

RELATÓRIO LABORATÓRIO 1

ORGANIZAÇÃO DE COMPUTADORES I

Emanuelle Guse - 23100486

EXERCÍCIO 1

1. Iniciando o programa, todas as variáveis foram inicializadas a fim de melhor visualização na memória. Os endereços estão mostrados na imagem abaixo.

Text Segment					Labels	
Bkpt	Address	Code	Basic	Source	Label	Address
	4194304	0x3c011001	lui \$1,4097	16: la \$t0, va		
	4194308	0x3428000c	ori \$8,\$1,12		newline	268500992
	4194312	0x3c011001	lui \$1,4097	17: la \$t1, vb	a	268500994
	4194316	0x34290010	ori \$9,\$1,16		c	268500998
	4194320	0x3c011001	lui \$1,4097	18: la \$t2, vc	va	268501004
	4194324	0x342a0014	ori \$10,\$1,20		vb	268501008
	4194328	0x3c011001	lui \$1,4097	19: la \$t3, vd	vc	268501012
	4194332	0x342b0018	ori \$11,\$1,24		vd	268501016
	4194336	0x3c011001	lui \$1,4097	20: la \$t4, ve	ve	268501020
	4194340	0x342c001c	ori \$12,\$1,28			
	4194344	0x8d300000	lw \$s0, 0(\$t1) # b	22: lw \$s0, 0(\$t1) # b		
	4194348	0x8d710000	lw \$s1, 0(\$t3) # d	23: lw \$s1, 0(\$t3) # d		

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	694222858	694353952	3	0	4	0	2	1
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0

obs.: as variáveis, no programa, foram chamadas de **v{letra}**, mas, para as explicações, serão chamadas apenas de **{letra}**.

2. Apesar de já conhecermos os endereços, nosso programa ainda não conhece, por isso carregamos esses valores nos registradores temporários de \$t0 a \$t4.

\$t0	8	268501004
\$t1	9	268501008
\$t2	10	268501012
\$t3	11	268501016
\$t4	12	268501020

A imagem mostra que agora esses registradores possuem o endereço das variáveis, mas ainda não os seus valores.

3. Após isso, vamos carregar os valores de **b**, **d** e **e** em outros registradores, dessa vez de \$s0 a \$s2. Descobrir o endereço no passo anterior é

\$s0	16	4
\$s1	17	2
\$s2	18	1

fundamental para que a palavra seja carregada. Esses registradores agora possuem o valor que declaramos no início do programa.

4. Finalmente, as operações começam a ser feitas.

4.1) Em um registrador temporário \$t5, é armazenada a adição imediata (*addi*) do conteúdo de **b** com o imediato 35. Esse valor será utilizado no futuro.

\$t4	12	268501020
\$t5	13	39
\$t6	14	0

4.2) Em um outro registrador temporário \$t6, é feita a soma entre os valores de **d** e de **e**.

\$t6	14	3
------	----	---

temporary (not preserved across call)

4.3) Por fim, utilizando o mesmo \$t6, fazemos a operação $c = c - a$, já que **c**, naquele momento, contém o valor de **d** + **e**. Nesse caso, o valor de **c** segue armazenado em \$t6.

\$t6	14	-36
------	----	-----

5. Apesar de a operação estar completa, os valores ainda não foram armazenados na memória. Por isso, carregamos o valor de \$t6 para **c** e, a fim de visualização, carreguei \$t5 para **a** também.

Text Segment					Labels	
Bkpt	Address	Code	Basic	Source	Label	Address
	4194304	0x3c011001	lui \$1,4097	16: la \$t0, va	trabalho1.1.asm	
	4194308	0x3429000c	ori \$8,\$1,12		newline	268500992
	4194312	0x3c011001	lui \$1,4097	17: la \$t1, vb	a	268500994
	4194316	0x34290010	ori \$9,\$1,16		c	268500998
	4194320	0x3c011001	lui \$1,4097	18: la \$t2, vc	va	268501004
	4194324	0x342a0014	ori \$10,\$1,20		vb	268501008
	4194328	0x3c011001	lui \$1,4097	19: la \$t3, vd	vc	268501012
	4194332	0x342b0018	ori \$11,\$1,24		vd	268501016
	4194336	0x3c011001	lui \$1,4097	20: la \$t4, ve	ve	268501020
	4194340	0x342c001c	ori \$12,\$1,28			
	4194344	0x8d300000	lw \$16,0(\$9)	22: lw \$s0, 0(\$t1) # b		
	4194348	0x8d710000	lw \$17,0(\$11)	23: lw \$s1, 0(\$t3) # d		

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	694222858	694353952	32	39	4	-36	2	1
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0

6. Para melhor compreensão na execução, imprimir os resultados finais na tela. Os métodos para isso, porém, serão melhor detalhados na explicação do próximo exercício.

O código possui 62 linhas, considerando quebras, comentários e que pseudo-instruções são apenas uma. Ignorando comentários e transformando as pseudos-instruções em instruções, o exercício 1 conta com 38 linhas em .text e 8 linhas em .data.

EXERCÍCIO 2

1. Inicialmente, armazenamos textos no formato *.asciiz*, que serão utilizados no futuro para realizar impressões na tela.

```
a: .asciiz "a) "  
c: .asciiz "c) "  
  
newline: .asciiz "\n"  
ib: .asciiz "insira o valor de b: "
```

a	268500992
c	268500996
newline	268501000
ib	268501002

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
268500992	2107745	2107747	1852375050
268501024	0	0	0

2. Após isso, da mesma forma que no exercício 1, alocamos os valores de **a** a **e** na memória, dessa vez sem se importar com o valor de **b**.

Text Segment					Labels				
Bkpt	Address	Code	Basic	Source	Label	Address			
	4194304	0x3c011001	lui \$1,4097	18: la \$t0, va	trabalho1-2.asm				
	4194308	0x34280020	ori \$8,\$1,32		a	268500992			
	4194312	0x3c011001	lui \$1,4097	19: la \$t1, vb	c	268500996			
	4194316	0x34290024	ori \$9,\$1,36		newline	268501000			
	4194320	0x3c011001	lui \$1,4097	20: la \$t2, vc	ib	268501002			
	4194324	0x342a0028	ori \$10,\$1,40		va	268501024			
	4194328	0x3c011001	lui \$1,4097	21: la \$t3, vd	vb	268501028			
	4194332	0x342b002c	ori \$11,\$1,44		vc	268501032			
	4194336	0x3c011001	lui \$1,4097	22: la \$t4, ve	vd	268501036			
	4194340	0x342c0030	ori \$12,\$1,48		ve	268501040			
	4194344	0x24020004	addiu \$2,\$0,4	25: li \$v0, 4					
	4194348	0x3c011001	lui \$1,4097	26: la \$a0, ib					
					Data Segment				
	Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
	268500992	2107745	2107747	1852375050	1634888051	1981837088	1919904865	543515680	2112098
	268501024	0	0	0	12	31	0	0	0
	268501056	0	0	0	0	0	0	0	0
	268501088	0	0	0	0	0	0	0	0

3. Após isso, carregamos os valores desses endereços nos registradores temporários de **\$t0** a **\$t4**.

\$t0	8	268501024
\$t1	9	268501028
\$t2	10	268501032
\$t3	11	268501036
\$t4	12	268501040

4. Finalmente, iniciamos nossas interações com o usuário. Para isso, vamos imprimir uma mensagem na tela e receber um valor.

4.1) Para a mensagem pedindo o valor, utilizamos o texto declarado no começo do programa. É carregado o valor imediato 4 no registrador \$v0, indicando que vamos imprimir uma string. Após isso, é carregado o endereço na mensagem no registrador \$a0, que armazena esse endereço. Finalmente, chamamos o syscall e imprimimos essa string.

\$v0	2	4
\$v1	3	0
\$a0	4	268501002

\$v0 recebe 4 e \$a0 o endereço de *ib*

Mars Messages

Run I/O

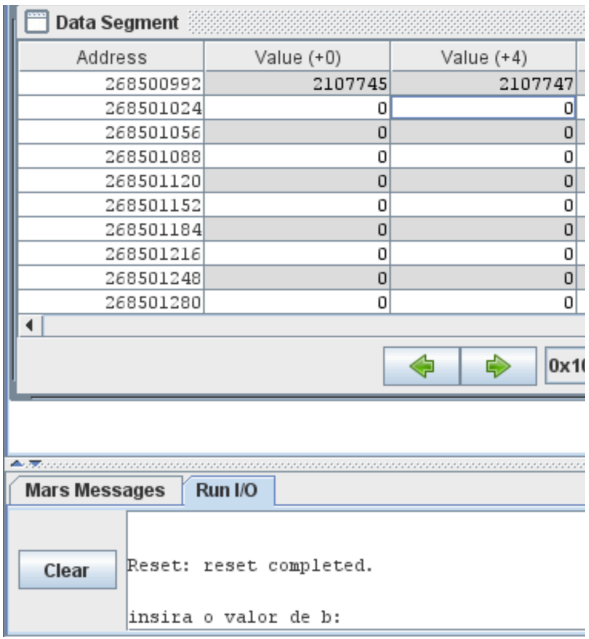
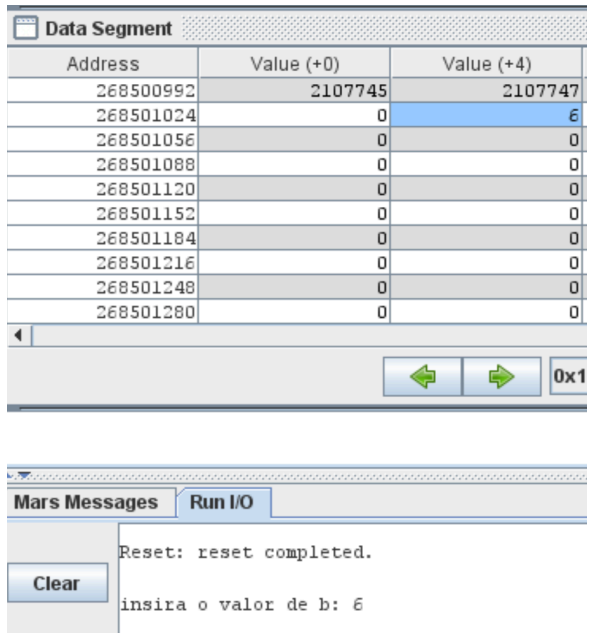
Clear

Reset: reset completed.

insira o valor de b:

4.2) Para receber o valor de **b**, começamos carregando o imediato 5 para o registrador \$v0, indicando a realização de uma leitura de inteiro. Com syscall sendo chamada, o valor inserido é automaticamente guardado em \$v0. Para finalizar, movemos do registrador \$v0 para o \$s0 e, só para garantir, armazenamos o valor de \$s0 utilizando o endereço armazenado em \$t1.

\$v0	2	6
\$s0	16	6

antes	depois
	

5. Após esse passo, carregamos os valores de d e de e nos registradores \$s1 e \$s2.

\$s1	17	12
\$s2	18	31

6. Fazemos as operações da mesma forma que foi feito no exercício passado.

6.1) Realização das operações exatamente da mesma forma.

\$t5	13	41
\$t6	14	43

Valores de \$t5 após a operação $b + 35$ e de \$t6 após $d + e$

\$t6	14	2
------	----	---

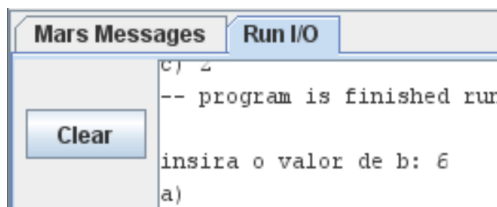
Valor de \$t6 após $t6 = t6 - t6$ ($c = c - a$)

6.2) Armazenamento dos valores de \$t5 e \$t6 em **a** e **c**.

Data Segment				
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)
268500992	2107745	2107747	1852375050	1634888051
268501024	41	6	2	12

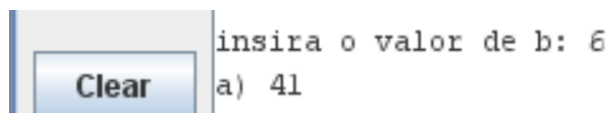
7. Apenas para comprovar, vamos carregar os novos valores de **a** e de **c** nos registradores temporários \$t5 e \$t6. Vamos imprimir esse valor na tela.

\$t3	11	41
\$t4	12	2

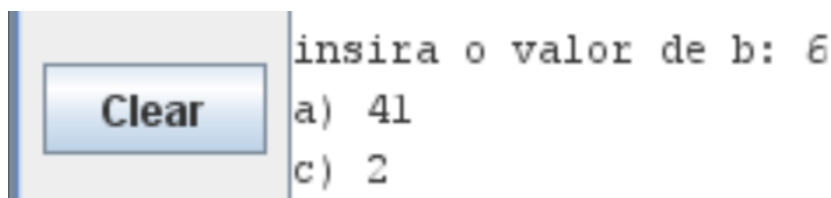


7.1) Carregando o imediato 4 para \$v0, indicamos que vamos imprimir uma string. Ela foi, também, definida no início do nosso programa.

7.2) Imprimimos o valor de **a**, dessa vez passando o imediato 1, para imprimir um inteiro.



7.3) Repetimos o mesmo passo, mas, passando os endereços referentes a **c**.



O código possui 74 linhas, considerando quebras, comentários e que pseudo-instruções são apenas uma. Ignorando comentários e transformando as pseudos-instruções em instruções, o exercício 2 conta com 45 linhas em .text e 9 linhas em .data.