

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

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

Introducción

Este informe documenta la implementación de algoritmos en ensamblador MIPS con la herramienta MARS. El propósito fue desarrollar y ejecutar scripts en ensamblador para resolver problemas concretos, explorando el funcionamiento de bajo nivel de un procesador y la manipulación de registros y memoria.

Se crearon tres scripts:

*Un script para identificar el número mayor en una lista de valores proporcionada por el usuario.

*Un script para identificar el número menor en la misma lista de valores.

*Un script para generar la serie Fibonacci hasta un límite definido por el usuario, y calcular la suma de los números generados.

Desarrollo de la actividad

Para la ejecución de esta actividad, se utilizó el simulador MARS (MIPS Assembler and Runtime Simulator), el cual permite compilar y ejecutar programas escritos en lenguaje ensamblador MIPS. A continuación, se detallan los pasos seguidos en la implementación de cada uno de los scripts:

1. Identificación del número mayor

- Se solicitó al usuario ingresar una cantidad de números entre 3 y 5.
- Se almacenaron los valores en registros.
- Se compararon los valores mediante instrucciones de comparación y salto condicional.
- Se imprimió el número mayor en la consola.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

Capturas de pantalla

```

1  .data
2  prompt_count: .asciiz "Ingrese la cantidad de números a comparar (3-5): "
3  prompt_number: .asciiz "Ingrese un número: "
4  result_msg: .asciiz "El número mayor es: "
5  numbers: .space 20 # Espacio para almacenar los números
6
7  .text
8  .globl main
9  main:
10 # Pedir cantidad de números
11 li $v0, 4
12 la $a0, prompt_count
13 syscall
14
15 li $v0, 5
16 syscall
17 move $t0, $v0 # Guardar cantidad en $t0
18
19 li $t1, 0 # Índice
20 li $t2, -2147483648 # Inicializar con el menor valor posible (min int)
21
22 loop_input:
23 beq $t1, $t0, find_max # Si ya ingresó todos los números, ir a buscar el mayor

```

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu \$2,\$0,0x00000004	11: li \$v0, 4
	0x00400004	0x3c011001	lui \$1,0x00001001	12: la \$a0, prompt_count
	0x00400008	0x34240000	ori \$4,\$1,0x00000000	
	0x0040000c	0x0000000c	syscall	13: syscall
	0x00400010	0x24020005	addiu \$2,\$0,0x00000005	15: li \$v0, 5
	0x00400014	0x0000000c	syscall	16: syscall
	0x00400018	0x00024021	addu \$8,\$0,\$2	17: move \$t0, \$v0 # Guardar cantidad en \$t0
	0x0040001c	0x24090000	addiu \$9,\$0,0x00000000	19: li \$t1, 0 # Índice
	0x00400020	0x3c018000	lui \$1,0xffff8000	20: li \$t2, -2147483648 # Inicializar con el menor valor posible (min int)
	0x00400024	0x342a0000	ori \$10,\$1,0x00000000	

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x72676e49	0x20657365	0x6320616c	0x69746e61	0x20646164	0x6e206564	0x72656d66	0x6120736f
0x10010020	0x6d6f6320	0x61726170	0x33282072	0x3a29352d	0x6e490020	0x73657267	0x6e752065	0x6d6f6e20
0x10010040	0x3a6f7265	0x6c450020	0x6d6f6e20	0x206f7265	0x6f79616d	0x73652072	0x0000203a	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

```

Ingrese la cantidad de números a comparar (3-5): 5
Ingrese un número: 4
Ingrese un número: 5
Ingrese un número: 9
Ingrese un número: 15
Ingrese un número: 16
El número mayor es: 16
-- program is finished running --

```

2. Identificación del número menor

- Se solicitó al usuario ingresar una cantidad de números entre 3 y 5.
- Se compararon los valores almacenados en registros.
- Se imprimió el número menor en la consola.

Capturas de pantalla

```

Edit Execute
Fibonacci_Santiago_Bohorquez.asm* BohorquezSantiago_Mayor.asm BohorquezSantiago_Menor.asm
7  .text
8  .globl main
9  main:
10     # Pedir cantidad de números
11     li $v0, 4
12     la $a0, prompt_count
13     syscall
14
15     li $v0, 5
16     syscall
17     move $t0, $v0 # Guardar cantidad en $t0
18
19     li $t1, 0      # Índice
20     li $t2, 2147483647 # Inicializar con el mayor valor posible (max int)
21
22 loop_input:
23     beq $t1, $t0, find_min # Si ya ingresó todos los números, ir a buscar el menor
24
25     li $v0, 4
26     la $a0, prompt_number
27     syscall
28
29     li $v0, 5
30     syscall
31     move $t3, $v0 # Guardar número en $t3
32
33     blt $t3, $t2, update_min # Si es menor, actualizar mínimo
34     j continue_input
35
36 update_min:
37     move $t2, $t3 # Guardar nuevo mínimo
38
39 continue_input:
40     addi $t1, $t1, 1
41     j loop_input

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

Text Segment	
Bkpt	Address Code Basic Source
	0x00400000 0x24020004 addiu \$2,\$0,0x00000004 11: li \$v0, 4
	0x00400004 0x3c011001 lui \$1,0x00001001 12: la \$a0, prompt_count
	0x00400008 0x34240000 ori \$4,\$1,0x00000000
	0x0040000c 0x0000000c syscall 13: syscall
	0x00400010 0x24020005 addiu \$2,\$0,0x00000005 15: li \$v0, 5
	0x00400014 0x0000000c syscall 16: syscall
	0x00400018 0x00024021 addiu \$8,\$0,\$2 17: move \$t0, \$v0 # Guardar cantidad en \$t0
	0x0040001c 0x24090000 addiu \$9,\$0,0x00000000 19: li \$t1, 0 # indice
	0x00400020 0x3c017fff lui \$1,0x00007fff 20: li \$t2, 2147483647 # Inicializar con el mayor valor posible (max int)
	0x00400024 0x342affff ori \$10,\$1,0x0000ffff
Data Segment	
Address	Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000	0x72676e49 0x20657365 0x6320616c 0x69746e61 0x20646164 0x6e206564 0x72656d66 0x6120736f
0x10010020	0x6d6f66320 0x61726170 0x33282072 0x3a29352d 0x6e490020 0x73657267 0x6e752065 0x6d6f6620
0x10010040	0x3a6f7265 0x6c450020 0x6d6f6620 0x206f7265 0x6f6e656d 0x73652072 0x0000203a 0x00000000
0x10010060	0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080	0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0	0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0	0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0	0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000

```

Ingrese la cantidad de números a comparar (3-5): 3
Ingrese un número: 5
Ingrese un número: 15
Ingrese un número: 16
El número menor es: 5
-- program is finished running --

```

3. Generación de la serie Fibonacci

- Se pidió al usuario ingresar la cantidad de términos a generar, con un límite máximo de 12.
- Se calculó la serie Fibonacci usando sumas sucesivas y almacenamiento en registros.
- Se imprimieron los valores de la serie junto con su suma total.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

Capturas de pantalla

```

Fibonacci_Santiago_Bohorquez.asm*  BohorquezSantiago_Mayor.asm  BohorquezSantiago_Menor.asm
1  .data
2  prompt: .asciiz "Ingrese la cantidad de números de la serie Fibonacci:"
3  error_msg: .asciiz "Número no válido. Intente nuevamente.\n"
4  result_msg: .asciiz "Serie Fibonacci: "
5  sum_msg: .asciiz "\nSuma total: "
6
7  .text
8  .globl main
9  main:
10     # Pedir la cantidad de números
11     ask_again:
12         li $v0, 4
13         la $a0, prompt
14         syscall
15
16         li $v0, 5
17         syscall
18         move $t0, $v0 # Guardar cantidad en $t0
19
20     # Validar que esté entre 1 y 12
21     blt $t0, 1, invalid_input
22     bgt $t0, 12, invalid_input
23     j continue_execution
24
25 invalid_input:
26     li $v0, 4
27     la $a0, error_msg
28     syscall
29     j ask_again # Volver a pedir el número
30
31 continue_execution:
32     # Mostrar mensaje de inicio
33     li $v0, 4
34     la $a0, result_msg
35     syscall

```

Text Segment		Basic		Source	
Bkpt	Address	Code			
	0x00400000	0x24020004	addiu \$2,\$0,0x00000004	12:	li \$v0, 4
	0x00400004	0x3c011001	lui \$1,0x00001001	13:	la \$a0, prompt
	0x00400008	0x34240000	ori \$4,\$1,0x00000000		
	0x0040000c	0x0000000c	syscall	14:	syscall
	0x00400010	0x24020005	addiu \$2,\$0,0x00000005	16:	li \$v0, 5
	0x00400014	0x0000000c	syscall	17:	syscall
	0x00400018	0x00024021	addu \$8,\$0,\$2	18:	move \$t0, \$v0 # Guardar cantidad en \$t0
	0x0040001c	0x29010001	slti \$1,\$8,0x00000001	21:	blt \$t0, 1, invalid_input
	0x00400020	0x14200004	bne \$1,\$0,0x00000004		
	0x00400024	0x2001000c	addi \$1,\$0,0x0000000c	22:	bgt \$t0, 12, invalid_input

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x72676e49	0x20657365	0x6320616c	0xe9746e61	0x20646164	0x6e206564	0x72656d66	0xe420736f
0x10010020	0xe16c2065	0x72657320	0x46206569	0xeef62629	0x69636361	0xfa4e0020	0x6f72656d	0x206f6e20
0x10010040	0x696ce176	0x202e6f64	0x65746e49	0x2065746e	0x7665756e	0x6e656d61	0x0a2e6574	0x72655300
0x10010060	0x46206569	0x6e6f6629	0x69636361	0x0a00203a	0x616d7553	0x746f7420	0x203a66c61	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Bohorquez Calderon	17/03/25
	Nombre: Santiago	

```

Ingrese la cantidad de números de la serie Fibonacci 5
Serie Fibonacci: 0 1 1 2 3
Suma total: 7
-- program is finished running --

```

Conclusiones

La implementación de algoritmos en ensamblador fue un ejercicio valioso para entender la arquitectura MIPS y sus instrucciones básicas. Pude apreciar la diferencia entre la programación de bajo y alto nivel, y la importancia de optimizar el código