Relatório - Laboratório 01
Disciplina: INE5411 – Organização de Computadores I
Integrantes: Lucas Pastre de Souza e Rodrigo Martins dos Santos.

Introdução:

O presente relatório busca mostrar o funcionamento dos exercícios 1 e 2, referentes à primeira atividade de laboratório da disciplina Organização e Arquitetura de Computadores do curso de Ciências da Computação na Universidade Federal de Santa Catarina (UFSC). Esses exercícios abordam conceitos de operações aritméticas em linguagem de montagem, leitura e escrita na memória, bem como chamadas de sistema para a entrada e saída de dados.

Exercício 1:

Inicialmente, a questão 1 da atividade propõe a realização dos seguintes cálculos:

$$a = b + 35$$
$$c = d^3 - (a+e)$$

Para isso, selecionamos alguns valores para as variáveis **b**, **d** e **e**, sendo respectivamente os valores **5**, **4** e **10**. Declaramos essas variáveis na memória de dados, conforme a imagem:

```
1  .data
2          #declarando variaveis do sistema
3          b:.word 5 # declarando b
4          c:.word 0 # declarando c
5          d:.word 4 # declarando d
6          e:.word 10 # declarando e
7
```

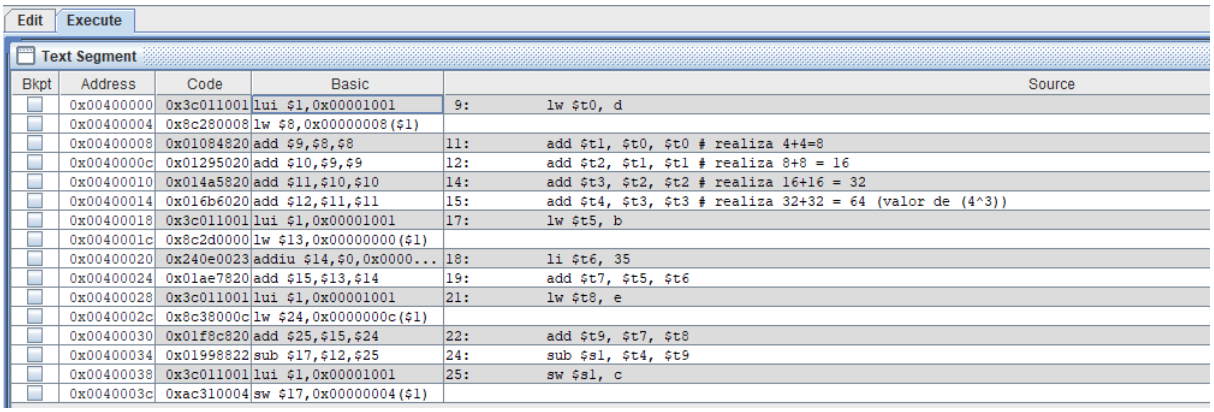
Posteriormente na seção de código, carregamos o valor da variável **d** e realizamos as operações de soma (add) necessárias para calcular o valor de **d^3**. Para isso foram utilizados 5 registradores ($t0, $t1, $t2, $t3, $t4). Isso é ilustrado na imagem abaixo:

```
8   .text
9           lw $t0, d
10
11          add $t1, $t0, $t0 # realiza 4+4=8
12          add $t2, $t1, $t1 # realiza 8+8 = 16
13
14          add $t3, $t2, $t2 # realiza 16+16 = 32
15          add $t4, $t3, $t3 # realiza 32+32 = 64 (valor de (4^3))
16
```

Após armazenar o valor de d³ no registrador $t4, carregamos o valor de **b** e também um imediato **35**, para realizar a operação **b + 35**. Então carregamos o valor de **e** e somamos ao valor de **a**, para calcular **a + e**. Após isso, realizamos uma operação de subtração (sub), para realizar **d³ - (a + e)** e calcular o valor final de **c**, que é armazenado na memória utilizando a operação de escrita sw (store word). Conforme a imagem abaixo:

```
17          lw $t5, b
18          li $t6, 35
19          add $t7, $t5, $t6
20
21          lw $t8, e
22          add $t9, $t7, $t8
23
24          sub $s1, $t4, $t9
25          sw $s1, c
26
```

O total de linhas na coluna Basic é 16, enquanto na coluna Source é de 12. Essa diferença ocorre por conta da instrução **lui**, que é gerada automaticamente e aparece na coluna Basic sempre que as instruções load word e store word são utilizadas para que os 16 bits mais significativos do endereço de memória sejam devidamente carregados e os 32 bits do endereço sejam montados corretamente. Veja a seguir a imagem da tabela de execução:



Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	9:	lw $t0, d
	0x00400004	0x8c280008	lw $8,0x00000008($1)		
	0x00400008	0x01084820	add $9,$8,$8	11:	add $t1, $t0, $t0 # realiza 4+4=8
	0x0040000c	0x01295020	add $10,$9,$9	12:	add $t2, $t1, $t1 # realiza 8+8 = 16
	0x00400010	0x014a5820	add $11,$10,$10	14:	add $t3, $t2, $t2 # realiza 16+16 = 32
	0x00400014	0x016b6020	add $12,$11,$11	15:	add $t4, $t3, $t3 # realiza 32+32 = 64 (valor de (4^3))
	0x00400018	0x3c011001	lui $1,0x00001001	17:	lw $t5, b
	0x0040001c	0x8c2d0000	lw $13,0x00000000($1)		
	0x00400020	0x240e0023	addiu $14,$0,0x0000...	18:	li $t6, 35
	0x00400024	0x01ae7820	add $15,$13,$14	19:	add $t7, $t5, $t6
	0x00400028	0x3c011001	lui $1,0x00001001	21:	lw $t8, e
	0x0040002c	0x8c38000c	lw $24,0x0000000c($1)		
	0x00400030	0x01f8c820	add $25,$15,$24	22:	add $t9, $t7, $t8
	0x00400034	0x01998822	sub $17,$12,$25	24:	sub $s1, $t4, $t9
	0x00400038	0x3c011001	lui $1,0x00001001	25:	sw $s1, c
	0x0040003c	0xac310004	sw $17,0x00000004($1)		

Exercício 2:

Na questão 2, declaramos os valores das variáveis **b, d** e **e** todos como 0, para que possam ser inseridos pelo usuário, em seguida inicializamos as mensagens a serem impressas no terminal. Conforme o código abaixo:

```
.data
# Declarando variáveis do sistema

    b: .word 0          # Declarando a variável 'b'
    c: .word 0          # Declarando a variável 'c'
    d: .word 0          # Declarando a variável 'd'
    e: .word 0          # Declarando a variável 'e'

    msg: .asciiz "C = " # Mensagem que será exibida antes do valor de 'c'
    input_b: .asciiz "Valor de b: "  # Mensagem solicitando o valor de 'b'
    input_d: .asciiz "Valor de d: "  # Mensagem solicitando o valor de 'd'
    input_e: .asciiz "Valor de e: "  # Mensagem solicitando o valor de 'e'
.text
```

Logo após isto na seção .text, começamos solicitando os valor de **b**, **d** e **e** ao usuário, em seguida carregamos o valor de **d** para o registrador $t0 e dois registradores temporários ($t1 e $t2) para armazenar os valores de **d²** e **d³** respectivamente, além de dois registradores temporários para ($t3 e $t4) para calcular os valores de **d²** e **d³**. Conforme a imagem abaixo:

```

    # Solicitar valor de 'b' ao usuário
    li $v0, 4               # Carrega a chamada de sistema para impressão de string
    la $a0, input_b         # Carrega o endereço da string "Valor de b: " no registrador $a0
    syscall                 # Realiza o print da string

    li $v0, 5               # Carrega a chamada de sistema para ler um inteiro
    syscall                 # Lê o valor do usuário e coloca no registrador $v0

    sw $v0, b               # Armazena o valor lido na variável 'b'

    # Solicitar valor de 'd' ao usuário
    li $v0, 4               # Imprimir a string "Valor de d: "
    la $a0, input_d         # Carrega o endereço da string
    syscall                 # Realiza o print da string

    li $v0, 5               # Chama o sistema para ler um inteiro
    syscall                 # Lê o valor do usuário

    sw $v0, d               # Armazena o valor lido na variável 'd'

    # Solicitar valor de 'e' ao usuário
    li $v0, 4               # Imprimir a string "Valor de e: "
    la $a0, input_e         # Carrega o endereço da string
    syscall                 # Realiza o print da string

    li $v0, 5               # Chama o sistema para ler um inteiro
    syscall                 # Lê o valor do usuário

    sw $v0, e               # Armazena o valor lido na variável 'e'

    # Começar os cálculos com d
    lw $t0, d               # Carrega o valor de 'd' para o registrador $t0
    li $t1, 0               # Inicializa $t1 com 0 (usado para armazenar d^2)
    li $t2, 0               # Inicializa $t2 com 0 (usado para armazenar d^3)

    move $t3, $t0           # Copia 'd' para $t3, contador usado para calcular d^2
    move $t4, $t0           # Copia 'd' para $t4, contador usado para calcular d^3
```

Na sequência, inicializamos um loop para calcular o valor de **d²** e outro loop para calcular o valor de **d³**, além de um label de saída. O loop que calcula o valor de **d²** compara o valor armazenado em $t3 com 0, se os valores forem iguais o loop pula para o segundo loop, se não o loop soma o valor de $t1 (d) com ele mesmo e subtrai 1 de $t3, até que $t3 seja igual a zero. Já o segundo loop calcula o valor de **d³** e segue a mesma lógica que o primeiro loop, comparando o valor de $t4 com 0, se forem iguais o código vai para o label de saída e executa o resto das somas, se não o loop soma o valor de $t2 (**d²**) com ele mesmo e subtrai 1 de $t4, até que $t4 seja igual a zero. Conforme o código abaixo:

```
    # Loop para calcular o quadrado de 'd' (d^2)
    quadrado:
        beq $t3, $zero, cubo  # Se $t3 for 0, pula para o cálculo do cubo
        add $t1, $t1, $t0     # Soma o valor de 'd' no acumulador $t1 (para d^2)
        sub $t3, $t3, 1       # Decrementa o contador $t3
        j quadrado            # Repete o loop até $t3 ser 0

    # Loop para calcular o cubo de 'd' (d^3)
    cubo:
        beq $t4, $zero, end   # Se $t4 for 0, pula para o fim dos cálculos
        add $t2, $t2, $t1     # Soma o valor de d^2 no acumulador $t2 (para d^3)
        sub $t4, $t4, 1       # Decrementa o contador $t4
        j cubo                # Repete o loop até $t4 ser 0

    # Fim dos cálculos
    end:
        lw $t5, b             # Carrega o valor de 'b' no registrador $t5
        li $t6, 35            # Carrega o valor 35 no registrador $t6
        add $t7, $t5, $t6     # Soma o valor de 'b' com 35 e armazena em $t7
        lw $t8, e             # Carrega o valor de 'e' no registrador $t8
        add $t9, $t7, $t8     # Soma o valor anterior (b + 35) com 'e' e armazena em $t9
        sub $s1, $t2, $t9     # Subtrai o valor de (b + 35 + e) de d^3 e armazena em $s1

        sw $s1, c       # Escrevendo o valor da variável c no registrador $s1

        # Imprimir a mensagem "C = "
        li $v0, 4             # Carrega a chamada de sistema para imprimir string
        la $a0, msg           # Carrega o endereço da string "C = "
        syscall               # Realiza o print da string

        # Imprimir o valor de 'c'
        li $v0, 1             # Carrega a chamada de sistema para imprimir inteiro
        move $a0, $s1         # Move o valor de 'c' para o registrador $a0
        syscall               # Realiza o print do valor de 'c'

        # Finalizar o programa
        li $v0, 10            # Carrega a chamada de sistema para encerrar o programa
        syscall               # Encerra o programa
```

Segue abaixo exemplo de como o resultado está sendo impresso no terminal do Mars com os seguintes inputs do usuário:

Inputs:
Valor de b: 5
Valor de d: 4
Valor de e: 10

Output:
C = 14

Perceba que o resultado do exercício 2 coincide com o resultado do exercício 1 quando os mesmos valores são inseridos.

O total de linhas na coluna Basic é 59, enquanto na coluna Source é de 46. Assim como dito no exercício 1, isto ocorre pois a coluna Basic representa as instruções de máquina que de fato o processador executa, enquanto a coluna Source é o código feito pelo programador, isto faz com que a coluna Basic possua algumas instruções a mais que o código-fonte, pois algumas instruções podem ser desdobradas em múltiplas instruções de máquina. Segue imagem de um trecho da tabela para exemplo:

Basic		Source	
addiu $2,$0,4	16:	li $v0, 4	# Carrega a chamada de sistema para impressão de string
lui $1,4097	17:	la $a0, input_b	# Carrega o endereço da string "Valor de b: " no registrador $a0
ori $4,$1,21			
syscall	18:	syscall	# Realiza o print da string
addiu $2,$0,5	20:	li $v0, 5	# Carrega a chamada de sistema para ler um inteiro
syscall	21:	syscall	# Lê o valor do usuário e coloca no registrador $v0
lui $1,4097	23:	sw $v0, b	# Armazena o valor lido na variável 'b'
sw $2,0($1)			
addiu $2,$0,4	26:	li $v0, 4	# Imprimir a string "Valor de d: "
lui $1,4097	27:	la $a0, input_d	# Carrega o endereço da string
ori $4,$1,34			
syscall	28:	syscall	# Realiza o print da string
addiu $2,$0,5	30:	li $v0, 5	# Chama o sistema para ler um inteiro
syscall	31:	syscall	# Lê o valor do usuário
lui $1,4097	33:	sw $v0, d	# Armazena o valor lido na variável 'd'
sw $2,8($1)			
addiu $2,$0,4	36:	li $v0, 4	# Imprimir a string "Valor de e: "
lui $1,4097	37:	la $a0, input_e	# Carrega o endereço da string
ori $4,$1,47			
syscall	38:	syscall	# Realiza o print da string
addiu $2,$0,5	40:	li $v0, 5	# Chama o sistema para ler um inteiro
syscall	41:	syscall	# Lê o valor do usuário
lui $1,4097	43:	sw $v0, e	# Armazena o valor lido na variável 'e'
sw $2,12($1)			
lui $1,4097	46:	lw $t0, d	# Carrega o valor de 'd' para o registrador $t0
lw $8,8($1)			
addiu $9,$0,0	47:	li $t1, 0	# Inicializa $t1 com 0 (usado para armazenar d^2)
addiu $10,$0,0	48:	li $t2, 0	# Inicializa $t2 com 0 (usado para armazenar d^3)
addu $11,$0,$8	50:	move $t3, $t0	# Copia 'd' para $t3, contador usado para calcular d^2
addu $12,$0,$8	51:	move $t4, $t0	# Copia 'd' para $t4, contador usado para calcular d^3
beq $11,$0,4	55:	beq $t3, $zero, cubo	# Se $t3 for 0, pula para o cálculo do cubo