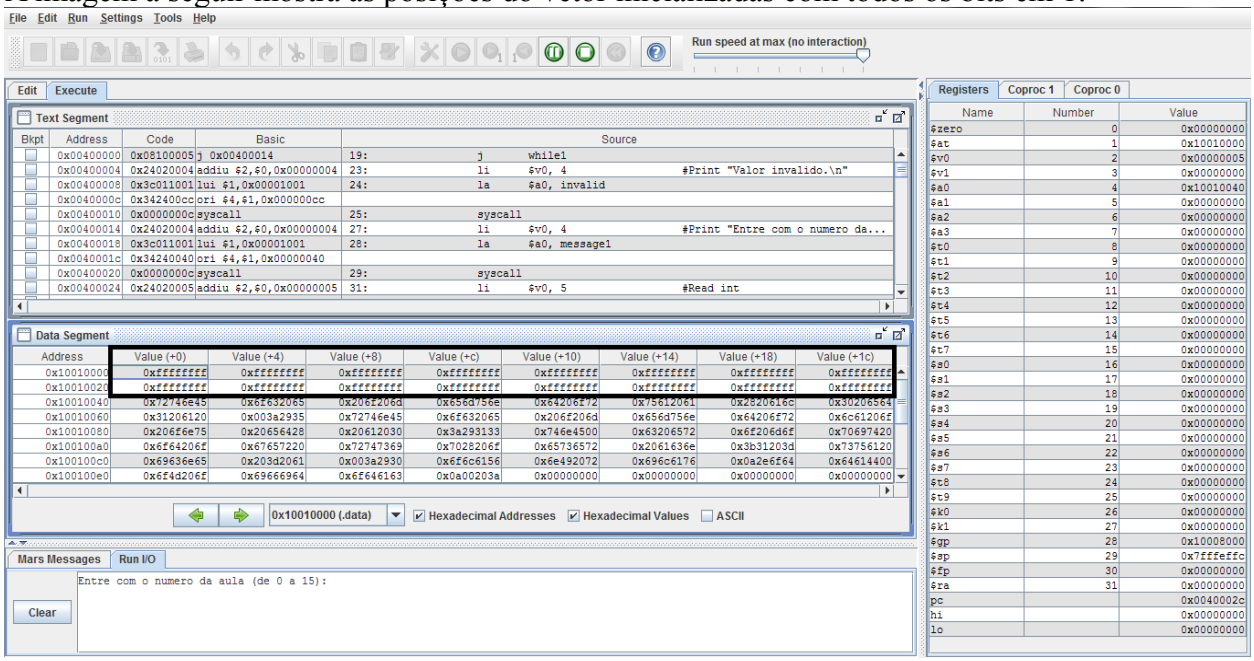
Versão do código em Assembly:

```
# Disciplina: Arquitetura e Organizacao de Computadores
# Atividade: Avaliacao 01 - Programacao em Linguagem de Montagem
# Programa 01
# Grupo: - Mauricio

.data   ######################      DADOS      ######################

presenca:       .word

0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xF
FFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFF
FFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
message1:       .asciiz  "Entre com o numero da aula (de 0 a 15):"
message2:       .asciiz  "Entre com o numero do aluno (de 0 a 31):"
message3:       .asciiz  "Entre com o tipo do registro (presenca = 1; ausencia = 0):"
invalid:        .asciiz "Valor Invalido.\n"
message4:       .asciiz  "Dado Modificado: "
quebra_linha:  .asciiz "\n"

.text   ######################      CODIGO      ######################

while:
                j       while1


invalid1:
                li      $v0, 4                  #Print "Valor invalido.\n"
                la      $a0, invalid
                syscall
while1:
                li      $v0, 4                  #Print "Entre com o numero da aula (de 0 a 15):"
                la      $a0, message1
                syscall

                li      $v0, 5                  #Read int
                syscall
                blt     $v0, 0, invalid1        #Desvia para invalid1 se for menor que 0
                bgt     $v0, 15, invalid1       #Desvia para invalid1 se for maior que 15

                sll     $v0, $v0, 2
                la      $s0, presenca($v0)      #Fazendo a soma para pegar o endereço
                j       while2


invalid2:
                li      $v0, 4                  #Print "Valor invalido.\n"
                la      $a0, invalid
                syscall
while2:
                li      $v0, 4                  #Print "Entre com o numero do aluno (de 0 a 31):"
```

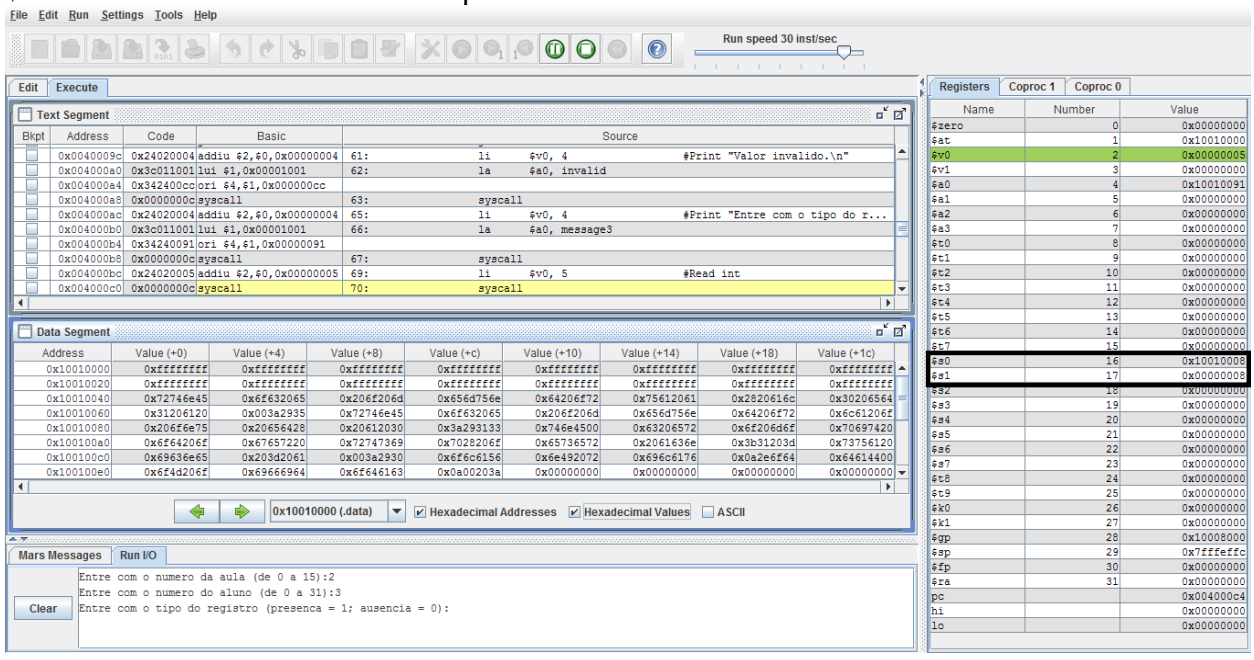
```
                la      $a0, message2
                syscall

                li      $v0, 5                  #Read int
                syscall
                blt     $v0, 0, invalid2        #Desvia para invalid2 se for menor que 0
                bgt     $v0, 31, invalid2       #Desvia para invalid2 se for maior que 31

                addi    $s1, $zero, 1           #Carrega valor base da mascara
                sllv    $s1, $s1, $v0           #Move para esquerda o tanto especificado
                j       while3


invalid3:
                li      $v0, 4                  #Print "Valor invalido.\n"
                la      $a0, invalid
                syscall
while3:
                li      $v0, 4                  #Print "Entre com o tipo do registro:"
                la      $a0, message3
                syscall

                li      $v0, 5                  #Read int
                syscall
                beq     $v0, 1, casopresenca    #Desvia para casopresenca se for igual a 1
                beq     $v0, 0, casoausencia    #Desvia para casoausencia se for igual a 0
                j       invalid3                #Desvia para invalid3 caso valor lido != 0,1

casopresenca:
                lw      $t0, ($s0)              #Carrega valor do vetor para registrador $t0
                or      $t0, $t0, $s1           #Aplica or para registrar presenca
                sw      $t0, ($s0)              #Salva de volta no vetor
                j       print

casoausencia:
                lw      $t0, ($s0)              #Carrega valor do vetor para registrador $t0
                xori    $s1, $s1, 0xFFFFFFFF    #Aplica xor com 1 para inverter os bits
                and     $t0, $t0, $s1           #Aplica and para registrar ausencia
                sw      $t0, ($s0)              #Salva valor no vetor
                j       print

print:
                li      $v0, 4                  #Print "Binario: "
                la      $a0, message4
                syscall

                li      $v0, 35                 #Print Posiçao do vetor em binario
                add     $a0, $t0, $zero
                syscall
```

```
li      $v0, 4                  #Print '\n'
la      $a0, quebra_linha
syscall
j       while
```

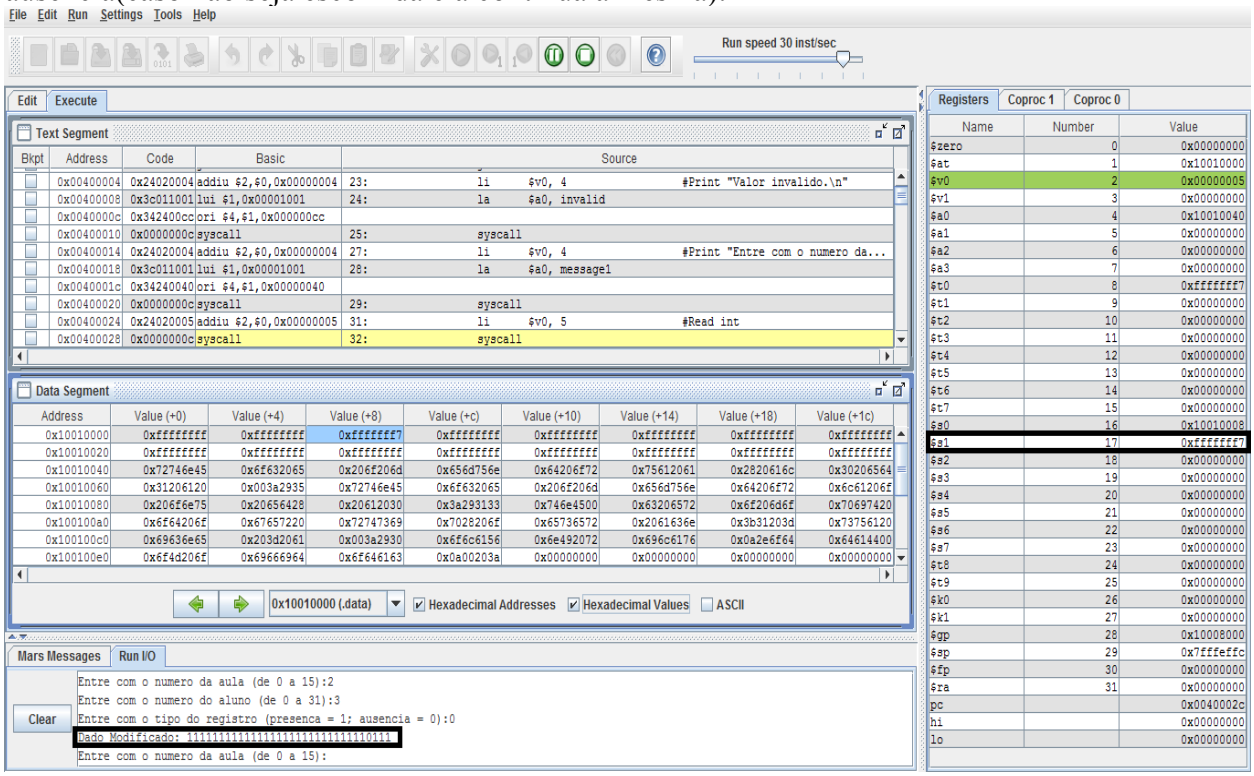
A imagem a seguir mostra as posições do vetor inicializadas com todos os bits em 1:



A imagem a seguir mostra a leitura de dados:
$s0 -> Endereço do vetor selecionado pelo usuario
$s1 -> Matriz de bits movimentada pelo usuario

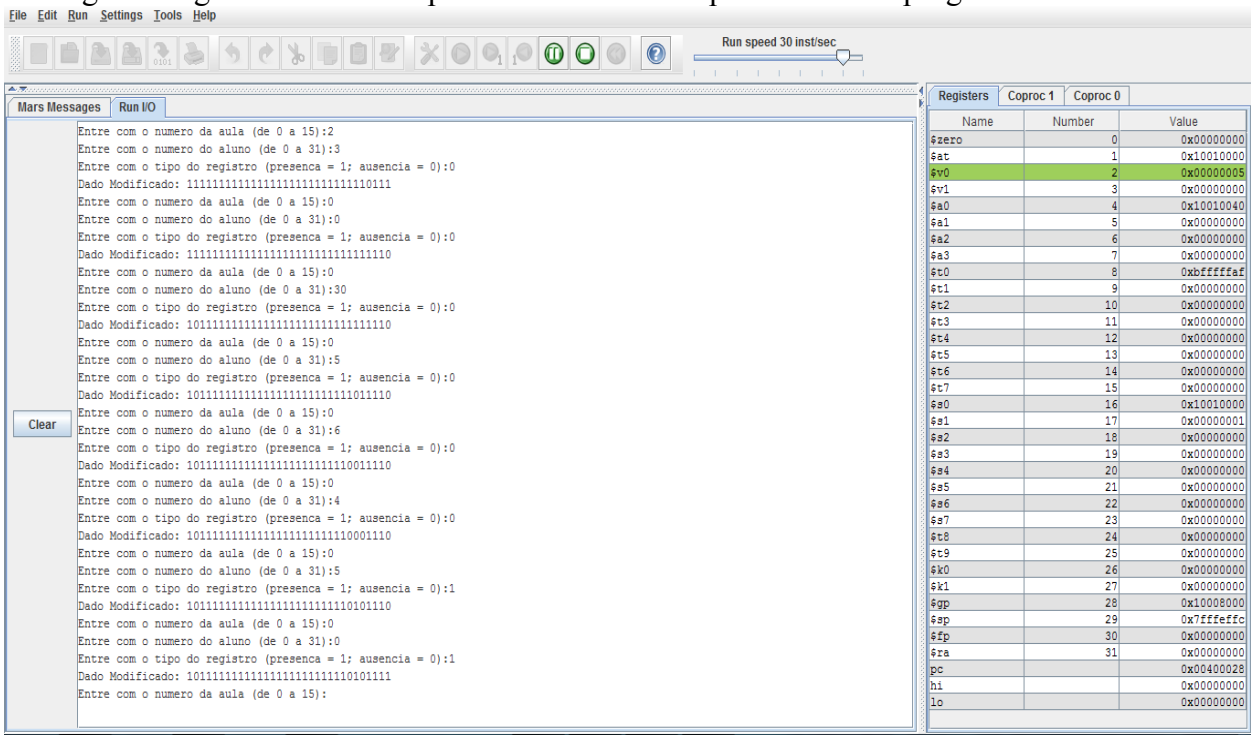


A imagem a seguir mostra como fica a mascara de bits($s1) caso seja escolhida a opção ausência(caso não seja escolhida ela continua a mesma):



Note que o programa também imprime o dado modificado, que seria o dado acessado a partir do endereço do vetor e o bit modificado a partir da operação aplicada com a mascara de bits, seja de presença = OR, ou de ausência = AND.

A imagem a seguir mostra testes para identificar o comportamento do programa:



Quadro de analise de instruções após uma execução única do loop(infinito):