

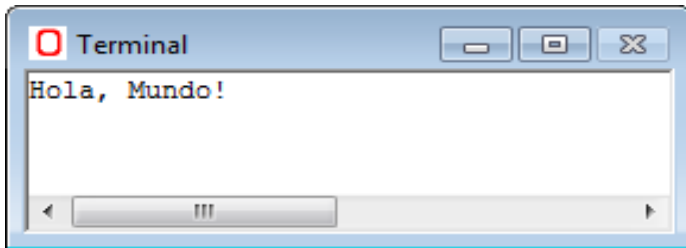
Winmips – RISC parte 4

Entrada - Salida

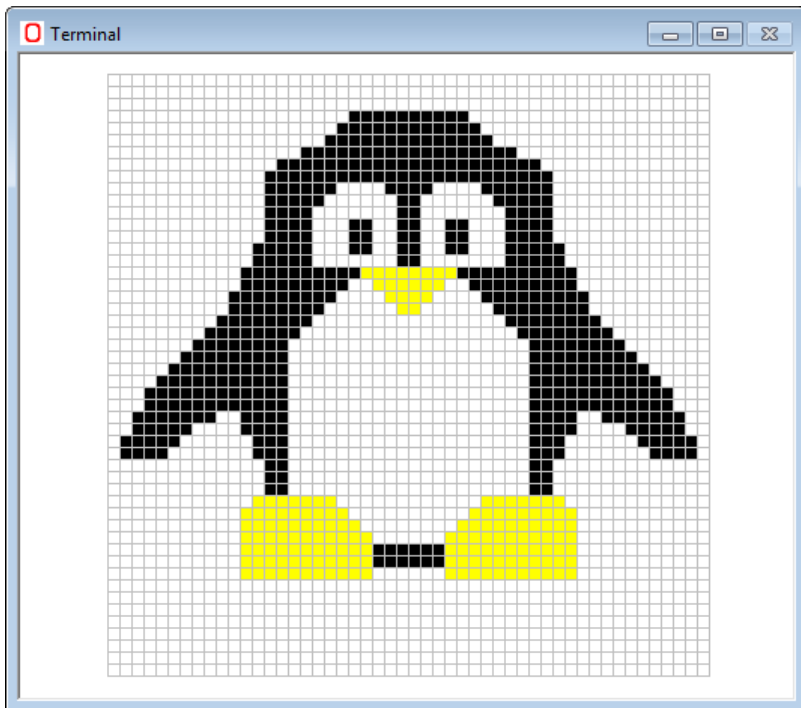
Arquitectura de computadoras

Entrada / Salida en Winmips

- Winmips posee una terminal que permite hacer algunas operaciones de entrada / salida por teclado y pantalla.



Entrada / Salida en modo texto: leer y escribir números y cadenas



Salida en modo gráfico: pintar pixeles en una pantalla de 50 x 50

Entrada / Salida en Winmips

- En el MSX88 tenemos espacios de memoria y entrada/salida bien diferenciados y se acceden con instrucciones diferentes:
 - MOV → Memoria
 - IN/OUT → Entrada/Salida
- En el Winmips accedemos al espacio de entrada/salida a través del espacio de memoria, por lo que usamos la misma instrucción en ambos casos:
 - LD/SD → Memoria
 - LD/SD → Entrada/Salida

Entrada / Salida en Winmips

- Existen dos direcciones de memoria en el winmips destinadas para entrada/salida.
- Al leer/escribir estas direcciones NO se lee ni escribe la memoria sino, que se lee/escribe el espacio de entrada/salida.
- La interacción con estos puertos nos permite realizar operaciones similares a las que hacíamos con las interrupciones por software del MSX88.

Puertos de Entrada / Salida en Winmips

- CONTROL

- Ubicado en la dirección 0x10000 (65536)
- Ocupa 8 bytes (64 bits)
- Se utiliza para almacenar un valor asociado a una operación:
 - Leer o escribir números enteros, flotantes y cadenas de texto
 - Escribir un pixel en pantalla gráfica
 - Borrar pantalla de texto o gráfica
- Solo tiene sentido escribirlo

- DATA

- Ubicado en la dirección 0x10008 (65544)
- Ocupa 8 bytes (64 bits)
- Se utiliza para leer/escribir un valor que depende del valor que se escribe en el puerto de control

Operaciones de Entrada

- Una operación de entrada:
 - Produce una entrada desde el teclado
 - No requiere parámetros (al menos para las operaciones que actualmente implementa el Winmips), no se escribe DATA
 - Se “ejecuta” cuando se escribe el valor de operación en el puerto de CONTROL. Bloquea la ejecución hasta que finalice la operación de entrada
 - Deja el valor de entrada en el registro DATA

Operaciones de Entrada

- Leer un número entero o flotante desde el teclado:
 - CONTROL ← 8
 - DATA → número entero o flotante
- Leer una tecla (código ascii) sin eco desde el teclado :
 - CONTROL ← 9
 - DATA → código ascii de la tecla presionada

Operaciones de Salida

- Una operación de salida:
 - Produce algún tipo de salida o efecto en pantalla de texto o pantalla gráfica
 - Puede requerir un parámetro. Cuando éste se necesita se escribe en DATA (siempre antes que el de control)
 - Se “ejecuta” cuando se escribe el valor de operación en el puerto de CONTROL. Inmediatamente aparece el resultado en pantalla

Operaciones de Salida

- Escribir un número entero sin signo:
 - DATA ← número natural
 - CONTROL ← 1
- Escribir un número entero con signo:
 - DATA ← número entero
 - CONTROL ← 2
- Escribir un número flotante:
 - DATA ← número flotante
 - CONTROL ← 3
- Escribir una cadena de texto (que termina en cero):
 - DATA ← dirección de comienzo de la cadena
 - CONTROL ← 4

Operaciones de Salida

- Escribir un pixel en pantalla gráfica:
 - DATA ← color del pixel (4 bytes)
 - DATA+4 ← coordenada Y del punto (1 byte)
 - DATA+5 ← coordenada X del punto (1 byte)
 - CONTROL ← 5
- Limpiar pantalla alfanumérica:
 - No requiere DATA
 - CONTROL ← 6
- Limpiar pantalla gráfica:
 - No requiere DATA
 - CONTROL ← 7

Ejercicio 1

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

Direcciones de mapeo, observar que CONTROL es una variable con la dirección y no el puerto de control

1	lwu	\$s0 , DATA(\$zero)	\$s0 = dirección de DATA
2	daddi	\$t0, \$zero, texto	\$t0= dirección de comienzo de texto
3	sd	\$t0, 0(\$s0)	DATA recibe el puntero de mensaje
4	lwu	\$s1 , CONTROL(\$zero)	\$s1 = dirección de CONTROL
5	daddi	\$t0, \$zero, 6	\$t0= 6, función 6 = limpiar pantalla
6	sd	\$t0, 0(\$s1)	CONTROL recibe 6 y limpia pantalla terminal
7	daddi	\$t0, \$zero, 4	\$t0= 4, función 4 = escribir cadena
8	sd	\$t0, 0(\$s1)	CONTROL recibe 4 y produce salida del mensaje
9	halt	<i>Nota: las instrucciones con valores inmediatos permiten 16bits, no es posible utilizar valores mayores a 65535, por eso CONTROL y DATA se cargan desde memoria</i>	

Ejercicio 6

.data

coorX: .byte 24

coordenada X de un punto

coorY: .byte 24

coordenada Y de un punto

color: .byte 255, 0, 255, 0 R,G,B,? → color magenta

CONTROL: .word32 0x10000

DATA: .word32 0x10008

DATA	255	0	255	?	24	24	?	?
	R	G	B		Y	X		

.text

1 lwu \$s6, CONTROL(\$zero)

\$s6= dirección de CONTROL

2 lwu \$s7, DATA(\$zero)

\$s7= dirección de DATA

3 daddi \$t0, \$zero, 7

\$t0= función 7, limpiar pantalla gráfica

4 sd \$t0, 0(\$s6)

Escribe registro de control, limpia pantalla

5 lbu \$s0, coorX(\$zero)

\$s0= coordenada X

6 sb \$s0, 5(\$s7)

Byte 5 DATA = coordenada X

7 lbu \$s1, coorY(\$zero)

\$s1= coordenada Y

8 sb \$s1, 4(\$s7)

Byte 4 de DATA = coordenada Y

9 lwu \$s2, color(\$zero)

\$s2= color para pixel (4 bytes)

10 sw \$s2, 0(\$s7)

Byte 0 a 3 = color para pixel

11 daddi \$t0, \$zero, 5

\$t0 = función 5, dibujar pixel

12 sd \$t0, 0(\$s6)

CONTROL recibe 5 y dibuje pixel

13 halt