Relatório do Laboratório 01 - Organização de Computadores

Grupo:

- Henrique Mateus Teodoro - 23100472
- Rodrigo Schwartz - 23100471

Exercício 1)

O exercício 1 consiste em realizar uma sequência de operações aritméticas básicas envolvendo as variáveis a, b, c, d, e. As operações são as seguintes:

➔ **a = b + 35;**
➔ **c = d – a + e;**

Inicialmente, como foi pedido para que todas as variáveis fossem armazenadas na memória de dados, com valores aleatórios, as variáveis são instanciadas e armazenadas na memória.

```
.data #dados que serão utilizados no programa
        a: .word 0
        b: .word 20
        c: .word 0
        d: .word 75
        e: .word 15
```

A memória, após a declaração dos dados em .data, ficou com os valores salvos, como mostra a imagem a seguir:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	0	20	0	75	15	0
0x10010020	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0

Em seguida, no seguimento .text, inicia-se o programa. A instrução “la” é utilizada para carregar o endereço de memória das variáveis criadas, como se fossem ponteiros da linguagem C. Assim, os endereços de a, b, c, d, e são carregados para os registradores $s0, $s1, $s2, $s3, $s4, respectivamente. Isso é feito no seguinte trecho de código:

```
.text #área de instruções que serão utilizados no programa
        la $s0, a               #carrega o endereço de a no reg $s0
        la $s1, b               #carrega o endereço de b no reg $s1
        la $s2, c               #salva o end de c em $s2
        la $s3, d               #salva o end de d em $s3
        la $s4, e               #salva o end de e em $s4
```

A imagem a seguir mostra a representação dos endereços, salvos nos registradores descritos após a execução das instruções:

Name	Number	Value
$s0	16	0x10010000
$s1	17	0x10010004
$s2	18	0x10010008
$s3	19	0x1001000c
$s4	20	0x10010010

Depois de carregar os endereços, uma série de instruções devem ser realizadas para continuar com os cálculos. O trecho de código da próxima imagem é responsável por fazer a operação a = b + 35.

- Primeiramente, faz-se necessário carregar o conteúdo de b, que está armazenado na memória. Para isso a instrução “lw” é utilizada, carregando uma palavra de memória na memória principal para um registrador. Para resgatar o valor de b, precisa-se do endereço de memória dele, que está salvo no registrador $s1 (registrador-base para o cálculo do endereço). Assim, o valor de b é trazido para o registrador $t0;
- Utilizando a instrução “addi”, o valor de b (armazenado em $t0) é somado com 35 (valor imediato) e armazenado no registrador $t1;
- Com base no registrador $s0, que contém o endereço de a, utiliza-se a instrução “sw” para escrever o conteúdo de $t1 na memória (escreve o resultado da soma em a, na memória de dados).

```
lw $t0, 0($s1)          #carrega o conteúdo de b em $t0
addi $t1, $t0, 35       #adiciona em $t1 o resultado de $t0 + 35
sw $t1, 0($s0)          #salva o resultado da operação em a
```

Registradores $t0 e $t1 após a execução:

Name	Number	Value
$t0	8	20
$t1	9	55

Memória após a execução (a recebe o valor da soma):

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	55	20	0	75	15

O próximo trecho de código é responsável por ler da memória os valores de d, e (seus endereços estão salvos nos registradores $s3 e $s4, respectivamente), e salvar em $t2 e $t3, utilizando a instrução “lw”:

```
lw $t2, 0($s3)          #carrega o conteúdo de d em $t2
lw $t3, 0($s4)          #carrega o conteúdo de e em $t3
```

Por fim, o código mostrado a seguir é responsável por efetuar a última operação: c = d – a + e.

- Primeiro, é efetuada a operação d - a, com os registradores temporários que armazenam seus valores, utilizando a instrução “sub”. O resultado é armazenado em $t4;
- Na próxima linha, utilizando “add”, é feita a operação (d - a) + e ($t3 contém o valor de e, e $t4 do cálculo da linha anterior). O resultado é armazenado ainda em $t4;
- Para finalizar, utiliza-se a instrução “sw” para escrever na memória o valor da conta anterior (armazenado em $t4) em c, utilizando como registrador-base $s4 (que contém o endereço de c).

```
sub $t4, $t2, $t1       #faz a sub de d - a e salva em $t4
add $t4, $t4, $t3       #faz a soma de (d-a) + e e salva em $t4
sw $t4, 0($s2)          #salva o resultado da operação em c
```

Os registradores após a execução do código permanecem com os seguintes valores (resultado da operação fica $t4):

$t0	8	20
$t1	9	55
$t2	10	75
$t3	11	15
$t4	12	35

Já a memória permanece com os seguintes valores (o resultado da operação é armazenado em c):

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	55	20	35	75	15

Em resumo:

a = b + 35, com b valendo 20
Logo a = 55.

c = d - a + e, com d valendo 75, a valendo 55, e e valendo 15
Logo c = 35.

Portanto, é possível verificar o funcionamento correto do programa. O programa do exercício 1 apresenta 18 linhas de assembly (sem pseudoinstruções), que são:

Basic
lui $1,4097
ori $16,$1,0
lui $1,4097
ori $17,$1,4
lui $1,4097
ori $18,$1,8
lui $1,4097
ori $19,$1,12
lui $1,4097
ori $20,$1,16
lw $8,0($17)
addi $9,$8,35
sw $9,0($16)
lw $10,0($19)
lw $11,0($20)
sub $12,$10,$9
add $12,$12,$11
sw $12,0($18)

Exercício 2)

O exercício 2 consiste em realizar as mesmas operações aritméticas realizadas no exercício 1. Entretanto, o valor b é variável, e deve ser fornecido pelo usuário (via teclado).

Inicialmente, são instanciadas na memória as variáveis que serão utilizadas para computar as operações aritméticas, assim como no exercício 1. A grande diferença agora, é que são armazenadas também duas variáveis do tipo string (array de chars), onde uma será utilizada para sinalizar ao usuário que ele deve inserir o valor de b desejado, e a outra mostrará o resultado final das operações realizadas.

```
data #dados que serão utilizados no programa
        a: .word 0
        b: .word 0
        c: .word 0
        d: .word 75
        e: .word 15
        string: .asciiz "Digite o valor de b: "
        string_result: .asciiz "O resultado de c é: "
```

Após a instanciação das variáveis, é possível confirmar que elas foram iniciadas corretamente na memória:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x0000004b	0x0000000f	0x69676944	0x6f206574	0x6c617620
0x10010020	0x6420726f	0x3a622065	0x204f0020	0x75736572	0x6461746c	0x6564206f	0xe9206320	0x0000203a
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Após isso, na área de instruções do programa, são carregados os endereços das variáveis a, b, c, d, e e nos registradores $s0, $s1, $s2, $s3, $s4, respectivamente.

```
.text #área de instruções que serão utilizados no programa
        la $s0, a               # carrega o endereço de a no reg $s0
        la $s1, b               # carrega o endereço de b no reg $s1
        la $s2, c               # salva o end de c em $s2
        la $s3, d               # salva o end de d em $s3
        la $s4, e               # salva o end de e em $s4
```

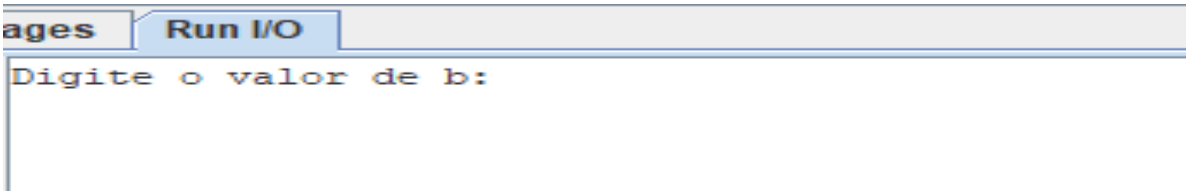
Estado dos registradores após a execução da instruções de carregamento de endereço (load address “la”):

$s0	16	0x10010000
$s1	17	0x10010004
$s2	18	0x10010008
$s3	19	0x1001000c
$s4	20	0x10010010

O próximo passo consiste em realizar a impressão da string “Digite o valor de b: “ no console. Essa etapa é realizada através do seguinte trecho de instruções:

```
li $v0, 4              # seta a operação de impressão de string
la $a0, string         # passa o end da string (ponteiro para string) para $a0
syscall                # faz a impressão da string
```

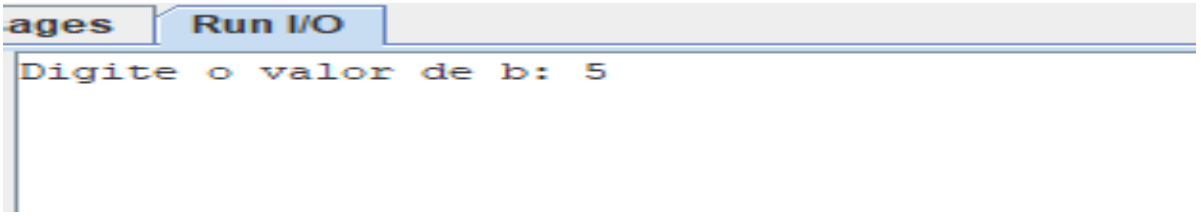
Após a execução dessa sequência de instruções,é feita a impressão da string no console:



Deseja-se então, que o usuário passe um valor escolhido para a variável b através do console. Para isso, o seguinte trecho de código é executado:

```
li $v0, 5              # seta a operação de leitura de int via console
syscall                # faz a leitura via console
move $t0, $v0          # passa o conteúdo lido armazenado em $v0 para $t0
sw $t0, 0($s1)         # pasas o conteúdo de $t0 para ser armazenado em b
```

Após esse conjunto de instruções, é pedido ao usuário o input:



Após o usuário dar a o input no console, o valor passado é armazenado em $v0 e transferido usando a instrução move para o reg $t0:

$v0	2	5
$v1	3	0
$a0	4	268501012
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	5

O conteúdo de $t0 (valor de b) é então armazenado no endereço da variável b, utilizando a instrução sw:

Address	Value (+0)	Value (+4)
0x10010000	0	5
0x10010020	1679848047	979509349
0x10010040	0	0
0x10010060	0	0

É então feita a operação **a = b + 35**, somando o resultado do valor passado para b (que está no reg $t1) com o imediato 35, e salva-se o valor na variável a:

```
addi $t1, $t0, 35        # adiciona em $t1 o resultado de b + 35
sw $t1, 0($s0)           # salva o resultado da operação em a
```

Resultado desse conjunto de instruções:

Address	Value (+0)	Value (+4)
0x10010000	40	5
0x10010020	1679848047	979509349
0x10010040	0	0

É então feito o carregamento do conteúdo das variáveis d e e, nos registradores $t2 e $t3, respectivamente, para realizar as próximas operações aritméticas:

```
lw $t2, 0($s3)           # carrega o conteúdo de d em $t2
lw $t3, 0($s4)           # carrega o conteúdo de e em $t3
```

$t2	10	75
$t3	11	15
$t4	12	0
$t5	13	0

Após isso, é feita a operação **c = d - a + e**, e o resultado da operação é salvo na variável c:

```
sub $t4, $t2, $t1       # faz a sub de d - a e salva em $t4
add $t4, $t4, $t3       # faz a soma de (d-a) + e e salva em $t4
sw $t4, 0($s2)          # salva o resultado da operação em c
```

E então o resultado dessa operação é salvo na variável c, utilizando a instrução sw:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	40	5	50

É então executado, por último, os blocos de instruções responsáveis por printar a variável string_result e o conteúdo da variável c, que consiste no resultado final das operações aritméticas:

```
li $v0, 4               # seta a operação de impressão de string
la $a0, string_result   # carrega o end de string_result para $a0
syscall                 # imprime string_result no console

li $v0, 1               # seta a operação de impressão de int
move $a0, $t4           # passa o conteúdo de $t4 (resultado da operação) para $a0
syscall                 #imprime o resultado final salvo em c
```

Onde obtêm-se:

ages | Run I/O

```
Digite o valor de b: 5
O resultado de c é: 50
```

O estado final da memória após a execução é:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	40	5	50	75	15

As variáveis a, b, e c foram iniciadas em 0, visto que teriam seus conteúdos sobrescritos. Já as variáveis d e e foram iniciadas em 75 e 15, respectivamente. Foi então atribuída a variável b o valor 5 (via teclado). São feitas então as operações:

a = b + 35, com b valendo 5
Logo a = 40.

c = d - a + e, com d valendo 75, a valendo 40, e e valendo 15
Logo c = 50.

Portanto, é possível verificar o funcionamento correto do programa. O programa do exercício 2 apresenta 32 linhas de assembly (sem pseudoinstruções), que são:

Basic
lui $1,4097
ori $16,$1,0
lui $1,4097
ori $17,$1,4
lui $1,4097
ori $18,$1,8
lui $1,4097
ori $19,$1,12
lui $1,4097
ori $20,$1,16
addiu $2,$0,4
lui $1,4097
ori $4,$1,20
syscall
addiu $2,$0,5
syscall
addu $8,$0,$2
sw $8,0($17)
addi $9,$8,35
sw $9,0($16)
lw $10,0($19)
lw $11,0($20)
sub $12,$10,$9
add $12,$12,$11
sw $12,0($18)
addiu $2,$0,4
lui $1,4097
ori $4,$1,42
syscall
addiu $2,$0,1
addu $4,$0,$12
syscall