

Atividade Prática I

Organização de Computadores I

Gabriel Salmoria e Gustavo Nunes Viana

- Atividade 1:

Realizar as seguintes operações:

$$a = b + 35;$$

$$c = d - a + e;$$

Resolução:

Primeiramente, definimos as variáveis que iremos utilizar para os valores de A, B, C, D e E. Para isso, vamos utilizar a keyword ".word" para registrar os valores em algum local desconhecido na memória.

```
.data
A: .word 0
B: .word 25
C: .word 0
D: .word 10
E: .word 20
```

Depois, iremos carregar os ENDEREÇOS das variáveis que iremos utilizar. Para isso, utilizaremos da pseudo-instrução "load address", que irá carregar o endereço das variáveis definidas anteriormente em registradores temporários.

```
.text
la $t0, A
la $t1, B
la $t2, C
la $t3, D
la $t4, E
```

Após carregar os endereços, iremos carregar os VALORES das variáveis. Para tal, faremos uso da instrução "Load Word", que irá carregar os valores verdadeiros das variáveis nos registradores source.

```
la $t0, A
la $t1, B
la $t2, C
la $t3, D
la $t4, E
```

Finalmente, iremos realizar as operações matemáticas. Primeiro iremos calcular a expressão:

$$a = b + 35;$$

Utilizando da instrução Adição imediata, ou Addi. Iremos passar os valores de A (registrador s0), B (registrador s1) e o valor imediato 35.

```
addi $s0, $s1, 35
```

Depois disso, iremos calcular a expressão:

$$c = d - a + e;$$

Para isso, iremos realizar duas operações:

$$c = d - a;$$

$$c = c + e;$$

Tal expressão deve ser quebrada em duas pela natureza das instruções matemáticas do Assembly, na qual, não podemos ter mais que três variáveis envolvidas na mesma operação.

```
sub $s2, $s3, $s0;  
add $s2, $s2, $s4;
```

Por fim, desejamos guardar o resultado das operações na memória de dados, para isso, iremos utilizar a instrução Store Word, ou sw.

```
sw $s2, 0($t2)
```

- Atividade 2:

Adaptar a Atividade 1 para que o valor de B seja fornecido pelo usuário, e o valor de C seja apresentado no terminal além de ser salvo na variável B

Resolução:

Nessa questão, o nosso desafio está em interagir com o sistema operacional tanto para receber o valor de B quanto para mostrar o valor de C. Para isso, iremos utilizar a função syscall, inicialmente vamos passar o valor 5, para receber o valor digitado pelo usuário.

```
li $v0, 5  
syscall  
move $s1, $v0
```

Além disso, iremos utilizar, ao fim do programa, a função syscall novamente, para mostrar o valor de C no terminal. Para isso, vamos passar o valor 1:

```
li $v0, 1  
add $a0, $a0, $s2  
syscall
```

Edit Execute					Registers Coproc 1 Coproc 0		
Text Segment					Name	Number	Value
Bkpt	Address	Code	Basic	Source			
	0x00400000	0x3c011001 lui	\$1, 4097	21: la \$t0, A	\$zero	0	0
	0x00400004	0x34280000 ori	\$8, \$1, 0		\$at	1	268500992
	0x00400008	0x3c011001 lui	\$1, 4097	22: la \$t1, B	\$v0	2	0
	0x0040000c	0x34290004 ori	\$9, \$1, 4		\$v1	3	0
	0x00400010	0x3c011001 lui	\$1, 4097	23: la \$t2, C	\$a0	4	0
	0x00400014	0x342a0008 ori	\$10, \$1, 8		\$a1	5	0
	0x00400018	0x3c011001 lui	\$1, 4097	24: la \$t3, D	\$a2	6	0
	0x0040001c	0x342b000c ori	\$11, \$1, 12		\$a3	7	0
	0x00400020	0x3c011001 lui	\$1, 4097	25: la \$t4, E	\$t0	8	268500992
	0x00400024	0x342c0010 ori	\$12, \$1, 16		\$t1	9	268500996
	0x00400028	0x8d100000 lw	\$16, 0(\$8)		\$t2	10	268501000
	0x0040002c	0x8d310000 lw	\$17, 0(\$9)		\$t3	11	268501004
	0x00400030	0x8d520000 lw	\$18, 0(\$10)		\$t4	12	268501008
					\$t5	13	0
					\$t6	14	0
					\$t7	15	0
					\$s0	16	60
					\$s1	17	25
					\$s2	18	-30
					\$s3	19	10
					\$s4	20	20
					\$s5	21	0
					\$s6	22	0
					\$s7	23	0
					\$s8	24	0
					\$s9	25	0
					\$k0	26	0
					\$k1	27	0
					\$gp	28	268468224
					\$sp	29	2147479548
					\$fp	30	0
					\$ra	31	0
					pc		4194380
					hi		0
					lo		0

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	25	-30	10	20	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0

No nosso código da Atividade 1 há 19 linhas de instrução, sem contar pseudo-instruções

Ao fim de uma execução completa, esse é o resultado do programa:

No nosso código da Atividade 2 há 21 linhas de instrução, sem contar pseudo-instruções

Ao fim de uma execução completa, esse é o resultado do programa:

Edit Execute					Registers Coproc 1 Coproc 0		
Text Segment					Name	Number	Value
Bkpt	Address	Code	Basic	Source			
	0x00400000	0x3c011001 lui	\$1, 4097	20: la \$t0, A	\$zero	0	0
	0x00400004	0x34280000 ori	\$8, \$1, 0		\$at	1	268500992
	0x00400008	0x3c011001 lui	\$1, 4097	21: la \$t2, C	\$v0	2	0
	0x0040000c	0x342a0004 ori	\$10, \$1, 4		\$v1	3	0
	0x00400010	0x3c011001 lui	\$1, 4097	22: la \$t3, D	\$a0	4	-10
	0x00400014	0x342b0008 ori	\$11, \$1, 8		\$a1	5	0
	0x00400018	0x3c011001 lui	\$1, 4097	23: la \$t4, E	\$a2	6	0
	0x0040001c	0x342c000c ori	\$12, \$1, 12		\$a3	7	0
	0x00400020	0x8d100000 lw	\$16, 0(\$8)		\$t0	8	268500992
	0x00400024	0x8d520000 lw	\$18, 0(\$10)		\$t1	9	0
	0x00400028	0x8d730000 lw	\$19, 0(\$11)		\$t2	10	268500996
	0x0040002c	0x8d940000 lw	\$20, 0(\$12)		\$t3	11	268501000
	0x00400030	0x24020005 addiu	\$2, \$0, 5		\$t4	12	268501004
	0x00400034	0x0000000c syscall			\$t5	13	0
	0x00400038	0x00028821 addu	\$17, \$0, \$2		\$t6	14	0
	0x0040003c	0x22300023 addi	\$16, \$17, 35		\$t7	15	0
	0x00400040	0x02709022 sub	\$18, \$19, \$16		\$s0	16	40
	0x00400044	0x02549020 add	\$18, \$18, \$20		\$s1	17	5
	0x00400048	0xad520000 sw	\$18, 0(\$10)		\$s2	18	-10
	0x0040004c	0x24020001 addiu	\$2, \$0, 1		\$s3	19	10
	0x00400050	0x00922020 add	\$4, \$4, \$18		\$s4	20	20
	0x00400054	0x0000000c syscall			\$s5	21	0
					\$s6	22	0
					\$s7	23	0
					\$s8	24	0
					\$s9	25	0
					\$k0	26	0
					\$k1	27	0
					\$gp	28	268468224
					\$sp	29	2147479548
					\$fp	30	0
					\$ra	31	0
					pc		4194392
					hi		0
					lo		0

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	-10	10	20	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0

```
-- program is finished running (dropped off bottom) --
5
-10
-- program is finished running (dropped off bottom) --
```