



**2º de Grado en Ingeniería Informática**  
**Sistemas Operativos**



## TEMA 5 – ENTRADA/SALIDA

### Bibliografía

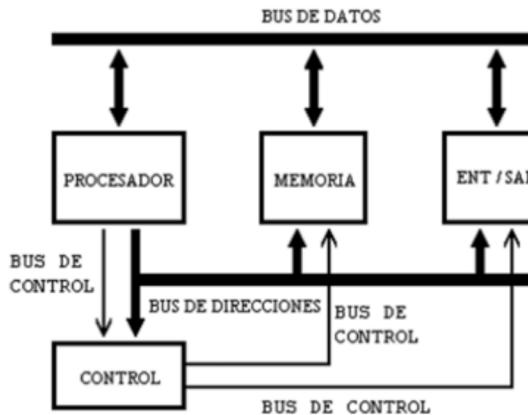
El contenido de este documento se ha elaborado, principalmente, a partir de las siguientes referencias bibliográficas, y con propósito meramente académico y no lucrativo:

- W. Stallings. *Sistemas operativos, 5ª edición*. Prentice Hall, Madrid, 2005.
- A. S. Tanenbaum. *Sistemas operativos modernos, 3a edición*. Prentice Hall, Madrid, 2009.
- A. Silberschatz, G. Gagne, P. B. Galvin. *Fundamentos de sistemas operativos, séptima edición*. McGraw-Hill, 2005.
- A. McIver, I. M. Flynn. *Sistemas operativos, 6ª edición*. Cengage Learning, 2011.
- J. A. Alamansa, M. A. Canto Diaz, J. M. de la Cruz García, S. Dormido Bencomo, C. Mañoso Hierro. *Sistemas operativos, teoría y problemas*. Editorial Sanz y Torres, S.L, 2002.
- F. Pérez, J. Carretero, F. García. *Problemas de sistemas operativos: de la base al diseño, 2ª edición*. McGraw-Hill, 2003.
- S. Candela, C. Rubén, A. Quesada, F. J. Santana, J. M. Santos. *Fundamentos de Sistemas Operativos, teoría y ejercicios resueltos*. Paraninfo, 2005.
- J. Aranda, M. A. Canto, J. M. de la Cruz, S. Dormido, C. Mañoso. *Sistemas Operativos: Teoría y problemas*. Sanz y Torres S.L, 2002.
- J. Carretero, F. García, P. de Miguel, F. Pérez, *Sistemas Operativos: Una visión aplicada*. Mc Graw Hill, 2001.

## 1 Sistema de Entrada/Salida

Los ordenadores actuales se basan en la **arquitectura de Von Neumann**. Los ordenadores con esta arquitectura constan de tres partes claramente diferenciadas: El procesador, la memoria, y uno o varios dispositivos de E/S.

### Máquina de Von Neumann



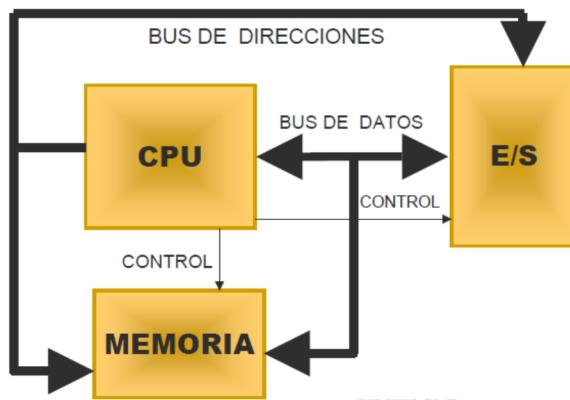
Para conectar estas partes se utilizan buses. **Un bus (o canal)** es un sistema digital que transfiere datos entre los componentes de una computadora y entre ésta y otros dispositivos. Está formado por cables, o por pistas en un circuito impreso o un circuito integrado.

Hay 3 tipos básicos de buses:

- **Bus de datos.** El bus de datos permite el intercambio de datos entre la CPU y el resto de unidades. Es bidireccional, es decir, transfiere tanto las instrucciones que provienen del procesador como las que se dirigen hacia él. Los buses de los ordenadores actuales son de 64 bits.
- **Bus de direcciones.** El bus de direcciones es un canal unidireccional del microprocesador (solo viaja información desde el procesador) totalmente independiente del bus de datos. La memoria RAM es direccionable, de forma que cada celda de memoria tiene su propia dirección. Las direcciones son un número que *selecciona una celda de memoria* dentro de la memoria principal o en el módulo de entrada/salida. Por tanto, el bus de direcciones transporta las direcciones de memoria al que el procesador desea acceder, para leer o escribir datos.

La capacidad de la memoria que se puede direccionar depende de la cantidad de bits que conforman el bus de direcciones, siendo  $2^n$  el tamaño máximo en bits del banco de memoria que se podrá direccionar con  $n$  líneas. Por ejemplo, para direccionar una memoria de 256 bits, son necesarias al menos 8 líneas, pues  $2^8 = 256$ . Adicionalmente pueden ser necesarias líneas de control para señalar cuando la dirección está disponible en el bus. Esto depende del diseño del propio bus.

- **Bus de control.** El bus de control gobierna el uso y acceso a las líneas de datos y de direcciones. Como éstas líneas están compartidas por todos los componentes, tiene que proveerse de determinados mecanismos que controlen su utilización. Las señales de control permiten que no haya colisión de información en el sistema. El bus de control permite también *sincronizar* las actividades y transacciones con los periféricos del sistema a través del módulo de E/S. Se trata por tanto de un bus bidireccional en la medida en que también transmite señales de respuesta del hardware.



Con la arquitectura **Von Neumann** debe haber un sistema que se encargue de gestionar y comunicar a los dispositivos de E/S con el procesador, descargando a la CPU de tanto trabajo como sea posible, hablamos de forma genérica del **sistema, módulo o controlador de E/S** (indistintamente).

El **módulo** de E/S es un circuito o *chip* integrado en la placa base, separado en muchos casos físicamente del procesador y la memoria, pero unido a estos mediante buses en la misma placa. Al conjunto de *chips* integrados en la placa madre y que se encargan de la gestión de los dispositivos de entrada-salida se le conoce como **chipset**. Dicho *chipset* transfiere y controla el flujo de información entre la memoria principal, el procesador y los periféricos.

El SO, normalmente, trata con el módulo de E/S, y no directamente con el dispositivo. Los periféricos se conectan a la placa base a través de una serie de módulos físicos, por ejemplo, en un *slot pci* para una tarjeta de sonido o para extender puertos *usb* en una máquina que no los incluya, o incluso *slots IDE* o *SATA* para discos duros.

De forma resumida, un módulo de E/S tiene las siguientes funciones:

- Envío de comandos a los dispositivos, recibir sus interrupciones y ocuparse de sus errores.
- Ofrecer una interfaz entre los dispositivos y el resto del sistema, incluyendo la CPU.
- Optimizar la E/S del sistema. Los dispositivos de E/S son muy lentos en comparación con la CPU, y si no se delega trabajo en los mismos dispositivos y en el propio modulo, estaríamos haciendo un mal uso del procesador.
- Permitir la conexión de nuevos dispositivos de E/S. Los dispositivos que no necesitan de

driver (se define a continuación) para su funcionamiento, se conocen como **Plug and Play**.

- Almacenamiento temporal de datos (*buffer*). Ya que la velocidad de acceso de la memoria es mucho más alta que la que proporcionan los dispositivos periféricos, el módulo de E/S dispone de una memoria local rápida con la que se comunica con la memoria y la CPU, así puede recibir rápidamente un bloque de datos, liberar el bus, y luego escribirlo en el dispositivo a la velocidad que éste proporcione
- Detección de errores. Debe ocuparse de detectar y comunicar a la CPU los errores mecánicos o eléctricos del dispositivo.

Cada dispositivo de E/S está hecho por un fabricante diferente y puede tener diferentes registros, *chips* físicos y diferente manera de trabajar, en comparación con otro dispositivo destinado al mismo uso pero de otro fabricante distinto. Se necesita entonces un conjunto de instrucciones que conformen un **protocolo de comunicación** entre el computador y el dispositivo físico. Para ello cada fabricante crea un programa software llamado **driver, manejador de dispositivo o controlador** (no confundir con ISR).

Los *drivers* se implementan mediante módulos añadidos al núcleo del sistema operativo, y son objetos software con una interfaz bien definida, que sirven para especificar al sistema operativo y al controlador de E/S cómo debe controlar y comunicarse con un dispositivo en particular. Dicho esto, es necesario que se distinga entre el controlador software y el controlador hardware. Encontrará lecturas bibliográficas en las que se haga referencia a “controlador”, en cuyo caso deberá entender el contexto en el que está leyendo para saber si se refiere al driver o al módulo de E/S.

# Elementos del motherboard

## Dónde se conectan los componentes

Es la placa más grande de una computadora. En ella se conectan todos los componentes, tanto internos como externos. Aquí veremos sus principales elementos.

Zócalo PCI

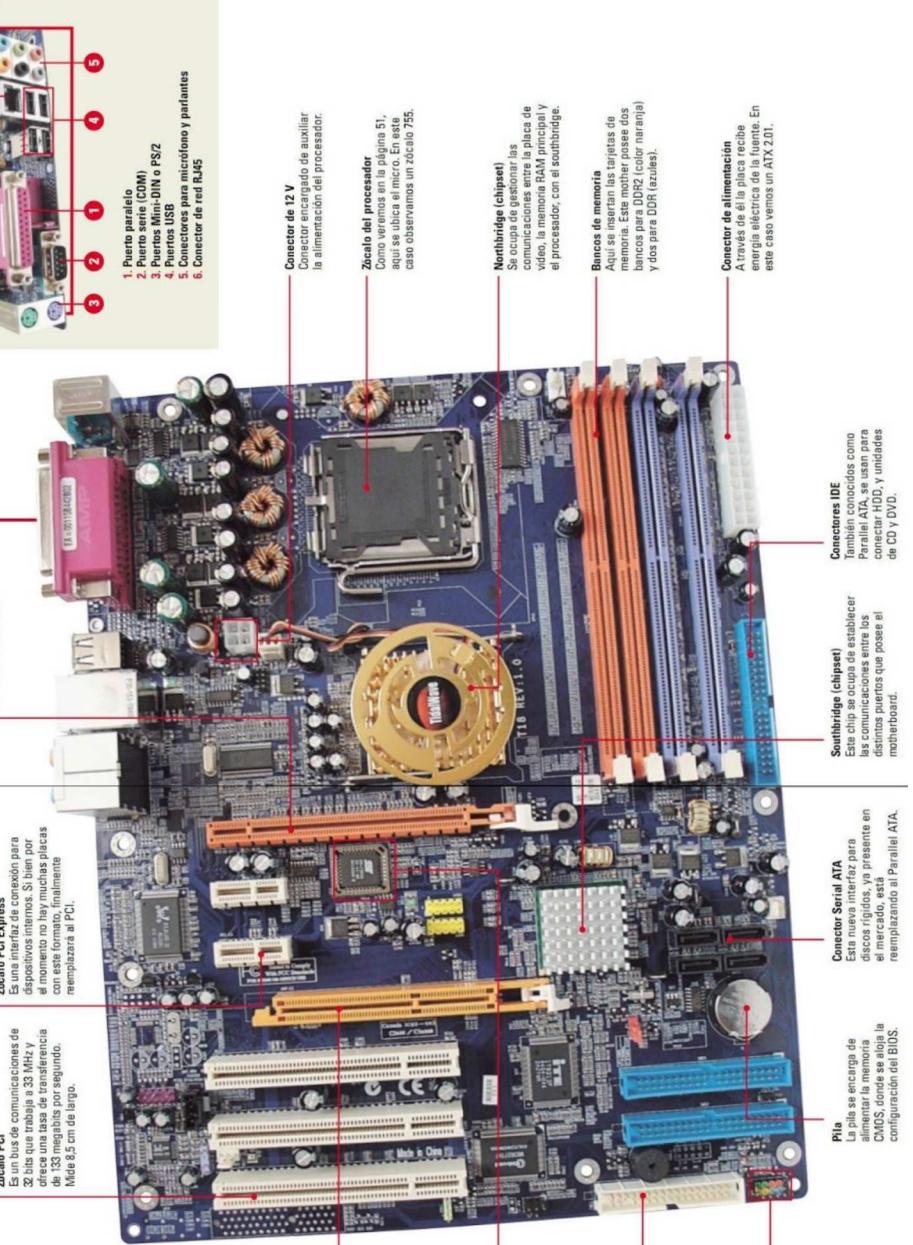
Es un bus de comunicaciones de 32 bits que trabaja a 33 MHz y ofrece una tasa de transmisión de 33 megabits por segundo. Mide 8.5 cm de largo.

Zócalo PCI Express

Es una interfaz de conexión para dispositivos internos. Si bien por el momento no hay muchas placas con este formato, finalmente reemplazará al PCI.

Zócalo PCI Express x16

Pensado para reemplazar al AGP, puede llegar a velocidades de hasta 16 Gbps con 32 bits de ancho.



## 2 Tipos de dispositivos de Entrada/Salida

Los dispositivos de E/S se pueden dividir básicamente en dos categorías<sup>1</sup>, dispositivos de **bloque** y de **caracter**:

- En un **dispositivo de bloque**, discos duros, CD-ROMs, memorias USB, los datos se transfieren en bloques de información indivisibles cuyas características básicas son:
  - Los bloques son de tamaño fijo, varían desde 512 bytes hasta 32.768 bytes, dependiendo del dispositivo y del formato del sistema de ficheros. Todas las transferencias se realizan en unidades de uno o más bloques completos.
  - Un software puede leer o escribir cada bloque de manera independiente de los demás.
  - Cada bloque se referencia o direcciona usando una dirección, que es única para cada uno.
- El otro tipo de dispositivo de E/S es el **dispositivo de carácter** (impresoras, interfaces de red, ratón, monitores, puertos de comunicación) los cuales se comunican con la CPU por medio de bytes individuales:
  - No están sujetos a un estructura de bloques.
  - No se pueden utilizar direcciones al no ser direccionables, no son memorias, sino que tienen un determinado *buffer* intermedio.
  - No se pueden realizar operaciones de búsqueda.
  - Envía o acepta un flujo de caracteres (flujo de bytes).

Dispositivo	Velocidad de transferencia de datos
Teclado	10 bytes/seg
Ratón	100 bytes/seg
Módem de 56K	7 KB/seg
Escáner	400 KB/seg
Cámara de video digital	3.5 MB/seg
802.11g inalámbrico	6.75 MB/seg
CD-ROM de 52X	7.8 MB/seg
Fast Ethernet	12.5 MB/seg
Tarjeta Compact Flash	40 MB/seg
FireWire (IEEE 1394)	50 MB/seg
USB 2.0	60 MB/seg
Red SONET OC-12	78 MB/seg
Disco SCSI Ultra 2	80 MB/seg
Gigabit Ethernet	125 MB/seg
Unidad de disco SATA	300 MB/seg
Cinta de Ultrium	320 MB/seg
Bus PCI	528 MB/seg

**Figura 5-1.** Velocidades de transferencia de datos comunes de algunos dispositivos, redes y buses.

<sup>1</sup>Al final en la red, toda la información viaja bit a bit, pero es el software el que debe ser capaz de ensamblar la información usando los tamaños correctos

### 3 Interrupciones

Asociado e inseparable a las técnicas de E/S, podemos terminar repasando muy brevemente las **interrupciones**.

A nivel de hardware, las interrupciones funcionan de la siguiente manera:

- 1) Cuando un módulo de E/S ha terminado el trabajo que se le asignó con respecto a un dispositivo de E/S, produce una interrupción. Para ello impone una señal en una línea de bus que especifica qué dispositivo desea atención.
- 2) Esta señal es detectada por la CPU, ésta deja lo que está haciendo cuando se termine la instrucción que esté ejecutando (etapa de comprobación de interrupción).
- 3) Un software llamado **controlador de interrupciones** analiza la señal para decidir qué rutina de interrupciones (ISR) se ha de ejecutar de una tabla llamada **vector de interrupciones** (contiene punteros a diferentes tipos de ISR), para obtener así un nuevo contador del programa.

Este contador del programa apunta al inicio del procedimiento ISR correspondiente. La ubicación del vector de interrupción se puede determinar de manera estática (*hardwired*) en la máquina, o puede estar en cualquier parte de la memoria, con un registro de la CPU, cargado por el sistema operativo, apuntando a su origen.

- 4) Si no hay otras interrupciones pendientes, la ISR comienza a ejecutarse (**se obvia el salvado de contexto para una mejor comprensión**). Si hay otra en progreso, o si otro dispositivo ha realizado una petición simultánea en una línea de petición de interrupción de mayor prioridad en el bus, el dispositivo sólo se ignora por el momento. En este caso, continúa imponiendo una señal de interrupción en el bus hasta que la CPU la atiende.
- 5) Poco después de que se empieza a ejecutar la ISR, ésta escribe cierto valor en uno de los puertos de E/S del controlador o modulo de E/S que produjo la interrupción. Este reconocimiento indica al controlador que ya puede emitir otra interrupción y que se está tratando la actual.

### 4 Técnicas de comunicación de E/S

Las operaciones que la CPU puede invocar sobre un módulo o controlador de E/S a través de la ejecución de instrucciones se clasifican en:

- **Operaciones de control.** Utilizadas para activar un dispositivo externo y especificarle qué debe hacer. Por ejemplo, se le puede indicar a una unidad de DVD que se rebobine o avance un registro.
- **Operaciones de estado.** Utilizadas para comprobar diversas condiciones de estado asociadas al controlador: ocupado, listo, no responde, error de comunicación.
- **Operaciones de transferencia.** Utilizadas para leer datos de un dispositivo externo o para

enviar o escribir datos en el mismo.

Hay tres técnicas (ver Tabla 11.1) para llevar a cabo las operaciones de E/S: **E/S programada**, **E/S dirigida por interrupciones** y **acceso directo a memoria** (*Direct Memory Access*, DMA). En la mayoría de los computadores, el DMA es la forma de transferencia predominante a la que el sistema operativo debe dar soporte.

**Tabla 11.1.** Técnicas de E/S.

	Sin interrupciones	Con interrupciones
Transferencia de E/S a memoria a través del procesador	E/S programada	E/S dirigida por interrupciones
Transferencia directa de E/S a memoria		Acceso directo a memoria (DMA)

## 4.1 E/S programada

A continuación se expone cómo se realiza una operación de E/S sin el uso de interrupciones, pero teniendo en cuenta el concepto de módulo o controlador de E/S, el cual actúa como intermediario entre procesador y dispositivo externo.

Cuando el procesador ejecuta un programa y encuentra una instrucción relacionada con la E/S (leer, escribir o simplemente comprobar el estado de un dispositivo), ejecuta esa instrucción generando uno o varios mandatos al módulo o controlador de E/S apropiado.

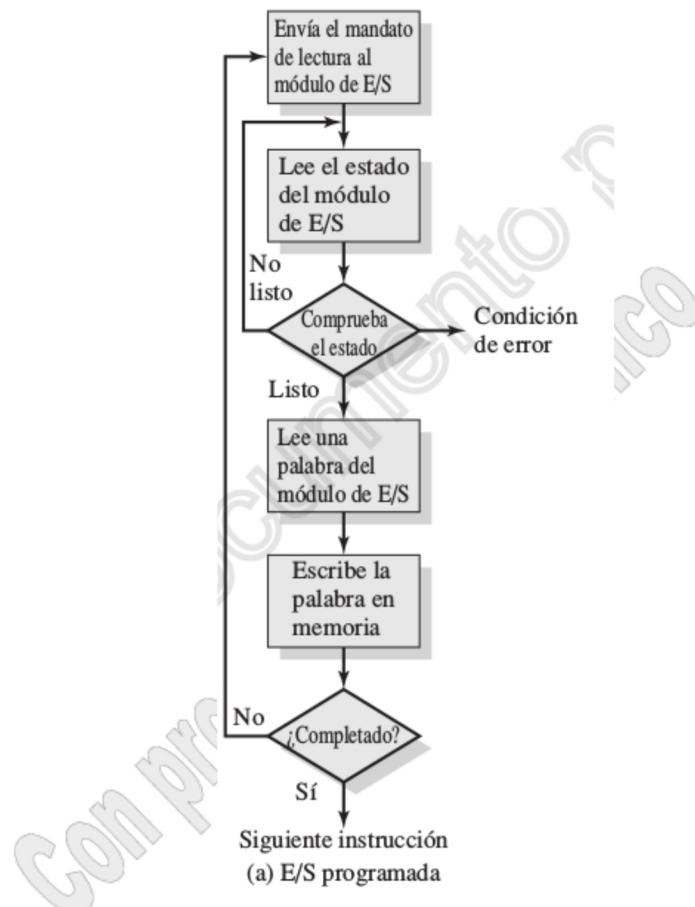
En el caso de la E/S programada, el módulo o controlador de E/S realiza la acción solicitada, pero no realiza ninguna acción para avisar al procesador de su finalización. Por tanto, después de que se invoca la instrucción de E/S, el procesador debe tomar un papel activo para determinar cuándo se completa la instrucción de E/S. Por este motivo, el procesador comprueba periódicamente el estado del módulo de E/S hasta que encuentra que se ha completado la operación.

La Figura 1.19a proporciona un ejemplo del uso de E/S programada para leer un bloque o palabra de datos de un dispositivo externo, por ejemplo de DVD, y almacenarlo en memoria principal. Si se necesitan leer varios bloques se hace uno a uno, por ejemplo, cada 16 bits. Dicha figura se muestra *desde el punto de vista de la CPU*:

- 1) Se envía la petición de lectura al controlador de E/S.
- 2) Se lee el estado del controlador de E/S para determinar si está listo, es decir, si ha terminado de leer la palabra que la CPU ha demandado.
- 3) Si no está listo, la CPU permanece en **espera activa** consumiendo ciclos de reloj, ya que tiene que estar continuamente haciendo comprobaciones de estado.
- 4) Si está listo el controlador de E/S, es decir, si ya se tiene la palabra solicitada, entonces la CPU trae dicha palabra leída desde el controlador, ya que éste ya la recuperó previamente

del DVD.

- 5) A continuación se envía la palabra desde la CPU hasta la memoria principal.
- 6) En caso de que se haya completado toda la operación de E/S se termina la operación.
- 7) En caso contrario, el controlador de E/S sigue trayendo palabras desde el DVD, por lo que la CPU tiene que volver a comprobar el estado del controlador para saber si la siguiente palabra ya está disponible.

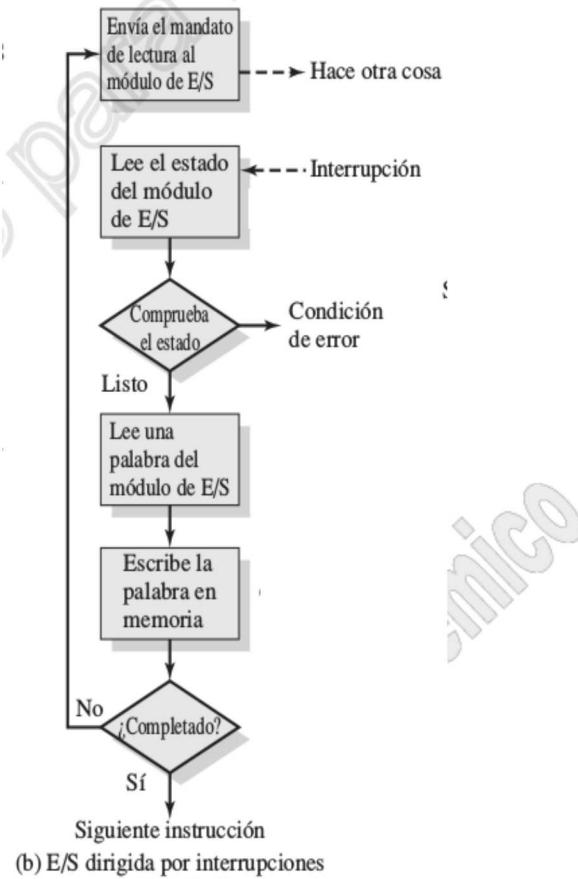


## 4.2 E/S dirigida por interrupciones

El problema de la E/S programada es que el procesador tiene que esperar mucho tiempo haciendo comprobaciones hasta que el módulo de E/S correspondiente esté listo para enviarle la siguiente petición. Como resultado, el nivel de rendimiento de todo el sistema se degrada gravemente.

Una alternativa es que el procesador genere un mandato de E/S y, acto seguido, continúe realizando algún otro trabajo útil mientras que el módulo de E/S realiza su función y se comunica con el dispositivo externo. Aquí entra en juego el concepto de *salvado y restauración de contexto*, ya que hay que sacar de la CPU al proceso actual hasta que el controlador este listo, y cargar un proceso que esté en estado Listo a través del *dispatcher*.

La Figura 1.19b muestra el esquema de una E/S dirigida por interrupciones.



Considere cómo funciona la E/S dirigida por interrupciones desde el **punto de vista del módulo de E/S**:

- 1) Para una operación de entrada, el módulo de E/S recibe un mandato de LECTURA del procesador.
- 2) El módulo de E/S pasa entonces a leer los datos de un periférico asociado.
- 3) Una vez que los datos están en el registro de datos del módulo, el módulo genera una interrupción que va a parar al procesador a través de los buses de control y lógica hardware correspondientes.
- 4) El módulo entonces espera hasta que el procesador pida sus datos, situándolos en el bus de datos.
- 5) Cuando el manejador de interrupciones invoque a la ISR correspondiente, se trate la interrupción y se lo indique al modulo de E/S, éste queda listo para otra operación de E/S.

Desde el **punto de vista del procesador**, las acciones correspondientes a una operación de lectura son:

- 1) El procesador genera un mandato de LECTURA (también puede ser de escritura).
- 2) Se salva el contexto del proceso actual, pasando a restaurar y ejecutar otro proceso que el planificador decida.
- 3) Al final de cada ciclo de instrucción, el procesador comprueba si hay interrupciones.
- 4) Cuando se produce un envío de interrupción por parte del módulo de E/S porque ya está disponible en el la palabra que se solicitadó, la CPU lo detecta y se salva el contexto del proceso actual.
- 5) Acto seguido se ejecuta el proceso manejador de interrupciones y la rutina ISR correspondiente, tramitándose así la interrupción y haciendo que la CPU traiga la palabra desde el controlador o modulo de E/S.

Como inciso, decir que habrá múltiples submódulos de E/S en un computador, por lo que se necesitan mecanismos para permitir que el procesador determine qué dispositivo causó la interrupción y para decidir, en caso de múltiples interrupciones, cuál debe manejar primero. Estos mecanismos los proporciona el software manejador de interrupciones junto con la lógica hardware necesaria para ello.

- 6) A continuación se pasa la palabra de la CPU a memoria principal.

La E/S dirigida por interrupciones es más eficiente que la E/S programada ya que elimina la espera innecesaria. Sin embargo, la E/S dirigida por interrupciones todavía consume mucho tiempo de procesador, puesto que cada palabra de datos que va desde el módulo de E/S hasta la memoria debe pasar a través del procesador.

### 4.3 Acceso directo a memoria (DMA)

La E/S dirigida por interrupciones, aunque más eficiente que la E/S programada, todavía requiere la intervención activa del procesador para transferir datos entre la memoria y un módulo de E/S, ya que cualquier transferencia de datos debe atravesar un camino a través del procesador.

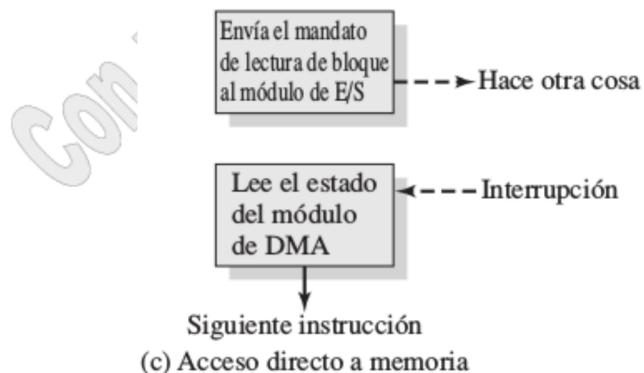
Por tanto, ambas formas de E/S, la programada y por interrupciones, sufren dos inconvenientes inherentes:

1. La tasa de transferencia de E/S está limitada por la velocidad con la que el procesador puede comprobar el estado de un dispositivo y ofrecerle servicio.
2. El procesador está involucrado en la gestión de una transferencia de E/S y se deben ejecutar varias instrucciones por cada transferencia de bloque de E/S.

Cuando se van a transferir grandes volúmenes de datos, por ejemplo grabar un CD-ROM virgen o copiar su contenido al disco duro, se requiere una técnica más eficiente, el acceso directo a memoria (**Direct Memory Access, DMA**). En ocasiones el acceso por DMA puede llevarla a cabo un módulo o chip de E/S separado y conectado en el bus del sistema.

Cuando el procesador desea leer o escribir mediante DMA en un dispositivo externo se producen las siguientes operaciones (la Figura 1.19c muestra un esquema de ello):

- 1) Se genera un mandato al módulo de DMA, enviándole la siguiente información:
  - Si se trata de una lectura o de una escritura.
  - La dirección del periférico de E/S involucrado.
  - La posición inicial de memoria en la que se desea leer los datos o donde se quieren escribir.
  - El número de palabras que se pretende leer o escribir.
- 2) A continuación la CPU continua con otro trabajo, produciéndose un salvado y la restauración de contexto otro proceso que indique el planificador.
- 3) Mientras tanto, como se ha delegado la operación de E/S al módulo de DMA, es éste quien se ocupa de la misma quitándole trabajo a la CPU. El módulo de DMA transferirá los bloques completo de datos, uno a uno, hacia la memoria o desde ella, sin pasar a través del procesador.
- 4) Al final de la transferencia se producirá interrupción por parte del módulo DMA.
- 5) La CPU detectará la interrupción y se realiza el salvado del contexto del proceso actual.
- 6) Acto seguido se ejecuta el proceso manejador de interrupciones, se ejecuta la ISR correspondiente a ese tipo de interrupción y se tramita.
- 7) El trámite simplemente puede ser el establecimiento de un OK que indica que todo ha ocurrido correctamente o incluso de un error. Si había algún proceso en estado bloqueado esperando a que se produjese la trasnferencia por DMA, esté también pasará a estado listo.



Con el mecanismo de transferencia por DMA no se tienen que hacer interrupciones constantemente cuando se acaba de transferir cada bloque. Esto hace que se aproveche aún más la capacidad de procesamiento de la CPU. Así, el procesador solo está involucrado al principio y al final de la transferencia.

## 5 Software de E/S en el espacio de usuario

Aunque la mayor parte del software de E/S está dentro del sistema operativo, una pequeña porción de éste consiste en bibliotecas vinculadas entre sí con programas de usuario. Las llamadas al sistema, incluyendo las llamadas al sistema de E/S, se realizan comúnmente mediante procedimientos de biblioteca.

Un ejemplo de uso de rutinas de biblioteca en C que tienen que ver con la E/S es *printf()*, que toma una cadena de formato y posiblemente unas variables como entrada, construye una cadena ASCII y después invoca a la llamada de sistema *write()* para imprimir la cadena. Como ejemplo de *printf()*, considere la instrucción:

```
printf("El cuadrado de %3d es %6d\n", i, i*i);
```

Esta instrucción da formato a una cadena que consiste en la cadena de 14 caracteres “El cuadrado de” seguida por el valor *i* como una cadena de 3 caracteres, después la cadena de 4 caracteres “es”, luego *i\*i* como 6 caracteres, y por último un salto de línea.

Cuando un programa en C contiene la llamada

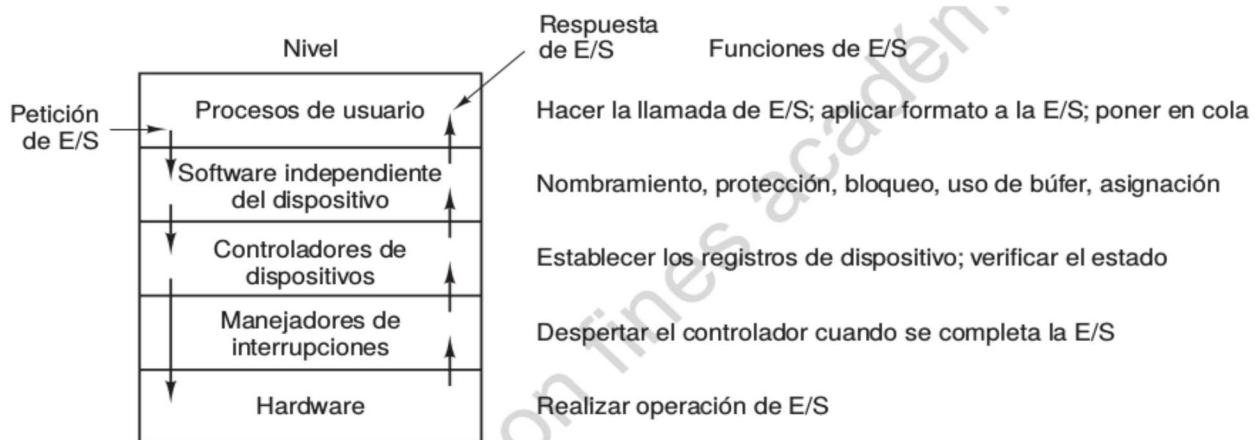
```
cuenta = write(fd, bufer, nbytes);
```

el procedimiento de biblioteca *write()* se vinculará y se incluirá en el programa binario presente en memoria en tiempo de ejecución (cuando compilamos y enlazamos). La colección de todos estos procedimientos de biblioteca es sin duda parte del sistema de E/S.

Ese procedimiento *write()* después podrá hacer uso de funciones asociadas a un *driver* de dispositivo, como se comentó en secciones anteriores. Una de las tareas de un driver es aceptar peticiones abstractas de lectura y escritura del software que está por encima de él (software independiente del dispositivo), y ver que se lleven a cabo correctamente.

En la figura 5-17 se resume el sistema de E/S, donde se muestran todos los niveles y las funciones principales de cada nivel:

- 1) Los procesos de usuario, es decir, el programa que hace la invocación de *printf()*, de *write()*, de *scanf()*, *close()*, etc.
- 2) El software independiente del dispositivo, que serán rutinas abstractas del kernel relacionadas con la operación a realizar. Estas rutinas no indican o no saben con el dispositivo concreto que se tratará o si es de un fabricante u otro.
- 3) El driver del dispositivo, **nombrado en la figura como controlador de dispositivo. En otras lecturas se puede encontrar incluso con el nombre de manejador de dispositivo (OJO, mirar siempre el contexto!!!)**. Este software adapta las rutinas independientes del dispositivo a rutinas concretas del dispositivo con el que se trate.
- 4) El software manejador de interrupciones y encargado de tramitarlas mediante una ISR. Solo entra en ejecución a la vuelta (fíjese en las flechas de la figura).
- 5) El hardware, incluyendo los módulos o controladores de E/S.



**Figura 5-17.** Niveles del sistema de E/S y las funciones principales de cada nivel.