

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	


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

Preparación para el laboratorio

Para poder realizar la práctica de laboratorio deberá usarse la aplicación MARS (MIPS Assembler and Runtime Simulator), que básicamente es un IDE para MIPS. El archivo con el cual podrá ejecutar la actividad es un archivo Executable Jar File (.jar) que tiene como nombre: Mars4_5 y que se encuentra en la siguiente carpeta:

> Archivos > EjecutableLaboratorio |<

onados + Carpeta Cargar

Nombre ▲	Fecha de creación	Fecha de modificación	Modificado por	Tamaño
 Mars4_5.jar	5:09	5:09		4.2 MB ✔

Dentro de las características de esta aplicación encontramos:

- Interfaz gráfica de usuario con control "apuntar y hacer clic" y editor integrado.
- Valores de registro y memoria fácilmente editables, similares a los de una hoja de cálculo.
- Visualización de valores en hexadecimal o decimal.
- Modo de línea de comandos para que los instructores prueben y evalúen fácilmente muchos programas.
- Registros de coma flotante, coprocesador1 y coprocesador2. Herramienta estándar: visualización y edición a nivel de bits de los registros de coma flotante de 32 bits (captura de pantalla).

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	

- Ejecución en un solo paso a velocidad variable.
- Utilidad "Tool" para el control MIPS de dispositivos simulados. Herramienta estándar: Caché.

Conceptos Previos

Assembler: es un lenguaje de programación de bajo nivel (también conocido como ensamblador) que se utiliza para escribir programas que se ejecutan directamente en un computador o en otros dispositivos electrónicos. Los programas escritos en ensamblador son traducidos a lenguaje de máquina, que es el lenguaje que entiende el procesador del computador.

A diferencia de los lenguajes de programación de alto nivel, como C++ o Java, que se enfocan en la abstracción y la simplificación del proceso de programación, el ensamblador es un lenguaje de programación muy cercano a la arquitectura del procesador. Esto significa que los programas escritos en ensamblador son muy eficientes y rápidos, ya que se aprovechan al máximo las capacidades del procesador y se pueden controlar todos los detalles del hardware. Los archivos de este lenguaje tienen la extensión *.asm

Descripción del laboratorio

En esta práctica vamos a trabajar la ejecución de código Assembler que deberá pedir o mostrar por consola ciertos datos que permita la ejecución de los scripts propuestos. Esta actividad puede realizar de manera individual o grupal (máximo 2 estudiantes). Los scripts que se deben desarrollar son los siguientes:

1. Número mayor (mínimo 3 números)
2. Número menor (mínimo 3 números)
3. Serie Fibonacci

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	

Entrega del laboratorio

Una vez finalizado el laboratorio deberás entregar un archivo comprimido (WinRar o WinZip) con los siguientes archivos:

a. Un archivo en formato PDF con un informe en el que aparezcan los siguientes puntos:

- ▶ Para cada uno de los scripts propuestos para la actividad se debe realizar 3 capturas de pantalla:
 - a. Antes de compilar
 - b. Después de compilar
 - c. Después de ejecutar

b. El código en assembler (*.asm) de cada uno de los scripts.

- ▶ El código Assembler utilizado para la generación en cada uno de los scripts debe estar cargado en GitHub (<https://github.com/>). Nota: si la actividad la realizan de forma grupal ambos integrantes debe publicar el enlace a su respectivo perfil en GitHub
- ▶ Cada una de las líneas debe estar comentada con la respectiva descripción de que realiza cada instrucción.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	

1. Número mayor (mínimo 3 números)

a. Antes de compilar

```

1 .data
2
3 num1: .asciz "Ingrese el primer número: "
4 num2: .asciz "Ingrese el segundo número: "
5 num3: .asciz "Ingrese el tercer número: "
6 mayor: .asciz "El número mayor es: "
7
8
9 .text
10 .globl main
11
12 main:
13 # pedir primer número
14 li $v0, 4
15 la $a0, num1
16 syscall
17
18 # Leer el primer número
19 li $v0, 5
20 syscall
21 move $t0, $v0
22
23 # pedir segundo número
24 li $v0, 4
25 la $a0, num2
26 syscall
27
28 # Leer el segundo número
29 li $v0, 5
30 syscall
31 move $t1, $v0
32
33 # Comparar los números
34 slt $t2, $t0, $t1
35 beq $t2, $zero, skip1
36 slt $t2, $t1, $t0
37 beq $t2, $zero, skip2
38 # Si llegamos aquí, $t0 es el mayor
39 la $a0, mayor
40 syscall
41
42 skip1:
43 skip2:
44
45 # Fin del programa
46 li $v0, 10
47 syscall

```

b. Después de compilar

```

Bkpt Address Code Basic Source
0x00400000 0x24020004 addiu $2,$0,0x00000004 14: li $v0, 4
0x00400004 0x3c011001 lui $1,0x00001001 15: la $a0, num1
0x00400008 0x34240000 ori $4,$1,0x00000000
0x0040000c 0x0000000c syscall 16: syscall
0x00400010 0x24020005 addiu $2,$0,0x00000005 19: li $v0, 5
0x00400014 0x0000000c syscall 20: syscall
0x00400018 0x00024021 addu $8,$0,$2 21: move $t0, $v0
0x0040001c 0x24020004 addiu $2,$0,0x00000004 24: li $v0, 4
0x00400020 0x3c011001 lui $1,0x00001001 25: la $a0, num2
0x00400024 0x3424001b ori $4,$1,0x0000001b

```

c. Después de ejecutar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	

c. Después de ejecutar

The screenshot shows the Mars IDE interface. The Text Segment displays assembly code for a Fibonacci program. The Data Segment shows memory values. The Registers window on the right shows the state of registers after execution, with \$a0 containing the result 35.

3. Serie Fibonacci

a. Antes de ejecutar

The screenshot shows the Mars IDE interface with the assembly code for a Fibonacci program. The code includes comments in Spanish and assembly instructions like addiu, lui, syscall, beq, and jal.

b. Después de ejecutar

The screenshot shows the Mars IDE interface. The Text Segment displays assembly code for a Fibonacci program. The Data Segment shows memory values. The Registers window on the right shows the state of registers after execution, with \$a0 containing the result 35.

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos: Mercado Santos	07/Abril/2023
	Nombre: Juan Carlos	

c. Después de ejecutar

The screenshot shows the Mars MIPS simulator interface. The **Text Segment** window displays the following assembly code:

```

9: li $v0, 4
10: la $a0, secuencia
11: syscall
12: li $v0, 5
13: syscall
14: li $v0, 5
15: syscall
16: beq $v0, 0, equalToZero
17: beq $v0, 0, equalToZero
18: move $a0, $v0
19: jal fibonacci
20: move $a0, $v0
21: jal fibonacci

```

The **Data Segment** window shows memory values. The value at address 0x10010000 is 0x10010000. The **Registers** window shows the state of MIPS registers. The **Mars Messages** window shows the output:

```

Introduce el índice de secuencia
20
El valor de fibonacci es: 6765
-- program is finished running --

```

Bibliografía

(s.f.). Obtenido de

<https://lorca.act.uji.es/asignatura/ig09/practicas/documentos/ensambladorR2000.pdf>

youtube. (26 de Febrero de 2021). Obtenido de Arquitectura y Tecnología de

Computadores URJC: <https://www.youtube.com/watch?v=wm8J7W8W1N4>