Informe de Práctica de Laboratorio III

José M. Díaz M. (25.682.785)
Sergio José Noguera (30.572.211)

29 de julio de 2025

1. Explique 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?

Se dice que un sistema utiliza Memory-Mappede I/O cuando los dispositivos de entrada y/o salida (E/S) del hardware, se comunican con la CPU como si fueran ubicaciones de memoria. La principal característica de esta llamada memoria es el espacio de direcciones unificado donde tanto la memoria principal como los registros de los dispositivos de E/S comparten el mismo bus de direcciones. Esto quiere decir que a cada dispositivo E/S se le tiene asignado un rango específico de direcciones dentro del espacio de direcciones general del sistema.

Los dispositivos de E/S suelen mapearse en una región de memoria reservada y dedicada para ellos, que no se utiliza para la RAM principal. En arquitecturas de 32 bits como MIPS, esta región se encuentra comúnmente en la parte superior del espacio de direcciones, es decir, en direcciones de memoria altas.

Tanto lw como sw son instrucciones para cargar y almacenar palabras respectivamente, son lo más común para acceder a la memoria. Su implicación en la Memory-Mapped I/O es que se utilizan para interactuar con los dispositivos de E/S, y esto lo hacen de las siguientes formas:

- lw (load word): Cuando la CPU ejecuta una instrucción lw con una dirección que corresponde a un registro de un dispositivo de E/S mapeado en memoria, el procesador lee el valor de ese registro del dispositivo. Se puede decir que es como leer una ubicación de memoria normal, pero en realidad, se está obteniendo información de estado o datos de entrada del periférico.

- sw (store word): De manera similar, cuando la CPU ejecuta una instrucción sw con una dirección que corresponde a un registro de un dispositivo de E/S, el procesador escribe el valor en ese registro del dispositivo. Esto permite al programa enviar comandos o datos de salida al periférico.

2. ¿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 principal diferencia entre Memory-Mapped I/O y la entrada/salida por puertos:

- Memory-Mapped I/O (MMIO): Tanto los dispositivos E/S, como sus registros de control y datos se mapean directamente en el espacio de direcciones que la memoria RAM. La CPU interactúa con los dispositivos utilizando las instrucciones lw y sw, para acceder fácilmente a la memoria, cabe aclarar que para CPU no hay distinción entre leer un byte de la RAM y leer un registro de estado.

- La entrada/salida por puertos: Los dispositivos de E/S no se mapean directamente en el espacio, e n su lugar tienen un espacio de direcciones separado y dedicado para ellos, distinto del espacio de direcciones de la memoria RAM. La CPU para acceder a estos dispositivos utiliza instrucciones de E/S especiales y dedicadas que dirigen la operación a un espacio de direcciones de E/S separado del espacio de direcciones de la memoria principal.

Ventajas del Memory-Mapped I/O (MMIO):

1. Simplifica el diseño del procesador y el conjunto de instrucciones del CPU haciendo que no se requiera de instrucciones especiales de E/S. Las mismas instrucciones lw y sw se usan para todo.

2. Mayor flexibilidad de acceso, Al ser tratados como memoria, los dispositivos de E/S pueden ser accedidos con todas las modalidades de direccionamiento disponibles para la memoria (directo, indirecto, indexado, etc.). Esto facilita la programación de controladores de dispositivos.

3. Ofrece mayor eficiencia potencial, ya que el acceso a E/S puede ser más rápido por utilizar los mismos mecanismos de bus y caché que la memoria.

4. Tiene Integración con La Unidad de Gestión de Memoria, esta puede aplicar protección de memoria y mapeo de direcciones virtuales también a los dispositivos de E/S, mejorando la seguridad y la gestión de recursos.

5. Menos pines de control: El bus de control puede ser más simple, ya que no necesita distinguir entre operaciones de memoria y E/S a nivel de señalización.

Desventajas del Memory-Mapped I/O (MMIO):

1. Tiene un consumo mayor en la memoria RAM, esto debido a que los dispositivos de E/S ocupan parte del espacio de direcciones de la memoria principal, lo que reduce la cantidad de RAM disponible para el programa.

2. Su Rendimiento variable, siendo que, si un dispositivo de E/S es muy lento, una operación de lectura/escritura a un registro mapeado en memoria puede afectar el rendimiento general

3. Los accesos a registros de E/S casi siempre deben ser no-cacheables, es decir no se puede almacenar en el cache, esto para evitar problemas de coherencia.

Ventajas de la entrada/salida por puertos:

1. Espacio de direcciones de E/S separado: Mantiene el espacio de direcciones de la memoria principal completamente disponible para la RAM, lo que es ventajoso en sistemas con espacio de direcciones limitado o cuando se desea maximizar la RAM utilizable.

2. Al requerir instrucciones especiales, el acceso a los puertos de E/S puede ser más fácilmente restringido por el sistema operativo, ofreciendo una mejor seguridad. Un programa de usuario en modo protegido no puede usar directamente IN/OUT sin permiso.

3. La distinción entre operaciones de memoria y E/S está explícita en el código, lo que puede mejorar la claridad a la hora de programar.

4. En sistemas con muchos dispositivos, la decodificación de direcciones para puertos puede ser más simple y económica si se usa un bus de E/S más pequeño.

DesVentajas de la entrada/salida por puertos:

1. Su conjunto de instrucciones es más complejo, ya que el CPU necesita instrucciones dedicadas para E/S (IN, OUT), lo que aumenta la complejidad del diseño del procesador.

2. Las instrucciones de E/S suelen ser más limitadas y menos flexibles, en sus modos de direccionamiento y en los registros que pueden utilizar.

3. Requiere señales de control adicionales en el bus para diferenciar entre accesos a memoria y accesos a puertos de E/S.

4. Menor velocidad potencial, esto debido a que el bus de E/S puede ser más lento que el bus de memoria principal, y las instrucciones especiales pueden introducir latencia adicional.

Razon por la que MIPS utiliza principalmen- te memory-mapped I/O:

La arquitectura RISC que utiliza MIPS, busca tener un conjunto de instrucciones pequeño, uniforme y fácil de decodificar para el procesador en este caso las ya mencionadas instrucciones lw y sw que al tratar la E/S de la llamada memoria, la CPU no necesita lógica de control separada para E/S. También Aprovecha muy bien de La Unidad de Gestión de Memoria y el sistema de memoria al mapear la E/S en memoria, ya que gestiona el acceso a los dispositivos de E/S con las mismas capacidades de protección de memoria y traducción de direcciones que se aplican a la RAM. En resumen, principalmente se usa debido a que ofrece una simplificada y fácil forma de generar código para operaciones de E/S.

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

En un sistema con Memory-Mapped I/O (E/S Mapeada en Memoria), los problemas pueden surgir cuando dos dispositivos intentan ocupar el mismo rango de direcciones de memoria, volviendo al sistema inestable o inoperable. Los Problemas se pueden definir como:

1. Problemas Comportamiento impredecible y corrupción de datos: Estos poder surgir tanto en lectura como escritura, el lectura, Si la CPU intenta leer de una dirección que es compartida por dos dispositivos, no se sabe cuál de los dos dispositivos responderá o si ambos intentarán poner datos en el bus al mismo tiempo, resultando en una lectura errónea o datos corruptos yen escritura si la CPU intenta escribir en una dirección solapada, ambos dispositivos podrían intentar interpretar y responder a la escritura simultáneamente haciendo que el sistema falle.

2. 2. Problemas Conflictivos de bus: Cuando dos dispositivos intentan responder a la misma dirección al mismo tiempo, ambos podrían intentar colocar datos en el bus de datos o controlar las líneas de control, causar fallos en la comunicación.

3. Problemas de Inestabilidad del sistema y bloqueos: Debido a la corrupción de datos y los conflictos de bus, el sistema operativo o las aplicaciones que intenten interactuar con estos dispositivos se encontrarán con valores incorrectos, respuestas inesperadas o simplemente no podrán comunicarse, lo que llevará a bloqueos del sistema, errores de hardware o reinicios inesperados.

4. Problemas de Imposibilidad de operar ambos dispositivos: Es posible que ninguno de los dispositivos funcionen correctamente, o que solo uno funcionen de manera intermitente, ya que sus operaciones se interferiría mutuamente.

Para evitar conflictos se puede:

1. Asignación única: En la fase de diseño del sistema, a cada dispositivo se le asigna un rango de direcciones único y no solapado dentro del espacio de direcciones de E/S. El decodificador de direcciones se diseña para reconocer únicamente su rango asignado.

2. Estándares de asignación de direcciones: reservar las direcciones altas del espacio de 32 bits para E/S, y luego subdividir esa región cuidadosamente,

y crear un mapa de memoria detallado y coherente que especifique qué rangos de direcciones están reservados para la RAM, qué rangos para la ROM y qué rangos para cada dispositivo de E/S específico (controlador de disco, controlador de red, temporizadores, etc.). Este mapa debe ser coherente en todo el sistema.

3. Los Sistema operativos adecuados: Los sistemas operativos modernos como Linux o Windows tienen mecanismos para gestionar recursos de hardware. Utilizan información proporcionada por el BIOS/firmware o detectan el hardware para asignar dinámicamente o confirmar las direcciones de E/S. Intentarán evitar conflictos, pero un conflicto a nivel de hardware o firmware es difícil de solucionar por software una vez que se ha producido. En sistemas más complejos, existen mecanismos como Plug and Play (PnP) o PCI/PCIe que permiten a los dispositivos negociar sus asignaciones de direcciones con el sistema operativo al inicio, reduciendo la probabilidad de conflictos manuales.

4. ¿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?

El memory-mapped I/O simplifica significativamente el diseño del conjunto de instrucciones de un procesador por varias razones fundamentales:

1 - Unificación del conjunto de instrucciones:

- Utiliza las mismas instrucciones para memoria y dispositivos de E/S.
- Elimina la necesidad de instrucciones especiales como IN y OUT.
- Reduce la complejidad del diseño del procesador al minimizar el número de instrucciones únicas necesarias.

2 - Simplificación de la arquitectura:

- Requiere menos lógica interna en el CPU.
- Reduce el tamaño físico del procesador.
- Consume menos energía al tener menos componentes.
- Facilita la implementación y verificación del diseño.

3 - Flexibilidad operacional:

- Permite utilizar todos los modos de direccionamiento disponibles para memoria también con dispositivos de E/S.
- Facilita las operaciones aritméticas y lógicas directamente sobre datos de dispositivos de E/S.
- Reduce el número de instrucciones necesarias para realizar operaciones complejas.

4 - Ventajas prácticas:

- Mejora la velocidad de acceso a dispositivos de E/S al usar el mismo mecanismo que para memoria.
- Simplifica significativamente la programación al eliminar la distinción entre operaciones de memoria y de E/S.
- Reduce la complejidad del código fuente necesario para manejar dispositivos periféricos.

Instrucciones Adicionales Necesarias en E/S por Puertos:

En el sistema de E/S por puertos (I/O Mapped I/O), se requieren tipos específicos de instrucciones adicionales que no existen en el sistema de memory-mapped I/O:

Instrucciones Básicas de E/S:

- IN (Entrada): Para leer datos desde dispositivos de entrada hacia registros del procesador.
- OUT (Salida): Para enviar datos desde registros del procesador hacia dispositivos de salida.

Variantes Específicas:

- INB/OUTB: Para operaciones de un byte.
- INW/OUTW: Para operaciones de dos bytes (palabra).
- INL/OUTL: Para operaciones de cuatro bytes (longitud).

Esta simplificación es especialmente importante en sistemas modernos donde la eficiencia y la simplicidad del diseño son fundamentales para el rendimiento general del sistema.

5. ¿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?

Cuando el procesador intenta acceder a una dirección de memoria que corresponde a un dispositivo, se produce una secuencia coordinada de eventos en los buses de datos y direcciones. El proceso funciona de la siguiente forma:

Secuencia de Eventos:

1) El procesador genera una señal de dirección en el bus de direcciones.
2) El dispositivo detecta su dirección específica en el bus de direcciones.
3) El controlador del dispositivo responde mediante las líneas de control.
4) Se establece la comunicación bidireccional a través del bus de datos.

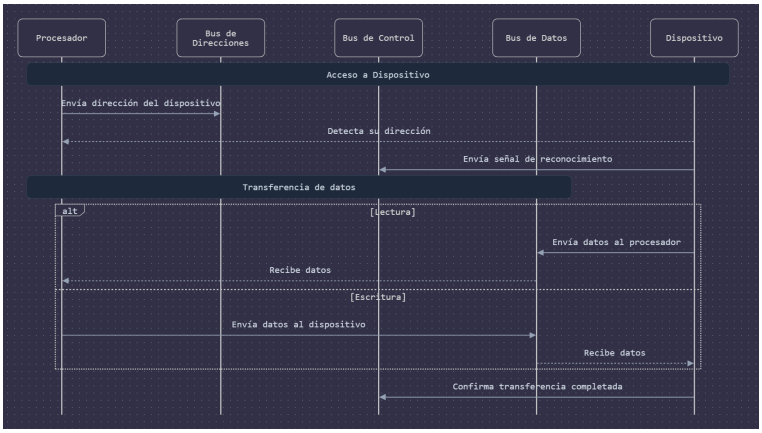

Figura 1: Procesador, Buses y Dispositivo

En la figura 1 se puede observar que:

- Las flechas sólidas (→) representan señales activas enviadas por los componentes.
- Las flechas punteadas (–¿) indican respuestas o confirmaciones.
- El bloque 'alt' muestra las dos posibilidades: lectura (el dispositivo envía datos) o escritura (el procesador envía datos).

Funcionamiento de los Buses:

1 - Bus de Direcciones:

- Transmite la dirección específica del dispositivo.
- Es independiente del bus de datos.
- Los dispositivos monitorean constantemente este bus para detectar su dirección.

2 - Bus de Control:

- Coordina el acceso entre todos los componentes.
- Evita colisiones en la transferencia de datos.
- Gestiona las señales de reconocimiento y confirmación.

3 - Bus de Datos:

- Permite la transferencia bidireccional de información.
- Se activa solo después de que el dispositivo reconoce su dirección.
- La dirección de transferencia depende del tipo de operación (lectura o escritura).

Periférico en lugar de la RAM:

El hardware determina si debe acceder a un periférico o a la RAM mediante un mecanismo llamado "mapeo de memoria", que divide el espacio de direcciones en dos regiones distintas. El proceso funciona de la siguiente forma:

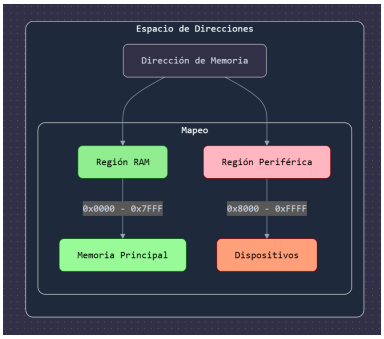

Figura 2: Mapeo de memoria

En la figura 2 se puede observar que:

- Las direcciones más bajas (0x0000 - 0x7FFF) corresponden a la memoria RAM.
- Las direcciones más altas (0x8000 - 0xFFFF) están reservadas para periféricos.
- Esta división es estática y conocida por el hardware.

Mecanismo de Decisión:

1) Cuando el procesador genera una dirección de memoria.
2) El hardware verifica automáticamente si cae dentro del rango de RAM o periféricos.
3) Basado en esta verificación, selecciona el camino apropiado.

Componentes Clave:

1 - Decodificador de Direcciones:

- Hardware especializado que examina los bits más significativos de la dirección.
- Actúa como un çonmutador"que dirige el acceso al destino correcto.
- Funciona en nanosegundos para mantener el rendimiento del sistema.

2 - Regiones de Memoria:

- RAM: Espacio para almacenar datos temporales y código en ejecución.
- Periféricos: Puertos de entrada/salida, registros de control y buffers de dispositivos.

6. ¿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, un programa normal sin privilegios no puede acceder directamente a un dispositivo mapeado en memoria. Este acceso está estrictamente controlado por el sistema operativo para garantizar la seguridad del sistema.

Razones técnicas:

1 - Control de privilegios:

- Los dispositivos mapeados en memoria operan en modo kernel (privilegios elevados).
- Los programas normales ejecutan en modo usuario (privilegios restringidos).
- El sistema operativo implementa una barrera de seguridad entre estos modos.

2 - Protecciones del sistema:

- La integridad de memoria protege el acceso a áreas sensibles.
- El aislamiento del núcleo previene accesos no autorizados.
- Las protecciones DMA (Acceso Directo a Memoria) evitan que dispositivos externos no autorizados accedan a la memoria del sistema.

Mecanismos de control:

1 - Control de acceso:

- Los dispositivos tienen privilegios efectivamente iguales al nivel VTL0 (nivel de confianza más bajo, donde se ejecuta el sistema operativo tradicional y las aplicaciones en modo usuario y modo kerne).
- Todo acceso DMA (Acceso Directo a Memoria) se marca con privilegios específicos.
- El sistema operativo valida cada petición de acceso.

2 - Capas de protección:

- Virtualización de hardware para aislar procesos críticos.
- Verificación de código antes de su ejecución.
- Monitoreo constante de accesos potencialmente peligrosos.

Acceso legítimo:

Para acceder a dispositivos mapeados en memoria, un programa debe:

1) Ejecutarse con privilegios elevados.
2) Utilizar APIs (Interfaces de Programación de Aplicaciones) del sistema operativo apropiadas.
3) Obtener permisos específicos del sistema.
4) Cumplir con las políticas de seguridad implementadas.

Mecanismos de Protección para Accesos no Autorizados:

Los sistemas modernos implementan múltiples capas de protección para prevenir accesos no autorizados a dispositivos mapeados en memoria. Estos mecanismos actúan en diferentes niveles, desde el hardware hasta el software, creando una defensa en profundidad contra amenazas potenciales.

Protecciones de Hardware:

1 - IOMMU (Unidad de Gestión de Memoria de Entrada/Salida):

- Actúa como guardián principal del acceso a memoria.
- Implementado como VT-d en Intel (Virtualización para E/S Dirigida) y AMD-Vi (AMD Virtualization: tecnología de virtualización desarrollada por AMD) en AMD (Advanced Micro Devices, Inc., es una empresa global de semiconductores).
- Controla y valida todos los accesos DMA (Acceso Directo a Memoria) al espacio de direcciones físico.

2 - Protección DMA (Acceso Directo a Memoria) del Kernel:

- Previne accesos directos no autorizados a la memoria.
- Especialmente importante para puertos PCI (Interconexión de Componentes Periféricos) y Thunderbolt.
- Activa automáticamente cuando el equipo está bloqueado o cerrada la sesión.

Protecciones del Sistema Operativo:

1 - Control de Memoria Virtual:

- El controlador de memoria virtual (MMU) gestiona el acceso a ubicaciones no autorizadas.
- Implementa separación estricta entre espacio de usuario y kernel.
- Valida cada petición de acceso a memoria.

2 - Sistema de Controladores:

- Los controladores deben ser probados y firmados digitalmente.
- Windows requiere firma digital de Microsoft para controladores.
- Prevención de carga de controladores no confiables.

Protecciones Adicionales:

1 - Integridad de Memoria:

- Crea un entorno aislado mediante virtualización de hardware.
- Verifica el código antes de su ejecución.
- Previne la manipulación por parte de programas malintencionados.

2 - Aislamiento del Núcleo:

- Protege los procesos principales de Windows del software malintencionado.
- Ejecuta procesos críticos en un entorno virtualizado.
- Previne la explotación de vulnerabilidades.

Medidas de Prevención:

1 - Seguridad Física:

- Protección contra dispositivos maliciosos conectados.
- Desactivación de puertos no utilizados en BIOS/UEFI.
- Control de acceso físico a los equipos.

2 - Protección de Datos:

- Cifrado de datos sensibles en memoria.
- No almacenar información confidencial sin cifrar en RAM.
- Implementación de medidas de seguridad adicionales para datos criptográficos.

Estos mecanismos trabajan en conjunto para crear una barrera robusta contra accesos no autorizados, protegiendo tanto el hardware como el software del sistema. La combinación de protecciones de hardware y software proporciona una defensa en profundidad que hace extremadamente difícil para los atacantes obtener acceso no autorizado a dispositivos mapeados en memoria.

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

La espera activa ocurre cuando un proceso consume recursos del CPU realizando verificaciones repetitivas sin realizar trabajo útil. Para evitar esperas activas innecesarias al interactuar con dispositivos, existen varias técnicas fundamentales que se pueden implementar. Las principales técnicas para optimizar estas interacciones:

1 - Optimización de Eventos y Temporizadores:

- Evitar temporizadores recurrentes (setInterval: método que ejecuta repetidamente una función o un fragmento de código a intervalos regulares, especificados en milisegundos) que puedan interferir con interacciones del usuario.
- Utilizar setTimeout (función que permite ejecutar un código o una función después de un período de tiempo especificado) de manera responsable, asegurándose de que las tareas no consuman demasiados recursos.
- Implementar limpieza adecuada de temporizadores cuando ya no sean necesarios.

2 - División de Tareas Costosas:

- Dividir operaciones pesadas en subtareas más pequeñas.
- Permitir que el sistema procese otras interacciones entre fragmentos.
- Utilizar scheduler.yield (método que permite a las aplicaciones web ceder el control del hilo principal del navegador, permitiendo que otras tareas de mayor prioridad se ejecuten) para permitir que el subproceso principal maneje otras tareas.

3 - Gestión Eficiente de Recursos:

- Precargar datos anticipadamente cuando sea posible.
- Implementar caché estratégicamente para reducir accesos a disco o red.
- Minimizar operaciones de E/S durante interacciones críticas.

4 - Optimización de Red y Conectividad:

- Agrupar transferencias de datos para minimizar el número de conexiones.
- Reutilizar conexiones existentes en lugar de crear nuevas.
- Implementar mecanismos de reintentos con backoff para fallos de red.

Al implementar estas técnicas de manera sistemática, es posible reducir significativamente las esperas activas innecesarias y proporcionar una experiencia

más fluida y responsiva para los usuarios. La clave está en encontrar un equilibrio entre el rendimiento óptimo y el consumo eficiente de recursos del dispositivo.

8. Análisis y Discusión de los Resultados

La mayor ventaja que tiene la Memory-Mapped I/O sobre la entrada/salida por puertos, es su capacidad de simplificar conjuntos de instrucciones (no especiales) del procesador para que puedan tener fácil acceso a la memoria, su diseño como arquitectura RISC, busca la uniformidad y la eficiencia en el diseño del hardware del procesador y la facilidad de programación, puede que dependa completamente de la RAM pero esto trae más beneficio que problemas, solo se tiene que ser cuidado al asegurarse que cada dispositivo tenga un rango de direcciones de memoria único y exclusivo para su operación, y conocer las técnicas para evitar esperas activas innecesarias al interactuar, como la optimización de eventos como la división de tareas costosas, la gestión eficiente de recursos y la optimización de red.

Una cosa a tomar muy en cuenta es que un programa sin privilegios no podrá acceder directamente a un dispositivo mapeado en memoria, debido a que el acceso está controlado por el sistema operativo que cuenta con sus mecanismos de protección como IOMMU y la gestión de memoria virtual para evitar accesos no autorizados, separando estrictamente el espacio de usuario y es sistema operativo que opera en modo kernel.