Luis Girona Perez 20083378P

Practica 10

Cuestión 4

➢ Transforma el programa echo de la cuestión 3 en el programa caps que muestra por la consola la mayúscula del carácter introducido por el teclado. Supón que todos los caracteres introducidos están en minúscula.

cuestion4.asm | cuestio5.asm

```
#Código de partida de la cuestión 2#
#         getc y putc             #
#                                 #
###################################
.text
main:
        li $t6, '\n'
        move $a0, $v0                   # lo llevamos a a0 para imprimirlo
        jal getc
        move $s0,$v0                    # almacenamos el valor en s0
        subi $v0, $v0, 32               # restamos 32 al valor introducido

        jal putc
        beq $s0 ,$t6, fin               # si el valor introducido es un salto de linea finaliza el programa
        j main

getc:
        lui $t0, 0xffff
        li $t1, 0                       # contador de bucle
        bucle:
                lw $t2, 0($t0)          # cargamos en t2 el dato introducido
                andi $t2, $t2, 1        # con esto extraemos el bit de ready
                addiu $t1, $t1, 1       # aumentamos el contador en 1
                beqz $t2, bucle         # si t2 es igual a 0, volvemos al bucle
                lw $v0, 4($t0)          # cargamos en v0 el dato introducido
                jr $ra

putc:
        lui $t0, 0xffff
        bucleput:
                lw $t1, 8($t0)          # cargamos en t1 el valor que hay en t0+8
                andi $t1, $t1, 0x0001   # ponemos el bit ready a 1
                beq $t1, $zero, bucleput        # si t1 es 0, volvemos a bucle2
                sw $a0, 12($t0)         # almacenamos el valor de $a0 en la direccion 12+$t0
                jr $ra

fin:
        li $v0,10
        syscall
```

El programa se ha hecho de manera que introduzcamos un texto y finalice cuando detecte un salto de línea.

Con getc, lo que hacemos es extraer los caracteres que hemos introducido, dentro de getc tenemos bucle, que se encarga de comprobar si el bit de ready esta a 1 o a 0. En caso de que estuviera a 1 guardaríamos en v0 el valor que tenga la dirección de memoria de $t0+4.

Con putc haríamos lo mismo, un bucle comprobando si el bit de ready es 0 o 1y guardando en el valor de a0 en la dirección e memoria 12+$t0.

Una vez volvemos al programa principal comprobamos si $s0 es un salto de línea, en caso de que lo fuera terminaríamos el programa. En caso contrario volveríamos a ejecutar el programa principal.

Cuestión 5

➢ Complétalo escribiendo la función read_string. Esta función tiene que leer del teclado la cadena de caracteres que introduzca el usuario y tiene que almacenarla en un buffer denominado cadena. La cadena finaliza cuando el usuario teclee un salto de línea. Posteriormente el programa muestra la cadena en la consola. Al escribir la función read_string no olvidéis meter en el buffer el carácter de salto de línea.

cuestion4.asm | cuestio5.asm

```
cadena: .space 32
.eqv ControlTeclado 0
.eqv BufferTeclado 4
.eqv ControlDisplay 8
.eqv BufferDisplay 12
.text
la $a0,cadena
jal read_string
la $a0,cadena
jal print_string
li $v0,10
syscall
####################################
#           Funcions               #
####################################
print_string:
        la $t0,0xFFFF0000
sync:
        lw $t1, ControlDisplay($t0)
        andi $t1,$t1,1
        beqz $t1,sync
        lbu $t1,0($a0)
        beqz $t1,final
        sw $t1, BufferDisplay($t0)
        addi $a0,$a0,1
        j sync
final:
        jr $ra
read_string:
        lui $t0,0xffff                          #Se hace una seleccion
        li $t1 , 0                              #Se inicia un contador de iteraciones
        getc:
                lw $t2 , ControlTeclado($t0)
                andi $t2, $t2 , 1               #Extrae bit previamente leido por teclado
                addiu $t1 , $t1 , 1             #Se incrementa contador
                beqz $t2 , getc                 #Si es 0 se vuelve a extraer el siguient byte por teclado
                lbu $t1,0($a0)                  #Carga el byte almacenado
                beqz $t1,final                  #Si el contador es 0 se salta al final
                lw $v0 , BufferTeclado($t0)
                j getc
```

En read_string, iniciamos el contador del bucle. Dentro de getc cargamos en t2 el valor de Controlteclado+t0. Extraemos el bit con andi e incrementamos el contador del bucle en 1.

Comprobamos con beqz si $t2 es igual a 0, si lo es volvemos a ejecutar getc(el bucle), si no lo es ejecutamos lbu, guardando en t1 el valor que se encuentra 0+$a0. Si $t1 es igual a 0, terminaríamos el programa, si no lo es cargaríamos en $v0 BufferTeclado+$t0 y volveríamos a ejecutar getc.

Práctica 11

Cuestión 9

➢ **Añade un programa principal a la rutina de tratamiento de excepciones de la actividad 4 que provoque una excepción por desbordamiento o dirección inválida y prueba el funcionamiento de la rutina de tratamiento de excepciones**

Cuestión 11

➢ Modifica la rutina de tratamiento de interrupciones para que escriba en el display del transmisor el carácter leído en el receptor. Haz que guarde en el registro $v0 el carácter leído. Escribe un programa principal apropiado para hacer pruebas que finalice cuando en el receptor se pulse un salto de línea.

Practica11ej11.asm

```
.kdata
contexto: .word 0,0,0,0

.ktext 0x80000180                       # direccion donde comienza la rutina

# guardo registros para utilizar en la rutina
la $k1, contexto
sw $at, 0($k1)                          # guardo en at el valor de la posicion 0 + k1
sw $t0, 4($k1)                          # guardo en $t0 el valor de la posicion 4 + $k1
sw $v0, 8($k1)                          # guardo en $vo el valor de la posicion 8 + $k1
sw $a0, 12($k1)                         # guardo en $a0 el valor de la posicion 12 + $k1

# compruebo si se trata de una interrupción
mfc0 $k0, $13                           # movemos al coprocesador0 para el registro cause
srl $a0, $k0, 2                         # extraemos el campo del código
andi $a0, $a0, 0x1f
bne $a0, $zero, acabamos                # si a0 vale 0 saltamos a acabamos

# tratamiento de la interrupcion de excepcion
li $t0, 0xffff0000
lb $a0, 4($t0)                          # lee carácter del teclado

li $v0, 11                              # escribe en la consola el carácter leído
syscall
```

```
# Antes de acabar se deja todo iniciado
acabamos:
        mtc0 $0, $13                    # iniciamos registro Cause
        mfc0 $k0, $12                   # leemos registro Status
        andi $k0, 0xfffd                # iniciamos el bit de excepción
        ori $k0, 0x11                   # habilitamos las interrupciones
        mtc0 $k0, $12                   # reescribimos registro Startus


# Restaurar registros
lw $at, 0($k1)                          # devuelvo a at el valor de 0+k1
lw $t0, 4($k1)                          # devuelvo a t0 el valor de 4+k1
lw $v0, 8($k1)                          # devielvo a v0 el valor de 8+k1
lw $a0, 12($k1)                         # devuelvo a a0 el valor de 12+k1
# Devolver en el programa de usuario
eret
```