**PRÁCTICA 4.1 (usar la versión 4.12 de Simula3MS)**

**ENTRADA/SALIDA POR CONSULTA DE ESTADO**

**IMPORTANTE: Si habéis actualizado Java después de instalar el simulador tendréis que volver a instalar el simulador porque si no, no funciona el simulador con entrada/salida.**

**Antes de ejecutar el programa debemos configurar el simulador para entrada salida por encuesta**

**Objetivos:**

* Aplicar los conceptos de entrada/salida.
* Aplicar el concepto de consulta de estado.

**Desarrollo / Comentario:**

La arquitectura MIPS utiliza E/S mapeada en memoria, en la que se emplean varias direcciones de memoria para referirse a los puertos de los dispositivos periféricos.

En Simula3MS la dirección base de la E/S mapeada en memoria es 0xffff0000. Dispone de dos dispositivos de E/S: el teclado (Entrada) y el monitor (Salida). Los puertos correspondientes a estos dispositivos son:

Teclado: Puerto de estado/control 🡪 0xffff0000

Puerto de datos 🡪 0xffff0004

Monitor: Puerto de estado/control 🡪 0xffff0008

Puerto de datos 🡪 0xffff000c

El bit 0 del registro de estado/control del teclado (llamado *ready*) cambia automáticamente de 0 a 1 cuando se introduce un carácter en el teclado. Este bit se pone automáticamente a 0 en el momento que el procesador recoge el carácter introducido (mediante una lectura del puerto de datos del teclado).

El bit 0 del registro de estado/control del monitor (llamado *ready*) indica si se ha terminado de escribir el carácter enviado a la pantalla. Si la pantalla está ocupada escribiendo el último carácter enviado, este bit estará a 0. Si ya se ha escrito el último carácter en la pantalla, el bit *ready* pasará a 1.

**Primera parte**: Observar el siguiente programa que lee un carácter introducido por el teclado y lo visualiza en el monitor

.text

.globl main

main:

li $t0, 0xffff0000 # dirección base de los puertos de E/S

kbready:

lw $t1, 0($t0) # leemos puerto de estado del teclado

andi $t1, $t1, 1 # aislamos el bit de estado del teclado

beq $t1, $0, kbready # preguntamos si está activo

lb $t2, 4($t0) # recoger el carácter del teclado (leemos del puerto de datos del teclado)

moready:

lw $t3, 8($t0) # leemos puerto de estado del monitor

andi $t3, $t3, 1 # aislamos el bit de estado del monitor

beq $t3, $0, moready # preguntamos si está activo

sb $t2, 12($t0) # mostrar el carácter por pantalla

addi $v0, $0, 10 # salir del programa

syscall

1. ¿En qué registro de la CPU se almacena el dato leído del teclado?

**$t2**

1. Ejecuta el programa con el simulador. Pulsa la tecla 7. ¿qué valor se almacena en el registro de datos del monitor?

**0x37**

1. ¿Qué significa ese número?

**Carácter ASCII 0x37 (7).**

1. ¿Cuántas teclas puedes visualizar?

**1**

1. ¿Cómo modificaríamos el programa para poder seguir pulsando teclas y visualizarlas en el monitor de manera infinita?

**Después del sb $t2, 12($t0), añadir la instrucción j kbready.**

1. Suponer que rediseñamos el sistema hardware y el bit de estado del teclado es el bit 5 en vez de ser el bit 0 ¿qué tendríamos que modificar en el programa anterior?

**andi $t1, $t1, 0x20**

**Segunda parte**: Empleando E/S por consulta de estado, se pide:

1. Escribe una función que lea una cadena de caracteres introducidos desde el teclado, los escriba en memoria a partir de la posición indicada en el registro $a0, y finalmente devuelva en el registro $v0 el número de caracteres leídos. La cadena termina cuando se pulsa la tecla ESC (código 0x3b).

**leer\_cadena:**

**add $v0, $0, $0 # $v0 contiene la cantidad de caracteres leídos**

**li $t0, 0xffff0000 # $t0 contiene la dirección base de los puertos de E/S**

**addi $t1, $0, 0x3B # $t1 contiene el carácter de salida**

**buclelectura:**

**lw $t2, 0($t0) # leemos puerto de estado del teclado**

**andi $t2, $t2, 1 # aislamos el bit de estado del teclado**

**beq $t2, $0, buclelectura # preguntamos si está activo**

**lb $t3, 4($t0) # recogemos el carácter del teclado**

**beq $t3, $t1, rtnlectura # si el carácter es el de salida, salimos de la función**

**sb $t3, 0($a0) # guardamos el carácter leído**

**addi $a0, $a0, 1 # pasamos al siguiente byte de la dirección**

**addi $v0, $v0, 1 # incrementamos la cantidad de caracteres leídos**

**j buclelectura**

**rtnlectura:**

**sb $0, 0($a0) # guardamos byte nulo al final de la cadena**

**jr $ra**

1. Escribe una función que muestre por pantalla los datos que hay a partir de la dirección de memoria indicada en el registro $a0, teniendo en cuenta que la longitud de la cadena a mostrar está en el registro $a1.

mostrar\_cadena:

**li $t0, 0xffff0000 # $t0 contiene la dirección base de los puertos de E/S**

**bucleescritura:**

**lw $t3, 8($t0) # leemos puerto de estado del monitor**

**andi $t3, $t3, 1 # aislamos el bit de estado del monitor**

**beq $t3, $0, bucleescritura # preguntamos si está activo**

**beq $a1, $0, rtnescritura**

**lb $t1, 0($a0)**

**sb $t1, 12($t0) # mostrar el carácter por pantalla**

**addi $a1, $a1, -1 # reducimos la cantidad de caracteres a mostrar**

**addi $a0, $a0, 1 # pasamos al siguiente carácter**

**j bucleescritura**

**rtnescritura:**

**jr $ra**

1. Escribe un programa que llame a las dos funciones anteriores. El programa debe pasar al registro $a1 como argumento de la función del apartado 2 la salida $v0 de la función del apartado1.

**main:**

**la $a0, cadena**

**jal leer\_cadena**

**la $a0, cadena**

**add $a1, $v0, $0**

**jal mostrar\_cadena**

**fin:**

**addi $v0, $0, 10**

**syscall**