

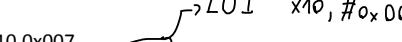
Exercício 1

- a) Considere o seguinte cálculo
 $3 + 7 - 4$
Escreva o programa que faz este cálculo. Utilize os registos x10, x11 e x12 para armazenar o valor das 3 constantes e coloque o resultado em x10.

b) Altere o programa para fazer
 $8 * (3 + 7 - 4)$
Utilizando uma instrução sll para efetuar a multiplicação por 8.

c) Considere a constante 824. Determine o valor do segundo símbolo hexadecimal usando as instruções and e srl.

d) Inverta o valor do 9º bit do número 824 utilizando a instrução EOR.

a) 

$$c) \quad f_{24}_{10} = \underline{\overline{338}}_h \\ 12 \text{ b}_1 t_0$$

```
li x10,0x338  
li x11,0x008  
and x11,x10,x11  
srl x10,x10,x11
```

6

```
li x10,0x338  
li x11,0x100  
xor x10,x10,x11
```

1

```
.data  
a: .byte 3  
b: .half 4523  
c: .word 100000
```

```

.text
lb x10,a
lh x11,b
lw x12,c
add x12,x12,x11
add x12,x12,x10

lw x11,x
li x13,0x10000000
addi x10,x10,7
add x10,x11,x10
slli x12,x11,2
add x10,x10,x12
mul x12,x11,x11
slli x12,x12,1
sub x10,x10,x12
mul x12,x11,x11
mul x12,x12,x11
add x10,x10,x12
slli x12,x12,1
add x10,x10,x12
sw x10,4(x13)

```

Exercício 2

- a) Considere o seguinte código em C

```
int a = 3, b = 7, c = 4, d;  
d = a + b - c + 2c;
```

Escreva o mesmo código em Assembly. Reserve espaço para as variáveis na memória utilizando a diretiva ".word". Utilize registos sempre que possível, já que um acesso a este é mais rápido do que um acesso à memória.

b) Escreva um programa que determine a média aritmética de quatro valores em memória nomeadamente:

vals .word 3, 7, 4, 6

a)	a: .word 3 b: .word 7 c: .word 4 d: .word 0	(-)	vals: .word 3,7,4,
	lw x10,a	lw x14,0x1000000	
	lw x11,b	lw x10,0(x14)	
	lw x12,c	lw x11,4(x14)	
	add x13,x11,x10	lw x12,8(x14)	
	sub x13,x13,x12	lw x13,12(x14)	
	slli x12,x12,1	add x10,x10,x11	
	add x13,x13,x12	add x10,x10,x12	
		srlt x10,x10,2	

Is: word 3 7 4

```
li x14,0x1000000  
lw x10,0(x14)  
lw x11,4(x14)  
lw x12,8(x14)  
lw x13,12(x14)  
add x10,x10,x11  
add x10,x10,x12  
add x10,x10,x13  
srlx x10,x10,2
```

c) a: .word 3
b: .word 1

```
c: .word 4  
d: .word 0  
lw x10,a  
lw x11,b  
lw x12,c  
slli x13,x10,7  
srli x11,x11,2  
slli x12,x12,3  
add x13,x13,x1  
add x13,x13,x1  
sw x13,12(x14)
```

Exercício 3

- a) Escreva o programa que conta o número de bits com o valor '1' numa constante de quatro bits utilizando as instruções srl e and. Teste o programa utilizando a constante 0xD.

b) Considere a seguinte lista de números:

```
vals .word 3, 7, 4, 6
```

Instrua uma segunda lista (vacc) cujo elemento i guarda o somatório dos elementos da lista vals i. Para o exemplo fornecido, a segunda lista seria constituída com os números 3,10,14,20.

Escreva o programa que corresponde ao seguinte código em C.

d = 128*a + b/4 + 8*c

d) Escreva o programa em assembly que permite calcular o valor do polinômio $y = 3x^3 - 2x^2 + 5x + 7$ dado o valor de x inteiro. As variáveis x e y estão armazenadas em memória.

e) Considere o seguinte código em C

```
char a = 3;      // 8 bits  
short b = 4523; // 16 bit  
long c = 100000; // 32 bit
```

Considere que estes operandos estão armazenados em memória utilizando as seguintes diretivas:

a	.byte	3
b	.half	4523
c	.word	10000

Escreva o código que lê estes valores da memória e obtém a sua soma.

.data
a: .word 0xcde8

```

.text lw x10,a lw x11,a
main:
srli x11,x11,1
slli x11,x11,1
beq x10,x0,end
beq x10,x11,zer0
bne x10,x11,um

zero:
srli x11,x11,1
srli x10,x10,1
jal main

.text
li x10,0x10000000
lw x11,0(x10)
add x12,x0,x11
sw x12,16(x10)
lw x11,4(x10)
add x12,x12,x11
sw x12,20(x10)
lw x11,8(x10)
add x12,x12,x11
sw x12,24(x10)
lw x11,12(x10)

```

```
um:  
srl x11,x11,1  
srl x10,x10,1  
addi x13,x13,1  
jal main
```

end