Informe de Práctica de Laboratorio IV

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

05 de agosto de 2025

1. Explique con un diagrama cómo funciona el ciclo de una interrupción de hardware (desde que ocurre el evento hasta que se atiende).

El ciclo de interrupción de hardware es un mecanismo fundamental que permite a los dispositivos periféricos comunicarse con el procesador cuando necesitan atención, y a su vez, le permite al procesador responder a estos eventos externos (como una señal de algún dispositivo de E/S) de manera asíncrona, pero eficiente, sin necesidad de realizar operaciones de consulta constante (polling o sondeo).

A continuación el proceso paso a paso:

1 - Ocurrencia de Evento Externo de Interrupción y Señal:

- Un dispositivo periférico (teclado, disco duro, temporizador, etc.) necesita atención del procesador.
- El dispositivo activa una línea específica (señal electrica) llamada IRQ (Interrupt Request - Solicitud/Señal de interrupción) que llegará al controlador de interrupciones (un circuito dentro del sistema. Ej: Los microcontroladores PIC).

2 - Recepción de Señal, Controlador de Interrupción y Priorización:

- La señal IRQ es recibida y procesada por el controlador de interrupciones.
- El controlador determina la prioridad y origen de la interrupción, y la compara con otras otras interrupciones pendientes.
- Si la interrupción tiene mayor prioridad, el controlador le informa al procesador/CPU enviándole una señal para que proceda a atenderla.

3 - Respuesta del Procesador, Guardado del Estado del Sistema y Desactivación:

- Antes de atender la interrupción, el procedador suspende o termina la ejecución del programa o tarea actual que se puede estar efectuando. Al finalizar la ejecución de la instrucción actual, el CPU chequea si hay interrupciones pendientes; esto lo hace al final de cada ciclo de intrucción.
- Se guarda en la pila el contexto o estado actual de la CPU, como el valor de los registros, el valor del contador de programa, la dirección de la siguiente instrucción a ejecutar (PC o Program Counter/Contador de Programa), y otros datos relevantes, para poder retomarlo después.
- Este proceso de guardar el contexto es crucial para poder volver a la tarea original sin perder información.
- Se desactivan temporalmente las interrupciones para evitar que el proceso de atención de la interrupción actual (la de mayor prioridad) sea afectado por otras interrupciones, especialmente las de menor prioridad.

4 - Acceso y Ejecución de la ISR:

- El procesador busca y ejecuta la Rutina de Servicio de Interrupción (ISR) correspondiente a la información del dispositivo que generó la interrupción.
- La ISR es un código o fragmento de código que que se ejecuta en respuesta a una señal de una interrupción específica. En este caso atiende específicamente al evento del dispositivo que generó la interrupción..
- La ISR realiza las acciones u operaciones necesarias para atender la solicitud del dispositivo, como leer datos, escribir datos, o realizar alguna otra operación específica. Por ejemplo, si la interrupción fue generada por un teclado, la ISR podría leer el carácter ingresado por el usuario.

5 - Restauración y Retorno:

- Una vez que la ISR ha terminado de atender la interrupción, la CPU restaura el estado guardado anteriormente en el paso 3.
- La restauración del estado anterior incluye los valores originales de los registros, el contador de programa, el PC, y otros datos.
- Esto le permite al CPU volver al flujo normal de ejecución y hacer que el programa o tarea que fue interrumpida continúe su ejecución desde el punto donde se detuvo.

En el siguiente diagrama se observará:

- Las flechas sólidas ($\rightarrow$) indican acciones o señales enviadas.
- Las flechas punteadas (- - ->) representan respuestas o confirmaciones.
- Los bloques azules destacan las fases críticas de guardado y restauración del estado.
- En la parte de arriba se muestran los cuatro componentes principales involucrados en el proceso.

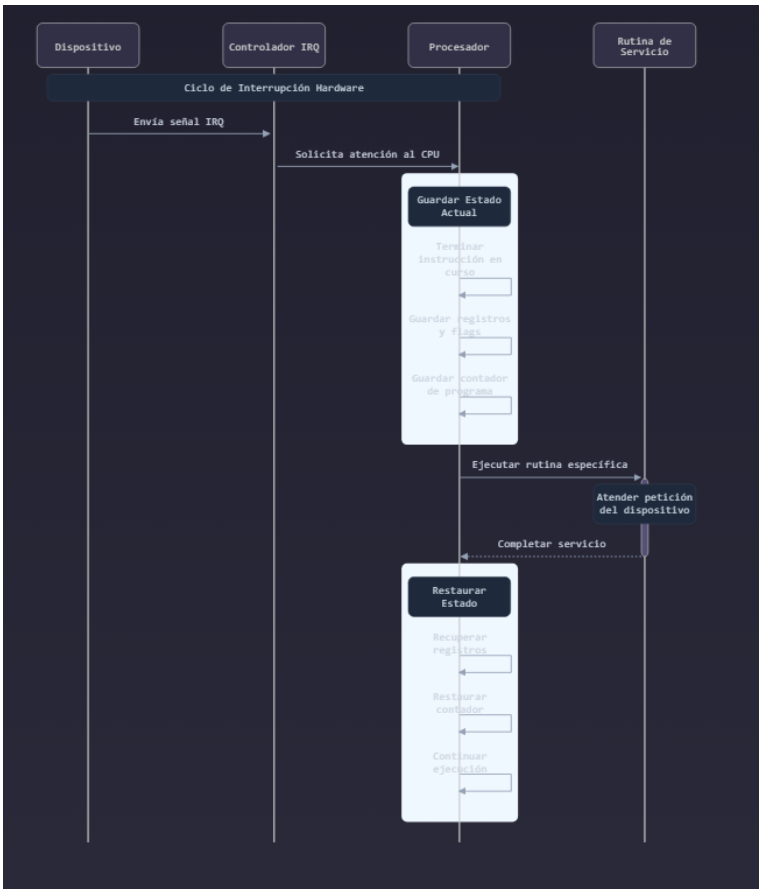

Figura 1: Dispositivo, Controlador IRQ y Procesador

2. ¿Qué diferencias hay entre gestionar E/S con sondeo y hacerlo con interrupciones?

La gestión de Entrada/Salida (E/S) en computadoras puede realizarse principalmente de dos formas: mediante sondeo (polling) o mediante interrupciones. A continuación se mostrará un cuadro con las diferencias entre ambos métodos:

CARACTERÍSTICA	SONDEO (POLLING)	INTERRUPCIONES
Método	El procesador pregunta periódicamente a los dispositivos si necesitan atención.	Los dispositivos notifican al procesador cuando necesitan atención.
Eficiencia	Muy ineficiente, consume recursos constantemente en preguntas innecesarias.	Altamente eficiente, el procesador solo actúa cuando es necesario.
Uso del CPU	Alto consumo de ciclos del procesador en preguntas repetitivas.	Libera al procesador para realizar otras tareas.
Complejidad	Implementación simple en hardware.	Requiere hardware adicional más complejo.
Latencia	Puede perder eventos si ocurren entre sondeos.	Responde inmediatamente a los eventos.
Aplicaciones típicas	Sistemas simples o con pocas operaciones de E/S.	Sistemas modernos multitarea y aplicaciones complejas.

En los diagramas (rojo claro y verde claro) se podrá observar lo siguiente:

- El diagrama rojo claro (lado izquierdo del cuadro) muestra cómo el procesador constantemente pregunta al dispositivo si necesita atención.
- El diagrama verde claro (lado derecho del cuadro) ilustra cómo el procesador continúa con sus tareas mientras espera que el dispositivo lo interrumpa.
- Las flechas indican el flujo de control y comunicación entre componentes.

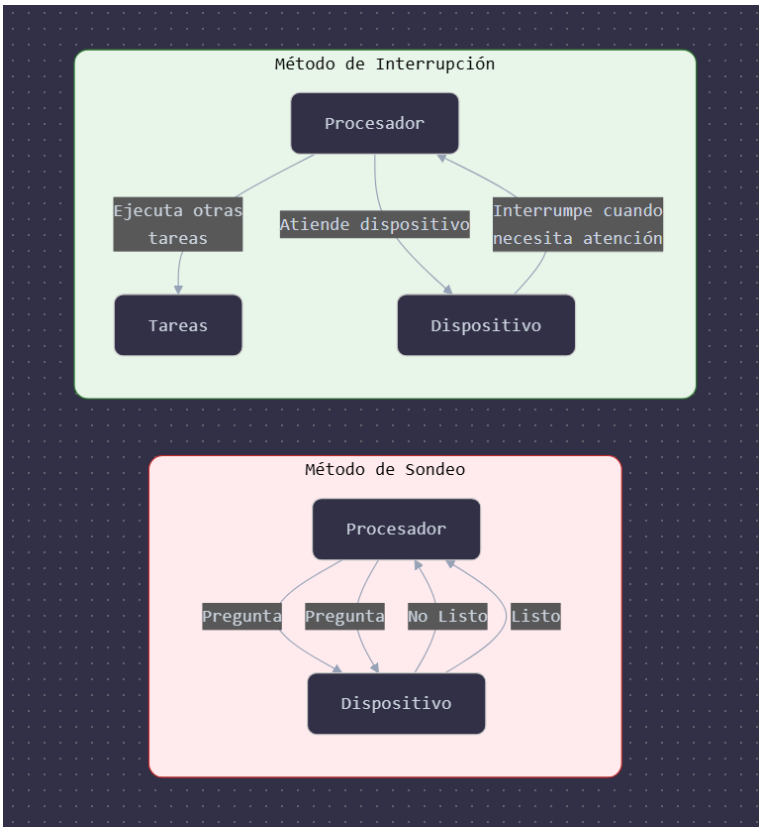

Figura 2: Sondeo e Interrupciones

3. ¿Qué ventajas tiene el uso de interrupciones en términos de uso del procesador?

Las interrupciones ofrecen varias ventajas significativas en términos de uso del procesador, que se pueden analizar detalladamente en el siguiente cuadro:

ASPECTO	VENTAJA	BENEFICIO PARA EL CPU
Eficiencia	El procesador solo actúa cuando es necesario.	Libera ciclos para otras tareas.
Gestión de tiempo	Permite ejecutar otras instrucciones mientras espera operaciones E/S.	Mayor rendimiento global del sistema.
Multitarea	Facilita la gestión simultánea de múltiples dispositivos.	Mejora la productividad del procesador.
Optimización	Evita el desperdicio de recursos en verificaciones innecesarias.	Reduce significativamente el consumo de ciclos.

Impacto en el Rendimiento:

Las interrupciones mejoran significativamente el rendimiento del sistema porque:

- Permiten una gestión eficiente de eventos asíncronos.
- Evitan que el procesador pierda tiempo esperando a los dispositivos.
- Mejoran la productividad global del sistema.
- Facilitan la implementación efectiva de sistemas multitarea.

4. ¿Qué registros especiales se utilizan en MIPS32 para gestionar interrupciones?

En MIPS32, se utilizan varios registros especiales que trabajan en conjunto para gestionar eficientemente las interrupciones. Estos registros son fundamentales para el control y manejo de las interrupciones en el procesador.

Registros Esenciales en MIPS32 para Interrupciones:

1 - Registro Status - ($12, CP0):

- Controla el modo de operación del procesador.
- Gestiona los niveles de prioridad.
- Permite habilitar/deshabilitar interrupciones.

Bits importantes:

- IE (Bit 0): Interrupt Enable (1 = habilitadas, 0 = deshabilitadas).
- EXL (Bit 1): Exception Level (1 = en excepción/interrupción, evita nuevas interrupciones).
- IM (Bits 8–15): Interrupt Mask (habilita/deshabilita líneas de interrupción específicas).
- UM (Bit 4): User Mode (0 = kernel, 1 = user).

2 - Registro Cause - ($13, CP0)

- Almacena información sobre la fuente de la interrupción.
- Identifica el tipo de excepción o interrupción.
- Ayuda en la priorización de interrupciones.

Bits importantes:

- IP (Bits 8–15): Interrupt Pending (muestra qué interrupciones están pendientes).
- ExcCode (Bits 2–6): Exception Code (código del tipo de excepción, ej: 0 = interrupción hardware).

3 - Registro EPC (Exception Program Counter) - ($14, CP0)

- Guarda la dirección de la instrucción que se estaba ejecutando cuando ocurrió la interrupción (para retornar después).
- Permite el retorno correcto después de atender la interrupción.
- Mantiene el contexto del programa interrumpido.

4 - Registro BadVAddr ($8, CP0)

Guarda la dirección de memoria que causó una excepción (ej: acceso inválido). No se usa en interrupciones simples.

5 - Registro Context ($4, CP0)

Ayuda en el manejo de excepciones de memoria (como TLB misses), pero no es crítico para interrupciones básicas.

Ejemplo de Configuración de Interrupciones:

```
# Habilitar interrupción del timer (IM[5]) y global (IE)
mfc0    $t0, $12                # Leer Status
ori     $t0, $t0, 0x8001        # IE=1, IM[5]=1
mtc0    $t0, $12                # Escribir Status
```

Figura 3: Para Habilitar Interrupciones del Temporizador

5. ¿Por qué es necesario guardar el contexto (registros) al entrar en una rutina de servicio?

El guardado del contexto (registros) es una operación fundamental cuando se entra en una rutina de servicio de interrupción. A continuación se mostrarán algunas razones principales:

Razones Principales:

1 - Preservación del Estado del Programa:

- El programa principal estaba en plena ejecución cuando ocurrió la interrupción.
- Los registros contenían valores importantes para esa ejecución.
- Es necesario mantener estos valores para poder continuar correctamente después.

2 - Independencia de Operaciones:

- La rutina de servicio necesita usar los registros para su propia ejecución.
- Si no se guardaran, los valores del programa original se perderían.
- Cada rutina de servicio puede necesitar diferentes registros.

Ejemplo Práctico:

Se considera el siguiente ejemplo simple:

```
# Programa principal
add $t0, $t1, $t2    # Sumando valores
mul $t3, $t0, $t4    # Multiplicando resultado

# Interrupción ocurre aquí
# Contexto debe guardarse para poder continuar la multiplicación
```

Figura 4: Ejemplo con una Suma y una Multiplicación

Si no se guardara el contexto:

- El valor de $t0 (resultado de la suma) se perdería.
- La multiplicación no podría completarse correctamente.
- El programa principal produciría resultados incorrectos.

6. Momentos en que pueden generarse excepciones en un sistema MIPS32

a) Enumera al menos 4 situaciones en las que se pueda generar una excepción (por ejemplo: desbordamiento aritmético, fallo de dirección, etc.).

b) Explica qué etapas del pipeline pueden provocar una excepción y por qué

Las excepciones en MIPS32 son eventos que alteran el flujo normal de ejecución del programa. Estos eventos pueden ocurrir en diferentes momentos y etapas del procesador.

Situaciones que Generan Excepciones:

1 - Desbordamiento Aritmético:

- Ocurre durante operaciones aritméticas que exceden el rango del registro.
- Ejemplo: Suma de dos números positivos que resulta en un valor mayor que $2^31 - 1$.
- Se detecta en la etapa de ejecución (EX) del pipeline.

2 - Fallo de dirección:

- Acceso a una dirección no alineada (ej: lw o sw en una dirección que no es múltiplo de 4).
- Ejemplo: lw $t0, 0x1001($zero) (0x1001 no es múltiplo de 4)..

3 - Acceso a Memoria Inválido:

- Se genera al intentar acceder a direcciones no mapeadas.
- Ocurre durante la etapa de memoria (MEM).
- Incluye violaciones de segmento y páginas no presentes.

4 - Instrucción No Valida:

- Se produce al decodificar una instrucción ilegal.
- Detectada en la etapa de decodificación (ID).
- Incluye códigos de operación inválidos.

5 - División por Cero:

- Ocurre durante operaciones de división.

- Se detecta en la etapa de ejecución (EX).
- Es una excepción aritmética crítica.

Etapas del Pipeline que Pueden Producir Excepciones:

1 - Etapa IF (Instruction Fetch - Búsqueda de Instrucción):

- Puede detectar instrucciones ilegales en la memoria.
- Se produce cuando el código de operación no es válido.
- La detección ocurre antes de la decodificación completa.

2 - Etapa ID (Decode - Decodificación):

- Detecta instrucciones no válidas después de la decodificación.
- Verifica permisos de acceso a registros.
- Identifica operaciones no permitidas en el modo actual.

3 - Etapa EX (Execute - Ejecutar):

- Es la etapa más propensa a excepciones.
- Detecta desbordamientos aritméticos.
- Identifica divisiones por cero.
- Verifica operaciones no permitidas.

4 - Etapa MEM (Memory Access - Acceso a Memoria)):

- Detecta fallos de alineación.
- Identifica violaciones de segmento.
- Verifica accesos a memoria inválidos.

5 - WB (Write Back - Escritura Diferida):

- En MIPS32, esta etapa no genera excepciones, ya que los errores se detectan en etapas anteriores (MEM o EX).

En el diagrama se puede observar:

- Las flechas muestran el flujo normal del pipeline (arriba) y los puntos de detección de excepciones (centro).
- Cada etapa puede generar diferentes tipos de excepciones.
- El proceso de manejo de excepciones (abajo) es común para todas las excepciones detectadas.

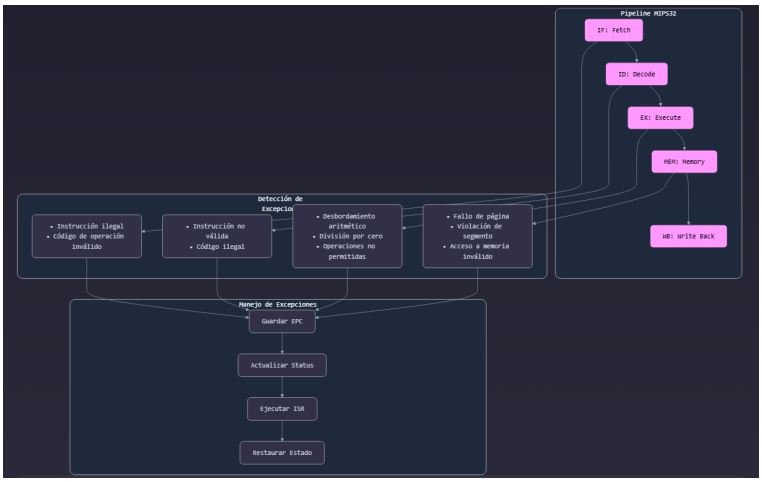

Figura 5: Etapas del Pipeline que Pueden Producir Excepciones

7. Estrategias de tratamiento de excepciones e interrupciones

a) Explica las diferencias entre interrupciones y excepciones (¿son síncronas o asíncronas?).

b) Describe brevemente dos estrategias para tratar excepciones en un sistema MIPS32 ¿Cómo se redirige la ejecución hacia la rutina de servicio? ¿Cuál es la función del registro EPC (Exception Program Counter)?

Las interrupciones y excepciones son mecanismos muy importantes para el rendimiento de los sistemas informáticos. Su correcta gestión, en gran medida a cargo del sistema operativo, es lo que permite que los programas de software convivan con el hardware y reaccionen de forma controlada a eventos inesperados, por eso primero se va a profundizar en las diferencias entre interrupciones y excepciones:

Característica	Interrupción	Excepción
Causa	Es un evento externo a la CPU.	Es un evento interno a la CPU, causado por la ejecución de una instrucción.
Sincronicidad	**Asíncrona.** Puede ocurrir en cualquier momento, independientemente del programa que se esté ejecutando en la CPU.	**Síncrona.** Siempre ocurre como resultado directo de una instrucción en particular que se está ejecutando en ese momento.
Ejemplos	• Un temporizador de hardware que expira. • Un dispositivo de E/S que termina una operación (teclado, disco duro, red).	• División por cero. • Acceso a una dirección de memoria no válida (violación de segmento). • Una instrucción ilegal.
Propósito	Notificar a la CPU sobre eventos externos para que pueda realizar una acción.	Indicar un error o una condición excepcional que el programa no puede manejar por sí mismo.
Tipos	Por hardware (E/S, temporizador) y por software (llamadas al sistema).	Fallos, trampas y abortos.
Recuperación	El sistema operativo atiende la interrupción y, por lo general, el programa original puede reanudarse sin problemas.	Su recuperación puede implicar corregir el error y continuar, pero a menudo resulta en la terminación del proceso.

Estrategias de Tratamiento de Excepciones en MIPS32

1. Uso del Vector de Inicio (BEV=1):Esta es la estrategia por defecto, especialmente al iniciar el sistema o en sistemas muy simples. Cuando el bit BEV en el registro Status es 1, todas las excepciones (tanto interrupciones de hardware como excepciones de software) son redirigidas a una única dirección de memoria fija: 0xBFC00200

2. Uso del Vector General (BEV=0):Esta es la estrategia utilizada por los sistemas operativos modernos. Cuando el bit BEV es 0, las excepciones son redirigidas a una dirección de memoria diferente: 0x80000080. Además, algunas excepciones específicas (como interrupciones o fallos de dirección) pueden ser

redirigidas a un vector secundario, lo que agiliza el proceso.

Explicación de como se redirije la ejecución hacia la rutina de servicio

1. El procesador detecta una condición de excepción.

2. El hardware automáticamente guarda el valor del contador de programa (PC) actual en el registro EPC (Exception Program Counter).

3. El hardware escribe un código de excepción (ExcCode) en el registro Cause para indicar la razón de la excepción.

4. El hardware modifica el registro Status, típicamente estableciendo el bit ExL (Exception Level) a 1. Esto deshabilita las interrupciones y eleva los privilegios del procesador al modo kernel.

5. El hardware carga en el contador de programa (PC) la dirección de memoria de la rutina de servicio de excepciones. Esta dirección es 0xBFC00200 o 0x80000080, dependiendo del estado del bit BEV

6. El procesador comienza a ejecutar el código de la rutina de servicio de excepciones.

¿Cuál es la función del registro EPC (Exception Program Counter)?

Su función principal es almacenar la dirección de memoria de la instrucción donde ocurrió la excepción, o la dirección de la siguiente instrucción, para que el sistema pueda retomar la ejecución del programa una vez que la excepción ha sido manejada.

Para fallos de una intrucion, el EPC guardara la dirección de la instrucción que intentó acceder a una página de memoria no presente. El manejador de excepciones carga la página en memoria y luego regresa al programa usando la instrucción eret. El procesador utiliza el valor de EPC para volver a ejecutar la instrucción que falló, la cual ahora tendrá éxito.

Para las execciones intencionadas, (trampas) el EPC guardara la dirección de la instrucción syscall. El manejador de excepciones realiza el servicio solicitado. Para regresar, el manejador de excepciones debe incrementar el valor de EPC en 4 bytes (el tamaño de una instrucción) antes de ejecutar eret. Esto es crucial para que la ejecución se reanude después de la instrucción syscall, y no vuelva a ejecutarla en un bucle infinito.

8. Habilitación de interrupciones en dispositivos y procesador

a) Explica cómo se habilitan las interrupciones en:

El teclado.

La pantalla.

El procesador (detalla qué bits del registro Status deben modificarse).

b) ¿Qué pasaría si habilitamos interrupciones en los dispositivos, pero no en el procesador?

Algunas de las interrupciones de las que se va a hablar a continuación (Teclado y pantalla) son nivel de hardware, es decir que además de depender del software (El procesador), dependen de un periférico con controlador inicializado y una conexión a la línea IRQ, así podrá generar interrupciones.

8.1. Interrupción por Teclado

- **A nivel de Hardware:**
  - El controlador del teclado tiene un búfer para las teclas presionadas.
  - Cuando una tecla es presionada el búfer tiene datos disponibles, el controlador genera una señal eléctrica en una de las líneas IRQ del sistema.
- **A nivel de Software:**
  - Se debe habilitar la interrupción IM0 en el registro Status del procesador MIPS32.
  - Cuando el procesador detecta la interrupción, salta a la rutina de servicio de interrupción (ISR) del teclado.
  - La ISR lee el carácter del búfer del teclado, lo procesa y luego borra la señal de interrupción en el controlador del teclado.
  - Finalmente, la ISR retorna, y el procesador retoma el programa que fue interrumpido.

8.2. Interrupción por Pantalla (Display)

- **A nivel de Hardware:**
  - Se puede usar un controlador de video (Sincronización vertical) genera una interrupción cuando el rayo de electrones (en monitores antiguos) o el barrido del framebuffer (en pantallas modernas) llega al final de la pantalla y se prepara para el siguiente cuadro.

- Una interrupción puede señalizar que una operación de dibujo compleja (ej. mover un bloque de memoria a la pantalla) ha finalizado.

- **A nivel de Software:**
  - El sistema operativo habilita la interrupción correspondiente a la línea IRQ del controlador de video.
  - La ISR del controlador de video se encarga de manejar el evento, como actualizar el buffer de la pantalla o notificar a un proceso de usuario que el dibujo ha terminado.

8.3. Interrupción por Procesador (Propiamente dicha):

- **A nivel de Hardware:**
  - El procesador establece el bit ExL en 1 cuando entra en un manejador de excepción, de esta forma todas las interrupciones externas (hardware) son deshabilitadas, incluso si el bit IEc está habilitado. Esto evita que una interrupción externa interrumpa la ejecución del manejador de excepción, lo que podría llevar a un estado inconsistente.
  - ExL también puede cambiar el modo de operación de la CPU a modo kernel (modo privilegiado), lo que permite al manejador de excepción acceder a recursos protegidos.
  - Es importante distinguir entre ExL y ERL. ExL se usa para excepciones "normales"de las que se espera regresar (como una syscall o un page fault). ERL se usa para errores graves e irrecuperables, como un reset, una NMI (interrupción no-mascarable) o un error de caché. Cuando ERL es 1, el procesador también está en modo kernel con las interrupciones deshabilitadas.

- **A nivel de Software:**
  - Aunque el hardware ya ha deshabilitado las interrupciones al establecer ExL, el manejador de excepción podría querer habilitarlas de nuevo en un punto seguro de su ejecución para poder atender interrupciones anidadas si la arquitectura lo permite, o para evitar un bloqueo del sistema. Esto se haría modificando el bit IEc (Interrupt Enable) y la máscara de interrupciones (IM). Esta instrucción genera una excepción interna en el procesador.
  - Cuando el manejador de excepción termina su trabajo, debe restaurar el estado del procesador para que el programa original pueda reanudar su ejecución. Para ello, se usa la instrucción eret (return from exception). La instrucción eret restaura automáticamente el valor de Status y el contador de programa (EPC) guardado para la excepción, lo que, entre otras cosas, limpia el bit ExL para reanudar el estado normal de interrupciones. Esta

8.4. ¿que pasaria si habilitamos interrupciones en los dispositivos, pero no en el procesador?

Si las interrupciones se habilitan en los dispositivos, pero no en el procesador, los dispositivos periféricos seguirán enviando señales eléctricas de interrupción (peticiones IRQ) al procesador cuando un evento ocurra, como al presionar una tecla o al terminar una operación de E/S.

Sin embargo, el procesador no las reconocerá ni las atenderá. Esto se debe a que, para que el procesador pueda responder a una interrupción, el bit de habilitación de interrupciones (IEc) en su registro Status debe estar activo. Si este bit está deshabilitado, el procesador ignorará las señales de interrupción que reciba de los dispositivos externos, sin importar si los dispositivos las están generando correctamente o no.

En resumen, la habilitación de interrupciones en los dispositivos es solo una parte del proceso; la habilitación a nivel de procesador es la que permite que el sistema responda y gestione las peticiones de interrupción de manera efectiva.

9. Procesamiento de interrupciones

a) Describe paso a paso qué ocurre cuando se produce una interrupción de reloj:

Desde que el evento ocurre hasta que la rutina de servicio termina.

¿Qué registros se guardan y restauran?

¿Qué hace el hardware y qué hace el software (sistema operativo o rutina)?

b) ¿Por qué es importante guardar el contexto (registros generales, EPC, Status) al entrar en la rutina?

Paso a Paso: Ciclo de una Interrupción de Reloj

9.1. Generación de la Señal de Interrupción (Hardware)

1. **Evento:** Un contador de hardware, integrado en un chip temporizador, alcanza un valor predefinido.
2. **Acción del Hardware:** El temporizador genera una señal eléctrica, una **petición de interrupción (IRQ)**, que envía a una de las líneas de interrupción del procesador. En la arquitectura MIPS32, esta suele ser la interrupción de hardware número 7 (`IP7`), reservada para el temporizador.

9.2. Detección y Preparación por la CPU (Hardware)

1. **Detección:** Al final de la ejecución de una instrucción, el procesador verifica si hay interrupciones pendientes y si están habilitadas (`IEc` en el registro `Status` es 1) y enmascaradas correctamente (`IM7` en el registro `Status` es 1).
2. **Guardado del estado del programa:** El procesador realiza las siguientes acciones automáticas:
   - Guarda el valor actual del **contador de programa (PC)** en el registro `EPC` (Exception Program Counter).
   - Modifica el registro `Status` del Coprocesador 0, estableciendo el bit `ExL` **a 1**, lo que deshabilita inmediatamente las interrupciones anidadas.

9.3. Redirección al Manejador (Hardware)

1. **Búsqueda del manejador:** El hardware consulta la tabla de vectores de excepciones para encontrar la dirección de memoria de la rutina de servicio.

2. **Salto:** El procesador carga la dirección del manejador de interrupciones en el contador de programa (PC), transfiriendo el control a la rutina de servicio del reloj.

9.4. Ejecución de la Rutina de Servicio (Software del Sistema Operativo)

1. **Identificación de la interrupción:** El manejador de excepciones del sistema operativo lee el registro `Cause` y su campo `ExcCode` para confirmar que se trata de la interrupción del temporizador (`IP7`).

2. **Guardado de contexto del proceso:** El kernel guarda todos los registros de propósito general, registros de punto flotante, etc., del proceso que fue interrumpido en su **Bloque de Control de Proceso (PCB)**.

3. **Atención al temporizador:** El manejador realiza sus tareas:

   - Actualiza la "hora"del sistema.
   - Decrementa el contador de tiempo restante del proceso actual (su *quantum*).
   - Vuelve a programar el temporizador para la próxima interrupción.

4. **Planificación (Scheduling):** El kernel verifica si el proceso actual ha agotado su *quantum* de tiempo. Si lo ha hecho, el planificador elige otro proceso de la cola de "listos"para que se ejecute.

5. **Restauración de contexto del nuevo proceso:** El kernel carga los registros de contexto del nuevo proceso (o del mismo si no hubo cambio), incluyendo su contador de programa (`PC`).

9.5. Fin de la Rutina de Servicio y Retorno

1. **Restauración del EPC:** El manejador de excepciones del kernel carga en el registro `EPC` la dirección donde debe continuar la ejecución del nuevo proceso (o del anterior).

2. **Instrucción `eret`:** El manejador ejecuta la instrucción `eret` (Return from Exception), la cual realiza las siguientes acciones:

   - Carga el valor de `EPC` en el contador de programa (PC).
   - Restaura el valor original del registro `Status`, lo que **limpia el bit `ExL`** y rehabilita las interrupciones.

9.6. Reanudación de la Ejecución del Programa

El procesador comienza a ejecutar el nuevo proceso (o el mismo proceso que fue interrumpido) como si nada hubiera pasado. Las interrupciones vuelven a estar habilitadas, y el ciclo continúa.

Importancia de guardar el contexto(registros generales,EPC,STAUS) al entrar a una rutina

1. Registros de propósito general: Guardar estos registros es vital para preservar los datos y variables con los que el programa estaba trabajando.

2. Registro EPC: Guardar estos registros es vital para preservar los datos y variables con los que el programa estaba trabajando.

3. Registro Status: Crítico para el çómo"volver a ejecutar. Preserva el estado de los bits de control del procesador (habilidad de interrupciones, modo de operación, etc.).

10. Interrupciones de reloj y control de ejecución

a) Explica cómo una interrupción de reloj puede usarse para:

Evitar que un programa quede en un bucle infinito.

Finalizar programas que superan un tiempo máximo de ejecución.

b) ¿Qué debe hacer el software si el programa finaliza antes de que ocurra la interrupción de reloj?

10.1. Explicación de cómo una interrupción de reloj puede usarse para:

Evitar que un programa quede en un bucle infinito.

- La interrupción de reloj permite la implementación de la **multitarea preventiva**. El sistema operativo configura un temporizador de hardware para generar una interrupción a intervalos regulares, por ejemplo, cada 10 milisegundos.
- Cada vez que ocurre esta interrupción, el control de la CPU se transfiere forzosamente al kernel del sistema operativo.
- El kernel, al detectar la interrupción, puede examinar el estado del programa que se estaba ejecutando. Si el mismo programa ha estado usando el procesador por un tiempo excesivamente largo (sin ceder el control), el kernel puede detectar que está .ªtascado.en un bucle infinito y tomar medidas, como terminar el proceso o asignarle un tiempo de ejecución más corto.
- De esta forma, ningún programa puede monopolizar la CPU indefinidamente, lo que evita que el sistema se cuelgue.

Finalizar programas que superan un tiempo máximo de ejecución.

- Esta es una aplicación directa del mismo mecanismo de interrupción. Cuando un programa se inicia, el sistema operativo le asigna lapso de tiempo de CPU.
- El temporizador de hardware se configura para que genere una interrupción al expirar este lapso de tiempo de CPU.
- Si un programa excede el tiempo asignado (es decir, no ha terminado antes de que ocurra la interrupción), el manejador de interrupciones del reloj se activa.

- En ese momento, el kernel sabe que el programa ha agotado su tiempo. La rutina de servicio del reloj, a través de su planificador, guarda el estado del programa actual, lo marca como "listoz elige otro programa para darle el control de la CPU.
- Si el tiempo máximo de ejecución de un programa se define como el total de sus lapsos de tiempo de CPU asignados, el sistema operativo puede decidir finalizar el programa en lugar de volver a ponerlo en la cola de "listos".

10.2. ¿Qué debe hacer el software si el programa finaliza antes de que ocurra la interrupción de reloj?

Si un programa finaliza por sí mismo (por ejemplo, mediante una llamada al sistema `exit`) antes de que se genere la interrupción de reloj programada, el software del sistema operativo debe realizar las siguientes acciones para gestionar la situación correctamente:

1. **Detectar la finalización:** El programa ejecutará una llamada al sistema (*syscall*) para solicitar su terminación. Esta *syscall* genera una excepción de software que transfiere el control al kernel.
2. **Cancelar la interrupción de reloj:** El kernel, al procesar la llamada de terminación, debe **desactivar proactivamente la interrupción de reloj pendiente** para ese proceso. Si no lo hiciera, la interrupción ocurriría de todos modos y sería espuria, ya que el proceso al que estaba asignado el *quantum* ya no existe. Esto podría llevar a un estado inconsistente en el planificador.
3. **Liberar recursos:** El sistema operativo lleva a cabo la limpieza del proceso. Esto incluye liberar la memoria, los archivos abiertos y otros recursos que el programa estaba utilizando.
4. **Seleccionar un nuevo proceso:** El planificador del sistema operativo selecciona inmediatamente otro proceso de la cola de "listos"para que se ejecute, sin tener que esperar a la interrupción del temporizador.

11. Análisis y Discusión de los Resultados.

Podemos concluir que una interrupción de hardware permite que los dispositivos externos puedan solicitar la atención de la CPU de manera eficiente, sin necesidad de que la CPU esté constantemente verificando el estado de cada dispositivo. El ciclo de interrupción garantiza que las tareas importantes no se pierdan y que los dispositivos puedan recibir la atención necesaria de forma oportuna.

Es importante es conocer las caracteristicas de las interrupciones y las excepciones, ambos son mecanismos para el rendimiento de sistemas informaticos, pero muy distintos entre sí, mientras que las interrupciones son eventos externos al computador, que por lo general utilizan hardware como lo puede ser el teclado y que tienen generalmente un ciclo de reloj, las excepciones son eventos internos del CPU involuntarios que terminan con el programa que se esté realizando, por ello es que existen estrategias de prevención de exenciones.

Hay que mencionar que la correcta gestión de interrupciones depende de guardar y restaurar el contexto del programa, incluyendo los registros de propósito general, el registro Status y el registro EPC, el cual es un registro fundamental porque almacena la dirección de la instrucción donde ocurrió la excepción, lo que permite al sistema volver al punto correcto del programa una vez que se ha manejado el.