

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

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

### Entrega del laboratorio

#### 1. Encontrar el NUMERO MAYOR:

Comparar números (mínimo 3 y máximo 5 números a comparar) y posteriormente, por consola mostrar un mensaje que indique cuál es el mayor de los números digitados.

##### a. Antes de compilar

Inicialmente, se definen los mensajes en **.data**. En este ejercicio se definen 4 mensajes: (1) Para ingresar la cantidad, (2) Validador que la cantidad este entre 3 y 5, (3) Ingresar un número y (4) El resultado del mayor número ingresado.

```

1  .data
2      msgCantidad: .asciiz "Ingrese la cantidad de numeros a comparar (min 3 - max 5): "
3      msgInvalido: .asciiz "Cantidad invalida. Debe ser entre 3 y 5.\n"
4      msgNumero:   .asciiz "Ingrese un numero: "
5      msgResultado:.asciiz "El mayor numero ingresado es: "
6

```

Seguidamente, en la parte **.text** se definen las instrucciones que el procesador va a ejecutar. En este caso, el proceso principal inicia, imprimiendo (*li \$v0, 4*) el mensaje de cantidad ("Ingrese la cantidad de número a comparar..."). Luego, se lee el número ingresado (*li \$v0, 5*) y se pasan los datos de \$v0 a \$t0; adicionalmente se debe validar el rango utilizando los comandos **blt** y **bgt**. El comando **blt**, condiciona el proceso **invalid** a ejecutarse cuando se cumpla la condición de ingresa un número menor que 3. El comando **bgt** condiciona las instrucciones del proceso principal a ejecutar el proceso **invalid** cuando se cumpla la condición de ingresar un número mayor que 5.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

```

7  .text
8  .globl main
9
10  main:
11  # Pedir cantidad de números
12  li $v0, 4
13  la $a0, msgCantidad
14  syscall
15
16  li $v0, 5      # leer entero
17  syscall
18  move $t0, $v0  # $t0 = cantidad
19
20  # Validar rango (3 <= n <= 5)
21  blt $t0, 3, invalid
22  bgt $t0, 5, invalid
23

```

El proceso principal continua, luego de preguntar la cantidad de los números a comparar, solicitando ingresar el primer número (**msgNumero**) que se leerá (**li \$v0, 5**) , luego se moverá a \$t1 y se actualizará el contador que se encuentra en \$t2.

```

24  # Ingreso del primer número
25  li $v0, 4
26  la $a0, msgNumero
27  syscall
28
29  # Leer el número
30  li $v0, 5
31  syscall
32  move $t1, $v0  # $t1 = primer número (asumimos que es el mayor inicial)
33
34  li $t2, 1      # contador = 1 (ya tenemos 1 numero)
35

```

En el renglón 37 se agregó una condición para ejecutar el proceso mostrar el resultado, si, se cumple que \$t0 (cantidad de números a ingresar) == \$t2 (contador). De no cumplirse la condición anterior, se procede a leer un nuevo número que se guardará en \$t3.

En el renglón 48, se da la instrucción de cambiar el número ubicado en \$t1 cada que \$t3 sea mayor al número comparado.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

```

36 leer_numeros:
37     beq $t2, $t0, mostrar    # si contador == cantidad → mostrar resultado
38
39     # Pedir otro numero
40     li $v0, 4
41     la $a0, msgNumero
42     syscall
43
44     li $v0, 5
45     syscall
46     move $t3, $v0    # $t3 = número actual
47
48     # Comparar si es mayor
49     ble $t3, $t1, no_cambio
50     move $t1, $t3    # actualizar mayor
51

```

El proceso de **no cambio** actualiza el contador en \$t2 y envía el flujo de nuevo a leer números

```

52 no_cambio:
53     addi $t2, $t2, 1
54     j leer_numeros
55

```

El proceso **mostrar** imprime el mensaje resultado (“El mayor número ingresado es:”) e imprime el entero (li \$v0, 1), que corresponde a ese número, para luego finalizar el programa (li \$v0, 10).

```

56 mostrar:
57     # Imprimir resultado
58     li $v0, 4
59     la $a0, msgResultado
60     syscall
61
62     li $v0, 1
63     move $a0, $t1
64     syscall
65
66     li $v0, 10    # salir
67     syscall
68

```

El proceso **invalid**, imprime el mensaje de “Cantidad invalida...”, cuando los números por comparar no se encuentran entre 3 y 5, luego vuelve a saltar al proceso principal.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

```

69  invalid:
70      li $v0, 4
71      la $a0, msgInvalido
72      syscall
73      j main

```

## b. Después de compilar

EditExecute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	4194304	0x24020004	addiu \$2,\$0,4	12: li \$v0, 4
<input type="checkbox"/>	4194308	0x3c011001	lui \$1,4097	13: la \$a0, msgCantidad
<input type="checkbox"/>	4194312	0x34240000	ori \$4,\$1,0	
<input type="checkbox"/>	4194316	0x0000000c	syscall	14: syscall
<input type="checkbox"/>	4194320	0x24020005	addiu \$2,\$0,5	16: li \$v0, 5 # leer entero
<input type="checkbox"/>	4194324	0x0000000c	syscall	17: syscall
<input type="checkbox"/>	4194328	0x00024021	addu \$8,\$0,\$2	18: move \$t0, \$v0 # \$t0 = cantidad
<input type="checkbox"/>	4194332	0x29010003	slti \$1,\$8,3	21: blt \$t0, 3, invalid
<input type="checkbox"/>	4194336	0x14200021	bne \$1,\$0,33	
<input type="checkbox"/>	4194340	0x20010005	addi \$1,\$0,5	22: bgt \$t0, 5, invalid

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1919381065	543519589	1663066476	1769238113	543449444	1847616868	1919249781	1629516655
268501024	1836016416	1634886000	1831346290	857763433	1830825248	891320417	2112041	1953390915
268501056	1684104297	1986947360	1684630625	1142959713	543515237	544367987	1920233061	540221541
268501088	775233657	1850277898	1936028263	1853169765	1836412448	980382309	1816461344	2036428064
268501120	1847620207	1919249781	1852383343	1936028263	544171105	540701541	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

0x10010000 (.data)

☐ Hexadecimal Addresses
☐ Hexadecimal Values
☐ ASCII

Mars MessagesRun I/O

Clear

Assemble: operation completed successfully.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

### c. Después de ejecutar

```

Mars Messages  Run I/O
Ingrese la cantidad de numeros a comparar (min 3 - max 5): 4
Ingrese un numero: 23
Ingrese un numero: 67
Ingrese un numero: 5
Ingrese un numero: 17
El mayor numero ingresado es: 67
-- program is finished running --

```

Registers	Coproc 1	Coproc 0	
Name	Number	Value	
\$zero	0	0	
\$at	1	268500992	
\$v0	2	10	
\$v1	3	0	
\$a0	4	67	
\$a1	5	0	
\$a2	6	0	
\$a3	7	0	
\$t0	8	4	
\$t1	9	67	
\$t2	10	4	
\$t3	11	17	
\$t4	12	0	
\$t5	13	0	

### d. Conclusiones

- **\$a0** y **\$t1** terminan con el mismo valor, porque el mayor valor se mueve a \$a0.
- **\$t3** finaliza en 17 porque fue el último número ingresado.
- **\$t1** finaliza en 67, debido a que en este registro siempre se guarda el mayor valor de los ingresados.
- **\$t0** (cantidad números) y **\$t2** (el contador) finaliza en 4 igual validando la igualdad que todos los números ingresados se leyeron y disparando el proceso de **leer\_numeros**.
- Para imprimir se utilizaron los comandos **li \$vo, 1** (Imprimir enteros), **li \$vo, 4** (Imprimir texto) y **li \$vo, 10** (Finalizar programa)
- Para leer el resultado del número menor se utilizó los comandos **li \$vo, 5**.
- **\$v0** finaliza en 10 indicando que el programa se terminó de ejecutar.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

## 2. Encontrar el NUMERO MENOR

Comparar números (mínimo 3 y máximo 5 números a comparar) y posteriormente, por consola mostrar un mensaje que indique cuál es el menor de los números digitados.

### a. Antes de compilar

Inicialmente, se definen los mensajes en **.data**. En este ejercicio se definen 4 mensajes: (1) Para ingresar la cantidad, (2) Validador que la cantidad este entre 3 y 5, (3) Ingresar un número y (4) El resultado del menor número ingresado.

```

1  .data
2      msgCantidad: .asciiz "Ingrese la cantidad de numeros a comparar (min 3 - max 5): "
3      msgInvalido: .asciiz "Cantidad invalida. Debe ser entre 3 y 5.\n"
4      msgNumero:   .asciiz "Ingrese un numero: "
5      msgResultado:.asciiz "El menor numero ingresado es: "
6

```

Seguidamente, en la parte **.text** se definen las instrucciones que el procesador va a ejecutar. En este caso, el proceso principal inicia, imprimiendo (*li \$v0, 4*) el mensaje de cantidad ("Ingrese la cantidad de número a comparar..."). Luego, se lee el número ingresado (*li \$v0, 5*) y se pasan los datos de \$v0 a \$t0; adicionalmente se debe validar el rango utilizando los comandos **blt** y **bgt**. El comando **blt**, condiciona el proceso **invalid** a ejecutarse cuando se cumpla la condición de ingresa un número menor que 3. El comando **bgt** condiciona las instrucciones del proceso principal a ejecutar el proceso **invalid** cuando se cumpla la condición de ingresar un número mayor que 5.

```

7  .text
8  .globl main
9
10     main:
11     # Pedir cantidad de números
12     li $v0, 4
13     la $a0, msgCantidad
14     syscall
15
16     li $v0, 5      # leer entero
17     syscall
18     move $t0, $v0  # $t0 = cantidad
19
20     # Validar rango (3 <= n <= 5)
21     blt $t0, 3, invalid
22     bgt $t0, 5, invalid
23

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

El proceso principal continua, luego de preguntar la cantidad de los números a comparar, solicitando ingresar el primer número (**msgNumero**) que se leerá (*li \$v0, 5*) , luego se moverá a \$t1 y se actualizará el contador que se encuentra en \$t2.

```

24      # Ingreso del primer número
25      li $v0, 4
26      la $a0, msgNumero
27      syscall
28
29      # Leer el número
30      li $v0, 5
31      syscall
32      move $t1, $v0    # $t1 = primer número (asumimos que es el menor inicial)
33
34      li $t2, 1        # contador = 1 (ya tenemos 1 numero)
35

```

En el renglón 37 se agregó una condición para ejecutar el proceso mostrar el resultado, si, se cumple que \$t0 (cantidad de números a ingresar) == \$t2 (contador). De no cumplirse la condición anterior, se procede a leer un nuevo número que se guardará en \$t3.

En el renglón 48, se da la instrucción de cambiar el número ubicado en \$t1 cada que \$t3 sea menor al número comparado (bge).

```

36 leer_numeros:
37     beq $t2, $t0, mostrar    # si contador == cantidad → mostrar resultado
38
39     # Pedir otro numero
40     li $v0, 4
41     la $a0, msgNumero
42     syscall
43
44     li $v0, 5
45     syscall
46     move $t3, $v0    # $t3 = número actual
47
48     # Comparar si es menor
49     bge $t3, $t1, no_cambio
50     move $t1, $t3    # actualizar menor
51

```

El proceso de **no cambio** actualiza el contador en \$t2 y envía el flujo de nuevo a leer números

```

52 no_cambio:
53     addi $t2, $t2, 1
54     j leer_numeros
55

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

El proceso **mostrar** imprime el mensaje resultado (“El mayor número ingresado es:”) e imprime el entero (li \$v0, 1), que corresponde a ese número, para luego finalizar el programa (li \$v0, 10).

```

56  mostrar:
57      # Imprimir resultado
58      li $v0, 4
59      la $a0, msgResultado
60      syscall
61
62      li $v0, 1
63      move $a0, $t1
64      syscall
65
66      li $v0, 10      # salir
67      syscall
68

```

El proceso **invalid**, imprime el mensaje de “Cantidad invalida...”, cuando los números por comparar no se encuentran entre 3 y 5, luego vuelve a saltar al proceso principal.

```

69  invalid:
70      li $v0, 4
71      la $a0, msgInvalido
72      syscall
73      j  main

```

## b. Después de compilar

Text Segment									
Bkpt	Address	Code	Basic	Source					
	4194304	0x24020004	addiu \$2,\$0,4	12: li \$v0, 4					
	4194308	0x3c011001	lui \$1,4097	13: la \$a0, msgCantidad					
	4194312	0x34240000	ori \$4,\$1,0						
	4194316	0x0000000c	syscall	14: syscall					
	4194320	0x24020005	addiu \$2,\$0,5	16: li \$v0, 5 # leer entero					
	4194324	0x0000000c	syscall	17: syscall					
	4194328	0x00024021	addiu \$9,\$0,\$2	18: move \$t0, \$v0 # \$t0 = cantidad					
	4194332	0x29010003	slti \$1,\$8,3	21: blt \$t0, 3, invalid					
	4194336	0x14200021	bne \$1,\$0,33						
	4194340	0x20010005	addi \$1,\$0,5	22: bgt \$t0, 5, invalid					

  

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)	
269500992	1919381065	543519589	1663066476	1769238113	543449444	1847616868	1919249781	1629516655	
269501024	1836016416	1634886000	1831346290	857763433	1830825248	891320417	2112041	1953390915	
269501056	1684104297	1986947360	1684630625	1142959713	543515237	544367987	1920233061	540221541	
269501088	775233657	1850277898	1936028263	1853169765	1836412448	980382309	1816461344	1852140832	
269501120	1847620207	1919249781	1852383343	1936028263	544171105	540701541	0	0	
269501152	0	0	0	0	0	0	0	0	
269501184	0	0	0	0	0	0	0	0	
269501216	0	0	0	0	0	0	0	0	
269501248	0	0	0	0	0	0	0	0	
269501280	0	0	0	0	0	0	0	0	
269501312	0	0	0	0	0	0	0	0	

```

Assemble: assembling C:\Users\Lenovo\Documents\Personal\Estudios\Pregrado Ing. Informática\Semestre 6\Estructura de Computadores\MI
Assemble: operation completed successfully.

```



Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

### c. Después de ejecutar

```

Mars Messages Run I/O
Ingrese la cantidad de numeros a comparar (min 3 - max 5): 3
Ingrese un numero: 45
Ingrese un numero: 17
Ingrese un numero: 12
El menor numero ingresado es: 12
-- program is finished running --

```

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	12
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	3
\$t1	9	12
\$t2	10	3
\$t3	11	12
\$t4	12	0
\$t5	13	0
\$t6	14	0

### d. Conclusiones

- **\$a0** y **\$t1** terminan con el mismo valor, porque el menor valor se mueve a **\$a0**.
- **\$t3** finaliza en 12 porque fue el último número ingresado, que coincide con ser el menor.
- **\$t1** finaliza en 12, debido a que en este registro siempre se guarda el menor valor de los ingresados.
- **\$t0** (cantidad números) y **\$t2** (el contador) finaliza en 3 igual validando la igualdad que todos los números ingresados se leyeron y disparando el proceso de **leer\_numeros**.
- Para imprimir se utilizaron los comandos **li \$vo, 1** (Imprimir enteros), **li \$vo, 4** (Imprimir texto) y **li \$vo, 10** (Finalizar programa)
- Para leer el resultado del número menor se utilizó los comandos **li \$vo, 5**.
- **\$v0** finaliza en 10 indicando que el programa se terminó de ejecutar.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

### 3. Suma NUMEROS FIBONACCI

Pedir al usuario por consola cuántos número de la serie Fibonacci desea generar, posteriormente, y con base en el número introducido generé la serie y muestre por consola el resultado.

Inicialmente, se definen los mensajes en **.data**. En este ejercicio se definen 4 mensajes: (1) Para ingresar la cantidad, (2) Validador que la cantidad este entre 3 y 5, (3) Ingresar un número y (4) El resultado del menor número ingresado.

#### a. Antes de compilar

Inicialmente, se definen los mensajes en **.data**. En este ejercicio se definen 4 mensajes: (1) Para ingresar la cantidad, (2) Validador que la cantidad este entre 3 y 5, (3) Ingresar un número y (4) El resultado del menor número ingresado

```

1  .data
2      msgCantidad: .asciiz "Ingrese la cantidad de numeros de la serie Fibonacci a generar: "
3      msgSerie:    .asciiz "La serie Fibonacci es: "
4      msgSuma:     .asciiz "\nLa suma de los numeros de la serie es: "
5

```

Seguidamente, en la parte **.text** se definen las instrucciones que el procesador va a ejecutar. En este caso, el proceso principal inicia, imprimiendo (*li \$v0, 4*) el mensaje de cantidad (" Ingrese la cantidad de número a comparar..."). Luego, se lee el número ingresado (*li \$v0, 5*) y se pasan los datos de \$v0 a \$t0

```

6  .text
7  .globl main
8
9  main:
10     # Pedir cantidad de numeros
11     li $v0, 4
12     la $a0, msgCantidad
13     syscall
14
15     li $v0, 5      # leer entero
16     syscall
17     move $t0, $v0  # $t0 = cantidad de terminos
18
19     # Inicializar variables
20     li $t1, 0      # primer numero = 0
21     li $t2, 1      # segundo numero = 1
22     li $t3, 0      # suma acumulada
23     li $t4, 0      # contador
24

```

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

Una vez, se leen la cantidad de números de la serie Fibonacci, se inician las variables temporales \$t1 (1er número), \$t2 (2do número), \$t3 (suma acumulada) y \$t4 (contador).

Los renglones 26 y 27 permiten imprimir el mensaje de la serie Fibonacci, que luego se va a completar a través del bucle Fibonacci.

```

25      # Imprimir mensaje serie
26      li $v0, 4
27      la $a0, msgSerie
28      syscall
29

```

El bucle Fibonacci va a mostrar la suma cuando la cantidad de números Fibonacci y el contador sean iguales. Inicia imprimiendo el 1er número y moviéndolo a \$a0. Luego, se configuran las comas y espacios del mensaje Fibonacci, agregando la condición **no\_coma** para que cuando las cantidades Fibonacci igualen al contador no agregue más comas. El comando **li \$v0, 11**, permite imprimir los caracteres (',' y ' ').

```

30 fibonacci_loop:
31     beq $t4, $t0, mostrar_suma    # si contador == cantidad -> terminar
32
33     # Imprimir numero actual (t1)
34     li $v0, 1
35     move $a0, $t1
36     syscall
37
38     # Imprimir una coma y espacio si no es el ultimo
39     addi $t4, $t4, 1
40     beq $t4, $t0, no_coma        # si ya es el ultimo, no poner coma
41     li $v0, 11
42     li $a0, ','
43     syscall
44     li $v0, 11
45     li $a0, ' '
46     syscall

```

El proceso **no\_coma**, primero suma en el registro \$t3 (suma acumulada), la información de las variables iniciales de \$t3 (0) y \$t1 (0). Luego, las instrucciones van a calcular la suma de \$t1 más \$t2 y se va guardar en el registro \$t5. El proceso **no\_coma** finaliza llevando a \$t1 la información de \$t2, llevando a \$t2 la información de \$t5 y haciendo un salto al bucle Fibonacci.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

```

48 no_coma:
49     # Acumular suma
50     add $t3, $t3, $t1
51
52     # Calcular siguiente numero
53     add $t5, $t1, $t2 # t5 = t1 + t2
54     move $t1, $t2     # actualizar: t1 = t2
55     move $t2, $t5     # actualizar: t2 = t5
56
57     j fibonacci_loop
58

```

El proceso *mostrar\_suma*, se ejecuta cuando en el bucle Fibonacci la cantidad de números Fibonacci y el contador sean iguales, entonces imprime el mensaje de la suma y luego imprime el número que corresponde a la suma acumulada, que se encuentra en \$t3, moviéndolo a \$a0. La línea 72 finaliza la ejecución del programa.

```

59 mostrar_suma:
60     # Imprimir mensaje suma
61     li $v0, 4
62     la $a0, msgSuma
63     syscall
64
65     # Imprimir resultado suma
66     li $v0, 1
67     move $a0, $t3
68     syscall
69
70     # Finalizar
71     li $v0, 10
72     syscall

```

## b. Después de compilar

Text Segment		Basic		Source	
Bkpt	Address	Code			
	4194304	0x24020004	addiu \$2,\$0,4	11:	li \$v0, 4
	4194308	0x3c011001	lui \$1,4097	12:	la \$a0, msgCantidad
	4194312	0x34240000	ori \$4,\$1,0		
	4194316	0x0000000c	syscall	13:	syscall
	4194320	0x24020005	addiu \$2,\$0,5	15:	li \$v0, 5 # leer entero
	4194324	0x0000000c	syscall	16:	syscall
	4194328	0x00024021	addu \$8,\$0,\$2	17:	move \$t0, \$v0 # \$t0 = cantidad de terminos
	4194332	0x24090000	addiu \$9,\$0,0	20:	li \$t1, 0 # primer numero = 0
	4194336	0x240a0001	addiu \$10,\$0,1	21:	li \$t2, 1 # segundo numero = 1
	4194340	0x240b0000	addiu \$11,\$0,0	22:	li \$t3, 0 # suma acumulada

  

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1919381065	543519589	1663066476	1769238113	543449444	1847616868	1919249781	1679848303
268501024	1634476133	1919251232	1176528233	1852793449	1768121185	1730175264	1919250021	540701281
268501056	543247360	1769104755	1766203493	1634627426	543777635	540701541	1632373248	1836413728
268501088	1701060705	1936682016	1836412448	1936683621	543515680	1931501932	1701409381	980641056
268501120	32	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	09/09/2025
	Nombre: JUAN FERNANDO	

### c. Después de ejecutar

Mars Messages	Run I/O
	<pre> Ingrese la cantidad de numeros de la serie Fibonacci a generar: 5 La serie Fibonacci es: 0, 1, 1, 2, 3 La suma de los numeros de la serie es: 7 -- program is finished running -- </pre>

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	7
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	5
\$t2	10	8
\$t3	11	7
\$t4	12	5
\$t5	13	8

### d. Conclusiones

- **\$a0** y **\$t3** terminan con el mismo valor, por que el paso final del proceso mostrar\_suma realiza esta acción
- **\$t0** y **\$t4** finalizan en 5, que fueron la cantidad de números Fibonacci solicitados. El proceso finaliza porque se cumple la igualdad entre ambos registros.
- **\$t1** finaliza en 5 y **\$t2** finaliza en 8 igual que **\$t5**, debido al proceso no\_coma que lleva el valor de **\$t2** a **\$t1** (3+2) y lleva el valor de **\$t5** a **\$t2** (5+3)
- Para imprimir se utilizaron los comandos **li \$vo, 1** (Imprimir enteros), **li \$vo, 4** (Imprimir texto), **li \$vo, 10** (Finalizar programa) y **li \$vo, 11** (Imprimir caracteres).
- **\$v0** finaliza en 10 indicando que el programa se terminó de ejecutar.

### ENLACE GITHUB

<https://github.com/JuanOcacionesA/EstructuraComputadores/tree/main>