

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

LABORATORIO #1: SIMULACIÓN Y OPTIMIZACIÓN DE UN PROGRAMA EN UN PROCESADOR ESCALAR SEGMENTADO

FUNDACION UNIVERSITARIA
INTERNACIONAL DE LA RIOJA

INGENIERÍA INFORMÁTICA

Bárbara Catalina Gómez Pérez

Profesor: Deivis Eduard Ramirez Martinez

BOGOTA, D.C.
2024

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

ÍNDICES DE CONTENIDO

1. Introducción
2. Desarrollo de la actividad.
 - 2.1 Solución script #1: Número mayor (mínimo 3 números).
 - 2.2 Solución script #2: Número menor (mínimo 3 números y máximo 5 números).
 - 2.3 Solución script #3: Serie Fibonacci.
3. Conclusiones.
4. Referencias

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

1. Introducción

En este laboratorio se desarrollaron tres scripts en lenguaje ensamblador MIPS. El objetivo principal de la práctica de laboratorio es evaluar la capacidad de interacción con el usuario a través de la consola, ejecutar cálculos matemáticos básicos y utilizar estructuras de control simples. Cada script creado cumple una función específica como comparar números hasta crear secuencias numéricas, utilizando operaciones aritméticas y lógicas elementales, así como manejo de bucles y condiciones. Este laboratorio también nos brinda la oportunidad de entender cómo opera el procesador internamente y de qué manera se llevan a cabo las instrucciones con lenguaje de programación de bajo nivel.

2. Desarrollo de la actividad.

2.1. Solución script #1: Número mayor (mínimo 3 números).

C:\Users\barba\OneDrive\Desktop\UNIR\4. Tercer semestre 2024-II\Primer módulo\Estructura de computadores\GomezBarbara_Mayor.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

GomezBarbara_Mayor.asm GomezBarbara_Menor.asm GomezBarbara_Fibonacci.asm

```
1 # Programa para encontrar el número mayor de una lista entre 3 y 5 números, el programa evalúa el rango de entrada digitado
2
3 .data
4 prompt_num: .asciiz "¿Cuántos números quiere comparar? (min 3, max 5): "
5 invalid_msg: .asciiz "Error: Ingrese un número entre 3 y 5.\n"
6 input_prompt: .asciiz "Ingrese un número: "
7 result_msg: .asciiz "El número mayor es: "
8
9 .text
10 .globl main
11
12 main:
13     # Preguntar al usuario cuántos números va a ingresar
14     li $v0, 4          # syscall para imprimir string
15     la $a0, prompt_num # cargar la dirección del mensaje
16     syscall
17
18     li $v0, 5          # syscall para leer un entero
19     syscall
20     move $t0, $v0      # almacenar el número ingresado en $t0
21
22     # Evaluar que el número ingresado esté entre 3 y 5
23     li $t1, 3          # valor mínimo permitido
24     li $t2, 5          # valor máximo permitido
25     blt $t0, $t1, error # si el número es menor que 3, ir a error
26     bgt $t0, $t2, error # si el número es mayor que 5, ir a error
27
28     # Se inicializa el contador y el valor mayor
29     li $t3, 0          # contador de números
30     li $t4, -2147483648 # valor inicial del número mayor (mínimo entero)
31
32 input_loop:
33     # Se solicita al usuario ingresar un número
34     li $v0, 4          # syscall para imprimir
```

Line: 6 Column: 47 Show Line Numbers

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$s9	27	0x00000000
\$gp	28	0x10000000
\$fp	29	0x7ffff000
\$sp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400000
\$hi		0x00000000
\$lo		0x00000000

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

C:\Users\barba\OneDrive\Desktop\UNIR\4. Tercer semestre 2024-II\Primer módulo\Estructura de computadores\GomezBarbara_Mayor.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
0x00400000	0x24020004	addiu \$2,\$0,0x00000...	14: li \$v0, 4	# syscall para imprimir string
0x00400004	0x3c011001	lui \$1,0x00001001	15: la \$a0, prompt_num	# cargar la dirección del mensaje
0x00400008	0x34240000	ori \$4,\$1,0x00000000		
0x0040000c	0x0000000c	syscall	16: syscall	
0x00400010	0x24020005	addiu \$2,\$0,0x00000...	18: li \$v0, 5	# syscall para leer un entero
0x00400014	0x0000000c	syscall	19: syscall	
0x00400018	0x00024021	addu \$8,\$0,\$2	20: move \$t0, \$v0	# almacenar el número ingresado en \$t0
0x0040001c	0x24050003	addiu \$9,\$0,0x00000...	23: li \$t1, 3	# valor mínimo permitido
0x00400020	0x24050005	addiu \$10,\$0,0x00000...	24: li \$t2, 5	# valor máximo permitido
0x00400024	0x0105082a	slt \$1,\$8,\$9	25: blt \$t0, \$t1, error	# si el número es menor que 3, ir a error
0x00400028	0x1420001c	bne \$1,\$0,0x0000001c		
0x0040002c	0x0148082a	slt \$1,\$10,\$8	26: bgt \$t0, \$t2, error	# si el número es mayor que 5, ir a error

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xe17543bf	0x736f746e	0x6dfa6e20	0x736f7265	0x69757120	0x20657265	0x706d6f63	0x72617261
0x10010020	0x6d28203f	0x33206e69	0x616d202c	0x29352078	0x4500203a	0x726f7272	0x6e49203a	0x73657267
0x10010040	0x6e752065	0x6dfa6e20	0x206f7265	0x72746e65	0x20332065	0x2e352079	0x6e49000a	0x73657267
0x10010060	0x6e752065	0x6dfa6e20	0x3a6f7265	0x6c450020	0x6dfa6e20	0x206f7265	0x6f79616d	0x73652072
0x10010080	0x00000203a	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
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Mars Messages Run I/O

Edit Execute

Primer_ejercicio GomezBarbara_Mayor.asm GomezBarbara_Menor.asm GomezBarbara_Fibonacci.asm

1 # Programa para encontrar el número mayor de una lista entre 3 y 5 números, el programa evalúa el rango de entrada digitado

2

3 .data

4 prompt_num: .asciiz "¿Cuántos números quiere comparar? (min 3, max 5): "

5 invalid_msg: .asciiz "Error: Ingrese un número entre 3 y 5.\n"

6 input_prompt: .asciiz "Ingrese un número: "

7 result_msg: .asciiz "El número mayor es: "

8

9 .text

10 .globl main

11

12 main:

13 # Preguntar al usuario cuántos números va a ingresar

14 li \$v0, 4 # syscall para imprimir string

15 la \$a0, prompt_num # cargar la dirección del mensaje

16 syscall

17

18 li \$v0, 5 # syscall para leer un entero

19 syscall

20 move \$t0, \$v0 # almacenar el número ingresado en \$t0

21

22 # Evaluar que el número ingresado esté entre 3 y 5

23 li \$t1, 3 # valor mínimo permitido

24 li \$t2, 5 # valor máximo permitido

25 blt \$t0, \$t1, error # si el número es menor que 3, ir a error

26 bgt \$t0, \$t2, error # si el número es mayor que 5, ir a error

Line: 6 Column: 47 Show Line Numbers

Mars Messages Run I/O

Clear

¿Cuántos números quiere comparar? (min 3, max 5): 4

Ingrese un número: 56

Ingrese un número: 89

Ingrese un número: 20

Ingrese un número: 98

El número mayor es: 98

-- program is finished running --

© Universidad Internacional de La Rioja (UNIR)

Tema 5. Actividades

4

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

El primer script le solicita al usuario que digite entre 3 y 5 números, con el fin de encontrar el número mayor ingresado. El programa inicia preguntando al usuario que indique cuántos números quiere comparar. Si el número ingresado está fuera del rango permitido (menos de 3 o más de 5 números), el programa muestra un mensaje de error y termina (Peralta, 2014).

The screenshot shows a window titled 'Mars Messages' with a 'Run I/O' button. The output text is as follows:

```

¿Cuántos números quiere comparar? (min 3, max 5): 2
Error: Ingrese un número entre 3 y 5.

-- program is finished running --

¿Cuántos números quiere comparar? (min 3, max 5): 6
Error: Ingrese un número entre 3 y 5.

```

Ejemplo mensaje de error cuando se digita un número fuera del rango permitido.

Si la cantidad ingresada es válida, el usuario podrá ingresar los números uno por uno, de esta manera el programa comparará cada entrada con el número mayor almacenado. Una vez se han ingresado todos los números, el programa mostrará el mayor en la consola.

Enlace programa Git Hub:

https://github.com/bcgomez/Lab1_EstructuraComputadores/blob/main/GomezBarbara_Mayor.asm

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

2.2. Solución script #2: Número menor (mínimo 3 números y máximo 5 números).

```

# Programa para encontrar el número menor de una lista entre 3 y 5 números, el programa evalúa el rango de entrada digitado

.data
prompt_num: .ascii "¿Cuántos números quiere comparar? (min 3, max 5): "
invalid_msg: .ascii "Error: Ingrese un número entre 3 y 5.\n"
input_prompt: .ascii "Ingrese un número: "
result_msg: .ascii "El número menor es: "

.text
.globl main

main:
    # Preguntar al usuario cuántos números va a ingresar
    li $v0, 4          # syscall para imprimir string
    la $a0, prompt_num # cargar la dirección del mensaje
    syscall

    # Evaluar que el número ingresado esté entre 3 y 5
    li $t1, 3          # valor mínimo permitido
    li $t2, 5          # valor máximo permitido
    blt $t0, $t1, error # si el número es menor que 3, es a error
    bgt $t0, $t2, error # si el número es mayor que 5, es a error

    # Inicializar el contador y el valor menor
    li $t3, 0          # contador de números
    li $t4, 2147483647 # valor inicial del número menor (máximo entero)

input_loop:
    # Se solicita al usuario ingresar un número
    li $v0, 4          # syscall para imprimir string
    la $a0, input_prompt
    syscall

    # Leer el número ingresado
    li $v0, 5          # syscall para leer un entero
    syscall
    move $t0, $v0      # almacenar el número ingresado en $t0

    # Comparar el número ingresado con el menor almacenado
    li $t5, $t4
    blt $t0, $t5, update_min
    j input_loop

update_min:
    li $t4, $t0
    li $t3, $t3 + 1
    j input_loop

error:
    # Imprimir mensaje de error
    li $v0, 4          # syscall para imprimir string
    la $a0, invalid_msg
    syscall

    # Salir del programa
    li $v0, 10         # syscall para salir
    syscall
  
```

El segundo script le solicita al usuario que ingrese entre 3 y 5 números, con el propósito de encontrar el número menor. El desarrollo del programa es muy parecido al anterior script, porque primero se evalúa que el número de entradas esté dentro del rango (entre 3 y 5 números), luego, se le solicita al usuario que ingrese los números uno por uno. A medida que se ingresan los números, cada número se compara con el menor almacenado. Cuando se terminan de ingresar todos los números, el programa imprime el menor.

```

Mars Messages
Run I/O

¿Cuántos números quiere comparar? (min 3, max 5): 2
Error: Ingrese un número entre 3 y 5.

-- program is finished running --

¿Cuántos números quiere comparar? (min 3, max 5): 6
Error: Ingrese un número entre 3 y 5.
  
```

Ejemplo mensaje de error cuando se digita un número fuera del rango permitido.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

C:\Users\barba\OneDrive\Desktop\UNIR\4. Tercer semestre 2024-II\Primer módulo\Estructura de computadores\GomezBarbara_Menor.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$a6	22	0x00000000
\$a7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu \$2,\$0,0x0000...	14: li \$v0, 4 # syscall para imprimir string
	0x00400004	0x3c011001	lui \$1,0x00001001	15: la \$a0, prompt_num # cargar la dirección del mensaje
	0x00400008	0x34240000	ori \$4,\$1,0x00000000	
	0x0040000c	0x0000000c	syscall	16: syscall
	0x00400010	0x24020005	addiu \$2,\$0,0x0000...	18: li \$v0, 5 # syscall para leer un entero
	0x00400014	0x0000000c	syscall	19: syscall
	0x00400018	0x00024021	addiu \$8,\$0,\$2	20: move \$t0, \$v0 # almacenar el número ingresado en \$t0
	0x0040001c	0x24090003	addiu \$9,\$0,0x0000...	23: li \$t1, 3 # valor mínimo permitido
	0x00400020	0x240a0005	addiu \$10,\$0,0x0000...	24: li \$t2, 5 # valor máximo permitido
	0x00400024	0x0109002a	blt \$1,\$0,\$9	25: blt \$t0, \$t1, error # si el número es menor que 3, ir a error
	0x00400028	0x1420001c	hne \$1,\$0,0x0000001c	
	0x0040002c	0x0148002a	blt \$1,\$10,\$8	26: bgt \$t0, \$t2, error # si el número es mayor que 5, ir a error
	0x00400030	0x1420001a	hne \$1,\$0,0x0000001a	
	0x00400034	0x240b0000	addiu \$11,\$0,0x0000...	29: li \$t3, 0 # contador de números

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xe17543bf	0x736f746e	0x6dfa6e20	0x736f7265	0x69757120	0x20657265	0x70646663	0x72617261
0x10010020	0xc228203f	0x33206e69	0x6164202c	0x25352078	0x4500203a	0x726f7272	0x6e49203a	0x73657267
0x10010040	0x6e752065	0x6dfa6e20	0x206f7265	0x72746e65	0x20332065	0x2e352079	0x6e49000a	0x73657267
0x10010060	0x6e752065	0x6dfa6e20	0x3a6f7265	0x6c450020	0x6dfa6e20	0x206f7265	0x6f6e656d	0x73652072
0x10010080	0x0000203a	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

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Primer_ejercicio GomezBarbara_Mayor.asm GomezBarbara_Menor.asm GomezBarbara_Fibonacci.asm

```
# Programa para encontrar el número menor de una lista entre 3 y 5 números, el programa evalúa el rango de entrada digitado

.data
prompt_num: .asciiz "¿Cuántos números quiere comparar? (min 3, max 5): "
invalid_msg: .asciiz "Error: Ingrese un número entre 3 y 5.\n"
input_prompt: .asciiz "Ingrese un número: "
result_msg: .asciiz "El número menor es: "

.text
.globl main

main:
    # Preguntar al usuario cuántos números va a ingresar
    li $v0, 4          # syscall para imprimir string
    la $a0, prompt_num # cargar la dirección del mensaje
    syscall

    li $v0, 5          # syscall para leer un entero
    syscall
    move $t0, $v0      # almacenar el número ingresado en $t0

    # Evaluar que el número ingresado esté entre 3 y 5
    li $t1, 3          # valor mínimo permitido
    li $t2, 5          # valor máximo permitido
    blt $t0, $t1, error # si el número es menor que 3, ir a error
    bgt $t0, $t2, error # si el número es mayor que 5, ir a error
```

Line: 33 Column: 48 Show Line Numbers

Mars Messages Run I/O

¿Cuántos números quiere comparar? (min 3, max 5): 4
Ingrese un número: 8
Ingrese un número: 5
Ingrese un número: 6
Ingrese un número: 7
El número menor es: 5
-- program is finished running --

Enlace Git Hub:
https://github.com/bcgomez/Lab1_EstructuraComputadores/blob/main/GomezBarbara_Menor.asm

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

2.3 Solución script #3: Serie Fibonacci.

```
# Programa para generar la serie Fibonacci y calcular la suma de los números

.data
prompt_fib: .asciiz "¿Cuántos números de la serie Fibonacci quiere generar? "
fib_msg: .asciiz "Los primeros números de la serie Fibonacci son: "
sum_msg: .asciiz "La suma de la serie es: "

.text
.globl main

main:
    # Se pregunta al usuario cuántos números de Fibonacci desea generar
    li $v0, 4          # syscall para imprimir string
    la $a0, prompt_fib # cargar el mensaje
    syscall

    li $v0, 5          # syscall para leer un entero
    syscall
    move $t0, $v0      # almacenar el número de elementos de Fibonacci en $t0

    # Se inicializan los primeros valores de la serie
    li $t1, 0          # primer número de la serie (0)
    li $t2, 1          # segundo número de la serie (1)
    li $t3, 0          # contador de la serie
    li $t4, 0          # acumulador para la suma

    # Se muestra el mensaje inicial
    li $v0, 4          # syscall para imprimir string
    la $a0, fib_msg    # cargar el mensaje de Fibonacci
    syscall

fib_loop:
    # Se imprime el número actual de la serie ($t1)
    li $v0, 1          # syscall para imprimir entero
    move $a0, $t1
    syscall

    # Se calcula el siguiente número de la serie
    add $t4, $t2, $t1
    li $t3, $t3 + 1
    li $t1, $t2
    li $t2, $t4
    bne $t3, $t0, fib_loop

    # Se calcula la suma final
    li $v0, 1          # syscall para imprimir entero
    move $a0, $t4
    syscall

    # Se muestra el mensaje final
    li $v0, 4          # syscall para imprimir string
    la $a0, sum_msg    # cargar el mensaje de la suma
    syscall
```

Execution completed successfully.

Assembly: assembling C:\Users\barba\OneDrive\Desktop\UNIR\4. Tercer semestre 2024-II\Primer módulo\Estructura de computadores\GomezBarbara_Fibonacci.asm

Assembly: operation completed successfully.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

The screenshot shows a MIPS assembly editor with the following code in the 'GomezBarbara_Fibonacci.asm' file:

```

1  # Programa para generar la serie Fibonacci y calcular la suma de los números
2
3  .data
4  prompt_fib:  .asciiz "%Cuántos números de la serie Fibonacci quiere generar?: "
5  fibo_msg:    .asciiz "Los primeros números de la serie Fibonacci son: "
6  sum_msg:     .asciiz "La suma de la serie es: "
7
8  .text
9  .globl main
10
11 main:
12  # Se pregunta al usuario cuántos números de Fibonacci desea generar
13  li $v0, 4      # syscall para imprimir string
14  la $a0, prompt_fib  # cargar el mensaje
15  syscall
16
17  li $v0, 5      # syscall para leer un entero
18  syscall
19  move $t0, $v0  # almacenar el número de elementos de Fibonacci en $t0
20
21  # Se inicializan los primeros valores de la serie
22  li $t1, 0      # primer número de la serie (0)
23  li $t2, 1      # segundo número de la serie (1)
24  li $t3, 0      # contador de la serie
25  li $t4, 0      # acumulador para la suma
26

```

The execution output in the 'Run I/O' window shows:

```

¿Cuántos números de la serie Fibonacci quiere generar?: 5
Los primeros números de la serie Fibonacci son: 01123La suma de la serie es: 7
-- program is finished running --

```

Enlace programa GitHub:

https://github.com/bcggomez/Lab1_EstructuraComputadores/blob/main/GomezBarbara_Fibonacci.asm

En este script, el programa genera la serie Fibonacci hasta un número de términos declarado por el usuario. Así como en los anteriores scripts, se evalúa que el número ingresado por el usuario esté dentro del rango (al menos 1 término, para esta serie). Una vez digitados los números de la serie, se genera los números de la serie Fibonacci a través de una suma sencilla de los dos términos anteriores y los va mostrando en la consola a medida que se calculan.

Fórmula empleada para calcular la serie de Fibonacci:

$$f_n = f_{n-1} + f_{n-2}$$

(Escobar, 2020)

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

Al final, el programa también calcula la suma de todos los números de la serie y muestra el resultado.

3. Conclusiones.

- En el desarrollo del laboratorio logré comprender las bases del lenguaje ensamblador MIPS, lo cual me permitió interactuar con el hardware a través de las instrucciones y comandos de este lenguaje de bajo nivel. De esta forma, el programa MARS me facilitó observar cómo un procesador ejecuta acciones sencillas como comparaciones y cálculos matemáticos entre números.
- En todos los scripts creados se pusieron en práctica validaciones para verificar que se cumplieran los límites de los rangos en la entrada de datos dada por los usuarios. Por medio de estas evaluaciones, se aseguró que en los dos primeros programas no de aceptara valores fuera del rango, imprimiendo el mensaje de error: "Error: ingrese un número entre 3 y 5".
- La interacción de la consola con el usuario se fortalece a través del uso de syscalls para la entrada y salida de datos (Vollmar, 2006). Es importante resaltar el uso de este comando, ya que es decisivo para la creación de programas interactivos y entendibles para el usuario.
- Se demostró eficiencia en la ejecución de operaciones secuenciales, como en la implementación de bucles en la serie Fibonacci para acciones reiterativas. Asimismo, se utilizaron estructuras de control simples para evaluar la entrada de números y su comparación, mejorando así el rendimiento del programa.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Gómez Pérez	7 de septiembre de 2024
	Nombre: Bárbara Catalina	

4. Referencias

- Escobar, E. V. (17 de Octubre de 2020). *Diagrama de flujo: Serie Fibonacci 1, 1, 2, 3, 5, 8, 13, 21,.* Obtenido de <https://www.youtube.com/watch?app=desktop&v=xRna7mcBNLw>
- Peralta, A. (26 de Diciembre de 2014). *MIPS Tutorial 24 Checking If a Number is Less than Another slt.* Obtenido de <https://www.youtube.com/watch?v=WF8jzQY0bh0&list=PL5b07qImA3P6zUdDf-o97ddfpvPFuNa5A&index=24>
- Vollmar, K. (1 de Marzo de 2006). *MARS: An Education-Oriented MIPS Assembly Language Simulator.* Obtenido de [file:///C:/Users/barba/Downloads/MarsAnEducation-Oriented_MIPS_AssemblyLanguaje%20\(1\).pdf](file:///C:/Users/barba/Downloads/MarsAnEducation-Oriented_MIPS_AssemblyLanguaje%20(1).pdf)