Solution of Lab Series No. 3

Exercise No. 1: What Does This Program Do?

- Affiche les neuf premières valeurs du tableau.
- Demande aux étudiants d'apporter les modifications nécessaires afin d'afficher les 10 entiers séparer par des virgules

Exercise No. 2: Finding the Minimum and Maximum in an Arr

```
.data
array: .word 12, 45, 7, 23, 89, 5, 16, 37, 78, 2  # Tableau initialisé
min:   .word 0
max:   .word 0
msg_min: .asciiz "Minimum: "
msg_max: .asciiz "\nMaximum: "

.text
.globl main
main:
   la $t0, array
   lw $t1, 0($t0)      #
   move $t2, $t1       #
   move $t3, $t1       #
   li $t4, 10          #
   li $t5, 0           #

loop:
   lw $t6, 0($t0)      #
   blt $t6, $t2, update_min  # Si $t6 < min, mettez à jour min
   bgt $t6, $t3, update_max  # Si $t6 > max, mettez à jour max
   j next

update_min:
   move $t2, $t6       # Mettre à jour min
   j next

update_max:
   move $t3, $t6       # Mettre à jour max

next:
   addi $t5, $t5, 1    # Incrémenter l'index
   addi $t0, $t0, 4    # Passer au prochain élément
   bne $t5, $t4, loop  # Continuer tant que $t5 < taille

# Afficher le résultat
   li $v0, 4           #
   la $a0, msg_min     #
   syscall

   li $v0, 1           #
   move $a0, $t2       #
   syscall

   li $v0, 4           #
   la $a0, msg_max     #
```

```
syscall

li $v0, 1           #
move $a0, $t3       #
syscall

li $v0, 10          #
syscall
```

Exercise No. 3: Calculating the Sum and Product of Array Elements

```
.data
array: .word 2, 3, 4, 5, 6, 7, 8, 9, 10, 11  # Tableau initialisé
sum:   .word 0
prod:  .word 1
msg_sum: .asciiz "Sum: "
msg_prod: .asciiz "\nProduct: "

.text
.globl main
main:
   la $t0, array       # Charger l'adresse du tableau dans $t0
   li $t1, 0           # Initialiser la somme à 0
   li $t2, 1           # Initialiser le produit à 1
   li $t3, 10          # Taille du tableau
   li $t4, 0           # Index

loop:
   lw $t5, 0($t0)      #
   add $t1, $t1, $t5   #
   mul $t2, $t2, $t5   #
   addi $t4, $t4, 1    #
   addi $t0, $t0, 4    #
   bne $t4, $t3, loop  #

# Afficher la somme
   li $v0, 4           #
   la $a0, msg_sum     #
   syscall

   li $v0, 1           #
   move $a0, $t1       #
   syscall

# Afficher le produit
   li $v0, 4           #
   la $a0, msg_prod    #
   syscall

   li $v0, 1           #
   move $a0, $t2       #
   syscall

   li $v0, 10          #
   syscall
```

Exercise No. 4: Displaying the Position of the Longest Consecutive Sequence of Zeros

```
.data
array: .word 1, 0, 1, 0, 1, 0, 0, 1, 0, 0  # Example array
size: .word 10                      # Size of the array
msg_start: .asciiz "Start position of the longest sequence of zeros: "

.text
.globl main
main:
   la $t0, array       # Load the address of the array into $t0
   lw $t1, size        # Load the size of the array into $t1
   li $t2, 0           # Current index (i)
   li $t3, 0           # Current sequence length
   li $t4, 0           # Maximum sequence length
   li $t5, -1          # Start position of the current sequence
   li $t6, -1          # Start position of the longest sequence

loop:
   beq $t2, $t1, finish #
   lw $t7, 0($t0)      #
   beqz $t7, zero_found #
   j non_zero          #

zero_found:
   beq $t3, 0, new_seq  #
   addi $t3, $t3, 1     #
   j next

new_seq:
   move $t5, $t2        #
   addi $t3, $t3, 1     #
   j next

non_zero:
   blt $t3, $t4, reset  #
   move $t4, $t3        #
   move $t6, $t5        #
   j reset

reset:
   li $t3, 0           #
   li $t5, -1          #
   j next

next:
   addi $t2, $t2, 1     #
   addi $t0, $t0, 4     #
   j loop              #

finish:
   # Display the result
   li $v0, 4           #
   la $a0, msg_start    #
   syscall

   li $v0, 1           #
```

```
   move $a0, $t6       #
   syscall

   li $v0, 10          #
   syscall
```

Exercise No. 5: Displaying Even and Odd Elements of a Matrix

```
.data
matrix: .space 36              # Allocate space for 3x3 matrix (9 integers, 4 bytes each)
msg_input: .asciiz "Enter a number: "
msg_even: .asciiz "Number of even elements: "
msg_odd: .asciiz "\nNumber of odd elements: "

.text
.globl main

main:
   li $t0, 0                # Initialize loop counter for 9 elements (matrix size)
   li $t1, 9                # Total number of elements (3x3)
   la $t2, matrix           # Base address of the matrix

input_loop:
   beq $t0, $t1, process_matrix #
   li $v0, 4                #
   la $a0, msg_input        #
   syscall

   li $v0, 5                #
   syscall
   sw $v0, 0($t2)           #
   addi $t2, $t2, 4         #
   addi $t0, $t0, 1         #
   j input_loop             #

process_matrix:
   li $t3, 0                #
   li $t4, 0
   la $t2, matrix
   li $t0, 0

count_loop:
   beq $t0, $t1, display_result
   lw $t5, 0($t2)
   rem $t6, $t5, 2
   beqz $t6, increment_even
   addi $t4, $t4, 1
   j next_element

increment_even:
   addi $t3, $t3, 1

next_element:
   addi $t2, $t2, 4
   addi $t0, $t0, 1
```

```
    j count_loop

display_result:
    li $v0, 4
    la $a0, msg_even
    syscall

    li $v0, 1
    move $a0, $t3
    syscall

    li $v0, 4
    la $a0, msg_odd
    syscall

    li $v0, 1
    move $a0, $t4
    syscall

    li $v0, 10
    syscall
```