Informe Practica de Laboratorio 3

Cesar Rojas, #31406902

Julio 2025

¿Cómo se organiza la memoria cuando un sistema utiliza memory-mapped I/O? ¿En qué región de memoria se suelen mapear los dispositivos? ¿Qué implicaciones tiene para las instrucciones lw y sw?

Al usar un sistema de E/S mapeado en memoria, se designa un espacio en dicha memoria para el uso del sistema. Esta es la region donde se 'mapean' los dispositivos E/S, y en esta se encuentran las direcciones y registros utilizados para interactuar con los dispositivos E/S.

La memoria principal y el sistema de E/S no se separan, comparten espacio en memoria para asi simplificar el trabajo del CPU. Las direcciones correspondientes a la E/S simplemente se denominan como tal.

Las instrucciones *lw* y *sw* se usan de igual manera para acceder a las porciones de memoria reservadas para la E/S, pero su función es un poco distinta. Al escribir o leer de direcciones correspondientes a la E/S, el sistema reconoce estas operaciones y actua de forma correspondiente, permitiendo al CPU interactuar con ellos.

¿Cuál es la principal diferencia entre memory-mapped I/O y la entrada/salida por puertos? ¿Qué ventajas y desventajas tiene cada enfoque? ¿Por qué MIPS32 utiliza principalmente memory-mapped I/O?

La E/S por puertos posee su propio espacio de memoria separado de la memoria principal, mientras que la E/S mapeada a memoria utiliza el espacio de memoria principal.

La principal ventaja de la E/S por puertos es dicho espacio de memoria separado. Esto permite que toda la memoria principal se utilice por otros sistemas, como la RAM o la ROM. La compartimentalización también es útil para evitar errores de memoria en donde un programa acceda a secciones de memoria reservadas para E/S o viceversa.

Sin embargo, la E/S por puertos también requiere de recursos adicionales (una nueva sección de memoria entera distinta), puede introducir un cierto nivel de latencia, requieren de operaciones distintas (no se podría usar *lw* o *sw*, por ejemplo) y aumentan la complejidad de las tareas realizadas por el CPU.

La E/S mapeada a memoria comparte espacio en memoria con otros sistemas, y existe el riesgo de acceso erroneo a memoria reservada. Sin embargo, las operaciones realizadas con este sistema tienden a ser mucho mas veloces ya que el CPU no debe acceder a una memoria distinta.

En un sistema con memory-mapped I/O: ¿Qué problemas pueden surgir si dos dispositivos usan direcciones solapadas? ¿Cómo se evita este conflicto?

Si dos dispositivos utilizan dos direcciones solapadas, pueden ocurrir varios errores, como lecturas de registros de estados erroneos o sobre-escritura de datos importantes antes de que estos sean leídos.

Evitar que esto ocurra se puede lograr de varias maneras, tanto a nivel de hardware como software. El *chipset* y la tarjeta madre se encargan de alocar espacios de memoria distintos a los dispositivos de E/S, y el BIOS (Basic Input/Output System) tambien juega un papel en determinar que espacio de memoria corresponde a que E/S.

¿Por qué se considera que el memory-mapped I/O simplifica el diseño del conjunto de instrucciones de un procesador? ¿Qué tipo de instrucciones adicionales serían necesarias si se usara E/S por puertos?

La E/S por memoria es mas simple y veloz, ya que no requiere de hardware adicional y simplemente hace uso de los componentes ya presentes. Las instrucciones utilizadas para interactuar con el sistema de E/S tambien ya se encuentran presentes (lw y sw).

Para la E/S por puertos se requiere de instrucciones adicionales para manejar la nueva memoria dedicada a la E/S. Estas instrucciones deben manejar el acceso a los puertos específicamente, en lugar de acceder a cualquier parte de la memoria. Un ejemplo son las instrucciones *IN* y *OUT*, para la lectura y escritura respectivamente.

¿Qué ocurre a nivel del bus de datos y direcciones cuando el procesador accede a una dirección de memoria que corresponde a un dispositivo? ¿Cómo sabe el hardware que debe acceder a un periférico en lugar de la RAM?

Las instrucciones simplemente escriben o leen de la dirección de memoria a la que se dirige. El hardware del sistema se encarga de decodificar la dirección y dirigir el bus a el dispositivo o componente correcto, sea RAM o E/S.

¿Es posible que un programa normal (sin privilegios) acceda a un dispositivo mapeado en memoria? ¿Qué mecanismos de protección existen para evitar accesos no autorizados?

No es posible para un programa sin privilegios acceder a las direcciones de un dispositivo mapeado en memoria. Esto es necesario para evitar problemas como sobrecargar los dispositivos, el uso erroneo de memoria reservada para E/S, y para asegurar que la seguridad de datos sensibles se mantenga.

El principal mecanismo para evitar que esto suceda son dichos privilegios. El sistema operativo solo les da el privilegio de escribir, leer y acceder a dispositivos mapeados en memoria a ciertos programas, para asegurarse de que solo los que sean necesarios posean acceso.

Los programas que poseen acceso directo a la E/S se conocen como *Drivers*.

¿Qué técnicas se pueden emplear para evitar esperas activas innecesarias al interactuar con dispositivos?

La mayoría de los sistemas contemporáneos hacen uso de *Interrupciones* en lugar de polling (chequear si el dispositivo esta listo una y otra vez).

Una interrupción es cuando un programa alerta al sistema que esta listo para actuar por su cuenta, en lugar de que el sistema lo revise constantemente. Esto permite que el sistema continué haciendo operaciones mientras espera a que el dispositivo este listo.

Análisis y Discusión de los Resultados

0.1 Sensor de Temperatura

El ejercicio de temperatura muestra por salida la temperatura y el código leídos. Si todo se ejecuta normalmente, en el orden debido, entonces el sensor se inicializara tan pronto como se escriba 0x2 en el registro SensorControl.

Una vez esto suceda se leerá la temperatura guardada en SensorDatos y se retornara esta junto a un código. Si el sensor estaba listo para leer la temperatura (SensorEstado = 1) el código retornado sera 0 si se ejecuto correctamente, o -1 si hay un error.

Si el código es -1, el sensor se debe re-inicializar (SensorEstado = -1). De lo contrario ya se leyó la temperatura (SensorEstado = 0).

0.2 Tensión Arterial

El programa espera hasta que se escriba el valor 1 en el registro TensionControl. Cuando esto sucede, inicia una medición de las tensiones Sistol y Diastol.

Esta medición ocurre en el procedimiento controlador_tension, donde se espera hasta que se escriba 1 en TensionEstado para mostrar por pantalla el resultado. En este ejercicio particular se le pide al usuario que ingrese los valores, pero en realidad otro dispositivo se encargaría de escribir los valores en TensionSistol y TensionDiastol, y actualizar el valor de TensionEstado a 1.