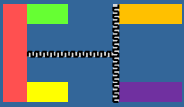


Práctica 4. PROGRAMA EN ENSAMBLADOR

dtic



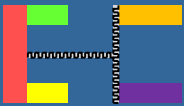


Práctica 4. PROGRAMA EN ENSAMBLADOR

Introducción

- ⊙ Se usará el simulador MARS para comprobar el funcionamiento del programa realizado en ensamblador.
- ⊙ La realización de las prácticas implica la entrega del trabajo propuesto que se detalla a continuación.
- ⊙ El formato de la memoria debe ser el mismo que el de las prácticas anteriores.

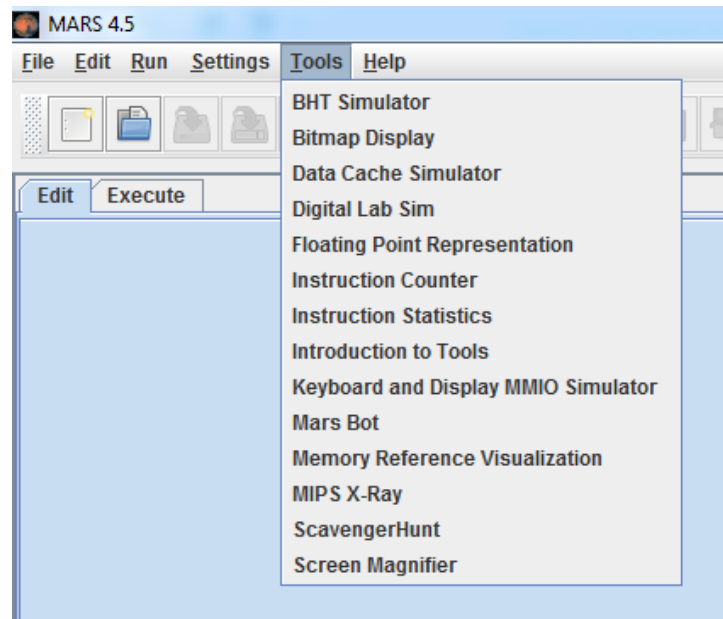


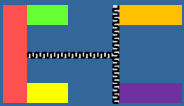


Práctica 4. PROGRAMA EN ENSAMBLADOR

Introducción

- ⦿ El sistema de entrada/salida en MIPS esta mapeado en memoria, es decir, que los puertos o registros de los periféricos se gestionan como direcciones de memoria.
- ⦿ La práctica 2 consiste en gestionar periféricos de Entrada/Salida en MIPS. Para ello, MARS dispone de diferentes TOOLS que son aplicaciones software independientes entre sí pero conectadas a MARS:

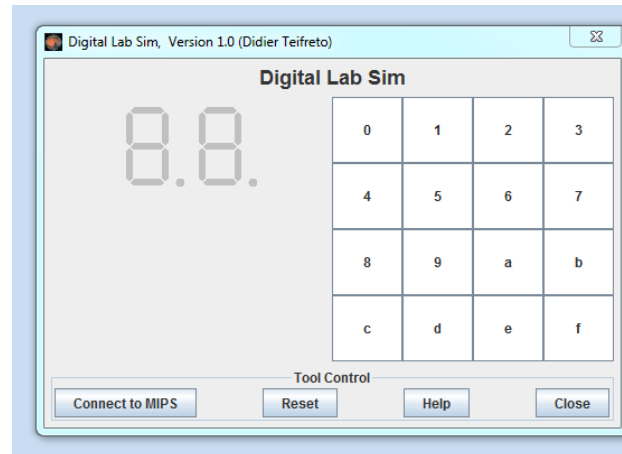


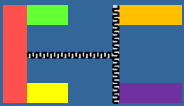


Práctica 4. PROGRAMA EN ENSAMBLADOR

Introducción

- Los periféricos se ejecutan seleccionándolo en el menú TOOLS de MARS
- Para conectar el periférico a MARS hay que pulsar el botón Connect to MIPS
- Para desconectar el periférico, hay que pulsar Disconnect from MIPS

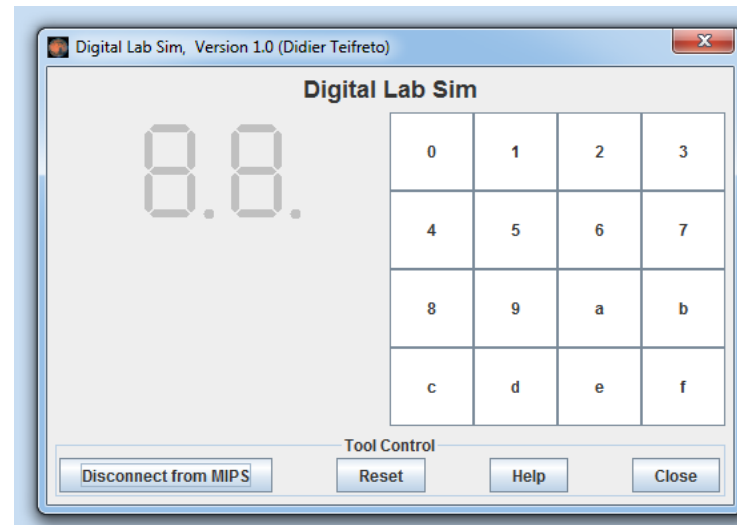


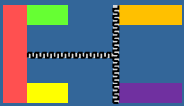


Práctica 4. PROGRAMA EN ENSAMBLADOR

Descripción

- El periférico Digital Lab Sim está formado por 3 partes:
 - 2 displays de siete segmentos
 - Teclado hexadecimal
 - Contador

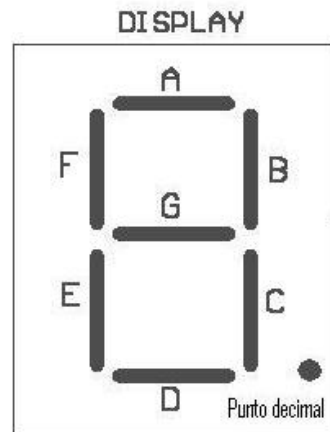


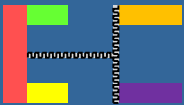


Práctica 4. PROGRAMA EN ENSAMBLADOR

Descripción

- Los displays del periférico Digital Lab Sim se gestionan de la siguiente forma:
- La dirección 0xFFFF0010 se corresponde con el display izquierdo
- La dirección 0xFFFF0011 se corresponde con el display derecho
- El bit 0 activa segmento A, el 1 activa segmento B, ..., el bit 7 activa el punto decimal





Práctica 4. PROGRAMA EN ENSAMBLADOR

Descripción

🎯 Escribir el siguiente programa (display.asm)

Ejemplo displays del periférico Digital Lab Sim

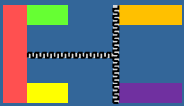
.text
main:

```
li $s0, 0xFFFF0010 # carga dirección base del display derecho  
li $s1, 0xFFFF0011 # carga dirección base del display izquierdo
```

```
li $t1, 0x01 # El bit 0 activa el segmento A  
sb $t1, 0($s0) # almacena en dirección del display derecho el valor de $t1
```

```
li $t1, 0x80 # El punto decimal  
sb $t1, 0($s1) # almacena en dirección del display izquierdo el valor de $t1
```

```
li $v0, 10          # Fin programa  
syscall
```



Práctica 4. PROGRAMA EN ENSAMBLADOR

Descripción

- ⦿ Conecta el periférico Digital Lab Sim
- ⦿ Ejecuta el programa y comprueba qué hace
- ⦿ Modifícalo para que aparezca el número 80 en los displays
- ⦿ Haz que se visualice de forma consecutiva los siguientes segmentos:
 - ⦿ A Display Derecho , A Display Izquierdo, F Display Izquierdo, E Display Izquierdo, D Display Izquierdo, D Display Derecho, C Display Derecho, B Display Derecho
- ⦿ Haz un bucle de la secuencia anterior
- ⦿ ¿Qué ocurre?





Ejercicio propuesto

Práctica

Realizar un nuevo programa en lenguaje ensamblador en MIPS que ordene un vector y lo imprima por la TOOLS Digital Lab Sim.

Consideraciones:

- ◉ El tamaño del vector se introduce por teclado y a continuación se introduce los valores del vector uno a uno. A continuación, pide por teclado si la ordenación es ascendente o descendente.
- ◉ El vector ordenado lo debe mostrar por la TOOLS.
- ◉ Los valores de las componentes del vector serán enteros comprendidos entre 0 y 99.
- ◉ Se debe controlar los errores en la entrada por teclado mostrando un mensaje de información.

