Práctica 10

Entrada/ salida 1

Jordi Blasco Lozano

Arquitectura de comuputadores

Grado en Inteligencia Artificial

## Indice:

[Indice: 2](#_Toc165394761)

[1. Actividad 1 3](#_Toc165394762)

## Actividad 1

**A screenshot of a computer

Description automatically generatedEnsambla el programa ejemplo de la actividad 1 pero antes de ejecutarlo pulsa en la opción connect to MIPS del simulador del teclado y pantalla**

**Haz diversas pruebas hasta que comprendas el funcionamiento del programa. Tendrás que teclear un carácter dentro del área de la ventana inferior. Comprobarás el carácter introducido mirando lo registro $v0.**

El carácter del teclado se mostrará en $v0 en formato ascii, es decir m = 109

**Ejecuta de nuevo el programa pero ahora disminuye su velocidad de ejecución,por ejemplo a 15 instrucciones por segundo.**

Ahora podemos ver claramente como itera en el bucle y incrimenta el contador hasta que le escribamos una letra

**Haz pruebas observando el segmento MMIO de la memoria.**

**Observa el contador para comprobar la diferencia de velocidad del programa y el usuario.**

Podemos comprobar que incluso escribiendo la letra rapido el contador se desplaza mucho, esto es debido a lavelocidad del procesador y al no limitarlo $t1 se incrementa muy rápido

Con limitador de instrucciones sin limitador de instrucciones

**Elimina momentáneamente la instrucción que lee el carácter del registro de datos del teclado y comprueba que el bit de ready permanece con el valor 1. Sólo pasará a cero si el programa lee el carácter.**

Al eliminar lw $v0, 4($t0) el bit de ready permanecerá a 1 ya que no lee el caracter

****

## Actividad 2

**Comprueba las similitudes del código de la actividad 2 con la de leer del teclado de la actividad 1.**

Ambos códigos tienen estructuras similares ya que usan instrucciones de lectura y escritura pero la actividad 1 se enfocaba en leer un carácter del teclado y la dos en escribirlo en la consola

**¿Cómo cambiaría el código si sustituyéramos la primera instrucción por le 0xffff0008?**

Al modificar la dirección base que se utiliza para acceder al registro de control del teclado provoca una excepción y el programa deja de funcionar



## Cuestión 1

# Actividad 1: Leer del teclado

lui $t0, 0xffff # Dirección del registro de control del teclado

li $t1, 0 # Inicia un contador de espera

b\_espera:

lw $t2, ($t0) # Lee registro control del teclado

# SINCRONIZACIÓN:

andi $t2, $t2, 1 # Extrae el bit de ready

addiu $t1, $t1, 1 # Incrementa el contador

#(cuenta las iteraciones)

beqz $t2, b\_espera # Si cero no hay carácter

# continuamos esperando

# TRANSFERENCIA:

lw $v0, 4($t0) # Lee el carácter del registro de datos

# PROCESAMIENTO (ACTIVIDAD 2):

# Escribir en la consola

lui $t0, 0xffff # Dirección del registro de control de la consola

b\_espera\_consola:

lw $t1, 8($t0) # Registro de control de la consola

# SINCRONIZACIÓN:

andi $t1, $t1, 0x0001 # Bit de ready de la consola

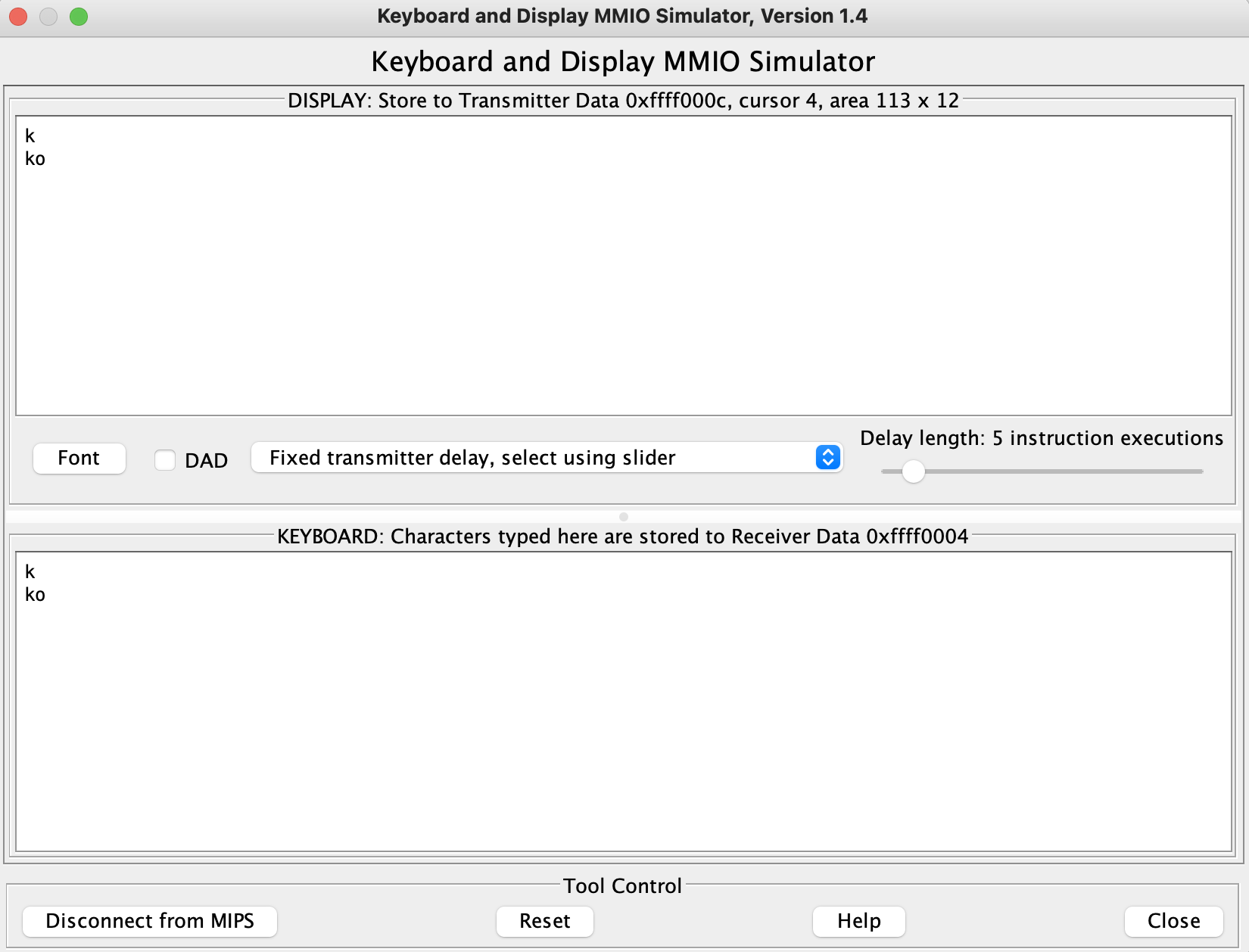
beq $t1, $0, b\_espera\_consola

# TRANSFERENCIA:

move $a0, $v0 # Mover el carácter leído a $a0 para escribirlo

sw $a0, 12($t0) # Escribe en la consola

j b\_espera # Regresa al bucle de espera para leer otro carácter



## Cuestión 2

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

.data

inCtrlReg: .word 0xffff0000

.text

# Función getc - Leer un carácter del teclado

# Función principal

main:

# Llama a la función getc para leer un carácter del teclado

jal getc

# Mueve el carácter leído ($v0) al argumento de la función putc ($a0)

move $a0, $v0

# Llama a la función putc para escribir el carácter leído en el teclado

jal putc

getc:

la $t0, inCtrlReg

lw $t1, 0($t0)

etiqueta: lw $t2, 0($t1)

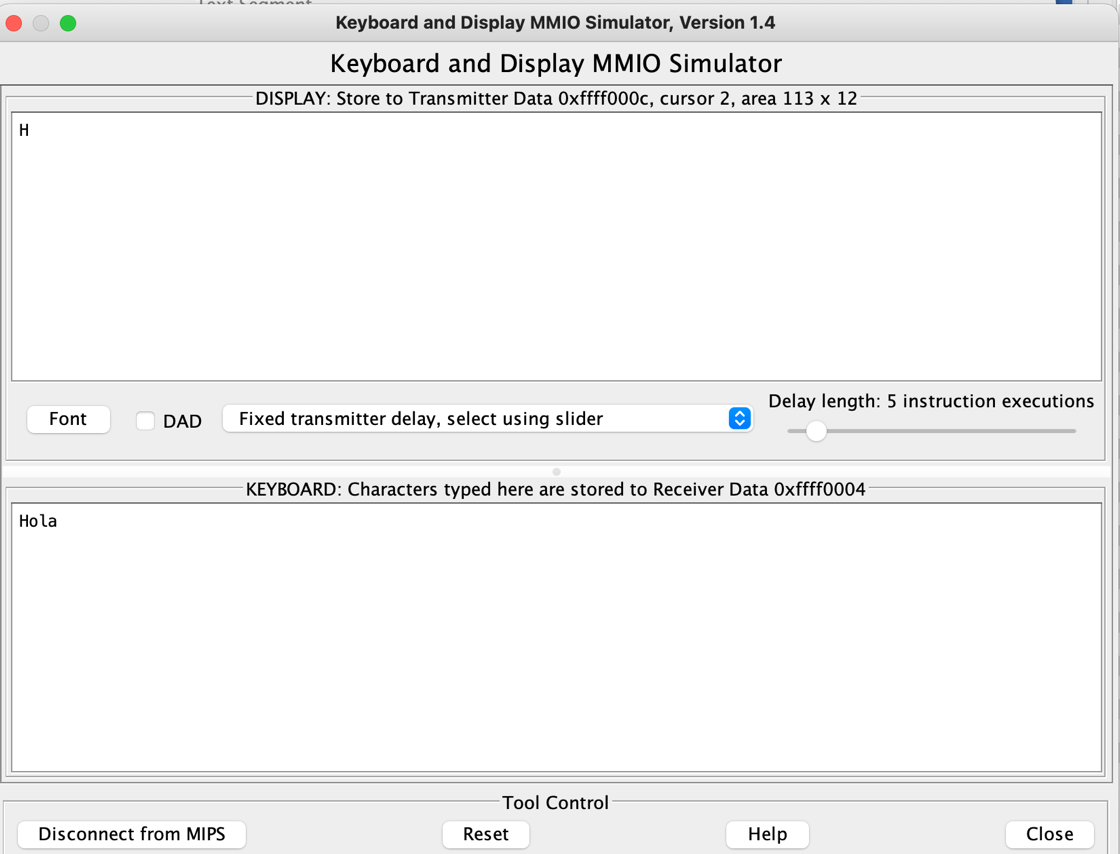
andi $t2, $t2, 1

beq $t2, 0, etiqueta

lw $v0, 4($t1)

jr $ra

**Prueba su funcionamiento y comprueba cómo varían los contenidos de los registros y la memoria MMIO.**

****



La H mayuscula en ascii es 72 que sería la primera letra que hemos puesto, pues el programa solo muetra la primer carácter que le pasemos pero sigue recogiendo caracteres de los siguientes caracteres que pongamos, por ejemplo la a minúscula que en ascii es 97

## Cuestión 3

**Diseña un programa ECHO (eco). Para lo cual simplemente debes iterar el código de la cuestión 2 hasta que el carácter introducido sea un salto de línea (‘/n’).**

.data  
inCtrlReg: .word 0xffff0000  
outCtrlReg: .word 0xffff0000  
barra: .asciiz "/n"  
  
.text  
# Función getc - Leer un carácter del teclado  
  
# Función principal  
main:  
    la $s0, barra  
    lw $s0, 0($s0)  
    # Llama a la función getc para leer un carácter del teclado  
    loop: jal getc  
    # Mueve el carácter leído ($v0) al argumento de la función putc ($a0)  
    move $a0, $v0  
    # Llama a la función putc para escribir el carácter leído en el teclado  
    jal putc  
    beq $a0, $s0, end  
    j loop  
     
getc:  
    la $t0, inCtrlReg  
    lw $t1, 0($t0)  
    etiqueta: lw $t2, 0($t1)  
    andi $t2, $t2, 1  
    beq $t2, 0, etiqueta  
    lw $v0, 4($t1)  
    jr $ra  
  
# Función putc - Escribir un carácter en el teclado  
putc:  
    la $t0, inCtrlReg  
    lw $t1, 0($t0)  
    etiqueta2: lw $t2, 8($t1)  
    andi $t2, $t2, 1  
    beq $t2, 0, etiqueta2  
    sw $a0, 12($t1)  
    jr $ra        # Retorna al llamador  
  
  
end:  
    li $v0, 10   # Código de la llamada al sistema para salir del programa  
    syscall

**Para probar el programa puedes activar previamente la casilla DAD. Esta casilla controla el retraso, en número de instrucciones, en las que aparecerá el carácter en la consola desde que se escribe en el registro de datos.**