Tema 5

Sistema de Entrada/Salida

5.1. Introducción

Una de las características básicas de un computador es su habilidad para intercambiar datos con otros dispositivos. Esta capacidad permite también su comunicación con el ser humano, tanto para aceptar datos como para comunicar resultados.

Existen una gran variedad de dispositivos, denominados dispositivos de entrada/salida (E/S) o periféricos, que permiten la comunicación del computador con su entorno. Las conexiones entre dispositivos de E/S, procesador y memoria se llevan a cabo mediante los buses. Un bus es un canal de comunicación compartido que utiliza un conjunto de cables para conectar múltiples subsistemas.

Para realizar la conexión entre la CPU y los dispositivos periféricos se necesita de módulos de E/S debido a que los dispositivos de E/S:

- Tienen unas velocidades de transmisión muy variadas, pero sensiblemente inferiores a la de la CPU.
- Suelen tener un ancho de palabra que no coincide con el ancho de palabra de la mayoría de los computadores. Pueden transmitir bits individuales, bytes sueltos, etc.
- Algunos son de lectura (por ej. el teclado), otros de escritura (por ej. el monitor) y otros simultáneamente de lectura y escritura (por ej. el modem).

La figura 5.1 muestra diferentes tipos de periféricos indicando si son de entrada, de salida o de almacenamiento, con quien interactúan (hombre o máquina) y sus velocidades de transferencia.

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	Input	Human	0.0001
Mouse	Input	Human	0.0038
Voice input	Input	Human	0.2640
Sound input	Input	Machine	3.0000
Scanner	Input	Human	3.2000
Voice output	Output	Human	0.2640
Sound output	Output	Human	8.0000
Laser printer	Output	Human	3.2000
Graphics display	Output	Human	800.0000–8000.0000
Cable modem	Input or output	Machine	0.1280–6.0000
Network/LAN	Input or output	Machine	100.0000–10000.0000
Network/wireless LAN	Input or output	Machine	11.0000–54.0000
Optical disk	Storage	Machine	80.0000–220.0000
Magnetic tape	Storage	Machine	5.0000–120.0000
Flash memory	Storage	Machine	32.0000–200.0000
Magnetic disk	Storage	Machine	800.0000–3000.0000

Figura 5.1: Dispositivos de E/S

5.2. Buses

Un bus es una vía de comunicación que conecta dos o más dispositivos. La característica clave de un bus es que se trata de un medio de transmisión compartido.

Las líneas que componen un bus se pueden clasificar en tres grupos funcionales:

- Las **líneas de datos** del bus proporcionan el camino para transmitir datos entre los módulos del sistema. El número de líneas del bus de datos determina el número máximo de bits que es posible transmitir al mismo tiempo.
- Las **líneas de dirección** se utilizan para designar (direccionar) la fuente o el destino de los datos enviados por el bus de datos. La anchura del bus de direcciones determina la cantidad máxima de memoria (y de dispositivos de E/S) direccionable en el sistema.
- Las **líneas de control** se emplean para gestionar el acceso y el uso de las líneas de datos y dirección, señalizando peticiones y reconocimientos e indicando qué tipo de información pasa por las líneas de datos.

La temporización es una parte fundamental del funcionamiento del bus ya que controla la forma en que se coordinan los sucesos sobre el bus:

- La **temporización síncrona** utiliza un protocolo para la comunicación que está gobernado por una señal de reloj. Todos los dispositivos conectados deben funcionar a la misma frecuencia de reloj. Se puede implementar con un sistema secuencial sencillo y puede funcionar a gran velocidad, pero no es adecuado para mezclar dispositivos con grandes diferencias de velocidad. Tiene una desventaja o problema fundamental que es el sesgo de la señal de reloj (*clock skew*). El sesgo de reloj es la

diferencia, en tiempo absoluto, entre los instantes en que dos elementos de estado reciben el flanco de reloj. Por culpa del sesgo los buses no pueden ser largos si son rápidos. Los buses de memoria suelen ser síncronos.

- En la **temporización asíncrona**, por el contrario, la ocurrencia de un suceso sobre el bus sigue otro suceso previo sin que tenga que producirse en un instante concreto simultáneamente con la aparición de un flanco de reloj. Los dispositivos implicados en la transferencia se coordinan mediante el intercambio de señales de control (**protocolo de *handshaking***). Ya que no hay señal de reloj común, los buses asíncronos pueden interconectar una gran variedad de dispositivos de diferentes velocidades. Como no hay problemas de sesgo de reloj, permiten distancias más largas. El inconveniente es que son más lentos debido a la sobrecarga introducida para sincronizar emisor y receptor y es más difícil predecir el tiempo que va a llevar una determinada transacción. Los buses de E/S son habitualmente asíncronos.

Al bus se conectan múltiples dispositivos y una señal transmitida por cualquiera de ellos puede ser recibida por todas las otras unidades conectadas. En un mismo instante de tiempo sólo es posible la transmisión por parte de un único dispositivo, sin embargo la recepción puede ser realizada por varios simultáneamente. Si dos dispositivos intentan transmitir en el mismo período de tiempo sus señales se solaparían y no se podrían recuperar, por lo que existen mecanismos de arbitraje.

Existen 2 tipos de arbitraje básicos, el centralizado y el distribuido. En el **arbitraje centralizado**, el controlador del bus o árbitro es el responsable de asignar el tiempo de utilización del bus. En el **arbitraje distribuido** no existe un controlador central. Cada módulo tiene la lógica de control suficiente para poder acceder al bus y todos actúan de forma cooperativa. Esto puede ser por autoselección, cuando cada dispositivo determina de forma independiente si él es el solicitante de más prioridad; o por detección de colisión, cuando varios dispositivos intentan acceder al bus a la vez se produce una colisión que es detectable. Una posible solución es esperar un tiempo aleatorio antes de volver a intentarlo. El mismo mecanismo se utiliza en muchas redes de datos.

5.3. Módulos de Entrada/Salida

Un módulo de E/S actúa como interfaz entre el procesador y uno o más dispositivos periféricos. Las funciones de los módulos de E/S son:

- Control y temporización: puesto que los recursos internos, como memoria principal y bus del sistema, deben compartirse entre distintas actividades, incluyendo la E/S.
- Comunicación con el procesador: que implica el decodificar órdenes que le llegan del procesador, intercambiar datos, informar de su estado,...
- Comunicación con los dispositivos: que implica intercambiar órdenes, información de estado y datos.

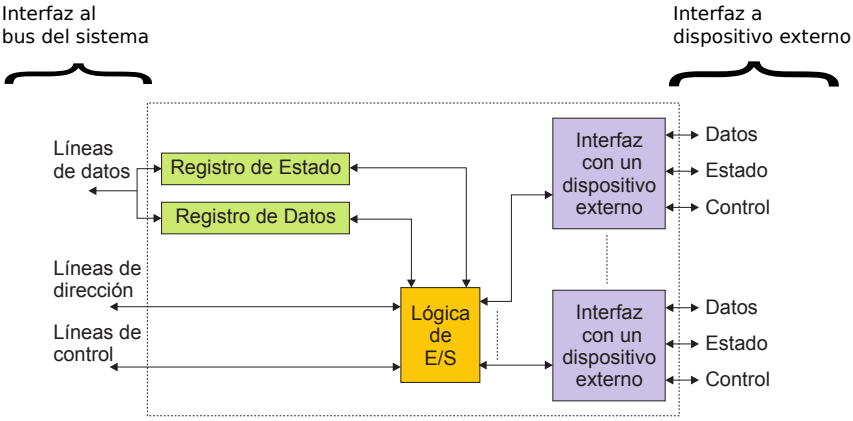

Figura 5.2: Diagrama de bloques del módulo de E/S

- Almacenamiento temporal de datos: puesto que la velocidad de transferencia desde y hacia la memoria principal y el procesador puede ser varios órdenes de magnitud superior que la de la mayoría de dispositivos periféricos. En ocasiones se reciben grandes bloques de datos que se emiten bit a bit hacia el periférico, ó bien se leen bit a bit y se agrupan antes de sacarlos del módulo.

- Detección de errores y comunicación de estos al procesador.

La Figura 5.2 muestra un diagrama de bloques de un módulo de E/S. El módulo se conecta al resto del computador a través de un conjunto de líneas (líneas de datos, líneas de dirección y líneas de control). Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos. Además, puede haber uno o más registros de estado que proporcionan información del estado presente. Estos registros internos pueden ser leídos y escritos por la CPU y son el medio de intercambio de datos con los dispositivos externos. La lógica que hay en el módulo interactúa con el procesador a través de una serie de líneas de control. Estas líneas las utiliza el procesador para proporcionar las órdenes al módulo de E/S. Algunas de estas líneas de control pueden ser utilizadas por el módulo de E/S (por ejemplo, para las señales de arbitraje del bus). El módulo también debe ser capaz de reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un conjunto único de direcciones. Por último, el módulo de E/S también posee lógica específica para la interfaz con cada uno de los dispositivos que controla.

Cuando el procesador precisa transferir datos con un periférico, comienza colocando en el bus de direcciones la dirección del módulo con el que tiene que comunicarse. Después el módulo informa a la CPU del estado del periférico (ocupado, libre desconectado, etc.). Si el periférico se encuentra disponible, el procesador solicita la transferencia mediante una orden al módulo de E/S. Cuando el periférico es de entrada, como sucede por ejemplo con el teclado, el carácter correspondiente a la tecla pulsada se almacena en el registro de datos del módulo de E/S y de allí se envía a la CPU. Si es de salida, como por ejemplo

una impresora, el carácter a imprimir lo envía el procesador por el bus de datos al registro de datos del módulo de E/S y de allí se manda a la impresora.

Existen módulos de E/S específicos para determinado tipo de periféricos, así como otros que se adaptan al funcionamiento de varios. Distinguiremos entre 2 tipos de módulos de E/S: utilizaremos el término **controlador** para los módulos de entrada salida más básicos, y el de **procesador** para aquellos con capacidad de ejecutar programas.

5.4. Direccionamiento de los módulos de E/S

Como ya hemos visto en la sección anterior, la comunicación entre la CPU y el módulo de E/S se realiza a través de los registros de estado y de datos, por tanto, estos registros tienen que poder ser direccionables por el procesador. La forma de direccionamiento varía según la arquitectura del procesador, existiendo dos aproximaciones:

- **E/S asignada al espacio de memoria** (*Memory-mapped I/O*, *MMIO*). La CPU trata los registros de los dispositivos de E/S como posiciones de memoria, utilizando las mismas instrucciones para acceso a memoria y a dispositivos de E/S. Con este esquema, una parte del mapa de memoria del sistema es reservado para E/S en lugar de para memoria, asignado las direcciones en ese rango a los distintos dispositivos de E/S. De esta forma, existe un único espacio de direcciones en el sistema, que se emplea tanto para la memoria principal como para la E/S. La ventaja es su sencillez, ya que no es necesario implementar instrucciones ni circuiteria especial para la E/S. Esta alternativa se utiliza sobre todo en arquitecturas RISC y sistemas empotrados[1]. Ejemplos de sistemas que utilizan esta aproximación son la familia de procesadores PowerPC de IBM y la familia de procesadores MIPS. En el caso del MIPS, las direcciones de memoria a partir de la `0x80000000` son solo accesibles en modo kernel (ver figura 5.3). Esta región incluye código para el manejo de excepciones, datos solo accesibles por el Sistema Operativo (SO) y direcciones de E/S asignadas a memoria que se utilizan para referirse a los registros de estado y de datos de los dispositivos periféricos (las direcciones de memoria de la `0xffff0000` en adelante).

- **E/S aislada** (*Isolated I/O*, *Instruction-based I/O*, *Port I/O* o *Port-mapped I/O*, *PMIO*). En este caso se mantienen espacios de direcciones diferentes para memoria y E/S, disponiendo el procesador de instrucciones especiales para el acceso a direcciones de E/S, distintas a las de acceso a memoria. Al tener los dispositivos de E/S un mapa de memoria separado de la memoria principal, la CPU utiliza una señal de control ($\bar{M}/IO$) para indicar si la dirección que está en el bus de memoria es de E/S, o bien se utiliza un bus de E/S propio. Ejemplos de sistemas que usan esta aproximación son la familia de procesadores x86 de Intel.

[1] Entendiendo por sistema empotrado un sistema computador de propósito especial contenido dentro de un dispositivo mayor.

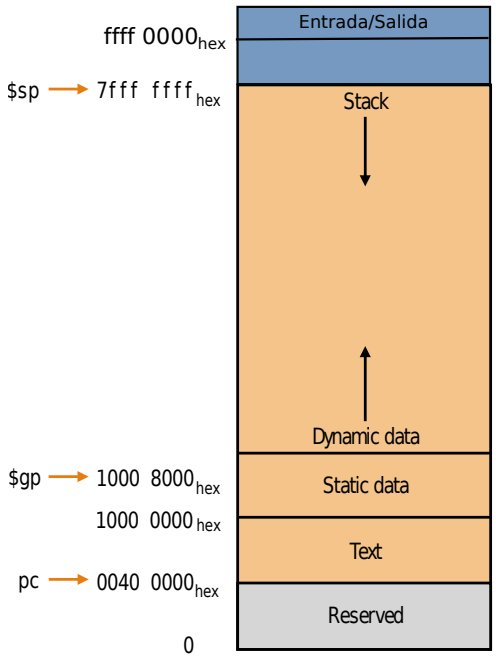

Figura 5.3: División del espacio de memoria en el MIPS

5.5. Gestión de la Entrada/Salida

Hasta el momento hemos hablado de las características de los módulos de E/S. Ahora veremos como se realiza la comunicación entre los módulos y la CPU. Hay 3 técnicas posibles:

- E/S **programada**: El programa corriendo en el procesador toma la iniciativa de conectar con los dispositivos de E/S y gestiona las transferencias de datos. Es la técnica más básica y la menos eficiente.
- E/S mediante **interrupciones**: Los dispositivos de E/S realizan una petición de interrupción al procesador. El programa se interrumpe mientras se gestiona la petición y al terminar esta se reanuda la ejecución. Mejora a la anterior ya que evita que la CPU tenga que comprobar continuamente el estado del dispositivo.
- **Acceso directo a memoria**: Existe un controlador especial que realiza el intercambio de datos entre memoria y los dispositivos de E/S mientras el procesador continúa su ejecución. Es la técnica más avanzada y la que mayor rendimiento permite alcanzar.

5.5.1. E/S programada

Es la técnica de gestión de E/S más simple. Los datos son intercambiados directamente entre el procesador y el dispositivo de E/S. La CPU ejecuta un programa que le da un

control completo y directo sobre la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de ordenes de lectura y escritura, y la transferencia de datos.

Cada vez que la CPU le envía una orden al dispositivo tiene que esperar a que la operación de E/S se complete, interrogando periódicamente al dispositivo sobre su estado, lo que se conoce como **encuesta** (*polling*). El dispositivo de E/S nunca interrumpe al procesador tras acabar una operación, simplemente indica el hecho con sus bits de estado.

El principal inconveniente de la E/S programada es que, al ser el procesador mucho más rápido que el dispositivo, se están desperdiciando muchos ciclos de reloj en encuestar al dispositivo, degradando el rendimiento del sistema. La ventaja es que puede acelerar las operaciones de E/S. Suele utilizarse en sistemas de propósito específico como cajeros automáticos o sistemas empotrados para el control o monitorización de eventos.

La figura 5.4 proporciona un ejemplo del uso de la E/S programada para leer un bloque de datos desde un dispositivo periférico y almacenarlo en memoria. Los datos se leen palabra por palabra (32 bits, por ejemplo). Por cada palabra leída el procesador debe permanecer en un ciclo de comprobación de estado hasta que determine que la palabra está disponible en el registro de datos del módulo de E/S. Si hay un dato listo, entonces transfiere el dato a la CPU, si no, vuelve a leer el registro de estado. Este diagrama de flujo resalta la principal desventaja de esta técnica: mantiene ocupada la CPU sin que pueda hacer otra cosa.

5.5.2. E/S con interrupciones

Con el mecanismo de interrupciones la CPU delega en el dispositivo de E/S la responsabilidad de comunicarse con el procesador cuando lo necesite. En este caso, el procesador no hace *polling* de ningún dispositivo, sino que queda a la espera de que estos le avisen cuando tengan algo que comunicarle (ya sea un evento, una transferencia de información, una condición de error, etc.). Este aviso se realiza activando una línea de petición de interrupción. Cuando el procesador recibe esta señal salta a ejecutar una rutina de tratamiento de interrupción que se encarga de atender al periférico que solicitó la interrupción.

Una interrupción de E/S es como las excepciones vistas en el capítulo anterior. Ambas son sucesos inesperados que afectan al procesador, pero las excepciones son sucesos internos y las interrupciones se producen externamente al procesador. Además, una interrupción de E/S es asíncrona con respecto a la ejecución de las instrucciones, es decir, se pueden producir e cualquier momento, cosa que no ocurre con las excepciones.

La figura 5.5 muestra un ejemplo de uso de E/S con interrupciones para leer un bloque de datos. El programa que se está ejecutando en la CPU realiza una petición de lectura a un controlador de E/S y a continuación pasa a otra tarea. En algún momento el controlador avisa mediante una interrupción de que el dato está disponible. El procesador pasa el control al programa de tratamiento de esa interrupción, que leerá el dato y lo guardará en memoria. El proceso se repite si es necesario, pero la CPU sólo estará ocupada

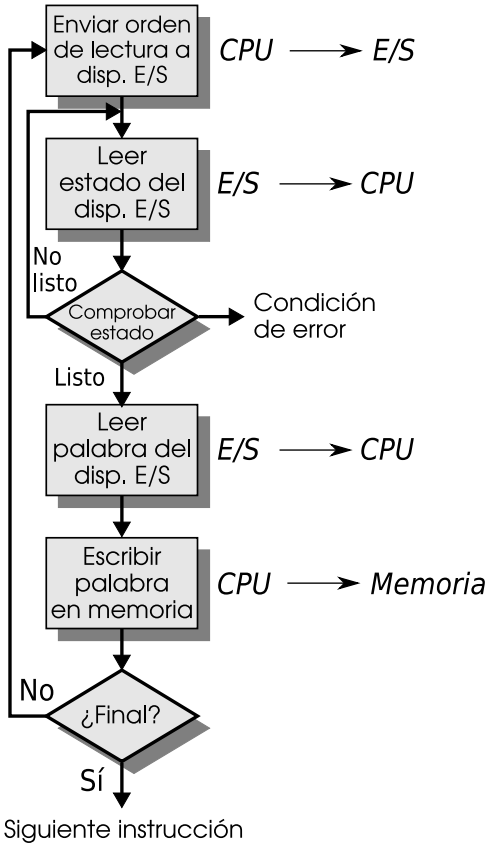

Figura 5.4: Ejemplo de E/S programada

el tiempo imprescindible. Con esta técnica, el procesador puede atender al programa principal mientras el periférico está manipulando la información. Se complica la lógica del controlador, en el que recae la misión de explorar el estado del periférico, para provocar una interrupción hardware cuando detecta que ya está disponible.

La secuencia de pasos en el tratamiento de una petición de interrupción es básicamente la siguiente:

1. El dispositivo envía una señal de interrupción al procesador.
2. El procesador termina la ejecución de la instrucción máquina en curso.
3. El procesador envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción.
4. El procesador se prepara para transferir el control a la rutina de interrupción. Salva el valor de contador de programa, PC, y los registros necesarios en la pila, de manera que la CPU, al terminar el proceso, pueda seguir ejecutando el programa a partir de la última instrucción.

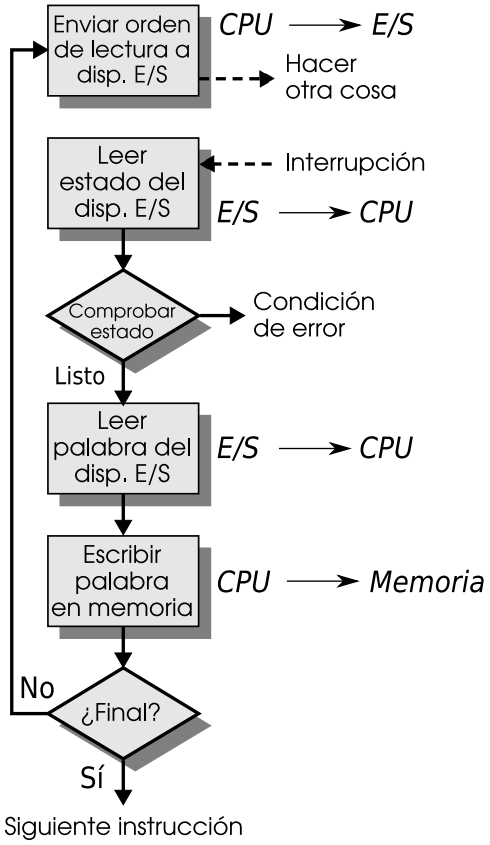

Figura 5.5: Ejemplo de E/S con interrupciones

5. El procesador carga en el PC la posición de inicio del programa de tratamiento de la interrupción solicitada (ISR, Interrupt Service Routine) y ejecuta esa rutina que tiene como objetivo atender al dispositivo que generó la interrupción. Según la arquitectura del computador y el diseño del SO, puede haber un solo programa, uno por cada tipo de interrupción o uno por cada dispositivo y cada tipo de interrupción. Si hay más de una rutina de gestión de interrupción, el procesador debe determinar qué programa llamar.

6. Una vez que la rutina de interrupción termina, el procesador restaura el estado que había guardado en la pila en el paso 4 y retorna al programa que se estaba ejecutando anteriormente.

El mecanismo anterior es muy simple, pero es solo válido cuando existe un único periférico conectado a la línea de petición de interrupciones. Cuando hay más de un dispositivo y cada uno de ellos puede generar de forma independiente la petición de una interrupción, surge la necesidad de identificar al periférico que ha generado la interrupción, decidir cuál tiene preferencia sobre los demás y proteger los servicios de interrupción de otras interrupciones.

La identificación del dispositivo se puede realizar mediante **líneas de interrupción**

dedicadas. En este caso cada dispositivo tiene asignada una línea de interrupción propia. El número de líneas de bus y de conexiones de la CPU que se pueden dedicar a líneas de interrupciones es limitado, por lo que esta técnica por si misma suele ser insuficiente. La alternativa es utilizar **líneas de interrupción compartidas**, es decir, cada línea de interrupción puede ser empleada por más de un módulo de E/S. En este caso, es necesario un mecanismo de identificación del módulo de E/S que provocó la interrupción. La identificación del dispositivo concreto se puede realizar de varias maneras, de las cuales veremos 3:

- **Encuesta software**: al detectar la CPU que se ha activado la línea de petición de interrupción, una rutina genérica de manejo de interrupciones se encarga de interrogar a los diferentes dispositivos comprobando el valor del registro de estado. Una vez identificado el dispositivo se ejecuta su rutina de servicio asociada.
- **Encadenamiento (daisy chain)**: cuando el procesador recibe una interrupción, activa la línea de reconocimiento de interrupción. La línea de reconocimiento de interrupción se conecta en cadena a todos los módulos de E/S. De esta forma, el primer módulo que recibe dicha señal y ha generado una interrupción, deja de propagar la señal y suministra, a través del bus de datos, la dirección en la que se encuentra su rutina de tratamiento. El proceso se ilustra en la figura 5.6. El periférico x hace la petición, lo que provoca $PI_x = 1$ y, en consecuencia, $PI = 1$. La CPU reconoce la petición de interrupción y activa RI. La señal RI se recibe en el primer periférico, si ese periférico no ha originado la petición propaga la señal RI activándola a su salida. El paso anterior se repite hasta que la señal llega al periférico que ha generado la interrupción, el cual bloquea la señal RI poniéndola a 0. El periférico entonces envía a la CPU la dirección de la rutina de tratamiento de su interrupción por el bus de datos. En el suministro de la dirección de la rutina de tratamiento se puede aplicar cualquiera de los métodos de direccionamiento que se estudiaron para las instrucciones, aunque se suelen imponer ciertas limitaciones o simplificaciones. Algunas alternativas clásicas son:
  - Direccionamiento absoluto: el periférico manda su dirección.
  - Direccionamiento relativo: el periférico manda parte de la dirección y la CPU la completa añadiendo bits o sumando una determinada cantidad, que siempre será fija. Esta alternativa tiene una ventaja sobre la anterior y es que permite especificar la dirección de comienzo con menos bits y por tanto simplifica el diseño. Ahora bien, tiene una desventaja principal y es que limita el número de dispositivos que podemos conectar.
  - Direccionamiento relativo indirecto: el periférico manda la posición relativa de la dirección en una tabla de direcciones residente en memoria. En este caso se habla de **interrupciones vectorizadas**
- **Arbitraje de bus**: un módulo de E/S debe, en primer lugar, disponer del control del bus antes de poder activar la línea de petición de interrupción. Así, solo un módulo puede activar la línea en un instante. Cuando el procesador detecta la interrupción,

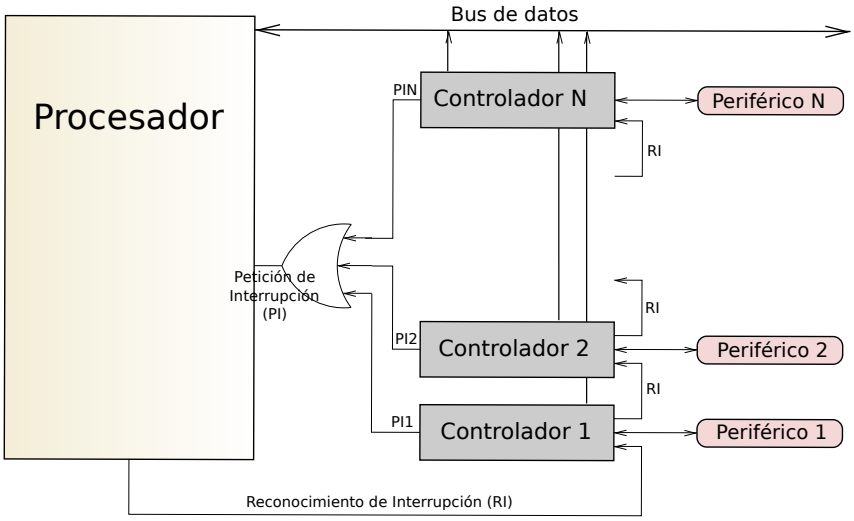

Figura 5.6: Encadenamiento de periféricos

responde mediante la línea de reconocimiento de interrupción. Después el módulo que solicitó la interrupción sitúa la dirección de su rutina de tratamiento en las líneas de datos.

Las técnicas de arriba sirven para identificar el módulo de E/S que solicita interrupción y para asignar prioridades cuando más de un dispositivo está pidiendo que se atienda su interrupción. Con varias líneas de interrupción, el procesador simplemente selecciona la línea con más prioridad. Con consulta software, el orden en el que se consultan los módulos determina la prioridad. De igual forma, el orden de los módulos en la conexión en cadena determina la prioridad. Finalmente, el arbitraje de bus puede emplear también un esquema de prioridad.

Si una petición de interrupción se produce mientras se atiende una anterior existen 2 alternativas. En un **sistema de interrupciones único** la ejecución del programa de servicio de una interrupción continúa hasta el final antes de la que CPU pueda aceptar una segunda petición de interrupciones. En un **sistema multinivel** se pueden atender peticiones de interrupción durante la ejecución del programa de servicio de otro dispositivo. En este caso se le asigna a cada causa de interrupción uno de entre varios niveles de prioridad de modo que una interrupción solo se atiende si su nivel es superior al de la interrupción cuyo programa de servicio se está ejecutando.

La forma de ignorar interrupciones es mediante el enmascaramiento (ver figura 5.7). Con este mecanismo se puede evitar que una interrupción se vea a su vez interrumpida por otra de igual o menor nivel de prioridad, permitiendo la ejecución sin interrupciones de tareas críticas y/o evitando que dispositivos lentos interrumpan a otros de mayor prioridad. También se puede pedir que ninguna interrupción interrumpa a la actual. Las excepciones/interrupciones más críticas, de hecho, no son enmascarables y tienen prioridad absoluta.

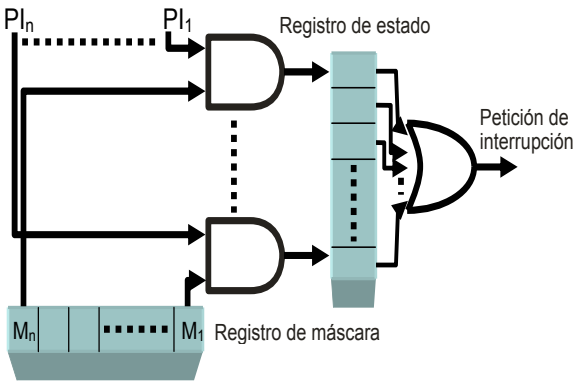

Figura 5.7: Enmascaramiento de las interrupciones

Al invocar un servicio de interrupción se pone en marcha un mecanismo muy similar a la llamada a una subrutina en el que la pila juega un papel muy importante tal y como se muestra en la figura 5.8. En este ejemplo el programa principal es interrumpido por una petición con código de interrupción 2. El programa que da servicio a esta interrupción se comienza a ejecutar, y entonces se recibe una petición de interrupción PI3 que es rechazada porque su nivel de prioridad es inferior a la actual. Posteriormente se recibe otra petición PI1 que sí debe ser atendida porque tiene mayor prioridad que PI2. Al terminar la ejecución del programa que da servicio a PI1 se regresa al programa de PI2, que también termina su ejecución y, antes de regresar al programa principal, el procesador permite la ejecución del programa de servicio de PI3, que ha estado pendiente todo ese tiempo. Finalmente se vuelve al programa principal. En la parte de abajo de la figura se ve esto mismo desde el punto de vista de la pila, del registro contador de programa (PC) y de un registro de excepciones (RE) que contiene información sobre la excepción o interrupción que está siendo atendida. En la pila se van guardando los valores de PC a los que hay que retornar (n+1 y q+1) y los valores de RE.

Interrupciones en el MIPS

Para el manejo de excepciones e interrupciones el MIPS utiliza el *coprocesador 0*. En este coprocesador existen una serie de registros que se utilizan para tratar este tipo de eventos:

- El registro *EPC* (*Exception Program Counter*): contiene la dirección de la instrucción que se estaba ejecutando cuando se ha producido la excepción.
- El registro *Status*: contiene la máscara de interrupciones y los bits de autorización (ver figura 5.9):

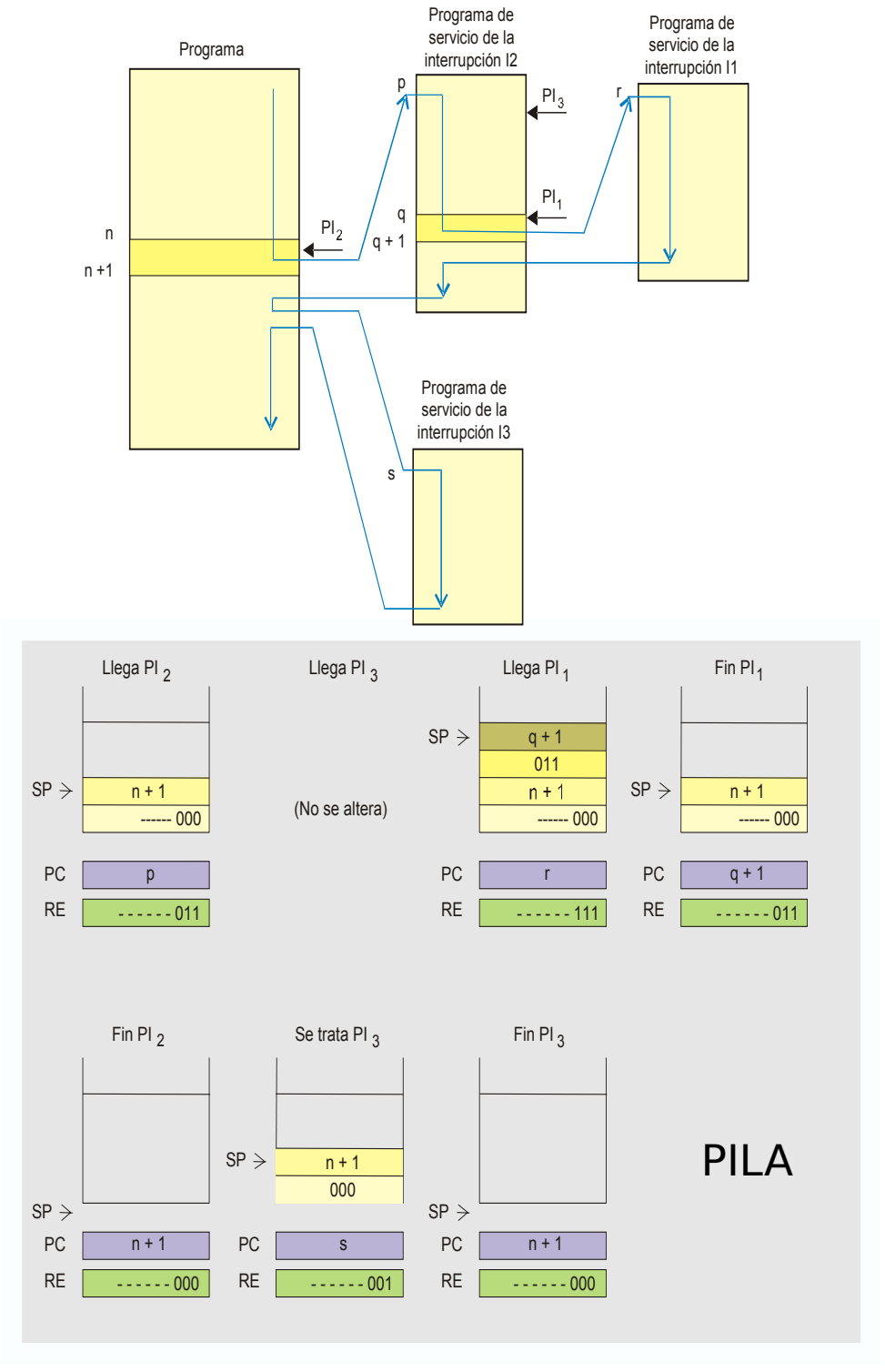

Figura 5.8: Cambios en memoria y en registros debido a interrupciones

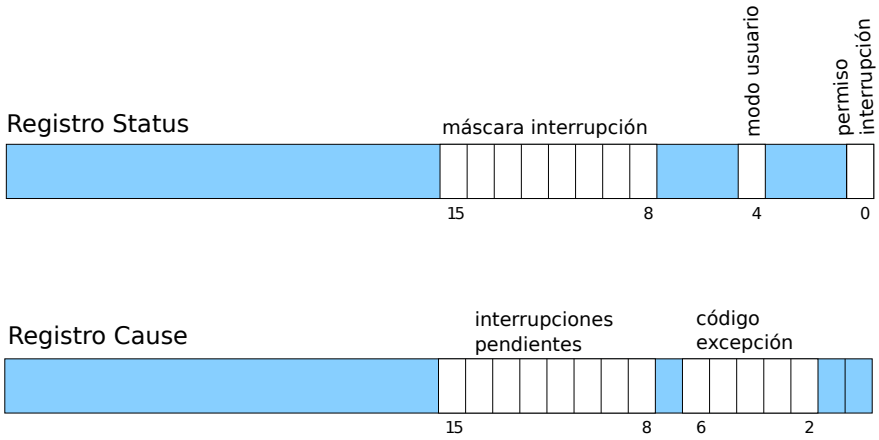

Figura 5.9: Principales bits de los registros *Status* y *Cause* del coprocesador 0 del MIPS

- El campo *máscara de interrupciones* contiene un bit para cada uno de los niveles de interrupción posibles (5 niveles hardware y 3 software). Un bit a 1 permite interrupción en ese nivel, un bit a 0 deshabilita las interrupciones de ese nivel.
- El bit *modo usuario* vale 0 si el programa estaba en el núcleo del SO cuando se ha producido la excepción y 1 si se estaba ejecutando en modo usuario.
- El bit *permiso interrupción* está a 1 si se permiten las interrupciones y a 0 si están deshabilitadas.

- El registro *Cause*: se utiliza para identificar el motivo de la excepción (ver figura 5.9):
  - Los 8 bits *interrupciones pendientes* se corresponden con los 8 niveles de interrupción. Cada bit pasa a 1 cuando se ha producido una interrupción en su nivel y no ha sido atendida.
  - El campo *código excepción* describe la causa de la excepción/interrupción. El valor 0 en este campo indica una interrupción externa y la rutina de manejo de interrupciones tiene que averiguar el causante de la interrupción encuestando a los dispositivos que están siendo manejados mediante interrupciones (*encuesta software*).

5.5.3. Acceso directo a memoria

El uso de E/S mediante interrupciones plantea problemas de rendimiento para grandes transferencias de datos. Por ejemplo, si se dispone de un módulo de E/S con un registro de datos de 1 byte y se desea realizar la transferencia de un bloque de 512 bytes, se producen 512 interrupciones y, por tanto, la rutina de tratamiento de interrupciones se ejecutara 512 veces. Esto implica que la operación de E/S todavía requiere una considerable utilización del procesador para la operación.

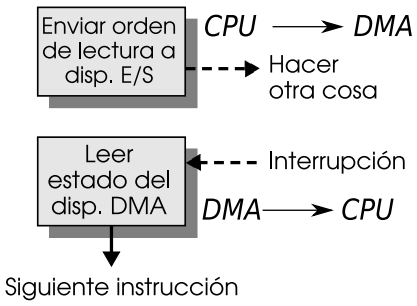

Figura 5.10: Ejemplo de E/S con acceso directo a memoria

Cuando hay que transferir grandes volúmenes de datos se requiere una técnica más eficiente: el acceso directo a memoria (*DMA, Direct Memory Access*).

El acceso directo a memoria se realiza con un controlador especializado que transfiere los datos entre el dispositivo y la memoria sin intervención del procesador. El dispositivo continúa usando el mecanismo de interrupciones para comunicarse con el procesador, pero solo al finalizar la transferencia o en caso de error.

La figura 5.10 proporciona un ejemplo del uso de la E/S con DMA para leer un bloque de datos desde un dispositivo periférico y almacenarlo en memoria. Hay tres pasos en una transferencia por DMA:

- El procesador inicia el DMA proporcionando la identidad del dispositivo, la operación a realizar, la dirección de memoria que es la fuente o destino de los datos a transferir y el número de bytes a transferir.
- El DMA inicia la operación en el dispositivo y se encarga de obtener el acceso al bus. Cuando los datos están disponibles (procedentes del dispositivo o de memoria), los transfiere. El controlador de DMA proporciona la dirección de memoria para las lecturas o escrituras. Si la petición requiere más de un transferencia por el bus, el controlador de DMA genera la siguiente dirección de memoria e inicia la siguiente transferencia. Usando este mecanismo, el controlador DMA completa la transferencia, que puede tener un tamaño de miles de bytes, sin molestar al procesador.
- Una vez finalizada la transferencia por DMA, el controlador interrumpe al procesador, el cual puede determinar, interrogando al controlador de DMA o examinando la memoria, si la operación ha terminado con éxito.

Las opciones para el acceso al bus por parte del controlador DMA son:

- Por **ráfagas**: cuando el DMA toma el control del bus no lo libera hasta haber transmitido el bloque de datos pedido.
- Por **robo de ciclos**: cuando el DMA toma el control del bus lo retiene solo un ciclo, transmite una palabra y lo libera.

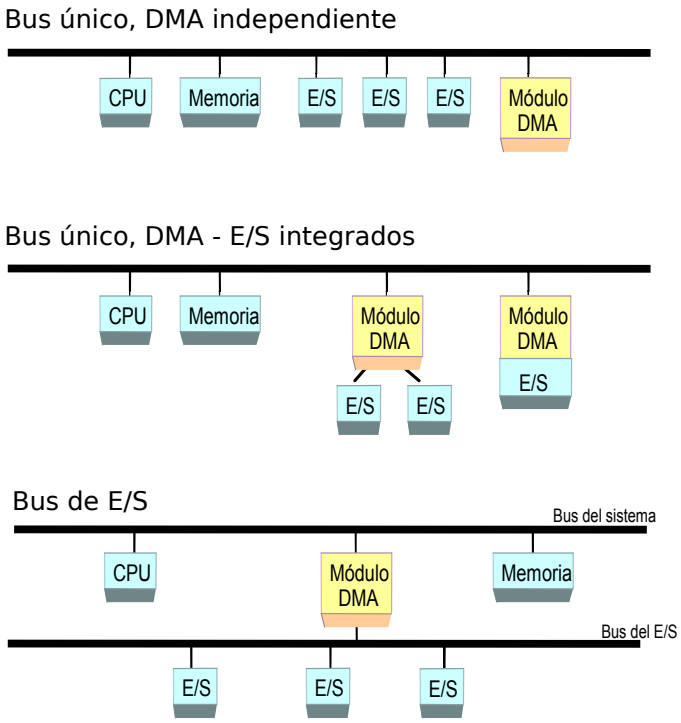

Figura 5.11: Configuraciones alternativas para el DMA

- DMA **transparente**: solo se roban ciclos cuando la CPU no está utilizando el bus del sistema. Elimina completamente la interferencia entre el controlador de DMA y la CPU.

Una operación de E/S gestionada mediante DMA puede degradar el rendimiento del procesador si se hace un uso intensivo del bus de memoria, ya que si el bus está ocupado en una transferencia DMA, el procesador no puede acceder a memoria principal. Existen varias configuraciones de DMA posibles según la compartición que se haga del bus. La figura 5.11 muestra la diferentes alternativas. En el caso más básico, todos los elementos comparten el bus y el tráfico es más intenso, con posibilidad de que se reduzca el rendimiento. Esto se puede evitar haciendo que los módulos de E/S no tengan acceso directo al bus, así solo existirá tráfico entre el módulo DMA y la memoria. Finalmente, puede existir un bus específico de E/S.

5.6. Procesadores de E/S

Para reducir todavía más la interferencia de la E/S con la CPU, los controladores de E/S se pueden hacer más inteligentes, haciendo que se comporten como un procesador en

sí mismo, con un repertorio de instrucciones especializado orientado a la E/S. Esto permite que la CPU pueda especificar una secuencia de actividades de E/S y ser interrumpida cuando se haya completado la secuencia entera. En este caso el módulo de E/S recibe el nombre procesador de E/S en vez de controlador de E/S.