

Asignatura	Datos del alumno	Fecha
<b>Estructura de Computadores</b>	Apellidos: Garzon Cantor	17/03/2025
	Nombre: María Isabel	

## Laboratorio #1: Simulación y optimización de un programa en un procesador escalar segmentado

### 1. Introducción

En este informe se documenta el desarrollo de tres programas en lenguaje ensamblador utilizando MARS 4.5. Los programas implementados son:

1. Búsqueda del número mayor: Permite ingresar un conjunto de números y determina el mayor de ellos.
2. Búsqueda del número menor: Similar al anterior, pero encuentra el menor de los valores ingresados.
3. Serie Fibonacci: Genera una cantidad determinada de números de la serie de Fibonacci e imprime la suma total.

Cada uno de estos programas fue desarrollado siguiendo las normas establecidas para el laboratorio y subido al repositorio de GitHub.

### 2. Desarrollo de la actividad

#### 2.1. Descripción de los programas

Código 1: Búsqueda del número mayor

- Se solicita al usuario la cantidad de números (entre 3 y 5).
- Se ingresan los números y se almacenan en memoria.
- Se recorre la lista comparando los valores para encontrar el mayor.
- Se imprime el número mayor en la consola.

Código 2: Búsqueda del número menor

- La estructura es similar al código anterior.
- En lugar de comparar los números para encontrar el mayor, se busca el menor.
- Se imprime el número menor en la consola.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Garzon Cantor	17/03/2025
	Nombre: María Isabel	

### Código 3: Serie Fibonacci

- Se solicita al usuario la cantidad de términos de la serie a generar.
- Se almacenan los valores en un arreglo.
- Se imprimen los valores de la serie.
- Se calcula e imprime la suma total de los términos generados.

## 3. Captura de pantalla

### 3.1. Búsqueda del número mayor

```

1  .data
2  msg1: .asciiz "Ingrese la cantidad de números (3-5): "
3  msg2: .asciiz "Ingrese un número: "
4  msg3: .asciiz "El número mayor es: "
5  newline: .asciiz "\n"
6  numbers: .word 0, 0, 0, 0, 0 # Espacio para almacenar hasta 5 números
7
8  .text
9  .globl main
10
11 main:
12 # Mostrar mensaje para pedir cantidad de números
13 li $v0, 4
14 la $a0, msg1
15 syscall
16
17 # Leer la cantidad de números
18 li $v0, 5
19 syscall
20 move $t0, $v0 # Guardar la cantidad en $t0
21
22 # Validar que esté entre 3 y 5
23 li $t1, 3
24 blt $t0, $t1, exit # Si es menor a 3, salir
25 li $t1, 5
26 bgt $t0, $t1, exit # Si es mayor a 5, salir
27
28 # Leer los números del usuario
29 li $t2, 0 # Índice del arreglo
30 read_loop:
31 li $v0, 4
32 la $a0, msg2
33 syscall
34
35 li $v0, 5
36 syscall
37 sw $v0, numbers($t2) # Guardar número en el arreglo
38
39 addi $t2, $t2, 4 # Siguiendo posición
40 sub $t0, $t0, 1 # Reducir contador

```

```

36 syscall
37 sw $v0, numbers($t2) # Guardar número en el arreglo
38
39 addi $t2, $t2, 4 # Siguiendo posición
40 sub $t0, $t0, 1 # Reducir contador
41 bgtz $t0, read_loop # Repetir si quedan números por ingresar
42
43 # Buscar el número mayor
44 la $t2, numbers # Dirección base del arreglo
45 lw $t3, 0($t2) # Inicializar con el primer número
46 li $t4, 1 # Contador de iteraciones
47 find_max:
48 lw $t5, 0($t2) # Cargar número actual
49 bgt $t5, $t3, update_max # Si es mayor, actualizar
50 j next
51 update_max:
52 move $t3, $t5 # Actualizar el número mayor
53 next:
54 addi $t2, $t2, 4 # Siguiendo posición
55 addi $t4, $t4, 1 # Aumentar contador
56 blt $t4, 5, find_max # Repetir si quedan elementos
57
58 # Mostrar resultado
59 li $v0, 4
60 la $a0, msg3
61 syscall
62
63 li $v0, 1
64 move $a0, $t3
65 syscall
66
67 # Salto de línea
68 li $v0, 4
69 la $a0, newline
70 syscall
71
72 exit:
73 li $v0, 10
74 syscall # Terminar programa
75

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Garzon Cantor	17/03/2025
	Nombre: María Isabel	

### 3.2. Búsqueda del número menor

```

1  .data
2  msg1: .asciiz "Ingrese la cantidad de números (3-5): "
3  msg2: .asciiz "Ingrese un número: "
4  msg3: .asciiz "El número mayor es: "
5  newline: .asciiz "\n"
6  numbers: .word 0, 0, 0, 0, 0 # Espacio para almacenar hasta 5 números
7
8  .text
9  .globl main
10
11 main:
12     # Mostrar mensaje para pedir cantidad de números
13     li $v0, 4
14     la $a0, msg1
15     syscall
16
17     # Leer la cantidad de números
18     li $v0, 5
19     syscall
20     move $t0, $v0 # Guardar la cantidad en $t0
21
22     # Validar que esté entre 3 y 5
23     li $t1, 3
24     blt $t0, $t1, exit # Si es menor a 3, salir
25     li $t1, 5
26     bgt $t0, $t1, exit # Si es mayor a 5, salir
27
28     # Leer los números del usuario
29     li $t2, 0 # Índice del arreglo
30 read_loop:

```

```

46     li $t4, 1 # Contador de iteraciones
47 find_max:
48     lw $t5, 0($t2) # Cargar número actual
49     blt $t5, $t3, update_min # Si es mayor, actualizar
50     j next
51 update_min:
52     move $t3, $t5 # Actualizar el número menor
53 next:
54     addi $t2, $t2, 4 # Siguiete posición
55     addi $t4, $t4, 1 # Aumentar contador
56     blt $t4, 5, find_min # Repetir si quedan elementos
57
58     # Mostrar resultado
59     li $v0, 4
60     la $a0, msg3
61     syscall
62
63     li $v0, 1
64     move $a0, $t3
65     syscall
66
67     # Salto de línea
68     li $v0, 4
69     la $a0, newline
70     syscall
71
72 exit:
73     li $v0, 10
74     syscall # Terminar programa
75

```

```

31     li $v0, 4
32     la $a0, msg2
33     syscall
34
35     li $v0, 5
36     syscall
37     sw $v0, numbers($t2) # Guardar número en el arreglo
38
39     addi $t2, $t2, 4 # Siguiete posición
40     sub $t0, $t0, 1 # Reducir contador
41     bgtz $t0, read_loop # Repetir si quedan números por ingresar
42
43     # Buscar el número mayor
44     la $t2, numbers # Dirección base del arreglo
45     lw $t3, 0($t2) # Inicializar con el primer número
46     li $t4, 1 # Contador de iteraciones
47 find_max:
48     lw $t5, 0($t2) # Cargar número actual
49     blt $t5, $t3, update_min # Si es mayor, actualizar
50     j next
51 update_min:
52     move $t3, $t5 # Actualizar el número menor
53 next:
54     addi $t2, $t2, 4 # Siguiete posición
55     addi $t4, $t4, 1 # Aumentar contador
56     blt $t4, 5, find_min # Repetir si quedan elementos
57
58     # Mostrar resultado
59     li $v0, 4
60     la $a0, msg3

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Garzon Cantor	17/03/2025
	Nombre: María Isabel	

### 3.3. Serie Fibonacci

```

1  .data
2  msg1: .asciiz "Ingrese la cantidad de términos de Fibonacci: "
3  msg2: .asciiz "Serie Fibonacci: "
4  msg3: .asciiz "Suma total: "
5  newline: .asciiz "\n"
6  fib: .space 20 # Espacio para 5 números (4 bytes cada uno)
7
8  .text
9  .globl main
10
11 main:
12     # Pedir cantidad de términos
13     li $v0, 4
14     la $a0, msg1
15     syscall
16
17     # Leer cantidad de términos
18     li $v0, 5
19     syscall
20     move $t0, $v0 # Guardar en $t0
21
22     # Validar si es mayor a 1
23     li $t1, 1
24     blt $t0, $t1, exit
25
26     # Inicializar Fibonacci
27     la $t2, fib
28     li $t3, 0
29     sw $t3, 0($t2) # fib[0] = 0
30     li $t5, 1

```

```

31     sw $t3, 4($t2) # fib[1] = 1
32
33     # Generar serie
34     li $t4, 2
35 gen_fib:
36     lw $t5, -4($t2) # fib[n-1]
37     lw $t6, -8($t2) # fib[n-2]
38     add $t5, $t5, $t6
39     sw $t5, $t2
40
41     addi $t2, $t2, 4
42     addi $t4, $t4, 1
43     blt $t4, $t0, gen_fib
44
45     # Mostrar Fibonacci
46     li $v0, 4
47     la $a0, msg2
48     syscall
49
50     li $t4, 0
51     la $t2, fib
52 print_fib:
53     lw $a0, 0($t2)
54     li $v0, 1
55     syscall
56 Symbol "find_mind"
57     li $v0, 4
58     la $a0, newline
59     syscall
60

```

```

64     # Calcular suma
65     li $t4, 0
66     la $t2, fib
67     li $t3, 0
68 sum_fib:
69     lw $t5, 0($t2)
70     add $t3, $t3, $t5
71
72     addi $t2, $t2, 4
73     addi $t4, $t4, 1
74     blt $t4, $t0, sum_fib
75
76     # Mostrar suma total
77     li $v0, 4
78     la $a0, msg3
79     syscall
80
81     li $v0, 1
82     move $a0, $t3
83     syscall
84
85     li $v0, 4
86     la $a0, newline
87     syscall
88
89 exit:
90     li $v0, 10
91     syscall
92
93

```

### 4. Conclusiones

Se logró implementar y ejecutar correctamente los tres programas en lenguaje ensamblador.

Asignatura	Datos del alumno	Fecha
<b>Estructura de Computadores</b>	Apellidos: Garzon Cantor	17/03/2025
	Nombre: María Isabel	

Se utilizó MARS 4.5 como entorno de desarrollo y simulación.

Se comprendieron mejor los conceptos de almacenamiento en memoria y bucles en ensamblador.

Los archivos fueron organizados y subidos a GitHub para su evaluación.

#### 5. Enlace al repositorio en GitHub

Los códigos fuente han sido subidos al siguiente repositorio:

[[https://github.com/MariaGarzon19/Laboratorio\\_Simulaci-n-y-Optimizaci-n](https://github.com/MariaGarzon19/Laboratorio_Simulaci-n-y-Optimizaci-n)]  
 ([https://github.com/MariaGarzon19/Laboratorio\\_Simulaci-n-y-Optimizaci-n.git](https://github.com/MariaGarzon19/Laboratorio_Simulaci-n-y-Optimizaci-n.git))

Los archivos están nombrados según la nomenclatura establecida:

- **GarzonMaria\_Mayor.asm**
- **GarzonMaria\_Menor.asm**
- **GarzonMaria\_Fibonacci.asm**