Relatório do Laboratório 01

Alunos: Vinicius Henrique Ribeiro (23200351) e Lucas Furlanetto Pascoali (23204339)
Professor: Marcelo Daniel Berejuck
Disciplina: Organização de Computadores

Questão 1

A primeira questão consistia em criar um algoritmo que realizasse os seguintes cálculos:

$$a = b + 35$$

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

Para resolver essa questão, é necessário ter os valores já inicializados em memória na seção .data, conforme mostrado na figura abaixo:

```
3   .data
4           b_: .word 1
5           d_: .word 2
6           e_: .word 3
7           c_: .word 4
```

Na imagem, temos os locais na memória declarados, sendo *b_*, *d_* e *e_* valores alocados no próprio código, ficando a critério do aluno a escolha dos números. Já o valor *c_* será o local onde guardaremos o resultado das operações.

O código em linguagem de montagem foi feito da seguinte maneira:

```
 9  .text
10          lw      $s0, b_
11          lw      $s1, d_
12          lw      $s2, e_
13
14          addi    $t0, $s0, 35    # a = b + 35
15          add     $t1, $t0, $s2   # a + e
16          mul     $t2, $s1, $s1   # d²
17          mul     $t2, $s1, $t2   # d³
18          sub     $t3, $t2, $t1   # t2 = d³ - (a + e)
19          sw      $t3, c_         # c = t2
```

Os valores de *b*, *d* e *e* são carregados da memória para os registradores *$s0*, *$s1* e *$s2*, respectivamente. Na linha 14 é feito o primeiro cálculo com o valor de *b*, sendo guardado em *$t0*. Logo em seguida, na linha 15, esse valor é utilizado para

efetuar a soma com a variável *e*, salvando a operação em *$t1*. Utilizamos a instrução *mul* para elevar *d* ao cubo nas linhas 16 e 17. Por fim, salvamos em *$t3* e escrevemos na memória com a instrução *sw*.

O código descrito acima tem, em *.text*, um total de 9 linhas. Porém, quando vamos à aba do *Basic* em *Execute*, vemos que há 13 linhas. Isso se dá pois as instruções *lw* e *sw* são pseudo instruções, sendo necessárias mais de uma linha "real" de código para a sua completa execução.

Questão 2

A questão 2 solicita que a questão 1 fosse alterada, agora, os valores das incógnitas b, d e e deveriam ser inseridos pelo usuário, por esse motivo, foram realizadas as seguintes alterações na seção .data, segue a imagem abaixo:

```
3  .data
4          c_:  .word 25
5          msg_b:  .asciiz "Entre com o valor de b: "
6          msg_d:  .asciiz "Entre com o valor de d: "
7          msg_e:  .asciiz "Entre com o valor de e: "
```

Podemos ver que comparado à questão 1, persistiu apenas o valor c_, que é o local onde guardaremos o resultado das operações. Já msg_b, msg_d e msg_e apenas armazenam uma mensagem que aparecerá na tela do usuário, apenas para fins estéticos. Já o código em linguagem de montagem ficou da seguinte forma:

```
9  .text
10         la      $a0, msg_b
11         li      $v0, 4
12         syscall                 # printa string para entrada de b
13         li      $v0, 5
14         syscall
15         add     $s0, $zero, $v0 # lê b
16
17         la      $a0, msg_d
18         li      $v0, 4
19         syscall                 # printa string para entrada de d
20         li      $v0, 5
21         syscall
22         add     $s1, $zero, $v0 # lê d
23
24         la      $a0, msg_e
25         li      $v0, 4
26         syscall                 # printa string para entrada de e
27         li      $v0, 5
28         syscall
29         add     $s2, $zero, $v0 # lê e
30
31         addi    $t0, $s0, 35    # a = b + 35
32         add     $t1, $t0, $s2   # a + e
33         mul     $t2, $s1, $s1   # d²
34         mul     $t2, $s1, $t2   # d³
35         sub     $t3, $t2, $t1   # t2 = d³ - (a + e)
36         sw      $t3, c_         # guarda o valor de t3 em c
37
38         li      $v0, 1
39         add     $a0, $zero, $t3
40         syscall                 # printa o valor de t3 (c) no terminal
```

Como precisamos que o usuário fornece os valores de b, d e e, realizamos o mesmo processo três vezes, no qual carregamos a respectiva mensagem (msg_b, msg_d ou msg_e) para o registrador $a0, após isso carregamos o valor 4 para o registrador $v0 e realizamos uma chamada de sistema, este processo imprime a mensagem carregada em $a0 para a tela do usuário.

Após isso, carregamos o valor 5 para o registrador $v0 e realizamos novamente uma chamada de sistema. O valor 5 é responsável por realizar a operação de ler um inteiro, então o inteiro que o usuário fornecer ficará salvo no próprio registrador $v0, por este motivo, realizamos a instrução *add* com um novo registrador como destino e o registrador $zero e o $v0 como operandos. Isso realiza o processo de mover o valor armazenado em $v0 para o registrador de destino passado para *add*.

Finalmente, após realizar esse processo três vezes, chegamos no cálculo, que é exatamente o mesmo descrito na questão 1, a única diferença é que após o cálculo, carregamos o valor 1 para o registrador $v0, ou seja, pretendemos imprimir um inteiro na tela, para isso, temos que colocar este inteiro no registrador de $a0 e realizar uma chamada de sistema. Nesse caso, queremos imprimir o resultado final, que está armazenado em $t3, logo realizamos a instrução *add* com $a0 como destino e os registradores $zero e $t3 como operandos, por fim realizamos a chamada de sistema e o resultado é impresso na tela.

O código da questão 2 tem, em .text, um total de 27 linhas. Porém, quando vamos à aba do *Basic* em *Execute*, vemos que há 31 linhas. Novamente, isso se dá devido à pseudo instruções, que neste caso são *la* e *sw*, que requisitam uma linha extra de código (em cada vez que aparecem).