CAPÍTULO 7

Entrada/Salida: introducción

Índice

La entrada/salida es el componente de un ordenador que se encarga de permitir su interacción con el mundo exterior. Si un ordenador no dispusiera de entrada/salida, sería totalmente inútil, con independencia de la potencia de su procesador y la cantidad de memoria, pues no podría realizar ninguna tarea que debiera manifestarse fuera de sus circuitos electrónicos. De la misma forma que se justifica su necesidad, se explica la variedad de la entrada/salida y de ahí su problemática: el mundo exterior, lejos de ser de la misma naturaleza electrónica que los circuitos del ordenador, se caracteriza por su variedad y su cambio. La entrada/salida debe ser capaz de relacionarse con este mundo diverso y, a la vez, con los dispositivos electrónicos del ordenador.

Los siguientes apartados explican cómo puede gestionarse esta relación haciendo que la entrada/salida sea a la vez, versátil, eficaz y manejable.

7.1. Generalidades y problemática de la entrada/salida

La primera imagen que se nos viene a la cabeza al pensar en un sistema informático es el ordenador personal, con un teclado y un ratón para interactuar con él y un monitor para recibir las respuestas de forma visual. Posiblemente tengamos en el mismo equipo unos altavoces para reproducir audio, una impresora para generar copias de nuestros trabajos, un disco duro externo y, por supuesto, una conexión a internet -aunque pueda no verse al no utilizar cables-. Todos estos elementos enumerados, aunque sean de naturaleza y función completamente distinta, se consideran dispositivos periféricos del ordenador y son una parte -en algunos casos la más alejada del ordenador, como los altavoces- de su entrada/salida.

Considerando la dirección, siempre referida al ordenador, en que fluyen los datos, vemos que unos son de salida, como la impresora o el monitor, otros de entrada, como el teclado y el ratón, mientras que algunos sirven como de entrada y de salida, como el disco duro y la conexión de red. La dirección de los datos se denomina *comportamiento* y es una característica propia de cada dispositivo. Podemos ver además que los dos últimos dispositivos del párrafo anterior no sirven para comunicarse con el usuario -un ser humano- a diferencia del teclado, ratón o monitor. Esto nos permite identificar otra característica de los dispositivos, que es su interlocutor, entendido como el ente que recibe o genera los datos que el dispositivo comunica con el ordenador. Entre los ejemplos anteriores es evidente determinar cuáles tienen un interlocutor humano. En el caso de la conexión de red, el interlocutor, a través de numerosos dispositivos intermedios -que normalmente también son pequeños ordenadores- acaba siendo otro ordenador personal o un servidor. En este ejemplo el interlocutor es una máquina, como ocurre también con otros muchos ordenadores presentes en sistemas empotrados que se comunican con controladores de motores, sistemas de regulación de iluminación u otra maquinaria generalmente electrónica o eléctrica. Pero hoy en día que los ordenadores están extendidos en todos los campos de la actividad humana, podemos tener uno regulando la temperatura de una caldera de vapor -con sensores midiendo la temperatura del aire en su interior-, midiendo la humedad del terreno en un campo de cultivo o el nivel de concentración de cierto soluto en una reacción química. En estos últimos ejemplos el interlocutor no es un ser humano ni una má-

quina, sino un sistema o fenómeno natural -dado que la entrada/salida comunica el ordenador con el mundo exterior-. Esta clasificación de interlocutores no pretende ser un dogma ni está exenta de consideraciones filosóficas. Según ella es evidente que una interfaz de un computador con las terminaciones nerviosas de un ratón en un experimento de bioingeniería no tiene interlocutor humano, pero ¿qué diríamos entonces si las terminaciones nerviosas fueran de una persona?

En el párrafo anterior hemos vuelto a comentar que la entrada/salida pone en contacto el ordenador con el mundo exterior. Esta afirmación no parece confirmarse cuando hablamos de un disco duro. Obviando el ejemplo del que hemos partido, el caso del disco externo, un disco duro es interno al ordenador y parte imprescindible de él, al menos para los ordenadores personales. Sin embargo, dado que el disco duro magnético basa su funcionamiento en piezas mecánicas en movimiento, participa de todas las demás características de los dispositivos de entrada/salida y por ello se considera como tal. Y una de estas características, especialmente importante en los discos duros, es la tasa de trasferencia de datos, es decir, la cantidad de datos por unidad de tiempo que intercambian ordenador y dispositivo. Esta tasa de transferencia influye mucho en la forma de gestionar la entrada/salida, adaptando a ella la forma de tratar cada dispositivo. Un teclado puede comunicar unos pocos bytes por segundo; un disco duro puede alcanzar varios gigabytes por segundo. Aunque todos los periféricos son lentos en relación con la velocidad del procesador -que puede tratar decenas de miles de millones de bytes por segundo- la propia diferencia de velocidades entre dispositivos requiere tratamientos bien diferenciados.

Además, la tasa de transferencia, considerada sin más, no describe correctamente el comportamiento temporal de los dispositivos, pudiendo ser medida de distintas formas, todas ellas correctas aunque no igualmente significativas. Veamos un par de ejemplos que permiten caracterizar mejor la tasa de transferencia. Comparando la reproducción de una película en alta definición almacenada en un disco duro con la pulsación de un teclado, es evidente que la primera actividad requiere mayor tasa de transferencia. Sin embargo, nuestra experiencia al disfrutar de la película no se verá mermada si, desde que ejecutamos el programa de reproducción hasta que aparecen las primeras imágenes transcurren diez o quince segundos. Sería imposible, por otra parte, trabajar con un ordenador si cada vez que pulsamos una tecla transcurrieran varios segundos -no ya diez, simplemente uno o dos- hasta que dicha pulsación se hace evidente en la respuesta del sistema. Estos ejemplos revelan los dos factores que caracterizan el comportamiento temporal de los dispositivos de entrada/salida:

- La *latencia*, que se entiende como el tiempo transcurrido desde

que se inicia una operación de entrada/salida hasta que el primer dato comunicado llega a su destino. En una operación de entrada sería el tiempo transcurrido desde que se inicia la petición de datos hasta que se recibe el primero de ellos. Un teclado, con una baja tasa de transferencia, requiere sin embargo una latencia de decenas de milisegundos para funcionar adecuadamente.

- La *productividad*, que se refiere a la cantidad de datos transferidos por unidad de tiempo, y que coincide con la primera definición que hemos dado de *tasa de transferencia*.

Al indicar un valor para la productividad se debe especificar adecuadamente cómo se ha realizado el cálculo. Teniendo en cuenta estas dos definiciones, la medida correcta de la productividad de una transacción debería incluir el tiempo de latencia, aunque durante él no se transmitan datos. En este caso la productividad vendría dada por la cantidad de datos recibidos dividida entre el tiempo transcurrido desde que se inició la transacción hasta que concluyó la recepción de datos. Si lo que se analiza es un dispositivo y no una transacción en particular, lo más ecuánime es dar una productividad media, teniendo en cuenta los tiempos de latencia y de transacción. Muchas veces se da, sin embargo, sobre todo en información comercial -orientada a demostrar correcta o incorrectamente las bondades de cierto producto-, la productividad máxima, que no tiene en cuenta el tiempo de latencia y considera el mejor caso posible para el funcionamiento del dispositivo.

En este apartado hemos presentado algunas generalidades de los dispositivos y sistemas de entrada/salida y hemos presentado tres propiedades que ayudan a su clasificación: su *comportamiento*, el *interlocutor* al que se aplican y la *tasa de transferencia*, matizada con los conceptos de *latencia* y *productividad*. Los sistemas de entrada/salida actuales son elevadamente complejos e incluso, por decirlo de alguna manera, jerárquicos. Estamos acostumbrados a utilizar dispositivos periféricos USB como los que hemos estado comentando -teclados, ratones, impresoras, etcétera-. Pues bien, el bus de entrada/salida USB -al igual que los SPI, I^2C y CAN, utilizados en sistemas empotrados- es a su vez un dispositivo periférico del ordenador, y debe ser tratado como tal. Una tarjeta de sonido conectada al bus PCI Express de un PC es un dispositivo periférico conectado directamente al sistema; una tarjeta igual -en su mayor parte- conectada a un bus USB es un dispositivo periférico conectado a un dispositivo de entrada/salida conectado al sistema. El tratamiento de ambas es idéntico en muchos aspectos, pero diferente en otros. Afortunadamente los sistemas operativos, a través de sus manejadores de dispositivos, estructurados de forma modular y jerárquica, son capaces de gestionar eficazmente esta complejidad. En este texto, en los siguien-

tes apartados, nos limitaremos a presentar los conceptos básicos de la estructura y la gestión de la entrada/salida, desde el punto de vista de la estructura de los computadores.

7.2. Estructura de los sistemas y dispositivos de entrada/salida

La función de la entrada/salida, como sabemos, es comunicar el ordenador con el mundo exterior. Si a esta afirmación unimos lo tratado en el apartado anterior, especialmente al hablar de los diferentes interlocutores de los dispositivos de entrada/salida, y la propia experiencia acerca de los incontables usos de los ordenadores en la vida actual, es fácil intuir que la estructura física de los elementos de entrada/salida es complicada e incluye diversas tecnologías. Efectivamente, todo dispositivo acaba relacionándose con el ordenador, por lo que dispone de circuitos electrónicos digitales de la misma tecnología. En el otro extremo, el dispositivo es capaz de generar luz, mover una rueda, medir la salinidad del agua o registrar los desplazamientos producidos en una palanca. Buena parte de esta estructura, pero no toda, es electrónica. Es sin embargo posible encontrar una estructura general a la que, como siempre con excepciones, se adaptan de una u otra forma todos los dispositivos de entrada/salida. Esta configuración incluye tres tipos de tecnología, que enumeradas desde el ordenador hacia el exterior serían las siguientes:

- Una parte formada por *circuitos electrónicos digitales* que comunica el dispositivo con el ordenador. Es la parte más genérica, propia del sistema informático y no de los diversos elementos del mundo exterior. Esta parte incluye todo lo necesario para la gestión de la entrada/salida en el ordenador, que iremos describiendo a lo largo de este documento.

- Una parte compuesta por *circuitos electrónicos analógicos*, que suele terminar en uno o varios componentes llamados *transductores* que transforman energía eléctrica en otro tipo de energía, o viceversa. Esta parte se encarga de la adaptación de los niveles eléctricos necesarios para comunicarse con el transductor, y de posibles tratamientos electrónicos de las señales -filtrados, amplificación, etcétera-.

- Una parte con componentes de una o varias *tecnologías no eléctricas* que comienza con los transductores y los adapta, en el ámbito de las magnitudes y características físicas propias del dispositivo.

En un ejemplo tan sencillo como un LED utilizado como dispositivo de salida, tenemos que la parte no eléctrica la constituye el propio encapsulado del diodo, con su color -o capacidad de difusión de la luz para un LED RGB- y su efecto de lente. Como vemos, ambas características son ópticas. La parte eléctrica estaría formada por el propio diodo semiconductor, que es en este caso el transductor, y la resistencia de polarización. La electrónica digital se encontraría en los circuitos de salida de propósito general -GPIO, como veremos más adelante- del microcontrolador al que conectamos el LED.

En el caso de un teclado comenzaríamos con las partes mecánicas de las teclas -incluyendo resortes y membranas, según el tipo- y los contactos eléctricos que completan los transductores. La electrónica analógica estaría formada por resistencias para adaptar los niveles eléctricos y diodos para evitar corrientes inversas. La parte digital, en los teclados más corrientes, la forma un microcontrolador que gestiona el teclado y encapsula la información de las teclas pulsadas en un formato estandarizado que se envía a través de un bus de entrada/salida estándar -USB hoy en día; antes PS/2 o el bus de teclado de los primeros PC-.

Si bien las tres partes mencionadas son necesarias en el funcionamiento del dispositivo, la gestión de la entrada/salida desde el ordenador requiere solo la primera, implementada con electrónica digital y compatible por tanto con el funcionamiento del resto de componentes del computador. Esto permite además que la estructura a grandes rasgos de este bloque sea común a todos los dispositivos y su funcionamiento, el propio de los circuitos digitales. Por ello es posible generalizar la gestión de la entrada/salida, al igual que los comportamientos del procesador y las memorias se adaptan a unas líneas generales bien conocidas. Vamos a ver ahora con detalle la estructura genérica, a nivel lógico o funcional -no a nivel estructural o de componentes físicos- de la parte digital de los dispositivos de entrada/salida. En apartados posteriores describiremos las distintas formas de gestionarla.

La parte de los dispositivos de entrada/salida común a la tecnología electrónica digital del ordenador y que permite relacionarlo con uno o varios periféricos recibe el nombre de *controlador de entrada/salida*. El controlador oculta al procesador las especificidades y la dificultad de tratar con el resto de componentes del periférico y le proporciona una forma de intercambio de información, una interfaz, genérica. Esta generalidad, como se ha dicho, tiene rasgos comunes para todos los tipos de dispositivos pero además tiene rasgos específicos para cada tipo que dependen de sus características. Por ejemplo, un controlador de disco duro se adapta a una especificación común para los controladores -el estándar IDE-ATA, por ejemplo- con independencia de la tecnología de fabricación y aspectos específicos de un modelo de disco en concreto, y estandariza la forma de tratar el disco duro mediante los programas

ejecutados por el procesador.

Para realizar la comunicación entre el procesador y el dispositivo, a través del controlador, existen un conjunto de espacios de almacenamiento, normalmente registros -también conocidos como *puertos*- a los que puede acceder el procesador que se clasifican, según su función, en tres tipos:

- *Registros de control*, que se utilizan para que el procesador configure parámetros en el dispositivo o le indique las operaciones de entrada/salida que debe realizar. Son registros en los que puede escribir el procesador, pero no el dispositivo.
- *Registros de estado*, que permiten al dispositivo mantener información acerca de su estado y del estado de las operaciones de entrada/salida que va realizando. Son registros que escribe el dispositivo y puede leer el procesador.
- *Registros de datos*, que sirven para realizar el intercambio de datos entre el procesador y el dispositivo en las operaciones de entrada/-salida. En el caso de salida, el procesador escribirá los datos que el periférico se encargará de llevar al mundo exterior. En el caso de entrada, el periférico escribirá los datos en estos registros, que de este modo serán accesibles para el procesador mediante lecturas.

Veamos un ejemplo de uso de estos registros a la hora de que el procesador se comunique con una impresora, a través de su controlador, para imprimir cierto documento. Aunque en realidad las cosas no sucedan exactamente de esta manera, por la estandarización de los formatos de documentos y gestión de impresoras, el ejemplo es suficientemente ilustrativo y válido. En primer lugar, el procesador configuraría en la impresora, a través de registros de control, el tamaño de papel, la resolución de la impresión y el uso o no de colores. Una vez realizada la configuración, el procesador iría enviando los datos a imprimir a través de los registros de datos, y al mismo tiempo estaría consultando los registros de estado, ya sea para detectar posibles errores -falta de papel o de tinta, atascos de papel-, ya sea para saber cuándo la impresora no acepta más datos -recordemos que el procesador es mucho más rápido- o ha terminado de imprimir la página en curso. Al acabar todas las páginas del documento, el procesador posiblemente avisaría a la impresora de tal circunstancia, mediante un registro de control, y aquélla podría pasar a un modo de espera con menor consumo.

Aunque la clasificación de los registros y sus características, tal y como se han presentado, son correctas desde un punto de vista teórico, es frecuente que en los controladores reales, para simplificar los circuitos y la gestión, se mezcle información de control y estado en un mismo

registro lógico -es decir, un único registro desde el punto de vista del procesador- e incluso que un bit tenga doble uso, de control y estado, según el momento. Un ejemplo común en los conversores analógico-digitales es disponer de un bit de control que escribe el procesador para iniciar la conversión -poniéndolo a `1`, por ejemplo- y que el dispositivo cambia de valor -a `0` en este ejemplo- cuando ha terminado -típica información de estado- y el resultado está disponible en un registro de datos.

Como se ha visto, cuando el procesador quiere realizar una determinada operación de entrada/salida debe leer o escribir en los registros del controlador. Por lo tanto, estos registros deben ser accesibles por el procesador a través de su conjunto de instrucciones. Este acceso puede realizarse de dos formas:

- Los registros de entrada/salida pueden formar parte del espacio de direcciones de memoria del ordenador. En este caso se dice que el sistema de entrada/salida está *mapeado en memoria*. El procesador lee y escribe en los registros de los controladores de la misma forma y mediante las mismas instrucciones con que lo hace de la memoria. Este esquema es el utilizado por la arquitectura ARM.

- Los registros de entrada/salida se ubican en un mapa de direcciones propio, independiente del mapa de memoria del sistema. En este caso se dice que el mapa de entrada salida es *independiente* o *aislado*. El procesador debe disponer de instrucciones especiales para acceder a los registros de entrada/salida. La ejecución de estas instrucciones se refleja en los circuitos externos del procesador, lo que permite al sistema distinguir estos accesos de los accesos a memoria y usar por tanto mapas distintos. Esta modalidad es utilizada por la arquitectura Intel de 32 y 64 bits, con instrucciones específicas tipo *in* y *out*.

Es necesario indicar que un procesador que dispone de instrucciones especiales de entrada/salida puede sin embargo utilizar un esquema mapeado en memoria, e incluso ambos. No es de extrañar por ello que el mapa del bus PCI Express y los dispositivos en un ordenador tipo PC incluyan regiones en memoria y otras en mapa específico de entrada/salida.

En este apartado hemos visto la estructura de los dispositivos de entrada/salida, que normalmente incluye un bloque de tecnología específica para interactuar con el mundo exterior, otro electrónico analógico que se relaciona con el anterior mediante transductores, y un bloque de electrónica digital, de la misma naturaleza que el resto de circuitos del

ordenador. Este bloque se llama *controlador del dispositivo* y facilita que el procesador se comunique con aquél mediante registros de control para enviar órdenes y configuraciones, registros de estado para comprobar el resultado de las operaciones y los posibles errores, y registros de datos para intercambiar información. Estos registros pueden ser accesibles en el mapa de memoria del procesador, mediante instrucciones de acceso a memoria, o en un mapa específico de entrada/salida que solo puede darse si el procesador incorpora instrucciones especiales. Veamos ahora cómo todos estos registros se usan para relacionar el procesador con los dispositivos.

7.3. Gestión de la entrada/salida

Aunque como hemos visto existen dispositivos con tasas de transferencia muy distintas, en general los periféricos son mucho más lentos que el procesador. Si un ordenador está ejecutando un solo programa y el flujo de ejecución depende de las operaciones de entrada/salida, esto no supondría un gran problema. El procesador puede esperar a que se vayan produciendo cambios en los dispositivos que se relacionan con el exterior, dado que su función consiste en ello. No es éste, sin embargo, el caso general. En el ejemplo que hemos utilizado para mostrar el uso de los distintos registros de los dispositivos, no parece razonable que el ordenador quede bloqueado esperando respuestas -señales de que el trabajo en curso ha terminado o indicaciones de error- de la impresora. Estamos más bien acostumbrados a seguir realizando cualquier otra actividad con el ordenador mientras la impresora va terminando hoja tras hoja, además sin percibir apenas disminución en el rendimiento del sistema.

Así pues, el aspecto fundamental de la gestión de la entrada/salida, que intenta en lo posible evitar que el procesador preste atención al dispositivo mientras no sea necesario, es la *sincronización*. Se pretende que el procesador y los dispositivos se sincronicen de tal modo que aquél solo les preste atención cuando hay alguna actividad que realizar -recoger datos si ya se han obtenido, enviar nuevos datos si se han consumido los anteriores, solucionar algún error o avisar al usuario de ello-. Sabemos que los registros de estado del controlador del dispositivo sirven para este fin, indicar que se ha producido alguna circunstancia que posiblemente requiere de atención. Por lo tanto, la forma más sencilla de sincronización con el dispositivo, llamada *prueba de estado*, *consulta de estado* o *encuesta* -en inglés, *polling*- consiste en que el procesador, durante la ejecución del programa en curso, lea de cuando en cuando los registros de estado necesarios y, si advierte que el dispositivo requiere atención, pase a ejecutar el código necesario para prestarla, posiblemente

contenido en una subrutina de gestión del dispositivo.

El código que aparece a continuación podría ser un ejemplo de consulta de estado.

ej-consulta-estado.s

```
        ldr     r7, =ST_IMPR    @ Dirección del registro de estado
        ldr     r6, =0x00000340 @ Máscara para diversos bits
        ldr     r0, [r7]        @ Leemos el puerto
        ands    r0, r6          @ y verificamos los bits
        beq     sigue           @ Seguimos si no hay avisos
        bl      TRAT_IMPR       @ Llamamos a la subrutina
sigue:
        ...                     @ Continuamos sin prestar atención
```

En este ejemplo se consulta el registro de estado de una impresora y, si alguno de los bits 6, 8 o 9 está a `1`, saltamos a una subrutina de tratamiento para gestionar tal circunstancia. Posiblemente dicha rutina vería cuáles de los tres bits están activos en el registro de estado, y emprendería las acciones correspondientes para gestionar esa circunstancia. Si ninguno de los bits está a `1`, el procesador ignora la impresora y continúa con su programa.

Esta forma de gestionar la entrada/salida es muy sencilla, no requiere mayor complejidad al procesador y puede usarse en todos los sistemas. En muchos de ellos, si están dirigidos por eventos de entrada/salida -es decir, el flujo de ejecución del programa se rige por acciones de entrada/salida y no por condiciones de datos- como ocurre en la mayor parte de los *sistemas empotrados*, es la forma más adecuada de sincronizarse con la entrada/salida.

Sin embargo, para otros casos, sobre todo en los *sistemas de propósito general*, esta forma de gestión presenta serios inconvenientes. Por una parte, el procesador debe incluir instrucciones para verificar cada cierto tiempo el estado del dispositivo. Esta verificación consume tiempo inútilmente si el dispositivo no requiere atención. En un sistema con decenas de dispositivos la cantidad de tiempo perdida podría ser excesiva. Por otra parte, al consultar el estado cada cierto tiempo, la latencia se incrementa y se hace muy variable. Si un dispositivo activa un bit de estado justo antes de que el procesador lea el registro, la latencia será mínima. Sin embargo, si el bit se activa una vez se ha leído el registro, este cambio no será detectado por el procesador hasta que vuelva a realizar una consulta. De esta manera, si se desea una baja latencia se ha de consultar a menudo el registro, lo que consumirá tiempo de forma inútil.

A la vista de los problemas descritos, sería bueno que fuera el propio dispositivo el que avisara al procesador en caso de necesitar su atención, sin que éste tuviera que hacer nada de forma activa. Dado que el pro-

cesador es el encargado de gestionar todo el sistema, en último término sería quien podría decidir qué dispositivos tienen permiso para avisarle y si hacer o no caso a los avisos una vez recibidos. Estas ideas se recogen en el mecanismo de gestión de la entrada/salida mediante interrupciones.

Según esta idea, el mecanismo de gestión de entrada/salida mediante interrupciones permite que cuando un dispositivo, con permisos para ello, activa un aviso en sus registros de estado, esto provoque una señal eléctrica que fuerza al procesador, al terminar de ejecutar la instrucción en curso, a saltar automáticamente al código que permite gestionar el dispositivo. Cuando se completa este tratamiento, el procesador continúa ejecutando la instrucción siguiente a la que estaba ejecutando cuando llegó la interrupción, como si hubiera retornado de una subrutina -pero con más implicaciones que estudiaremos a continuación-. El símil más usado es el de la llamada telefónica, que llega mientras estamos leyendo tranquilamente un libro. Al sonar el teléfono -señal de interrupción- dejamos una marca en la página que estamos leyendo, y vamos a atenderla. Al terminar, continuamos con la lectura donde la habíamos dejado. De esta manera los dispositivos son atendidos con muy poca latencia y los programas de aplicación no necesitan incluir instrucciones de consulta ni preocuparse de la gestión de la entrada/salida.

De las explicaciones anteriores se deduce que el mecanismo de interrupciones se sustenta mediante el hardware del procesador y de la entrada/salida. Efectivamente, no todos los procesadores están diseñados para poder gestionar interrupciones, aunque en realidad hoy en día solo los microcontroladores de muy bajo coste no lo permiten. Veamos los elementos y procedimientos que necesitan incluir el procesador y los dispositivos para que se pueda gestionar la entrada/salida mediante interrupciones:

- El aviso le llega al procesador, como hemos dicho, mediante una señal eléctrica. Esto requiere que el procesador -o su núcleo, en los procesadores y microcontroladores con dispositivos integrados- disponga de una o varias líneas -pines o contactos eléctricos- para recibir interrupciones. Estas líneas se suelen denominar *líneas de interrupción* o *IRQn* -de *Interrupt Request*, en inglés- donde la *n* indica el número en caso de haber varias. Los controladores de los dispositivos capaces de generar interrupciones han de poder a su vez generar estas señales eléctricas, por lo que también disponen de una -o varias en algunos casos- señales eléctricas de salida para enviarlas al procesador.

- El procesador, guiado por el código con el que ha sido programado, es el gestor de todo el sistema. Así pues, debe poder seleccionar qué dispositivos tienen permiso para interrumpirlo y cuáles no. Es-

to se consigue mediante los bits de *habilitación de interrupciones*. Normalmente residen en registros de control de los controladores de dispositivos, que tendrán uno o más según los tipos de interrupciones que puedan generar. Además el procesador dispone de uno o varios bits de control propios para deshabilitar totalmente las interrupciones, o hacerlo por grupos, según prioridades, etcétera. Más adelante explicaremos este aspecto con más detalle.

- La arquitectura de un procesador especifica cómo debe responder a la señalización de una interrupción habilitada. La organización del procesador y sus circuitos deben permitir que, al acabar la ejecución de una instrucción, se verifique si hay alguna interrupción pendiente y, en caso afirmativo, se cargue en el contador de programa la dirección de la primera instrucción del código que debe atenderla, lo que se conoce como *rutina de tratamiento* o *rutina de servicio de la interrupción*, *RTI*. El procesador suele realizar más acciones en respuesta, como el cambio de estado a modo privilegiado o supervisor, el uso de otra pila u otro conjunto de registros, la deshabilitación automática de las interrupciones, etcétera. Todos estos cambios deben deshacerse al volver, para recuperar el estado en que se encontraba el procesador al producirse la interrupción y poder continuar con el código de aplicación. Más adelante se analizará con más detalle el comportamiento del procesador para tratar una interrupción.

- El último mecanismo que debe proveer el hardware del procesador, según lo especificado en su arquitectura, tiene que ver con la obtención de la dirección de inicio de la RTI, la dirección a la que saltar cuando se recibe una interrupción. Esta dirección, que puede ser única o dependiente de la interrupción en particular, recibe el nombre de *vector de interrupción*. En el caso más sencillo hay una única línea IRQ y un solo vector de interrupción. La RTI debe entonces consultar todos los dispositivos habilitados para determinar cuál o cuáles de ellos activaron sus bits de estado y deben por tanto ser atendidos. En los casos más complejos existen distintas interrupciones, identificadas con un número de interrupción diferente y asociadas a un vector diferente. Para ello, el procesador debe tener diversas líneas de interrupción independientes o implementar un protocolo especial de interrupción en que, además de una señal eléctrica, el dispositivo indica al procesador el número de interrupción. En este caso, cada causa de interrupción puede tener su propia RTI, o bien unos pocos dispositivos se agrupan en la misma, haciendo más rápida la consulta por parte de la RTI.

El mecanismo de interrupciones ha demostrado ser tan eficaz que se

ha generalizado más allá de la entrada/salida en lo que se llama *excepciones -exceptions* o *traps* en inglés-. Una excepción sirve para señalar cualquier circunstancia fuera de lo habitual -por lo tanto, excepcional- durante el funcionamiento de un procesador en su relación con el resto de componentes del ordenador. En este marco más amplio, las interrupciones son excepciones generadas por los dispositivos de entrada/salida. Otras excepciones se utilizan para señalar errores -accesos a memoria inválidos, violaciones de privilegio, división por cero, etcétera- que en muchos casos pueden ser tratados por el sistema y dar lugar a extensiones útiles. Por ejemplo, el uso del disco duro como extensión de la memoria principal se implementa mediante la excepción de fallo de página; las excepciones de coprocesador no presente permiten incorporar emuladores al sistema, como la unidad en coma flotante emulada en los antiguos PC. Entre las excepciones se incluyen también las generadas voluntariamente por software mediante instrucciones específicas o registros a tal efecto de la arquitectura, que al provocar un cambio al modo de ejecución privilegiado, permiten implementar las llamadas al sistema como se verá al estudiar sistemas operativos.

Así pues, las interrupciones son, en la mayoría de los procesadores, un tipo de excepciones asociadas a los dispositivos de entrada/salida y su gestión. Una diferencia fundamental con el resto de excepciones radica en el hecho de que las interrupciones no se generan con la ejecución de ninguna instrucción -un acceso a memoria incorrecto se genera ejecutando una instrucción de acceso a memoria; una división por cero al ejecutar una instrucción de división- por lo que son totalmente asíncronas con la ejecución de los programas y no tienen ninguna relación temporal con ellos que pueda ser conocida a priori.

Vamos a describir con más detalle todos los conceptos sobre interrupciones expuestos más arriba. Para ello recorreremos las sucesivas fases relacionadas con las interrupciones en la implementación y ejecución de un sistema, comentando exclusivamente los aspectos pertinentes a este tema.

El uso de interrupciones para la gestión de la entrada/salida se debe tener en cuenta desde el diseño del hardware del sistema. El procesador seleccionado deberá ser capaz de gestionarlas y contar, por tanto, con una o varias líneas externas de interrupción. Es posible además que utilice interrupciones vectorizadas y el número de interrupción -asociado al vector- se entregue al señalarla mediante cierto protocolo de comunicación específico. En este caso habrá que añadir al sistema algún controlador de interrupciones. Estos dispositivos suelen ampliar el número de líneas de interrupción del sistema para conectar diversos dispositivos y se encargan de propagar al procesador las señales en tales líneas, enviando además el número de interrupción correspondiente. Desde el punto de vista del procesador, el controlador de interrupciones funciona como un

sencillo dispositivo de entrada/salida. Ejemplos de estos dispositivos son el *NVIC* utilizado en la arquitectura ARM y que se verá más adelante o el *8259A* asociado a los procesadores Intel x86. En estos controladores se puede programar el número de interrupción asociado a cada línea física, la prioridad, las máscaras de habilitación y algún otro aspecto relacionado con el comportamiento eléctrico de las interrupciones.

Es interesante comentar que una interrupción puede señalarse eléctricamente mediante un flanco o mediante nivel. En el primer caso, una transición de estado bajo a alto o viceversa en la señal eléctrica produce que se detecte la activación de una interrupción. En el segundo caso, es el propio nivel lógico presente en la línea, `1` o `0`, el que provoca la detección. Existe una diferencia fundamental en el comportamiento de ambas opciones, que tiene repercusión en la forma de tratar las interrupciones. Un flanco es un evento único y discreto, no tiene duración real, mientras que un nivel eléctrico se puede mantener todo el tiempo que sea necesario. Así pues, un dispositivo que genera un flanco, ha mandado un aviso al procesador y posiblemente no esté en disposición de mandar otro mientras no sea atendido. Un dispositivo que genera una interrupción por nivel, continúa generándola hasta que no sea atendido y se elimine la causa que provocó la interrupción. Así, una rutina de tratamiento de interrupciones debe verificar que atiende todas las interrupciones pendientes, tratando adecuadamente los dispositivos que las señalaron. De esta manera se evita perder las que se señalan por flanco y se dejan de señalar las que lo hacen por nivel. Más adelante, al comentar las RTI, volveremos a tratar esta circunstancia. Los dispositivos capaces de generar interrupciones deberán tener sus líneas de salida de interrupción conectadas a la entrada correspondiente del procesador o del controlador de interrupciones, y por supuesto, tener sus registros accesibles en el mapa de memoria o de entrada/salida mediante la lógica de decodificación del sistema.

Una vez diseñado el hardware hay que programar el código de las diversas RTI y configurar los vectores de interrupción para que se produzcan las llamadas adecuadamente. Las direcciones de los vectores suelen estar fijas en el mapa de memoria del procesador, es decir en su arquitectura, de forma que a cada número de interrupción le corresponde una dirección de vector determinada. Algunos sistemas permiten configurar el origen de la tabla de vectores de interrupción, entonces la afirmación expuesta antes hace referencia, no al valor absoluto del vector, sino a su desplazamiento con respecto al origen. Sea como sea, los vectores de interrupción suelen reservar una zona de memoria de tamaño fijo -y reducido- en la que no hay espacio para el código de las RTI. Para poder ubicar entonces el código de tratamiento con libertad en la zona de memoria que se decida al diseñar el software del sistema, y con espacio suficiente, se utilizan dos técnicas. La más sencilla consiste en

dejar entre cada dos vectores de interrupción el espacio suficiente para una instrucción de salto absoluto. Así pues, al llegar la interrupción se cargará el PC con la dirección del vector y se ejecutará la instrucción de salto que nos llevará efectivamente al código de la RTI en la dirección deseada. La segunda opción requiere más complejidad para el hardware del procesador. En este caso, en la dirección del vector se guarda la dirección de inicio de la RTI, usando una especie de direccionamiento indirecto. Cuando se produce la interrupción, lo que copiamos en el PC no es la dirección del vector, sino la dirección contenida en el vector, lo que otorga al sistema flexibilidad para ubicar las RTI libremente en memoria. Esta última opción es la utilizada en las arquitecturas ARM e Intel de 32 y 64 bits.

Las dos etapas explicadas tienen lugar durante el diseño del sistema y están ya realizadas antes de que éste funcione. El hardware del sistema y el código en ROM ya están dispuestos cuando ponemos el ordenador en funcionamiento. La siguiente etapa es la configuración del sistema, que se realiza una vez al arrancar, antes de que comience el funcionamiento normal de las aplicaciones. Esta fase es sencilla y conlleva únicamente la asignación de prioridades a las interrupciones y la habilitación de aquéllas que deban ser tratadas. Por supuesto, en sistemas complejos pueden cambiarse de forma dinámica ambas cosas, según las circunstancias del uso o de la ejecución, aunque es normal que se realice al menos una configuración básica inicial. Es conveniente aprovechar esta descripción para comentar algo más sobre la prioridad de las interrupciones. En un sistema pueden existir numerosas causas de interrupción y es normal que algunas requieran un tratamiento mucho más inmediato que otras. En un teléfono móvil inteligente es mucho más prioritario decodificar un paquete de voz incorporado en un mensaje de radiofrecuencia durante una conversación telefónica que responder a un cambio en la orientación del aparato. Como las interrupciones pueden coincidir en lapsos temporales pequeños, es necesario aportar mecanismos que establezcan prioridades entre ellas. De esta manera, si llegan varias interrupciones al mismo tiempo el sistema atenderá exclusivamente a la más prioritaria. Además, durante una RTI es normal que se mantengan deshabilitadas las interrupciones de prioridad inferior a la que se está tratando. Para ello el sistema debe ser capaz de asignar una prioridad a cada interrupción o grupo de ellas; esto se asocia normalmente, igual que el vector, al número de interrupción o a la línea física. Una consecuencia de la priorización de interrupciones es que las rutinas de tratamiento suelen concluir revisando las posibles interrupciones pendientes, que se hayan producido mientras se trataba la primera. De esta manera no se pierden interrupciones por una parte y se evita por otra que nada más volver de una RTI se señale otra que hubiera quedado pendiente, con la consiguiente pérdida de tiempo. Cuando varios dispositivos comparten el

mismo número y vector de interrupción, entonces la priorización entre ellas se realiza en el software de tratamiento, mediante el orden en que verifica los bits de estado de los distintos dispositivos.

Una vez el sistema en funcionamiento, ya configurado, las interrupciones pueden llegar de forma asíncrona con la ejecución de instrucciones. En este caso el procesador, al terminar la instrucción en curso, de alguna de las formas vistas, carga en el contador de programa la dirección de la RTI. Previamente debe haber guardado el valor que contenía el contador de programa para poder retornar a la instrucción siguiente a la que fue interrumpida. Al mismo tiempo se produce un cambio a modo de funcionamiento privilegiado -por supuesto, solo en aquellos procesadores que disponen de varios modos de ejecución- y se deshabilitan las interrupciones. Bajo estas circunstancias comienza la ejecución de la RTI.

Dado que una interrupción puede ocurrir en cualquier momento, es necesario guardar el estado del procesador -registros- que vaya a ser modificado por la RTI, lo que suele hacerse en la pila. Si la rutina habilita las interrupciones para poder dar paso a otras más prioritarias, deberá preservar también la copia del contador de programa guardada por el sistema -que frecuentemente se almacena en un registro especial del sistema-. Posteriormente, la rutina de servicio tratará el dispositivo de la forma adecuada, normalmente intentando invertir el menor tiempo posible. Una vez terminado el tratamiento, la rutina recuperará el valor de los registros modificados y recuperará el contador de programa de la instrucción de retorno mediante una instrucción especial -normalmente de retorno de excepción- que devuelva al procesador al modo de ejecución original.

En algunos sistemas, diseñados para tratar las excepciones de forma especialmente eficiente, todos los cambios de estado y de preservación de los registros comentados se realizan de forma automática por el hardware del sistema, que dispone de un banco de registros de respaldo para no modificar los de usuario durante la RTI. Este es el caso de la arquitectura ARM, en que una rutina de tratamiento de interrupción se comporta a nivel de programa prácticamente igual que una subrutina.

En este apartado hemos visto que la sincronización entre el procesador y los dispositivos es el punto clave de la gestión de la entrada/salida. De forma sencilla, el procesador puede consultar los bits de estado de un dispositivo para ver si necesita atención, con lo que se gestiona mediante consulta de estado o encuesta. Si el procesador incorpora el hardware necesario, puede ser el dispositivo el que le avise de que necesita su atención, mediante interrupciones. En este mecanismo, el procesador debe incorporar más complejidad en sus circuitos, pero los programas pueden diseñarse de forma independiente de la entrada/salida. Para la gestión de excepciones, que generaliza e incluye la de interrupciones, el proce-

sador debe ser capaz de interrumpir la ejecución de la instrucción en curso y de volver a la siguiente una vez terminado el tratamiento; de saber dónde comienza la rutina de servicio RTI mediante vectores; de establecer prioridades entre aquéllas y de preservar el estado para poder continuar la ejecución donde se quedó al llegar la interrupción.

7.4. Transferencia de datos y DMA

La transferencia de datos entre el procesador y los dispositivos se realiza, como se ha visto, a través de los registros de datos. En el caso de un teclado, un ratón u otros dispositivos con una productividad de pocos bytes por segundo, no es ningún problema que el procesador transfiera los datos del dispositivo a la memoria, para procesarlos más adelante. De esta forma se liberan los registros de datos del dispositivo para que pueda registrar nuevas pulsaciones de teclas o movimientos y clicks del ratón por parte del usuario. Esta forma sencilla de movimiento de datos, que se adapta a la estructura de un computador vista hasta ahora, se denomina *transferencia de datos por programa*. En ella el procesador ejecuta instrucciones de lectura del dispositivo y escritura en memoria para ir transfiriendo uno a uno los datos disponibles. Análogamente, si se quisiera enviar datos a un dispositivo, se realizarían lecturas de la memoria y escrituras en sus registros de datos para cada uno de los datos a enviar.

Consideremos ahora que la aplicación que se quiere ejecutar consiste en la reproducción de una película almacenada en el disco duro. En este caso, el procesador debe ir leyendo bloques de datos del disco, decodificándolos de la forma adecuada y enviando por separado -en general- información a la tarjeta gráfica y a la tarjeta de sonido. Todos los bloques tratados son ahora de gran tamaño, y el procesador no debe únicamente copiarlos del dispositivo a la memoria o viceversa, sino también decodificarlos, extrayendo por separado el audio y el vídeo, descomprimiéndolos y enviándolos a los dispositivos adecuados. Y todo ello en tiempo real, generando y enviando no menos de 24 imágenes de unos 6 megabytes y unos 25 kilobytes de audio por segundo. Un procesador actual de potencia media o alta es capaz de realizar estas acciones sin problemas, en el tiempo adecuado, pero buena parte del tiempo lo invertiría en mover datos desde el disco a la memoria y de ésta a los dispositivos de salida, sin poder realizar trabajo más productivo. En un sistema multitarea, con muchas aplicaciones compitiendo por el uso del procesador y posiblemente también leyendo y escribiendo archivos en los discos, parece una pérdida de tiempo de la CPU dedicarla a estas transferencias de bloques de datos. Para solucionar este problema y liberar al procesador de la copia de grandes bloques de datos, se ideó una técnica llamada

acceso directo a memoria, o *DMA -Direct Memory Access* en inglés- en la que un dispositivo especializado del sistema, llamado *controlador de DMA*, se encarga de realizar dichos movimientos de datos.

El controlador de DMA es capaz de copiar datos desde un dispositivo de entrada/salida a la memoria o de la memoria a los dispositivos de entrada/salida. Hoy en día es frecuente que también sean capaces de transferir datos entre distintas zonas de memoria o de unos dispositivos de entrada/salida a otros. Recordemos que el procesador sigue siendo el gestor de todo el sistema, en particular de los buses a través de los cuales se transfieren datos entre los distintos componentes: memoria y entrada/salida. Por ello, los controladores de DMA deben ser capaces de actuar como maestros de los buses necesarios, para solicitar su uso y participar activamente en los procesos de arbitraje necesarios. Por otra parte, para que el uso del DMA sea realmente eficaz, el procesador debe tener acceso a los recursos necesarios para poder seguir ejecutando instrucciones y no quedarse en espera al no poder acceder a los buses. Aunque no vamos a entrar en detalle, existen técnicas tradicionales como el *robo de ciclos*, que permiten compaginar sin demasiada sobrecarga los accesos al bus del procesador y del controlador de DMA. Hoy en día, sin embargo, la presencia de memorias caché y el uso de sistemas complejos de buses independientes, hace que el procesador y los controladores de DMA puedan realizar accesos simultáneos a los recursos sin demasiados conflictos utilizando técnicas más directas de acceso a los buses.

Desde el punto de vista del procesador, el controlador de DMA es un dispositivo más de entrada/salida. Cada controlador dispone de varios canales de DMA, que se encargan de realizar copias simultáneas entre parejas de dispositivos. Cada canal se describe mediante un conjunto de registros de control que indican los dispositivos de origen y de destino, las direcciones de inicio de los bloques de datos correspondientes y la cantidad de datos a copiar. Tras la realización de las copias de datos el controlador indica el resultado -erróneo o correcto- mediante registros de estado, siendo también capaz de generar interrupciones.

Los controladores de DMA actuales son muy potentes y versátiles. Es normal que puedan comunicarse mediante los protocolos adecuados con ciertos dispositivos de entrada/salida para esperar a que haya datos disponibles y copiarlos luego en memoria hasta llenar el número de datos programado. Esto por ejemplo liberaría al procesador de tener que leer los datos uno a uno de un conversor analógico-digital, ahorrándose los tiempos de espera inherentes a la conversión. También es frecuente que puedan encadenar múltiples transferencias separadas en una sola orden del procesador, mediante estructuras de datos enlazadas que residen en memoria. Cada una de estas estructuras contiene la dirección de origen, de destino, el tamaño a copiar y la dirección de la siguiente estructura. De esta manera se pueden leer datos de diversos buffers de un disposi-

tivo y copiarlos en otros tantos pertenecientes a distintas aplicaciones, respondiendo así a una serie de llamadas al sistema desde diferentes clientes.

En este apartado se ha descrito la *transferencia de datos por programa*, que es la forma más sencilla de copiar datos entre los dispositivos de entrada/salida y la memoria, ejecutando las correspondientes instrucciones de lectura y escritura por parte del procesador. Se ha descrito el *acceso directo a memoria* como otra forma de transferir datos que libera de esa tarea al procesador, y se han comentado las características de los dispositivos necesarios para tal técnica, los controladores de DMA.

Bibliografía

[Adv95] Advanced RISC Machines Ltd (ARM) (1995). *ARM 7TDMI Data Sheet*.
URL http://www.ndsretro.com/download/ARM7TDMI.pdf

[Atm11] Atmel Corporation (2011). *ATmega 128: 8-bit Atmel Microcontroller with 128 Kbytes in-System Programmable Flash*.
URL http://www.atmel.com/Images/doc2467.pdf

[Atm12] Atmel Corporation (2012). *AT91SAM ARM-based Flash MCU datasheet*.
URL http://www.atmel.com/Images/doc11057.pdf

[Bar14] S. Barrachina Mir, G. León Navarro y J. V. Martí Avilés (2014). *Conceptos elementales de computadores*.
URL http://lorca.act.uji.es/docs/conceptos_elementales_de_computadores.pdf

[Cle14] A. Clements (2014). *Computer Organization and Architecture. Themes and Variations. International edition.* Editorial Cengage Learning. ISBN 978-1-111-98708-4.

[Shi13] S. Shiva (2013). *Computer Organization, Design, and Architecture, Fifth Edition*. Taylor & Francis. ISBN 9781466585546.
URL http://books.google.es/books?id=m5KlAgAAQBAJ