Relatório

Mateus Barbosa e Matheus de Oliveira Rocha

Universidade do Vale do Itajaí - UNIVALI

Escola do Mar, Ciência e Tecnologia

Ciência da Computação

{mateus.barbosa, matheus.rocha}@edu.univali.br

**Arquitetura e Organização de Processadores**

Avaliação 01 – Programação em linguagem de montagem

**Thiago Felski Pereira**

**10/04/2023**

1. Introdução

Este documento é o relatório descrevendo a implementação de 2 programas usando a Linguagem de Montagem do Risc-V, mostrando os valores usado e as estatísticas referentes a execução das Instruções. Além de recriar o código usando uma linguagem de Alto Nível, sendo ela C/C++.

2. Programa 01

**2.1 Enunciado:** Implemente um programa que leia dois vetores via console e, após a leitura dos vetores, troque os conteúdos dos dois vetores. Por fim, o programa deve imprimir esse novo vetor na tela.

**2.2 Código fonte em Linguagem de Alto Nível C/C++**

// Disciplina : Arquitetura e Organização de Computadores

// Atividade : Avaliação 01 – Programação em Linguagem de Alto nível

// Programa 01

// Grupo : - Mateus Barbosa

// - Matheus de Oliveira Rocha

using namespace std;

#include <iostream>

#define MAX\_TAM 8

#define MIN\_TAM 1

int main()

{

int input\_vec\_tam;

// Solicita tamanho dos vetores

do

{

cout << "\nEntre com o tamanho dos vetores (máx = 8): ";

cin >> input\_vec\_tam;

if (input\_vec\_tam < MIN\_TAM || input\_vec\_tam > MAX\_TAM)

cout << "\nValor inválido!\n";

} while (input\_vec\_tam < MIN\_TAM || input\_vec\_tam > MAX\_TAM);

int Vetor\_A[input\_vec\_tam], Vetor\_B[input\_vec\_tam];

// Preenche Vetor\_A com os valores digitados pelo usuario

for (int i = 0; i < input\_vec\_tam; i++)

{

cout << "\nVetor\_A[" << i << "] = ";

cin >> Vetor\_A[i];

}

// Preenche Vetor\_B com os valores digitados pelo usuario

for (int i = 0; i < input\_vec\_tam; i++)

{

cout << "\nVetor\_B[" << i << "] = ";

cin >> Vetor\_B[i];

}

// Troca os valores dos Vetores

int auxiliar; // Usado como variavel temp para armazenar o valor de um dos Vetores

for (int i = 0; i < input\_vec\_tam; i++)

{

auxiliar = Vetor\_A[i];

Vetor\_A[i] = Vetor\_B[i];

Vetor\_B[i] = auxiliar;

}

// Mostra Vetor\_A com os valores tracodos com o Vetor\_B

for (int i = 0; i < input\_vec\_tam; i++)

{

cout << "\nVetor\_A[" << i << "] = " << Vetor\_A[i];

}

// Mostra Vetor\_B com os valores tracodos com o Vetor\_A

for (int i = 0; i < input\_vec\_tam; i++)

{

cout << "\nVetor\_B[" << i << "] = " << Vetor\_B[i];

}

return 0;

}

**2.3 Código fonte em Linguagem de Montagem do Risc-V**

**Explicação da Lógica do Código:**O progrma solicita ao usuário o tamanho de dois vetores, em seguida, solicita ao usuário os valores que compõem o primeiro vetor e, em seguida, os valores que compõem o segundo vetor. A lógica por trás do programa é simples, mas é um pouco extensa.

Inicialmente, há um segmento de dados onde é definido o valor dos vetores Vetor\_A e Vetor\_B com o valor 0, além do tamanho máximo e mínimo permitido para o vetor (8 e 1, respectivamente) e as strings que serão usadas no decorrer do programa.

Em seguida, há um segmento de código que começa com a carga dos registradores t0 e t1 com o valor de min\_tam e max\_tam, respectivamente. Em seguida, são carregados os registradores s10 e s11 com Vetor\_A e Vetor\_B, respectivamente.

O programa entra em um loop chamado "loop\_define\_vetor\_tam" onde o usuário é solicitado a inserir o tamanho dos vetores. Se o valor inserido pelo usuário for menor que 1, uma mensagem de erro será exibida e o loop será executado novamente. Se o valor inserido pelo usuário for maior que 8, uma mensagem de erro será exibida e o loop será executado novamente. Caso contrário, o programa avança para o loop "define\_Vetor\_A\_contador".

A próxima função "define\_Vetor\_A\_contador" define um contador (t0) como 0 e carrega o texto "texto\_input\_Vetor\_A" para o registrador t1. Em seguida, ele chama a função "loop\_preenche\_Vetor\_A".

O loop "loop\_preenche\_Vetor\_A" solicita que o usuário insira um valor para a posição atual do vetor, exibe o índice da posição atual e armazena o valor digitado pelo usuário no vetor. O loop é executado até que o contador (t0) seja maior ou igual ao tamanho informado pelo usuário. Se o contador for menor que o tamanho informado, o programa solicita o próximo valor do usuário, caso contrário, o programa passa para a próxima etapa, que é a definição do segundo vetor.

A próxima função "define\_Vetor\_B\_contador" é semelhante à função "define\_Vetor\_A\_contador", mas carrega "texto\_input\_Vetor\_B" no registrador t1 e chama a função "loop\_preenche\_Vetor\_B", que solicita os valores do segundo vetor.

O loop "loop\_preenche\_Vetor\_B" é semelhante ao "loop\_preenche\_Vetor\_A", mas é usado para preencher o vetor Vetor\_B.

Após os 2 vetores estarem preenchido, é executada a função chamada "troca\_valores\_vetor", que é responsável por trocar o conteúdo dos dois vetores "Vetor\_A" e "Vetor\_B" entre si. O código usa um loop para percorrer os vetores e realiza a troca dos valores de cada posição. O loop começa no rótulo "loop\_troca\_valores\_vetor" e termina quando o contador t0 é maior ou igual ao tamanho informado pelo usuário, após passa a executar a instrução na label "print\_Vetor\_A".

Dentro do loop, as instruções calculam as posições do Vetor\_A e Vetor\_B na memória, carregam os valores nas respectivas posições em registradores auxiliares e, em seguida, trocam os valores entre si usando as instruções "sw" (store word) e "lw" (load word). Por fim, o contador é incrementado e o loop é reiniciado com a instrução "j" (jump) para "loop\_troca\_valores\_vetor".

Após isso, vai para a parte final do código em que se mostra os valores no terminal que é feita pela função print\_Vetor\_A começa inicializando o contador t0 com o valor zero e em seguida chama o loop loop\_print\_Vetor\_A. O loop verifica se o contador é maior ou igual ao tamanho do vetor s0. Se sim, ele sai do loop e chama a função print\_Vetor\_B. Se não, ele calcula a posição atual do vetor A, imprime a string "A[i] = " e o valor naquela posição do vetor A. Em seguida, ele incrementa o contador t0 e reinicia o loop.

A função print\_Vetor\_B é similar, com a diferença que ela calcula a posição atual do vetor B e imprime a string "B[i] = " e o valor naquela posição do vetor B.

Ambas as funções usam a syscall PrintString para imprimir as strings na tela e a syscall PrintInt para imprimir os valores dos vetores. Ao final do loop da função print\_Vetor\_B, o programa chama a função Exit que faz uma syscall Exit para finalizar a execução do programa.

**Código do Risc-V:**

# Disciplina: Arquitetura e Organização de Computadores

# Atividade: Avaliação 01 – Programação em Linguagem de Montagem

# Programa 01

# Grupo: - Mateus Barbosa

# - Matheus de Oliveira Rocha

.data # Segmento de Dados

Vetor\_A: .word 0, 0, 0, 0, 0, 0, 0, 0 # Inicializa um vetor com 8 posições com o valor 0

Vetor\_B: .word 0, 0, 0, 0, 0, 0, 0, 0 # Inicializa um vetor com 8 posições com o valor 0

max\_tam: .word 8

min\_tam: .word 1

texto\_input\_max\_tam: .asciz "\nEntre com o tamanho dos vetores (máx = 8): "

texto\_valor\_invalido: .asciz "\nValor inválido!\n"

texto\_input\_Vetor\_A: .asciz "\nVetor\_A["

texto\_input\_Vetor\_B: .asciz "\nVetor\_B["

texto\_fecha\_index: .asciz "] = " # Essa string será usada para evitar ter que fazer algo muito trabalhoso para adicionar o número do contador

.text # Segmento de Código

lw t0, min\_tam

lw t1, max\_tam

la s10, Vetor\_A # Carrega o Vetor\_A no registrador s10

la s11, Vetor\_B # Carrega o Vetor\_B no registrador s11

loop\_define\_vetor\_tam:

# Imprime: String texto\_input\_max\_tam

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_max\_tam # Carrega ao registrador a0 o texto\_input\_max\_tam

ecall # Chama a syscall

# Solicita: Int tamanho do vetor

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s0, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s0

bge s0, t0, if\_maior\_que # Se o valor de s0 (sem sinal) for maior que 1, vai para a função if\_maior\_que

# Imprime: String valor\_invalido

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_valor\_invalido # Carrega ao registrador a0 o valor\_invalido

ecall # Chama a syscall

j loop\_define\_vetor\_tam # "Pula" para a função loop\_define\_vetor\_tam

if\_maior\_que: # Função que verifica se o valor informado é menor que tam

ble s0, t1, define\_Vetor\_A\_contador # Se o valor de s0 (sem sinal) for menor que max\_tam, vai para a função loop\_preenche\_vetor\_A

# Imprime: String valor\_invalido

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_valor\_invalido # Carrega ao registrador a0 o valor\_invalido

ecall # Chama a syscall

j loop\_define\_vetor\_tam # "Pula" para a função loop\_define\_vetor\_tam

define\_Vetor\_A\_contador:

li t0, 0 # Define o counter para o valor 0

la t1, texto\_input\_Vetor\_A # Carrega o texto texto\_input\_vetor\_A no registrador t1

j loop\_preenche\_Vetor\_A # Executa a função loop\_preenche\_Vetor\_A

loop\_preenche\_Vetor\_A:

bge t0, s0, define\_Vetor\_B\_contador # Se o contador foi maior que o tamanho informado pelo usuario, sai do loop

# Imprime: String texto\_input\_vetor\_A

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_Vetor\_A # Carrega ao registrador a0 o texto\_input\_vetor\_A

ecall # Chama a syscall

# Imprime: Int contador ou i

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

add a0, zero, t0 # Carrega ao registrador a0 o contador

ecall # Chama a syscall

# Imprime: String texto\_fecha\_index

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_fecha\_index # Carrega ao registrador a0 o texto\_fecha\_index

ecall # Chama a syscall

# Solicita: Int valor na posição atual

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s1, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s0

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t2, s10, t1 # Calcula a posicao no Vetor\_A desde o seu comeco: comeco\_do\_Vetor + (4 \* i)

# Adiciona o valor na posiocao calculada do Vetor\_A

sw s1, 0(t2) # Guarda o valor informado pelo usuario no s1, com um offset de 0 bits, no Vetor\_A (s10)

addi t0, t0, 1 # Incrementa o contador: i = i + 1

j loop\_preenche\_Vetor\_A # "Reinicia" o loop

define\_Vetor\_B\_contador:

li t0, 0 # Define o counter para o valor 0

la t1, texto\_input\_Vetor\_B # Carrega o texto texto\_input\_vetor\_B no registrador t1

j loop\_preenche\_Vetor\_B # Executa a função loop\_preenche\_Vetor\_B

loop\_preenche\_Vetor\_B:

bge t0, s0, troca\_valores\_vetor # Se o contador foi maior que o tamanho informado pelo usuario, sai do loop

# Imprime: String texto\_input\_vetor\_B

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_Vetor\_B # Carrega ao registrador a0 o texto\_input\_vetor\_B

ecall # Chama a syscall

# Imprime: Int contador ou i

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

add a0, zero, t0 # Carrega ao registrador a0 o contador

ecall # Chama a syscall

# Imprime: String texto\_fecha\_index

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_fecha\_index # Carrega ao registrador a0 o texto\_fecha\_index

ecall # Chama a syscall

# Solicita: Int valor na posição atual

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s1, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s0

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t2, s11, t1 # Calcula a posicao no Vetor\_A desde o seu comeco: comeco\_do\_Vetor + 4 \* i

# Adiciona o valor na posiocao calculada do Vetor\_A

sw s1, 0(t2) # Guarda o valor informado pelo usuario no s1, com um offset de 0 bits, no Vetor\_B (s11)

addi t0, t0, 1 # Incrementa o contador: i = i + 1

j loop\_preenche\_Vetor\_B # "Reinicia" o loop

troca\_valores\_vetor:

li t0, 0

j loop\_troca\_valores\_vetor

loop\_troca\_valores\_vetor:

bge t0, s0, print\_Vetor\_A # Se o contador foi maior que o tamanho informado pelo usuario, sai do loop

# Calcula a posicao no Vetor\_A (s10)

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t2, s10, t1 # Calcula a posicao no Vetor\_A desde o seu comeco: comeco\_do\_Vetor + 4 \* i

lw t3, 0(t2) # Carrega no registrador t3 (auxiliar), sem offset de bits, o conteudo do Vetor\_A na posicao atual

# Calcula a posicao no Vetor\_B (s11)

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t4, s11, t1 # Calcula a posicao no Vetor\_B desde o seu comeco: comeco\_do\_Vetor + 4 \* i

lw t5, 0(t4) # Carrega no registrador t4, sem offset de bits, o conteudo do Vetor\_B na posicao atual

sw t5, 0(t2) # Vetor\_B[i] = Vetor\_A[i]

sw t3, 0(t4) # Vetor\_A[i] = Vetor\_B[i]

addi t0, t0, 1 # Incrementa o contador: i = i + 1

j loop\_troca\_valores\_vetor # "Reinicia" o loop

print\_Vetor\_A:

li t0, 0

j loop\_print\_Vetor\_A

loop\_print\_Vetor\_A:

bge t0, s0, print\_Vetor\_B # Se o contador foi maior que o tamanho informado pelo usuario, sai do loop

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t2, s10, t1 # Calcula a posicao no Vetor\_A desde o seu comeco: comeco\_do\_Vetor + 4 \* i

# Imprime: String texto\_input\_vetor\_A

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_Vetor\_A # Carrega ao registrador a0 o texto\_input\_vetor\_A

ecall # Chama a syscall

# Imprime: Int contador ou i

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

add a0, zero, t0 # Carrega ao registrador a0 o contador

ecall # Chama a syscall

# Imprime: String texto\_fecha\_index

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_fecha\_index # Carrega ao registrador a0 o texto\_fecha\_index

ecall # Chama a syscall

# Imprime: Int valor na posição atual

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

lw a0, 0(t2) # Adciciona ao registrador a0, o valor daquela posicao do vetor

ecall # Chama a syscall

addi t0, t0, 1 # Incrementa o contador: i = i + 1

j loop\_print\_Vetor\_A # "Reinicia" o loop

print\_Vetor\_B:

li t0, 0

j loop\_print\_Vetor\_B

loop\_print\_Vetor\_B:

bge t0, s0, exit # Se o contador foi maior que o tamanho informado pelo usuario, sai do loop

slli t1, t0, 2 # Move 2 bits para a esquerda: 4 \* i

add t2, s11, t1 # Calcula a posicao no Vetor\_A desde o seu comeco: comeco\_do\_Vetor + 4 \* i

# Imprime: String texto\_input\_vetor\_A

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_Vetor\_B # Carrega ao registrador a0 o texto\_input\_vetor\_B

ecall # Chama a syscall

# Imprime: Int contador ou i

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

add a0, zero, t0 # Carrega ao registrador a0 o contador

ecall # Chama a syscall

# Imprime: String texto\_fecha\_index

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_fecha\_index # Carrega ao registrador a0 o texto\_fecha\_index

ecall # Chama a syscall

# Imprime: Int valor na posição atual

addi a7, zero, 1 # Adiciona o valor 1 (PrintInt) ao registrador de serviço a7

lw a0, 0(t2) # Adciciona ao registrador a0, o valor daquela posicao do vetor

ecall # Chama a syscall

addi t0, t0, 1 # Incrementa o contador: i = i + 1

j loop\_print\_Vetor\_B # "Reinicia" o loop

exit:

addi a7, zero, 10 # Usa a diretir Exit (10)

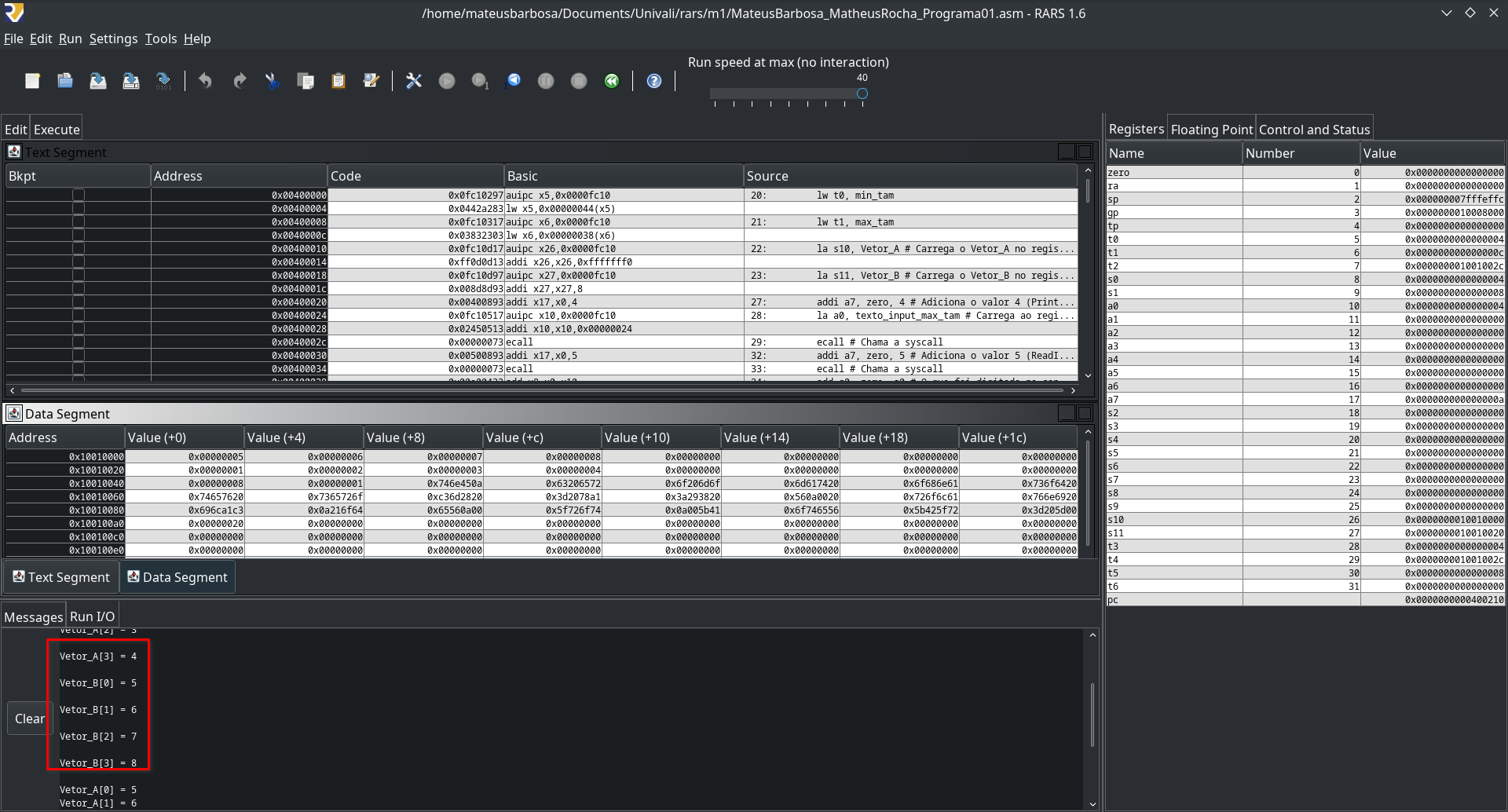
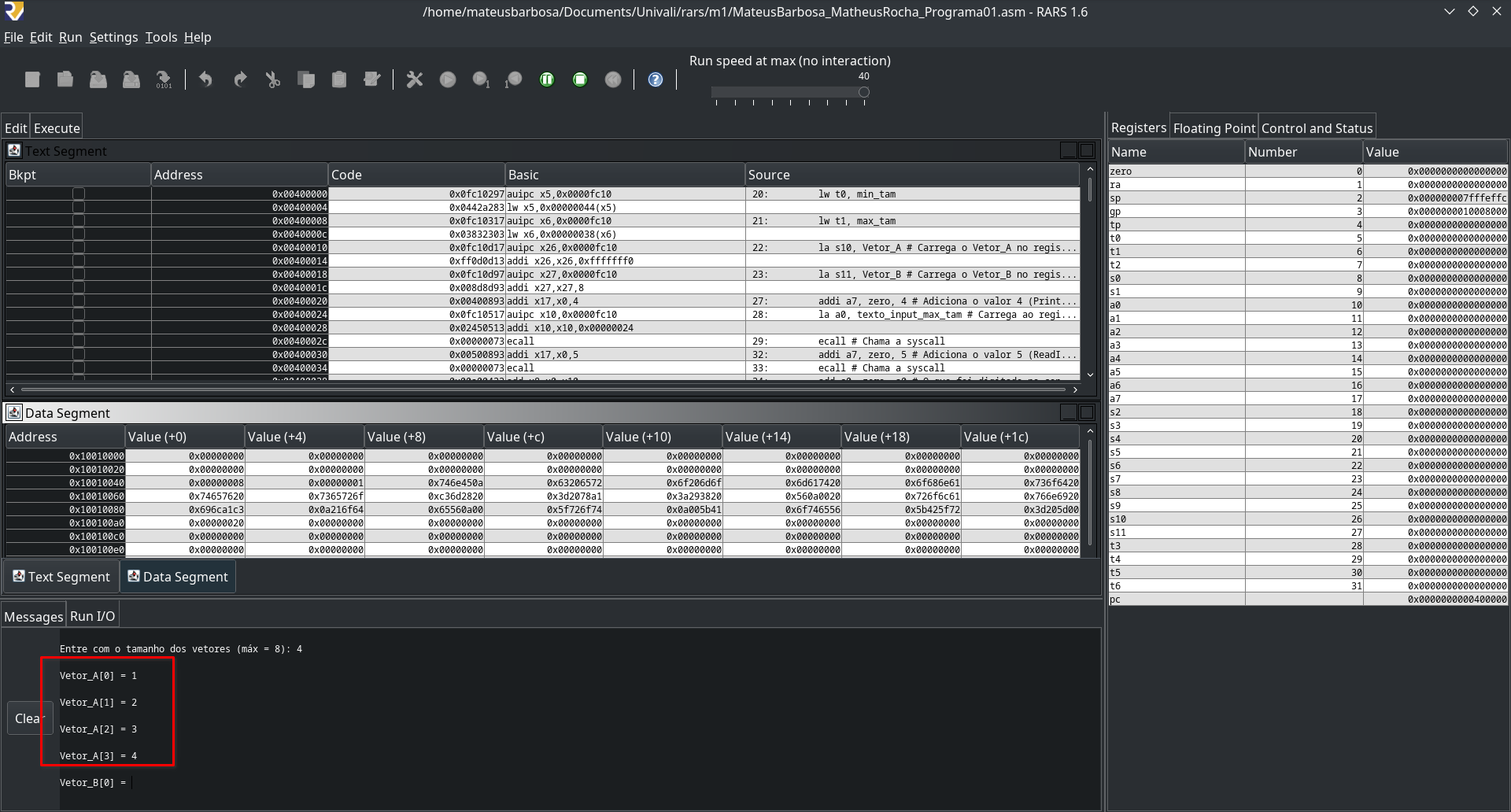
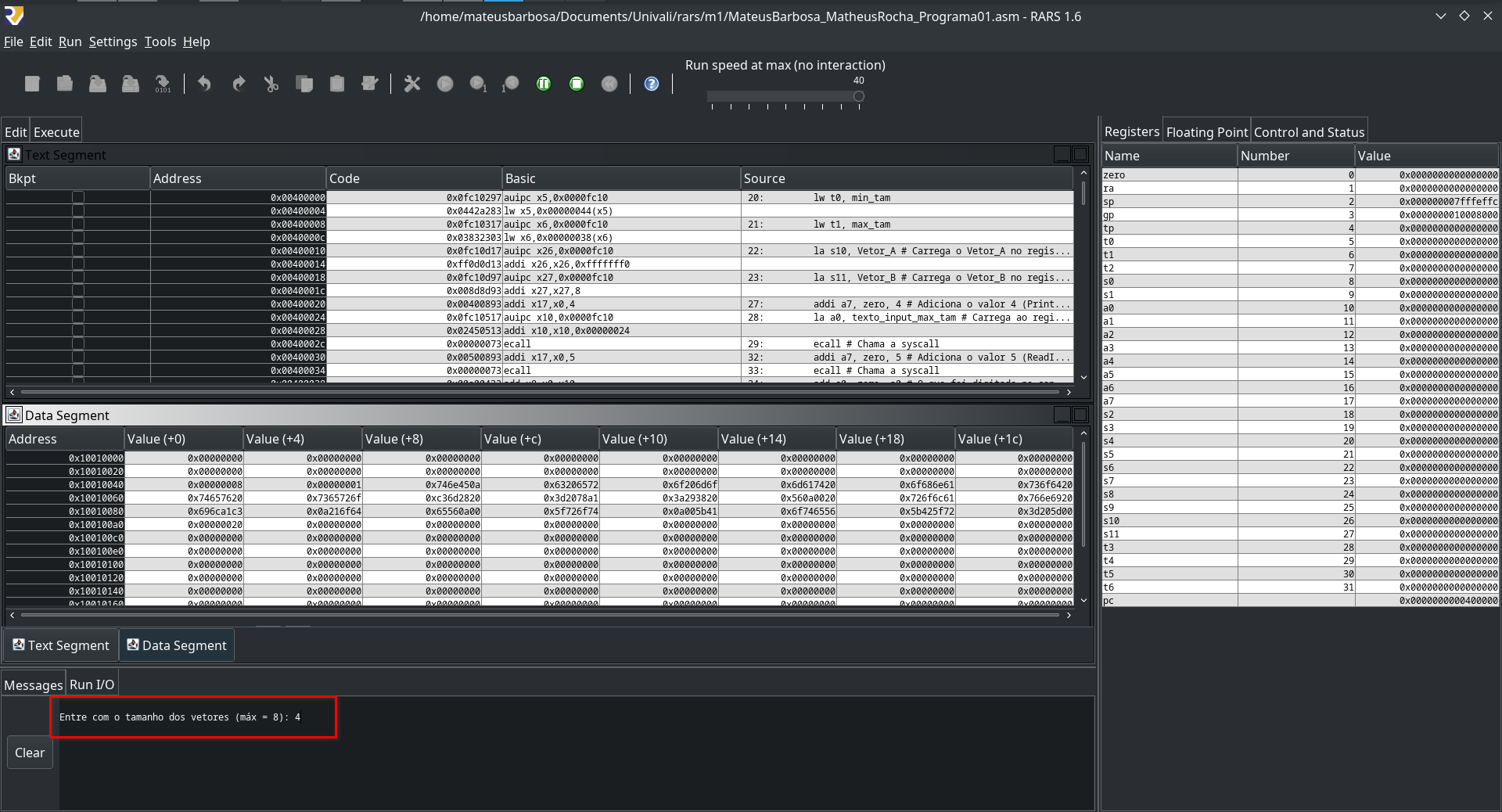
ecall

**2.4 Resultados**

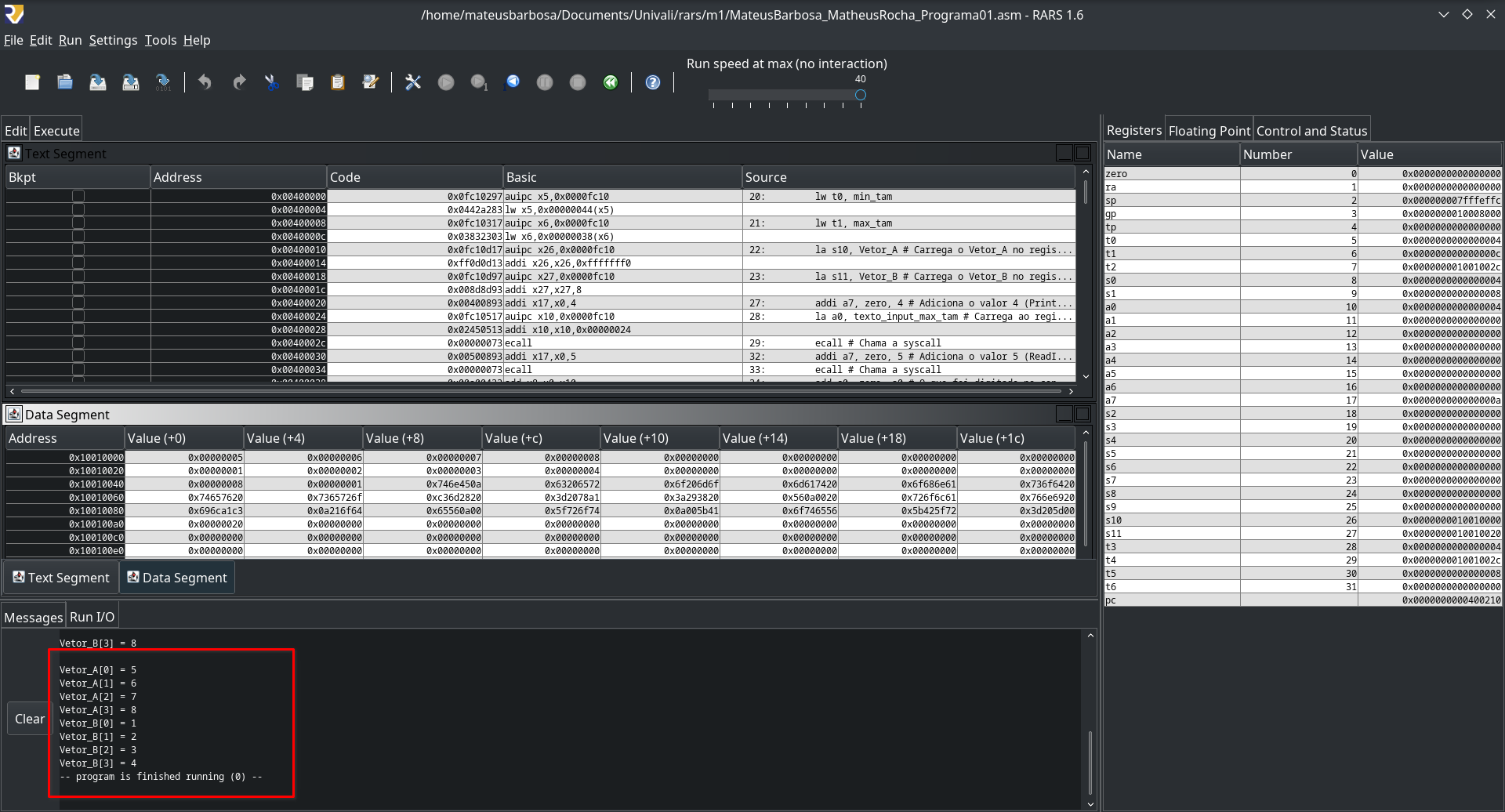
Informações da execução:

* Tamanho dos Vetores: 4
* Elementos do Vetor\_A em ordem de inserção: 1, 2, 3, 4
* Elementos do Vetor\_B em ordem de inserção: 5, 6, 7, 8
* Resultado Final da Operação de troca:
  + Vetor\_A: 5, 6, 7, 8
  + Vetor\_B: 1, 2, 3, 4

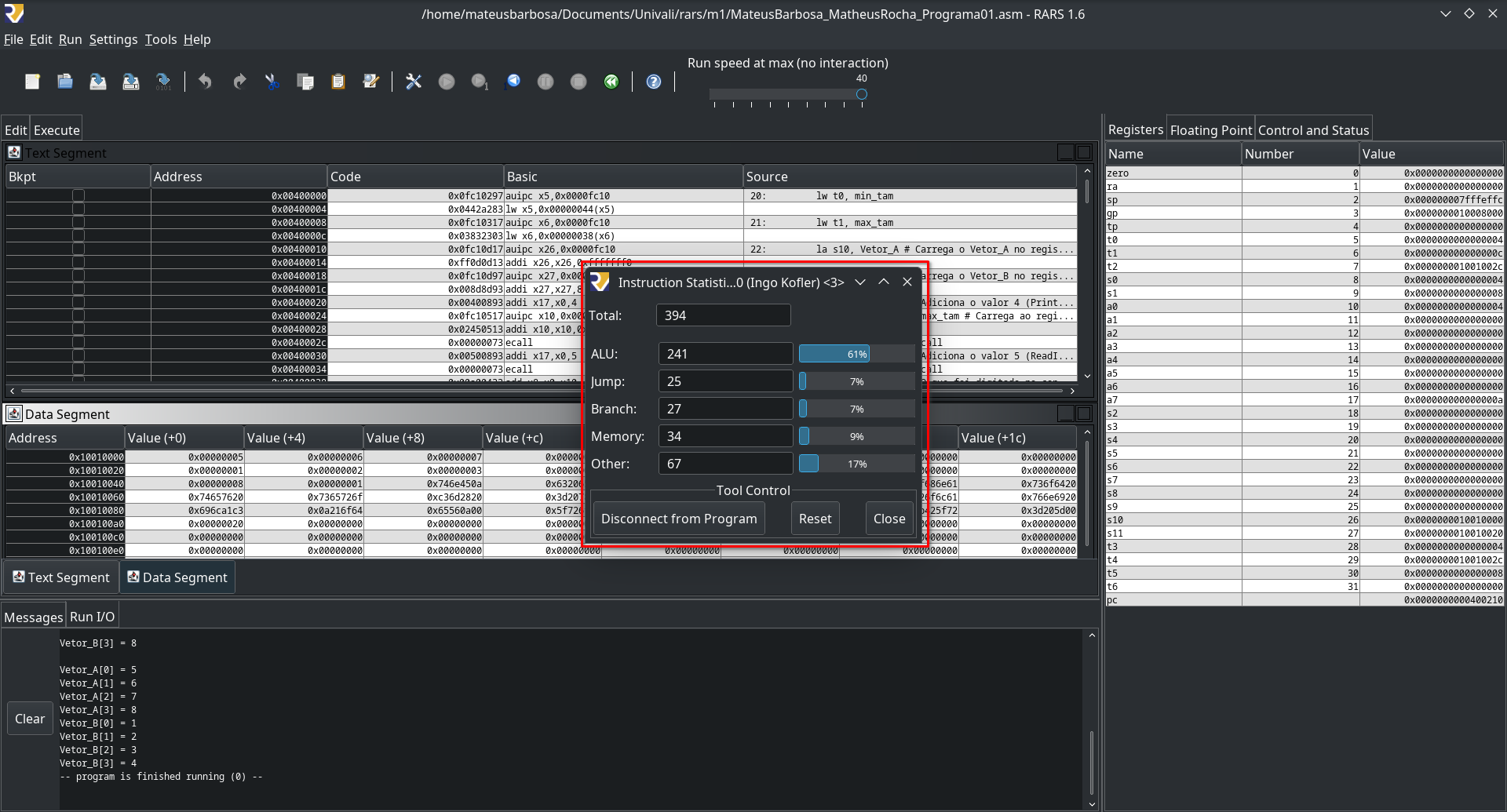
Abaixo são as telas de capturas (Prints) da execução do Programa inserindo os valores informados acima:



Abaixo é o resultado da operação de trocar os valores dos vetores um com o outro.



Abaixo é a inclusão do Instruction Statistics no Programa e seus resultados de cada instrução realizada:



3. Programa 02

**3.1 Enunciado:** Considere uma disciplina com 16 dias de aulas e que é necessário registrar a presença de cada aluno em cada dia, consumindo o mínimo de variáveis em memória. Assumindo que a turma tenha no máximo 32 alunos, implemente um programa que utilize um vetor de 16 elementos inteiros (um elemento para cada dia) para registrar a frequência dos alunos de cada disciplina e que ofereça ao professor funções para registrar a presença e ausência de qualquer aluno em qualquer dia.

**3.2 Código fonte em Linguagem de Alto Nível C/C++**

// Disciplina : Arquitetura e Organização de Computadores

// Atividade : Avaliação 01 – Programação em Linguagem de Alto nível

// Programa 02

// Grupo : - Mateus Barbosa

// - Matheus de Oliveira Rocha

using namespace std;

#include <iostream>

#include <string>

#define MAX\_AULA 15

#define MAX\_ALUNO 31

#define MAX\_REGISTRO 1

int main()

{

string Vetor\_Dias[MAX\_AULA]; // Por simplicidade, utilizamos uma string contendo os 32 bits dos alunos

// Preenche todos os dias com todos os alunos presentes

for (int i = 0; i < MAX\_AULA; i++)

{

Vetor\_Dias[i] = "1111111111111111111111111111111";

}

while (true)

{

// Solicita o número da aula

int input\_aula;

do

{

cout << "\nEntre com o número da aula (de 0 a 15): ";

cin >> input\_aula;

if (!cin.good()) // Sai deste loop se o input não for número

break;

else if (input\_aula < 0 || input\_aula > MAX\_AULA)

cout

<< "\nValor inválido";

} while (input\_aula < 0 || input\_aula > MAX\_AULA);

if (!cin.good()) // Sai do loop principal se o input não for número

break;

// Solicita o número do aluno

int input\_aluno;

do

{

cout << "\nEntre com o número do aluno (de 0 a 31): ";

cin >> input\_aluno;

if (!cin.good()) // Sai deste loop se o input não for número

break;

else if (input\_aluno < 0 || input\_aluno > MAX\_ALUNO)

cout << "\nValor inválido";

} while (input\_aluno < 0 || input\_aluno > MAX\_ALUNO);

if (!cin.good()) // Sai do loop principal se o input não for número

break;

// Solicita o tipo do registro

int input\_registro;

do

{

cout << "\nEntre com o tipo do registro (presença = 1; ausência = 0): ";

cin >> input\_registro;

if (!cin.good()) // Sai deste loop se o input não for número

break;

else if (input\_registro < 0 || input\_registro > MAX\_REGISTRO)

cout

<< "\nValor inválido";

} while (input\_registro < 0 || input\_registro > MAX\_REGISTRO);

if (!cin.good()) // Sai do loop principal se o input não for número

break;

Vetor\_Dias[input\_aula][(MAX\_ALUNO - 1) - input\_aluno] = to\_string(input\_registro)[0]; // Para simular que as string são bits, inserimos da direita para a esquerda, isso é alcançado usando a formula (MAX\_ALUNO - 1) - input\_aluno

}

// O código abaixo simularia o campo Data Segment na aba Execute do RARS

cout << "\nData Segment:\n";

for (int i = 0; i < MAX\_AULA; i++)

{

cout << i << "\t=> " << Vetor\_Dias[i] << endl;

}

return 0;

}

**3.3 Código fonte em Linguagem de Montagem do Risc-V**

**Explicação da Lógica do Código:**

O programa utiliza uma sequência de strings formatadas e chamadas de sistema para receber as entradas do usuário (aula, aluno e tipo de registro) e validar as entradas fornecidas.

Na seção ".data", são definidos os dados, incluindo o vetor de dias (Vetor\_Dias) que guarda o registro de presença/falta de cada aluno em cada aula, uma máscara de bits (mask) que será utilizada posteriormente e várias strings formatadas que serão utilizadas nas chamadas de sistema.

Na seção ".text", a função "main" é definida como ponto de partida para o programa, que então chama a função "loop\_define\_aula". Dentro dessa função, o programa exibe a string "texto\_input\_aula" e recebe um valor inteiro do usuário que corresponde ao número da aula. Em seguida, o programa verifica se o valor fornecido é maior ou igual a zero e menor ou igual a "max\_aula". Se for, ele passa para a função "loop\_define\_aluno", caso contrário, exibe a string "texto\_valor\_invalido" e chama a função "loop\_define\_aula" novamente.

A função "loop\_define\_aluno" é semelhante à função "loop\_define\_aula", exibindo a string "texto\_input\_aluno" e recebendo um valor inteiro do usuário que corresponde ao número do aluno. Em seguida, o programa verifica se o valor fornecido é maior ou igual a zero e menor ou igual a "max\_aluno". Se for, ele passa para a função "loop\_define\_registro", caso contrário, exibe a string "texto\_valor\_invalido" e chama a função "loop\_define\_aluno" novamente.

A função "loop\_define\_registro" é novamente semelhante às funções anteriores, exibindo a string "texto\_input\_registro" e recebendo um valor inteiro do usuário que corresponde ao tipo de registro (1 para presença e 0 para falta). Em seguida, o programa verifica se o valor fornecido é maior ou igual a zero e menor ou igual a "max\_registro". Se for, ele usa a máscara de bits (mask) para definir o registro de presença/falta do aluno na aula correspondente, atualizando o vetor de dias (Vetor\_Dias), e em seguida retorna à função "loop\_define\_aula", possibilitando a re-execução do programa, mas mantendo os valores informados anteriomente. Caso contrário, exibe a string "texto\_valor\_invalido" e chama a função "loop\_define\_registro" novamente.

**Código do Risc-V:**

# Disciplina: Arquitetura e Organização de Computadores

# Atividade: Avaliação 01 – Programação em Linguagem de Montagem

# Programa 02

# Grupo: - Mateus Barbosa

# - Matheus de Oliveira Rocha

.data # Segmento de Dados

Vetor\_Dias: .word 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,

0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

mask: .word 0x00000001

texto\_input\_aula: .asciz "\nEntre com o número da aula (de 0 a 15): "

texto\_input\_aluno: .asciz "\nEntre com o número do aluno (de 0 a 31):"

texto\_input\_registro: .asciz "\nEntre com o tipo do registro (presença = 1; ausência = 0): "

texto\_valor\_invalido: .asciz "\nValor Inválido"

max\_aula: .word 15

max\_aluno: .word 31

max\_registro: .word 1

.text # Segmento de Codigo

main: # Função usada como ponto de partida do código, e retorno de loops

lw t0, mask # Carrega a mascara de bits no registrador t0

la s0, Vetor\_Dias # Carrega o Vetor\_dias no registrador s0

j loop\_define\_aula

loop\_define\_aula:

# Imprime: String texto\_input\_aula

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_aula # Carrega ao registrador a0 o texto\_input\_max\_tam

ecall # Chama a syscall

# Solicita: Int numero da aula

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s1, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s1

bge s1, zero, if\_aula\_maior\_que # Se o valor de s1 for maior ou igual que 0, vai para a função if\_maior\_que

j loop\_define\_aula # "Pula" para a função loop\_define\_aula

if\_aula\_maior\_que:

lw t1, max\_aula

ble s1, t1, loop\_define\_aluno # Se o valor de s1 for menor ou igual que max\_aluno, vai para a função loop\_define\_aluno

# Imprime: String valor\_invalido

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_valor\_invalido # Carrega ao registrador a0 o valor\_invalido

ecall # Chama a syscall

j loop\_define\_aula # "Pula" para a função loop\_define\_aluno

loop\_define\_aluno:

# Imprime: String texto\_input\_aluno

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_aluno # Carrega ao registrador a0 o texto\_input\_max\_tam

ecall # Chama a syscall

# Solicita: Int numero da aluno

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s2, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s2

bge s2, zero, if\_aluno\_maior\_que # Se o valor de s2 for maior ou igual que 0, vai para a função if\_aluno\_maior\_que

j loop\_define\_aluno # "Pula" para a função loop\_define\_aluno

if\_aluno\_maior\_que:

lw t1, max\_aluno

ble s2, t1, loop\_define\_registro # Se o valor de s2 for menor ou igual que max\_aluno, vai para a função loop\_define\_registro

# Imprime: String valor\_invalido

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_valor\_invalido # Carrega ao registrador a0 o valor\_invalido

ecall # Chama a syscall

j loop\_define\_aluno # "Pula" para a função loop\_define\_aluno

loop\_define\_registro:

# Imprime: String texto\_input\_aluno

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_input\_registro # Carrega ao registrador a0 o texto\_input\_registro

ecall # Chama a syscall

# Solicita: Int numero da aluno

addi a7, zero, 5 # Adiciona o valor 5 (ReadInt) ao registrador de serviço a7

ecall # Chama a syscall

add s3, zero, a0 # O que foi digitado no console (registrador a0), é salvo no registrador s3

bge s3, zero, if\_registro\_maior\_que # Se o valor de s3 for maior ou igual que 0, vai para a função if\_registro\_maior\_que

j loop\_define\_registro # "Pula" para a função loop\_define\_presenca

if\_registro\_maior\_que:

lw t1, max\_registro

ble s3, t1, presenca\_ou\_ausencia # Se o valor de s3 for menor ou igual que max\_registro, vai para a função presenca\_ou\_ausencia

# Imprime: String valor\_invalido

addi a7, zero, 4 # Adiciona o valor 4 (PrintString) ao registrador de serviço a7

la a0, texto\_valor\_invalido # Carrega ao registrador a0 o valor\_invalido

ecall # Chama a syscall

j loop\_define\_registro # "Pula" para a função loop\_define\_presenca

presenca\_ou\_ausencia:

sll t2, t0, s2 # Desloca para a mascara (t0) esquera de acordo com o numero do aluno (s2)

slli t3, s1, 2 # Move 2 bits para a esquerda de acordo com o dia: 4 \* dia

add t4, s0, t3 # Calcula a posicao no Vetor\_Dias desde o seu comeco: comeco\_do\_Vetor + (4 \* dia)

lw t5, 0(t4) # Armazena em t5 o valor dos bits do dia informado (t4) com offset de 0 bits

beq s3, zero, ausencia # Se for 0, esta ausente

beq s3, t1, presenca # Se for 1 (max\_registro), esta presente

j exit # Tratamento de error, nunca devera cair nesse jump, devido as validacoes anteriores

ausencia:

not t2, t2 #Inverte a mascara

and t6, t5, t2 # Altera o bit do aluno usando a mascara (t2) usando and

sw t6, 0(t4) # Armazena os bites com a alteracao da mascara na posicao do Vetor\_Dias

j main

presenca:

or t6, t5, t2 # Altera o bit do aluno usando a mascara (t2) usando or

sw t6, 0(t4) # Armazena os bites com a alteracao da mascara na posicao do Vetor\_Dias

j main

exit:

addi a7, zero, 10 # Usa a diretriz Exit (10)

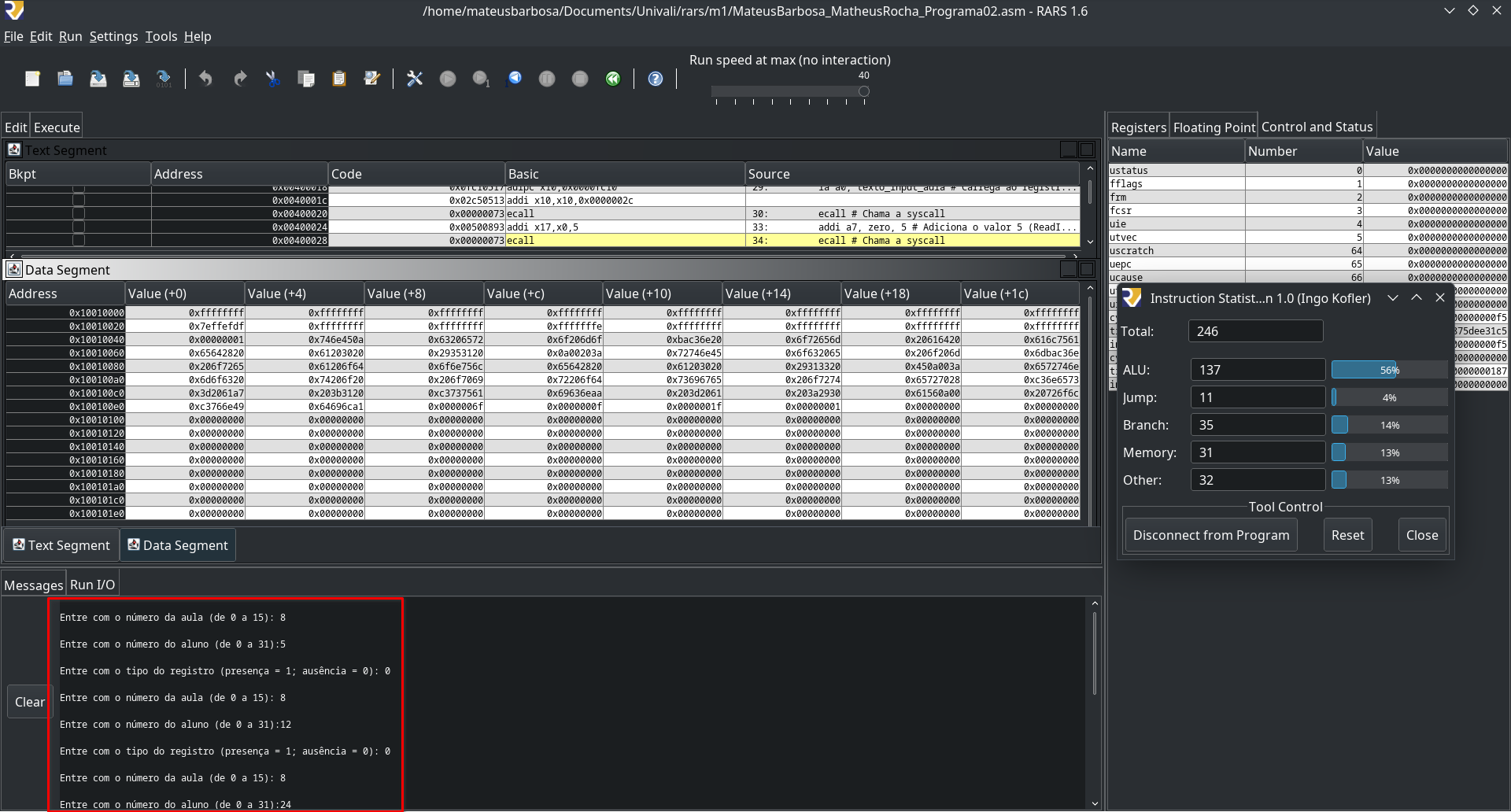
ecall

**2.4 resultados**

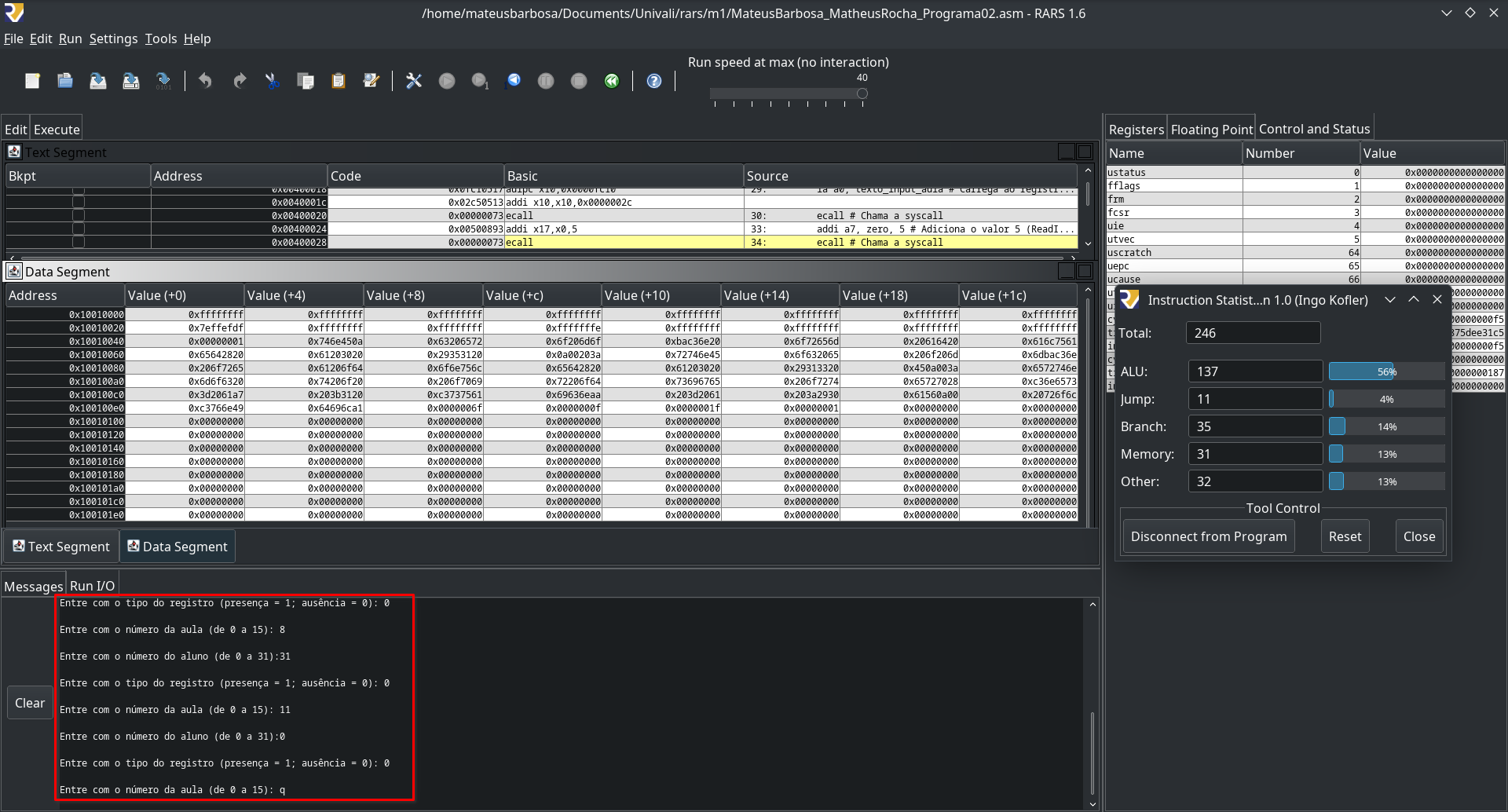
Informações da execução:

* Apenas após sair do programa que será possível visualizar as informações armazenadas no Vetor
* Na primeira execução, informamos que os seguintes alunos (5, 12, 24 e 31) não estão presentes no dia 8.
* Na segunda execução, informamos que no dia 1, o aluno 0 será inicialmente informado como ausente e depois retonará como presente. E o aluno 1 estará ausente.
* Para sair do programa, digite um valor diferente de um inteiro, uma letra por exemplo.

Inserção de informação do aluno 5, 12 e 24:



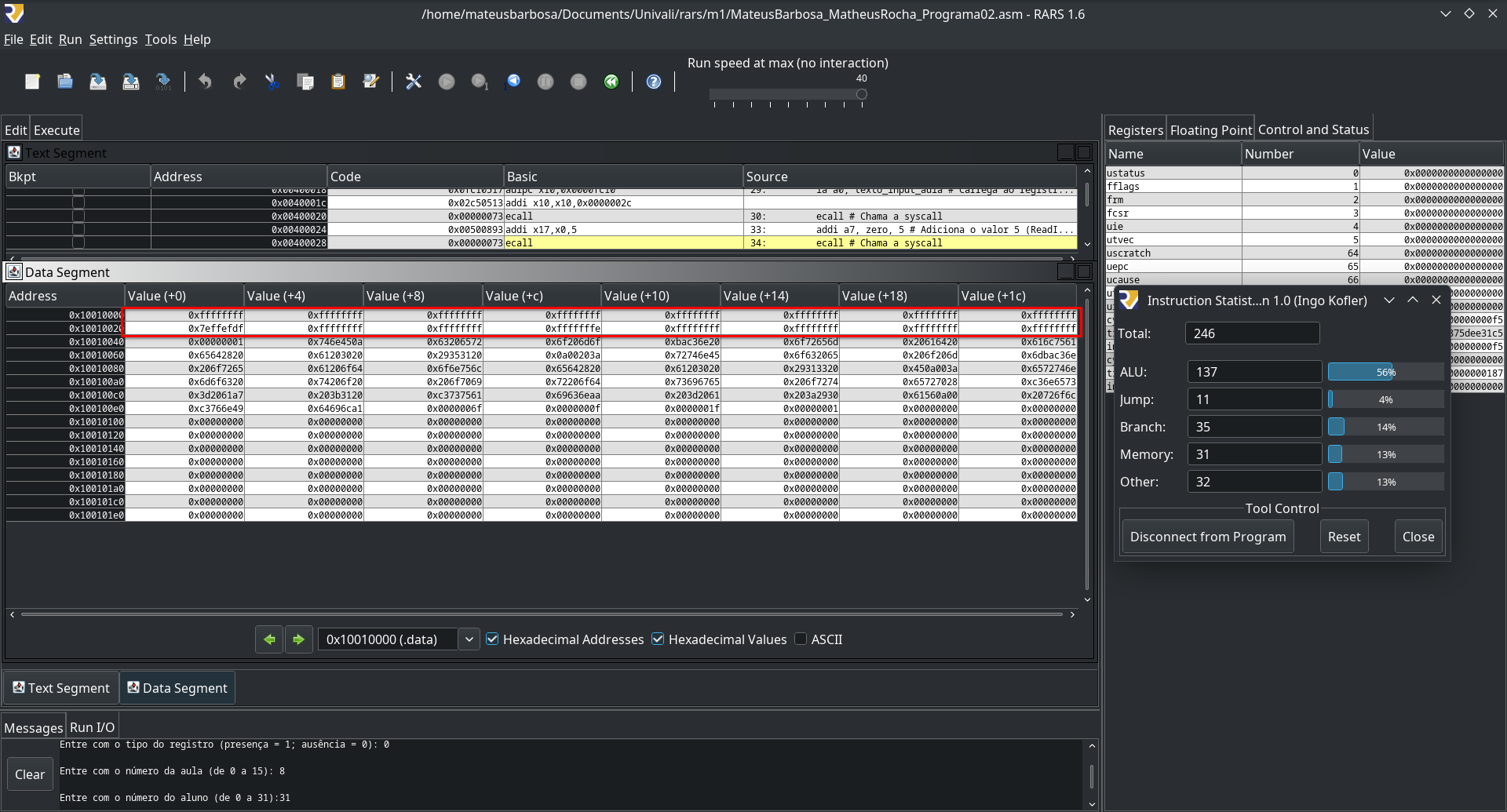
Inserção de informação do aluno 31:



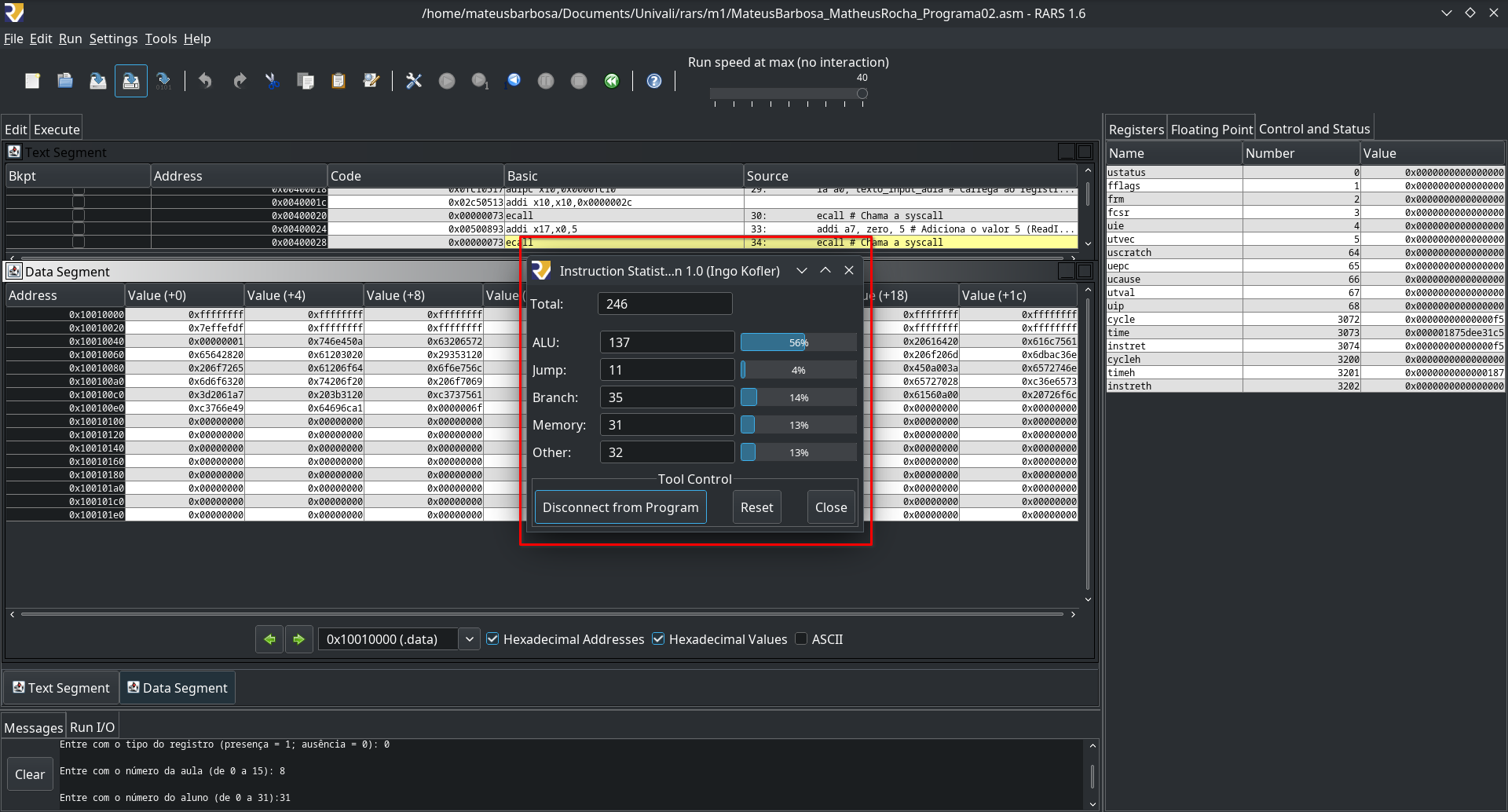
Valor final do Vetor\_Dias:

* Valor do Vetor\_Dias na posição 8:
  + Em Hexadecimal: 0x7effefdf
  + Em Binário: **0**111 111**0** 1111 1111 111**0** 1111 11**0**1 1111

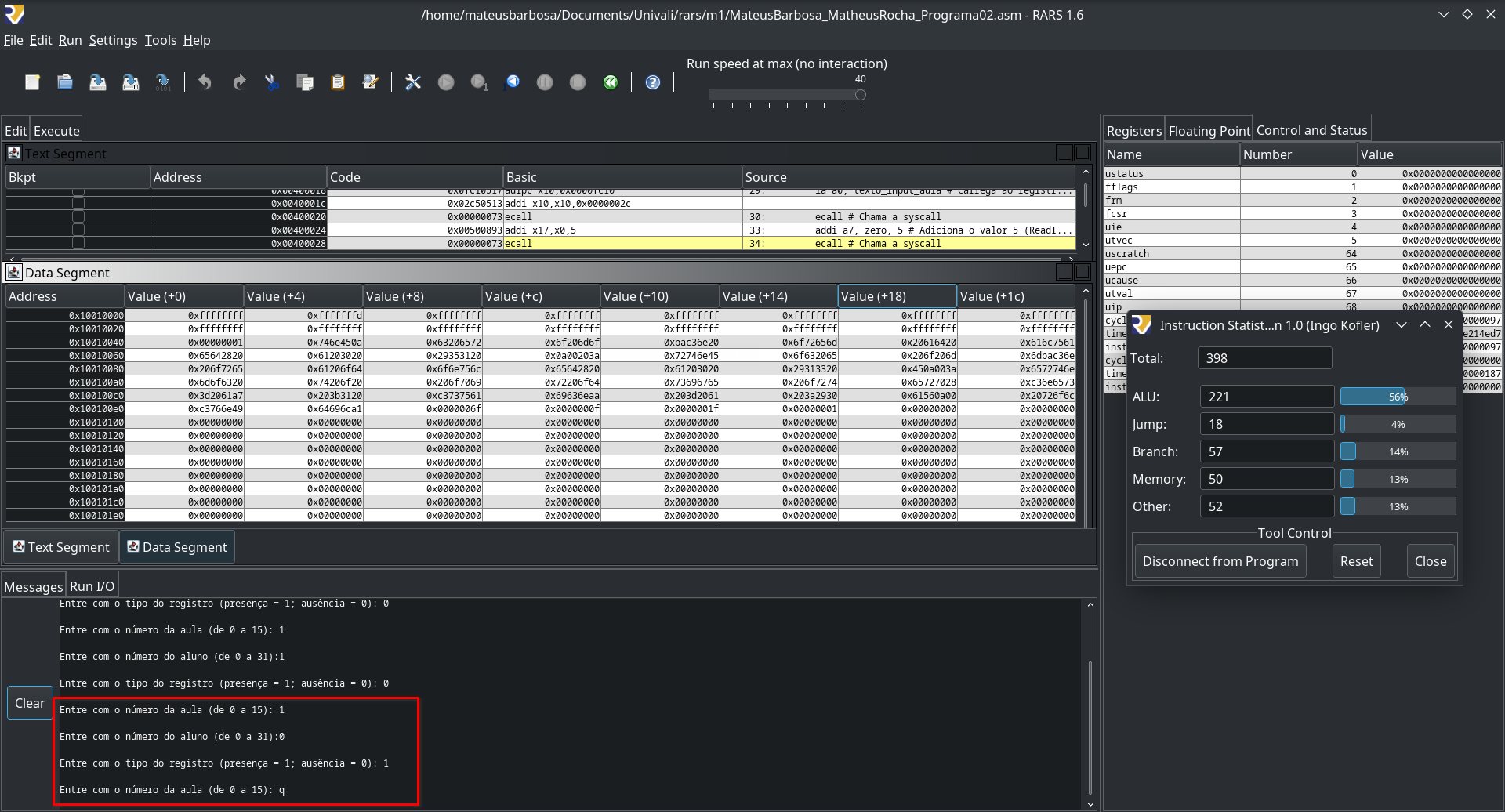
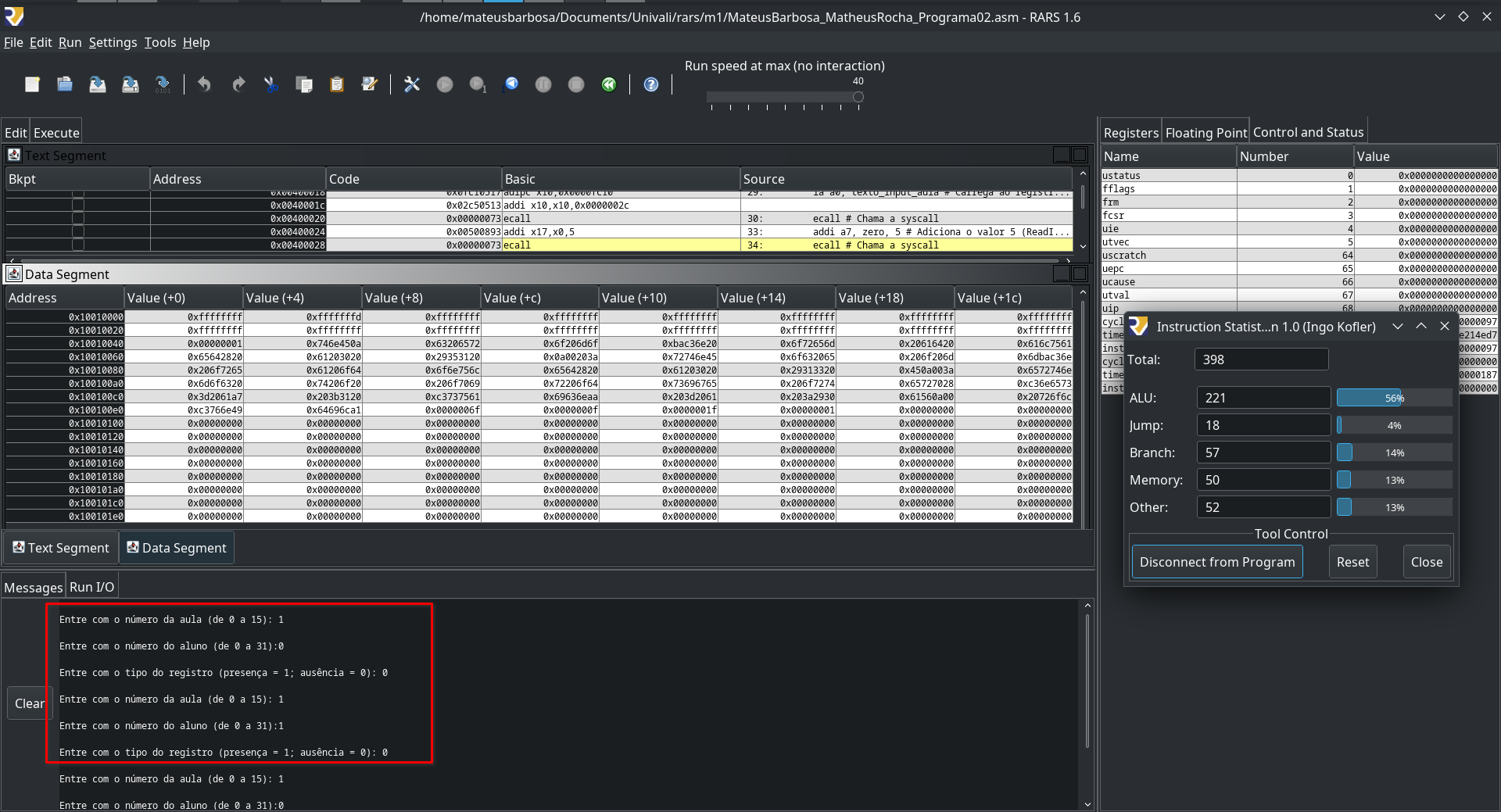
Como pode-se visualizar nos binários acima, os alunos de número 5, 12, 24 e 31 estão ausentes (com o valor 0).



Abaixo é a inclusão do Instruction Statistics no Programa e seus resultados de cada instrução realizada anteriormente:



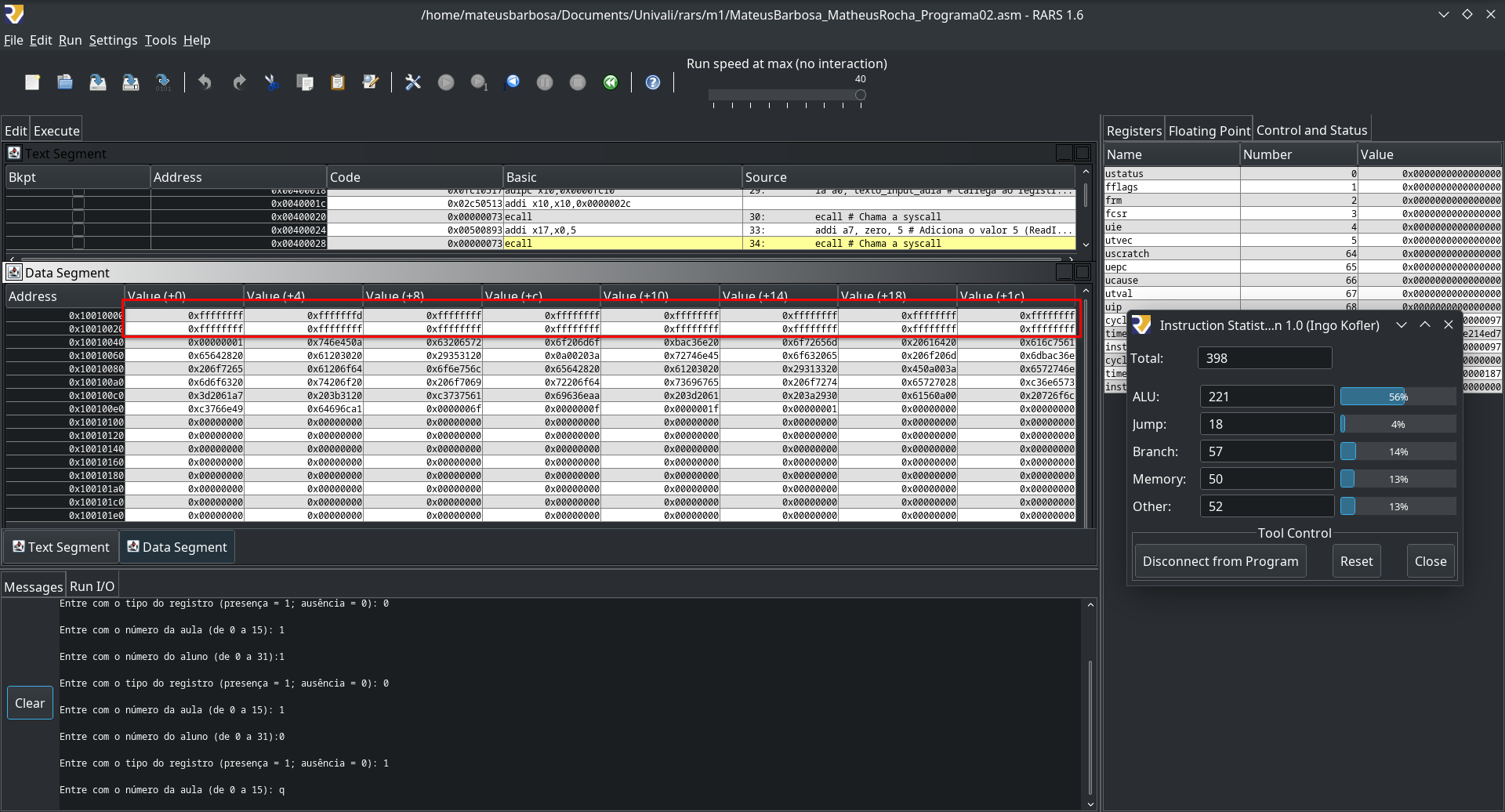
Abaixo será a execução do programa definindo no dia 1 que aluno de número 0 estava ausente e se tornou presente, e o aluno 1 se manterá ausente:



Valor final do Vetor\_Dias:

* Valor do Vetor\_Dias na posição 1:
  + Em Hexadecimal: 0xfffffffd
  + Em Binário: 11111111 11111111 11111111 11111101

E nesse segundo teste, podemos avaliar que o aluno número zero se tornou presente, e o aluno de número 1 continua ausente nesse dia.



Abaixo é a inclusão do Instruction Statistics no Programa e seus resultados de cada instrução realizada anteriormente:

