EJERCICOS SOBRE PROGRAMACIÓN EN ENSAMBLADOR DEL PROCESADOR *MIPS R2000*

EJERCICIO 1.−

¿Cómo almacenaría los siguientes datos en memoria?

- **La cadena de caracteres: El resultado de la suma es:**
- **El numero en coma flotante simple precisión: 5.46e−2**
- **El byte: 255 (0xff)**
- **La media palabra: 0x74af**
- **El carácter `a':**

a) cadena: .asciiz El resultado de la suma es

b) .float 5.46e−2

c) .byte 255

d) .half 0x74af

e).ascii a

EJERCICIO 2.−

Escribir el código en ensamblador que realiza las siguientes acciones:

- **Imprime el siguiente mensaje: La media es: 55.63**
- **Lee tu nombre por teclado: Introduzca su nombre:**
- **Crear un procedimiento que imprime una cadena y un entero que se pase como parámetro**
- **Crear un procedimiento que realice la suma de dos números en coma flotante y devuelva el resultado en $v0**

a)

```
.data

num: .ascii La media es: 55.63

.text

main:

li $2, 4

la $4, num

syscall

li $v0, 10
```

```
syscall

b)

.data

cadena2: .asciiz Introduzca su nombre:

.text

main:

la $a0, cadena2

li $v0,4

syscall

li $v0, 8

syscall

la $a0, PEPE

li $v0, 8

syscall

c) .data

cad: .asciiz "Escribe un numero:"

.text

.globl __start

__start: li $2, 4

la $4, cad

syscall

li $2, 5

syscall

move $v1, $2

li $2, 10

syscall
```

d)

```
.data

var_a: .float 0.01

var_b: .float 0.001

.text

main:

jal suma

mov.s $f12, $v0

li $v0,2

syscall

li $v0,10

syscall

suma:

l.s, $f4, var_a

l.s, $f2, var_b

add.s $f12,$f2,$f4

jr $31
```

EJERCICIO 3.−

¿Cuál es la etiqueta que marca el comienzo de nuestro programa en caso de que load trap file? ¿y si load trap file no esta seleccionado?

Cuando se carga el manejador de instrucciones, la etiqueta que marca el comienzo del programa debe de ser main en lugar de _start, que se empleará cuando load trap file no está seleccionado.

EJERCICIO 4.−

Si la directiva .data se utiliza sin ningún argumento. ¿Dónde almacenará los datos que lleven asociados otra directiva que acompañe? ¿A partir de que dirección?

Si la directiva .data se utiliza sin ningún argumento, los datos que lleven asociados otras directivas que le acompañen se almacenarán en el argumento de datos, a partir de la dirección 0x10000000H.

EJERCICIO 5.−

Para un programa de usuario, es seguro en algún momento usar los registros$k0 y $k1.

No es seguro usar los registros $k0 y $k1, porque están reservados para el uso del núcleo operativo y pueden borrar lo que se encuentra almacenado.

EJERCICIO 6.−

Indíquese la instrucción MIPS o la mínima secuencia de instrucciones para implementar la sentencia x := y * 100. Supóngase que x se encuentra en el registro $11 e y en el $12.

mult $11, $12, 100

EJERCICIO 7.−

Indíquese la instrucción MIPS o la mínima secuencia de instrucción que realiza la sentencia a[23]:= a[24] + x; supóngase que a es un vector de 100 elementos que comienza en la dirección de memoria 0x1000A000 y la variable x se encuentra en el registro $15.

load

lw $t0, 0x1000A18

add $ti, $t0, $15

sw $ti, 0x1000A17

EJERCICIO 8.−

El siguiente trata de copia palabras desde la dirección de memoria que indica el registro $4 en la dirección que indica el registro $5; el registro $2 lleva la cuenta de las palabras copiadas. El programa se detiene cuando se encuentra una palabra igual a cero. No se han de guardar los contenidos de los registros $3, $4 y $5. La palabra de terminación(que estará a cero) debe ser leída pero no copiada

bucle: lw, $3, 0($4) #lee siguiente palabra fuente

addi $2, $2, 1 # incrementa número de palabra copiadas

sw $3, 0($5) # copia la palabra

addi $4, $4, 1 # avanza puntero a siguiente palabra fuente

addi $5, $5, 1 # avanza puntero a siguiente palabra destino

bne $3, $0, bucle # va a bucle si palabra copiada no es cero

El programa anterior hay multitud de fallos (bugs). Determínese estos fallos y cámbiese el programa para que funcione perfectamente

.text

main: lw $3, 0($4)

```
beq $3, $0, fin

j bucle

bucle: addi $2, $2,1

sw $3, 0($5)

addi $4, $4,1

j main

fin: lw $3, 0($4)

lw $4, 0($4)

lw $5, 0($4)

.end
```

EJERCICIO 9.−

Utilícese el programa anterior (tal y como se proporciona, es decir con los errores), determínese el formato de instrucción para cada una de las instrucciones que lo componen, así como los valores decimales >(o hexadecimal) de cada campo del formato. ¿ Que tamaño ocupa en memoria?

El formato para la instrucción es addi, y el valor hexadecimal que ocupa es el 0x20.

El valor hexadecimal que ocupa bne es el 0x14.

El valor hexadecimal que ocupa lw es el 0x8c.

El valor hexadecimal que ocupa sw es el 0xac.

EJERCICIO 10.−

Escríbase una subrutina en lenguaje MIPS que implemente el procedimiento siguiente

PROCEDIMIENTO máximo (a, b: entero; var max: entero)

Empezar

Si (a>=b) entonces max:=a;

Sino max := b;

Fin_procedimiento

Téngase en cuenta las siguientes suposiciones

- **Los argumentos se pasan a la subrutina utilizando los registros destinados al efecto $4, ;$7.**
- **El resultado de la subrutina hay que devolverlo a través del registro $2.**

- **No se origina desbordamiento en las operaciones, y su resultado nunca excede de 32 bits.**
- **La preservación de registros en la pila se lleva a cabo mediante el convenio de guardar invocada (callee save).**

```
.data

int: .asciiz "El máximo entre "

y:.asciiz "y "

es:.asciiz "es: "

eq:.asciiz "¿y si fueran iguales?"

.text

main: la $a0, int

li $2, 4

syscall

li $2, 5

syscall

add $t0, $2, $0

la $a0, y

li $2, 4

syscall

li $2, 5

syscall

add $t1, $2, $0

la $a0, es

li $2, 4

syscall

subu $t2, $t0, $t1

bgtz $t2, print_a

beq $t2, $0, print_b
```

```
add $a0, $t1, $0

li $2, 1

syscall

j fin

print_a: add $a0, $t0, $0

li $2, 1

syscall

j fin

print_b: la $a0, eq

li $2, 4

syscall

fin: li $2, 10

syscall
```

EJERCICIO 11.−

Escríbase una subrutina en lenguaje MIPS que implemente una función que retorne el valor del cubo de un número introducido por teclado.

```
.data

var: .asciiz "El cubo de:"

var1:.asciiz "es:"

.text

main: la $a0, var

li $2, 4

syscall

li $2, 5

syscall

add $t0, $2, $0

la $a0, var1
```

```
li $2, 4

syscall

mul $t1, $t0, $t0

mul $a0, $t1, $t0

li $2, 1

syscall

li $2, 10

syscall
```

EJERCICIO 12.−

Implementar una función que diga si un valor introducido por teclado es par o impar.

```
.data

par: .asciiz "yo pa mi que es par"

impar:.asciiz "hasta mi abuela sabe que es impar"

inic:.asciiz "El número introducido es: "

.text

main: addi $t0, $0, 2

la $a0, inic

li $2, 4

syscall

li $2, 5

syscall

div $2, $t0

mfhi $t1

beq $t1, $0, print_1

la $a0, impar

li $2, 4
```

```
syscall

j fin

print_1: la $a0, par

li $2, 4

syscall

j fin

fin: li $2, 10

syscall
```

EJERCICIO 13.−

Implementar un programa que escribe los números del 1 al 10

```
.text

main: addi $t1, $0, 10

addi $t0, $0, 1

add $a0, $t0, $0

li $2, 1

syscall

bucle: addi $t0, $t0, 1

add $a0, $t0, $0

li $2, 1

syscall

bne $t1, $t0, bucle

li $2, 10

syscall
```

EJERCICO 14.−

Implementar un programa que implemente la función de una contraseña.

.data

```
princ:.asciiz "Introduzca la contraseña:"
error:.asciiz "contraseña no válida, preparados para autodestrucción"
ok:.asciiz "eres el jefe"
.text
main: addi $s0, $0, 230181
la $a0, princ
li $2, 4
syscall
li $2, 5
syscall
add $t0, $2, $0
beq $s0, $t0, vale
la $a0, error
li $2, 4
syscall
tal: addi $a0, $0, 1
li $2, 1
syscall
j tal
fin: li $2, 10
syscall
vale: la $a0, ok
li $2, 4
syscall
j fin
```

EJERCICIO 15.−

Implementar la siguiente función recursiva:

Función suma (n: entero)

Si (n = 1) entonces suma := 1

Si no suma := n + suma (n − 1)

Fin_función

```
.data
var1:.asciiz "El resultado de la suma es: "
var2:.asciiz "Introduzca un número "
.text
main: addi $t0, $0, 1
la $a0, var2
li $2, 4
syscall
li $2, 5
syscall
beq $t0, $2, print1
add $t1, $2, $0
add $t2, $2, $0
sumar: addi $t1, $t1, −1
add $t2, $t1, $t2
beq $t1, $t0, print2
j sumar
print1: la $a0, var1
li $2, 4
syscall
addi $a0, $0, 1
```

```
li $2, 1

syscall

j fin

print2: la $a0, var1

li $2, 4

syscall

add $a0, $t2, $0

li $2, 1

syscall

j fin

fin: li $2, 10

syscall
```

EJERCICIO 16.−

Escribir un programa que lea tu nombre, y tu primer apellido y los escriba en la consola de salida

```
.data

var:.asciiz "JOSÉ ABRIL"

.space 100

.text

_start: la $4, var

li $2, 4

syscall

li $2,5

syscall

li $2, 10

syscall
```