CAPÍTULO 21

DISPOSITIVOS DE ENTRADA/SALIDA

"Cuando el usuario construye un sendero, le da nombre, inserta éste en su libro de códigos y lo teclea en su teclado... Así se cree que los objetos físicos se reúnen de fuentes muy separadas y se enlazan para formar un libro nuevo. Es más que esto, pues cualquier objeto se puede unir en numerosos senderos."
Vannevar Bush, Como podemos pensar, 1945 (donde profetiza lo que ahora es la World Wide Web)

"El valor de un programa es proporcional al peso de su salida."
Sexta ley de la programación de computadoras

TEMAS DEL CAPÍTULO

En este capítulo se revisarán la estructura y principios operativos de algunos dispositivos comunes de entrada/salida, tanto dispositivos que ofrecen o presentan datos como los que capturan datos para archivar o para su posterior procesamiento. Además del tipo de presentación o grabación de datos, los dispositivos I/O se pueden categorizar por sus tasas de datos, desde dispositivos de entrada de datos muy lentos (teclados) hasta subsistemas de almacenamiento de gran ancho de banda. Se verá que la tasa de datos requerida puede dictar la forma en que interactúan el CPU, la memoria y los dispositivos I/O. Cada vez más, la fuente de entrada, o recipiente de salida, de una computadora es otra computadora que se liga a ella a través de una red. Por esta razón, la creación de redes (*networking*) de dispositivos I/O también se discute en este capítulo.

21.1 Dispositivos y controladores de entrada/salida

El procesador y la memoria son mucho más rápidas que la mayoría de los dispositivos I/O. En general, mientras más cerca esté una unida al CPU, más rápido se accede a ella. Por ejemplo, si la memoria principal está a 15 cm de distancia del CPU (figura 3.11*a*), sólo la propagación de señal toma 1 ns en cada vía, pues las señales electrónicas viajan casi a la mitad de la rapidez de la luz; a este retardo de

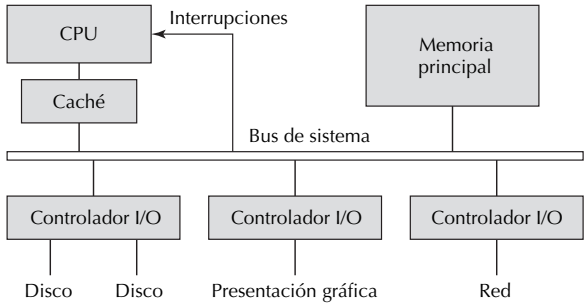


Figura 21.1 Entrada/salida vía un solo bus común.

propagación de señal de viaje redondo de 2 ns se le deben sumar varios retardos lógicos, tiempo de *buffer*, retardo de arbitraje de bus y la latencia de acceso a memoria. Además de los retardos adicionales debidos a distancia creciente desde el CPU, muchos dispositivos I/O también son más lentos según su naturaleza electromecánica.

En la tabla 3.3 se mencionan algunos dispositivos I/O junto con sus tasas de datos aproximadas y dominios de aplicación. Además del tipo de entrada o salida de datos, que se usaron en la tabla 3.3, como el criterio de clasificación primario, los dispositivos I/O se pueden categorizar con base en otras características de tecnología o aplicación. El medio que se usa para presentación de datos (copia impresa contra copia electrónica o blanda), medida del involucramiento humano (manual contra automática) y calidad de producción (bosquejo preliminar contra calidad de publicación o fotografía) son algunas de las características relevantes.

Los modernos dispositivos I/O son muy inteligentes y con frecuencia contienen su propio CPU (usualmente un microcontrolador o procesador de propósito general de bajo perfil), memoria y capacidades de comunicación, que cada vez más incluyen conectividad de red. Las interfases con tales dispositivos, que pueden tener muchos megabytes de memoria para *buffer*, son diferentes del control de los primeros dispositivos I/O, que tenían que alimentarse con información, o dirigirse para enviar datos, un byte a la vez.

Los dispositivos se comunican con el CPU y la memoria mediante un controlador I/O conectados a un bus compartido. Los sistemas simples de bajo perfil pueden usar el mismo bus para I/O que se usa para enviar datos de ida y vuelta entre el CPU y la memoria (figura 21.1). Puesto que el bus común en la figura 21.1 contiene algunas líneas de dirección para muchos accesos a memoria, es sencillo dejar que las mismas líneas seleccionen los dispositivos para operaciones I/O al asignarles una porción del espacio de dirección de memoria. Este tipo de *I/O mapeada por memoria*, y otras estrategias para control de I/O a través de programación, se cubrirán en el capítulo 22. Los sistemas más elaborados o de alto rendimiento pueden tener un bus I/O separado para garantizar que no se quita ancho de banda de las transferencias CPU-memoria (figura 21.2). Cuando existen múltiples buses en un sistema, se ponen en interfaz mutua a través de *adaptadores de bus*. En el capítulo 23 se cubrirán los buses, estándares relevantes y métodos de creación de interfases.

Los controladores I/O en las figuras 21.2 y 21.2 satisfacen las reglas siguientes:

1. Aislar el CPU y a la memoria de detalles de operación del dispositivo I/O y requisitos específicos de interfaz.
2. Facilitar la expansión en capacidades e innovación en tecnología de dispositivos I/O sin impactar el diseño y operación del CPU.
3. Gestionar las (potencialmente amplias) incompatibilidades de rapidez entre procesador/memoria en un lado y los dispositivos I/O en el otro, a través de *buffering*.
4. Convertir los formatos de datos o codificaciones y forzar los protocolos de transferencia de datos.

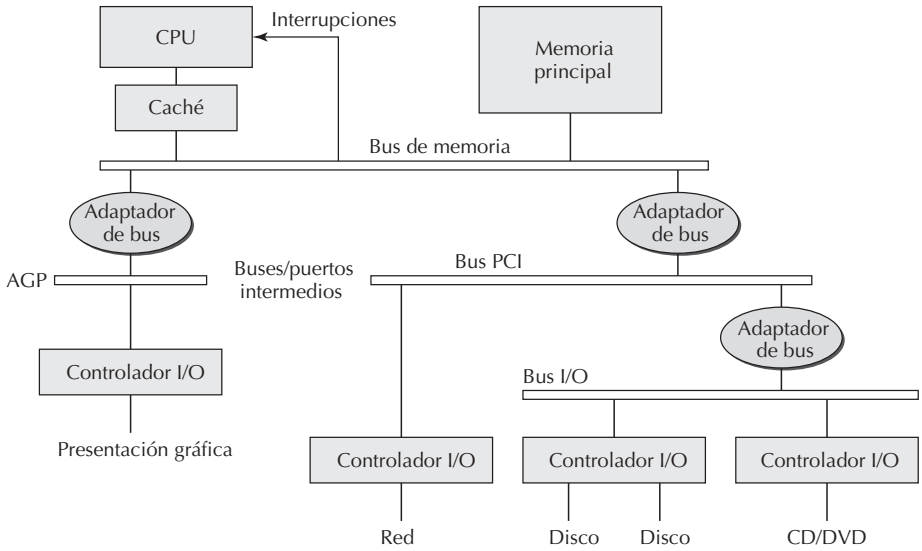


Figura 21.2 Entrada/salida mediante buses I/O intermedio y dedicado (se explicarán en el capítulo 23); el AGP (puerto gráfico acelerado) es un puerto compartido, no un bus.

Los controladores I/O son computadoras por derecho propio. Cuando se activan, corren programas especiales para guiarlos a través de las operaciones requeridas de transferencia de datos y protocolos asociados.

Por esta razón introducen a las operaciones I/O latencias no triviales que se deben considerar en la determinación del rendimiento I/O. Según los tipos de sistema y dispositivo, no son raras latencias de controlador I/O de unos cuantos milisegundos.

21.2 Teclado y ratón

Un *teclado* consiste de un arreglo de teclas que se usa para ingresar información a la computadora u otro dispositivo digital. Los teclados más pequeños, como los que se encuentran en los teléfonos, o en una área separada en muchas computadoras de escritorio, se les conoce como *teclado numérico* (*keypad*). Casi todos los teclados alfanuméricos siguen la plantilla QWERTY, nombre derivado de las etiquetas de las primeras teclas en la hilera superior de letras en el teclado de máquina de escribir estándar. Con los años se han hecho serios intentos por introducir otras plantillas de teclas que harían más fácil alcanzar letras del alfabeto de uso más frecuente, y aumentar la rapidez de entrada de datos. Se dice que la plantilla QWERTY se eligió para reducir deliberadamente la rapidez de tecleo con intención de evitar el atasco de los martillos mecánicos de las primeras máquinas de escribir. Desafortunadamente, la familiaridad con la plantilla QWERTY y la gran experiencia en su uso han evitado la adopción de estas plantillas más sensibles. Sin embargo, se ha progresado algo en el área de colocaciones de teclas no convencionales para dar al usuario comodidad durante el tecleo. Los teclados que siguen tales diseños se denominan *teclados ergonómicos*.

La figuras 21.3 muestra dos diseños específicos para una tecla que es capaz de cerrar un circuito eléctrico al presionar el casquete de la tecla unida a un ensamble de émbolo o a una membrana montada sobre pistones poco profundos. Existen muchos otros diseños en uso. Los interruptores de membrana son baratos, son de uso común en las aplicaciones de sistemas incrustados, como el panel de control

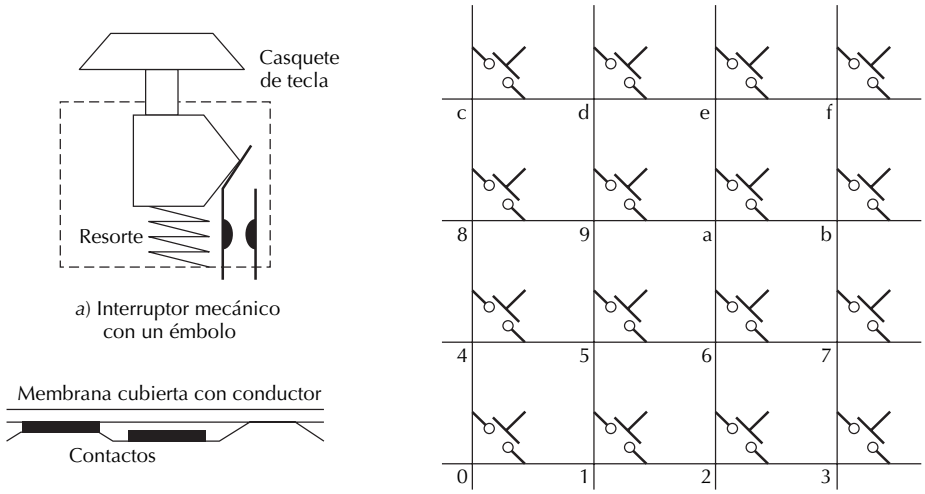
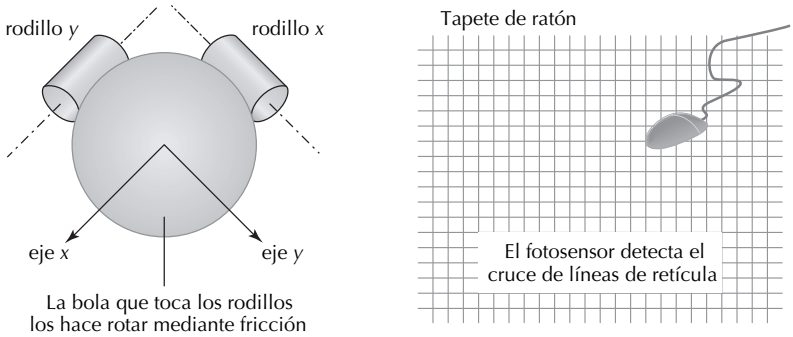


Figura 21.3 Dos diseños de interruptores mecánicos y la plantilla lógica de un teclado numérico hexa.

de un horno de microondas. Los interruptores mecánicos, como el que se muestra en la figura 21.3*a*, se usan en teclados de escritorio estándar en vista de la retroalimentación táctil y su construcción más fuerte. Puesto que los contactos se pueden ensuciar con el paso del tiempo, otros tipos de interruptores se apoyan en la fuerza inducida forma magnética en lugar de mecánicamente para cerrar un par de contactos. De esta forma, los contactos se pueden colocar en un coto sellado para aumentar la confiabilidad. Sin importar qué tipo de contacto se use, el circuito se puede reabrir y volver a cerrar muchas veces con cada presión de tecla. Este efecto, conocido como *rebote de contacto*, se puede evitar al no usar la señal de la tecla directamente sino dejarla que active un *flip-flop*, cuya salida se vuelve alta con una presión de tecla y permanece así sin importar la longitud o severidad del rebote de contacto. También es posible lidiar con el efecto del rebote de contacto con diseño adecuado del software que maneja adquisición de datos del teclado.

Sin importar la plantilla física, las teclas de un teclado o teclado numérico con frecuencias se ordenan en un arreglo cuadrado o rectangular 2D desde un punto de vista lógico. Cada tecla está en la intersección de un circuito de hilera y de columna. Presionar una tecla provoca cambios eléctricos en los circuitos de hilera y columna asociados con la tecla, ello permite la identificación de la tecla que se presionó (figura 21.3*c*). Entonces un codificador convierte las identidades de hilera y columna en código de símbolo único (usualmente ASCII), o secuencia de códigos, para que se transmitan a la computadora. La plantilla física del teclado y el ordenamiento de las teclas no son relevantes para los procesos de detección y codificación.

Con el propósito obtener la salida de cuatro bits que representa la tecla presionada en el teclado numérico hexa de la figura 21.3*c*, las señales de hilera se postulan a su vez, acaso con el uso de un contador de anillo de cuatro bits que sigue la secuencia de conteo 0001, 0010, 0100 y 1000. Conforme una señal se postula en una hilera particular, las señales de columna se observan. La codificación de dos bits del número de hilera unido a la codificación de dos bits del número de columna proporciona la salida de cuatro bits. Un teclado de 64 teclas puede estar ordenado lógicamente en un arreglo 8×8 (aun cuando el ordenamiento físico de las teclas pueda no ser cuadrado). Entonces se produce el código de salida de seis bits en forma similar al procedimiento para el teclado hexa. Si se desea la representación ASCII del símbolo, se puede usar una tabla de consulta con 64 entradas.



a) Ratón mecánico

b) Ratón óptico

Figura 21.4 Ratones mecánico y óptico simple.

Además de un teclado, la mayoría de las computadoras de escritorio y laptop tiene dispositivos apuntadores que permitirán al usuario elegir opciones de menús, seleccionar texto y objetos gráficos para edición, y realizar una diversidad de otras operaciones. De hecho, algunos dispositivos sin teclado se apoyan exclusivamente en el señalamiento para funciones de control y captura de datos. El dispositivo apuntador usado más comúnmente es el *ratón*, llamado así por su forma en algunos de los primeros diseños y el alambre con forma de cola que lo conecta a la computadora. Los ratones modernos vienen en varias formas, muchos de ellos usan enlaces inalámbricos con la computadora.

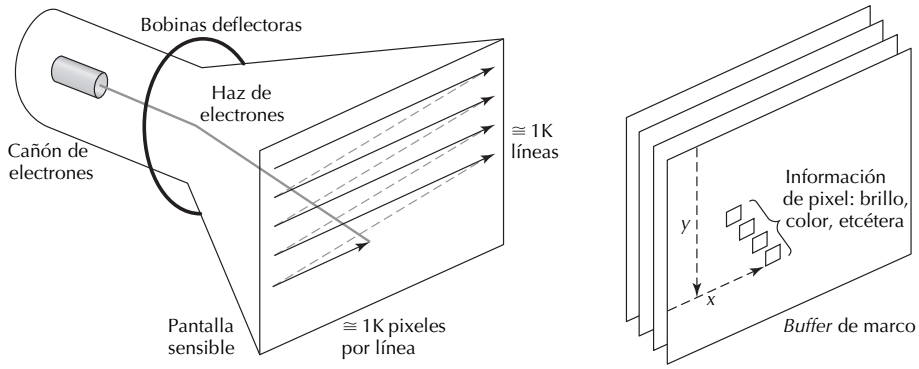
En un ratón mecánico, dos contadores se asocian con los rodillos *x* y *y* (figura 21.4*a*). Levantar el ratón establece los contadores a cero. Cuando el ratón se arrastra, los contadores aumentan o disminuyen de acuerdo con la dirección de movimiento. La computadora usa los valores del contador para determinar la dirección y extensión del movimiento que se aplicará al cursor. Los ratones ópticos son más precisos y menos proclives a fallo debido a la reunión de polvo e hilachos en sus partes. Los ratones ópticos más simples detectan líneas de retícula en un tapete de ratón especial (figura 21.4*b*). Las versiones nuevas, más avanzadas, usan pequeñas cámaras digitales que les permiten detectar movimiento sobre cualquier superficie.

Con frecuencia, un *touchpad* (almohadilla táctil) sustituye un ratón para aplicaciones en las laptop. La ubicación de un dedo que toca el tapete se percibe (a través de circuitos de hilera y columna) y el movimiento del cursor se determina según cuán lejos, rápido y en qué dirección se movió el dedo. Una *pantalla táctil* es similar, pero, en lugar de un tapete separado, la superficie de la pantalla de presentación se usa para apuntar. Otros dispositivos apuntadores incluyen la *trackball* (esfera móvil, que en esencia representa un ratón mecánico boca abajo cuya gran bola se manipula a mano) y el *joystick* (palanca de control) que encuentra aplicaciones en muchos juegos de computadora, así como en escenarios de control industrial. Algunas computadoras laptop usan un pequeño *joystick* incrustado en el teclado para apuntar.

En la actualidad los teclados y dispositivos apuntadores son muy inteligentes y con frecuencia tienen procesadores dedicados (microcontroladores) interconstruidos.

21.3 Unidades de presentación visual

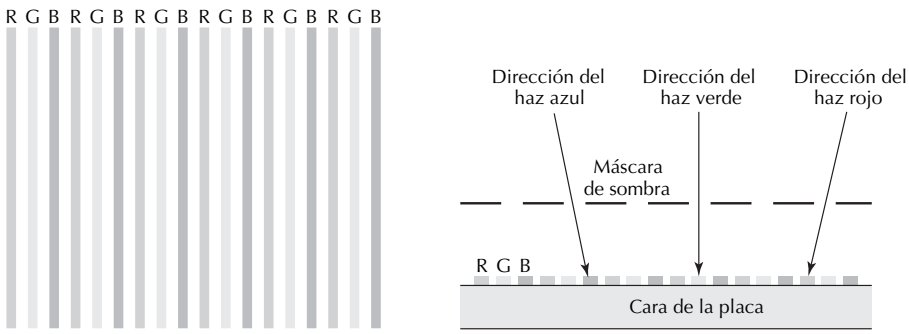
La presentación visual de símbolos e imágenes es el método de salida primario para la mayoría de las computadoras. Las opciones disponibles varían desde pequeñas pantallas monocromas (de los tipos usados en los teléfonos celulares y calculadoras baratos) hasta los grandes dispositivos de presentación de alta resolución, y más bien costosos, dirigidos al uso de artistas gráficos y publicistas profesionales. Hasta hace poco el *tubo de rayos catódicos* (CRT) era el tipo principal de dispositivo de presentación



a) Formación de imagen en un CRT

b) Datos que definen la imagen

Figura 21.5 Unidad de presentación CRT y almacenamiento de imagen en *buffer* de marco.



a) Tiras de color RGB

b) Uso de máscara de sombra

Figura 21.6 Esquema de color RGB de las modernas pantallas CRT.

visual para las computadoras de escritorio, las pantallas de panel plano se reservaban para su uso en laptop y otros dispositivos digitales portátiles. Es predecible que la mayoría de los voluminosos, pesados y grandes consumidores de electricidad CRT gradualmente serán sustituidos por las unidades de pantalla de panel plano, cuyos costos son accesibles y cuyas pequeñas huellas y menor generación de calor son ventajas importantes para el hogar y la oficina.

Las pantallas CRT funcionan mediante un haz de electrones que barre la superficie de una pantalla sensible, y crea un pixel oscuro o claro conforme pasa cada punto (figura 21.5*a*). Los primeros CRT eran *monocromáticos*. En tales CRT, el haz de electrones golpea una capa de fósforo en la parte posterior del vidrio de la pantalla. Esta capa de fósforo emite luz porque es golpeada por una corriente de electrones que viaja a muy alta rapidez, y la intensidad del haz de electrones que pasa un punto específico determina el nivel de brillantez. Para proteger la capa de fósforo del bombardeo directo de los electrones, detrás de ella se coloca una capa de aluminio; ésta actúa como contacto eléctrico. La tecnología del CRT de color pasó por varias etapas. Los primeros intentos usaron un CRT monocromático que se veía mediante la rotación de filtros de color mientras se desplegaban a la vez los pixeles asociados con varios colores. Otros diseños incluyeron la sustitución de los filtros mecánicos con otros controlados electrónicamente y el uso de múltiples capas de fósforo, cada uno emitiendo luz de color diferente.

Los CRT modernos en uso común actualmente se basan en el método tricolor de "máscara de sombra" introducido por los televisores RCA en la década de 1950. Los detalles de diseño varían, pero

es representativo el esquema que se usa en los tubos Trinitron de Sony. Como se muestra en la figura 21.6, estrechas tiras de fósforo que producen tres colores de luz diferentes y que se colocan en la parte posterior de la cara de vidrio del tubo. Los tres colores son rojo, verde y azul, que propicia el nombre "RGB" (por sus siglas en inglés) al esquema resultante. Tres haces de electrones separados exploran la superficie de la pantalla, cada uno llega en un ángulo ligeramente diferente. Una máscara de sombra, que representa una placa metálica con aberturas u hoyos, fuerza a cada haz a impactar sólo tiras de fósforo de un color particular. La separación de dos tiras consecutivas del mismo color, conocida como *densidad de presentación* (*display pitch*), dicta la resolución de la imagen resultante.

Los tres haces de electrones se controlan con base en una representación de la imagen que se desea presentar. Todo pixel está asociada con cuatro (imágenes simples de 16 colores) a 32 bits de datos en un *buffer de marco*. Si la resolución de la pantalla es de 1 K × 1 K pixeles, y cada pixel tiene 64K colores posibles, entonces el *buffer* de marco necesita 2 MB de espacio. Este espacio se puede proporcionar en una *memoria de video* dedicada o como parte del espacio de dirección de memoria principal (*memoria compartida*). Con frecuencia, las memorias de video dedicadas son de puerto dual para permitir el acceso simultáneo mediante el CPU para modificar la imagen o por el controlador de pantalla para presentarla. A tales memorias de video a veces se les conoce como VRAM. La figura 21.5 muestra el movimiento de barrido del haz de electrones, conforme cubre toda la superficie de la pantalla sensible 30-75 veces por segundo (conocida como *velocidad de regeneración*, *refresh rate*) y un *buffer* de marco que almacena cuatro bits de datos por pixel. En la práctica, hasta 32 bits de datos se necesitan por pixel:

32 bits, ocho para cada R, G, B, A ("color verdadero").

16 bits, cinco para cada R y B, seis para G ("color alto").

Ocho bits o 256 diferentes colores en formato VGA.

En las descripciones mencionadas, "A" significa "alfa" (un cuarto componente que se usa para controlar la mezcla de colores) y VGA representa un antiguo arreglo gráfico de video que todavía se soporta por compatibilidad.

Ejemplo 21.1: Rendimiento total de video Considere una unidad de presentación visual con una resolución de 1 024 × 768 pixeles y una velocidad de regeneración de 70 Hz. Calcule el rendimiento total requerido por la memoria de video para soportar este despliegue, si supone ocho, 16 o 32 bits de datos por pixel.

Solución: El número de pixeles a los que se accede por segundo es $1\,024 \times 768 \times 70 \cong 55\text{M}$. Esto último implica un rendimiento total de 55 MB/s, 110 MB/s o 220 MB/s, según el ancho de datos de pixel. Con un acceso de 55M pixeles por segundo, están disponibles alrededor de 18 ns para cada lectura de pixel, incluso si se ignora el tiempo desperdiciado debido a márgenes izquierdo y derecho y el tiempo de retorno del haz. En consecuencia, si se deben lograr las tasas de datos deseadas con VRAM típicas, en cada acceso se deben leer múltiples pixeles.

Físicamente, las pantallas CRT son voluminosas, requieren alto consumo de energía y generan gran cantidad de calor. Además, la imagen presentada por un CRT sufre de distorsión así como de foco y color no uniformes. Por estas razones, las pantallas de cristal líquido en panel plano (LCD) son populares, en especial a la luz de su continua mejora en la calidad de imagen y su costo económico. La figura 21.7*a* muestra una pantalla de *matriz pasiva*. Cada punto de intersección entre una hilera y una columna representa una pequeña persiana óptica que se controla mediante la diferencia entre los

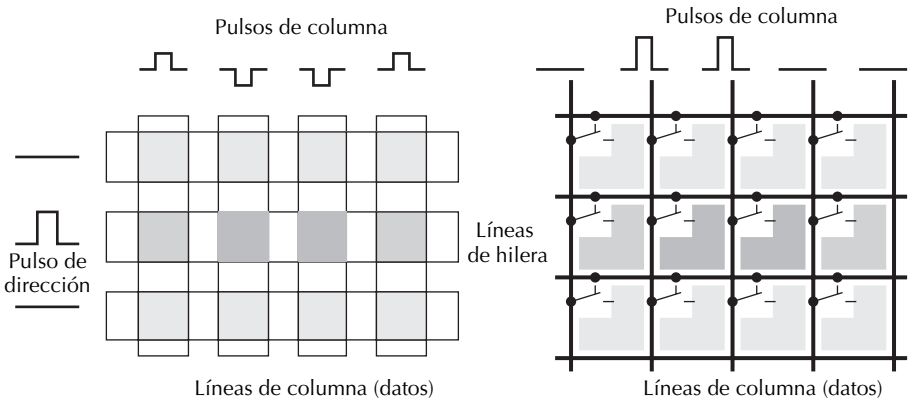


Figura 21.7 Presentaciones visuales LCD pasiva y activa.

voltajes de hilera y columna. Puesto que el voltaje de hilera se proporciona a toda celda en una hilera y las celdas además reciben voltaje cruzado de otras hileras, el contraste y la resolución tienden a ser bajos. Las *matrices activas* son similares, excepto que en cada intersección se planta un transistor de película delgada. Un transistor que se enciende permite que las celdas de cristal líquido asociadas se carguen a los voltajes en las líneas de columna. De esta forma, una imagen que se "escribe" hilera por hilera se conserva hasta el siguiente ciclo de regeneración.

Otras tecnologías de pantalla de panel plano incluyen las que se basan en diodos emisores de luz (LED, por sus siglas en inglés) y el fenómeno de plasma. Las pantallas LED consisten de arreglos de LED de uno o más colores, y cada LED se "direcciona" a través de un índice de hilera y otro de columna. Los LED electrónicos son adecuados para presentaciones que se deben ver en exteriores, requieren enorme grado de brillantez. Un desarrollo reciente en las pantallas de LED es el surgimiento de los LED orgánicos (OLED, por sus siglas en inglés), que requieren menos energía que los LED ordinarios y ofrecen una variedad de colores, incluido el blanco. La operación de las pantallas de plasma es similar a la de una lámpara de neón. Explotan la propiedad de ciertas mezclas de gases que se descomponen en plasma cuando se sujetan a un campo eléctrico muy intenso. El plasma conduce la electricidad y convierte una parte de la energía eléctrica en luz visible. En la actualidad, las pantallas de plasma son muy caras; por tanto, tienen aplicaciones limitadas.

Muchas computadoras también ofrecen salida compatible con el formato de presentación en televisión, ello permite que cualquier TV actúe como unidad de presentación. La misma señal compatible con TV se puede proporcionar a un *proyector de video* que produce una réplica de la imagen en el CRT o unidad de pantalla de panel plano de la computadora en una pantalla del tamaño de una pared. Estas tecnologías de presentación son útiles para presentaciones basadas en computadora en todas partes, desde pequeñas salas de juntas o salones de clase hasta grandes auditorios.

21.4 Dispositivos de entrada/salida de copia impresa

A pesar de las numerosas predicciones de que las oficinas sin papel, o acaso una sociedad sin papel, harían obsoletos los dispositivos I/O de copia impresa, aún existe la necesidad de tales equipos. Los documentos en papel, lejos de caer en desuso, han proliferado con el uso creciente de computadoras en los negocios y escenarios domésticos.

La entrada de copia impresa se acepta a través de *escáneres* (*digitalizadores)*. Un digitalizador funciona como una unidad de pantalla CRT, pero en dirección contraria. Mientras que una pantalla CRT

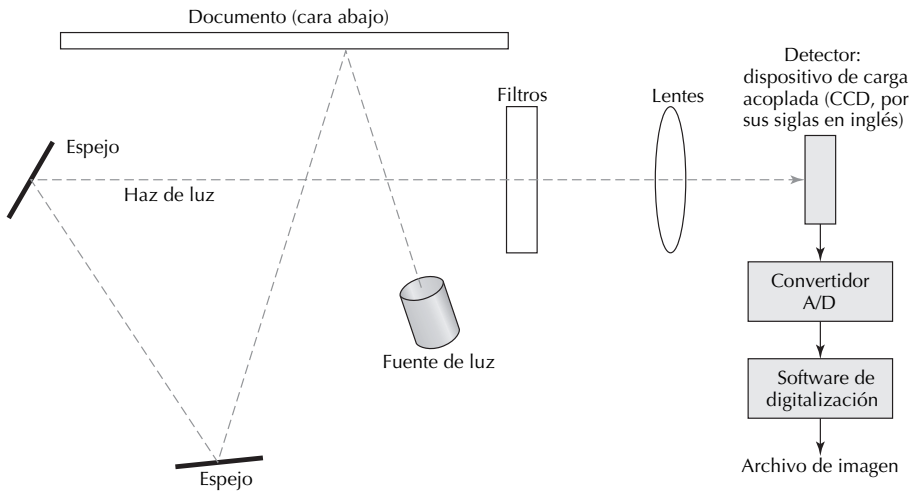


Figura 21.8 Mecanismo de digitalización para entrada de copia impresa.

convierte señales electrónicas en niveles de color y brillo con un movimiento de barrido, un escáner percibe el color y la intensidad y los convierte en señales electrónicas conforme recorre una imagen línea por línea y punto a punto dentro de cada línea (figura 21.8). Esta información se almacena en memoria en uno de muchos formatos estándar. Los escáneres se caracterizan por resolución espacial, expresada en número de pixeles o puntos por pulgada (por ejemplo, 1 200 dpi) y fidelidad de color (número de colores). En relación con el tipo de entrada de documento al escáner, tales dispositivos se dividen en los tipos *alimentador de hoja*, *cama plana*, *de cabeza* y *portátiles*. Los escáneres de cama plana tienen la ventaja de no provocar daño al documento original debido a doblado y permiten la digitalización de un libro y páginas de periódico. Los escáneres de cabeza y portátiles pueden digitalizar documentos grandes sin ser voluminosos o costosos. Los escáneres baratos están haciendo obsoletas las máquinas de fax.

Para el caso del texto digitalizado, la imagen se puede convertir en forma simbólica a través de software de *reconocimiento óptico de caracteres* (OCR, por sus siglas en inglés) y se almacena como archivo de texto. Este tipo de conversión de imagen a texto mediante digitalización y OCR se usa comúnmente para poner en línea libros antiguos y otros documentos. El reconocimiento de la escritura manual también se ha vuelto cada vez más importante conforme proliferan las aplicaciones de los *asistentes digitales personales* (PDA, por sus siglas en inglés) y las *computadoras portátiles* sin teclado.

El desarrollo de impresoras modernas es una de las historias de éxito que inspiran temor en informática. Las primeras impresoras para computadora funcionaban en forma parecida a las antiguas máquinas de escribir mecánicas; usaban dispositivos de formación de caracteres que golpeaban sobre una banda de tela o plástico mediante un mecanismo parecido a un martillo para imprimir los caracteres sobre papel uno tras otro (*impresoras de caracteres*). Para aumentar la rapidez de tales impresoras, se desarrollaron mecanismo especiales que permitían la impresión de líneas completas a la vez, ello condujo a una amplia variedad de *impresoras de línea*. Gradualmente, las impresoras de línea evolucionaron de las ruidosas y voluminosas máquinas (con tamaño de refrigerador) a unidades más pequeñas y silenciosas. Pronto hubo conciencia de que la formación de caracteres mediante la selección de un subconjunto de puntos en una matriz 2D de puntos (figura 21.9) conduciría a mayor flexibilidad en los conjuntos de caracteres arbitrarios de soporte, así como en la formación de imágenes. Por tanto, las *impresoras de matriz de punto* gradualmente sustituyeron a muchas tecnologías de impresión del tipo impacto.

Las impresoras modernas básicamente imprimen una gran imagen en matriz de punto de la página que está compuesta de archivos de texto o gráfico, mediante *postscript* u otros formatos intermedios de

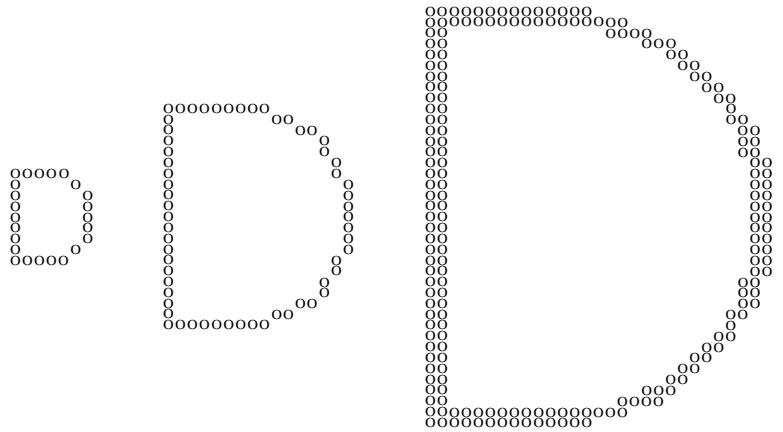
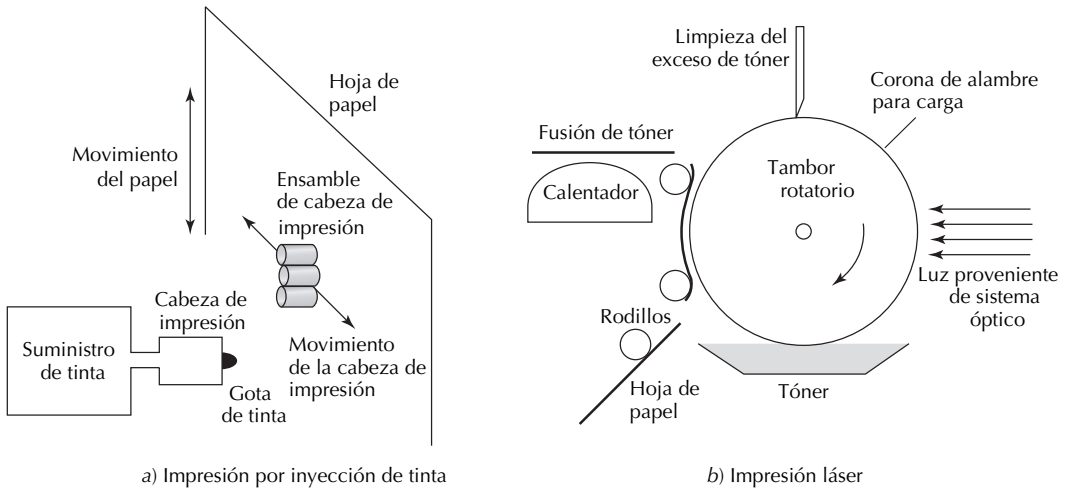


Figura 21.9 Formación de la letra "D" mediante matrices de punto de varios tamaños.



a) Impresión por inyección de tinta

b) Impresión láser

Figura 21.10 Impresoras de inyección de tinta y láser.

archivo de impresora. Cada punto en la imagen de la página se caracteriza por intensidad y color. Las *impresoras de inyección de tinta* (figura 21.10*a*), relativamente lentas y baratas, imprimen los puntos uno o pocos a la vez. Las gotas de tinta se expulsan desde la cabeza de impresión mediante varios mecanismos como calentamiento (que conduce a la expansión de una burbuja de aire dentro de la cabeza) o generación de ondas de choque a través de un transductor de cristal piezoeléctrico al lado del depósito. Las *impresoras láser* más grandes y rápidas (figura 21.10*b*) forman una copia de la imagen a imprimir en la forma de patrones de carga eléctrica en un tambor rotatorio e imprimen toda la página a alta rapidez.

Al igual que un escáner, una impresora de matriz de punto se caracteriza por su resolución espacial, expresada en número de pixeles o puntos por pulgada (por ejemplo, 1 200 dpi), y fidelidad de color (número de colores). El rendimiento total de impresión, otra característica clave de rendimiento para las impresoras de computadora, varía de unas cuantas a muchos centenares de páginas por minuto (ppm), en ocasiones varían incluso para la misma impresora dependiendo de la resolución y requisitos de color. Muchas impresoras antiguas requerían papel especial cubierto con químicos o empacado en rollos para simplificar el mecanismo de alimentación de papel. Las impresoras modernas usan papel ordinario, se denominan *impresoras de papel normal*.

Las impresoras para blanco y negro depositan gotas de tinta o funden partículas de toner en el papel de acuerdo con los requisitos del documento. Los niveles de gris se crean al dejar que se muestre parte del blanco del papel subyacente. Las impresoras de color funcionan de modo similar a los CRT de color en que crean varios colores a partir de tres colores. Sin embargo, los tres colores que se usan en las impresoras son cian (azul-verde), magenta y amarillo (*yellow*), que en conjunto forman el esquema de color CMY (por sus siglas en inglés), es diferente del RGB de los CRT. Las razones para la diferencia tienen que ver con la forma en que el ojo humano percibe el color. El esquema de color CMY representa la ausencia de los colores RGB (cian es la ausencia de rojo, etc.). De este modo, el cian absorbe rojo, el magenta absorbe verde y el amarillo absorbe azul. El esquema de color RGB es aditivo, ello significa que un color deseado se crea al sumar la cantidad adecuada de cada uno de los tres colores primarios. El esquema CMY es sustractivo y forma un color deseado al remover los componentes adecuados de la luz blanca. La mezcla de estos tres colores en cantidades iguales absorbería los tres colores primarios y dejaría negro. Sin embargo, el blanco así producido es más bien insatisfactorio, especialmente en vista de la extrema sensibilidad del ojo humano ante cualquier color corrido en negro. Por esta razón, la mayoría de las impresoras de color usan el esquema CMYK, donde K representa al negro.

La impresión a color es mucho más complicada que la pantalla a color. Las razones incluyen la dificultad de controlar exactamente el tamaño y alineación de los puntos de los diversos colores y la posibilidad de que los colores se corran si se colocan muy juntos. Mayores problemas surgen de la resolución reducida cuando se deben soportar diferentes niveles de intensidad (escalas de grises en negro y blanco). Por ejemplo, una forma de crear la ilusión de cinco niveles de gris (0, 25, 50, 75 y 100%) es dividir el área de impresión en bloques de pixeles de 2×2, y colocar 0-4 pixeles negros en un bloque para crear los cinco niveles. Sin embargo, esto último reduce la resolución por un factor de 2 en cada dirección (un global cuadruplicado).

Aunque las impresoras modernas pueden producir salida de copia impresa de alta calidad, también están disponibles dispositivos de salida de copia impresa especializada para requerimientos particulares. Los ejemplos incluyen *plotters* (trazadores), que se usan para producir dibujos técnicos y arquitectónicos, usualmente en papeles muy grandes, y las *impresoras fotográficas*, que difieren de las impresoras normales sólo en la calidad de sus mecanismos de impresión y los tipos de papel que aceptan.

El uso de máquinas de oficina que combinan escáner/impresora es cada vez más común. Tales máquinas pueden ofrecer capacidades de transmisión de fax y fotocopiado con poco hardware adicional. Las máquinas de fax digitalizan documentos en archivos de imagen antes de la transmisión, de modo que, cuando hay presente capacidad de digitalización, el resto es sencillo. De hecho, los documentos cada vez más se envían mediante escaneo seguido por transmisión como adjunto (*attachment*) a un correo electrónico, en lugar de máquinas de fax. El copiado es, en esencia, digitalización seguido por impresión.

Ejemplo 21.2: Rendimiento total de datos de una copiadora digital Si supone una resolución de 1 200 dpi, área de copiado de 8.5 pulgadas $\times$ 11 pulgadas y color de 32 bits en una copiadora digital compuesta de un escáner seguido por una impresora láser, derive el rendimiento total de datos necesarios para soportar un rendimiento total de copiado de 20 ppm (páginas por minuto).

Solución: La tasa de datos para imprimir un tercio de una página por segundo es $8^1/_2 \times 11 \times 1\,200^2 \times 4/3 \cong 180$ MB/s. Si supone que los datos provenientes del escáner se almacenan en una memoria y luego se recuperan para impresión, la tasa de datos a soportar por la memoria es de 360 MB/s. Esto está bien adentro de las capacidades de las modernas DRAM. Sin embargo, es posible reducir sustancialmente esta tasa al aprovechar el espacio en blanco de la mayoría de los documentos (compresión de datos).

21.5 Otros dispositivos de entrada/salida

Además de las memorias secundaria y terciaria que constituyen los dispositivos I/O usados comúnmente para almacenamiento estable y archivado de datos (capítulo 19), están disponibles muchos otros tipos de unidades de entrada y/o salida. En esta sección se revisan las más importantes de dichas opciones para completar el estudio acerca de los dispositivos I/O.

Una cámara digital fija o de video captura imágenes para ingresar a una computadora en forma muy similar a un escáner. La luz entrante proveniente del exterior de la cámara se convierte en pixeles y se almacena en memoria *flash* o algún otro tipo de unidad de memoria no volátil. Las cámaras digitales fijas se caracterizan por su resolución en términos del número de pixeles en cada imagen capturada. Por ejemplo, una cámara de un megapixel puede tener una resolución de $1\,280 \times 960$. Las cámaras con resoluciones de cinco o más megapixeles son muy costosas y usualmente no se necesitan en aplicaciones sofisticadas, pues una cámara de dos megapixeles puede entregar impresiones de calidad fotográfica de 20×25 cm en impresoras de inyección de tinta. Las cámaras digitales pueden usar zoom óptico tradicional para acercar los objetos; también pueden tener zomm digital que usa algoritmos de software para acercar una parte particular de la imagen digital, pero en el proceso se reduce la calidad de la imagen. Algunas cámaras digitales fijas pueden tomar películas cortas de resolución más bien baja. Las videocámaras digitales son capaces de capturar videos de gran calidad, mientras que las webcams se usan para capturar imágenes en movimiento con el propósito de dar seguimiento, videoconferencias y aplicaciones similares en las que la calidad y suavidad del movimiento de la imagen no son cruciales. Las fotografías y películas se almacenan en las computadoras en una diversidad de formatos estándar, usualmente en forma comprimida para reducir los requisitos de almacenamiento. Los ejemplos más comunes incluyen JPEG (por sus siglas en inglés, *Joint Photographic Experts Group* = Grupo conjunto de expertos fotográficos) y GIF (por sus siglas en inglés, *Graphic Interchange Format* = Formato de intercambio gráfico), para imágenes, y MPEG y Quick Time para películas.

Imágenes tridimensionales se pueden capturar para su proceso por computadora mediante escáneres de volumen. Un método consiste en proyectar una línea láser sobre el objeto 3D para digitalizarlo y capturar la imagen de la línea donde interseca el objeto mediante cámaras de alta resolución. A partir de imágenes capturadas y la información acerca de la posición y orientación de la cabeza que digitaliza, se calculan las coordenadas de superficie del objeto.

La entrada de audio se captura mediante micrófonos. La mayoría de las computadoras de escritorio y laptop vienen con micrófono, o un par estéreo, y contiene una tarjeta sonora que puede capturar sonidos mediante un puerto de micrófono. Para almacenar en una computadora, el sonido se digitaliza al tomar muestras de la forma ondulatoria a intervalos regulares. La fidelidad del sonido se mejora al aumentar la tasa de muestreo y al usar más bits para representar cada muestra (por decir, 24 bits para resultados profesionales, en lugar de 16 bits). En virtud de que estas provisiones conducen a aumento en los requisitos de almacenamiento, usualmente el sonido se comprime antes de almacenarlo. Los ejemplos de formatos estándar para audio comprimido incluyen MP3 (abreviatura para MPEG-1, capa 3), Real Audio, Shockwave Audio y el formato WAV de Microsoft Windows. MP3 comprime los archivos de audio hasta casi 1 MB por minuto y conduce a audio con calidad de CD porque en el curso de la compresión remueve sólo componentes de sonido que superan el rango auditivo humano. Además de esos formatos, se pueden usar los de película MPEG y Quick Time para almacenar sonido al ignorar el componente de video.

Ahora es frecuente el uso de sensores para proporcionar información acerca del ambiente y otras condiciones de interés para las computadoras. Por ejemplo, existen tableros de sensores en un automóvil y muchos miles de aquéllos en una planta industrial moderna. A continuación se presenta una lista parcial de sensores usados comúnmente y sus aplicaciones.

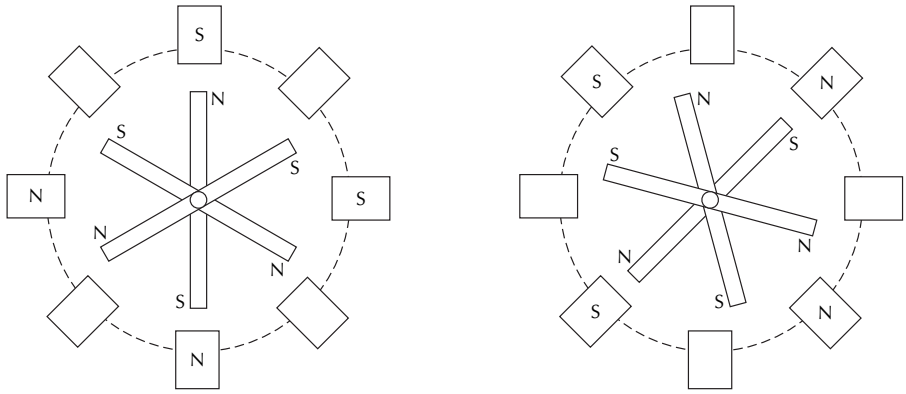
- Las fotoceldas constituyen los *sensores de luz* más simples para controlar luces nocturnas, semáforos, sistemas de seguridad, cámaras y juguetes. Una fotocelda incorpora una resistencia variable que cambia con la luz (desde muchos miles de ohms en la oscuridad hasta casi un kilohm en luz brillante), ello facilita detectar la cantidad de luz mediante un convertidor analógico-digital.
- Los sensores de temperatura son de dos tipos. Un detector de contacto mide su propia temperatura y deduce un objeto con el que está en contacto, si supone equilibrio térmico. La medición se basa en la variación de las propiedades del material (como resistencia eléctrica) con la temperatura. Los detectores de no contacto miden la radiación infrarroja u óptica que reciben de un objeto.
- Los detectores de presión convierten la deformación o alargamiento en materiales a cambios en algunas propiedades eléctricas mensurables. Por ejemplo, el alambre en zigzag incrustado en un sustrato plástico en un *extensímetro* debido a deformación, con lo que su resistencia cambia ligeramente. Los detectores de presión microelectromecánica ofrecen mayor sensibilidad y precisión. Los detectores de presión se usan en tuberías, control de motores, alas de aviones etcétera.

Las innovaciones en el diseño de sensores son de inmenso interés como resultado de la creciente demanda en el control incrustado, los sistemas de seguridad y aplicaciones militares. Las nuevas tecnologías que se siguen incluyen sensores microelectromecánicos (MEM), particularmente para presión, y biosensores, que incorporan componentes biológicos, ya sea en los mecanismos usados para detectar o en el fenómeno detectado. Los sensores MEM ofrecen mayor sensibilidad y precisión, así como tamaño pequeño. Las ventajas de los biosensores incluyen bajo costo, mayor sensibilidad y economía de energía. Se usan en control de la contaminación, análisis bacterial, diagnóstico médico y minado, entre otras aplicaciones.

La salida de imagen mediante computadoras se produce al "renderizar" varios tipos de dispositivos de presentación visual (sección 21.3) o imprimir/graficar en papel (sección 21.4). De manera adicional, las imágenes se pueden transferir directamente a microfilme para archivar o proyectar en una pantalla durante presentaciones audiovisuales. Lo último se hace usualmente al conectar un videoproyector al puerto de salida de pantalla de una computadora de escritorio o laptop o a un puerto especial que ofrece salida para formato de TV. Las salidas gráficas más exóticas incluyen imágenes estereográficas para aplicaciones de realidad virtual e imágenes holográficas.

La salida de audio se produce a partir de archivos de audio, y la tarjeta de sonido convierte archivos almacenados posiblemente comprimidos en ondas eléctricas adecuadas para enviar a una o más bocinas. Además de los formatos de archivo de audio ya mencionados en conexión con la entrada de audio, la salida de sonido se puede producir a partir de archivos MIDI (por sus siglas en inglés, *Musical Instrument Digital Interface*, interfaz digital de instrumento musical) mucho más compactos. Los archivos MIDI no contienen audio grabado sino instrucciones con el fin de que la tarjeta de sonido produzca notas particulares para ciertos instrumentos musicales. Por esta razón, los archivos MIDI usualmente son 100 o más veces más pequeños que archivos MP3 comparables, pero tienen mucho menos fidelidad. Los sintetizadores de habla también constituye un área de gran interés, pues permite una interfaz de usuario más natural en muchos contextos. La síntesis de habla se puede realizar mediante la unión de fragmentos pregrabados o con conversión algorítmica texto a habla.

Ahora es común que las computadoras se usen para control directo de una diversidad de dispositivos mecánicos. Esto último se hace mediante *actuadores* que convierten señales eléctricas a fuerza y movimiento. Los *motores de velocidad gradual* (también llamados *motores de paso*) son los componentes más usados. Un motor de velocidad gradual es capaz de pequeñas rotaciones al recibir uno o más pulsos de control. Entonces una rotación se puede convertir a movimiento lineal, si se desea, o usarse directamente para mover cámaras, brazos robóticos y muchos otros tipos de dispositivos. La figura 21.11 muestra las funciones de un motor de velocidad gradual típico. El rotor consta de imanes



a) Etapa inicial

b) Después de la rotación

Figura 21.11 Principios de operación del motor de velocidad gradual.

separados 60°. El estator se divide en ocho secciones con 45° de espaciamiento. Suponga que la mitad de los segmentos de estator se magnetizan como se muestra en la figura 21.11*a*). Ahora, si la mitad restante de los segmentos de estator se magnetizan como en la figura 21.11*b*) y la corriente de magnetización se remueve del primer conjunto, el rotor gira 15° (la diferencia entre el espaciamiento de 60° del rotor y el espaciamiento de 45° del estator).

Hay alguna esperanza de que, eventualmente, dispositivos sin partes mecánicas permitirán la construcción de generadores de movimiento más pequeños, más ligeros, más fuertes y de bajo consumo eléctrico. Los *polímeros electroactivos* parecidos a músculos, que se expanden y contraen en respuesta a la estimulación eléctrica, representan una tecnología útil que actualmente se persigue [BarC01].

21.6 Redes de dispositivos de entrada/salida

En forma creciente, entrada y salida involucran transferencias de archivos mediante una red. Por ejemplo, enviar un documento impreso a una impresora que tiene un gran *buffer* de datos y un CPU no es muy diferente de enviar un archivo a otra computadora. Los dispositivos periféricos basados en red permiten compartir recursos inusuales o rara vez usados y también ofrecen opciones de respaldo en el evento de falla o sobrecarga de un recurso local (figura 21.12). Por ejemplo, en un escenario de oficina, un grupo de trabajo puede compartir una costosa impresora láser a color, con otra impresora ubicada en un departamento diferente, designada como respaldo en caso de descompostura. Las I/O en red también ofrecen flexibilidad en la colocación de dispositivos I/O de acuerdo con necesidades dinámicamente cambiantes y permiten la actualización o sustitución, mientras que reduce el cableado y los problemas de conectividad. Más aún, según la universalidad de los estándares de comunicación en red, como IP y Ethernet (sección 23.1), las I/O en red mejoran la compatibilidad y facilitan la interoperatibilidad.

En ninguna parte los beneficios de las I/O en red son más evidentes que en el control de procesos industriales. En tales sistemas, tableros de cientos o miles de sensores, actuadores y controladores deben interactuar con una computadora central o con nodos de plataforma distribuida. El uso de una red con una topología de árbol o ciclo, en lugar de conexiones punto a punto, conduce a reducción significativa en el cableado, con la acompañante reducción en costos de operación y mantenimiento. La misma red puede usarse para intercambiar datos, diagnósticos, configuración y calibración, ello ahorra costos. Tal nivel reducido de alambrado se puede evitar si se usa red inalámbrica.

En el contexto anterior, entrada/salida se mezcla ampliamente con funciones de procesamiento y control, y las fronteras entre las diversas funciones se vuelven confusas. Un sensor con un procesador

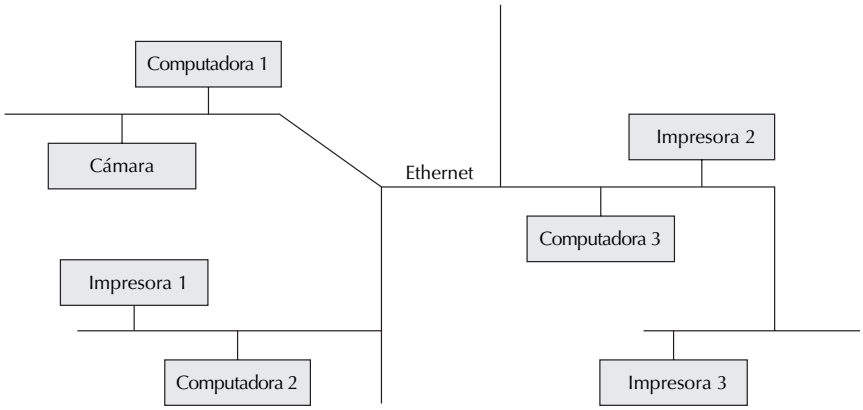


Figura 21.12 Con periféricos habilitados por red, la I/O se realiza mediante transferencia de filtros.

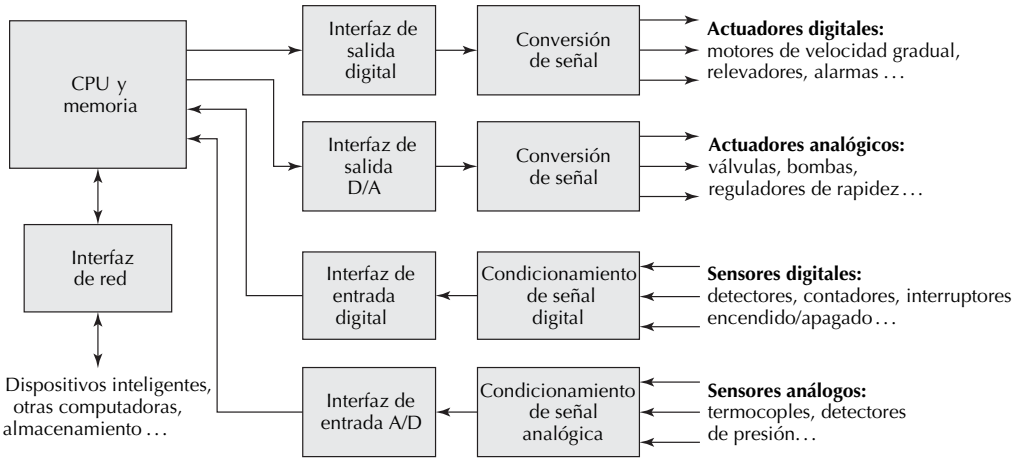


Figura 21.13 La estructura de un sistema de control de ciclo cerrado basado en computadora.

interconstruido y adaptador de red, integrado en un solo chip que use MEM y otras tecnologías, se visualiza como un dispositivos de entrada o como un nodo en un sistema de procesamiento distribuido (sección 28.4). Se trata de un dispositivo de entrada en el sentido de que su principal función y meta final es proporcionar información acerca de su entorno a un proceso de control o algoritmo. Es un nodo en el sentido de que es capaz de cooperar con otros nodos, correr autodiagnósticos, realizar autocalibración o calibración mutua, y realiza varias funciones relacionadas con recolección de datos tradicionalmente relegadas a un procesador central. Lo último incluye tareas relacionadas a operación confiable y precisa: filtrar ruido, reintento de transmisión, adaptación a cambios en el entorno, etcétera.

La figura 21.13 muestra la estructura de un sistema de control de ciclo cerrado basado en computadora con interfases especializadas para manejar sensores y actuadores de varios tipos. Con dispositivos habilitadores por red, la mayoría de éstos desaparecen o se incorporan en el subsistema sensor/actuador. Todo pasa por la computadora de control vía la interfaz de red, ello simplifica su labor. El diseñador del software para tal sistema de control basado en computadora pueden enfocarse en los requisitos algorítmicos y de rendimiento relevantes, no así en las necesidades de interfaz específicas para cada tipo de dispositivo.

PROBLEMAS

21.1 Teclados y sus interruptores

a) Los interruptores mecánicos que se muestran en la figura 21.3*a* parecen requerir gran cantidad de movimiento vertical para cerrar el contacto eléctrico. Si examina el teclado en una moderna computadora notebook ultradelgada, observará que las teclas se mueven muy poco. Investigue el diseño de tales interruptores de movimiento pequeño.

b) Algunos teclados se publicitan como "a prueba de derrames". ¿Cuáles son las implicaciones de esta propiedad sobre el diseño de interruptores y otras partes de un teclado?

c) Mencione otras dos propiedades físicas de un teclado que considere importantes para un usuario.

d) Mencione dos propiedades negativas de un teclado que conducirían a insatisfacción o rechazo del usuario.

21.2 Codificación de teclado

Diseñe un codificador para el teclado numérico de la figura 21.3*c* y suponga que la salida debe ser:

a) Un dígito hexa de cuatro bits.

b) Un carácter ASCII de ocho bits que represente al dígito hexa.

21.3 Codificación de teclado

a) La discusión de codificación de teclado en la sección 21.2 supuso tácitamente que cuando mucho una tecla se presiona en un momento dado. La mayoría de los teclados soportan dobles presiones de teclas (por ejemplo, una tecla ordinaria más una de las teclas especiales shift, alt o control) para permitir un conjunto más amplio de símbolos. Especule acerca de cómo se logra la codificación en este caso.

b) En máquinas basadas en Windows, al menos se soporta una triple presión de tecla (Alt + control + delete). ¿Puede pensar en una buena razón para incluir esta característica, que ciertamente complica el proceso de codificación?

c) En las previsiones de las partes *a*) y *b*), múltiples presiones de tecla conducen a la transmisión de un símbolo del teclado a la computadora. Lo opuesto también puede ser cierto para algunos teclados: múltiples símbolos enviados como resultado de una sola presión de tecla. ¿Por qué es útil esta característica? Describa una forma de efectuar el codificador requerido.

21.4 Dispositivos apuntadores

Realice el siguiente experimento en la touchpad de 5×6 cm^2 de una computadora laptop. Mueva el cursor a la esquina superior izquierda de la pantalla de 21×28 cm^2. Coloque un dedo en la esquina superior izquierda del touchpad y arrástrelo rápidamente hacia la esquina inferior derecha de la almohadilla. La nueva posición del cursor está cerca de la esquina inferior derecha de la pantalla. Ahora, mueva el dedo en la dirección opuesta a lo largo de la misma ruta diagonal, pero más lentamente. Cuando el dedo llegue a la esquina superior izquierda de la touchpad, el cursor está cerca del centro de la pantalla. ¿Qué le dice este experimento acerca de cómo se procesan los datos de posición del touchpad? Justifique sus conclusiones y discuta sus implicaciones prácticas.

21.5 Rendimiento total de la memoria de video

En el ejemplo 21.1 la tasa de datos se derivó con la suposición de que todos los pixeles en el *buffer* de marco se deben actualizar en cada marco.

a) Mencione tres aplicaciones en las que sólo una fracción muy pequeña de los pixeles cambie de un marco al siguiente.

b) Describa una forma de enviar datos hacia el *buffer* de marco que permite actualización selectiva de los pixeles o pequeñas ventanas dentro del área de presentación.

c) Discuta la cabecera debida a la transmisión selectiva de la parte *b*), tanto en términos de envío de bits adicionales (además de los datos de pixel reales) como de tiempo de procesamiento adicional necesario para extraer los datos de pixel.

21.6 Tablero de marcador

Algunas pantallas de tablero de marcador monocromas se construyen a partir de arreglos 2D de bombillas luminosas que se pueden encender y apagar bajo control programado.

a) Describa cómo se puede controlar un tablero de este tipo de 100×250 mediante 16 señales de datos provenientes de un microcontrolador.

b) Discuta los beneficios y perjuicios de dos formas de ordenar las bombillas en el tablero: la j-ésima bombilla de la hilera $2i + 1$ alineada verticalmente con la

j-ésima bombilla de la hilera 2*i*, contra su alineación con el punto medio entre la *j*-ésima y la (*j* + 1)-ésima bombillas en la hilera 2*i*.

c) ¿Es factible construir un tablero de color de este tipo?

21.7 Resolución ajustable en monitores

Los diseños y métodos descritos para monitores CRT y de panel plano de la sección 21.3 sugieren espaciamientos fijos de pixel. Sin embargo, en una computadora personal típica, el usuario puede fijar la resolución (número de pixeles en la pantalla) mediante un programa de utilidad de sistema. ¿Cómo se implementan realmente dichas variaciones? En otras palabras, las aberturas de la máscara de sombra en la figura 21.6*b* o las líneas de hilera y columna en la figura 21.7 no cambian. De este modo, ¿qué cambia cuando se varía la resolución?

21.8 Escáner

Las especificaciones para un escáner indican que tiene una resolución de 600 dpi en la dirección *x* y 1 200 dpi en la dirección *y*.

a) ¿Por qué cree que las resoluciones son diferentes en las dos direcciones?

b) ¿Qué tipo de procesamiento de software permitiría imprimir una imagen capturada por este escáner en una impresora de 1 200 dpi?

c) Repita la parte *b*) para una impresora de 600 dpi.

21.9 Rendimiento total de datos en una copiadora

Un memorando típico de oficina, la página de un libro u otro documento que se copia contienen principalmente espacio en blanco.

a) Discuta cómo se puede usar esta información para reducir la tasa de datos requerida para una copiadora, como se derivó en el ejemplo 21.1.

b) ¿Se pueden lograr ahorros similares en las tasas de datos o espacio de almacenamiento cuando una página (por ejemplo, la página de una revista con fondo negro y fuente blanca) no incluye espacio blanco?

21.10 Tecnologías de impresoras

Con base en la investigación que realice acera de impresoras de inyección de tinta y láser, compare y contraste las dos tecnologías con respecto a los siguientes atributos:

a) Calidad de salida en términos de resolución y contraste.

b) Calidad de salida en términos de durabilidad (falta de borrado con el tiempo).

c) Latencia y rendimiento total de la impresión.

d) Costo de la tinta o toner por página impresa.

e) Costo total de propiedad por página impresa.

f) Facilidad de uso, que incluye dimensiones físicas, ruido y generación de calor.

21.11 Impresoras de tambor

Un tipo particular de impresora de línea usada hace muchos años era la impresora de tambor. Un gran tambor metálico rotaba a alta rapidez junto a una ruta de papel. El tambor tenía tantas tiras como caracteres en una línea; por decir, 132. En cada tira, las letras y símbolos de interés aparecían en una forma resaltada. También había 132 martillos alineados a lo largo de la longitud del tambor. Cada martillo se controlaba individualmente y golpeaba sobre una cinta justo cuando pasaba por abajo el carácter deseado para dicha posición.

a) ¿Por qué tal impresora tendía a embarrar los caracteres?

b) ¿Cómo se relaciona la rapidez de impresión de una impresora de tambor, con la latencia de acceso a memoria de disco? En particular, compare con los discos de cabeza por pista y múltiples cabezas.

c) Analice las latencias de peor caso y caso promedio para imprimir una línea.

21.12 Detección y medición vía fotoceldas

Una aplicación de las fotoceldas en las fábricas está en la clasificación de objetos que se mueven en una banda transportadora. Si supone que los objetos no se apilan y que no se traslapan en longitud, en la banda transportadora:

a) Especifique cómo puede usarse una sola fotocelda para clasificar objetos por longitud. Establezca claramente todas sus suposiciones.

b) Proponga un esquema para clasificar objetos de altura fija por sus alturas.

c) Repita la parte *b*), pero esta vez suponga que la altura puede variar a lo largo del objeto.

d) Proponga un esquema para clasificar objetos en cubos, esferas y pirámides.

21.13 Motores de rapidez gradual

Un motor de rapidez gradual rota 15° por cada pulso de control recibido. Especifique los tipos de mecanismo que necesitaría si tuviese que usar este motor para controlar el movimiento lateral de la cabeza de impresión para una impresora de inyección de tinta de 600 dpi. Observe que los mecanismos requeridos deben convertir el movimiento rotacionaldel motor a movimiento lineal con granos más finos.

21.14 Dispositivos de entrada primitivos

Los conmutadores basculantes y *jumpers* constituyen primitivos mecanismos de entrada para dispositivos que no tenían un teclado u otros dispositivos de entrada. Un conmutador basculante puede fijar un bit de entrada a 0 o 1. Un *jumper* hace lo mismo al conectar la entrada a un voltaje constante que representa 0 o 1. Por lo general, se usan cuando se necesitan pocos bits de datos de entrada y la entrada cambia de manera irregular.

a) Identifique dos aplicaciones en las que se usen tales dispositivos de entrada.
b) Compare los conmutadores basculantes y *jumpers* en relación con la facilidad de uso y flexibilidad.
c) Describa cómo ocho conmutadores basculantes y un botón pueden ofrecer una forma de ingresar más de ocho bits de datos.

21.15 Dispositivos especiales de entrada

Investigue los siguientes temas en relación con los dispositivos especiales de entrada y escriba un informe de dos páginas acerca de cada uno.

a) Por qué los números impresos al fondo de la mayoría de los cheques bancarios (incluidos el número ID del banco y el número de cuenta) tiene formas inusuales.
b) Cómo el código de producto universal (UPC), que se encuentra en la mayoría de los productos, codifica datos y cómo los lee mediante un escáner en el contador de salida.
c) Cómo las tarjetas perforadas de 80 columnas (conocidas como Hollerith) codificaban datos y cómo se compara su densidad de almacenamiento, en bits por centímetro cúbico, con los discos y otros tipos de memoria actuales.
d) Cómo funcionan los lectores de tarjetas de crédito, que se encuentran en muchas tiendas, cajeros automáticos y estaciones de servicio para la venta de gasolina.
e) Cómo el texto escrito a mano, que se ingresa mediante un estilete, se percibe y acepta en las PDA o PC tablet.
f) Cómo se comunican con la computadora los teclados, ratones y controles remoto inalámbricos (para cambiar diapositivas en una presentación).

21.16 Dispositivos de salida especiales

Investigue los siguientes temas en relación con los dispositivos de salida especiales y escriba un informe de dos páginas acerca de cada uno.

a) Cómo se produce la salida Braille para usuarios invidentes.
b) Cómo las impresoras pequeñas, unidas a las calculadoras que imprimen, difieren de las impresoras ordinarias de inyección de tinta o láser.
c) Cómo funcionan los dispositivos de presentación por proyección trasera.
d) Cómo funciona una pantalla de segmento de línea (por ejemplo, la pantalla LED de siete segmentos para números) y por qué ya no son de uso extenso.

REFERENCIAS Y LECTURAS SUGERIDAS

[BarC01] Bar-Cohen, Y., *Electroactive polymer (EAP) actuators as artificial muscles: reality, potential, and challenges*, SPIE Monograph, vol. PM98, 2001.

[Clem00] Clements, A., *The Principles of Computer Hardware*, Oxford University Press, 2000.

[Howa92] Howard, W. E., "Thin-Film-Transistor/Liquid Crystal Display Technology: An Introduction", *IBM J. Research and Development*, vol. 36, núm. 1, pp. 3-10, enero de 1992.

[Mint98] Mintchell, G. A., "Networked I/O Strategies Connect", *Control Engineering International*, noviembre de 1988. www.manufacturing.net/ctl/index.asp

[Myer02] Myers, R. L., *Display interfaces: fundamentals and standards*,Wiley, 2002.

CAPÍTULO 24

CONMUTACIÓN CONTEXTUAL E INTERRUPCIONES

"Cuando tu trabajo habla por sí mismo, no interrumpas."
Henry J. Kaiser

"Como decía el otro día…"
Luis Ponce de León, al resumir una conferencia interrumpida por cinco años de prisión

TEMAS DEL CAPÍTULO

Los usuarios y programadores comunes usualmente no necesitan preocuparse por los detalles de la operación de los dispositivos I/O y cómo controlarlos. Estas tareas se relegan al sistema operativo, al que los programas usuarios llaman siempre que se necesite I/O. A su vez, el sistema operativo activa los controladores del dispositivo que generan procesos I/O asíncronos que realizan sus tareas y reportan de vuelta al CPU mediante interrupciones. Cuando se detecta una interrupción, el CPU puede cambiar el contexto, hacer a un lado el programa activo y ejecutar una rutina de atención de interrupción. Este tipo de conmutación contextual también es viable entre diferentes programas usuarios, o entre hilos del mismo programa, como un mecanismo para evitar largas esperas debidas a la indisponibilidad de datos o recursos de sistema.

24.1 Peticiones al sistema por I/O

Como se mencionó en la sección 7.6, y de nuevo al final de la sección 22.2, la mayoría de los usuarios no se involucran en detalle de transferencias I/O entre dispositivos y memoria, sino que usan peticiones de sistema para lograr transferencias de datos I/O. Una razón para I/O indirecta a través del sistema operativo constituye la necesidad de proteger datos y programas contra daño accidental o deliberado. Otra significa que es más conveniente para el usuario, así como para el sistema, apoyarse en rutinas de sistema operativo para proporcionar una interfaz clara y uniforme a tales dispositivos en una forma que sea, en la

medida de lo posible, independiente de dispositivo. Una rutina de sistema inicia la I/O, que usualmente se realiza mediante DMA, después cede el control al CPU. Más tarde, este último recibirá una interrupción que significa la conclusión de I/O y permitirá que la rutina I/O realice su etapa de limpieza.

Por ejemplo, al leer datos de un disco, el usuario debe evitar acceder o modificar datos o archivos del sistema que pertenezcan a otro usuario. Adicionalmente, durante una operación de lectura se pueden encontrar una docena o más tipos de errores. Usualmente cada uno de los errores tiene un *bit flag* asociado en el registro de estatus del dispositivo. Los ejemplos incluyen número inválido de cilindro (pista, sector), violación de *cheksum*, temporización anómala y dirección de memoria inválida. Estos bits de estatus se deben verificar en cada operación I/O e iniciar acción apropiada para lidiar con cualquier anomalía. Tiene perfecto sentido liberar al usuario de tales complejidades al proporcionar rutinas de atención I/O, cuyas interfases de usuario se enfoquen en la transferencia de datos requeridas y no en la mecánica de la transferencia en sí misma.

La conveniencia e independencia de dispositivo se logran al proporcionar varias capas de control entre el programa usuario y los dispositivos I/O de hardware. Las capas típicas incluyen:

Kernel OS → Subsistema I/O → *Driver* de dispositivo → Controlador de dispositivo → Dispositivo

El *driver* del dispositivo es una liga de software entre los dos componentes hardware a la derecha y el sistema operativo y su subsistema I/O a la izquierda. Como tal, es dependiente tanto de OS como de dispositivo. Un nuevo dispositivo I/O que no siga una interfaz hardware/software ya establecida se debe introducir en el mercado con *drivers* de dispositivo para varios sistemas operativos populares. La idea es capturar, tanto como sea posible, las características de un dispositivo I/O en su *driver* asociado, de modo que el subsistema I/O necesite lidiar sólo con ciertas categorías generales, y que cada una abarque muchos dispositivos I/O.

El sistema operativo (OS, por sus siglas en inglés) representa el software de sistema que controla todo en una computadora. Sus funciones incluyen gestión de recursos, cronograma de tareas, protección y manejo de excepción. El subsistema I/O dentro del sistema operativo maneja las interacciones con dispositivos I/O y gestiona las transferencias de datos reales. Con frecuencia consta de una capa básica que está estrechamente involucrada con características de hardware y otros módulos de apoyo que pueden no requerirse para todos los sistemas. El subsistema I/O básico para el caso del sistema operativo Windows es el BIOS (*basic input/output system*) que, además del I/O básico, tiene cuidado de autocargar (*booting*) el sistema en el encendido inicial. Las rutinas BIOS, por lo general, se almacenan en ROM o memoria flash para evitar pérdidas en el evento de interrupción de energía.

Al agrupamiento antes mencionado de dispositivos I/O en un pequeño número de tipos genéricos se le conoce como abstracción I/O. En lo que sigue, se revisan cuatro de las abstracciones I/O más útiles que se encuentran usualmente en los sistemas operativos modernos:

Flujo de caracteres I/O.

Bloque de I/O.

Sockets de red.

Relojes o temporizadores.

La abstracción de flujo de caracteres I/O trata la entrada o salida como una corriente de caracteres. Un nuevo carácter de entrada se agrega al final de la corriente de entrada, mientras que otro nuevo carácter de salida se adelanta al dispositivo de salida después del carácter previo. Este tipo de comportamiento es capturado por las peticiones de función de sistema `get(•)` y `put(•)`, donde "`get`" obtiene un carácter de entrada y "`put`" envía un carácter a la salida. Con el uso de estos primitivos, es relativamente sencillo construir rutinas I/O que ingresen o saquen tiras de muchos caracteres. De hecho, al examinar las primeras ocho líneas de la tabla 7.2 se observa que el I/O orientado a carácter es el único tipo proporcionado en

MiniMIPS (y su simulador) al nivel del lenguaje ensamblador. El flujo de caracteres I/O es particularmente adecuado para dispositivos como teclados, ratones, módems, impresoras y tarjetas de audio.

La abstracción bloque de I/O es adecuada para discos y otros dispositivos orientados a bloque. En este sentido, tres peticiones básicas de sistema pueden capturar la esencia del bloque de I/O: `seek(•)`, `read(•)` y `write(•)`. La primera de éstas, *seek*, se necesita para especificar cuál bloque se debe transferir, mientras que *read* y *write* realizan la transferencia de datos I/O real hacia y desde memoria, respectivamente. Es posible construir un sistema de archivos mapeados a memoria en lo alto de tales primitivos I/O orientados por bloque. La idea es que la petición de sistema para I/O regrese la dirección virtual de los datos requeridos en lugar de iniciar la transferencia de datos. En consecuencia, los datos se cargan automáticamente en memoria con el primer acceso a ellos a través de la dirección virtual proporcionada. De esta forma, el acceso al sistema de archivos se vuelve muy eficiente porque sus transferencias de datos se manejan mediante el sistema de memoria virtual.

Los sockets de red son dispositivos I/O especiales que soportan comunicación de datos a través de redes de computadoras. Una petición de sistema I/O a este respecto puede crear un socket, hacer que un socket local se conecte a una dirección remota (un socket creado por otra aplicación), detectar aplicaciones remotas que solicitan ser conectados en un socket local, o enviar/recibir paquetes. Los servidores, que usualmente soportan muchos sockets, adicionalmente pueden requerir instalaciones para detección más eficiente de cuáles sockets tienen paquetes en espera de ser recibidos y cuáles tienen espacio para aceptar un nuevo paquete para transmisión.

Los relojes y temporizadores, que son componentes esenciales en la implementación de aplicaciones de control de tiempo real con microcontroladores, también se encuentran en procesadores de propósito general en vista de su utilidad para determinar el tiempo actual del día, medir tiempo transcurrido e impulsar eventos en tiempos preestablecidos. Un *temporizador de intervalo programable* constituye un mecanismo de hardware que se puede preestablecer a un espacio de tiempo deseado y generar una interrupción cuando dicha cantidad de tiempo haya transcurrido. El sistema operativo usa temporizadores de intervalo para programar con base en un cronograma tareas periódicas o asignar una *rebanada de tiempo* a una tarea en un ambiente multitarea. También puede permitir a los procesos de usuario utilizar esta facilidad. En el último caso, la solicitud del usuario para temporizadores de intervalo más allá del número disponible en hardware puede ser atendido al establecer temporizadores virtuales. La cabecera de mantener estos temporizadores virtuales es pequeña en comparación con los intervalos que se miden en aplicaciones típicas. Por ejemplo, un espacio de tiempo de 1 ms corresponde a 10^6 tics de reloj con un reloj de 1 GHz.

24.2 Interrupciones, excepciones y trampas

Ya se discutió (secciones 22.4 y 22.5) cómo se usan las interrupciones para liberar al CPU de la microgestión del proceso de transferencia de datos I/O. En ésta y en las dos secciones siguientes, se tratará acerca de cómo se implementan y manipulan las interrupciones.

Las interrupciones en una computadora se parecen mucho a las interrupciones de teléfono y correo electrónico durante el tiempo de trabajo o de estudio de una persona (figura 24.1). Cuando un teléfono suena, se hace a un lado lo que se está haciendo (marcar una página que se lee en un libro, golpear el botón guardar en la computadora, presionar el botón *mute* en la TV) y platicar con quien llama. Después de colgar se pueden escribir algunas notas acerca de lo que se necesita hacer como resultado de la llamada telefónica y regresar al trabajo principal. Los estudios demuestran que toma entre uno y dos minutos "recuperarse" de la interrupción y continuar trabajando a la misma tasa anterior a que el teléfono sonara. Algo similar se aplica cuando se tiene una alerta de correo electrónico, aunque, por razones desconocidas, acaso tengan que ver con la relativa dificultad de leer y escribir frente a hablar,

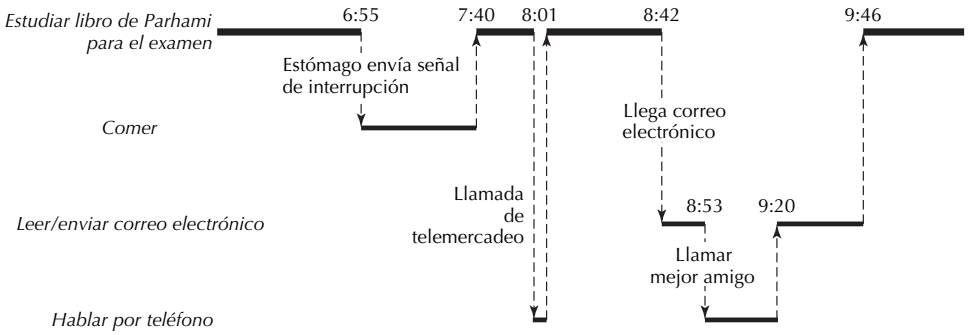


Figura 24.1 Las nociones de interrupciones e interrupciones anidadas.

el tiempo de recuperación es poco mayor en este caso. Se pueden evitar o deshabilitar completamente tales interrupciones si se desconecta el cordón del teléfono o se revisa el correo electrónico cada dos horas, en lugar de continuamente. Esta situación es similar a lo que hace el CPU cuando deshabilita las interrupciones o las restringe sólo a las de alta prioridad.

La *interrupción* es tanto el término general usado para un CPU que desvía su atención de la tarea actual que se ejecuta hacia algún evento inusual o impredecible que demanda su involucramiento (por cualquier razón) y el tipo específico de interrupción causado por entrada/salida y otras unidades de hardware. En este contexto, a veces se habla de *interrupciones de hardware*. El CPU se puede desviar de la tarea actual por causas en otras dos categorías:

Excepciones, que son causadas por operaciones ilegales, como dividir entre 0, intentar ingresar a localidad de memoria no existente o ejecutar instrucciones privilegiadas en modo usuario. Las excepciones son impredecibles por naturaleza y más bien infrecuentes.

Trampas, o interrupciones de software, que son peticiones deliberadas a sistema operativo cuando se necesitan servicios particulares. A diferencia de las interrupciones de hardware y las excepciones, las trampas están preplaneadas en el diseño de un programa y no son raras.

Gran parte de nuestra discusión de este capítulo trata de la manipulación de caminos de interrupción de hardware. Sin embargo, las excepciones y trampas son manipuladas de manera semejante en estas requeridas sin un lado de inicio de ejecución del programa y llamada a una rutina de software especial para tratar con la causa.

En el esquema más simple existe sólo una petición de la línea interrupción dentro del CPU. Múltiples dispositivos que requieren interrumpir el CPU comparten de esta línea. Manipular las interrupciones en esta mínima configuración se discute en la sección 24.3. De forma más general, existen múltiples líneas de solicitud de interrupción de varias prioridades. El siguiente ejemplo muestra por

Ejemplo 24.1: Niveles de prioridad de interrupciones en la computadora de un automóvil Suponga que en un automóvil se usa un microcontrolador que tiene cuatro niveles de prioridad para interrupciones. Las unidades que interrumpen al controlador incluyen un subsistema de detección de impacto, que necesita atención dentro de 0.1 ms, junto con otros cuatro subsistemas, a saber:

Subsistema	Tasa de interrupción	Máx tiempo atención (ms)
Combustible/encendido	500/s	1
Temperatura de motor	1/s	100
Pantalla de tablero	800/s	0.2
Acondicionador de aire	1/s	100

Discuta cómo se pueden asignar las prioridades para garantizar el tiempo de respuesta de 0.1 ms para el detector de impacto y manejar cada una de las otras unidades críticas antes de que se produzca la siguiente interrupción.

Solución: Al detector de impacto se le debe dar prioridad pues su tiempo de respuesta requerido es menor que el tiempo de atención para cada uno de los otros tipos de interrupción; ningún otro subsistema puede compartir este alto nivel de prioridad. Al acondicionador de aire se le puede dar menor prioridad porque su función no es crucial para la seguridad. Todavía quedan dos niveles de prioridad y tres subsistemas. Como consecuencia de que el monitor de temperatura del motor tiene un tiempo de atención máximo de 100 ms, su prioridad es baja, similar a la de los subsistemas de pantalla y combustible/encendido, que necesitan muchos cientos de periodos de servicios por segundo. Finalmente, los últimos pueden compartir el mismo nivel de prioridad, pues que cada uno se puede atender después del otro sin exceder el tiempo disponible antes de la siguiente interrupción. Por ejemplo, si la atención del subsistema combustible/encendido comienza justo antes de la interrupción de pantalla, transcurrirán 1.2 ms antes de que ambas rutinas de atención de interrupción se completen, mientras que hay 1/800 s = 1.25 ms disponibles antes de que llegue la siguiente interrupción de pantalla de tablero. Observe que se ignoraron el tiempo de atención para las interrupciones del subsistema de detección de impacto que tiene prioridad, porque ocurren muy rara vez; cuando lo hacen, lo demás carece de importancia.

qué puede ser útil tener múltiples niveles de prioridad para interrupciones. Las interrupciones anidadas y los conflictos en su manejo se discuten en la sección 24.4.

24.3 Manejo de interrupciones simples

En esta sección se supone una sola línea de interrupción que va hacia, y una señal de reconocimiento de interrupción producida por, el CPU. En la sección 24.4 se discutirán múltiples líneas de interrupción con niveles de prioridad jerárquicos y el anidado asociado de actividades de atención de interrupción.

Las señales de solicitud de interrupción de todas las unidades participantes están conectadas a la línea IntReq en el CPU (figura 24.2). Hay un flip-flop *interrupt mask* (máscara de interrupción) que el CPU puede fijar para deshabilitar interrupciones. Si IntReq se postula y las interrupciones no se enmascaran, se fija un flip-flop dentro del CPU para registrar la presencia de una solicitud de interrupción y para:

Reconocer la interrupción mediante la postulación de la señal IntAck.

Notificar a la lógica de siguiente dirección del CPU que está pendiente una interrupción.

Establecer la máscara de interrupción de modo que no se acepte ninguna nueva interrupción.

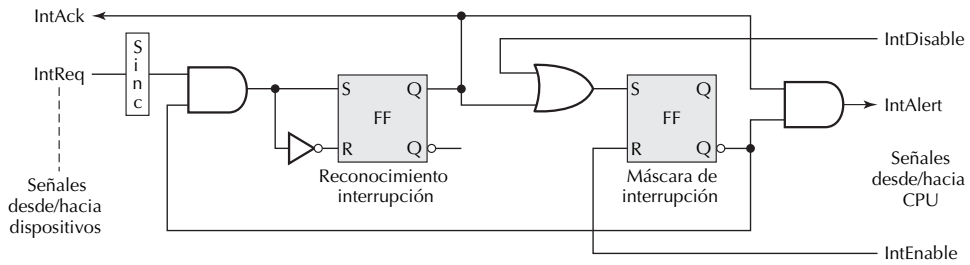


Figura 24.2 Lógica de interrupción simple para el MicroMIPS de ciclo sencillo.

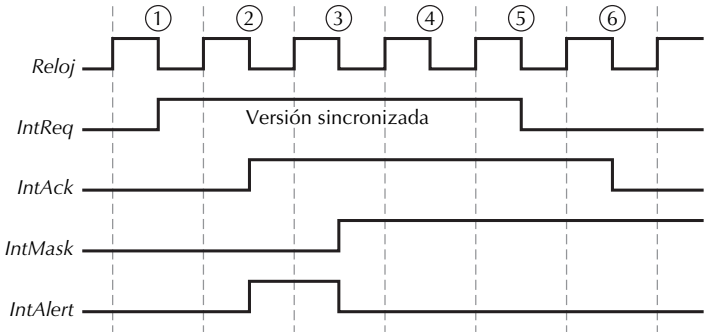


Figura 24.3 Temporización de las señales de solicitud de interrupción y de reconocimiento.

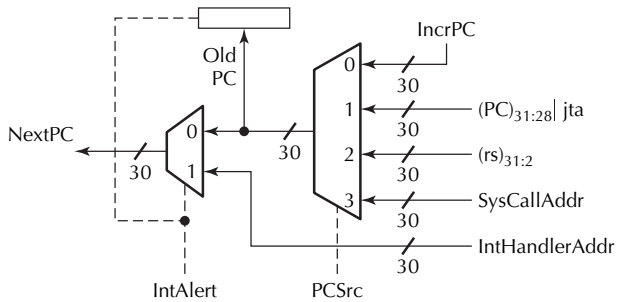


Figura 24.4 Parte de la lógica de siguiente dirección para MicroMIPS de ciclo sencillo, con capacidad de interrupción añadida (comparar la parte inferior izquierda de la figura 13.4).

Luego que los dispositivos notan la postulación de la señal IntAck, despostulan sus señales de solicitud, ello conduce al restablecimiento del flip-flop de reconocimiento de interrupción. Esta secuencia de eventos se muestra en la figura 24.3. Mientras tanto, después del reconocimiento de interrupción y antes de que se fije la máscara de interrupción, se produce una señal de alerta de interrupción (IntAlert) que ordena al CPU iniciar la ejecución de una rutina de interrupción. Lo anterior se hace al cargar la dirección de inicio del manipulador de interrupción en el PC y salvar el contenido previo del PC a usarse como la dirección de retorno cuando se atendió la interrupción.

Considere, por ejemplo, cómo se puede añadir tal capacidad de interrupción simple a la implementación MicroMIPS de ciclo sencillo del capítulo 13. La figura 24.4 muestra la parte modificada en la lógica de siguiente dirección de la figura 13.4 para permitir que la dirección de una rutina de manipulación de interrupción se cargue en PC siempre que se postule la señal IntAlert. Desde luego, el contenido de PC se debe salvar y usar como la dirección de retorno después de que la interrupción se atienda. Se puede suponer la inclusión de un registro especial que se cargue con el contenido antiguo (original) del PC siempre que se postule IntAlert. Entonces es necesario proporcionar una instrucción "retorno de interrupción" que coloque el contenido PC salvado en el PC y postule la señal IntEnable de modo que las nuevas interrupciones se acepten nuevo. Observe que no se puede usar el registro `$ra` para salvar la dirección de retorno de interrupción como se hizo para los procedimientos. La razón es que una interrupción puede ocurrir en cualquier momento, incluso dentro de un procedimiento que todavía no haya salvado (o no necesite salvar) su dirección de retorno en la pila.

Puesto que sólo hay una señal de interrupción en el CPU, la primera orden del negocio para el manipulador de interrupciones consiste en determinar la causa de la interrupción de modo que se puedan tomar acciones adecuadas. Una forma de hacer esto último es que el manipulador de interrupciones sondee todos los dispositivos I/O para ver cuál solicitó atención. El procedimiento para ello es similar al sondeo de I/O discutido en la sección 22.3. El sondeo puede ocurrir, y la atención se puede brindar, en el orden de prioridades de dispositivo. Una alternativa al sondeo es proporcionar la señal de recono-

cimiento de interrupción sólo al dispositivo de mayor prioridad y usar una cadena en margarita para pasar la señal a otros dispositivos en orden descendente de prioridades. Esto es similar al adelantamiento de la señal de garantía de bus en el ordenamiento de la figura 23.10. Un dispositivo que solicite atención puede enviar su identidad sobre el bus de dirección o de datos al recibir la señal de reconocimiento del CPU. Cualquier dispositivo de prioridad baja no verá la señal de reconocimiento de interrupción y continuará postulando su señal de solicitud de interrupción hasta que reciba un reconocimiento. Observe que el sondeo es más flexible que el encadenamiento en margarita en cuanto que permite que las prioridades se modifican fácilmente. La desventaja del sondeo es que desperdicia mucho tiempo interrogando muchos dispositivos I/O siempre que alguno de ellos genera una interrupción.

Luego que el CPU aprende la identidad del dispositivo (de mayor prioridad) que solicitó la interrupción, salta hacia el segmento apropiado de la rutina de atención de interrupción que maneja el tipo de solicitud particular. Este proceso se puede hacer más eficiente al permitir al dispositivo suministrar la dirección de inicio para su manipulador de interrupción deseado; de esta forma, la transición hacia dicho manipulador puede ocurrir inmediatamente. Un beneficio de este método, que se conoce como *interrupción vectorizada*, consiste en que un dispositivo puede proporcionar diferentes direcciones dependiendo de su tipo de solicitud; por tanto, evita interrogaciones o niveles adicionales de rodeos. En cualquier caso, la rutina de atención de interrupción debe salvar y, al terminar, restaurar cualquier registro que necesite usar en el curso de su ejecución. Esto es muy parecido a lo que ocurre en un procedimiento, excepto que aquí todos los registros tienen el mismo estatus en relación a salvar y restaurar (a un procedimiento ordinario se le permite usar algunos registros sin salvarlos; vea las convenciones de uso de registro MiniMIPS en la figura 5.2).

El ordenamiento anterior se puede extender a múltiples niveles de interrupciones con diferentes prioridades mediante el uso de un codificador de prioridad (figura 24.5). Cada línea de solicitud de interrupción a la izquierda representa un conjunto de dispositivos con prioridades idénticas. Si se reciben una o más solicitudes de interrupción, se postula la señal IntReq para notificar al CPU. Adicionalmente, al CPU se proporciona la identidad de la línea de solicitud de mayor prioridad postulada. De esta forma, o el CPU puede identificar únicamente al dispositivo interruptor (cuando una línea de solicitud representa un solo dispositivo) o de otro modo sondeará un subconjunto de todos los dispositivos. Esto es más eficiente que tener que sondear todos los dispositivos con cada interrupción. Si el CPU debe habilitar o deshabilitar cada tipo de interrupción por separado, se debe modificar el esquema que se muestra en la figura 24.5 para mover la capacidad de máscara de interrupción al lado de entrada del codificador de prioridad. Se deja a los lectores proporcionar los detalles de la modificación requerida.

La implementación MicroMIPS de ciclo sencillo considerada hasta el momento da seguimiento continuamente a la condición de interrupción y comienza a ejecutar la rutina de atención de interrupción en el ciclo de reloj siguiente a la postulación de IntAlert. Agregar capacidad de interrupción a la implementación MicroMIPS multiciclo del capítulo 14 se realiza en forma directa y no requiere ver más allá del reconocimiento del estado de control adecuado en el que se deben verificar las solicitudes de interrupción. Los detalles se dejan como ejercicio.

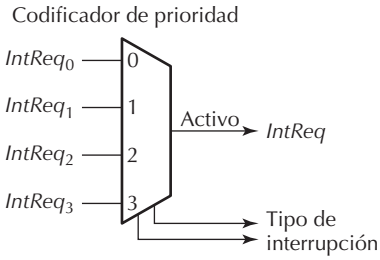


Figura 24.5 Uso de un codificador de prioridad para producir una señal de solicitud de interrupción sencilla y un tipo de interrupción a partir de múltiples solicitudes.

Sin embargo, agregar una capacidad de interrupción a los diseños MicroMIPS encauzados de los capítulos 15 y 16 requiere más cuidado. Los conflictos involucrados son idénticos a los de las excepciones, según se discutió en la sección 16.6. Para el caso de una *pipeline* lineal simple, similar a las de las figuras 15.9 y 15.11, se puede limpiar al permitir que todas las instrucciones, actualmente en varias etapas del encauce, corran hasta su conclusión; eso puede requerir que se inserten $q - 1$ burbujas en un encauce de q etapas. Por ejemplo, si una de las instrucciones de la *pipeline* encuentra un fallo de caché de datos, ocurrirá un largo atasco de *pipeline* mientras se accede a la memoria principal. Para el caso de memoria virtual, existe la posibilidad de una falla de página con su latencia todavía mayor, ello no puede ser admisible para las interrupciones que requieren un tiempo de respuesta corto. Una alternativa es anular todas las instrucciones parcialmente completadas que todavía no afectan memoria de datos o contenidos de registro, mientras se tiene cuidado de que, como resultado, no se introducen inconsistencias. Por ejemplo, en la ruta de datos encauzada de la figura 15.11, varias instrucciones afectan la caché de datos y los contenidos de registro en diferentes etapas de la *pipeline*. Si una instrucción ya tuvo un efecto irreversible, dicha instrucción, y todas las instrucciones por delante de ella en la *pipeline*, deben correr hasta su conclusión como consecuencia de evitar inconsistencias.

Con una *pipeline* bifurcada o conflicto de instrucción fuera de orden (figura 16.8), se puede o limpiar (*flush*) la *pipeline* al permitir que todas las instrucciones corran hasta su conclusión, o se descarten todas las instrucciones parcialmente ejecutadas cuyos resultados todavía no se hayan comprometido. Esto último requiere que el siguiente valor PC asociado con la última instrucción comprometida se mantenga en el retiro y comprometa parte de la *pipeline*. El último enfoque desperdicia algún trabajo que se realizó en las instrucciones no comprometidas, pero permite una reacción más rápida ante las interrupciones. Observe que, en un procesador moderno, docenas de instrucciones pueden estar en varias etapas de conclusión, y algunas de ellas encuentran retardos significativos como resultado de razones resaltadas en párrafos precedentes. Por tanto, en virtud de que la lógica de retiro y compromiso ya asegura la consistencia de todas las instrucciones completadas, se deben descartar todas las instrucciones no comprometidas.

24.4 Interrupciones anidadas

Una idea obvia para manipular múltiples dispositivos I/O de alta rapidez o aplicaciones de control sensibles al tiempo consiste en permitir el anidado de interrupciones de modo que una solicitud de interrupción de máxima prioridad pueda atribuirse (*preempt*) el programa que manipula una interrupción de baja prioridad. La ejecución de este último se resumiría cuando la interrupción de alta prioridad se haya atendido. Sin embargo, las interrupciones anidadas son más difíciles de organizar que las solicitudes de procedimiento anidado; lo último se manipula correctamente con la ayuda de una pila para almacenamiento de variables locales y salvamento de direcciones de retorno. Las dificultades al lidiar con interrupciones anidadas surgen de su temporización impredecible. Un programador que escribe un procedimiento que pide otro de éstos asegura que la segunda petición no se haga hasta después de que todos los pasos previos se hayan completado (figura 6.2). Algo similar no se sostiene para las interrupciones: una interrupción de alta prioridad puede ocurrir después de una prioridad menos importante, quizá después de que sólo una sola instrucción se haya ejecutado. Por tanto, es importante que interrupciones posteriores se deshabiliten hasta que el manipulador de interrupciones haya tenido oportunidad de salvar suficiente información para permitir que la manipulación de la interrupción de prioridad más baja se resuma luego que la prioridad alta se haya atendido.

La necesidad de las interrupciones anidadas es evidente a partir del ejemplo 24.1, se relaciona con las garantías de tiempo de respuesta de CPU requerido para ciertos tipos de interrupción. Si el esquema de interrupción de cuatro niveles que se muestra en la figura 24.5 se utilizase para la aplicación de con-

trol del ejemplo 24.1, una interrupción con cualquier nivel de prioridad tendría un tiempo de respuesta de 100 ms, pues la atención de una interrupción del acondicionador de aire pudo iniciarse al llegar la solicitud de interrupción de mayor prioridad. Esto último es inaceptable. Como otro ejemplo, un *drive* de disco que lee a 3 MB/s y transfiere datos al CPU en porciones de 4 B (ejemplo 22.7) requiere que su solicitud de interrupción se atienda dentro de 0.75 μs. Si hubiera algún tipo de interrupción cuya atención tarda más de 0.75 μs, y si durante la atención de una interrupción no se acepta otra similar, entonces el hecho de dar al controlador de disco la máxima prioridad no resolvería el problema; se requiere algún tipo de capacidad de atribución.

Considere ahora los pasos que sigue el CPU para atender una interrupción, como se enumeran en la sección 22.4. Si supone una implementación no encauzada, las siguientes acciones ocurren en ruta hacia, y durante, la atención de la interrupción.

1. Deshabilitar (enmascarar) todas las interrupciones.
2. Pedir la rutina de atención de interrupción; salvar el PC como en una petición de procedimiento.
3. Salvar el estado del CPU.
4. Salvar información mínima acerca de la interrupción en la pila.
5. Habilitar las interrupciones (o al menos las de máxima prioridad).
6. Identificar la causa de la interrupción y atender la solicitud subyacente.
7. Restaurar el estado del CPU a lo que existía antes de la última interrupción.
8. Regresar de la rutina de atención de interrupción.

Otra interrupción se puede aceptar después del paso que se indica en el paso 5. En consecuencia, mientras más rápido se ejecuten los primeros cinco pasos, más baja será la latencia de peor caso antes de que se acepte una interrupción de máxima prioridad. Por ejemplo, la interrupción de prioridad más alta debe esperar la ejecución de estos cinco pasos en el peor caso. Cualquier interrupción de prioridad más baja esperará el tiempo de atención del peor caso de todas las posibles interrupciones de máxima prioridad, incluidas las nuevas que puedan ocurrir en el curso de atención de alguna de tales interrupciones.

La figura 24.6 muestra la relación de un programa de aplicación con dos interrupciones anidadas. La primera interrupción (prioridad más baja) se detecta después de la conclusión de la instrucción *a*)

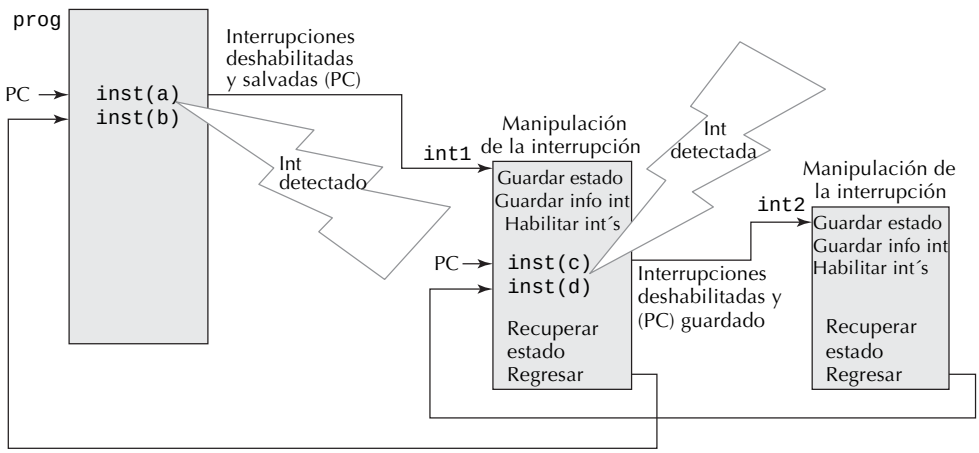


Figura 24.6 Ejemplo de interrupciones anidadas.

y antes de la instrucción *b*) inicia su ejecución dentro del programa de aplicación. A este punto, el PC contiene la dirección de la instrucción *b*), guardando el contenido que permitirá al hardware regresar al punto de la interrupción y continuar ejecutando la aplicación como si nada hubiera pasado entre las instrucciones *a*) y *b*). Todas las interrupciones se deshabilitan (enmascaran) inmediatamente y el control se transfiere automáticamente a un manipulador de interrupción. Luego que el manipulador de interrupción haya salvado el estado del CPU y alguna información mínima acerca de la interrupción, habilita todas las interrupciones de prioridad más alta antes de proceder a manipular la interrupción existente. Si poco después de este punto se detecta una segunda interrupción (prioridad más alta) entre las instrucciones *c*) y *d*), entonces se repite el proceso y se abandona la atención de la interrupción de prioridad más baja en favor de la recién llegada.

24.5 Tipos de conmutación contextual

A la transferencia de control de un programa que corre a un manipulador de interrupción, o desde el manipulador para una interrupción de prioridad baja hacia el de una interrupción de prioridad mayor, se le conoce como *conmutación contextual*. El contexto de un proceso de ejecución consiste del estado de CPU, que incluye contenidos de todos los registros, junto con ciertos elementos de información de gestión de memoria que permiten que el proceso accese a su espacio de dirección. Observe que pedir un procedimiento no se considera una conmutación de contexto porque el procedimiento generalmente opera dentro del mismo contexto que el programa que lo pide; por ejemplo, puede acceder a las variables globales del programa solicitante y pasarle los parámetros a través del archivo de registro o pila. Además de conmutaciones contextuales obligatorias debidas a interrupciones, el sistema operativo puede realizar conmutación contextual voluntaria debido a que se relaciona con el mejoramiento del rendimiento global del sistema. Este tipo de conmutación contextual se realiza dentro del marco de *multiprogramación* o *multitareas*, esos conceptos se explican en esta sección. Un término relacionado, *multihilos*, se explicará en la sección 24.6.

La ejecución de secuencias de instrucciones dentro de un programa se puede retardar por varias razones. Éstas incluyen tanto atascos cortos para eventos como fallos de caché o fallos TLB (capítulos 18 y 20), como esperas mucho más largas para fallas de página y solicitudes I/O. Hasta ahora, en este libro se vieron tales atascos y esperas como contribuyentes a reducir el rendimiento. Éste constituye un enfoque correcto en lo concerniente a la ejecución de una sola tarea secuencial. Sin embargo, así como el tiempo de una persona no se desperdicia realmente si ojea el periódico o lee mensajes de correo electrónico mientras cuelga el teléfono, los recursos de la computadora o del CPU tampoco se desperdician si hay otro programa o tarea en ejecución mientras una tarea encuentra un retardo prolongado.

El uso de multiprogramación para traslapar los periodos de espera de un programa con la ejecución de otros programas, comienza con la compartición de tiempo de los sistemas de cómputo de principios de la década de 1960. En un sistema que comparte tiempo, múltiples programas de usuario residen en la memoria principal de la computadora, ello permite al CPU cambiar entre ellos con relativa facilidad. Para asegurar justicia o parcialidad, a cada programa se le puede asignar una *rebanada de tiempo* fijo en el CPU, y la tarea se hace a un lado en favor de otra al final de su tiempo asignado, incluso si no encuentra esperas. El número de tareas simultáneamente activas en memoria principal representa el *grado de multiprogramación*. Es benéfico tener un alto grado de multiprogramación porque hace más probable que el CPU encuentre trabajo útil incluso en el evento de que mucha tareas encuentran esperas prolongadas. Por otra parte, poner muchas tareas en memoria principal significa que cada una obtiene una asignación de memoria más pequeña, que está en detrimento del rendimiento. La multitarea es esencialmente la misma idea que la multiprogramación, excepto que las tareas activas pueden pertenecer al mismo usuario o incluso pueden representar a diferentes partes (subcálculos) de un solo programa.

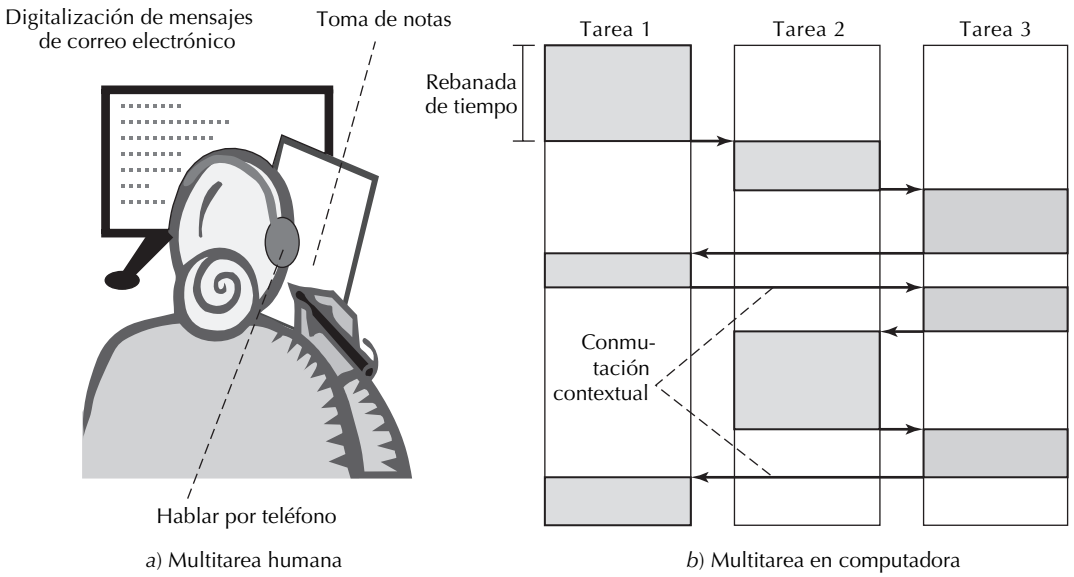


Figura 24.7 Multitarea en humanos y computadoras.

La figura 24.7 muestra multitarea en humanos y computadoras. Como se ve en la figura 24.7*b*), un CPU puede ejecutar muchas tareas al correr la atención de una a otra a través de la conmutación contextual. El CPU cambia de ida y vuelta entre diferentes tareas a un ritmo rápido que crea la ilusión de procesamiento paralelo, aun cuando, en un momento dado, sólo se ejecute una sola tarea. Para cambiar contexto, se salva el estado del proceso en ejecución en un bloque de control de proceso y se activa un nuevo proceso desde su principio o desde un punto salvado anteriormente. Salvar y restaurar docenas de registros y otra información de control con cada conmutación contextual toma muchos microsegundos que, en cierta forma, es tiempo desperdiciado. Sin embargo, los cientos o incluso miles de ciclos de reloj empleados en una conmutación contextual todavía son significativamente menores que muchos millones de ciclos de reloj requeridos para manipular una falla de página o una operación I/O.

En el contexto anterior, una invocación específica de un programa o una tarea con frecuencia se refieren como *proceso*. Observe que dos procesos distintos pueden ser diferentes invocaciones del mismo programa o tarea. En este sentido, el *multiprocesamiento* se podría usar como un término amplio para multiprogramación o multitarea. Sin embargo, en la literatura acerca de arquitectura de computadoras, el multiprocesamiento tiene un significado técnico que se relaciona con la ejecución concurrente de muchos procesos, en oposición a la aparente concurrencia apenas discutida.

Como consecuencia de que la cabecera de la conmutación contextual no es trivial, la conmutación contextual simple es práctica sólo para usar largos periodos de espera y no se puede usar para recuperar esperas más cortas debido a fallos de caché y similares. Además de la cabecera tangible para salvar y restaurar estados de proceso, la conmutación contextual involucra otros tipos de cabecera que son mucho más difíciles de cuantificar. Por ejemplo, compartir los cachés de instrucción y datos por muchos procesos puede conducir a conflictos y sus acompañantes fallos de caché para cada proceso. De igual modo, cuando los procesos no son completamente independientes uno de otro, la sincronización (para asegurar que las dependencias de datos se cumplen) desperdicia algún tiempo y otros recursos. El siguiente ejemplo captura los efectos de cabecera tangibles en conmutación contextual.

Ejemplo 24.2: Cabecera de interrupciones anidadas Suponga que cada vez que una interrupción de mayor prioridad atribuye una prioridad baja, se incurre en una cabecera de 20 μs para salvar y restaurar el estado de CPU y otra información pertinente. ¿Esta cabecera es aceptable para la aplicación de sistema de control presentada en el ejemplo 24.1?

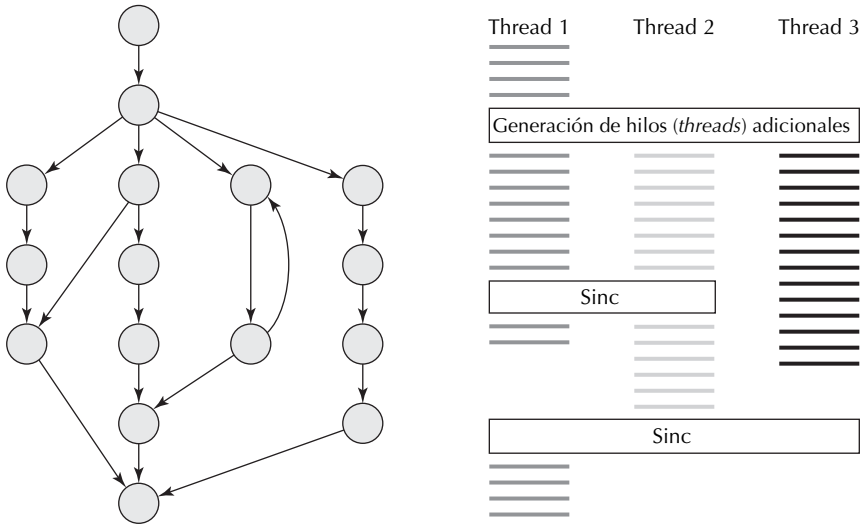
Solución: Si ignora la prioridad dada al detector de impacto, porque no genera una interrupción durante operación normal del sistema, existen tres niveles de interrupción. Se mencionan a continuación, con sus tasas de repetición y tiempos de servicio asociados: combustible/encendido (500/s, 1 ms) y pantalla de tablero (800/s, 0.2 ms) tienen la mayor prioridad, temperatura de motor (1/s, 100 ms) está en el nivel medio, y el acondicionador de aire (1/s, 100 ms) tiene la menor prioridad. Incluso, al suponer que la ejecución de los manipuladores de interrupción para temperatura de motor y acondicionamiento de aire se extienden sobre todo un segundo debido a atribuciones repetidas, cada cual se atribuye no más de 1 300 tiempos, para una cabecera total de $2 \times 1\,300 \times 20\ \mu s = 52$ ms. Si considera esta cabecera de atribución junto con el tiempo de atención de interrupción total de $500 \times 1 + 1 \times 100 + 800 \times 0.2 + 1 \times 100 = 860$ ms, se observa que todavía habrá un margen de seguridad igual a $1\,000 - 860 - 52 = 88$ ms en cada segundo de tiempo real. De este modo, la cabecera de conmutación contextual es aceptable en este ejemplo de aplicación.

Debido a la importancia de la conmutación contextual en la manipulación eficiente de interrupciones y multitarea, algunas arquitecturas proporcionan recursos de hardware para auxiliar a este propósito. Por ejemplo, suponga que un CPU contiene cuatro archivos de registro idénticos cada uno de los cuales se puede elegir como el archivo de registro activo al establecer una tag de control de dos bits. En consecuencia, la conmutación contextual entre hasta cuatro procesos activos se podría realizar sin salvar o restaurar alguno de los contenidos de registro, al ajustar la tag de control de dos bits. Este es un buen ejemplo de *conmutación contextual de cabecera baja*. Una alternativa menos drástica es proporcionar instrucciones de máquina especiales que salven o restauren todo el archivo de registro en una operación. Es probable que tales instrucciones sean más rápidas que una secuencia de instrucciones que tratan con salvar o restaurar los registros uno a la vez.

24.6 Hilos y multihilos

Como se representa en la figura 24.8, los *hilos* (*threads*) son corrientes de instrucciones (pequeños segmentos de programas) dentro de la misma tarea o programa que se pueden ejecutar concurrentemente con, y durante la mayor parte independientemente de, otros hilos. Como tal, proporcionan una fuente ideal de instrucciones independientes para llenar la *pipeline* de ejecución de CPU y hacer buen uso de las múltiples unidades de función, como las que se muestran en la figura 16.8. A los hilos a veces se les llama *procesos ligeros* porque comparten una gran cantidad y, por tanto, cambiar entre ellos no involucra tanta cabecera como los procesos convencionales (o pesados).

Las arquitecturas que pueden manipular muchos hilos se conocen como *superhiladas*, donde "súper" caracteriza el ancho del hilado en forma muy parecida a como *superpipelining* se nombra debido a la profundidad de la *pipeline*. El término *hiperhilado* también se usa para una forma más flexible de multihilo (simultáneo), pero aquí no se tratarán las diferencias sutiles de los dos esquemas. El multihilo divide efectivamente los recursos de un solo procesador en dos o más *procesadores lógicos*, donde cada procesador lógico ejecuta instrucciones desde un solo hilo. La figura 24.9 muestra cómo las instrucciones de tres hilos diferentes se pueden mezclar en las varias etapas de la *pipeline* de ejecución en un procesador de doble emisión.



a) Gráfica de tareas de un programa

b) Estructura de hilos de una tarea

Figura 24.8 Un programa dividido en tareas (subcálculos) o hilos.

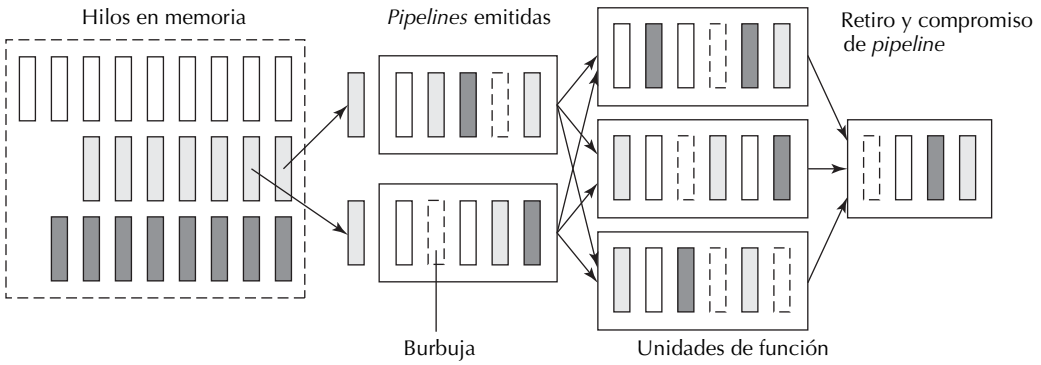


Figura 24.9 Instrucciones de múltiples hilos conforme avanzan en su ruta hacia una *pipeline* de ejecución de un procesador.

Observe que, incluso con multihilos, pueden aparecer algunas burbujas de *pipeline*. Sin embargo, es probable que el número de burbujas sea mucho menor cuado están disponibles para ejecución un gran número de hilos. Construir un programa en forma de multihilos es parecido a liberar al CPU en gran medida de la carga de descubrir paralelismo en el nivel de instrucción y permitir que múltiples instrucciones se emitan a partir de partes dispares de un programa en vez de estar confinadas a instrucciones cercanas en el flujo de control del programa.

PROBLEMAS

24.1 Uso de temporizadores de intervalo

Una computadora tiene un temporizador de intervalo que se puede fijar a una longitud de tiempo deseada (por decir, 0.1 s), en cuyo final se genera una interrupción.

a) Describa cómo usaría tal temporizador para construir un reloj análogo, con el uso de un motor de velocidad gradual para salida. El motor necesita controlar sólo la segunda manecilla del reloj, las manecillas de los minutos y las horas se mueven adecuadamente como resultado.

b) ¿Cuáles son las fuentes de imprecisión en el tiempo que muestra el reloj de la parte *a*)?

24.2 Nociones de manipulación de interrupciones

Considere las interrupciones al comer, en la llamada telefónica y al atender la alerta de correo electrónico que se muestran en la figura 24.1.

a) Escribe (en idioma llano) una rutina de manipulación de interrupción para estos eventos. Ponga especial atención a la posibilidad de anidar interrupciones. Establezca sus suposiciones.

b) Agregue dos niveles de interrupción más a los dos ya presentes en la figura. Describa las nuevas interrupciones en detalle y establezca por qué son de mayor prioridad que las existentes.

c) Discuta brevemente cómo estas interrupciones y sus manipulaciones son diferentes de las que están en la computadora.

24.3 Petición de procedimiento frente a interrupciones

Explique las diferencias entre una petición de procedimiento y la transferencia de control a un manipulador de interrupción. Por ejemplo, ¿por qué en el caso de una petición de procedimiento es adecuado cambiar el contenido del PC (es decir, se permite la conclusión en forma normal de las instrucciones adelante de la petición de procedimiento en la *pipeline*), mientras que para una interrupción ésta se debe limpiar?

24.4 Múltiples líneas de solicitud de interrupción

a) Muestre cómo, en la figura 24.5, cada tipo de interrupción se puede enmascarar separadamente.

b) Agregue lógica de reconocimiento de interrupción a la figura 24.5. Asegúrese de que, cuando se reconozca la interrupción de mayor prioridad, la postulación continua de una señal de solicitud de interrupción de prioridad más baja no conduce a reconocimiento falso.

24.5 Interrupciones en MicroMIPS multiciclo

La adición de capacidad de interrupción a la implementación MicroMIPS de ciclo sencillo del capítulo 13 se discutió en la sección 24.3. Discuta cómo se puede agregar la misma capacidad al MicroMIPS multiciclo del capítulo 14. *Sugerencia:* Piense en términos de la máquina de estado de control de la figura 14.4.

24.6 Interrupciones aninadas

Tres dispositivos D_1, D_2 y D_3 están conectados a un bus y usan I/O basado en interrupción. Discuta cada uno de los siguientes escenarios en dos formas, una al suponer el uso de una sola línea de solicitud de interrupción y otra al suponer el uso de dos líneas de solicitud de interrupción con prioridades altas y bajas. En cada caso, especifique cuándo y cómo las interrupciones se habilitan o deshabilitan.

a) Las interrupciones no estarán anidadas.

b) Las interrupciones para D_1 y D_2 no están anidadas en relación una con otra, pero las interrupciones desde D_3 se deben aceptar en cualquier momento.

c) Se permitirán tres niveles de anidado, con D_1 en el extremo bajo y D_3 en el extremo alto del espectro de prioridad.

24.7 Conceptos multitarea

Los humanos realizan gran cantidad de multitareas en la vida diaria. En la figura 24.7a se muestra un ejemplo.

a) ¿Qué características de las tareas se requieren para hacerlas sensibles a multitarea por humanos?

b) ¿Qué tipos de tarea no son adecuados para multitarea por humanos?

c) Relacione sus respuestas con las partes *a*) y *b*) para multitarea en una computadora.

d) Construye un escenario realista en el que un humano realice cinco o más tareas concurrentemente.

e) ¿Hay un límite sobre el número máximo de tareas concurrentes que se puedan realizar por un humano?
f) Relacione su respuesta a las partes *d*) y *e*) a la multitarea en una computadora.

24.8 Granularidad de la rebanada de tiempo en multiprogramación

Considere las siguientes suposiciones simplificadoras en un sistema de multiprogramación. Los tiempos de ejecución de tarea están distribuidos uniformemente en el rango [0, $T_{máx}$]. Cuando una rebanada de tiempo de duración τ se asigna a una tarea, se usa completamente y sin desperdicio (esto implica que cálculo e I/O están traslapados), excepto en la última rebanada de tiempo, cuando la tarea se completa; en promedio, la mitad de esta última rebanada de tiempo se desperdicia. La cabecera de tiempo de conmutación contextual es una constante c y no hay otra cabecera.

a) Determine el valor óptimo de τ para minimizar el desperdicio, si supone que $T_{máx}$ es mucho más grande que τ.
b) ¿Cómo cambia la respuesta a la parte *a*) si los tiempos de ejecución de tarea están distribuidos uniformemente en [$T_{mín}$, $T_{máx}$]? Establezca sus suposiciones.

24.9 Múltiples niveles de prioridad de interrupción

Siguiendo la introducción del ejemplo 24.1, defina un sistema en tiempo real con múltiples niveles de prioridad de interrupción, donde los requerimientos de tiempo de respuesta se pueden satisfacer sin necesidad de interrupciones anidadas. Luego, caracterice tales sistemas en general.

24.10 Auxiliares de hardware para conmutación contextual

a) ¿Cuál de las instrucciones "complejas" mencionadas en la tabla 8.1 facilitan la implementación de interrupciones anidadas y por qué?
b) Considere una arquitectura de conjunto de instrucciones de su elección y mencione todas sus instrucciones de máquina que se hayan proporcionado para hacer más eficiente la conmutación contextual.

24.11 Cabecera de conmutación contextual

En el ejemplo 24.2, ¿cuál es la cabecera de atribución máxima que sería aceptable?

24.12 Control de la frecuencia de sondeo

En el ejemplo 22.4 se afirmó que es necesario interrogar un teclado al menos diez veces por segundo para asegurarse de que no se pierda ninguna presión de tecla del usuario. Con base en lo que aprendió en cada capítulo, ¿cómo se puede asegurar de que el sondeo con frecuencia se hace suficiente, pero no con demasiada frecuencia para desperdiciar el tiempo?

24.13 Mecanismos de interrupción en procesadores reales

Para un microprocesador de su elección, describa el mecanismo de manejo de interrupción y su impacto en la ejecución de instrucción. Ponga especial atención a los siguientes aspectos de interrupciones:

a) Señalamiento y reconocimiento.
b) Habilitación y deshabilitación.
c) Prioridad y anidado.
d) Instrucciones especiales.
e) Hardware frente a software.

24.14 Niveles de prioridad de interrupción

Existen muchas similitudes entre la manipulación de múltiples niveles de prioridad de interrupción y arbitraje de bus, como se discutió en la sección 23.4. Para cada uno de los conceptos siguientes en interrupciones o arbitraje de bus, indique si hay o no una contraparte en la otra área. Justifique su respuesta en cada caso.

a) Prioridad rotatoria.
b) Encadenamiento margarita.
c) Arbitraje distribuido.
d) Enmascaramiento de interrupción.
e) Anidado de interrupción.

24.15 Interrupción de software

La instrucción `syscall` de MiniMIPS (sección 7.6) permite que un programa que corre se interrumpa a sí mismo y pase el control al sistema operativo. El microprocesador ARM tiene una instrucción similar `SWI` (interrupción de software), donde la atención que se so-

licita está especificada en los ocho bits de orden inferior de la instrucción misma en lugar de a través del contenido de un registro, como en MiniMIPS. Cada uno de los servicios proporcionados por el sistema operativo tienen una rutina asociada en ARM y las direcciones de inicio de estas rutinas se almacenan en una tabla.

a) Describa el mecanismo de interrupción de ARM e indique cómo `SWI` encaja en él.

b) Proporcione un panorama del conjunto de instrucciones de ARM, con enfoque en cualquier característica inusual o única.

c) Identifique las instrucciones ARM que el sistema operativo puede usar para realizar la tarea de transferir control a la rutina de sistema adecuada.

24.16 Circuito de interrupción multinivel

En este problema considere el diseño de un circuito de interrupción de prioridad multinivel externo al procesador. Este último tiene una sola solicitud de interrupción y una línea de reconocimiento de interrupción. El circuito a diseñar tiene un registro interno de tres bits que contiene el nivel de prioridad actual: un número de 0 a 7. Ocho líneas de solicitud de interrupción R_0-R_7 entran al circuito, y R_0 tiene la máxima prioridad. Cuando se postula una señal de solicitud de prioridad mayor que el nivel de tres bits almacenado, se envía al procesador una señal de solicitud de interrupción. Cuando el procesador reconoce la interrupción, el nivel de interrupción se actualiza inmediatamente y se envía una correspondiente señal de reconocimiento (la señal de reconocimiento A_i corresponde a la señal de solicitud R_i).

a) Presente el diseño del circuito de interrupción de prioridad externo según se describe.

b) Explique el funcionamiento de su circuito, y ponga especial atención a la forma en la que el registro de nivel de tres bits se actualiza para contener un valor más pequeño o más grande.

REFERENCIAS Y LECTURAS SUGERIDAS

[Egge97] Eggers, S. J., J. S. Emer, H. M. Levy, J. L. Lo, R. L. Stamm y D. M. Tullsen, "Simultaneous Multithreading: A Platform for Next Generation Processors", *IEEE Micro*, vol. 17, núm. 5, pp. 12-18, septiembre/octubre 1997.

[Heur04] Heuring, V. P., and H. F. Jordan, *Computer System Design and Architecture*, Prentice Hall, 2a. ed., 2004.

[Marr02] Marr, D. T., *et al*., "Hyper-Threading Technology Architecture and Microarchitecture", *Intel Technology J*., vol. 6, núm. 1, 14 de febrero de 2002. Disponible en: http://www.intel.com/technology/itj/

[Silb02] Silberschatz, A., P. B. Galvin y G. Gagne, *Operating System Concepts*, Wiley, 6a. ed., 2002.

[Ward90] Ward, S. A. y R. H. Halstead Jr, *Computation Structures*, MIT Press, 1990.