Sesión 4 Estructura de Computadores

Temario: Estructura de switch en ensamblador MIPS

```
.text
.globl main
main:
        li $s0, 0               #Esta es la opción que se mostrará por pantalla
        bne $s0, 0, c1          #Salta a c1 si s0 es diferente de 0
        la $a0, Badajoz         #Carga en a0, la cadena Badajoz
        b cx                    #Salto incondicional a Cx (caso default de switch)

c1:     bne $s0, 1, c2
        la $a0, Caceres
        b cx

c2:     bne $s0, 2, c3
        la $a0, Merida
        b cx

 c3:    bne $s0, 3, c4
        la $a0, Plasencia
        b cx

c4:     la $a0, Otras

cx:     li $v0, 4               #Imprime por pantalla la cadena que ha elegido el usuario
        syscall
        li $v0, 10
        syscall

.data
Badajoz: .asciiz "Badajoz\n"
Caceres: .asciiz "Caceres\n"
Merida: .asciiz "Merida\n"
Plasencia: .asciiz "Plaencia\n"
Otras: .asciiz "Otras\n"

```

En este ejemplo, al ser la elección del usuario un 0, se muestra por pantalla Badajoz.

```
Badajoz

-- program is finished running --
```

Ejercicio para entregar:

Pedir un número al usuario. Dependiendo del número, hay que hacer:

- **0: sumar por posición los 2 vectores y almacenar el resultado en un tercer vector.**
- **1: restar por posición los 2 vectores y almacenar el resultado en un tercer vector.**
- **Default: poner las posiciones del array Resultado a 0.**

Y, mostrar por pantalla el vector resultado.

```
#Ejercicio 8

.data
array1: .word 1, 3, 5, 7
array2: .word 0, 2, 4, 6
arrayFinal: .word 0, 0, 0, 0
size: .word 4
string: .asciiz "Ejercicio 8. Cálculo sobre arrays\n 0. Sumar por posición\n 1. Restar por posición\n Resto: vector solución a 0\n"
pedir: .asciiz "Introduce la elección: "
string2: .asciiz "Array resultado: "
salto: .asciiz ", "

.text
.globl main

main:   li $v0, 4               #Cadena 1 por pantalla
        la $a0, string
        syscall

        li $v0, 4               #Cadena pedir por pantalla
        la $a0, pedir
        syscall

        li $v0, 5               #Coger la elección del usuario por teclado
        syscall
        add $a3, $zero, $v0     #Guardo en a3 la elección del usuario

        jal calc                #salto a calculo

        li $v0, 4
        la $a0, string2         #Cadena 2 por pantalla
        syscall


        la $s0, arrayFinal      #Guardo en s0, el arrayFinal
        li $t0, 0               #Índice segundo bucle

mostar: bge $t0, $t2, ipx       #Procedimiento para mostrar el array por pantalla

        lw $a0, 0($s0)
        li $v0, 1
        syscall
        li $v0, 4
        la $a0, salto
        syscall
        addi $s0, $s0, 4
        addi $t0, $t0, 1
        b mostar

ipx:    li $v0, 10
        syscall

#Cálculo-------------------------------------------------
calc:
        la $a0, array1          #Parámetro: array1
        la $a1, array2          #Parámetro: array2
        la $a2, arrayFinal      #Parámetro: array final

        li $t0, 0               #Índice
        lw $t2, size            #Tamaño array


bucle:  bge $t0, $t2, fuera     #Saltará al final cuando el índice sea igual o mayor al tamaño del array
        lw $t3, 0($a0)          #t0 cojo la posición del array 1
        lw $t4, 0($a1)          #t1 cojo la posición del array 2

        bne $a3, 0 , c1         #Empieza el switch. Si la elección es diferente a 0, salta a c1
        add $t5, $t3, $t4       #Si es 0, hace la suma de los elementos del array de la misma posición
        sw $t5, 0($a2)          #Cargo el resultado en la posción del arrayFinal correspondiente

        b finS                  #Salto incondicional a la etiqueta para incrementar el índice de los arrays

c1:     bne $a3, 1, c2          #Si la elección es diferente a 1, salta a c2
        sub $t5, $t3, $t4       #Si es 1, hace la resta de los elementos del array de la misma posición
        sw $t5, 0($a2)          #Cargo el resultado en la posción del arrayFinal correspondiente

        b finS                  #Salto incondicional a la etiqueta para incrementar el índice de los arrays


c2:     addi $t7, $t7, 0        #caso default del Switch
        sw $t7, 0($a2)          #Pone la posición del arrayFinal a 0
        b finS                  #Salto incondicional a la etiqueta para incrementar el índice de los arrays


finS:   addi $t0, $t0, 1        #Fynción para incrementar el índice de los arrays-
        addi $a0, $a0, 4
        addi $a1, $a1, 4
        addi $a2, $a2, 4
        b bucle                 #Salto incondicional a bucle. Aquí se producen las iteracciones


fuera: jr $ra                   #Salto incondicional a la dirección alamcenada en el registro $ra, que coincide con la dirección de retorno de jal.
```

A continuación, voy a enseñar capturas de ejemplo de los 3 posibles resultados:

Suma (elección 0): 1+0, 3+2, 5+4, 7+6 = **1, 5, 7, 13**

```
Ejercicio 8. Cálculo sobre arrays
 0. Sumar por posición
 1. Restar por posición
 Resto: vector solución a 0
Introduce la elección: 0
Array resultado: 1, 5, 9, 13,
-- program is finished running --
```

Resta (elección 1): 1-0, 3-2, 5-4, 7-6 = **1, 1, 1, 1**

```
Ejercicio 8. Cálculo sobre arrays
 0. Sumar por posición
 1. Restar por posición
 Resto: vector solución a 0
Introduce la elección: 1
Array resultado: 1, 1, 1, 1,
-- program is finished running --
```

Default: **0, 0, 0, 0**

```
Ejercicio 8. Cálculo sobre arrays
 0. Sumar por posición
 1. Restar por posición
 Resto: vector solución a 0
Introduce la elección: 6
Array resultado: 0, 0, 0, 0,
-- program is finished running --
```