Práctica 10

- Cuestión 4.

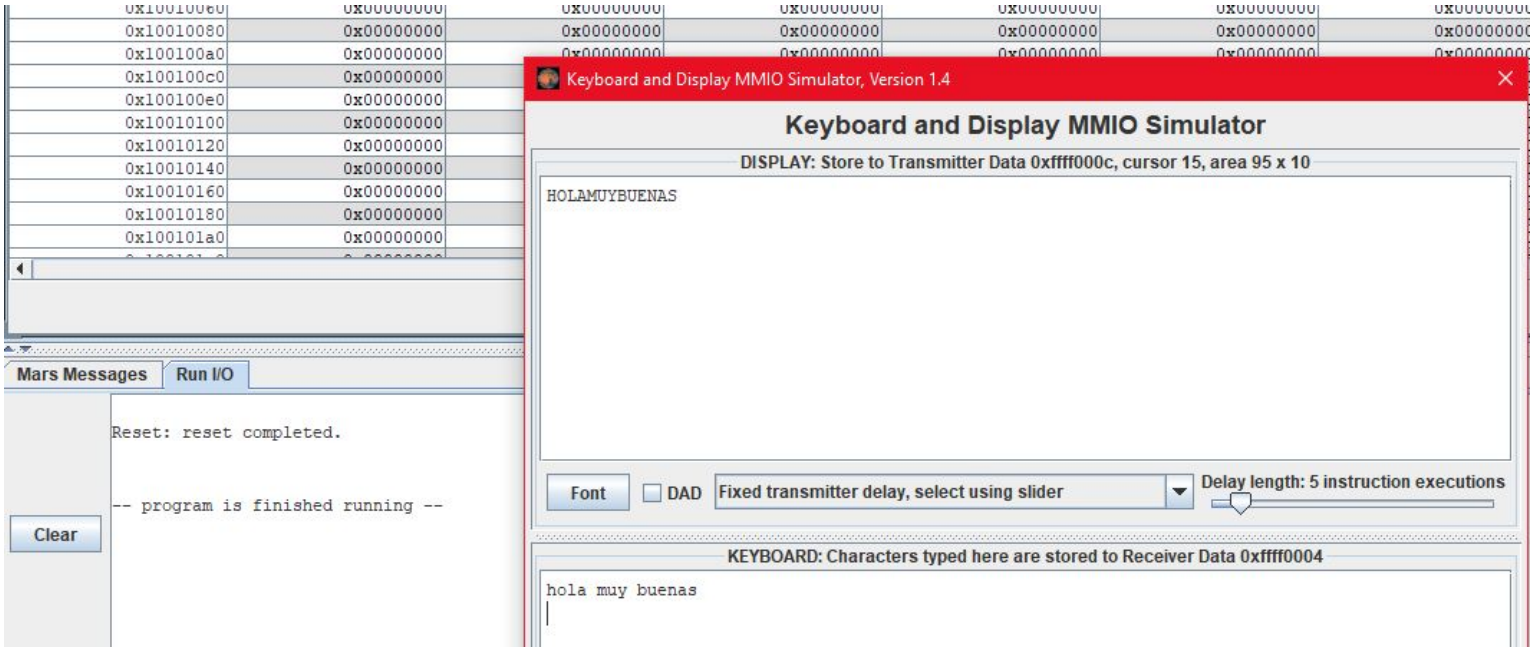
➢ Transforma el programa echo de la cuestión 3 en el programa caps que muestra por la consola la mayúscula del carácter introducido por el teclado. Supón que todos los caracteres introducidos están en minúscula.

Explicación: *Usando los primeros ejemplos, podemos fácilmente concatenarlos para leer y escribir en la pantalla, y en medio de ambas llamadas, comprobar si se ha recibido un intro, de ser así se saldrá del programa, y de no serlo, se convertirá en mayúscula (restar 32 al valor ascii), se imprimirá, y volverá a esperar a que el usuario inserte un carácter.*

```
# Cuestion 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.

#########################
#         DATOS         #
#########################
.data
.eqv ControlTeclado 0  # Constantes.
.eqv BufferTeclado 4
.eqv ControlDisplay 8
.eqv BufferDisplay 12

#########################
#     INSTRUCCIONES     #
#########################
.text
# SELECCIÓN:
jal setInitialValues
lui $t0, 0xffff # Direc. del registro de control del teclado
```

```
r_espera:
  lw $t2, ControlTeclado($s0) # Lee registro control del teclado
  andi  $t2, $t2, 1 # SINCRONIZACIÓN: Extrae el bit de ready
beqz $t2, r_espera  # Si cero no hay carácter, continuamos esperando

jal getc

beq $v0, $s1, exit

jal toUppercase

w_espera:
  lw $t1, ControlDisplay($s0) # registro control
  andi $t1, $t1, 1       # SINCRONITZACIÓN: bit de ready Sincroniza
beq $t1, $0, w_espera

jal putc

j r_espera


############################
#        SECCIONES         #
############################

setInitialValues:
la $s0, 0XFFFF0000  # Direcciones de inicio de E/S
li $s1, '\n'        # Codigo ascii de retorno de carro
jr $ra

getc:
lw $v0, BufferTeclado($s0) # Lee registro de datos del teclado, Codigo ascii de tecla
guardado en $v0
jr $ra

putc:
sw $a0, BufferDisplay($s0) # Escribe en la consola
jr $ra
```

```
############################
#         FUNCIONES        #
############################


# Pasa un valor entero representando un caracter ASCII a su valor entero mayuscular.
toUppercase:
subi $a0, $v0, 32     # Le restamos 32 para que se convierta en minuscula y lo dejamos en
$a0.
jr $ra

# Cerramos el programa
exit:
li $v0, 10            # Función 10. Acaba programa
syscall
```



(Los espacios no aparecen ya que se le resta 32 a todos los caracteres introducidos)

- Cuestión 5.

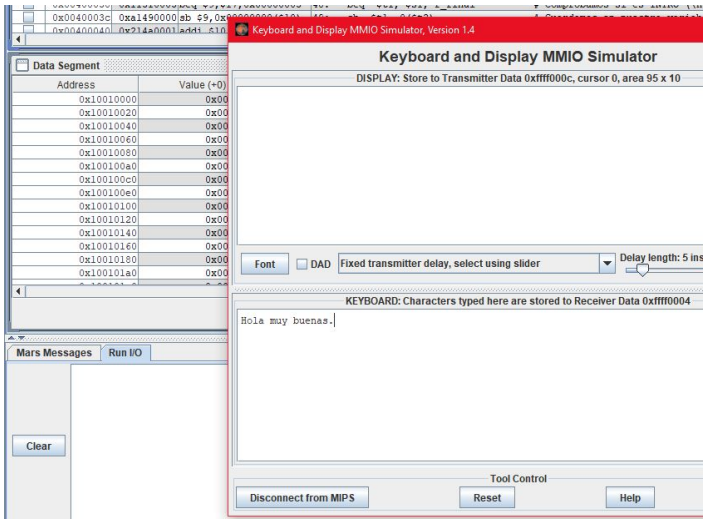
➢ Complétalo escribiendo la función read_string. Esta función tiene que leer del teclado la cadena de caracteres que introduzca el usuario y tiene que almacenarla en un buffer denominado cadena. La cadena finaliza cuando el usuario teclee un salto de línea. Posteriormente el programa muestra la cadena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

Explicación: *La lógica detrás de "read_string" es muy similar a la que ya se da en el código de partida para "print_string", lo único que hay que cambiar es el orden de ciertas instrucciones, y las instrucciones en sí, pero el esquema es muy similar al "print_string". Read_string: Comprobamos cuando el teclado está disponible para su lectura (cuando un carácter ha sido introducido), y cargamos el carácter desde el registro de memoria asignado a entrada/salida, comprobamos si es un intro, si lo es, pasamos a "print_string", si no lo es, lo guardamos en nuestra variable "cadena" y aumentamos la posición del registro que usamos para acceder a "cadena" y volvemos a esperar a que el siguiente carácter sea introducido.*

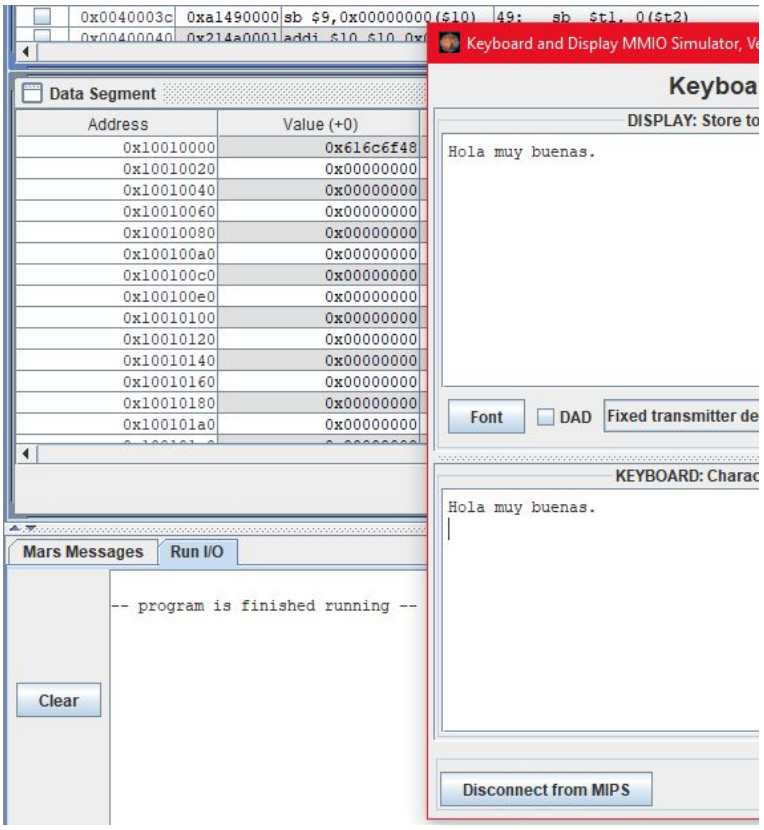
```
#######################################
#                                     #
# Código de partida de la cuestión 5  #
#          print string               #
#                                     #
#######################################
# Complétalo escribiendo la función read_string. Esta función 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.
```

```
.data
cadena: .space 32
.eqv ControlTeclado 0
.eqv BufferTeclado 4
.eqv ControlDisplay 8
.eqv BufferDisplay 12

.text
jal setInitialValues
jal read_string
jal print_string

jal exit
###################################
#             SECCIONES           #
###################################
setInitialValues:
la $s0, 0xFFFF0000      # Direcciones de inicio de E/S
li $s1, '\n'            # Codigo ascii de retorno de carro
jr $ra

###################################
#             Funcions             #
###################################
read_string:
la $t2, cadena
r_sync:                         # Revisamos cuando tenemos un caracter leeido.
  lw   $t1, ControlTeclado($s0)
  andi $t1, $t1, 1              # Si se ha leeido, el bit estará a 1, comprobamos.
beqz $t1, r_sync
  lw   $t1, BufferTeclado($s0)  # Obtenemos el caracter.
  beq  $t1, $s1, r_final        # Comprobamos si es INTRO (\n)
  sb  $t1, 0($t2)               # Guardamos en nuestra variable.
  addi $t2, $t2, 1              # Aumentamos el salto de registro para concatenar el
siguiente caracter.
j r_sync
r_final:
jr $ra
```

```
print_string:
la $t2, cadena
w_sync:
  lw   $t1, ControlDisplay($s0)
  andi $t1, $t1, 1
beqz $t1, w_sync
  lbu  $t1, 0($t2)
  beqz $t1, w_final
  sw   $t1, BufferDisplay($s0)
  addi $t2, $t2, 1
j w_sync
w_final:
jr $ra

exit:
li  $v0, 10
syscall
```



(1ra foto, sin darle a intro)



(2nda foto, después de darle a intro)

Práctica 11

- Cuestión 7.

Supón que el contenido del registro Cause ($13) tiene los siguientes valores
después de haberse producido una excepción.

Explicación: *Obtienes los últimos 2 bytes, los separas en 2 grupos de 4 bits, y obtienes los bits desde la posición 6 a la 2 (incluidas), el número en binario de ese grupo de bits (de 6 a 2) es el código de causa de excepción.*

Cause	Fuente de la excepción
0x00000000	
0x00000020	
0x00000024	
0x00000028	
0x00000030	

```
Cuestión 7.

Supón que el contenido del registro Cause ($13) tiene los siguientes valores
después de haberse producido una excepción.

-------------------------------------------------------------------------------------------------
| Cause                                                  | Fuente de la excepción
|-----------------------------------------------------------------------------------------------
|                              (7 [65432] 10) <- del 7 al 0, Ultimos 2 bytes, Ultimos 8 bits.
|                               -  -----  --
| 0x00000000 -> 0x000000 (00) -> (0 [00000] 00) -> [0]  -> Int
| 0x00000020 -> 0x000000 (20) -> (0 [01000] 00) -> [8]  -> SyS0
| 0x00000024 -> 0x000000 (24) -> (0 [01001] 00) -> [9]  -> Bp
| 0x00000028 -> 0x000000 (28) -> (0 [01010] 00) -> [10] -> RI
| 0x00000030 -> 0x000000 (30) -> (0 [01100] 00) -> [12] -> Ov
|-----------------------------------------------------------------------------------------------
| Número | Nombre   | Causa de la excepción
|-----------------------------------------------------------------------------------------------
| 0      | Int      | Interrupción (Hardware)
| 4      | AdEL     | Excepción por dirección errónea (Load o busca de instrucción)
| 5      | AdES     | Excepción por dirección errónea (Store)
| 8      | SyS0     | Excepción syscall
| 9      | Bp       | Excepción por punto de ruptura (breakpoint)
| 10     | RI       | Excepción por instrucción reservada
| 12     | Ov       | Excepción por desbordamiento aritmético
| 13     | Tr       | Trap
| 15     | FPE      | Excepción de coma flotante
-------------------------------------------------------------------------------------------------

( 0x000000 (10) ->  0 [00100] 00 -> [4] -> AdEL <- Para la Cuestion 12 )
```

- Cuestión 11.

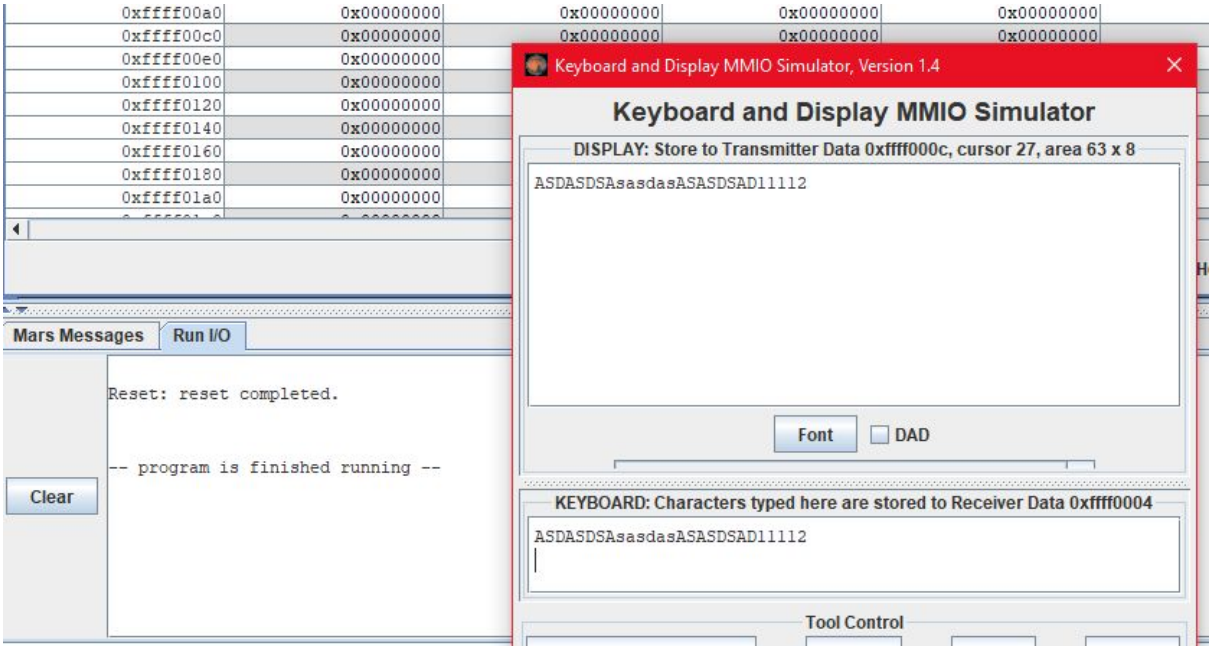
➢ Modifica la rutina de tratamiento de interrupciones para que escriba en el display del transmisor el carácter leído en el receptor. Haz que guarde en el registro $v0 el carácter leído. Escribe un programa principal apropiado para hacer pruebas que finalice cuando en el receptor se pulse un salto de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

Explicación: *Basándonos en el ejemplo que ya nos dan, solo tenemos que cambiar donde se imprime el carácter, guardar el carácter en la pos. de memoria donde se ha guardado $v0 para que cuando acabe se guarde en $v0, habilitar la captura de señales y la emisión de señales, y crear un bucle infinito en el programa principal que salga de él solo sí se ha introducido un intro.*

```
# Cuestion 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.

# Reserva de espacio para guardar registros en kdata
.kdata
contexto: .word 0,0,0,0    # espacio para alojar cuatro registros

.ktext 0x80000180          # Dirección de comienzo de la rutina
# Guardar registros a utilizar en la rutina.
la $k1, contexto
sw $at, 0($k1)             # Guardamos $at
sw $t0, 4($k1)
sw $v0, 8($k1)
sw $a0, 12($k1)
```

```
# Comprobación de si se trata de una interrupción, si no lo es acabamos.
mfc0 $k0, $13 # Registro Cause
srl $a0, $k0, 2 # Extraemos campo del código
andi $a0, $a0, 0x1f
bne $a0, $zero, acabamos # Solo procesamos aqui E/S

# Tratamiento de la interrupción
li $t0, 0xffff0000        # Registro de control del teclado
lb $a0, 4($t0)            # Lee carácter del teclado
beq $a0, $s1, exit
sw $a0, 12($t0)            # Escribe en la consola
sw $a0, 8($k1)             # Guardamos en 8(k1), cuando acabe, estará disponible en $v0


# Antes de acabar se podría dejar todo iniciado:
acabamos: mtc0 $0, $13    # Iniciar registro Cause
mfc0 $k0, $12             # Leer registre Status
andi $k0, 0xfffd          # Iniciar bit de excepción
ori  $k0, 0x11            # Habilitar interrupciones
mtc0 $k0, $12             # reescribir registre Startus

# Restaurar registros
lw $at, 0($k1)            # Recupero $at
lw $t0, 4($k1)
lw $v0, 8($k1)
lw $a0, 12($k1)

# Devolver en el programa de usuario
eret

.text
li $s1, '\n'        # Codigo ascii de retorno de carro

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 registre Status
ori $a0, 0xff11    # Habilitar todas las interrupciones
mtc0 $a0, $12      # reescribir el registro status

bucle:

jal bucle

exit:
li $v0, 10
syscall # syscall 10 (exit)
```



- Cuestión 12

➢ Escribe una rutina general de tratamiento de excepciones que permite 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.

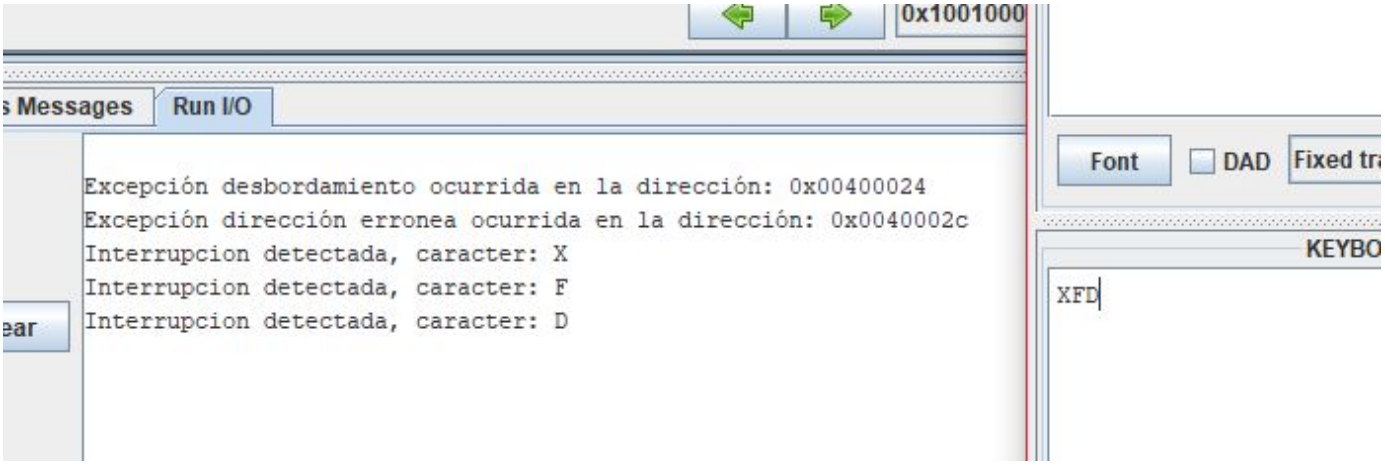
Explicación: *Basándonos en el ejemplo de tratamiento de excepciones, solo tenemos que añadir lo anteriormente hecho en la Cuestión 11, un mensaje extra y mostrar los mensajes dependiendo del codigo de excepcion hallado en el Cause, y tener en cuenta la forma de acabar el tratamiento dependiendo si es una cuestión o una interrupción.*

```
# Cuestión 12

# 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.

.kdata
registros: .word 0, 0, 0, 0 # Espacio para guardar 4 registros
mis1:.asciiz "\nExcepcion direccion erronea ocurrida en la direccion: "
mis2:.asciiz "\nExcepcion desbordamiento ocurrida en la direccion: "
mis3:.asciiz "\nInterrupcion detectada, caracter: "

.ktext 0x80000180     # Dirección de comienzo de la rutina
# Salvar los registros a utilizar
la $k1, registros
sw $at, 0($k1)
sw $t0, 4($k1)
sw $v0, 8($k1)
sw $a0, 12($k1)
```

```
mfc0 $a0, $13          # $a0 <= registro Cause
andi $a0, $a0, 0x3C    # extraemos en $a0 el código de excepción

# Detectamos sólo dos excpcions
li $s0, 0x0030         # código Desbordamiento
li $s1, 0X0010         # código error de dirección load

beq $a0, $s0, Desbordo
beq $a0, $s1, Lectura
beq $a0, $zero, Interrupcion

# Excepcion Lectura.
Lectura:
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

j FinExcepcion

# Excepcion Desbordo.
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

j FinExcepcion

# Interrupcion teclado.
Interrupcion:
la $a0, mis3
li $v0, 4
syscall
```

```
li $t0, 0xffff0000
lb $a0, 4($t0)        # Escribimos la tecla que se ha pulsado
li $v0, 11
syscall

j FinInterrupcion


# Iniciamos registro Vaddr del coprocesador 0
mtc0 $zero, $8

FinExcepcion:
# Restauramos los registros
la $k1, registros
lw $at, 0($k1)
lw $t0, 4($k1)
lw $v0, 8($k1)
lw $a0, 12($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

FinInterrupcion:
# Restauramos los registros
la $k1, registros
lw $at, 0($k1)
lw $t0, 4($k1)
lw $v0, 8($k1)
lw $a0, 12($k1)
```

```
mtc0 $0, $13              # Iniciar registro Cause
mfc0 $k0, $12             # Leer registre Status
andi $k0, 0xfffd          # Iniciar bit de excepción
ori  $k0, 0x11            # Habilitar interrupciones
mtc0 $k0, $12             # reescribir registro Startus

eret                      # Vuelve al programa de usuario


.text

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

mfc0 $a0, $12     # leer registre status
ori $a0, 0xff11   # Habilitar todas las interrupciones
mtc0 $a0, $12     # reescribir el registro status




# Prueba Desbordo
li $t0, 0x7FFFFFFF
addi $t2, $t0, 1 #Detecta el desbordamiento

# Prueba Lecutra
li $t0, 0xFFFFFFFF
lw $t1, 0($t0)

# Prueba Iterrupcion teclado
bucle:

jal bucle

exit:
li $v0, 10
syscall # syscall 10 (exit)
```


0x1001000
Messages
Run I/O
Excepción desbordamiento ocurrida en la dirección: 0x00400024
Excepción dirección erronea ocurrida en la dirección: 0x0040002c
Interrupcion detectada, caracter: X
Interrupcion detectada, caracter: F
Interrupcion detectada, caracter: D
Font
DAD
KEYBO
XFD