Asignatura	Datos del alumno	Fecha
Estructura de	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

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.

```
msgCantidad: .asciiz "Ingrese la cantidad de numeros a comparar (min 3 - max 5): "
       msgInvalido: .asciiz "Cantidad invalida. Debe ser entre 3 y 5.\n"
3
       msgNumero: .asciiz "Ingrese un numero: "
       msgResultado:.asciiz "El mayor numero ingresado 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; 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	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

```
7 .text
8 .globl main
9
10
       main:
      # Pedir cantidad de números
11
       li $v0, 4
12
       la $aO, msgCantidad
13
       syscall
14
15
       li $v0, 5
                       # leer entero
16
17
       syscall
       move $t0, $v0 # $t0 = cantidad
18
19
20
       # Validar rango (3 <= n <= 5)
       blt $t0, 3, invalid
21
       bgt $t0, 5, invalid
22
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 $aO, msgNumero
27
        syscall
28
        # Leer el número
29
30
        li $v0, 5
        syscall
31
        move $t1, $v0 # $t1 = primer número (asumimos que es el mayor inicial)
32
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 \$11 cada que \$t3 sea mayor al número comparado.

Asignatura	Datos del alumno	Fecha
Estructura de	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

```
36 leer numeros:
        beq $t2, $t0, mostrar # si contador == cantidad → mostrar resultado
37
38
        # Pedir otro numero
39
        li $v0, 4
40
        la $a0, msgNumero
41
        syscall
42
43
        li $v0, 5
44
        syscall
45
46
        move $t3, $v0 # $t3 = número actual
47
        # Comparar si es mayor
48
49
        ble $t3, $t1, no cambio
        move $t1, $t3 # actualizar mayor
50
51
```

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

```
52 no cambio:
       addi $t2, $t2, 1
53
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
        li $v0, 4
58
        la $a0, msgResultado
59
        syscall
60
61
        li $v0, 1
62
        move $a0, $t1
63
        syscall
64
65
        li $v0, 10
                         # salir
66
        syscall
67
68
```

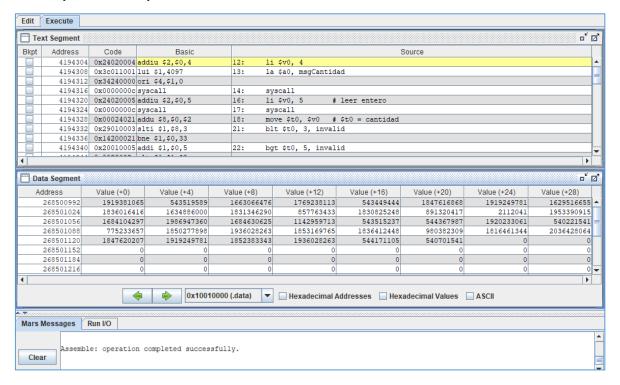
El proceso invalid, imprime el mensaje de "Cantidad invalida...", cuando los números por comparar no se encuentran entre 3 y 5, luego vuelve a saltar al proceso principal.

-	-	-
1	^	í
4	-	
c	Ξ	
ı	,	,
ď	۰	
ċ		
-	_	
	,	
	١	1
	5	
	-	
1	٦	r
٩	-	١
	(
	١	1
_	-	
-	7	
	ſ	J
	-	
	7	
	2	
	2	
	(١
	÷	
	è	
١,	2+0220	
	i	
	۷	
٠	7	
-	,	
	•	
	-	
_	200000000000000000000000000000000000000	
	÷	í
	(J
	۷	
	1	1
		ě
	g	
۰	ě	
	(
ľ	7	
,	-	
(C	١

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

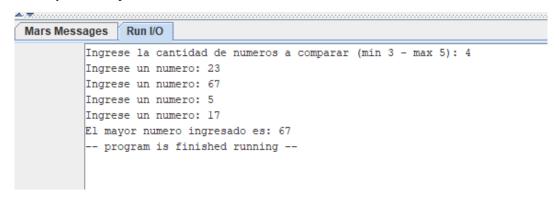
```
invalid:
69
70
        li $v0, 4
71
        la $a0, msgInvalido
        syscall
72
73
        j main
```

b. Después de compilar



Asignatura	Datos del alumno	Fecha
Estructura de	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

c. Después de ejecutar



Registers Co	proc 1 Coproc	0
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$vl	3	0
\$ a 0	4	67
\$al	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	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

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: "
```

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:
      # Pedir cantidad de números
11
12
       li $v0, 4
       la $aO, msgCantidad
13
14
       syscall
15
       li $v0, 5
16
                       # leer entero
17
       svscall
       move $t0, $v0 # $t0 = cantidad
18
19
20
       # Validar rango (3 <= n <= 5)
       blt $t0, 3, invalid
21
       bgt $t0, 5, invalid
22
23
```

Asignatura	Datos del alumno	Fecha
Estructura de	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

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
        li $v0, 4
25
        la $aO, msgNumero
26
27
        syscall
28
        # Leer el número
29
        li $v0, 5
30
31
        syscall
        move $t1, $v0 # $t1 = primer número (asumimos que es el menor inicial)
32
33
        li $t2, 1
                       # contador = 1 (ya tenemos 1 numero)
34
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 \$11 cada que \$13 sea menor al número comparado (bge).

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

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

Asignatura	Datos del alumno	Fecha
Estructura de	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

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
        la $a0, msgResultado
59
60
        syscall
61
        li $v0, 1
62
63
        move $a0, $t1
        syscall
64
65
        li $v0, 10
                         # salir
66
        syscall
67
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:

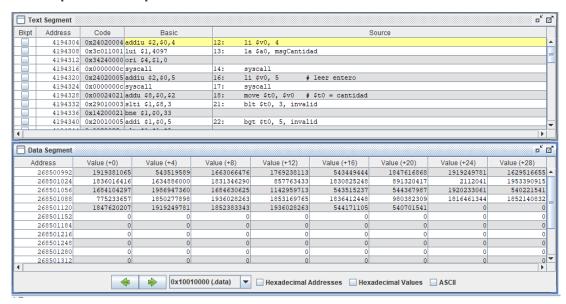
70 li $v0, 4

71 la $a0, msgInvalido

72 syscall

73 j main
```

b. Después de compilar



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	Apellidos: OCACIONES ALVAREZ	00/00/2025
Computadores	Nombre: JUAN FERNANDO	09/09/2025

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 Cop	proc 1 Coproc ()
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$vl	3	0
\$ a 0	4	12
\$al	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	
\$tl	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	00/00/2025	
	Nombre: JUAN FERNANDO	09/09/2025	

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

Asignatura	Datos del alumno	Fecha	
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	00/00/2025	
	Nombre: JUAN FERNANDO	09/09/2025	

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.

```
# Imprimir mensaje serie
25
26
        li $v0, 4
27
        la $aO, msgSerie
        syscall
28
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:
       beq $t4, $t0, mostrar_suma # si contador == cantidad → terminar
31
32
       # Imprimir numero actual (t1)
33
       li $v0, 1
34
35
       move $a0, $t1
36
       syscall
37
        # Imprimir una coma y espacio si no es el ultimo
38
        addi $t4, $t4, 1
39
       beq $t4, $t0, no_coma # si ya es el ultimo, no poner coma
40
       li $v0, 11
41
       li $a0, ','
42
        syscall
43
44
        li $v0, 11
        li $a0, ' '
45
        syscall
46
```

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.

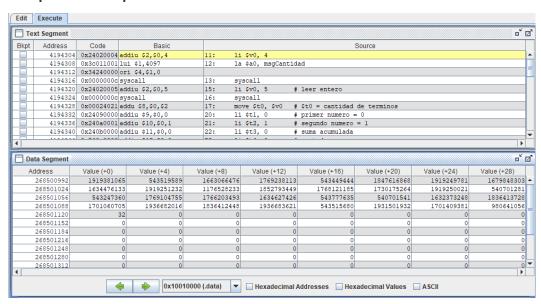
Asignatura	Datos del alumno	Fecha	
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	00/00/2025	
	Nombre: JUAN FERNANDO	09/09/2025	

```
48 no_coma:
49
        # Acumular suma
50
        add $t3, $t3, $t1
51
        # Calcular siguiente numero
52
        add $t5, $t1, $t2 # t5 = t1 + t2
53
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
        li $v0, 4
61
        la $a0, msgSuma
62
63
        syscall
64
        # Imprimir resultado suma
65
        li $v0, 1
66
        move $a0, $t3
67
68
        syscall
69
        # Finalizar
70
        li $v0, 10
71
        syscall
72
```

b. Después de compilar



Asignatura	Datos del alumno	Fecha	
Estructura de Computadores	Apellidos: OCACIONES ALVAREZ	00/00/2025	
	Nombre: JUAN FERNANDO	09/09/2025	

c. Después de ejecutar

```
Mars Messages Run I/O

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 --
```

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