PRACTICA 6

Procesador RISC: utilizando la E/S

Objetivos: Familiarizarse con el desarrollo de programas para procesadores con sets reducidos de instrucciones (RISC). Resolver problemas y verificarlos a través de simulaciones. Dominar el uso del puerto de E/S provisto en el WinMIPS64.

Puerto de E/S mapeado en memoria de datos – Resumen

CONTROL y DATA son direcciones de memoria fijas para acceder al puerto de E/S. Aplicando distintos comandos a través de CONTROL, es posible producir salidas o ingresar datos a través de la dirección DATA. Las direcciones de memoria de CONTROL y DATA son 0x10000 y 0x10008 respectivamente. Como el set de instrucciones del procesador MIPS64 no cuenta con instrucciones que permitan cargar un valor inmediato de más de 16 bits (como es el caso de las direcciones mencionadas), resulta conveniente definirlas en la memoria de datos, así:

```
CONTROL: .word32 0x10000
   DATA: .word32 0x10008
```

De esa manera, resulta sencillo cargar esas dos direcciones en registros mediante:

```
    lwu  $s0, DATA($zero)     ; Carga el valor 0x10000 en $s0
    lwu  $s1, CONTROL($zero)  ; Carga el valor 0x10008 en $s1
```

- Si se escribe en DATA un número entero y se escribe un 1 en CONTROL, se interpretará el valor escrito en DATA como un entero sin signo y se lo imprimirá en la pantalla alfanumérica de la terminal.
- Si se escribe en DATA un número entero y se escribe un 2 en CONTROL, se interpretará el valor escrito en DATA como un entero con signo y se lo imprimirá en la pantalla alfanumérica de la terminal.
- Si se escribe en DATA un número en punto flotante y se escribe un 3 en CONTROL, se imprimirá en la pantalla alfanumérica de la terminal el número en punto flotante.
- Si se escribe en DATA la dirección de memoria del comienzo de una cadena terminada en 0 y se escribe un 4 en CONTROL, se imprimirá la cadena en la pantalla alfanumérica de la terminal.
- Si se escribe en DATA un color expresado en RGB (usando 4 bytes para representarlo: un byte para cada componente de color e ignorando el valor del cuarto byte), en DATA+4 la coordenada Y, en DATA+5 la coordenada X y se escribe un 5 en CONTROL, se pintará con el color indicado un punto de la pantalla gráfica de la terminal, cuyas coordenadas están indicadas por X e Y. La pantalla gráfica cuenta con una resolución de 50x50 puntos, siendo (0, 0) las coordenadas del punto en la esquina inferior izquierda y (49, 49) las del punto en la esquina superior derecha.
- Si se escribe un 6 en CONTROL, se limpia la pantalla alfanumérica de la terminal.
- Si se escribe un 7 en CONTROL, se limpia la pantalla gráfica de la terminal.
- Si se escribe un 8 en CONTROL, se permitirá ingresar en la terminal un número (entero o en punto flotante) y el valor del número ingresado estará disponible para ser leído en DATA.
- Si se escribe un 9 en CONTROL, se esperará a que se presione una tecla en la terminal (la cuál no se mostrará al ser presionada) y el código ASCII de dicha tecla estará disponible para ser leído en DATA.

1) El siguiente programa produce la salida de un mensaje predefinido en la ventana Terminal del simulador WinMIPS64. Teniendo en cuenta las condiciones de control del puerto de E/S (en el resumen anterior), modifique el programa de modo que el mensaje a mostrar sea ingresado por teclado en lugar de ser un mensaje fijo.

```
        .data
texto:  .asciiz  "Hola, Mundo!"     ; El mensaje a mostrar
CONTROL: .word32  0x10000
DATA:   .word32  0x10008

        .text
        lwu    $s0, DATA($zero)     ; $s0 = dirección de DATA
        daddi  $t0, $zero, texto    ; $t0 = dirección del mensaje a mostrar
        sd     $t0, 0($s0)          ; DATA recibe el puntero al comienzo del mensaje

        lwu    $s1, CONTROL($zero)  ; $s1 = dirección de CONTROL
        daddi  $t0, $zero, 6        ; $t0 = 6 -> función 6: limpiar pantalla alfanumérica
        sd     $t0, 0($s1)          ; CONTROL recibe 6 y limpia la pantalla

        daddi  $t0, $zero, 4        ; $t0 = 4 -> función 4: salida de una cadena ASCII
        sd     $t0, 0($s1)          ; CONTROL recibe 4 y produce la salida del mensaje
        halt
```

2) Escriba un programa que utilice sucesivamente dos subrutinas: La primera, denominada `ingreso`, debe solicitar el ingreso por teclado de un número entero (de un dígito), verificando que el valor ingresado realmente sea un dígito. La segunda, denominada `muestra`, deberá mostrar en la salida estándar del simulador (ventana Terminal) el valor del número ingresado expresado en letras (es decir, si se ingresa un '4', deberá mostrar 'CUATRO'). Establezca el pasaje de parámetros entre subrutinas respetando las convenciones para el uso de los registros y minimice las detenciones del cauce (ejercicio similar al ejercicio 6 de Práctica 2).

3) Escriba un programa que realice la suma de dos números enteros (de un dígito cada uno) utilizando dos subrutinas: La denominada `ingreso` del ejercicio anterior (ingreso por teclado de un dígito numérico) y otra denominada `resultado`, que muestre en la salida estándar del simulador (ventana Terminal) el resultado numérico de la suma de los dos números ingresados (ejercicio similar al ejercicio 7 de Práctica 2).

4) Escriba un programa que solicite el ingreso por teclado de una clave (sucesión de cuatro caracteres) utilizando la subrutina `char` de ingreso de un carácter. Luego, debe comparar la secuencia ingresada con una cadena almacenada en la variable `clave`. Si las dos cadenas son iguales entre si, la subrutina llamada `respuesta` mostrará el texto "Bienvenido" en la salida estándar del simulador (ventana Terminal). En cambio, si las cadenas no son iguales, la subrutina deberá mostrar "ERROR" y solicitar nuevamente el ingreso de la clave.

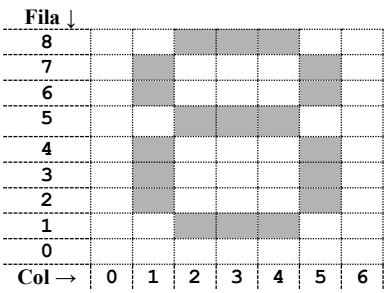
5) Escriba un programa que calcule el resultado de elevar un valor en punto flotante a la potencia indicada por un exponente que es un número entero positivo. Para ello, en el programa principal se solicitará el ingreso de la base (un número en punto flotante) y del exponente (un número entero sin signo) y se deberá utilizar la subrutina `a_la_potencia` para calcular el resultado pedido (que será un valor en punto flotante). Tenga en cuenta que cualquier base elevada a la 0 da como resultado 1. Muestre el resultado numérico de la operación en la salida estándar del simulador (ventana Terminal).

6) El siguiente programa produce una salida estableciendo el color de un punto de la pantalla gráfica (en la ventana Terminal del simulador WinMIPS64). Modifique el programa de modo que las coordenadas y color del punto sean ingresados por teclado.

```
          .data
coorX:    .byte   24                  ; coordenada X de un punto
coorY:    .byte   24                  ; coordenada Y de un punto
color:    .byte   255, 0, 255, 0      ; color: máximo rojo + máximo azul => magenta
CONTROL:  .word32 0x10000
DATA:     .word32 0x10008

          .text
          lwu    $s6, CONTROL($zero)  ; $s6 = dirección de CONTROL
          lwu    $s7, DATA($zero)     ; $s7 = dirección de DATA

          daddi  $t0, $zero, 7        ; $t0 = 7 -> función 7: limpiar pantalla gráfica
          sd     $t0, 0($s6)          ; CONTROL recibe 7 y limpia la pantalla gráfica

          lbu    $s0, coorX($zero)    ; $s0 = valor de coordenada X
          sb     $s0, 5($s7)          ; DATA+5 recibe el valor de coordenada X
          lbu    $s1, coorY($zero)    ; $s1 = valor de coordenada Y
          sb     $s1, 4($s7)          ; DATA+4 recibe el valor de coordenada Y
          lwu    $s2, color($zero)    ; $s2 = valor de color a pintar
          sw     $s2, 0($s7)          ; DATA recibe el valor del color a pintar

          daddi  $t0, $zero, 5        ; $t0 = 5 -> función 5: salida gráfica
          sd     $t0, 0($s6)          ; CONTROL recibe 5 y produce el dibujo del punto
          halt
```

7) Se desea realizar la demostración de la transformación de un carácter codificado en ASCII a su visualización en una matriz de puntos con 7 columnas y 9 filas. Escriba un programa que realice tal demostración, solicitando el ingreso por teclado de un carácter para luego mostrarlo en la pantalla gráfica de la terminal.

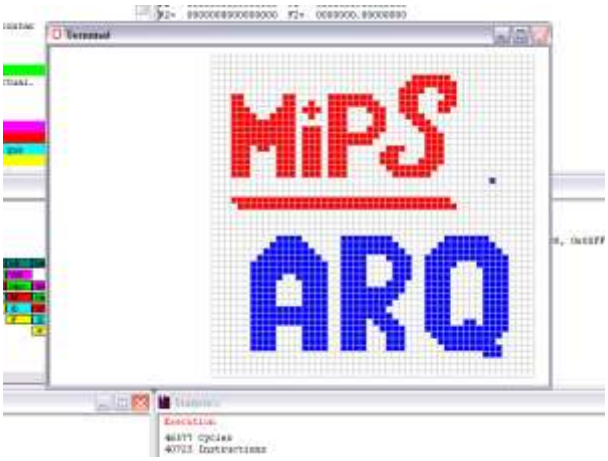
El carácter '8' es representado como:



8) El siguiente programa implementa una animación de una pelotita rebotando por la pantalla. Modifíquelo para que en lugar de una pelotita, se muestren simultáneamente varias pelotitas (cinco, por ejemplo), cada una con su posición, dirección y color particular.

```
        .data
CONTROL:        .word32 0x10000
DATA:           .word32 0x10008
color_pelota:   .word32 0x00FF0000  ; Azul
color_fondo:    .word32 0x00FFFFFF  ; Blanco

        .text
        lwu     $s6, CONTROL($zero)
        lwu     $s7, DATA($zero)
        lwu     $v0, color_pelota($zero)
        lwu     $v1, color_fondo($zero)
        daddi   $s0, $zero, 23      ; Coordenada X de la pelota
        daddi   $s1, $zero, 1       ; Coordenada Y de la pelota
        daddi   $s2, $zero, 1       ; Dirección X de la pelota
        daddi   $s3, $zero, 1       ; Dirección Y de la pelota
        daddi   $s4, $zero, 5       ; Comando para dibujar un punto
loop:   sw      $v1, 0($s7)      ; Borra la pelota
        sb      $s0, 4($s7)
        sb      $s1, 5($s7)
        sd      $s4, 0($s6)
        dadd    $s0, $s0, $s2    ; Mueve la pelota en la dirección actual
        dadd    $s1, $s1, $s3
        daddi   $t1, $zero, 48      ; Comprueba que la pelota no esté en la columna de más
        slt     $t0, $t1, $s0    ; a la derecha. Si es así, cambia la dirección en X.
        dsll    $t0, $t0, 1
        dsub    $s2, $s2, $t0
        slt     $t0, $t1, $s1    ; Comprueba que la pelota no esté en la fila de más arriba.
        dsll    $t0, $t0, 1      ; Si es así, cambia la dirección en Y.
        dsub    $s3, $s3, $t0
        slti    $t0, $s0, 1      ; Comprueba que la pelota no esté en la columna de más
        dsll    $t0, $t0, 1      ; a la izquierda. Si es así, cambia la dirección en X.
        dadd    $s2, $s2, $t0
        slti    $t0, $s1, 1      ; Comprueba que la pelota no esté en la fila de más abajo.
        dsll    $t0, $t0, 1      ; Si es así, cambia la dirección en Y.
        dadd    $s3, $s3, $t0
        sw      $v0, 0($s7)      ; Dibuja la pelota.
        sb      $s0, 4($s7)
        sb      $s1, 5($s7)
        sd      $s4, 0($s6)
        daddi   $t0, $zero, 500    ; Hace una demora para que el rebote no sea tan rápido.
demora: daddi   $t0, $t0, -1    ; Esto genera una infinidad de RAW y BTS pero...
        bnez    $t0, demora     ; ¡hay que hacer tiempo igualmente!
        j       loop
```

9) Escriba un programa que le permita dibujar en la pantalla gráfica de la terminal. Deberá mostrar un cursor (representado por un punto de un color particular) que pueda desplazarse por la pantalla usando las teclas ‘a’, ‘s’, ‘d’ y ‘w’ para ir a la izquierda, abajo, a la derecha y arriba respectivamente. Usando la barra espaciadora se alternará entre modo desplazamiento (el cursor pasa por arriba de lo dibujado sin alterarlo) y modo dibujo (cada punto por el que el cursor pasa quedará pintado del color seleccionado). Las teclas del ‘1’ al ‘8’ se usarán para elegir uno entre los ocho colores disponibles para pintar.



Observaciones: Para poder implementar este programa, se necesitará almacenar en la memoria la imagen completa de la pantalla gráfica.

Si cada punto está representado por un byte, se necesitarán 50x50x1 = 2500 bytes. El simulador WinMIPS64 viene configurado para usar un bus de datos de 10 bits, por lo que la memoria disponible estará acotada a 2^{10}=1024 bytes.

Para poder almacenar la imagen, será necesario configurar el simulador para usar un bus de datos de 12 bits, ya que 2^{12}=4096 bytes, los que si resultarán suficientes. La configuración se logra yendo al menú “Configure → Architecture” y poniendo “Data Address Bus” en 12 bits en lugar de los 10 bits que trae por defecto.