Estructura de los computadores

Practica 10 11

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3

Índice

Practica 10

Ejemplos

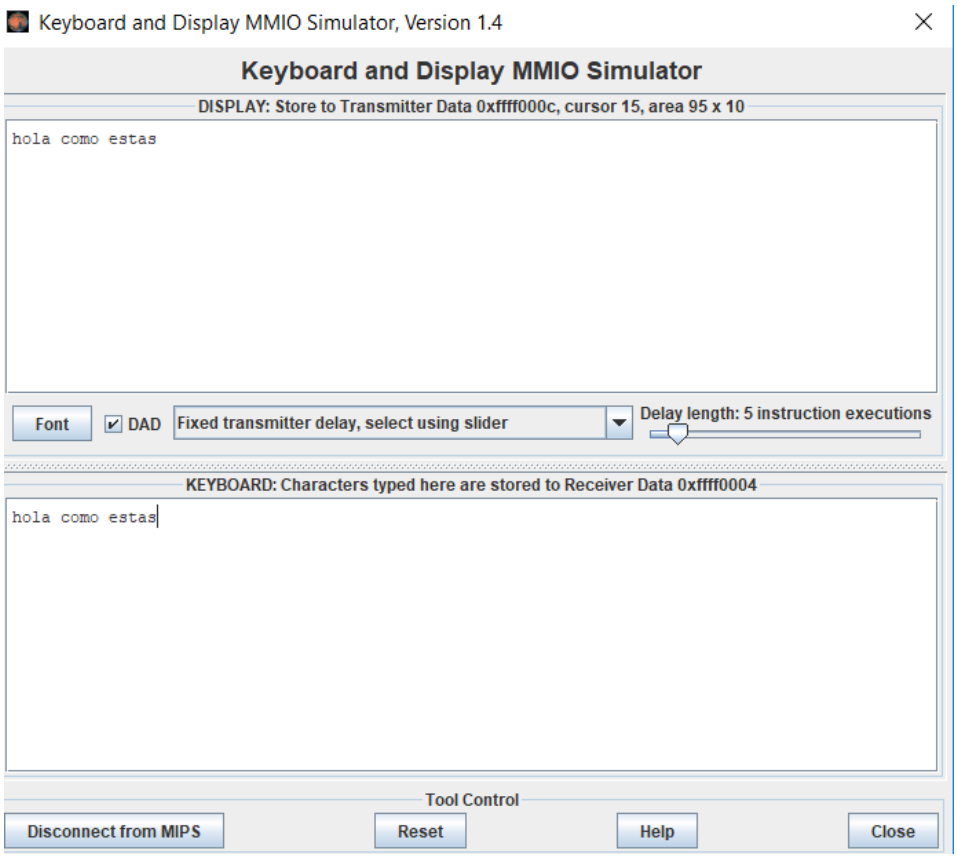
1. Getc y putc.

1.1. Completadlo con las funciones getc (leer un carácter del teclado) y putc (escribir un carácter en teclado) vía polling.

```
.text
main:
        jal getc
        move $a0, $v0
        jal putc

        j main
end:
        li $v0,10
        syscall
```

```
getc:
        #Leer del teclado
        lui $t0, 0xffff # Direc. del registro de control del teclado
        li $t1, 0
        inicio:
        b_espera:
        lw $t2, ($t0) #Lee registro control del teclado
        #SINCRONIZACIÓN:
        andi $t2, $t2, 1 #Extrae el bit de ready
        addiu $t1, $t1, 1
        #(cuenta las iteraciones)
        beqz $t2, b_espera # Si cero no hay carácter
        #TRANSFERENCIA:
        lw $v0, 4($t0) #Lee registro de datos del teclado
```

```
putc:
        #Escribir en la consola
        lui $t0,0xffff #ffff0000; SELECCIÓN
        b_esperac:
        lw $t1,8($t0) #registre control
        andi $t1,$t1,0x0001 #bit de ready SINCRONIZACIÓN
        beq $t1,$0,b_esperac
        sw $a0,12($t0) # TRANSFERENCIA

        jr $ra
```

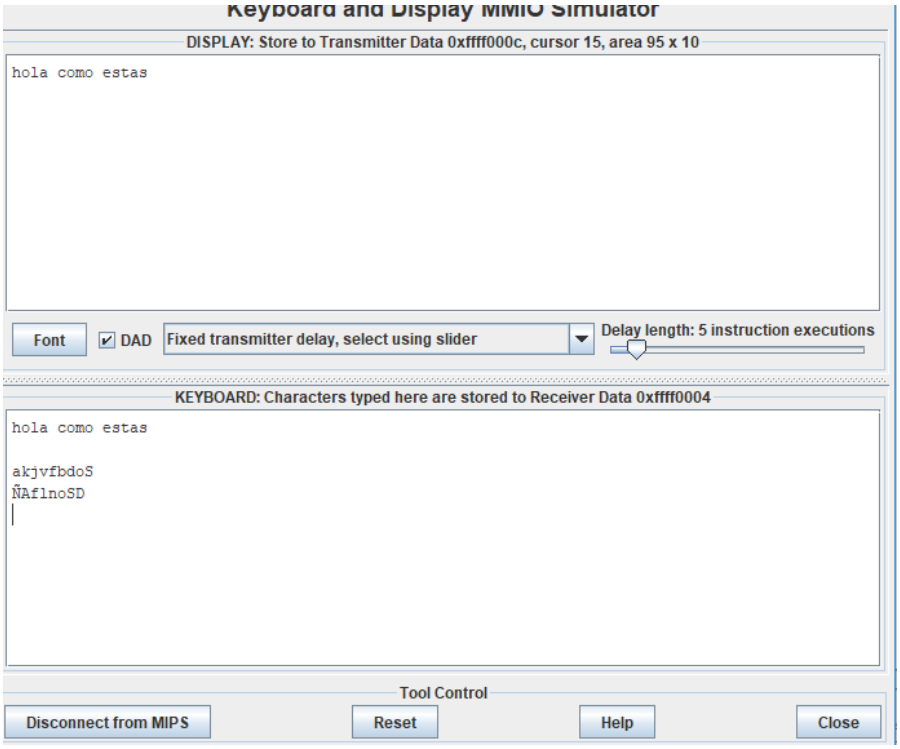


2. Programa echo

2.1. Iterad el código anterior hasta que el carácter introducido sea un salto de línea (‘/’).

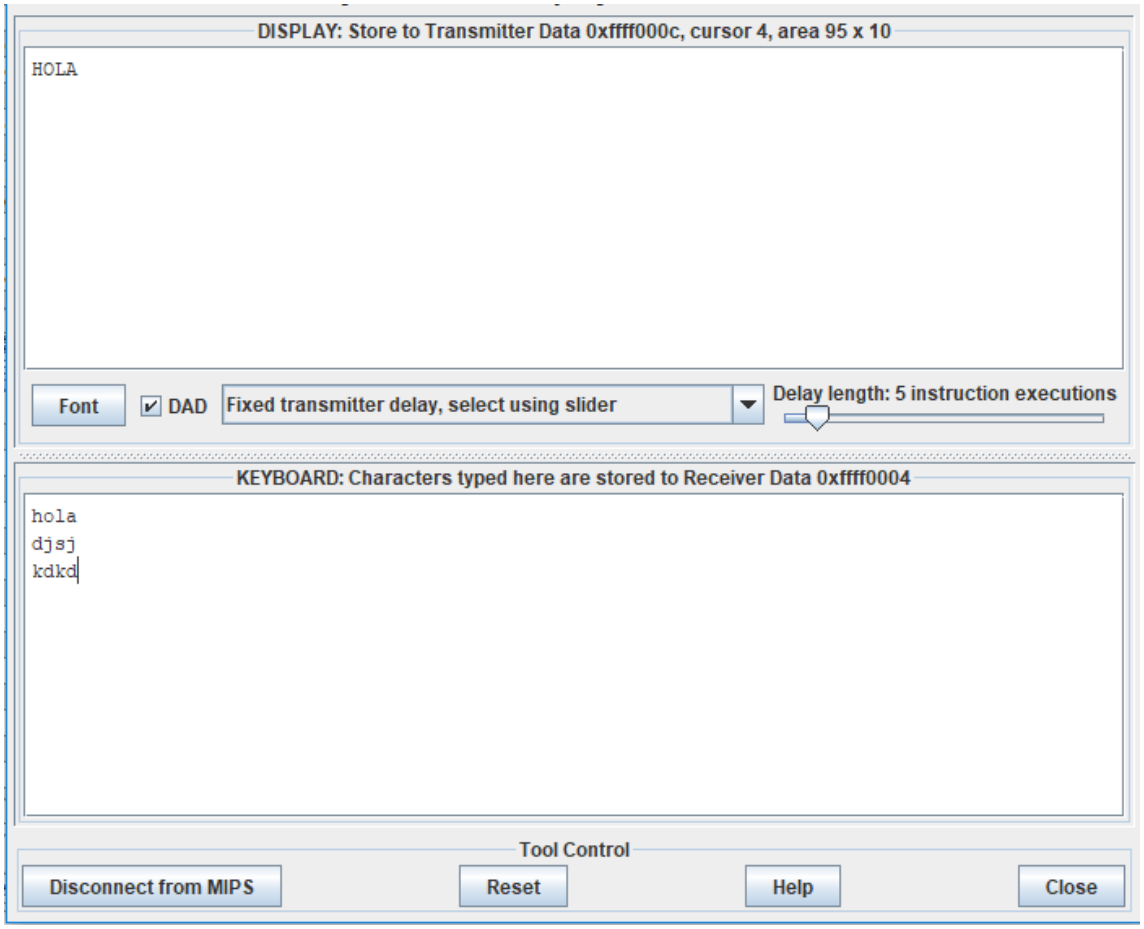
```
li $t4,'\n'
main:
        jal getc
        move $a0, $v0
        jal putc

        beq $v0,$t4,end
        j main
```

Simplemente seria añadir estas instrucciones en el main


Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 15, area 95 x 10
hola como estas
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
hola como estas
akjvfbdoS
ÑAflnoSD
Tool Control
Disconnect from MIPS
Reset
Help
Close

Entregas

Transforma el programa echo 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.



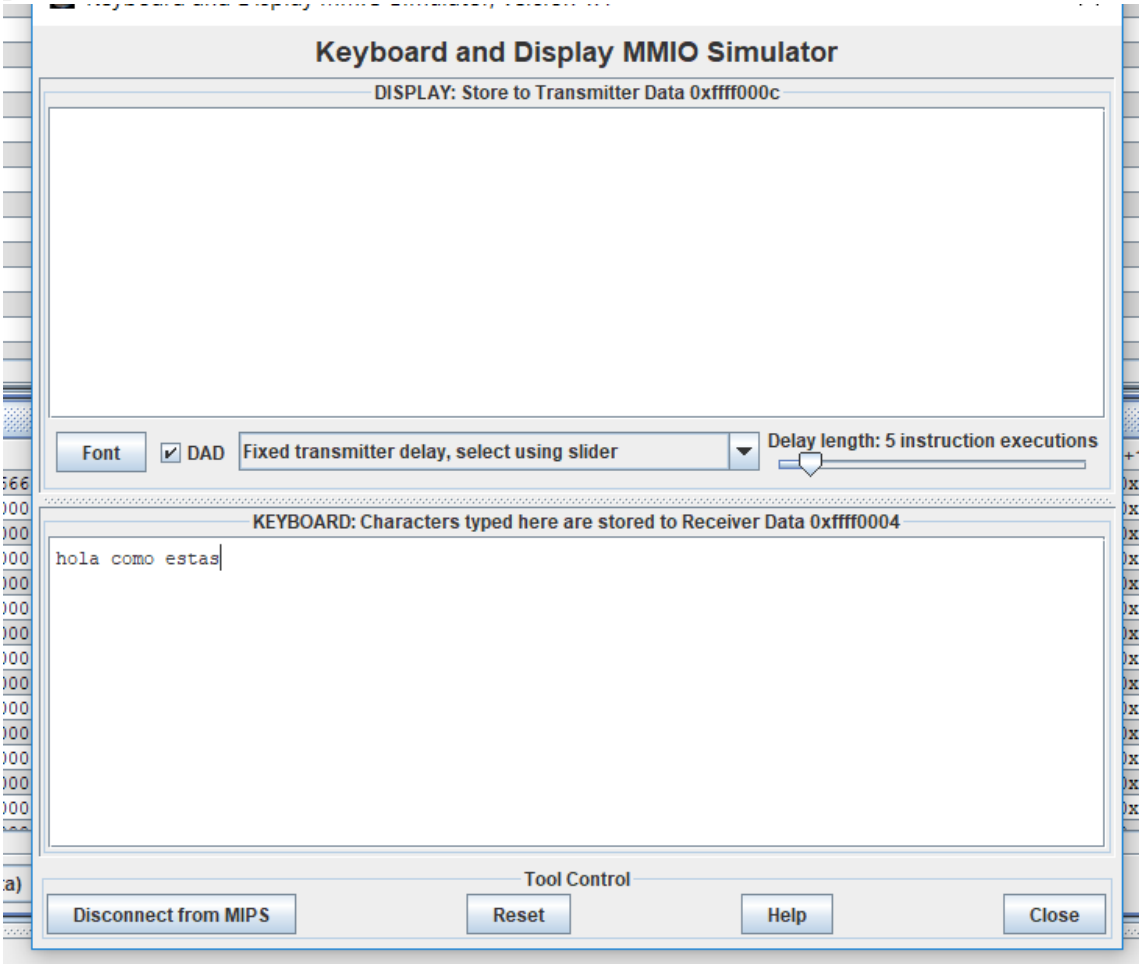
La palabra hola la pone en mayúsculas y al darle al intro se detiene y no muestra por pantalla

La modificación seria la siguiente:

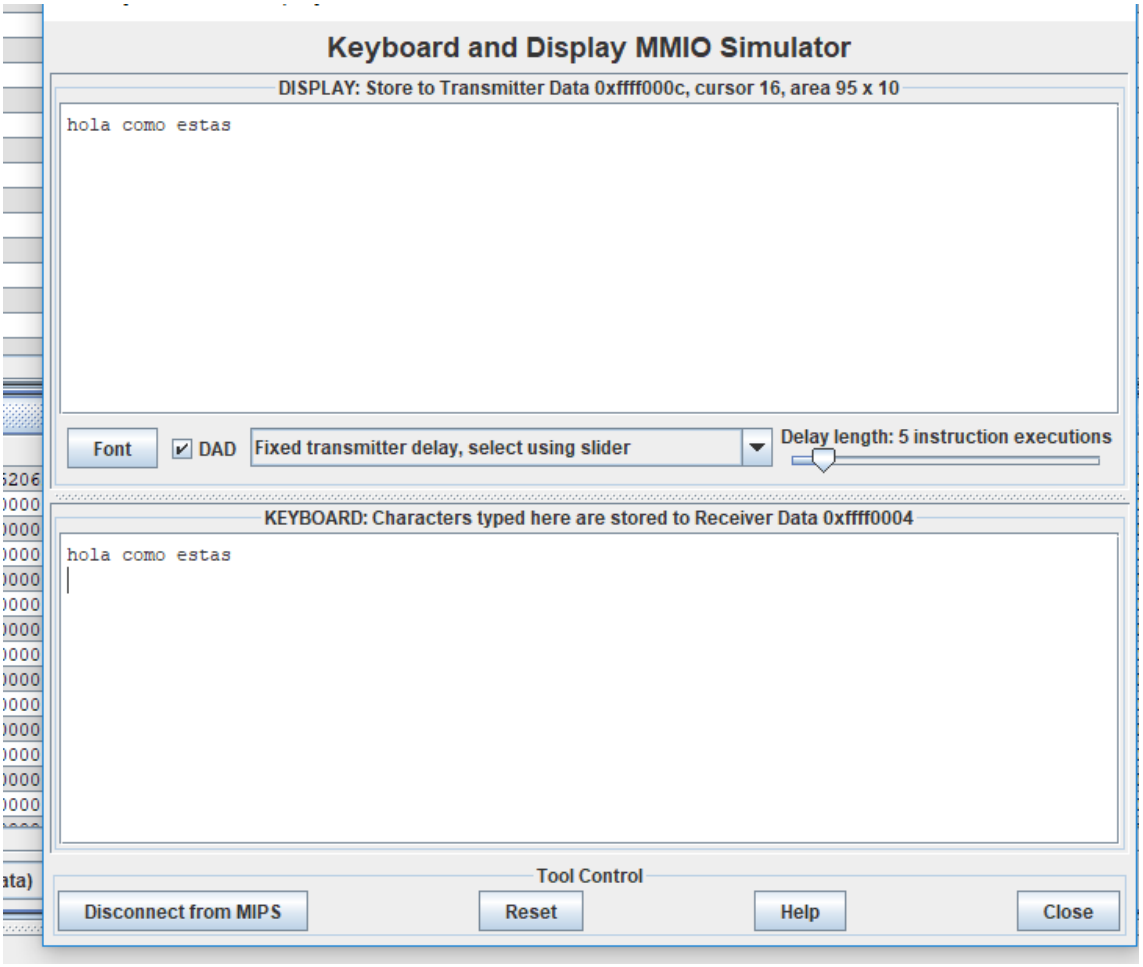
```
putc:
        #Escribir en la consola
        #Carácter de salida en $a0
        lui $t0,0xffff #ffff0000; SELECCIÓN
        b_esperac:
        lw $t1,8($t0) #registre control
        andi $t1,$t1,0x0001 #bit de ready SINCRONIZACIÓN
        beq $t1,$0,b_esperac
        addi $a0,$a0,-32# antes de hacer la transferencia resto 32 posiciones de la tabla asci
        sw $a0,12($t0) # TRANSFERENCIA

        jr $ra
```

Dado el siguiente código, 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.

```
read_string:
    la $t0,0xFFFF0000
    li $t3,0
sync_read:
    lw $t1,ControlTeclado($t0) # almacenar el caracter leido en la memoria
    andi $t1,$t1, 1
    beqz $t1,sync_read # si es \n salir de read_string
    lw $t2,BufferTeclado($t0) # si no volver a leer el siguiente caracter
    sb $t2,cadena($t3)
    addi $t3,$t3,1
    beq $t2,'\n',final_read
    j sync_read
final_read:
    jr $ra
```



Al meter un salto de linea:



Practica 11

Ejemplos

3. Excepciones e interrupciones en el MARS

3.1. Dado el siguiente código

3.1.1. Observa el código. ¿Qué instrucción causará la excepción?

addi $t2, $t0, 1

3.1.3. ¿Cuáles son los valores de los registros del coprocesador 0 antes y después de producirse la excepción?

antes

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000

después

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff13
$13 (cause)	13	0x00000030
$14 (epc)	14	0x0040000c

3.1.4. ¿Cuál es el significado de los distintos campos de los registros del coprocesador 0 después de producirse la excepción?

Excepción por desbordamiento

3.2. Dado el siguiente código

3.2.2. ¿Cuáles son los valores de los registros del coprocesador 0 antes y después de producirse la excepción?

antes

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000

Después

Name	Number	Value
$8 (vaddr)	8	0x0000007c
$12 (status)	12	0x0000ff13
$13 (cause)	13	0x00000014
$14 (epc)	14	0x00400004

3.2.3. ¿Cuál es el significado de los distintos campos de los registros del coprocesador 0 después de producirse la excepción?

Excepción por dirección errónea

3.3. Dado el siguiente código

3.3.1. Cuáles son los valores de los registros del coprocesador 0 antes y después de producirse la excepción?

antes

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000

después

Name	Number	Value
$8 (vaddr)	8	0x10010003
$12 (status)	12	0x0000ff13
$13 (cause)	13	0x00000010
$14 (epc)	14	0x00400008

3.3.2. ¿Cuál es el significado de los distintos campos de los registros del coprocesador 0 después de producirse la excepción?

Excepción por dirección

4. Supón que el contenido del registro Cause ($13) tiene los siguientes valores después de haberse producido una excepción. Rellena la tabla indicando cual ha sido la causa que ha provocado la excepción en cada caso:

Cause	Fuente de la excepción
0x00000000	Interrupción (Hardware)
0x00000020	Excepción syscall
0x00000024	Excepción por punto de ruptura (breakpoint)
0x00000028	Excepción por instrucción reservada
0x00000030	Excepción por desbordamiento aritmético

1. Rutina de tratamiento de excepciones software (traps)

1.1. ¿Cuál es la secuencia de instrucciones que permite averiguar el código de excepción que ha causado la excepción?

```
#Detectamos solo dos excpcions
li $s0, 0x0030 # código Desbordamiento
li $s1, 0X0014 # código error de dirección store
beq $a0, $s0, Desbordo
bne $a0, $s1, salida
```

1.2. ¿Qué conjunto de instrucciones permiten incrementar el registro EPC en 4?

```
la $a0, mis1
li $v0, 4
syscall
mfc0 $a0, $14 # $a0 <= EPC, donde ha ocurrido la excepción
li $v0, 34
syscall # Escribimos EPC en hexadecimal
```

1.3. ¿Qué sucede si ocurre una excepción aritmética por división por 0?

1.4. ¿Que pasaría si no se incrementara el registro EPC en 4?

Si PC no se incrementara no pasaría a la siguiente instrucción

1.5. ¿Por qué otra instrucción podrías sustituir la instrucción eret? ¿Cómo quedaría?

Jr $k0

2. Rutina de tratamiento de interrupciones (excepciones hardware)

2.1. Estudia el código de la rutina de tratamiento de interrupciones anterior. ¿Qué hace la rutina para dar servicio a la interrupción? ¿De dónde proviene la interrupción?

Se restauran los registros estatus.

De dispositivos de entrada

2.2. ¿Cuál es la secuencia de instrucciones que permite averiguar si la excepción ocurrida se debida a una interrupción?

```
mfc0 $k0, $13 # Registro Cause
srl $a0, $k0, 2 # Extraemos campo del código
andi $a0, $a0, 0x1f
bne $a0, $zero, acabamos # Sólo procesamos aquí E/S
```

2.3. ¿Cuáles diferencias se observan entre la rutina de tratamiento de interrupciones y la rutina de tratamiento de excepciones?

En el de excepciones se utiliza el registro EPC para avanzar a la siguiente instrucción; en las interrupciones no, en esta se utiliza los campos de E/S

2.4. ¿Podrían incluirse los dos tratamientos en una misma rutina

Si, teniendo en cuenta que el EPC no se incrementa, pero en excepciones si

Entregas

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.

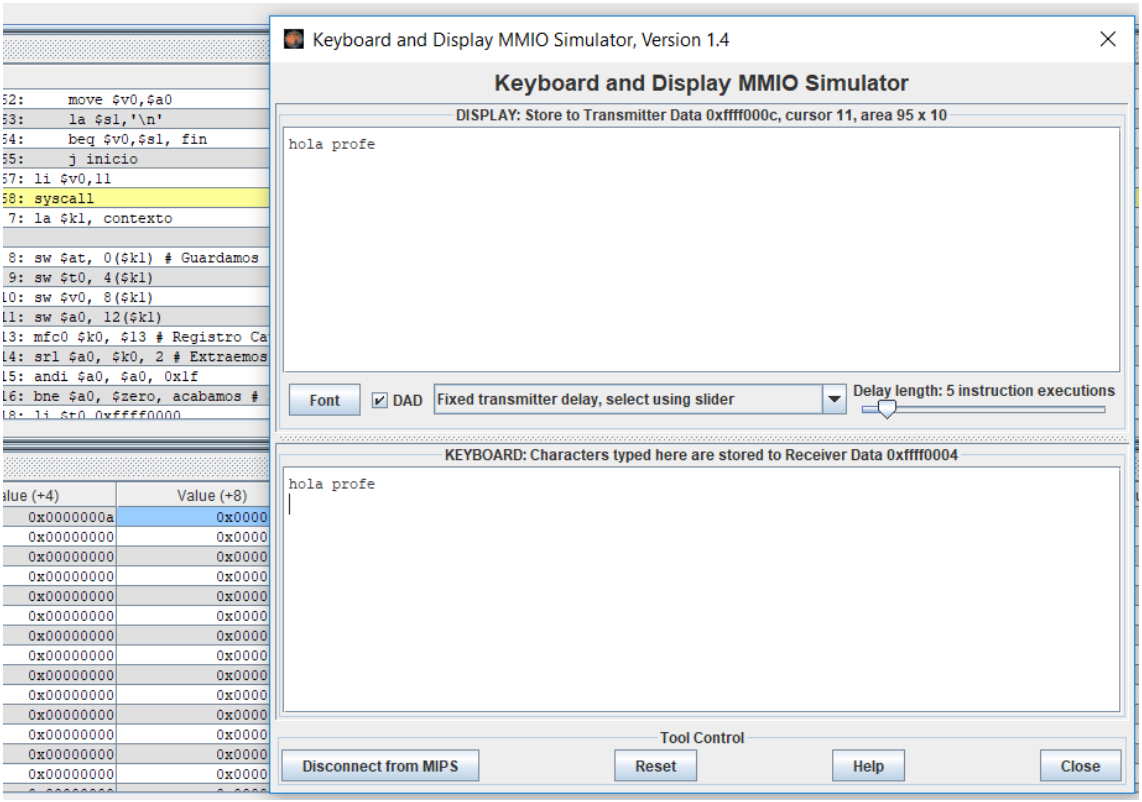
A la ruta de tratamiento de interrupciones habría que añadirle al principio el siguiente código:

```
.data
A: .asciiz " Pulsa teclas: \n"

.text

lui $t0,0xffff # Dirige del registro de control
lw $t1,0($t0) # Registre de control del receptor
ori $t1,$t1,0x0002 # Habilitar interrupciones del teclado
sw $t1,0($t0) # Actualizamos registro de control

mfc0,$a0,$12 # leer registro Status
ori $a0,0xff1
mtc0 $a0,$12

la $a0,A
li $v0,4
syscall

inicio:
    move $v0,$a0
    la $s1,'\n'
    beq $v0,$s1, fin
    j inicio
fin:
li $v0,11
syscall
```

Y para que la rutina de datos desde la linea 17-23

```
1  # Reserva de espacio para guardar registros en kdata
2  .kdata
3  contexto: .word 0,0,0,0 # espacio para alojar cuatro registros
4  .ktext 0x80000180 # Dirección de comienzo de la rutina
5
6  # Guardar registros a utilizar en la rutina.
7  la $k1, contexto
8  sw $at, 0($k1) # Guardamos $at
9  sw $t0, 4($k1)
10 sw $v0, 8($k1)
11 sw $a0, 12($k1)
12 #Comprobación de si se trata de una interrupción
13 mfc0 $k0, $13 # Registro Cause
14 srl $a0, $k0, 2 # Extraemos campo del código
15 andi $a0, $a0, 0x1f
16 bne $a0, $zero, acabamos # Sólo procesamos aquí E/S
17 #Tratamiento de la interrupción
18 li $t0,0xffff0000
19 lb $a0,4($t0) #Lee carácter del teclado
20 jal putc #Mostrar un registro leido
21 li $v0,11
22 syscall
23 # Antes de acabar se podría dejar todo iniciado:
24 acabamos: mtc0 $0, $13 # Iniciar registro Cause
25 mfc0 $k0, $12 # Leer registre Status
26 andi $k0, 0xfffd # Iniciar bit de excepción
27 ori $k0, 0x11 # Habilitar interrupciones
28 mtc0 $k0, $12 # reescribir registre Startus
29 # Restaurar registros
30 lw $at, 0($k1) # Recupero $at
31 lw $t0, 4($k1)
32 lw $v0, 8($k1)
33 lw $a0, 12($k1)
34 ## SI TENIENDO EN CUENTA QUE EL EPC NO SE INCREMENTA, PERO EN EXCEPCIONES SI.
35 # Devolver en el programa de usuario
36 putc:
37     lw $t1,8($t0) #registro control
38     andi $t1,$t1,0x0001 #bit de ready SINCRONIZACIÓN
39     beq $t1,$0,putc
40     sw $a0,12($t0) #TRANSFERENCIA
41     eret
42
```

Una prueva seria la siguiente:



Como vemos al meter el salto de línea el código finaliza

Escribe una rutina general de tratamiento de excepciones que permita tratar excepciones por desbordamiento aritmético, error por lectura al intentar el acceso a una dirección no alineada e interrupciones de teclado. En los tres casos se tiene que escribir un mensaje en la consola del MARS de la excepción tratada. Escribe el programa de prueba apropiado para probar los tres casos.

Esta seria la ruta de tratamiento

```
#############################################################
##        ÁREA DE DATOS DE LA RUTINA DE TRATAMIENTO        ##
#############################################################
.kdata
registros: .word 0,0,0,0 # Espacio para guardar 4 registros
mis1:.asciiz"\nExcepcion dirección errónea ocurrida en la dirección:"
mis2:.asciiz "\nExcepcion desbordamiento ocurrida en la dirección: "
mis3:.asciiz "\nEn cualquier caso continuamos el programa..."
mis4: .asciiz "\nExcepción por dirección no alineada "
#################################################################
##  EMPIEZA CÓDIGO DE LA RUTINA DE TRATAMIENTO De EXCEPCIONES  ##
#################################################################
```

```
.ktext 0x80000180 # Dirección de comienzo de la rutina
# Salvar los registros a utilizar
la $k1,registros
sw $at,0($k1) # Es importante guardar el registro $at
sw $v0,4($k1)
sw $a0,8($k1)
mfc0 $a0,$13 # $a0 <= registro Cause
andi $a0,$a0, 0x3C # extraemos en $a0 el código de excepción
#Detectamos sólo tres excpciones
li $s0,0x0030 # código Desbordamiento
li $s1,0x0014 # código error de dirección store
li $s2,0x00000024 #código dirección no alineada
andi $s2,$s2,100
beq $a0,$s0,Desbordo
bne $a0,$s1,salida
la $a0,mis1
li $v0,4
syscall
mfc0 $a0,$14 # $a0 <= EPC, donde ha ocurrido la excepción
li $v0,34
syscall # Escribimos EPC en hexadecimal
Desbordo:
    la $a0,mis2
    li $v0,4
    syscall

    mfc0 $a0, $14 # $a0 <= EPC, donde ha ocurrido la excepción
    li $v0,34
    syscall # Escribimos EPC en hexadecimal
salida:
    la $a0,mis3
    li $v0,4
    syscall
#Restauramos los registros
la $k1,registros
lw $at,0($k1)
lw $v0,4($k1)
lw $a0,8($k1)
#Iniciamos registro Vaddr del coprocesador 0
mtc0 $zero, $8
#Cómo se trata de excepciones se actualiza el registro EPC
mfc0 $k0, $14 # $k0 <= EPC
addiu $k0, $k0, 4 # Incremento de $k0 en 4
mtc0 $k0, $14 # Ahora EPC apunta a la siguiente instrucción
eret # Vuelve al programa de usuario
```

Para forzar las excepciones ponemos esto:

```
9  .text
0  li $t0, 0x7FFFFFFF
1  addiu $t1, $t0,1 #Se ignora el desbordamiento
2  addi $t2, $t0, 1 #Detecta el desbordamiento
3
```

```
Excepcion desbordamiento ocurrida en la dirección: 0x0040000c
En cualquier caso continuamos el programa...
-- program is finished running (dropped off bottom) --
```

```
eret # vuelve al programa de usuari
.text
# Excepción por dirección errónea
li $a0, 5
sw $a0, 124($zero)
```

```
Excepcion dirección errónea ocurrida en la dirección:0x00400004
En cualquier caso continuamos el programa...
-- program is finished running (dropped off bottom) --
```

```
# Excepción por dirección no alineada en Load
.data
vector: .word 1, 3, 5, 7, 11, 13
.text

la $t0, vector
lw $t0, 3($t0)
```

```
Excepción por dirección no alineada
En cualquier caso continuamos el programa...
-- program is finished running (dropped off bottom) --
```