ESTRUCTURA DE COMPUTADORES

Tema 9/Ejercicio de clase_2

Se asume el mismo interfaz de Disco Duro (HD) del Ejercicio_1 de clase, así como la misma descripción de los registros COMMAND y STATUS de dicho interfaz . Supóngase que el driver del controlador de HD dispone de la siguiente función:

Función	Índice (en $v0)	Parámetros
Read_Disk	900	$a0: Puntero a buffer de memoria $a1: Número de ciclos de transferencia $a3: Coordenadas del sector

***Nota**: Se asume un entorno multitarea*

a) Escríbase el código de la función **Read_Disk** para el modo **PIO** y sincronización por **Interrupción**, asumiendo la existencia de sendas variables del sistema, `Puntero` y `Contador`, como se observa en el recuadro

```
        .kdata
Puntero:  .word 0
Contador: .word 0
```

```
#Esta función debe configurar el interfaz para realizar una lectura de bloque en modo
#PIO y sincronización por interrupción. Se requiere el empleo de dos variables de
#estado auxiliares para contener provisionalmente los valores del puntero a memoria
# y del contador que se pasan como parámetros, dado que en modo PIO no están
#operativos los registros Pointer y Counter del interfaz para realizar la transferencia
        .ktext
Read_Disk:  la $t0,0xFFFF00E0
        sw $a0, Puntero         #Carga puntero a memoria en la variable auxiliar Puntero
        sw $a1, Contador        #Carga contador en la variable auxiliar Contador
        sw $a3, 12($t0)         #Carga ID del bloque en Block_ID
        li $t1, 0x14            # PIO/DMA= 00; A=1; R/W= 0; IE= 1
        sw $t1, 0($t0)          #Inicializa registro COMMAND
        jal suspende_proceso    #Suspende el proceso que invocó la función
        j retexc                #Salto al código de salida del manejador
```

b) Escríbase el código de la Interrupción **Int0** para el modo **PIO**

```
#Esta rutina se invocará cuando el periférico esté preparado para iniciar la transferencia
        .ktext
Int0:   la $t0,0xFFFF00E0
        lw $a0, Puntero         #carga en $a0 dirección buffer de memoria desde variable sistema
        lw $a1, Contador        #carga en $a1 número de ciclos de transferencias desde variable sistema

#Bucle de transferencia por programa (modo PIO)
bucle:  lw $a2, 8($t0)          #Lee word del registro Data
        sw $a2, 0($a0)          #Lo escribe a memoria en la dirección apuntada por $a0
        addi $a0, $a0, 4        #Incrementa el puntero $a0 en 4
        addi $a1, $a1, -1       #Decrementa contador $a1 en 1
        bnez $a1, bucle         #Continua en el bucle de transferencia mientras $a1<>0
        li $t1, 0x80            # PIO/DMA= 00; A=0; R/W= 0; IE= 0; CL= 1
        sw $t1, 0($t0)          #Cancela e inhibe la interrupción en interfaz (Reg. COMMAND)
        jal activa_proceso      #Activa proceso suspendido al invocar Read_Disk
        j retexc                #Salto al código de salida del manejador
```

c) Escríbase el código de invocación de **Read_Disk** en programa usuario para modo **PIO**

```
#El código de invocación de la función Read_Disk desde el programa de usuario se limita a pasar los parámetros que la función requiere, junto al número de función
                .text
                la $a0, Mem_Block       #pasa dirección buffer de memoria como parámetro en $a0
                li $a1, 256             #pasa número de transferencias como parámetro en $a1
                li $a3, 0x44442222      #pasa ID_Bloque como parámetro en $a3
                li $v0, 900             #número de función 900
                syscall                 #invoca llamada al sistema
                . . . . . .
```

d) Escríbase el código de la función **Read_Disk** para el modo **DMA** y sincronización por **Interrupción**

```
    #Esta función debe configurar el interfaz para realizar una lectura de bloque en modo DMA y
    #sincronización por interrupción. Dado que en modo DMA sí están operativos los registros Pointer y Counter
    #del interfaz para realizar la transferencia, la función se limitará inicializar dichos registros con los
    #parámetros que se le han pasado

                .ktext
Read_disk:      la $t0,0xFFFF00E0
                sw $a0, 16($t0)         #guarda dirección buffer en reg. Pointer
                sw $a1, 20($t0)         #guarda número de transferencias en reg. Counter
                sw $a3, 12($t0)         #establece ID_Bloque en reg. Block_ID
                li $t1, 0x17
                sw $t1, 0($t0)          #Habilita interrupción en interfaz y fija DMA
                                        # PIO/DMA= 11; A=1; R/W= 0; IE= 1
                jal suspende_proceso    #Suspende el proceso que invocó la función
                j retexc                #Salto al código de salida del manejador
```

e) Escríbase el código de la Interrupción **Int0** para el modo **DMA**

```
#Esta rutina se invocará cuando el DMA haya completado la transferencia y el bloque solicitado se halle almacenado en memoria (en el buffer cuya dirección o puntero se le pasó como parámetro a la función Read_disk). La rutina se limitará a cancelar la interrupción y a activar el proceso suspendido.
                .ktext
Int0:           la $t0,0xFFFF00E0
                 li $t1, 0x80
                sw $t1, 0($t0)          #Inhibe interrupción en interfaz y cancela interrupción
                                        # PIO/DMA= 00; A=0; R/W= 0; IE= 0; CL= 1  (desconfigura adaptador)
                jal activa_proceso      #Activa proceso suspendido al invocar Read_Disk
                j retexc                #Salto al código de salida del manejador
```

f) Escríbase el código de invocación de **Read_Disk** en programa usuario para modo **DMA**

```
#Obsérvese que el código de invocación de Read_Disk es independiente del modo de transferencia PIO o DMA
                .text
                la $a0, Mem_Block       #pasa dirección buffer de memoria como parámetro en $a0
                li $a1, 256             #pasa número de transferencias como parámetro en $a1
                li $a3, 0x44442222      #pasa ID_Bloque como parámetro en $a3
                li $v0, 900             #número de función 900
                syscall                 #invoca llamada al sistema
                . . . . . .
```