CAPÍTULO ONCE

Organización de entrada-salida

EN ESTE CAPÍTULO

11-1 Dispositivos periféricos

El subsistema de entrada-salida de una computadora, denominado E/S, proporciona un modo de comunicación eficiente entre el sistema central y el ambiente externo. Los programas y datos deben introducirse a la memoria de la computadora para su procesamiento y los resultados que se obtienen de los cálculos deben grabarse o registrarse para el usuario. Una computadora no tiene ningún propósito útil sin la capacidad de recibir información de una fuente externa y de transmitir los resultados de manera comprensible.

El medio más familiar de introducir información en una computadora es a través de un teclado tipo máquina de escribir, que permite a una persona introducir información alfanumérica en forma directa. Cada vez que se oprime una tecla, la terminal envía un carácter codificado en binario a la computadora. La velocidad más alta posible para introducir información de esta manera depende de la velocidad para teclear de una persona. Por otra parte, la unidad de procesamiento central es un dispositivo extremadamente rápido capaz de ejecutar operaciones a muy alta velocidad. Cuando se transfiere a un procesador información de entrada mediante un teclado lento,

E/S

el procesador estará ocioso la mayor parte del tiempo, mientras espera que llegue la información. Para utilizar una computadora de manera eficiente, deben prepararse con anticipación una gran cantidad de programas y datos, y deben transmitirse a un medio de almacenamiento como discos o cintas magnéticas. La información del disco se transfiere después a la memoria de la computadora a gran velocidad. Los resultados de los programas también se transfieren a un almacenamiento de alta velocidad, como discos, desde los cuales se pueden transferir, más tarde, a una impresora para proporcionar una salida impresa de los resultados.

Los dispositivos que están bajo el control directo de la computadora están conectados en línea. Estos dispositivos están diseñados para leer información hacia adentro o afuera de la unidad de memoria ante un comando de la CPU y se considera que son parte del sistema total de la computadora. Los dispositivos de entrada o salida conectados a la computadora también se llaman *periféricos*. Entre los periféricos más comunes están los teclados, los monitores y las impresoras. Los periféricos que proporcionan almacenamiento auxiliar para el sistema son cintas y discos magnéticos. Los periféricos son dispositivos electromecánicos y electromagnéticos de cierta complejidad. Aquí sólo se proporcionará un breve análisis de su funcionamiento, sin entrar en detalles de su construcción interna.

periférico

monitor y teclado

Los monitores de video son los periféricos de uso más común. Consisten en un teclado como dispositivo de entrada y una pantalla como dispositivo de salida. Hay diferentes tipos de monitores de video, pero los más populares utilizan un tubo de rayos catódicos (CRT). El CRT contiene un cañón electrónico que envía un haz de electrones a una pantalla fosforescente al frente del tubo. El haz puede desviarse en forma horizontal y vertical. Para producir un patrón en la pantalla, una rejilla dentro del CRT recibe un voltaje variable que hace que el haz entre en contacto con la pantalla y la haga brillar en puntos seleccionados. Las señales horizontales y verticales desvían el haz y lo hacen barrer por el tubo, haciendo que aparezca en la pantalla un patrón visual. Una característica de los monitores es un cursor que marca la posición en que se insertará el siguiente carácter en la pantalla. El cursor puede moverse a cualquier posición en la pantalla, sobre un carácter único, una palabra o cualquier línea. Las teclas de edición agregan o borran información con base en la posición del cursor. El monitor puede operar en forma de carácter único, de donde todos los caracteres introducidos en la pantalla a través del teclado se trasmiten a la computadora en forma simultánea. En el modo de bloque, el texto editado se almacena primero en una memoria local dentro de la terminal. El texto se transfiere a la computadora como un bloque de datos.

impresora

Las impresoras proporcionan un registro permanente, sobre el papel, de los datos o el texto de salida de la computadora. Hay tres tipos básicos de impresoras de caracteres: de margarita, de matriz de puntos y laser. La impresora de margarita contiene un disco con los caracteres colocados en la orilla de la circunferencia. Al imprimir un carácter, la rueda gira a la posición

apropiada y después un imán, al que se aplica una corriente, la oprime contra la cinta. La impresora de matriz de puntos contiene un conjunto de puntos a lo largo del mecanismo de impresión. Por ejemplo, una impresora de matriz de 5 × 7 puntos que imprime 80 caracteres por línea tiene varias líneas horizontales y cada una consta de 5 × 80 = 400 puntos. Cada punto puede imprimirse o no, dependiendo de los caracteres específicos que están impresos en la línea. La impresora laser utiliza un tambor fotográfico rotatorio que se emplea para imprimir las imágenes de caracteres. Después, el patrón se transfiere sobre el papel igual que en una máquina copiadora.

cinta magnética

Las cintas magnéticas se utilizan principalmente para almacenar archivos de datos: por ejemplo, el registro de la nómina de una compañía. El acceso es secuencial y consta de registros que pueden accesarse uno después de otro, conforme la cinta se mueve a lo largo de un mecanismo estacionario de lectura-escritura. Es uno de los métodos más baratos y lentos para almacenar y tiene la ventaja de que las cintas pueden quitarse cuando no se usan. Los discos magnéticos tienen superficies rotatorias de alta velocidad, con una cubierta de material magnético. El acceso se consigue al mover un mecanismo de lectura-escritura sobre una pista en la superficie magnetizada. Sobre todo, los discos se utilizan para el almacenamiento de grandes cantidades de programas y datos. Las cintas y discos se analizan más adelante en la sección 12-1, junto con su papel como memoria auxiliar.

disco magnético

Otros dispositivos de entrada y salida que se encuentran en sistemas de computadora son los graficadores digitales, los lectores de caracteres ópticos y magnéticos, los convertidores analógicos-digitales y equipo diverso de adquisición de datos. No todas las entradas provienen de personas y no todas las salidas están dirigidas a ellas. Las computadoras se utilizan para controlar varios procesos en tiempo real, como provisión de partes de máquinas, procedimientos de ensamblado en línea y procesos químicos e industriales. Para tales aplicaciones, debe proporcionarse un método para captar las condiciones de estado del proceso y enviar señales de control al proceso que se controle.

La organización de entrada-salida de una computadora es una función del tamaño de la computadora y de los dispositivos conectados a ella. La diferencia entre un sistema grande y pequeño depende en gran parte de la cantidad de circuitería que tiene disponible la computadora para comunicarse con unidades periféricas y la cantidad de periféricos conectados al sistema. Como cada periférico se comporta en forma diferente de los demás, sería prohibitivo entrar en detalles de las interconexiones necesarias entre la computadora y cada periférico. En este capítulo se presentan ciertas técnicas comunes a la mayoría de los periféricos.

Caracteres alfanuméricos ASCII

Los dispositivos de entrada y salida que comunican con las personas y la computadora, por lo general se relacionan con la transferencia de informa-

ASCII

ción alfanumérica hacia y desde el dispositivo y la computadora. El código binario estándar para los caracteres alfanuméricos es el ASCII por sus siglas en inglés (*American Standard Code for Information Interchange*, Código Estándar

TABLA 11-1 Código estándar norteamericano para intercambio de información (American Standard Code for Information Interchange, ASCII)

	$b_7 b_6 b_5$							
$b_4 b_3 b_2 b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

Caracteres de control

NUL	Nulo	DLE	Escape de enlace de dato
SOH	Comienzo de encabezado	DC1	Control de dispositivo 1
STX	Comienzo de texto	DC2	Control de dispositivo 2
ETX	Fin de texto	DC3	Control de dispositivo 3
EOT	Fin de trasmisión	DC4	Control de dispositivo 4
ENQ	Consulta	NAK	Reconocimiento negativo
ACK	Reconocimiento	SYN	Inactivo síncrono
BEL	Campana	ETB	Fin de bloque de transmi
BS	Retroceso	CAN	Cancelar
HT	Tabulador horizontal	EM	Fin de medio
LF	Alimentación de línea	SUB	Sustituto
VT	Tabulador vertical	ESC	Escape
FF	Alimentación de forma	FS	Separador de archivo
CR	Retorno de carro	GS	Separador de grupo
SO	Tecla de mayúsculas oprimida	RS	Separador de registro
SI	Tecla de mayúsculas sin oprimir	US	Separador de unidad
SP	Espacio	DEL	Borrar

byte

dar Norteamericano para Intercambio de Información). Utiliza siete bits para codificar 128 caracteres como se muestra en la tabla 11-1. Los siete bits del código se representan mediante b_1 a b_7, donde b_7 es el bit más significativo. Por ejemplo, la letra A se representa en ASCII como 1000001 (columna 100, renglón 0001). El código ASCII contiene 94 caracteres que pueden imprimirse y 34 caracteres que no se imprimen y que se utilizan para diversas funciones de control. Los caracteres imprimibles consisten en 26 letras mayúsculas de la A a la Z; 26 letras minúsculas; los 10 números del 0 al 9, y 32 caracteres especiales como %, *, y $.

Los 34 caracteres de control se representan en la tabla ASCII con nombres abreviados. Se listan de nuevo debajo de la tabla con sus nombres funcionales. Los caracteres de control se utilizan para direccionar datos y formar el texto impreso en un formato preestablecido. Existen tres tipos de caracteres de control: afectadores de formato, separadores de información y caracteres de control de información. Los afectadores de formato son caracteres que controlan la distribución de la impresión. Incluye los controles de máquina de escribir común, como retroceso (backspace, BS), tabulación horizontal (*horizontal tabulation*, HT) y retorno de carro (*carriage return*, CR). Los separadores de información se utilizan para separar los datos en divisiones como párrafos y páginas. Incluyen caracteres como separador de registro (*record separator*, RS) y separador de archivo (*file separator*, FS). Los caracteres de control de comunicación son útiles durante la transmisión de texto entre terminales remotas. Algunos ejemplos de caracteres de control son inicio de texto (*start of text*, STX) y fin de texto (*end of text*, ETX), los cuales se utilizan para enmarcar un mensaje de texto cuando se transmite por un medio de comunicación.

El ASCII es un código de 7 bits, pero la mayoría de las computadoras manipula una cantidad de 8 bits como una unidad única llamada *byte*. Por lo tanto, con mucha frecuencia, los caracteres ASCII se almacenan uno por byte. El bit extra en ocasiones se utiliza para otros propósitos, dependiendo de la aplicación. Por ejemplo, algunas impresoras reconocen los caracteres ASCII de 8 bits cuando el bit más significativo se desactiva en 0. Los 128 caracteres adicionales de 8 bits con el bit más significativo activado en 1 se utilizan para otros símbolos, como el alfabeto griego o una fuente de tipo cursivo. Cuando se utiliza en la comunicación de datos, puede emplearse el octavo bit para indicar la paridad del carácter del código binario

11-2 Interface de entrada-salida

La interface de entrada-salida proporciona un método para transferir información entre dispositivos de almacenamiento interno y de E/S externas. Los periféricos conectados a una computadora necesitan enlace de comunicación especial para funcionar como una interface con la unidad de procesamiento central. El propósito del enlace de comunicación es resolver las diferencias

que existen entre la computadora central y cada periférico. Las diferencias principales son:

1. Los periféricos son dispositivos electromecánicos y electromagnéticos y su manera de operación es diferente a la de la CPU y la memoria, que son dispositivos electrónicos. Por lo tanto, puede requerirse una conversión de valores de señales.
2. La velocidad de transferencia de datos de los periféricos, por lo general, es menor que la velocidad de transferencia de la CPU y, en consecuencia, puede necesitarse un mecanismo de sincronización.
3. Los códigos de datos y los formatos en los periféricos son diferentes del formato de la palabra en la CPU y en la memoria.
4. Los modos de operación de los periféricos son diferentes uno de otro y cada uno debe estar controlado para no perturbar la operación de otros periféricos conectados a la CPU.

Para resolver estas diferencias, los sistemas de computadora incluyen componentes de circuitería especiales entre la CPU y los periféricos para supervisar y sincronizar todas las transferencias de entrada y salida. Estos componentes se llaman *interfaces*, porque se comunican tanto con el canal del procesador como con el dispositivo periférico.

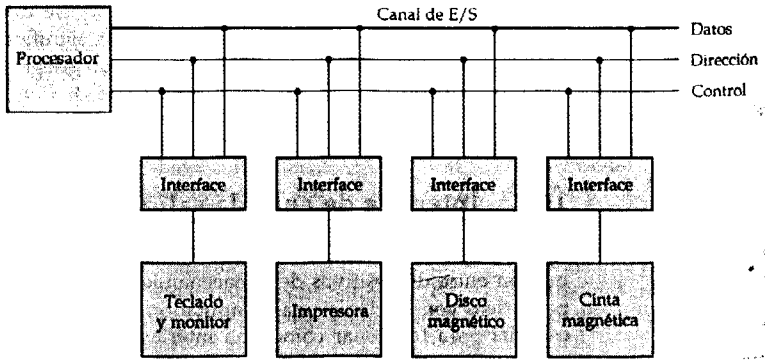
interface

Además, cada dispositivo puede tener su propio controlador que supervisa las operaciones del mecanismo particular en el periférico.

Canal de E/S y módulos de interface

Un enlace de comunicación típico entre el procesador y varios periféricos se muestra en la figura 11-1. El canal de E/S, consta de líneas de datos, líneas

Figura 11-1 Conexión de canal de E/S a dispositivos de entrada-salida.



de dirección y líneas de control. Se emplean el disco magnético, la impresora y la terminal en casi cualquier computadora de propósito general. La cinta magnética se utiliza en algunas computadoras para el almacenamiento de respaldos. Cada dispositivo periférico tiene asociada una unidad de interface. Cada interface decodifica la dirección y el control que se recibe del canal de E/S, y las interpreta para el periférico y proporciona señales para el controlador del periférico. También sincroniza el flujo de datos y supervisa la transferencia entre el periférico y el procesador. Cada periférico tiene su propio controlador que opera el dispositivo electromecánico particular. Por ejemplo, el controlador de impresora se hace cargo del movimiento de papel, la temporización de la impresión y la selección de los caracteres que se imprimen. Un controlador puede alojarse en forma separada o puede integrarse físicamente con el periférico.

El canal de E/S del procesador se conecta a todas las interfaces del periférico. Para comunicarse con un dispositivo particular, el procesador coloca una dirección de dispositivo en las líneas de direccionamiento. Cada línea conectada al canal de E/S contiene un decodificador de dirección que monitorea las líneas de direccionamiento. Cuando la interface detecta su propia dirección, activa la trayectoria entre las líneas del canal y el dispositivo que controla. Todos los periféricos cuyas direcciones no corresponden a la dirección en el canal, son inhabilitados por su interface.

Al mismo tiempo que queda disponible la dirección en las líneas de direccionamiento, el procesador proporciona un código de función en las líneas de control. La interface seleccionada responde al código de función y avanza a ejecutarlo. El código de función se denomina un comando de E/S y es, en esencia, una instrucción que se ejecuta en la interface y está conectada a la unidad periférica. La interpretación del comando depende del periférico que direcciona el procesador. Una interface puede recibir cuatro tipos de comandos. Se clasifican como control, estado, salida de datos y entrada de datos.

comando de E/S

Se emite un *comando de control* para activar el periférico e informarle qué hacer. Por ejemplo, una unidad de cinta magnética puede instruirse para que haga regresar la cinta un registro, para que se rebobine o para que empiece a moverse hacia adelante. El comando de control particular que se emite depende del periférico y cada periférico recibe su propia secuencia distinta de comandos de control, dependiendo de su modo de operación.

comando de control

Se utiliza un *comando de estado* para probar diversas condiciones de estado en la interface y un periférico. Por ejemplo, es posible que la computadora quiera comprobar el estado del periférico antes de que se inicie una transferencia. Durante la transferencia, pueden ocurrir uno o más errores que detecta la interface. Estos errores se representan al activar bits en un registro de estado que el procesador pueda leer en ciertos intervalos.

estado

Un *comando de salida de datos* hace que la interface responda transfiriendo datos del canal a uno de sus registros. Consideremos un ejemplo con una unidad de cinta. La computadora comienza a mover la cinta al emitir

datos de salida

un comando de control. Después, el procesador monitorea el estado de la cinta mediante un comando de estado. Cuando la cinta está en la posición correcta, el procesador envía un comando de salida de datos. La interface responde a la dirección y al comando y transfiere la información de las líneas de datos del canal a su registro intermedio (buffer). En seguida, la interface comunica con el controlador de la cinta y envía los datos que se van a almacenar.

datos de entrada

El *comando de entrada de datos* es lo opuesto al de salida de datos. En este caso, la interface recibe datos del periférico y los coloca en su registro intermedio. El procesador verifica si los datos están disponibles mediante un comando de estado y después envía un comando de entrada de datos. La interface coloca los datos sobre las líneas de datos, donde el procesador los acepta.

E/S versus canal de memoria

Además de comunicarse con su espacio de E/S, el procesador debe comunicarse con la unidad de memoria. Como el canal de E/S, el canal de memoria contiene datos, direcciones y líneas de control de lectura/escritura. Existen tres maneras que pueden utilizar los canales de la computadora para comunicarse con la memoria y las E/S:

1. Utilizar dos canales separados, uno para la memoria y el otro para las E/S.
2. Utilizar un canal común para memoria y E/S pero tener líneas de control separadas para cada una.
3. Utilizar un canal común para memoria y E/S con líneas de control comunes.

En el primer método, la computadora tiene conjuntos de canales de datos, de control y de direcciones independientes, uno para accesar la memoria y el otro para las E/S. Esto se hace en computadoras que proporcionan un procesador de E/S separado (*I/O processor, IOP*) además de la unidad de procesamiento central (CPU). La memoria se comunica con la CPU y el IOP por medio de un canal de memoria. El IOP se comunica también con los dispositivos de entrada y salida mediante un canal de E/S separado, con sus direcciones, datos y líneas de control. El propósito del IOP es proporcionar una trayectoria independiente para la transferencia de información entre dispositivos externos y la memoria interna. El procesador de E/S en algunas ocasiones se denomina *canal de datos*. En la sección 11-7 analizamos con mayor detalle la función del IOP.

IOP

E/S aislada versus E/S mapeada en memoria

Muchas computadoras utilizan un canal común para transferir información entre la memoria o la E/S y la CPU. La diferencia entre una transferencia

de memoria y una transferencia de E/S se reconoce mediante líneas de lectura y escritura separadas. La CPU especifica si la dirección en las líneas de dirección es para una palabra de memoria o para un registro de interface al habilitar una o dos líneas posibles de lectura o escritura. Las líneas de control *E/S lectura* y *E/S escritura* se habilitan durante una transferencia de E/S. Las líneas de control de *lectura de memoria* y *escritura de memoria* se habilitan durante una transferencia a memoria. Esta configuración aisla todas las direcciones de interface de E/S de las direcciones asignadas a memoria y se denomina *método de E/S aislada* para asignar direcciones en un canal común.

E/S aislada

En la configuración *E/S aislada*, la CPU tiene instrucciones distintas de entrada y salida, y cada una de estas instrucciones se asocia con la dirección de un registro de interface. Cuando la CPU recupera y decodifica el código de operación de una instrucción de entrada-salida, coloca la dirección asociada con la instrucción dentro de las líneas de dirección comunes. Al mismo tiempo, habilita la línea de control de lectura de E/S (para entrada) o de escritura de E/S (para salida). Esto informa a los componentes externos conectados al canal común que la dirección en las líneas de dirección es para un registro de interface y no para una palabra de memoria. Por otra parte, cuando la CPU recupera una instrucción o un operando de la memoria, coloca la dirección de memoria en las líneas de dirección y habilita la línea de control de lectura de memoria o de escritura de memoria. Esto informa a los componentes externos que la dirección es para una palabra de memoria y no para una interface de E/S.

El método E/S aislada separa la memoria y las direcciones de E/S para que los valores de la dirección de memoria no se afecten con la asignación de direcciones de interface, porque cada uno tiene su propio espacio de direccionamiento. La otra alternativa es utilizar el mismo espacio de direccionamiento para memoria y E/S. Este es el caso de las computadoras que emplean sólo un conjunto de señales de lectura y escritura y no hacen diferencia entre direcciones de memoria y E/S. Esta configuración se denomina *E/S mapeada en la memoria*. La computadora trata a un registro de interface como parte del sistema de memoria. Las direcciones asignadas para registros de interface no pueden utilizarse para palabras de memoria, lo cual reduce el rango de direcciones de memoria disponible.

mapeada en la memoria

En una organización de E/S mapeada en la memoria, no hay instrucciones específicas de entrada o salida. La CPU puede manipular datos de E/S que residen en registros de interface con la misma instrucción que se utiliza para manipular palabras de memoria. Cada interface se organiza como un conjunto de registros que responden a peticiones de lectura y escritura en el espacio de direccionamiento normal. De manera típica, se reserva un segmento del espacio de direccionamiento total para registros de interface pero, en general, se pueden colocar en cualquier dirección mientras no existe también una palabra de memoria que responda a la misma dirección.

Las computadoras con E/S mapeada en memoria pueden utilizar instrucciones de tipo memoria para accesar datos de E/S. Esto permite a la

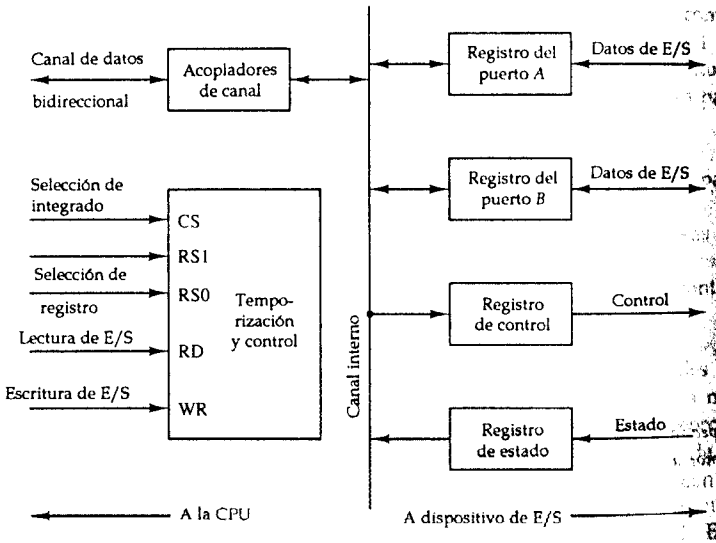
computadora utilizar las mismas instrucciones para transferencia de entrada-salida o para transferencia de memoria. La ventaja es que las instrucciones de carga y almacenamiento utilizadas para leer y escribir en la memoria pueden utilizarse para introducir y sacar datos de los registros de E/S. En una computadora típica, hay más instrucciones de referencia a memoria que instrucciones de E/S. Con las E/S mapeadas en memoria todas las instrucciones que hacen referencia a memoria también están disponibles para E/S.

Ejemplo de interface E/S

puerto de E/S

Un ejemplo de una interface de E/S se muestra en forma de diagrama de bloques en la figura 11-2. Consta de dos registros de datos llamados *puertos*,

Figura 11-2 Ejemplo de unidad de interface de E/S.



CS	RS1	RS0	Registro seleccionado
0	×	×	Ninguno: canal de datos en alta impedancia
1	0	0	Registro del puerto *A*
1	0	1	Registro del puerto *B*
1	1	0	Registro de control
1	1	1	Registro de estado

un registro de control, un registro de estado, acopladores de canal y circuitos de temporización de control. La interface se comunica con la CPU mediante el canal de datos. Las entradas de selección de integrado y de selección de registro determinan la dirección asignada a la interface. Lectura de E/S y escritura de E/S son dos líneas de control que especifican una entrada o salida, respectivamente. Los cuatro registros comunican en forma directa con un dispositivo de E/S conectado a la interface. Los datos de E/S hacia y desde el dispositivo pueden transferirse al puerto *A* o al puerto *B*. La interface puede operar con un dispositivo de salida, con un dispositivo de entrada o con un dispositivo que requiere tanto entradas como salidas. Si la interface está conectada a una impresora, sólo sacará datos, y si da servicio a un lector de caracteres, sólo introducirá datos. Una unidad de disco magnético transfiere datos en ambas direcciones pero no al mismo tiempo, por lo que la interface puede utilizar líneas bidireccionales. Se pasa un comando al dispositivo de E/S al enviar una palabra al registro apropiado de la interface. En un sistema como este, no se necesita el código de función en el canal de E/S, porque el comando se envía al registro de control, la información de estado se recibe del registro de estado y los datos se transfieren hacia los registros de los puertos *A* y *B*. Por lo tanto, la transferencia de datos, el control y la información de estado se realizan siempre mediante el canal de datos común. La diferencia entre datos, control o información de estado se determina del registro particular de la interface con el que se comunica la CPU.

El registro de control recibe información de control de la CPU. Al cargar los bits apropiados dentro del registro de control, la interface y el dispositivo de E/S conectados a ella pueden colocarse en diversos modos de operación. Por ejemplo, el puerto *A* puede definirse como un puerto de entrada y el puerto *B* como un puerto de salida. Pueden darse instrucciones a una unidad de cinta magnética para que rebobine la cinta o para que se arranque con un movimiento hacia adelante. Los bits en el registro de estado se utilizan para condiciones de estado y para registrar errores que pueden ocurrir durante la transferencia de datos. Por ejemplo, un bit de estado puede indicar que el puerto *A* ha recibido un nuevo conjunto de datos del dispositivo de E/S. Otro bit del registro de estado puede indicar que ha ocurrido un error de paridad durante la transferencia.

Los registros de la interface comunican con la CPU por medio del canal de datos bidireccional. El canal de direcciones selecciona la unidad de interface por medio de la entrada de selección de integrado y de las dos entradas de selección de registros. En forma externa, debe proporcionarse un circuito (por lo general un decodificador) para detectar la dirección asignada a los registros de la interface. El circuito habilita la entrada de selección de integrado (*chip select*, CS) cuando se selecciona la interface mediante el canal de direcciones. Las dos entradas de selección de registro *RS1* y *RS0*, por lo general, se conectan a las dos líneas menos significativas del canal de direcciones. Estas dos entradas seleccionan uno de los cuatro registros en la

interface, según se especifica en la tabla que acompaña al diagrama. El contenido del registro seleccionado se transfiere a la CPU mediante el canal de datos cuando se habilita la señal de lectura de E/S. La CPU transfiere información binaria al registro seleccionado mediante el canal de datos cuando la habilita la entrada E/S escritura.

11-3 Transferencia asíncrona de datos

Las operaciones internas en un sistema digital se sincronizan mediante pulsos de reloj proporcionados por un generador de pulsos común. Los pulsos de reloj se aplican a todos los registros dentro de una unidad y todas las transferencias de datos entre registros internos ocurren en forma simultánea durante el transcurso de un pulso de reloj. Se designan dos unidades, por ejemplo la CPU y la interface de E/S, independientes una de la otra. Si los registros de la interface comparten un reloj común con los registros de la CPU, se dice que la transferencia entre las dos unidades es síncrona. En la mayoría de los casos, la temporización interna en cada unidad es independiente de la otra, que en ese caso utiliza su propio reloj para sus registros internos. En ese caso, se dice que las dos unidades son asíncronas una de la otra. Este enfoque se utiliza con mucha frecuencia en la mayoría de los sistemas de computadora.

La transferencia de datos asíncrona entre dos unidades independientes requiere que se transmitan señales de control entre las unidades que se comunican para indicar el momento en el cual se están transmitiendo datos.

pulso de habilitación

Una manera de conseguir esto es mediante un pulso de habilitación (*estroboscópico*), proporcionado por medio de una de las unidades para indicar a la otra unidad cuándo tiene que ocurrir la transferencia. Otro método muy generalizado es acompañar cada conjunto de datos que se transfiere con una señal de control que indica la presencia de datos en el canal. La unidad que recibe el conjunto de datos responde con otra señal de control para reconocer la recepción de los datos. Este tipo de acuerdo entre dos unidades independientes se conoce como *reconocimiento mutuo (handshaking)*.

reconocimiento mutuo

El método de pulso estroboscópico y el método de reconocimiento de transferencia de datos asíncronos no se limitan a las transferencias de E/S. De hecho, se utilizan ampliamente en numerosas ocasiones que requieren la transferencia de datos entre dos unidades independientes. En general, consideramos la unidad que transmite como la fuente, y la unidad que recibe como el destino. Por ejemplo, la CPU es la unidad fuente durante una transferencia de salida o de escritura y es la unidad destino durante una transferencia de entrada o de lectura. Se acostumbra especificar la transferencia asíncrona entre dos unidades independientes mediante un diagrama de temporización que muestra la relación de temporización que debe existir entre las señales de control y los datos en los canales. La secuencia de control durante una transferencia asíncrona depende de si la transferencia la inicia la unidad fuente o la unidad destino.

diagrama de temporización

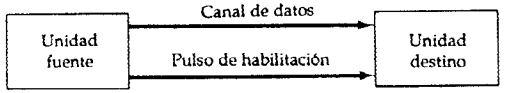
Control de habilitación

El método de control de habilitación (estroboscópico) de transferencia asíncrona de datos emplea una línea de control única para temporizar cada transferencia. La unidad fuente o la unidad destino pueden activar el pulso de habilitación. La figura 11-3(a) muestra una transferencia iniciada por la fuente. El canal de datos lleva la información binaria de la unidad fuente a la unidad destino. De manera típica, el canal tiene líneas múltiples para transferir un byte o palabra completos. El estroboscopio es una línea única que informa a la unidad destino cuándo está disponible en el canal una palabra de datos válida.

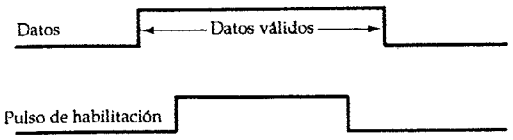
Como se muestra en el diagrama de temporización de la figura 11-3(b), primero la unidad fuente coloca los datos en el canal de datos. Después de un breve retraso, para asegurar que los datos se establecen en un valor regular, la fuente activa el pulso de habilitación. La información en el canal de datos y la señal de habilitación se quedan en estado activo durante un tiempo suficiente para permitir que la unidad destino reciba los datos. Con frecuencia la unidad destino utiliza un flanco descendente del pulso de habilitación para transmitir el contenido del canal de datos a sus registros internos. La fuente quita los datos del canal en un breve período previo a la deshabilitación del pulso de habilitación. En realidad, la fuente no tiene que cambiar la información en el canal de datos, el hecho de que la señal de habilitación esté deshabilitada indica que el canal de datos no contiene datos válidos. Habrá disponibles nuevos datos válidos sólo después de que se vuelva a habilitar el pulso de control de habilitación.

La figura 11-4 muestra una transferencia de datos iniciada por la unidad destino. En este caso, la unidad destino activa el pulso de habilitación, informando a la fuente que proporcione los datos. La unidad fuente

Figura 11-3 Pulso de habilitación iniciado por fuente para transferencia de datos.



a) Diagrama de bloque



b) Diagrama de temporización

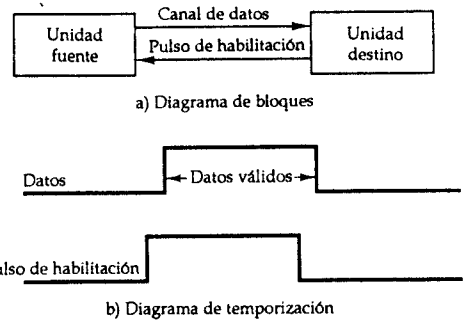


Figura 11-4 Pulso de habilitación iniciado por destino para transferencia de datos.

responde colocando la información binaria solicitada en el canal de datos. Los datos deben ser válidos y permanecer en el canal el tiempo suficiente para que la unidad destino los acepte. Puede utilizarse otra vez el flanco descendente del pulso de habilitación para activar un registro destino. Después la unidad destino habilita el pulso. La fuente quita los datos del canal después de un intervalo de tiempo predeterminado.

En muchas computadoras, el pulso de habilitación se controla en realidad mediante pulsos de reloj en la CPU. La CPU está siempre a cargo de los canales e informa a las unidades externas cómo transferir datos. Por ejemplo, el pulso de habilitación de la figura 11-3 debe ser una señal de control de escritura de memoria de la CPU a una unidad de memoria. La fuente, en este caso la CPU, coloca una palabra en el canal y le informa a la unidad de memoria, la cual es el destino, que ésta es una operación de escritura. De igual manera, el pulso de habilitación de la figura 11-4 puede ser una señal de control de lectura de memoria de la CPU a una unidad de memoria. El destino, la CPU, inicia la operación de lectura para informar a la memoria, la cual es la fuente, que coloque la palabra seleccionada en el canal de datos.

La transferencia de datos entre la CPU y una unidad de interface es similar a la que se acaba de describir. Por lo general, la transferencia de datos entre una interface y un dispositivo de E/S es controlada por un conjunto de líneas de reconocimiento mutuo.

Reconocimiento mutuo

La desventaja del método de habilitación es que la unidad fuente que inicia la transferencia no tiene manera de saber si la unidad destino ha recibido en realidad los datos que colocó en el canal. De igual manera, una unidad

destino que inicie la transferencia no puede saber si la unidad fuente ha colocado realmente los datos en el canal. El método de reconocimiento mutuo (handshake) resuelve este problema al introducir una segunda señal de control que proporciona una respuesta a la unidad que inicia la transferencia.

control de dos líneas

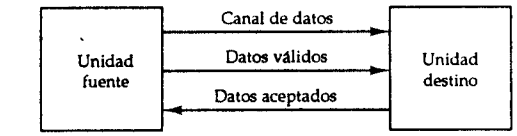
El principio básico del método de reconocimiento mutuo de dos líneas de transferencia de datos es el siguiente. Una línea de control está en la misma dirección que el flujo de datos en el canal, de la fuente al destino. La utiliza la unidad fuente para informar a la unidad destino si hay datos válidos en el canal. La otra línea de control está en la dirección opuesta. La utiliza la unidad destino para informar a la fuente si puede aceptar datos. La secuencia de control durante la transferencia depende de la unidad que inicia la transferencia.

La figura 11-5 muestra el procedimiento de transferencia de datos cuando lo inicia la fuente. Las dos líneas de reconocimiento mutuo son *datos válidos*, que genera la unidad fuente y, *datos aceptados*, generada por la unidad destino. El diagrama de temporización muestra el intercambio de señales entre las dos unidades. La secuencia de eventos listada en la parte (c) muestra los cuatro estados posibles que puede tener el sistema en cualquier momento. La unidad fuente inicia la transferencia al colocar los datos en el canal y habilitar su señal de *datos válidos*. La unidad destino activa la señal de *datos aceptados* después de que acepta los datos del canal. En seguida, la unidad fuente deshabilita su señal de *datos válidos*, la cual invalida los datos en el canal, después la unidad destino deshabilita su señal de *datos aceptados* y el sistema pasa a su estado inicial. La unidad fuente no envía los datos siguientes hasta después que la unidad destino muestra su disponibilidad para aceptar nuevos datos al deshabilitar su señal de *datos aceptados*. Este esquema permite retrasos arbitrarios de un estado al siguiente y que cada unidad responda a su propia velocidad de transferencia de datos. La velocidad de transferencia está determinada por la unidad más lenta.

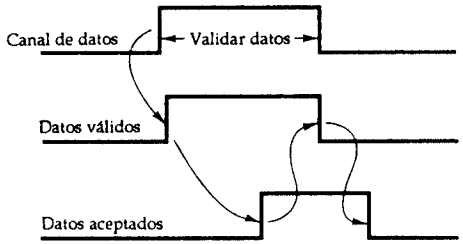
La transferencia que utiliza líneas de reconocimiento mutuo iniciada por la unidad destino se muestra en la figura 11-6. Nótese que el nombre de la señal generada por la unidad destino se ha cambiado a *preparada para datos*, con el fin de que refleje su nuevo significado. En este caso, la unidad fuente no coloca datos en el canal hasta que recibe la señal *preparada para datos* de la unidad destino. De ahí en adelante, el procedimiento de reconocimiento mutuo sigue el mismo patrón que en el caso iniciado por la unidad fuente. Nótese que la secuencia de eventos en ambos casos sería idéntica si consideramos la señal de *preparada para datos* como el complemento de *datos aceptados*. De hecho, la única diferencia entre la transferencia iniciada por la fuente y la iniciada por el destino, es en la elección de su estado inicial.

El esquema de reconocimiento mutuo proporciona un alto grado de flexibilidad y confiabilidad porque el término exitoso de una transferencia de datos se basa en la participación activa de ambas unidades. Si una unidad está defectuosa, no se completará la transferencia de datos. El error puede detectarse mediante un mecanismo de *tiempo transcurrido*, que produce una

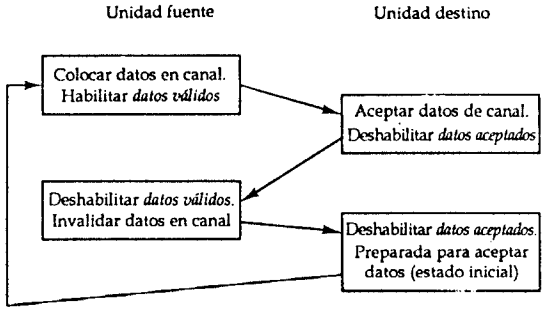
tiempo transcurrido



(a) Diagrama de bloque



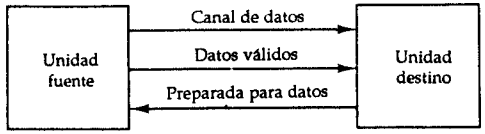
b) Diagrama de temporización



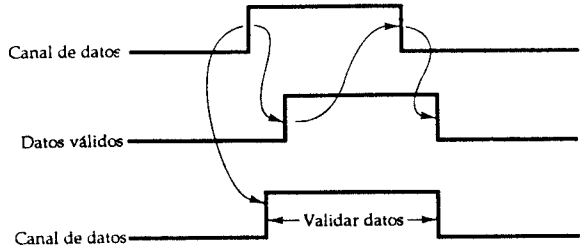
c) Secuencia de eventos

Figura 11-5 Transferencia iniciada por fuente utilizando reconocimiento mutuo.

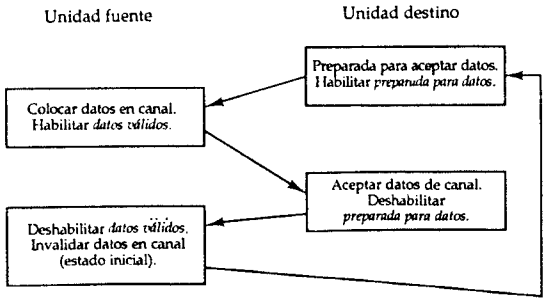
alarma si la transferencia de datos no se completa dentro de un tiempo predeterminado. La señal de tiempo transcurrido se implanta mediante un reloj interno que inicia un tiempo de conteo cuando la unidad habilita una de sus señales de control de reconocimiento. Si la señal de reconocimiento mutuo que proviene del destino no responde dentro de un cierto tiempo, la unidad considera que ha ocurrido un error. La señal de tiempo transcurrido puede utilizarse para interrumpir el procesador y, por lo tanto, para ejecutar una rutina de servicio que realice una acción de recuperación de error conveniente.



a) Diagrama de bloque



b) diagrama de temporización



c) Secuencia de eventos

Deshabilitar *datos válidos.*

Figura 11-6 Transferencia iniciada por destino utilizando reconocimiento mutuo.

Transferencia serial asíncrona

La transferencia de datos entre dos unidades puede hacerse en forma paralela o serial. En la transmisión de datos paralela, cada bit en el mensaje tiene su propia trayectoria y todo el mensaje se transmite al mismo tiempo. Esto significa que un mensaje de n bits debe transmitirse a través de n trayectorias conductoras separadas. En la transferencia de datos serial, cada bit en el mensaje se envía en secuencia uno a la vez. Este método requiere el uso de un par de conductores o un conductor y una

tierra común. La transmisión paralela es más rápida pero requiere muchas líneas. También se utiliza para distancias cortas y cuando la velocidad es importante. La transmisión serial es más lenta pero es menos cara porque sólo requiere un par de conductores.

síncrona

La transmisión serial puede ser síncrona o asíncrona. En la transmisión síncrona, las dos unidades comparten una frecuencia de reloj común y los bits se transmiten en forma continua a la velocidad que dictan los pulsos de reloj. En la transmisión serial de larga distancia, cada unidad es manejada por un reloj separado de la misma frecuencia. En forma periódica se transmiten señales de sincronización entre las dos unidades para mantener sus relojes en sincronía uno con el otro. En la transmisión asíncrona, sólo se envía información binaria cuando está disponible y la línea está desocupada cuando no hay información por transmitir. Esto contrasta con la información síncrona, en la cual los bits deben transmitirse en forma continua para conservar sincronizada la frecuencia de reloj en ambas unidades. La transmisión serial síncrona se analiza más adelante en la sección 11-8.

asíncrona

Una técnica serial de transmisión asíncrona de datos, utilizada en muchas terminales interactivas emplea bits especiales que se insertan en ambos extremos del código de carácter. Con esta técnica, cada carácter consta

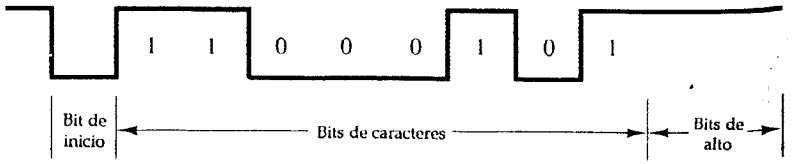
bit de inicio

de tres partes: un *bit de inicio*, los bits del carácter y los bits de paro. La convención es que el transmisor está en el estado 1 cuando no se transmiten caracteres. El primer bit, llamado el bit de inicio, es siempre 0 y se utiliza para indicar el comienzo de un carácter. El último bit, llamado el bit de paro, es siempre 1. En la figura 11-7 se muestra un ejemplo de este formato.

Un carácter transmitido puede detectarlo el receptor a partir del conocimiento de las reglas de transmisión:

1. Cuando no se está enviando un carácter, la línea se mantiene en el estado 1.
2. La iniciación de una transmisión de carácter se detecta a partir del bit de inicio, el cual es siempre 0.
3. Los bits de caracteres siempre van después del bit de inicio.
4. Después de que se transmite el último bit de carácter, se detecta un bit de paro cuando la línea retorna al estado 1, por al menos un tiempo correspondiente a un bit.

Figura 11-7 Transmisión serial asíncrona.



Al usar estas reglas, el receptor puede detectar el bit de inicio cuando la línea pasa de 1 a 0. Un reloj en el receptor examina la línea en los tiempos de bit convenientes. El receptor conoce la velocidad de transferencia de los bits y la cantidad de bits de caracteres que debe aceptar. Después de que se transmiten los bits de caracteres, se envían uno o dos bits de paro. Los bits de paro están siempre en el estado 1 y marcan el fin del carácter para dar a entender el estado desocupado o de espera.

bit de paro

Al final del carácter, la línea se conserva en el estado 1 por un período de al menos uno o dos tiempos correspondientes a un bit, para que el transmisor y el receptor puedan volverse a sincronizar. El tiempo que la línea permanece en este estado depende de la cantidad de tiempo requerida por el equipo para volverse a sincronizar. Algunas terminales electromecánicas antiguas utilizan dos bits de paro, pero las terminales más nuevas utilizan sólo un bit. La línea permanece en el estado 1 hasta que se transmite otro carácter. El tiempo de paro asegura que no llegará un carácter nuevo durante uno o dos tiempos de bit.

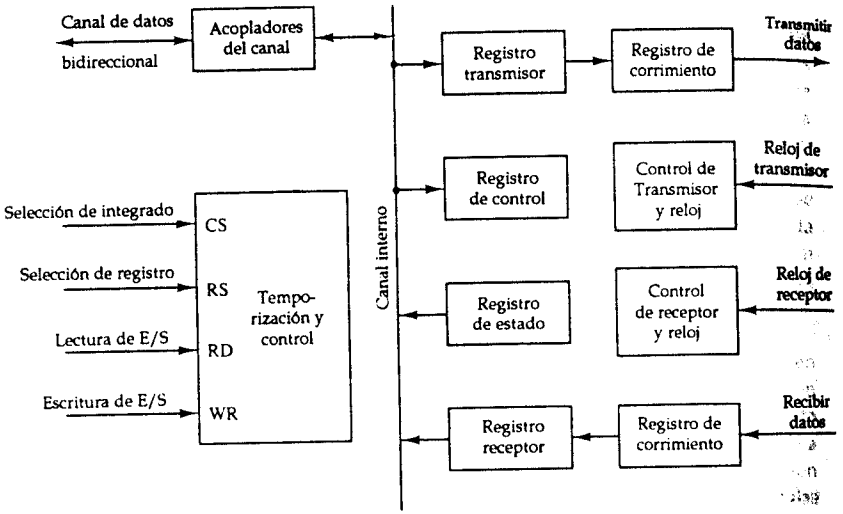
Como ejemplo, consideremos la transmisión serial de una terminal cuya velocidad de transferencia es de 10 caracteres por segundo. Cada carácter transmitido consta de un bit de inicio, ocho bits de información, y dos bits de paro, para un total de 11 bits. 10 caracteres por segundo significan que cada carácter necesita 0.1s para la transferencia. Como se van a transmitir 11

velocidad en baudios

bits, se sabe que el tiempo para un bit es 9.09 ms. La *velocidad en baudios* se define como la velocidad a la cual se transmite información serial y es equivalente a la transferencia de datos en bits por segundo. Diez caracteres por segundo con un formato de 11 bits tiene una velocidad de transferencia de 110 baudios. La terminal tiene un teclado y una impresora. Cada vez que se oprime una tecla, una terminal envía 11 bits en forma serial a lo largo de una línea. Para imprimir un carácter en la impresora, debe recibirse un mensaje de 11 bits a través de otra línea. La interface de la terminal consta de un transmisor y un receptor. El transmisor acepta un carácter de 8 bits de la computadora y procede a enviar un mensaje serial de 11 bits a través de la línea de la impresora. El receptor acepta el mensaje serial de 11 bits de la línea del teclado e introduce el código de carácter de 8 bits dentro de la computadora. Están disponibles circuitos integrados diseñados en forma específica para proporcionar la interface entre la computadora y terminales interactivas similares. Tal circuito se denomina una *interface de comunicación asíncrona* o *receptor-transmisor asíncrono universal* (*universal asynchronous receiver-transmitter*, UART).

Interface de comunicación asíncrona

El diagrama de bloque de una interface de comunicación asíncrona se muestra en la figura 11-8. Funciona como transmisor y receptor. La interface se inicializa para un modo de transferencia particular mediante un byte de control que se carga dentro de su registro de control. El registro transmisor



CS	RS	Operación	Registro seleccionado
0	×	×	Ninguno: canal de datos en alta impedancia
1	0	WR	Registro de transmisor
1	1	WR	Registro de control
1	0	RD	Registro receptor
1	1	RD	Registro de estado

Figura 11-8 Diagrama de bloque de una interface típica de comunicación asíncrona.

acepta un byte de datos de la CPU a través del canal de datos. Este byte se transfiere a un registro de corrimiento para transmisión serial. La parte receptora recibe información serial dentro de otro registro de corrimiento y, cuando se acumula un byte de datos completo, se transfiere al registro receptor. La CPU puede seleccionar que el registro receptor lea el byte a través del canal de datos. Los bits en el registro de estado se utilizan para banderas de entrada y salida y para registrar ciertos errores que pueden ocurrir durante la transmisión. La CPU puede leer el registro de estado para comprobar el estado de los bits de bandera y determinar si ha ocurrido algún error. Las líneas de control de selección de integrado y de lectura y escritura

transmisor

receptor

comunican con la CPU. La entrada de selección de integrado (CS) se utiliza para seleccionar la interface a través del canal de direcciones. La selección de registro (RS) se asocia con los controles de lectura (RD) y escritura (WR). Dos registros son sólo de escritura y dos son sólo de lectura. El registro seleccionado es una porción del valor RS y de los estados RD y WR, según se lista en la tabla que acompaña al diagrama.

La operación de una interface de comunicación asíncrona se inicializa mediante la CPU al enviar un byte al registro de control. El procedimiento de inicialización coloca la interface en un modo de operación específico porque define ciertos parámetros, como la velocidad en baudios que se va a utilizar, cuántos bits hay en cada carácter, la decisión de generar una comprobación de la paridad y cuántos bits de alto se añaden a cada carácter. Dos bits en el registro de estado se usan como banderas. Un bit se utiliza para indicar si el registro transmisor está vacío y otro se emplea para indicar si el registro receptor está lleno.

La operación de la parte de transmisor de la interface es la siguiente. La CPU lee el registro de estado y comprueba la bandera para observar si está vacío el registro transmisor. Si está vacío, la CPU transfiere un carácter al registro transmisor y la interface desactiva la bandera para marcar lleno el registro. El primer bit en el registro de corrimiento del transmisor se activa en 0 para generar un bit de inicio. El carácter se transfiere en paralelo del registro transmisor al registro de desplazamiento y se agrega la cantidad apropiada de bits de alto al registro de corrimiento. Después se marca vacío el registro transmisor. Ahora puede transmitirse el carácter un bit a la vez al colocar los datos en el registro de corrimiento a la velocidad de baudios especificada. La CPU puede transferir otro carácter al registro transmisor, después de comprobar la bandera en el registro de estado. Se dice que la interface es de registro *doble búffer* porque puede cargarse un nuevo carácter tan pronto como el anterior comienza la transmisión.

La operación de la parte receptora de la interface es similar. La entrada de recepción de datos está en el estado 1 cuando la línea está inactiva. El control receptor monitorea la línea de recepción de datos en busca de una señal 0 para detectar la ocurrencia de un bit de inicio. Una vez que se ha detectado un bit de inicio, los bits de carácter que entran se colocan en el registro de corrimiento en donde se recorren a la velocidad de baudio establecida. Después de recibir los bits de datos, la interface comprueba la paridad y los bits de alto. En seguida, se transfiere el carácter, en paralelo, del registro de corrimiento al registro de recepción, sin los bits de inicio y de alto. La bandera en el registro de estado se activa para indicar que el registro receptor está lleno. La CPU lee el registro de estado y comprueba la bandera y, si está activa, lee los datos del registro receptor.

La interface verifica cualesquiera errores posibles durante la transmisión y activa bits apropiados en el registro de estado. La CPU puede leer el registro de estado en cualquier momento para comprobar si han ocurrido errores. Tres errores posibles que comprueba la interface durante la transmi-

sión son el error de paridad, el error de configuración y el error de superposición. Un error de paridad ocurre si la cantidad de dígitos 1 en los datos recibidos no corresponde a la paridad correcta. Un error de posición ocurre si no se detecta el número correcto de bits de alto al final del carácter recibido. Un error de superposición ocurre si la CPU no lee el carácter del registro receptor antes de que el siguiente quede disponible en el registro de corrimiento. Los errores de estancamiento dan como resultado una pérdida de caracteres en el flujo de datos recibido.

Buffer primero en entrar, primero en salir

FIFO

Un buffer primero en entrar, primero en salir (*first-in, first-out*, FIFO) es una unidad de memoria de localidades adyacentes que almacena información de manera que el primer dato que entra es el primero que sale. Un búffer FIFO tiene terminales de entrada y salida separadas. La característica importante de este buffer es que puede introducir y sacar datos a dos velocidades diferentes y que los datos de salida están siempre en el mismo orden en el cual se introdujeron al buffer. Cuando se coloca entre dos unidades, el FIFO puede aceptar datos de la unidad fuente a una velocidad de transferencia y enviar los datos a la unidad destino a otra velocidad. Si la unidad fuente es más lenta que la unidad destino, el búffer puede llenarse con datos a una velocidad lenta y después vaciarse a una velocidad más rápida. Si la fuente es más rápida que el destino, el FIFO es útil para aquellos casos en donde los datos fuente arriban en grandes cantidades que llenan el buffer, pero que el tiempo entre esos arribos es suficientemente largo para que la unidad destino vacie alguna o toda la información del buffer. Por lo tanto, un buffer FIFO puede ser útil en algunas aplicaciones cuando se transfieren datos en forma asíncrona. El buffer FIFO apila los datos conforme llegan y los entrega en el mismo orden cuando se necesitan.

El diagrama lógico de un buffer FIFO típico 4 × 4 se muestra en la figura 11-9. Consta de cuatro registros de 4 bits *RI*, $I = 1, 2, 3, 4$ y un registro de control con flip-flops F_i, $i = 1, 2, 3, 4$, uno para cada registro. El FIFO puede almacenar cuatro palabras de cuatro bits cada una. Puede aumentarse la cantidad de bits por palabra al elevar la cantidad de bits en cada registro y puede aumentarse la cantidad de palabras al elevar la cantidad de registros.

Un flip-flop F_i en el registro de control que está activo en 1, indica que una palabra de datos de 4 bits está almacenada en el registro *RI* correspondiente. Un 0 en F_i indica que el registro correspondiente no contiene datos válidos. El registro de control dirige el movimiento de datos por los registros. Cada vez que el bit F_i del registro de control está activado ($F_i = 1$) y el bit F_{i+1} se reactiva ($F'_{i+1} = 1$), se genera un pulso de reloj que hace que el registro $R(I + 1)$ acepte los datos del registro *RI*. La misma transición de reloj activa F_{i+1} en 1 y desactiva F_i en 0. Esto provoca que la bandera de control se mueva una posición a la derecha junto con los datos. Los datos en el registro se mueven hacia abajo del FIFO, hacia la salida, mientras existan posiciones

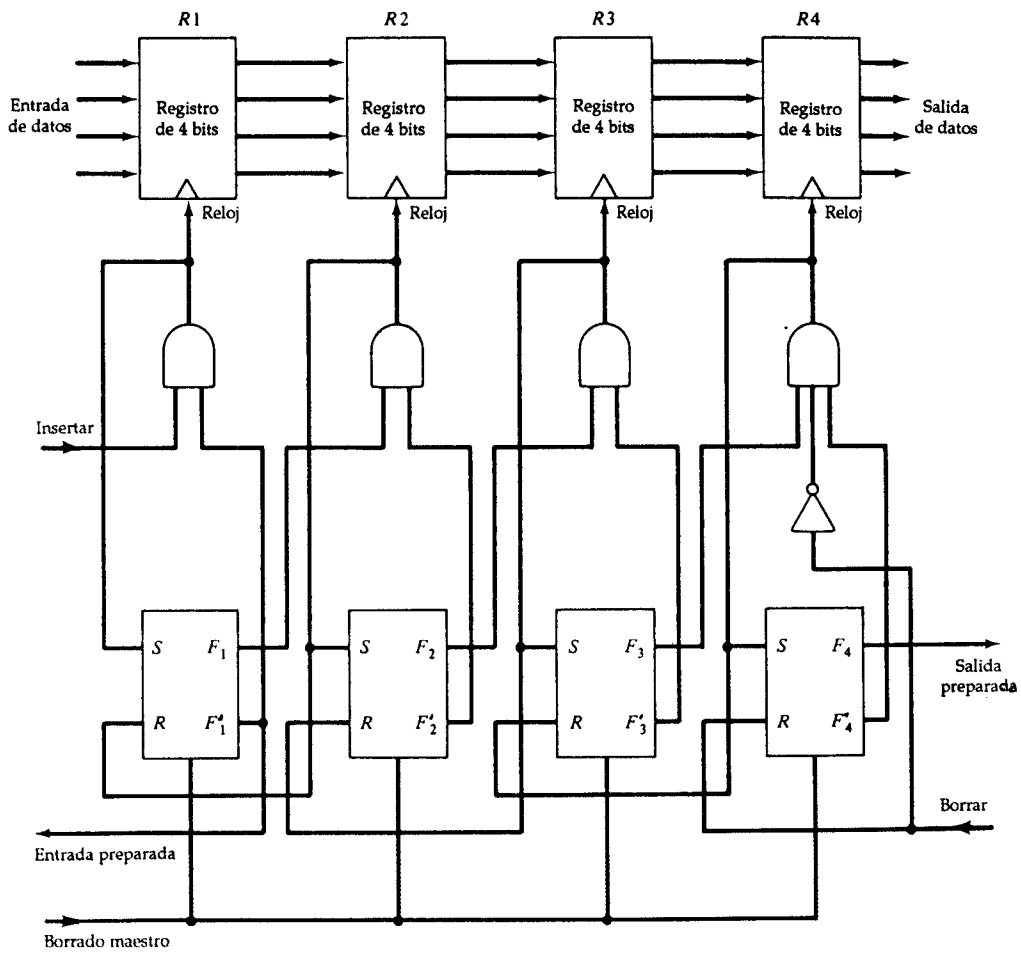


Figura 11-9 Diagrama de circuito de buffer FIFO 4 × 4.

vacías adelante de él. Esta operación de avance se detiene cuando los datos alcanzan el registro *RI* con el siguiente flip-flop F_{i+1} activado en 1 o cuando alcanzan el último registro *R*4. Se utiliza un borrado general maestro para inicializar todos los flip-flops del registro de control en 0.

Se insertan datos dentro del buffer siempre y cuando esté habilitada la señal *entrada preparada*. Esto ocurre cuando se desactiva el primer flip-flop F_1, indicando que el registro *RI* está vacío. Se cargan los datos de las líneas

de entrada al habilitar el reloj en $R1$ mediante la línea de control *insertar*. El mismo reloj activa F_3, lo cual deshabilita el control *entrada preparada*, lo que indica que el FIFO está ahora ocupado y no puede aceptar más datos. El proceso de avance comienza siempre y cuando $R2$ esté vacío. Los datos en $R1$ se transfieren dentro de $R2$ y se desactiva F_1. Esto habilita la línea *entrada preparada*, indicando que las entradas están ahora disponibles para otra palabra de datos. Si el FIFO está lleno, F_1 permanece activa y la línea *entrada preparada* permanece en el estado 0. Nótese que las dos líneas de control *entrada preparada* e *insertar* constituyen un par de líneas de reconocimiento mutuo iniciadas por el destino.

Los datos recorren la pila de registros hasta el extremo de salida. La línea de control *salida preparada* está habilitada cuando se activa el último flip-flop de control $F4$, indicando que hay datos válidos en el registro de salida $R4$. La unidad destino acepta los datos de salida de $R4$, y esa misma unidad deshabilita la señal de control *borrar*. Esto desactiva F_4, haciendo que se deshabilite la *salida preparada*, lo cual indica que los datos en la salida ya no son válidos. Sólo después de que la señal *borrar* regresa a 0 pueden moverse los datos de $R3$ dentro de $R4$. Si el FIFO está vacío, no habrá datos en $R3$ y F_4 permanecerá en estado inactivo. Nótese que las dos líneas de control *salida preparada* y *borrar* constituyen un par de líneas de reconocimiento mutuo iniciadas por la fuente.

11-4 Modos de transferencia

La información binaria recibida de un dispositivo externo por lo general se almacena en la memoria para su procesamiento posterior. La información transferida de la computadora central a un dispositivo externo se origina en la unidad de memoria. La CPU sólo ejecuta las instrucciones E/S y puede aceptar los datos en forma temporal, pero la fuente o destino final es la unidad de memoria. La transferencia de datos entre la computadora central y los dispositivos de E/S puede manejarse en diversos modos. Algunos modos utilizan la CPU como una trayectoria intermedia; otros transfieren los datos directamente a y de la unidad de memoria. La transferencia de datos de y a periféricos puede manejarse en uno de tres modos posibles.

1. E/S programada
2. E/S iniciada por interrupción
3. Acceso directo a memoria (DMA)

E/S programada

Las *operaciones de E/S programadas* son el resultado de instrucciones de E/S escritas en el programa de la computadora. Cada transferencia de datos se inicia mediante una instrucción en el programa. Por lo general, la transferencia es hacia y desde un registro de CPU o periférico. Se necesitan otras instrucciones para transferir los datos hacia y desde la CPU y la memoria.

Transferir datos bajo el control del programa requiere que la CPU realice un monitoreo constante de periféricos. Una vez que se inicia una transferencia de datos, es necesario que la CPU monitoree la interface para ver cuándo puede volverse a hacer una transferencia. Depende de las instrucciones programadas y ejecutadas en la CPU, observar en detalle todo lo que acontece en la unidad de interface y en el dispositivo de E/S.

En el método de E/S programada, la CPU permanece en un ciclo de programa hasta que la unidad de E/S indica que está preparada para transferencia de datos. Este es un proceso que consume bastante tiempo porque mantiene ocupado el procesador en forma innecesaria. Puede evitarse al usar una opción de interrupción y comandos especiales para informar a la interface que emita una señal de solicitud de interrupción cuando están disponibles los datos del dispositivo. Mientras tanto, la CPU puede avanzar a ejecutar otro programa. En ese lapso, la interface sigue monitoreando dispositivos. Cuando la interface determina que el dispositivo está preparado para transferencia de datos, genera una solicitud de interrupción a la computadora. Cuando se detecta la señal de interrupción externa, la CPU detiene un momento la tarea que está procesando, transfiere el control a un programa de servicio para procesar la transferencia de E/S y después regresa a la tarea que ejecutaba originalmente.

interrupción

La transferencia de datos bajo E/S programada es entre la CPU y un periférico. En el acceso directo a memoria, (DMA), la interface transfiere datos hacia adentro y hacia afuera de la unidad de memoria por medio del canal de memoria. La CPU inicia la transferencia al proporcionar a la interface la dirección inicial y la cantidad de palabras necesarias que se van a transmitir y después avanza a ejecutar otras tareas. Cuando se hace la transferencia, el DMA solicita ciclos de memoria mediante el canal de memoria. Cuando el controlador de memoria concede la solicitud, el DMA transfiere los datos directamente a la memoria. La CPU sólo retrasa su operación de acceso a memoria para permitir la transferencia directa de E/S a memoria. Como la velocidad de los periféricos por lo general es menor que la velocidad del procesador, las transferencias a memoria de E/S no son frecuentes en comparación con el acceso a memoria del procesador. La transferencia DMA se analiza con mayor detalle en la sección 11-6.

DMA

Muchas computadoras combinan la lógica de interface con los requisitos para acceso directo a memoria en una unidad y la llaman procesador de E/S IOP. El IOP puede manejar muchos periféricos a través de un DMA y la opción de interrupción. Con tal sistema, la computadora se divide en tres módulos separados: la unidad de memoria, la CPU y el IOP. Los procesadores E/S se presentan en la sección 11-7.

IOP

Ejemplo de E/S programada

En el método de E/S programada, el dispositivo de E/S no tiene acceso directo a la memoria. Una transferencia de un dispositivo de E/S a memoria

requiere que la CPU ejecute varias instrucciones, incluyendo una instrucción de entrada para transferir los datos del dispositivo a la CPU y una instrucción de almacenamiento para transferir los datos de la CPU a la memoria. Pueden necesitarse otras instrucciones para verificar que están disponibles los datos del dispositivo y para contar la cantidad de palabras transferidas.

Un ejemplo de transferencia de datos de un dispositivo de E/S por medio de una interface a la CPU se muestra en la figura 11-10. El dispositivo transfiere bytes de datos uno a la vez, conforme están disponibles. Cuando está disponible un byte de datos, el dispositivo lo coloca en el canal de E/S y habilita su línea de datos válidos. La interface acepta el byte en su registro de datos y habilita la línea de datos aceptados. La interface activa un bit en el registro de estado que denominaremos bit de "*bandera*" o bit *F*. Ahora el dispositivo puede deshabilitar la línea de datos válidos, pero no transferirá otro byte hasta que la interface deshabilite la línea de datos aceptados. Esto se apega al procedimiento de reconocimiento mutuo establecido en la figura 11-5.

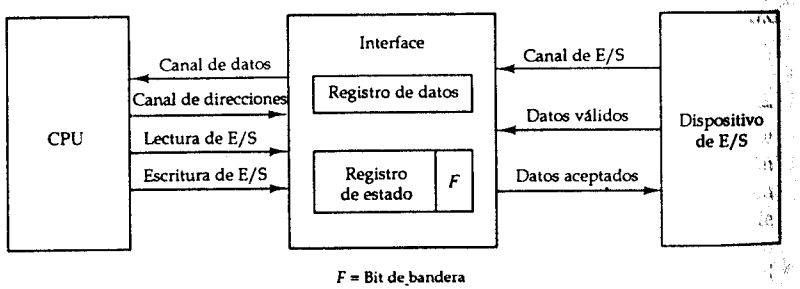


Figura 11-10 Transferencia de datos de dispositivo E/S a CPU.

Está escrito un programa para la computadora con el fin de comprobar la bandera en el registro de estado, para determinar si se ha colocado un byte en el registro de datos mediante el dispositivo de E/S. Esto se hace al leer el registro de estado dentro del registro de la CPU y comprobar el valor del bit de bandera. Si la bandera es igual a 1, la CPU lee los datos del registro de datos. Después, la CPU o la interface desactivan el bit de bandera a 0, dependiendo de cómo están diseñados los circuitos de la interface. Una vez que se desactiva la bandera, la interface deshabilita la línea de datos aceptados y el dispositivo puede transferir, entonces, el siguiente byte de datos.

Un diagrama de flujo del programa que debe escribirse para la CPU, se muestra en la figura 11-11. Se considera que el dispositivo está enviando

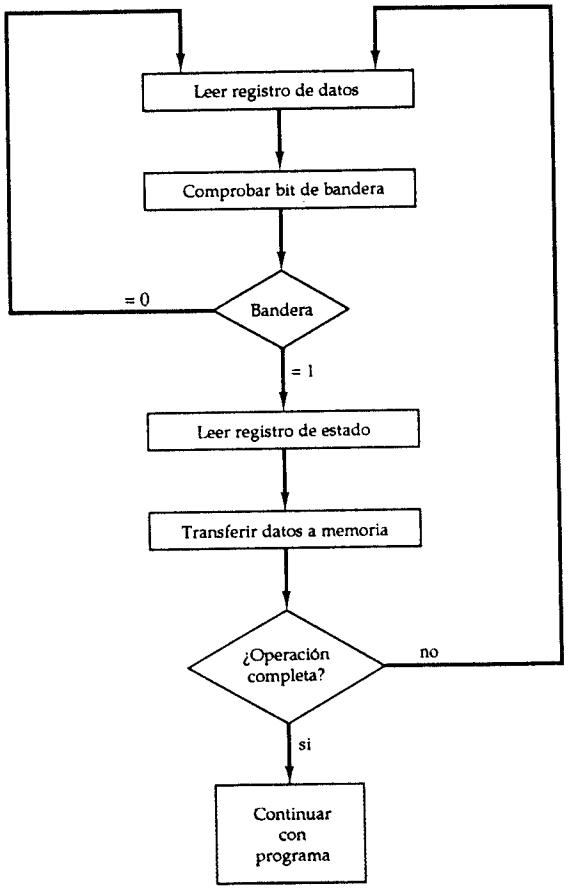


Figura 11-11 Diagrama de flujo para programa de CPU para introducir datos.

una secuencia de bytes que debe almacenarse en la memoria. La transferencia de cada byte requiere de tres instrucciones:

1. Leer el registro de estado.
2. Comprobar el estado del bit de bandera y transferir el control al paso 1 si no está activado o al paso 3 si lo está.
3. Leer el registro de datos.

Cada byte se lee en el registro de la CPU y después se transfiere a la memoria con una instrucción de almacenar. Una tarea común de programación de E/S es transferir un bloque de palabras de un dispositivo de E/S y almacenarlas en un buffer de memoria. Un programa que almacena caracteres de entrada en el búffer que utiliza las instrucciones definidas en el capítulo 6 se lista en la tabla 6-21.

El método de E/S programada es particularmente útil en computadoras pequeñas de baja velocidad o en sistemas que están dedicados a monitorear un dispositivo en forma continua. La diferencia en la velocidad de transferencia de información entre la CPU y el dispositivo de E/S hace ineficiente este tipo de transferencia. Para apreciar por qué es ineficiente, consideremos una computadora típica que puede ejecutar las dos instrucciones que leen el registro de estado y comprueban la bandera en un micro-segundo. Consideremos que el dispositivo de entrada transfiere sus datos a una velocidad promedio de 100 bytes por segundo. Esto es equivalente a un byte cada 10000 micro-segundos. Esto significa que la CPU comprobará la bandera 10,000 veces entre cada transferencia. La CPU está gastando tiempo mientras comprueba la bandera en lugar de hacer alguna otra tarea de procesamiento útil.

rutina de E/S

E/S iniciada por interrupción

Una alternativa para que la CPU no tenga que monitorear en forma constante la bandera, es permitir que la interface informe a la computadora cuando esté lista para transferir datos. Este modo de transferencia utiliza la opción de interrupción. Mientras la CPU está corriendo un programa común no comprueba la bandera. Sin embargo, cuando se activa la bandera, se interrumpe momentáneamente el avance de la computadora con el programa actual y se informa que se ha activado la bandera. La CPU se desvía de lo que está haciendo para atender la transferencia de entrada o salida. Después que termina la transferencia, la computadora retorna al programa previo para continuar lo que estaba haciendo antes de la interrupción.

La CPU responde a la señal de interrupción al almacenar la dirección de retorno del contador de programa dentro de una pila de memoria y después transferirse el control a una rutina de servicio que procesa la transferencia de E/S requerida. La manera en que el procesador elige la dirección de transferencia a la rutina de servicio varía de una unidad a otra. En principio, existen dos métodos para lograr esto. Uno se llama *interrupción con vector* y el otro *interrupción sin vector*. En una interrupción sin vector, la dirección de transferencia de control del programa se asigna a una posición física en la memoria. En una interrupción con vector, la fuente que interrumpe proporciona la información de la transferencia a la computadora. Esta información se llama el *vector de interrupción*. En algunas computadoras el vector de interrupción es la primera dirección de la rutina de servicio de E/S. En otras computadoras el vector de interrupción es una dirección que

interrupción con vector

apunta a una posición en la memoria en la cual se almacena la dirección inicial de la rutina de servicio de E/S. En la sección 11-5 se muestra un sistema de interrupción con vector.

Consideraciones de programación

El análisis anterior se relacionó con la circuitería necesaria para lograr la interface de dispositivos de E/S hacia un sistema de computadora. Una computadora también debe tener rutinas de programación para controlar los periféricos y para la transferencia de datos entre el procesador y los periféricos. *Las rutinas de E/S* deben emitir comandos de control para activar el periférico y para comprobar el estado del dispositivo, con el fin de determinar cuándo está listo para transferencia de datos. Una vez que está preparado, la información se transfiere dato a dato hasta terminar. En algunos casos, se proporciona entonces un comando de control para ejecutar una función de dispositivo, como detener la cinta o imprimir caracteres. Con frecuencia las transferencias están acompañadas de comprobación de errores y otros pasos útiles. En las transferencias controladas por interrupción, los programas de E/S deben enviar comandos al periférico para interrumpir cuando esté preparado y para dar servicio a la interrupción cuando ocurra. En la transferencia DMA, los programas de E/S deben inicializar el canal de DMA para comenzar su operación.

El control por programa del equipo de entrada-salida es una empresa compleja. Por esta razón, los fabricantes proporcionan rutinas de E/S para los periféricos estándar como parte del sistema de computadora. Por lo general, se incluyen dentro del sistema operativo. La mayoría de los sistemas operativos contienen diversos programas de E/S para soportar la línea de periféricos particular que se ofrece para la computadora. Las rutinas de E/S están disponibles como procedimientos de sistemas operativos y el usuario hace referencia a las rutinas establecidas para especificar el tipo de transferencia requerido, sin entrar a detallados programas en lenguaje de máquina.

11-5 Prioridad de interrupción

La transferencia de datos entre la CPU y un dispositivo de E/S la inicia la CPU. Sin embargo, la CPU no puede comenzar la transferencia a menos que el dispositivo esté preparado para comunicarse con la CPU. La disponibilidad del dispositivo puede determinarse de una señal de interrupción. La CPU responde a la solicitud de interrupción al almacenar la dirección de retorno del PC dentro de una pila de memoria y después el programa se transfiere a una rutina de servicio que procesa la transferencia solicitada. Como se analizó en la sección 8-7, algunos procesadores también salvan dentro de la pila la palabra de estado del procesador (PSW) y cargan una

nueva PSW para la rutina de servicio. Aquí no consideramos la PSW para no complicar el análisis de las interrupciones de E/S.

En una aplicación típica, se conectan varios dispositivos de E/S a la computadora, y cada dispositivo puede originar una solicitud de interrupción. La primera tarea del sistema de interrupción es identificar la fuente de la interrupción. También existe la posibilidad de que varias fuentes soliciten servicio en forma simultánea. En este caso, el sistema debe decidir también a cuál dispositivo atender primero.

prioridad de interrupción

Una *prioridad de interrupción* es un sistema que establece una prioridad entre las diversas fuentes para determinar qué condición se va atender primero cuando llegan al mismo tiempo dos solicitudes. El sistema también puede determinar cuales condiciones se permiten para interrumpir a la computadora mientras se da servicio a otra interrupción. Se asignan niveles de interrupción de alta prioridad a solicitudes que, si se posponen o interrumpen, pueden producir consecuencias serias. Los dispositivos con transferencias de alta velocidad como discos magnéticos reciben una alta prioridad y los dispositivos lentos como los teclados reciben baja prioridad. Cuando dos dispositivos interrumpen la computadora al mismo tiempo, la computadora atiende al dispositivo con mayor prioridad.

encuesta

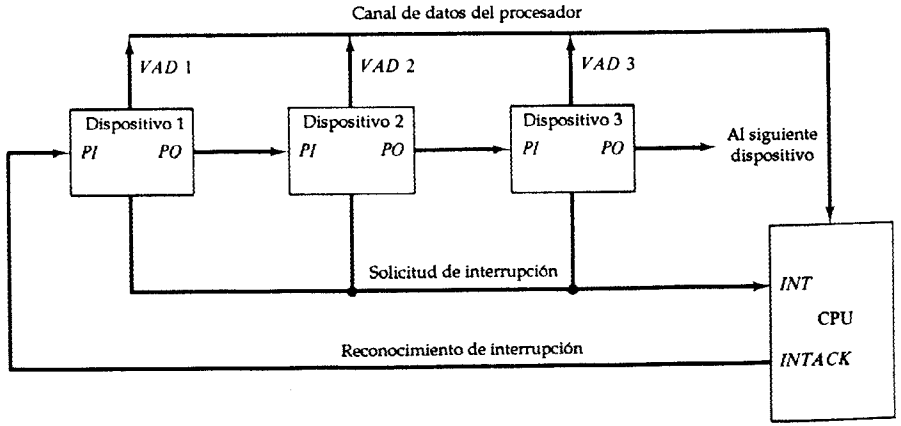
Puede establecerse la prioridad de interrupciones simultáneas mediante programación o circuitería. Se usa un procedimiento de "encuesta" para identificar la fuente de prioridad más alta por medio de programación. En este método existe una dirección de transferencia de control común para todas las interrupciones y el programa cuida que las interrupciones comiencen en la dirección de transferencia y registra las fuentes de interrupción en secuencia. El orden en la cual se prueba determina la prioridad de cada interrupción. Se prueba primero la fuente de prioridad más alta y, si su señal de interrupción está activada, el control se transfiere a una rutina de servicio para esta fuente. De otra manera, se prueba la fuente con la siguiente prioridad hacia abajo y así sucesivamente. Por lo tanto, la rutina de servicio inicial para todas las interrupciones consiste en un programa que prueba las fuentes de interrupción en secuencia y transfiere el control a una de varias rutinas de servicio posibles. La rutina de servicio particular alcanzada, pertenece al dispositivo de prioridad más alta entre todos los dispositivos que interrumpieron a la computadora. La desventaja del método de programación es que, si hay muchas interrupciones, el tiempo requerido para registrarlas puede exceder el tiempo disponible para atender el dispositivo de E/S. En esta situación, puede utilizarse una unidad de circuito de prioridad de interrupción para acelerar la operación.

Una unidad de prioridad de interrupción de circuitería funciona como un administrador general en un ambiente de sistema de interrupciones. Acepta solicitudes de interrupción de muchas fuentes, determina cuál de las solicitudes que llegan tiene la prioridad más alta y emite una solicitud de interrupción a la computadora con base en esta determinación. Para acelerar la operación, cada fuente de interrupción tiene su propio vector de interrupción para accesar en forma directa su propia rutina de servicio. Por lo tanto, no se necesita el registro porque todas las decisiones las establece la unidad de circuito de prioridad de interrupción. La función de prioridad de circuitería puede establecerla una conexión serial o paralela de líneas de interrupción. La conexión serial también se conoce como el método de cadena circular.

Prioridad de cadena de margaritas

El método de cadena de margaritas (daisy chain) de establecer prioridad consiste en una conexión serial de todos los dispositivos que solicitan una interrupción. El dispositivo con la prioridad más alta se coloca en la primera posición, seguido por los dispositivos de prioridad inferior hasta el dispositivo con la más baja prioridad, que se coloca al último en la cadena. Este método de conexión entre tres dispositivos en la CPU se muestra en la figura 11-12. La línea de solicitud de interrupción es común a todos los dispositivos y forma una conexión de lógica alambrada. Si cualquier dispositivo tiene su señal de interrupción en el estado de nivel bajo, la línea de interrupción va al estado de nivel bajo y habilita la entrada de interrupción en la CPU. Cuando no está pendiente ninguna interrupción, la línea de interrupción permanece en el estado de nivel alto y la CPU no reconoce interrupciones. Esto es equivalente a una operación OR lógica negativa. La CPU responde a una solicitud de interrupción al habilitar la línea de reconocimiento de interrupción. El dispositivo 1 recibe esta señal en su entrada de prioridad (*PI*). La señal de reconocimiento pasa al siguiente dispositivo por medio de

Figura 11-12 Interrupción con prioridad por cadena de margaritas.



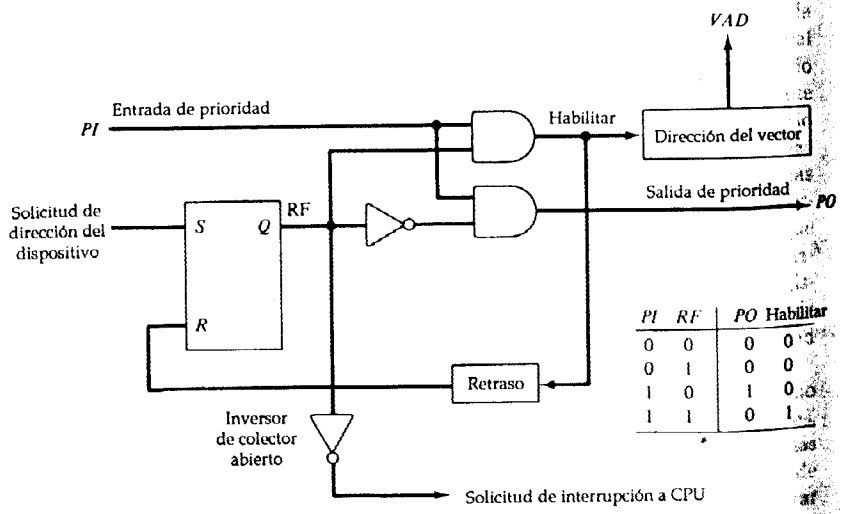
dirección de vector

la salida de prioridad (*PO*) sólo si el dispositivo 1 no está solicitando una interrupción. Si el dispositivo 1 tiene una interrupción pendiente, bloquea la señal de reconocimiento del siguiente dispositivo al colocar un 0 en su salida *PO*. Entonces, procede e inserta su propia dirección de vector (VAD) de interrupción, dentro del canal de datos, para que la CPU la utilice durante el ciclo de interrupción.

Un dispositivo con un 0 en su entrada *PI* genera un 0 en su salida *PO* para informar al dispositivo con la siguiente prioridad más baja que ha bloqueado la señal de reconocimiento. Un dispositivo que solicita una interrupción y tiene un 1 en su entrada *PI* acepta la señal de reconocimiento al colocar un 0 en su salida *PO*. Si el dispositivo no tiene interrupciones pendientes, transmite la señal de reconocimiento al siguiente dispositivo al colocar un 1 en su salida *PO*. Por lo tanto, el dispositivo con *PI* = 1 y *PO* = 0 es el que tiene la prioridad más alta y que solicita una interrupción y este dispositivo coloca su dirección de vector (VAD) en el canal de datos. El arreglo de cadena de margaritas le da la prioridad más alta al dispositivo que recibe la señal de reconocimiento de interrupción de la CPU. Entre más lejos está el dispositivo de la primera posición, es menor su prioridad.

La figura 11-13 muestra la lógica interna que debe incluirse dentro de cada dispositivo cuando está conectado en un esquema de cadena de margaritas. El dispositivo activa su flip-flop *RF* cuando desea interrumpir a la CPU. La salida del flip-flop *RF* recorre un inversor de colector abierto, un

Figura 11-13 Una etapa del arreglo de prioridad de cadena de margarita.



circuito que proporciona la lógica de alambrado para la línea de interrupción común. Si *PI* = 0, *PO* y la línea de habilitación a la VAD son iguales a 0, sin considerar el valor de *RF*. Si *PI* = 1 y *RF* = 0, entonces *PO* = 1 y se deshabilita la dirección del vector. Esta condición pasa la señal de reconocimiento al siguiente dispositivo a través de *PO*. El dispositivo está activo cuando *PI* = 1 y *RF* = 1. Esta condición coloca un 0 en *PO* y habilita la dirección del vector para el canal de datos. Se considera que cada dispositivo tiene su propia dirección de vector distinta. Se desactiva el flip-flop *RF* después de un retraso suficiente para asegurar que la CPU ha recibido la dirección de vector.

Interrupción de prioridad paralela

El método de interrupción de prioridad paralela utiliza un registro cuyos bits se activan en forma separada, mediante la señal de interrupción de cada dispositivo. La prioridad se establece de acuerdo con la posición de los bits en el registro. Además del registro de interrupción, el circuito puede incluir un registro de máscara, cuyo propósito es controlar el estado de cada solicitud de interrupción. Puede programarse el registro de máscara para deshabilitar interrupciones de prioridad menor mientras se está atendiendo un dispositivo de prioridad más alta. También puede proporcionar una opción que permita que un dispositivo de prioridad alta interrumpa la CPU mientras se atiende un dispositivo de prioridad menor.

lógica de prioridad

La lógica de prioridad para un sistema de cuatro fuentes de interrupción se muestra en la figura 11-14. Consta de un registro de interrupción cuyos bits individuales se activan mediante condiciones externas y se desactivan mediante instrucciones de programa. El disco magnético, como es un dispositivo de alta velocidad, recibe la prioridad más alta. La impresora tiene la siguiente prioridad, seguida por una lectora de caracteres y un teclado. El registro de máscara tiene la misma cantidad de bits que el registro de interrupción. Mediante instrucciones de programa, es posible activar o desactivar cualquier bit en el registro de máscara. Cada bit de interrupción y su bit de máscara correspondientes, se aplican a una compuerta AND para producir las cuatro entradas hacia un codificador de prioridad. De esta manera, se reconoce una interrupción sólo si el programa activó en 1 su bit de máscara correspondiente. El codificador de prioridad genera dos bits de la dirección de vector, la cual se transfiere a la CPU.

Otra salida del codificador activa un flip-flop de estado de interrupción (IST) cuando ocurre una interrupción que no está enmascarado. El programa puede habilitar o deshabilitar el flip-flop de habilitación de interrupción (*interupt enable*, IEN) para proporcionar un control general sobre el sistema de interrupciones. Las salidas de *IST* que recibieron la función AND con *IEN* proporcionan una señal de interrupción común para la CPU. La señal de reconocimiento de interrupción (INTACK) de la CPU habilita los buffers del canal en el registro de salida y se coloca una dirección de vector VAD dentro

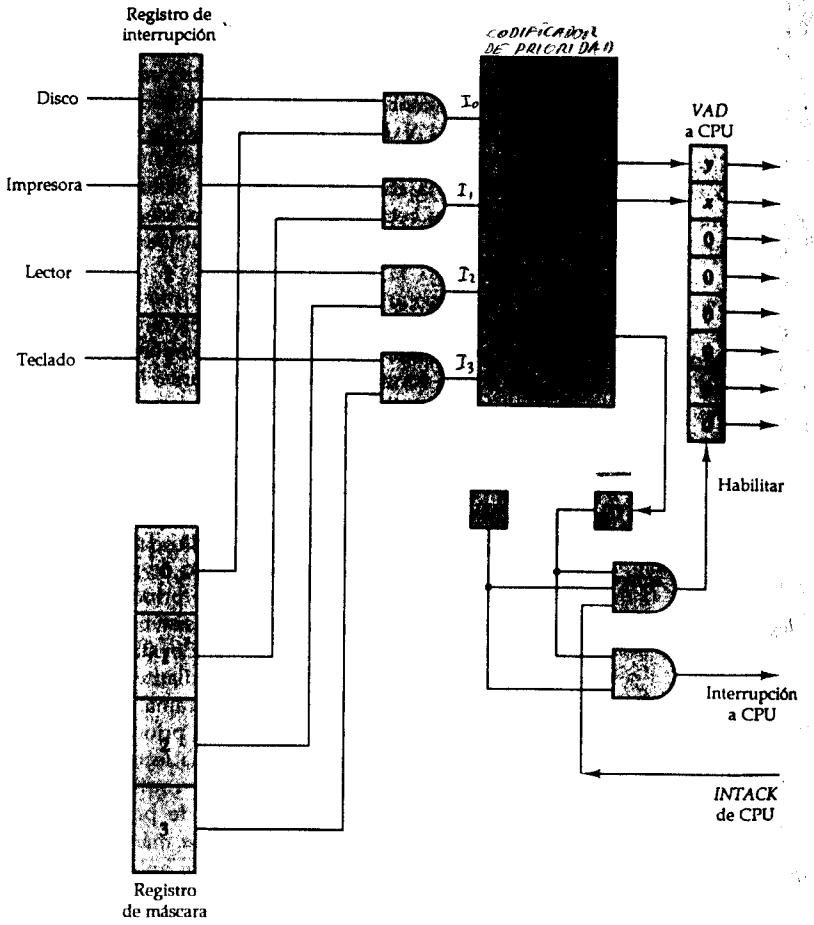


Figura 11-14 Circuitería de prioridad de interrupción.

del canal de datos. Ahora explicaremos el circuito codificador de prioridad y después analizaremos la interacción entre el controlador de prioridad de interrupción y la CPU.

Codificador de prioridad

El codificador de prioridad es un circuito que implanta la función de prioridad. La lógica del codificador de prioridad es que, si dos o más entradas

TABLA 11-2 Tabla de verdad de codificador de prioridad

Entradas				Salidas			Funciones booleanas
I_0	I_1	I_2	I_3	x	y	IST	
1	×	×	×	0	0	1	
0	1	×	×	0	1	1	$x = I_0'I_1'$
0	0	1	×	1	0	1	$y = I_0'I_1 + I_0'I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	×	×	0	

llegan al mismo tiempo, la entrada que tiene la prioridad mayor tendrá preferencia. La tabla de verdad de un codificador de prioridad de cuatro entradas se proporciona en la tabla 11-2. Las letras × en la tabla representan condiciones no significativas. La entrada I_0 tiene la prioridad mayor; por lo tanto, sin considerar el valor de las otras entradas, cuando esta entrada es 1, la salida genera una salida $xy = 00$. I_1 tiene el siguiente nivel de prioridad. La salida es 01 si $I_1 = 1$ siempre y cuando $I_0 = 0$, sin considerar los valores de las entradas de prioridad menor. Se genera la salida para I_2 sólo si las entradas de prioridad mayor son 0, y así sucesivamente hacia abajo en el nivel de prioridad. Se activa el estado de interrupción (*IST*) sólo si una o más entradas son iguales a 1. Si todas las entradas son 0, se desactiva *IST* a 0 y las otras salidas del codificador no se utilizan, por lo que se marcan con condiciones no significativas. Esto se debe a que la dirección de vector no se transfiere a la CPU cuando *IST* = 0. Las funciones booleanas listadas en la tabla especifican la lógica interna del codificador. Por lo general, una computadora tendrá más de cuatro fuentes de interrupción. Por ejemplo, un codificador de prioridad con ocho entradas generará una salida de tres bits.

La salida del codificador de prioridad se utiliza para que forme parte de la dirección de vector con cada fuente de interrupción. Puede asignarse cualquier valor a los otros bits de la dirección de vector. Por ejemplo, la dirección de vector puede formarse al agregarse ceros a las salidas x y y del codificador. Con esta selección se le asignan los números binarios 0, 1, 2, y 3 a los vectores de interrupción para los cuatros dispositivos de E/S.

Ciclo de interrupción

El flip-flop *IEN* de habilitación de interrupción que se muestra en la figura 11-14 puede activarse o desactivarse mediante instrucciones de programa. Cuando *IEN* se desactiva, la CPU no considera la solicitud de interrupción que proviene del *IST*. El bit *IEN* controlado por programa le permite al

programador elegir entre usar o no la opción de interrupción. Si se ha insertado una instrucción en el programa para desactivar *IEN*, significa que el usuario no desea que su programa se interrumpa. Una instrucción para activar *IEN* indica que la opción de interrupción se usará mientras corre el programa actual. La mayoría de las computadoras incluyen circuitería interna que desactiven *IEN* a 0 cada vez que el procesador reconoce una interrupción.

Al final de cada ciclo de interrupción la CPU comprueba *IEN* y la señal de interrupción de *IST*. Si alguna de ellas es igual a 0, el control continúa con la siguiente instrucción. Si ambas *IEN* e *IST* son iguales a 1, la CPU pasa a un ciclo de interrupción. Durante el ciclo de interrupción la CPU ejecuta la siguiente secuencia de microoperaciones:

$SP \leftarrow SP - 1$ Decrementador apuntador de pila
$M[SP] \leftarrow PC$ Salvar *PC* dentro de la pila
$INTACK \leftarrow 1$ Habilitar reconocimiento de interrupción
$PC \leftarrow VAD$ Transferir dirección de vector a *PC*
$IEN \leftarrow 0$ Deshabilitar interrupciones posteriores
ir a recupera la instrucción

La CPU salva la dirección de retorno en *PC* dentro de la pila. Después reconoce la interrupción al habilitar la línea *INTACK*. La unidad de prioridad de interrupción responde colocando un vector de interrupción único dentro del canal de datos de la CPU. La CPU transfiere la dirección de vector dentro del *PC* y desactivan *IEN* antes de pasar a la siguiente fase de búsqueda. Durante la siguiente fase de búsqueda, la instrucción que se leerá de la memoria será la que esté localizada en la dirección de vector.

Rutinas de programación

Un sistema de prioridad de interrupciones es una combinación de técnicas de circuitería y programación. Hasta aquí, hemos analizado los aspectos de la circuitería de un sistema de prioridad de interrupciones. La computadora también debe tener rutinas de programación para atender las solicitudes de interrupción y para controlar los registros de la circuitería de interrupción. La figura 11-15 muestra el programa que debe residir en la memoria para manejar el sistema de interrupciones. Cada dispositivo tiene su propio programa de servicio que puede alcanzarse mediante una instrucción de brinco (JMP) almacenada en la dirección asignada al vector. El nombre simbólico de cada rutina representa la dirección inicial del programa de servicios. La pila que se muestra en el diagrama se usa para almacenar la dirección de retorno después de cada interrupción.

programa de servicio

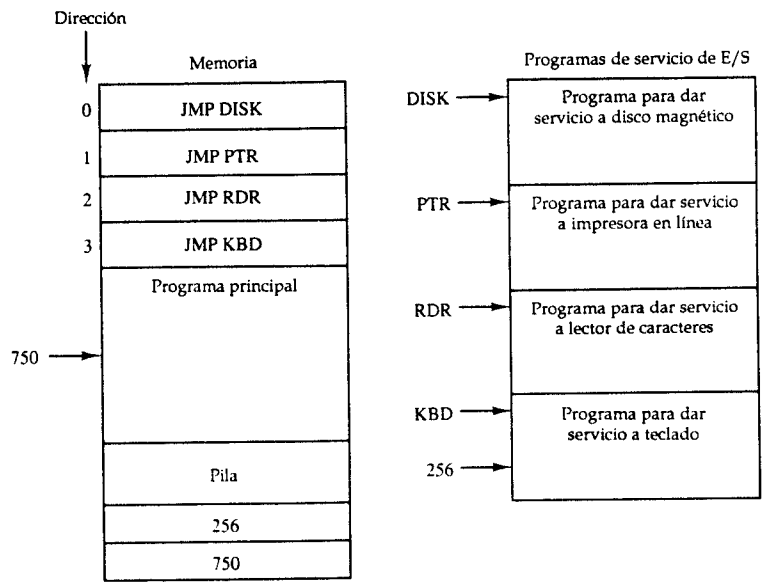


Figura 11-15 Programa almacenado en memoria para atender interrupciones.

Para proporcionar un ejemplo específico consideremos que el teclado activa su bit de interrupción mientras la CPU ejecuta la instrucción en la localidad 749 del programa principal. Al final del ciclo de instrucción, la computadora va a un ciclo de interrupción. Salva la dirección de retorno 750 en la pila y después acepta la dirección de vector 00000011 del canal y la transfiere al *PC*. La instrucción en la localidad 3 se ejecuta después, produciendo una transferencia de control a la rutina de interrupción KBD. Ahora supongamos que el disco activa su bit de interrupción cuando la CPU ejecuta la instrucción en la dirección 255 en el programa KBD. La dirección 256 se salva dentro de la pila y el control se transfiere al programa de servicio DISK. La última instrucción en cada rutina es un retorno de la instrucción de interrupción. Cuando termina el programa de servicio del disco, la instrucción de retorno se recupera de la pila y coloca 256 en el *PC*. Esto retorna el control a la rutina KBD para que continúe atendiendo el teclado. Al final del programa KBD, la última instrucción recupera la dirección de retorno de la pila y retorna el control al programa principal en la dirección 750. Por lo tanto, un dispositivo de prioridad mayor puede interrumpir a un dispositivo de prioridad menor. Se considera que el tiempo para dar servicio a una interrupción de alta prioridad es corto en comparación con la velocidad de

transferencia del dispositivo de baja prioridad, por lo que no ocurre ninguna pérdida de información.

Operaciones inicial y final

Cada rutina de servicio de interrupción debe tener un conjunto de operaciones inicial y final para controlar los registros en el sistema de circuitería de interrupción. Recuerde que la habilitación de interrupción *IEN* se desactiva al final de un ciclo de interrupción. Este flip-flop se debe activar de nuevo para habilitar solicitudes de interrupción de prioridad mayor, pero no antes de que se hayan deshabilitado las interrupciones de prioridad menor. La secuencia inicial de cada rutina de servicio de interrupción debe contener instrucciones para controlar la circuitería de interrupción de la siguiente manera:

1. Desactivar los bits del registro de máscara de prioridad inferior.
2. Desactivar el bit de estado de interrupción *IST*.
3. Salvar el contenido de los registros del procesador.
4. Activar el bit de habilitación de interrupción *IEN*.
5. Proceder con la rutina de servicio.

Los bits del registro de máscara de nivel inferior (incluye el bit de la fuente que interrumpió) se desactivan para evitar que estas condiciones habiliten la interrupción. Aunque a las fuentes de interrupción de prioridad menor se les asignan bits de número mayor en el registro de máscara, la prioridad puede cambiarse si se desea, porque el programa puede utilizar cualquier configuración de bits para el registro de máscara. El bit de estado de interrupción debe desactivarse para que pueda volverse a activar cuando ocurra una interrupción de prioridad mayor. El contenido de los registros de procesador se salvan porque puede necesitarlos el programa que se ha interrumpido después de que el control retorne a él. Después, se activa la habilitación de interrupción *IEN* para permitir que otras interrupciones (de prioridad mayor) y la computadora procedan a atender la solicitud de interrupción.

La secuencia final de cada rutina de servicio de interrupción debe contener instrucciones para controlar la circuitería de interrupción de la siguiente manera:

1. Activar el bit de habilitación de interrupción *IEN*.
2. Recuperar el contenido de los registros del procesador.
3. Desactivar el bit en el registro de interrupción que pertenece a la fuente que ha sido atendida.
4. Reactivar los bits de prioridad de nivel menor en el registro de cubierta.
5. Restablecer la dirección de retorno dentro del *PC* y establecer *IEN*.

Debe activarse el bit en el registro de interrupción que pertenece a la fuente de la interrupción para que esté disponible de nuevo cuando la fuente requiera otra interrupción. Los bits de prioridad menor en el registro de máscara (incluyendo el bit de la fuente que se está interrumpiendo) se activan para volver a habilitar sus interrupciones. El retorno al programa interrumpido se consigue al cargar la dirección de retorno en el *PC*. Nótese que la circuitería debe diseñarse para que no ocurran interrupciones mientras se ejecutan los pasos 2 al 5; de otra manera, puede perderse la dirección de retorno y la información de los registros de máscara y del procesador puede ser ambigua si se reconoce una instrucción mientras se ejecutan las operaciones en este paso. Por esta razón, inicialmente se desactiva *IEN* y más tarde se reactiva después de que se transfiere la dirección de retorno al *PC*.

Las operaciones final y inicial que acabamos de listar se denominan *operaciones generales de gestión* o *quehaceres domésticos*. No son parte del propio programa de servicio pero son esenciales para procesar interrupciones. Todas las operaciones generales pueden implantarse mediante programación. Esto se hace al insertar las instrucciones adecuadas al principio y al final de cada rutina de servicio. La circuitería puede hacer en forma automática algunas de las operaciones generales de gestión. La circuitería puede salvar dentro de la pila el contenido de los registros del procesador antes de transferir el control a la rutina de servicio. También se le pueden asignar otras operaciones iniciales y finales. De esta manera, es posible reducir el tiempo entre la recepción de una interrupción y la ejecución de las instrucciones que atienden la fuente de interrupción.

11-6 Acceso directo a memoria (DMA)

La transferencia de datos entre un dispositivo de almacenamiento rápido como un disco magnético y la memoria, con frecuencia está limitada por la velocidad de la CPU. Quitar la CPU de la trayectoria y permitir que el dispositivo periférico maneje en forma directa los canales de memoria mejoraría la velocidad de transferencia. Esta técnica de transferencia se llama acceso directo a memoria (*Direct Memory Access*, DMA). Durante una transferencia DMA, la CPU está inactiva y no tiene el control de los canales de memoria. Un controlador DMA funciona sobre los canales para manejar la transferencia en forma directa entre el dispositivo de E/S y la memoria.

La CPU puede colocarse en un estado inactivo de diversas maneras. Un método común que se usa con mucha frecuencia en los microprocesadores, es deshabilitar los canales mediante señales de control especiales. La figura 11-16 muestra dos señales de control en la CPU que facilitan la transferencia DMA. El controlador DMA utiliza la entrada de *solicitud de canal* (*BR*) para solicitar a la CPU que entregue el control de los canales. Cuando esta entrada está activa, la CPU termina la ejecución de la instrucción actual y coloca el canal de direcciones, el canal de datos, y las líneas de lectura y escritura

solicitud de canal

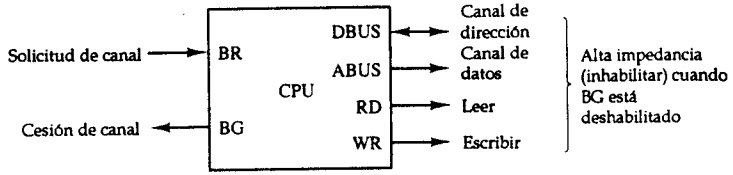


Figura 11-16 Señales del canal de CPU para transferencia DMA.

dentro de un estado de alta impedancia. El estado de alta impedancia es equivalente a un circuito abierto, lo que significa que la salida está desconectada y no tiene un significado lógico (véase la sección 4-3). La CPU activa la salida de *cesión de canal* (*BG*) para informar a la DMA externa que los canales están en estado de alta impedancia. La DMA que originó la solicitud de canal puede ahora tomar el control de los canales para conducir transferencias de memoria sin la intervención del procesador. Cuando la DMA termina la transferencia, deshabilita la línea de solicitud de canal. La CPU deshabilita la cesión del canal, toma el control de los canales y retorna a su operación normal.

cesión de canal

Cuando la DMA toma el control del canal del sistema, se comunica directamente con la memoria. La transferencia puede hacerse de varias maneras. En la *transferencia de DMA en ráfagas*, es una secuencia de bloque, que consiste en varias palabras de memoria, se transfiere en una ráfaga continua mientras el controlador DMA domina los canales de memoria. Este modo de transferencia se necesita para dispositivos rápidos como discos magnéticos, en donde la transmisión de datos no puede detenerse o hacerse lenta hasta que sea transferido todo un bloque. Una técnica alterna llamada *robo de ciclo*, permite al controlador DMA transferir una palabra de datos a la vez, después de lo cual debe retornar el control a los canales de la CPU. La CPU sólo retrasa su operación por un ciclo de memoria para permitir que la transferencia de E/S de memoria directa "robe" un ciclo de memoria.

transferencia en ráfaga

robo de ciclo

Controlador DMA

El controlador DMA necesita los circuitos usuales de una interface para comunicarse con la CPU y el dispositivo de E/S. Además necesita un registro de direccionamiento, un registro de cuenta de palabras, y un conjunto de líneas de direccionamiento. El registro de direccionamiento y las líneas de direccionamiento se utilizan para la comunicación directa con la memoria. El registro de cuenta de palabras especifica la cantidad de palabras que deben transferirse. La transferencia de datos puede hacerse directamente entre el dispositivo y la memoria bajo el control del DMA.

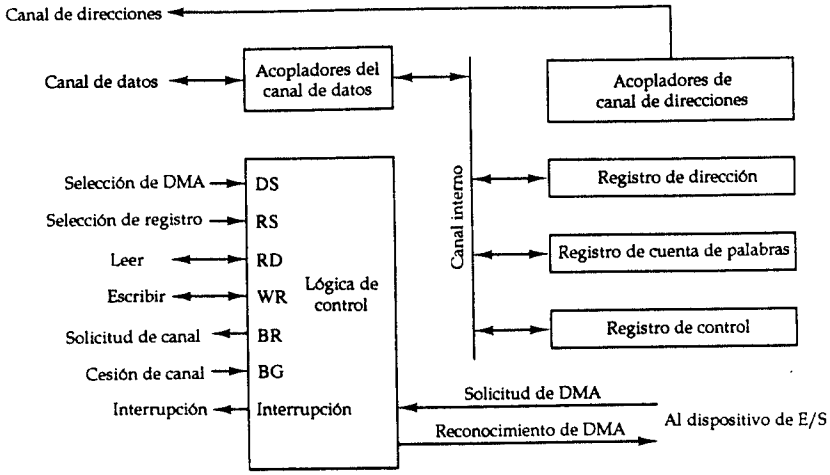


Figura 11-17 Diagrama de bloques de controlador DMA.

La figura 11-17 muestra el diagrama de bloque de un controlador de DMA típico. La unidad se comunica con la CPU mediante el canal de datos y las líneas de control. Los registros en el DMA se seleccionan mediante la CPU por medio del canal de datos al habilitar las entradas de selección de DMA (DS) y de selección de registros (RS). Las entradas lectura (RD) y escritura (write, WR) son bidireccionales. Cuando la entrada de cesión de canal (BG) es 0, la CPU puede comunicarse con los registros de DMA por medio del canal de datos para leer de o escribir en los registros DMA. Cuando *BG* = 1, la CPU ha entregado los canales y el DMA puede comunicarse directamente con la memoria al especificar una dirección en el canal de direcciones y activar el control *RD* o *WR*. DMA se comunica con el periférico externo mediante líneas de solicitud y reconocimiento al utilizar un procedimiento preestablecido de reconocimiento mutuo.

El controlador de DMA tiene tres registros: un registro de direccionamiento, un registro de cuenta de palabras y un registro de control. El registro de direccionamiento contiene una dirección para especificar la localidad deseada en la memoria. Los bits de direccionamiento van a través de búffers al canal de direcciones. El registro de direccionamiento se incrementa después que cada palabra se transfiere a la memoria. El registro de cuenta de palabras contiene la cantidad de palabras que se van a transferir. Este registro se decrementa en uno con cada transferencia de palabras y realiza una prueba interna en busca de cero. El registro de control especifica el modo de transferencia. Todos los registros en el DMA aparecen ante la CPU

como registros de interface de E/S. Por lo tanto, la CPU puede leer o escribir dentro de los registros DMA bajo el control de programa mediante el canal de datos.

Primero, la CPU inicializa el DMA. Después de eso, el DMA empieza y continúa la transferencia de datos entre la memoria y la unidad periférica hasta que se transfiere un bloque completo. El proceso de inicialización es esencialmente un programa que consiste en instrucciones de E/S que incluyen la dirección para seleccionar registros DMA particulares. La CPU inicializa el DMA al enviar la siguiente información por el canal de datos:

1. La dirección inicial del bloque de memoria en donde están disponibles los datos (para lectura) o donde se van a almacenar los datos (para escritura).
2. La cuenta de palabras, que es el número de palabras en el bloque de memoria.
3. Un control para especificar el modo de transferencia como de lectura o de escritura.
4. Un control para iniciar la transferencia DMA.

La dirección inicial se almacena en el registro de direccionamiento. La cuenta de palabras se almacena en el registro de cuenta de palabras y la información de control en el registro de control. Una vez que se inicializa el DMA, la CPU detiene la comunicación con el DMA, sólo que reciba una señal de interrupción o que desee comprobar cuántas palabras se han transferido.

Transferencia DMA

La posición del controlador DMA entre los otros componentes en un sistema de computadora se ilustra en la figura 11-18. La CPU se comunica con el DMA mediante los canales de dirección y de datos como con cualquier unidad de interface. El DMA tiene su propia dirección, la cual activa las líneas *DS* y *RS*. La CPU inicializa el DMA por medio del canal de datos. Una vez que el DMA recibe el comando de control de inicio, puede comenzar la transferencia entre el dispositivo periférico y la memoria.

Cuando el dispositivo periférico envía una solicitud al DMA, el controlador de DMA activa la línea *BR*, informando a la CPU que ceda los canales. La CPU responde con su línea *BG*, informando al DMA que sus canales están deshabilitados. Después el DMA pone el valor de su registro de direccionamiento dentro del canal de dirección, activa la señal *RD* o *WR* y envía un reconocimiento DMA al dispositivo periférico. Nótese que las líneas *RD* y *WR* en el controlador DMA son bidireccionales. La dirección de la transferencia depende del estado de la línea *BG*. Cuando *BG* = 0, *RD* y *WR* son líneas de entrada que permiten a la CPU comunicarse con los registros internos del DMA. Cuando *BG* = 1, *RD* y *WR* son líneas de salida del

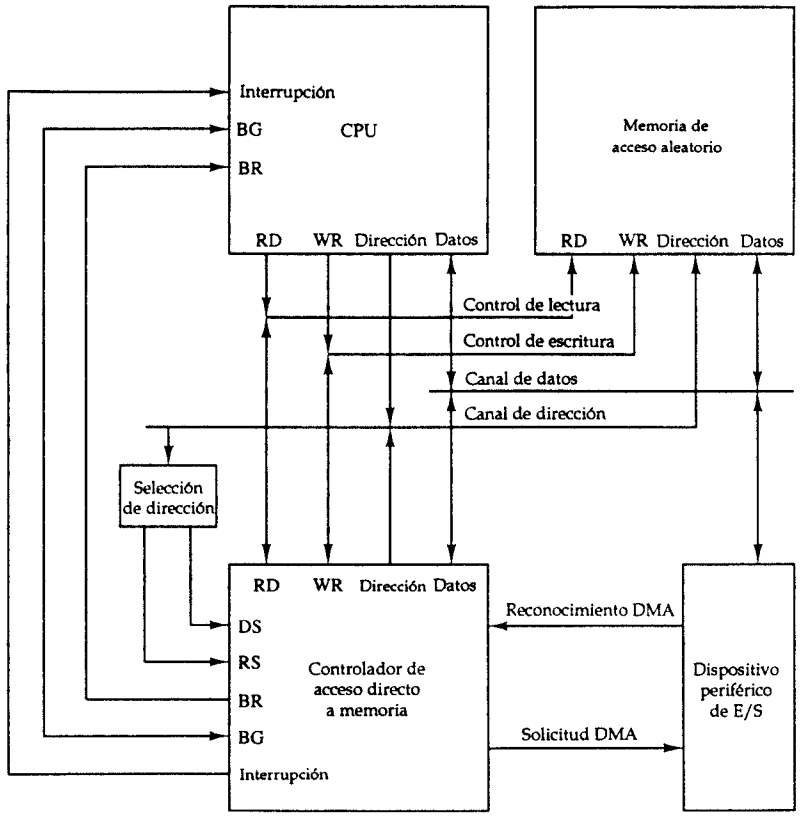


Figura 11-18 Transferencia de DMA en un sistema de computadora.

controlador DMA a la memoria de acceso aleatorio para especificar la operación de lectura o escritura para los datos.

Cuando el dispositivo periférico recibe un reconocimiento de DMA, coloca una palabra en el canal de datos (para escritura) o recibe una palabra del canal de datos (para lectura). Por lo tanto, el DMA controla las operaciones de lectura o escritura y proporciona la dirección para la memoria. En seguida, la unidad periférica puede comunicarse con la memoria por medio del canal de datos para una transferencia directa entre las dos unidades mientras la CPU está momentáneamente deshabilitada.

Por cada palabra que se transfiere, el DMA incrementa su registro de direccionamiento y decrementa su registro de cuenta de palabras. Si la

cuenta de palabras no alcanza cero, el DMA comprueba la línea de solicitud que proviene del periférico. Para un dispositivo de alta velocidad, la línea estará activa tan pronto como se termine la transferencia previa. Después se inicia una segunda transferencia, y el proceso continúa hasta que se ha transferido todo el bloque. Si la velocidad del periférico es más lenta, la línea de solicitud DMA puede venir un poco más tarde. En este caso, el DMA deshabilita la línea de solicitud de canal para que la CPU pueda continuar ejecutando su programa. Cuando el periférico solicita una transferencia, el DMA vuelve a solicitar los canales.

Si el registro de cuenta de palabras llega a cero, el DMA detiene cualquier transferencia posterior y retira su solicitud de canal. También informa a la CPU de la terminación mediante una interrupción. Cuando la CPU responde a la interrupción, lee el contenido del registro de cuenta de palabras. El valor cero de este registro indica que todas las palabras se transfirieron exitosamente. La CPU puede leer este registro en cualquier momento para comprobar la cantidad de palabras que ya se han transferido.

Un controlador DMA puede tener más de un canal. En este caso, cada canal tiene un par de señales de control para señales de solicitud y reconocimiento que están conectadas a dispositivos periféricos separados. Cada canal también puede tener su propio registro de direccionamiento y registro de cuenta de palabras dentro del controlador DMA. Puede establecerse una prioridad entre los canales para que los canales con alta prioridad sean atendidos antes que los canales de prioridad menor.

La transferencia DMA es muy útil en muchas aplicaciones. Se utiliza para transferencias rápidas de información entre discos magnéticos y memoria. También es útil para actualizar la pantalla en una terminal interactiva. De manera típica, una imagen de la pantalla de la terminal se conserva en la memoria y puede actualizarse bajo el control del programa. El contenido de la memoria puede transferirse a la pantalla en forma periódica, mediante una transferencia de DMA.

11-7 Procesador de entrada-salida (IOP)

En lugar que cada interface se comunique con la CPU, una computadora puede incorporar uno o más procesadores externos y asignarles la tarea de comunicarse directamente con todos los dispositivos E/S. Un procesador de entrada-salida (IOP), puede clasificarse como un procesador con capacidad de acceso directo a memoria que comunica con dispositivos de E/S. En esta configuración, el sistema de computadora puede dividirse en una unidad de memoria y varios procesadores que comprenden la CPU y uno o más IOP. Cada IOP atiende tareas de entrada y salida, relevando a la CPU de los "quehaceres domésticos" que involucran las transferencias de E/S. Un procesador que comunica con terminales remotas por teléfono y otros medios de comunicación en forma serial se llama un procesador de comunicación de datos (DCP).

procesamiento de E/S

El IOP es similar a una CPU, excepto que está diseñado para manejar los detalles del procesamiento de E/S. A diferencia del controlador de DMA, que debe ser inicializado por completo por la CPU, el IOP puede buscar en memoria y ejecutar sus propias instrucciones. Las instrucciones IOP están específicamente diseñadas para facilitar las transferencias de E/S. Además, el IOP puede ejecutar otras tareas de procesamiento, como aritmética, lógica, transferencia de control y traducción de código.

El diagrama de bloque de una computadora con dos procesadores se muestra en la figura 11-19. La unidad de memoria ocupa una posición central y puede comunicarse con cada procesador mediante acceso directo a memoria. La CPU es responsable del procesamiento de datos necesarios en la solución de tareas computacionales. El IOP proporciona una trayectoria para transferencia de datos con diversos dispositivos periféricos y la unidad de memoria. Por lo general, la CPU tiene la tarea de inicializar un programa de E/S. En lo sucesivo el IOP opera en forma independiente de la CPU y continúa transmitiendo datos de dispositivos externos y la memoria.

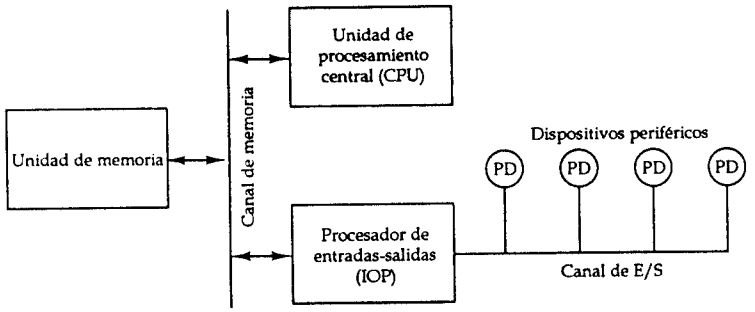


Figura 11-19 Diagrama de bloque de una computadora con procesador de E/S.

Los formatos de datos de los dispositivos periféricos difieren de los formatos de datos de la memoria y la CPU. El IOP debe estructurar palabras de datos de muchas fuentes diferentes. Por ejemplo, puede ser necesario tomar cuatro bytes de un dispositivo de entrada y guardarlos en una palabra de 32 bits antes de la transferencia a memoria. Los datos se reúnen en el IOP a la velocidad del dispositivo y a la capacidad de bits mientras la CPU ejecuta su propio programa. Después de que los datos se reúnen en una palabra de memoria, se transfieren del IOP directamente dentro de la memoria al "robar" un ciclo de memoria de la CPU. De igual manera, una palabra de salida transferida de la memoria al IOP se dirige del IOP al dispositivo de salida a la velocidad del dispositivo y a su capacidad de bits.

La comunicación entre el IOP y los dispositivos conectados a él es similar al método de transferencia de control del programa. La comunicación con la memoria es similar al método de acceso directo a memoria. La manera en la cual se comunican la CPU y el IOP depende del nivel de sofisticación incluido en el sistema. En computadoras a escala muy grande, cada procesador es independiente de los demás y cualquiera puede iniciar una operación. En la mayoría de sistemas computacionales, la CPU es el amo que dirige mientras el IOP es un procesador esclavo. Se asigna a la CPU la tarea de iniciar todas las operaciones, pero las instrucciones de E/S se ejecutan en el IOP. Las instrucciones de CPU proporcionan operaciones para iniciar una transferencia de E/S y también para probar condiciones de estado de E/S necesarias para tomar decisiones sobre varias actividades de E/S. A su vez, el IOP solicita la atención de la CPU mediante una interrupción y responde a la solicitudes de la CPU al colocar una palabra de estado en una localidad determinada de la memoria, que se examinará después, un programa de la CPU. Cuando se desea una operación de E/S, la CPU informa al IOP dónde encontrar el programa de E/S y después deja los detalles de la transferencia al IOP.

comandos

Las instrucciones que un IOP lee de la memoria en ocasiones se denominan *comandos*, para distinguirlos de las instrucciones que lee la CPU. De cualquier manera una instrucción y un comando tienen funciones similares. Los comandos son preparados por programadores experimentados y se almacenan en la memoria. Las palabras de comando constituyen el programa para el IOP. La CPU informa al IOP dónde encontrar los comandos en la memoria cuándo es hora de ejecutar el programa E/S.

Comunicación CPU-IOP

La comunicación entre la CPU y el IOP puede tomar diferentes formas, dependiendo de la computadora particular considerada. En muchos casos, la unidad de memoria actúa como un centro de mensajes en donde cada procesador deja información para el otro. Para apreciar la operación de un IOP típico, mostraremos un ejemplo específico del método con el que se comunican la CPU y el IOP. Este es un ejemplo simplificado que omite muchos detalles operativos para proporcionar un panorama de los conceptos básicos.

La secuencia de operaciones puede realizarse como se muestra en el diagrama de flujo de la figura 11-20. La CPU envía una instrucción para probar la trayectoria de IOP. El IOP responde al insertar una palabra de estado en la memoria para que la CPU la compruebe. Los bits de la palabra de estado indican la condición del IOP y el dispositivo de E/S. Por ejemplo una condición que sobrecarga de IOP, dispositivo ocupado con otra transferencia o dispositivo preparado para una transferencia de E/S. La CPU hace referencia a la palabra de estado en la memoria para decidir qué hacer enseguida. Si todo está en orden, la CPU envía la instrucción para comenzar

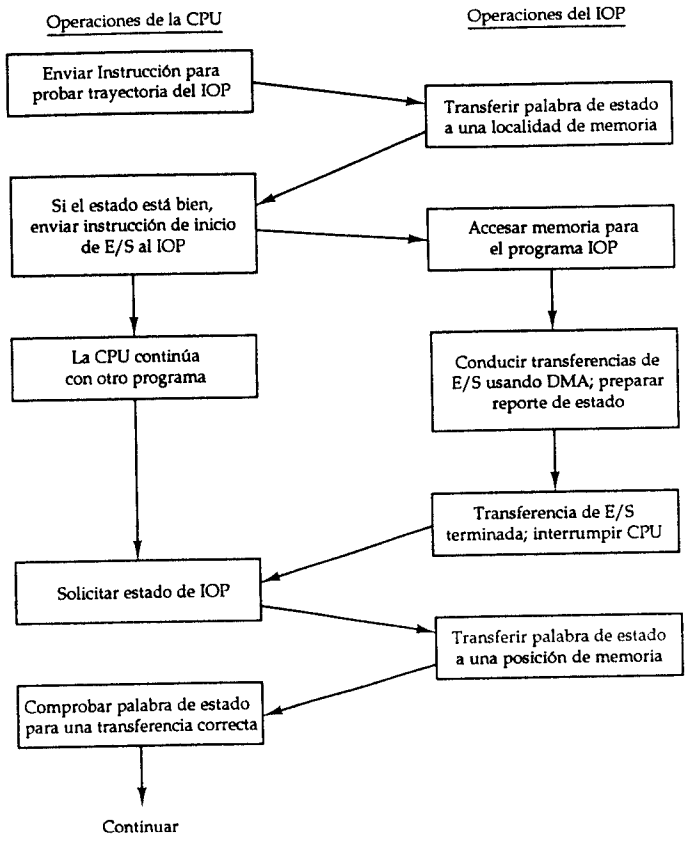


Figura 11-20 Comunicación CPU-IOP.

la transferencia de E/S. La dirección de memoria recibida con esta instrucción le dice al IOP en dónde encontrar su programa.

Ahora la CPU puede continuar con otro programa mientras el IOP está ocupado con el programa de E/S. Ambos programas hacen referencia a la memoria mediante una transferencia DMA. Cuando el IOP termina la ejecución de su programa, envía una solicitud de interrupción a la CPU. La CPU responde a la interrupción al emitir una instrucción para leer el estado del IOP. El IOP responde al colocar el contenido de su reporte de estado dentro de una localidad de memoria específica. La palabra de estado indica si ha terminado la transferencia o si ha ocurrido cualquier error durante ella. De

la inspección de los bits en la palabra de estado, la CPU determina si la operación de E/S ha terminado satisfactoriamente sin errores.

El IOP cuida todas las transferencias de datos entre varias unidades de E/S y la memoria mientras la CPU procesa otro programa. El IOP y la CPU compiten por el uso de la memoria, por lo que se limita el número de dispositivos que pueden estar en operación por el tiempo de acceso a la memoria. En la mayoría de los sistemas no es posible saturar la memoria mediante dispositivos de E/S porque la velocidad de estos es mucho menor que la de la CPU. Sin embargo, algunas unidades que son rápidas, como los discos magnéticos, pueden utilizar una apreciable cantidad de los ciclos de memoria disponibles. En ese caso, la velocidad de la CPU puede deteriorarse porque con frecuencia tendrá que esperar que el IOP conduzca transferencias de memoria.

Canal de E/S IBM 370

El procesador de E/S en la computadora IBM 370 se denomina *canal*. Una configuración de sistema de computadoras típico incluye varios canales y cada uno de ellos está conectado a uno o más dispositivos de E/S. Existen tres tipos de canales: multiplexor, selector y bloque multiplexor. El canal multiplexor puede conectarse a varios dispositivos de velocidad lenta y media y puede operar con varios dispositivos de E/S en forma simultánea. El canal selector está diseñado para manejar una operación de E/S a la vez y se usa normalmente para controlar un dispositivo de alta velocidad. El canal de bloque multiplexor combina las características de los canales multiplexor y selector. Proporciona una conexión a varios dispositivos de alta velocidad, pero todas las transferencias de E/S pueden conducirse en un bloque de datos completo a diferencia de un canal multiplexor, que sólo puede transferir un byte a la vez.

La CPU se comunica en forma directa con los canales por medio de líneas de control dedicadas e indirectamente por medio de áreas de almacenamiento reservadas en la memoria. La figura 11-21 muestra los formatos de palabra asociados con la operación del canal. El formato de instrucción de E/S tiene tres campos: código de operación, dirección del canal y dirección del dispositivo. El sistema de computadora puede tener varios canales y a cada uno se le asigna una dirección. De igual manera, cada canal puede estar conectado a varios dispositivos y a cada dispositivo se le asigna una dirección. El código de operación especifica una de ocho instrucciones E/S: iniciar E/S, iniciar envío rápido de E/S, probar E/S, borrar E/S, detener E/S, detener dispositivo, probar canal y almacenar identificación del canal. Los canales direccionados responden a cada una de las instrucciones de E/S y o las ejecutan. También establecen uno de cuatro códigos de condición en un registro de procesador llamado palabra de estado del procesador (PSW). La CPU puede comprobar el código de condición en la PSW para determinar el resultado de la operación de E/S. El significado de los cuatro códigos de

condición es diferente para cada instrucción de E/S. Pero, en general, especifican si el canal o el dispositivo están ocupados, si son operacionales o no, si hay interrupciones pendientes, si la operación de E/S comenzó exitosamente y si se almacenó una palabra de estado en la memoria por medio del canal.

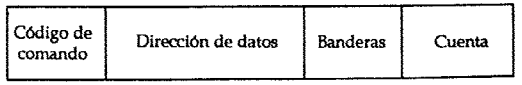
El formato de la palabra de estado del canal se muestra en la figura 11-21(b). Siempre está almacenado en la posición 64 en la memoria. El campo de clave es un mecanismo de protección utilizado para evitar el acceso no autorizado por parte de un usuario a la información que pertenece a otro usuario o al sistema operativo. El campo de dirección de la palabra de estado proporciona la dirección de la última palabra del comando utilizada por el canal. El campo de cuenta proporciona la cuenta residual cuando se terminó la transferencia. El campo de cuenta mostrará cero si la transferencia se terminó exitosamente. El campo de estado identifica las condiciones en el dispositivo y el canal y cualesquiera errores ocurridos durante la transferencia.

La diferencia entre las instrucciones iniciar E/S e iniciar envío rápido de E/S es que la última requiere menos tiempo de CPU para su ejecución, cuando el canal recibe una de estas dos instrucciones, hace referencia a la localidad 72 de la memoria para la dirección de la primera palabra de comando del canal (CCW). El formato de la palabra de comando del canal se muestra en la figura 11-21(c). El campo de dirección de datos especifica la primera dirección de un búffer de memoria y el campo de cuenta proporciona la cantidad de bits que participan en la transferencia. El campo de comando especifica una operación de E/S y los bits de bandera proporcionan

Figura 11-21 Relación de formatos de palabra. E/S de la IBM 370.

a) Formato de instrucciones de E/S

b) Formato de palabra de estado del canal



(c) Formato de palabra de comando del canal

información adicional para el canal. El campo de comando corresponde a un código de operación que especifica uno de seis tipos de operaciones E/S:

1. *Escribir.* Transferir datos de la memoria a un dispositivo de E/S.
2. *Leer.* Transferir datos de un dispositivo de E/S a la memoria.
3. *Leer en reversa.* Leer una cinta magnética cuando se mueve en reversa.
4. *Controlar.* Se usa para iniciar una operación que no implica transferencia de datos, como rebobinar una cinta o posicionar un mecanismo de acceso a disco.
5. *Detectar.* Le informa al canal que transfiera su palabra de estado de canal a la localidad de memoria 64.
6. *Transferir en el canal.* Se utiliza en lugar de una instrucción de brinco. Aquí el campo de dirección de datos especifica la dirección de la siguiente palabra de comando que va a ejecutar el canal.

Un ejemplo de un programa de canal se muestra en la tabla 11-3. Consta de tres palabras de comando. La primera produce una transferencia dentro de una cinta magnética de 60 bytes desde la memoria comenzando en la dirección 4000. Las siguientes dos palabras de comando ejecutan una función similar con una porción diferente de la memoria y de cuenta de bytes. Las seis banderas en cada palabra de control especifican ciertas interrelaciones entre las palabras de comando. La primera bandera se activa en 1 en la primera palabra de comando para especificar "encadenamiento de datos". Da como resultado la combinación de 60 bytes de la primera palabra de comando con los 20 bytes de su sucesor dentro de un registro de 80 bytes. Los 80 bytes se escriben en cintas sin ninguna separación o salto, aunque se utilizaron dos secciones de memoria. La segunda bandera se activa en 1 en la segunda palabra de comando para especificar "encadenamiento de comandos". Le informa al canal que la siguiente palabra de comando utilizará el mismo dispositivo de E/S, en este caso, la cinta. El canal le informa a la unidad de cinta que comience a insertar un intervalo de registro en la cinta y avance a leer la siguiente palabra de comando de la memoria. Después se escriben los 40 bytes de la tercera palabra de comando en una cinta como un registro separado. Cuando todas las banderas son igual a cero, significa terminar las operaciones de E/S para el dispositivo de E/S particular.

TABLA 11-3 Ejemplo de programa de canal de IBM-370

Comando	Dirección	Banderas	Cuenta
Escribir en cinta	4000	100000	60
Escribir en cinta	6000	010000	20
Escribir en cinta	3000	000000	40

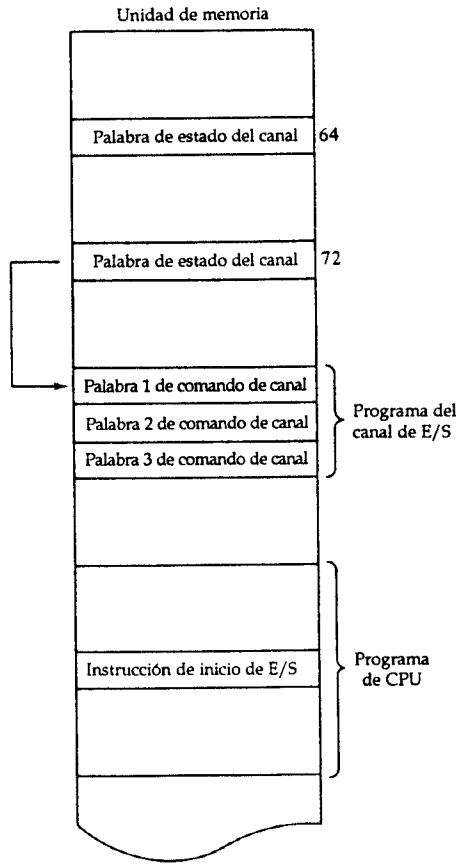


Figura 11-22 Posición de la información en la memoria operaciones de E/S de la IBM 370.

Un mapa de la memoria que muestra toda la información pertinente para el procesamiento de E/S se ilustra en la figura 11-22. La operación comienza cuando el programa de la CPU encuentra una instrucción de iniciar de E/S. El E/S va después a la posición de memoria 72 para obtener una palabra de dirección de canal. Esta palabra contiene la dirección inicial del programa de canal de E/S. En seguida, el canal avanza a ejecutar el programa especificado mediante las palabras de comando de canal. El canal construye una palabra de estado durante la transferencia y la almacena en

la posición 64. Cuando ocurre una interrupción, la CPU puede hacer referencia a la posición de memoria 64 para la palabra de estado.

IOP de Intel 8089

El procesador de E/S Intel 8089 está contenido en un encapsulado de circuito integrado de 40 bits. Dentro del 8089 hay dos unidades independientes llamadas *canales*. Cada canal combina las características generales de una unidad de procesador con las de un controlador de acceso directo a memoria. El 8089 está diseñado para funcionar como un IOP en un sistema de microcomputadora en donde el microprocesador 8086 se utiliza como CPU. La CPU 8086 inicia una operación de E/S al construir un mensaje en la memoria que describe la función que se va a ejecutar. El IOP 8089 lee el mensaje de la memoria, realiza la operación, y notifica a la CPU cuando ha terminado.

En contraste con el canal IBM 370, que sólo tiene seis comandos de E/S básicos, el IOP 8089 tiene 50 instrucciones básicas que pueden operar sobre bits individuales o sobre bytes, o sobre palabras de 16 bits. El IOP puede ejecutar programas de manera similar a la CPU, excepto que el conjunto de instrucciones se escoje en forma específica para proporcionar un procesamiento eficiente de entrada-salida. El conjunto de instrucciones incluye instrucciones de transferencia de datos generales, operaciones aritméticas y lógicas básicas, operaciones de transferencia condicional e incondicional y posibilidades de llamar y retornar subrutina. El conjunto también incluye instrucciones especiales para iniciar transferencias de DMA y emitir solicitudes de interrupción a la CPU. Proporciona una transferencia de datos eficiente entre dos componentes conectados al canal de sistema, como E/S a memoria, memoria a memoria o E/S a E/S.

Un sistema de microcomputadora que utiliza el par de circuitos integrados 8086/8089 se muestra en la figura 11-23. El 8086 funciona como la CPU y el 8089 como el IOP. Las dos unidades comparten una memoria común mediante un controlador de canal conectado a un canal de sistema, que Intel denomina "multicanal". El IOP utiliza un canal local para comunicarse con diferentes unidades de interface conectadas a dispositivos de E/S. La CPU comunica con el IOP al habilitar la línea de *atención al canal*. La línea de *selección* la utiliza la CPU para seleccionar uno de los dos canales en el 8089. El IOP obtiene la atención de la CPU al enviar una solicitud de interrupción.

La CPU y el IOP se comunican uno con el otro al escribir mensajes mutuos en la memoria del sistema. La CPU prepara el área de mensaje y la señala el IOP al habilitar la línea de atención del canal. El IOP lee el mensaje, ejecuta las funciones de E/S requeridas y ejecuta el programa de canal conveniente. Cuando el canal ha terminado su programa, emite una solicitud de interrupción a la CPU.

El esquema de comunicación consta de secciones de programa llamadas "bloques", las cuales se almacenan en la memoria como se muestra en la figura 11-24. Cada bloque contiene información de control y parámetros, al igual que un apuntador de dirección a su bloque sucesor. La dirección del bloque de control se pasa a cada canal IOP durante la inicialización. La bandera ocupado indica si el IOP está ocupado o preparado para ejecutar una nueva operación de E/S. La CPU especifica una palabra de comando de canal (CCW) para indicar el tipo de operación requerida del IOP. La CCW

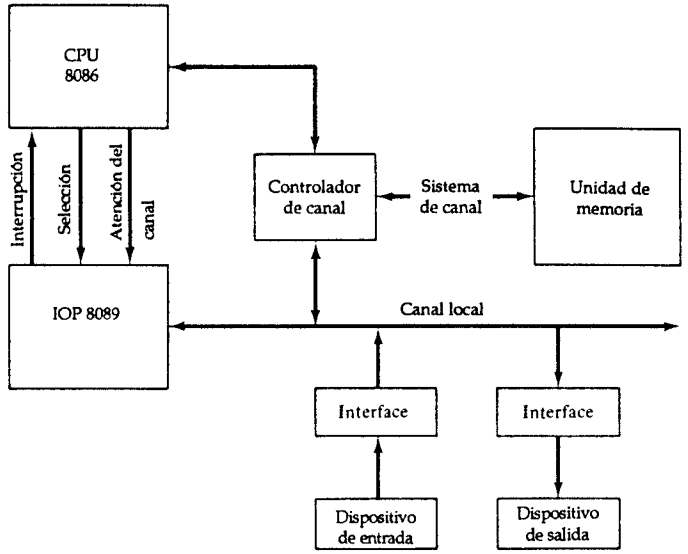


Figura 11-23 Diagrama de bloque del sistema microprocesador Intel 8086/8089.

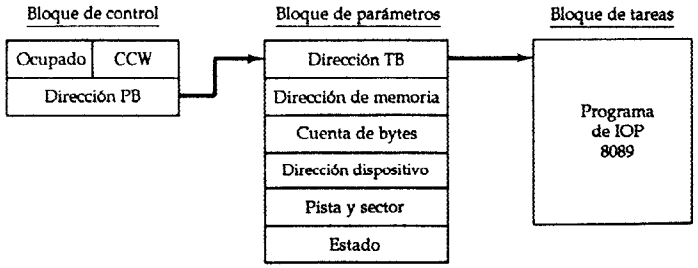


Figura 11-24 Posición de la información en memoria para operaciones de E/S en el sistema de microcomputadora Intel 8086/8089.

en el 8089 no tiene el mismo significado que la palabra de comando en el canal de IBM. Aquí, la CCW se parece más a una instrucción de E/S que especifica una operación para el IOP, como iniciar operación, suspender operación, reanudar operación, y parar programa de E/S. El bloque de parámetros contiene datos variables que el programa IOP debe utilizar para realizar su tarea. El bloque de tareas contiene el programa real que se va a ejecutar en el IOP.

La CPU y el IOP trabajan juntos por medio de los bloques de control y de parámetros. La CPU obtiene el uso de la memoria compartida después de comprobar la bandera de ocupado para asegurarse que el IOP está disponible. Después, la CPU llena la información en el bloque de parámetros y escribe un comando de "comenzar operación" en la CCW. Después de que se han inicializado bloques de comunicación, la CPU habilita la señal de atención del canal para informar al IOP que comience su operación de E/S. Después, la CPU continúa con otro programa. El IOP responde a la señal de atención del canal al colocar la dirección del bloque de control dentro de su contador de programa. El IOP hace referencia al bloque de control y activa la bandera de ocupado. Entonces checa la operación en la CCW. Se transfieren las direcciones PB (bloque de parámetros) y TB (bloque de tareas) dentro de los registros IOP internos. El IOP comienza a ejecutar el programa en el bloque de tareas utilizando la información en el bloque de parámetros. Las entradas en el bloque de parámetros depende del dispositivo de E/S. Los parámetros listados en la figura 11-24 son convenientes para transferencia de datos hacia o desde un disco magnético. La memoria de dirección especifica la dirección inicial de un buffer de memoria. La cuenta de bytes proporciona la cantidad de bytes que se van a transferir. La dirección del dispositivo especifica el dispositivo de E/S particular que se va a usar. Los números del canal y sector ubican los datos sobre el disco. Cuando se termina la operación de E/S, el IOP almacena sus bits de estado en la localidad de la palabra de estado del bloque de parámetros e interrumpe la CPU. La CPU puede hacer referencia a la palabra de estado para comprobar si la transferencia se ha completado satisfactoriamente.

11-8 Comunicación serial

Un procesador de comunicación de datos es un procesador de E/S que distribuye y recoge datos de muchas terminales remotas conectadas mediante el teléfono y otras líneas de comunicación. Es un procesador de E/S especializado diseñado para comunicar directamente con redes de comunicación de datos. Una red de comunicación puede contener cualquiera de una amplia variedad de dispositivos, como impresoras, dispositivos interactivos de exhibición visual, sensores digitales o una opción de computación remota. Con el uso de un procesador de comunicación de datos, la computadora puede dar servicio a fragmentos de cada demanda de red de manera intercalada y

procesador de comunicación de datos

módem

por lo tanto tener un desempeño aparente de servir a muchos usuarios a la vez. De esta manera, la computadora puede operar eficientemente en un ambiente de tiempo compartido.

La diferencia más notable entre un procesador de E/S y un procesador de comunicación de datos es la manera en la que el procesador se comunica con los dispositivos de E/S. Un procesador de E/S se comunica con los periféricos a través de un canal de E/S común que consta de muchas líneas de datos y de control. Todos los periféricos comparten el canal común y lo utilizan para transferir información hacia y desde el procesador de E/S. Un *procesador de comunicación de datos* se comunica con cada terminal por medio de un solo par de cables. La información de control y de datos se transmite de manera serial con el resultado de que la velocidad de transferencia es mucho más lenta. La tarea del procesador de comunicación de datos es transmitir y recoger información digital hacia y desde cada terminal, determinar si la información es de datos o de control y responder a todas las solicitudes de acuerdo a procedimientos establecidos con anticipación. Es obvio que el procesador también debe comunicarse con la CPU y la memoria de la misma manera que cualquier procesador de E/S.

La manera en que están conectadas las terminales remotas a un procesador de comunicación de datos es mediante líneas telefónicas u otras opciones de comunicación privadas o públicas. Como las líneas de comunicación estaban diseñadas originalmente para la comunicación de canal y las computadoras se comunican en términos de señales digitales, debe utilizarse alguna forma de conversión. Los convertidores se denominan *conjuntos de datos, acopladores acústicos* o *módems* (de "modulador-demodulador"). Un módem convierte las señales digitales en tonos de audio que se transmiten por líneas telefónicas y también convierten tonos de audio de la línea a señales digitales para el uso de la máquina. Se utilizan varios esquemas de modulación al igual que diferentes grados de medios de comunicación y velocidades de transmisión. Una línea de comunicación puede estar conectada a una interface síncrona o asíncrona, dependiendo del método de transmisión de la terminal remota. Una interface asíncrona recibe datos seriales con bits de inicio y paro en cada carácter, como se muestra en la figura 11-7. Este tipo de interface es similar a la unidad de interface de comunicación asíncrona presentada en la figura 11-8.

La transmisión síncrona no utiliza los bits de inicio y paro para delimitar los caracteres y por lo tanto usa el enlace de comunicación de manera más eficiente. Los dispositivos de alta velocidad utilizan la transmisión síncrona para obtener esta eficiencia. Los módems utilizados en la transmisión síncrona tienen relojes internos que se inicializan a la frecuencia que se están transmitiendo los bits en la línea de comunicación. Para una operación adecuada, se necesita que los relojes en los módems transmisor y receptor estén sincronizados en todas las ocasiones. Sin embargo, la línea de comunicación contiene sólo los bits de datos a partir de los cuales debe obtenerse la información de reloj. La sincronización de la secuencia la consigue el

módem receptor de las transiciones de señal que ocurren en los datos recibidos. Cualquier cambio de secuencia que puede ocurrir entre los relojes transmisor y receptor se ajusta continuamente al mantener el reloj receptor en la frecuencia del flujo de bits que llega. El modem transfiere los datos recibidos junto con el reloj a la unidad de interface. La interface o terminal en el lado transmisor también utiliza la información de reloj de su módem. De esta manera, se mantiene la misma velocidad de bits en el transmisor y en el receptor.

A diferencia de la transmisión asíncrona, en la cual cada carácter puede enviarse en forma separada con sus propios bits de inicio y paro, la transmisión síncrona debe enviar un mensaje continuo para mantener la sincronización. El mensaje consta de un grupo de bits transmitidos en forma secuencial como un bloque de datos. Todo el bloque se transmite con caracteres de control especiales al comienzo y al final del bloque. Los caracteres de control al inicio del bloque proporcionan la información necesaria para separar los bits que llegan en caracteres individuales.

Una de las funciones del procesador de comunicación de datos es verificar la presencia de errores de transmisión. Puede detectarse un error al comprobar la paridad en cada carácter recibido. Otro procedimiento utilizado en terminales asíncronas en las que interviene un operador humano es escuchar el eco de los caracteres. El procesador reconoce los caracteres transmitidos del teclado a la computadora y los retransmite a la impresora terminal. El operador debe darse cuenta de que ha ocurrido un error durante la transmisión si el carácter impreso no es igual a la tecla que se ha oprimido.

transferencia de bloque

En la transmisión síncrona, en la cual se transmite un bloque de caracteres completo, cada carácter tiene un bit de paridad que debe verificar el receptor. Después de que se envía el bloque completo, el transmisor envía un carácter más que constituye una paridad sobre la longitud del mensaje. Este carácter se llama una comprobación de redundancia longitudinal (*longitudinal redundancy check*, LRC) y es la acumulación de las OR exclusivas de todos los caracteres recibidos. La estación receptora calcula la LRC conforme recibe caracteres y la compara con la LRC transmitida. Las LRC calculadas deben ser iguales para que los mensajes no contengan error. Si el receptor encuentra un error en el bloque transmitido, le informa a quien envía que retransmita el mismo bloque de nuevo. Otro método utilizado para comprobar errores en la transmisión es la comprobación de redundancia

CRC

cíclica (*ciclyc redundancy check*, CRC). Este es un código de polinomio que se obtiene de los bits de mensaje al pasarlos a través de un registro de corrimiento retroalimentado que contiene varias compuertas OR exclusivas. Este tipo de código es conveniente para detectar errores de ráfaga que ocurren en el canal de comunicación.

Pueden transmitirse datos entre dos puntos en tres modos diferentes: símplex, semidúplex o dúplex completo. Una línea *símplex* transmite información sólo en una dirección. Este modo se utiliza rara vez en comunicación de datos porque el receptor no puede comunicarse con el transmisor para

indicarle la aparición de errores. Los ejemplos de transmisión símplex son la programación de radio y televisión.

Un sistema de transmisión *semidúplex* es uno que puede transmitir en ambas direcciones pero sólo pueden transmitirse los datos en una dirección a la vez. Se necesitan un par de cables para este modo. Una situación común es la de un modem que actúa como el transmisor y otro como receptor. Cuando termina la transmisión en una dirección, el papel de los modems se invierte para habilitar la transmisión en sentido opuesto. El tiempo requerido para cambiar una línea semidúplex de una dirección a otra se llama tiempo de vuelta.

dúplex completo

Una transmisión *dúplex completa* puede enviar y recibir datos en ambas direcciones simultáneamente. Esto puede lograrse mediante un enlace de cuatro cables con un par de líneas diferentes dedicadas a cada dirección de transmisión. Alternativamente, un circuito de dos líneas puede soportar comunicación dúplex completa si el espectro de frecuencia se subdivide en dos bandas de frecuencia no sobrepuestas para crear canales de recepción y transmisión separados en el mismo par de líneas físico.

Las líneas de comunicación, los módems y otro equipo utilizado en la transmisión de información entre dos o más estaciones se denomina *enlace de datos*. La transferencia de información ordenada en un enlace de datos se

protocolo

consigue mediante un *protocolo*. Un protocolo de control de enlace de datos es un conjunto de reglas que siguen las computadoras y terminales interconectadas para asegurar la transferencia de información ordenada. El propósito de un protocolo de enlace de datos es establecer y terminar una conexión entre dos estaciones, identificar al transmisor y al receptor, asegurarse que todos los mensajes pasen en forma correcta sin errores y manejar todas las funciones de control implícitas en una secuencia de transferencias de datos. Los protocolos se dividen en dos categorías principales, de acuerdo a las técnicas de ordenamiento de mensajes. Estos son el protocolo orientado a caracteres y el protocolo orientado a bits.

Protocolo orientado a caracteres

El protocolo orientado a caracteres se basa en el código binario de un conjunto de caracteres. El código que se usa con mayor frecuencia es el ASCII (*código estándar norteamericano para intercambio de información*). Es un código de 7 bits en el que se usa un octavo bit para paridad. El código tiene 128 caracteres, de los cuales 95 son gráficos y 33 son de control. Los caracteres gráficos incluyen las letras mayúsculas y minúsculas, los diez números y una variedad de símbolos especiales. En la tabla 11-1 puede encontrarse una lista de los caracteres ASCII. Los caracteres de control se utilizan para dirigir datos, arreglar la prueba en un formato deseado y para la distribución de la página impresa. Los caracteres que controlan la transmisión se llaman caracteres de *control de comunicación*. Estos caracteres se listan en la tabla 11-4. Cada carácter tiene un código de 7 bits y se hace referencia a él mediante un símbolo de tres

TABLA 11-4 Caracteres ASCII de control de comunicación

Código	Símbolo	Significado	Función
0010110	SYN	Inactivo síncrono	Establece sincronismo
0000001	SOH	Inicio de encabezado	Encabezado de mensaje de bloque
0000010	STX	Inicio de texto	Antecede un bloque de texto
0000011	ETX	Fin de texto	Termina un bloque de texto
0000100	EOT	Fin de transmisión	Concluye una transmisión
0000110	ACK	Reconocimiento	Reconocimiento afirmativo
0010101	NAK	Reconocimiento negativo	Reconocimiento negativo
0000101	ENQ	Consulta	Consulta si la terminal está encendida
0010111	ETB	Fin de bloque de transmisión	Fin de bloque de datos
0010000	DLE	Escape de enlace de datos	Carácter de control especial

letras. El papel de carácter en el control de la transmisión de datos se explica en forma breve en la columna de función de la tabla.

carácter SYN

El carácter SYN sirve como un agente de sincronización entre el transmisor y el receptor. Cuando se utiliza el código ASCII de 7 bits con un bit de paridad impar en la posición más significativa, el carácter SYN asignado tiene el código de 8 bits 00010110 que tiene la propiedad que, ante la ocurrencia de un corrimiento circular, se repite a sí mismo sólo después de un ciclo de 8 bits completo. Cuando el transmisor empieza a enviar caracteres de 8 bits, envía primero unos cuantos caracteres, y después envía el mensaje real. La serie continua de bits iniciales que acepta el receptor se comprueba en busca de un carácter SYN. En otras palabras, con cada pulso de reloj, el receptor comprueba los últimos 8 bits recibidos. Si no coinciden los bits del carácter SYN el receptor acepta el siguiente bit y rechaza el bit anterior de orden superior y vuelve a comprobar los últimos 8 bits recibidos en busca de un carácter SYN. Esto se repite después de cada pulso de reloj y después de cada bit recibido hasta que se reconoce un carácter SYN. Una vez que se detecta un carácter SYN, el receptor ha "armado" un carácter. De ahí en adelante el receptor cuenta cada 8 bits y los acepta como un sólo carácter. Por lo general, el receptor comprueba dos caracteres SYN consecutivos para que no exista la duda que el primero ocurrió como resultado de una señal de ruido en la línea. Además, cuando el transmisor está desocupado y no tiene ningunos caracteres de mensaje que enviar, emplea una serie continua de caracteres SYN. El receptor reconoce estos caracteres como una condición para sincronizar la línea y pasa a su estado inactivo síncrono. En este estado, las dos unidades son síncronas en bits y caracteres, aunque no se comunique información significativa.

Los mensajes se transmiten por el enlace de datos con un formato establecido que consiste en un campo de encabezado, un campo de texto y un campo de comprobación de error. Un formato de mensaje típico para un protocolo orientado a caracteres se muestra en la figura 11-25. Los dos caracteres SYN aseguran la sincronización adecuada al inicio del mensaje. Después de los caracteres SYN está el encabezado, que comienza con carácter SOH (inicio de encabezado). El encabezado consiste en información de dirección y control. El carácter STX termina el encabezado y significa el comienzo de la transmisión del texto. La parte de texto del mensaje es variable en longitud y puede contener cualesquiera caracteres ASCII, excepto los caracteres de control de comunicación. El campo de texto termina con el carácter ETX. El último campo es un carácter de comprobación de bloque (*block check character*, BCC) utilizado para la comprobación de error. Por lo regular, es una comprobación de redundancia longitudinal (*longitudinal redundancy check*, LRC) o una comprobación de redundancia cíclica (*cyclic redundancy check*, CRC).

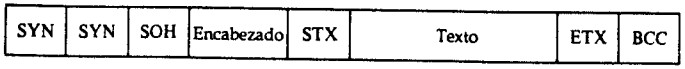


Figura 11-25 Formato de mensaje típico para protocolo orientado a caracteres.

El receptor acepta el mensaje y calcula su propia BCC. Si la BCC transmitida no coincide con la BCC calculada por el receptor, el receptor responde con un carácter de reconocimiento negativo (NAK). Después el mensaje se retransmite y se vuelve a comprobar. Lo normal es que se intente retransmitir varias veces antes de que se considere que la línea está defectuosa. Cuando la BCC transmitida coincide con la que calcula el receptor, la respuesta es un reconocimiento positivo utilizando el carácter ACK.

Ejemplo de transmisión

Para apreciar la función de un procesador de comunicación de datos, veamos mediante un ejemplo específico el método con el cual se comunican una terminal y el procesador. La comunicación con la unidad de memoria y la CPU es similar a cualquier procesador de E/S.

Un mensaje típico que podría enviarse de una terminal al procesador se lista en la tabla 11-5. Al observar este mensaje se descubre que hay varios caracteres de control que se usan para la formación del mensaje. Cada carácter, incluyendo los caracteres de control, se transmite en forma serial como un código binario de 8 bits que consta de un código ASCII de 7 bits más un bit de paridad impar en la octava posición más significativa. Se utilizan los dos caracteres SYN para sincronizar el receptor y el transmisor. El encabezado comienza con el carácter SOH y continúa con dos caracteres que especifican la dirección de la terminal. En este ejemplo particular, la dirección es T4, pero en general, pueden tener cualquier conjunto de dos o más caracteres gráficos. El carácter STX termina el encabezado y comunica

TABLA 11-5 Transmisión típica de una terminal a procesador

Código	Símbolo	Comentarios
0001 0110	SYN	Primer carácter sincronizado
0001 0110	SYN	Segundo carácter sincronizado
0000 0001	SOH	Inicio de encabezado
0101 0100	T	La dirección de la terminal es T4
0011 0100	4	
0000 0010	STX	Inicio de transmisión de texto
0101 0010		
0100 0101	Solicitud de saldo de la cuenta No. 1234	El texto enviado es una solicitud para responder con el saldo de la cuenta 1234
.		
.		
.		
1011 0011		
0011 0100		
1000 0011	ETX	Fin de transmisión de texto
0111 0000	LRC	Carácter de paridad longitudinal

el comienzo de la transmisión de texto. Los datos de texto que nos interesan aquí son "solicitar el balance la cuenta 1234". Los caracteres individuales para este mensaje no se listan en la tabla porque ocuparían demasiado espacio. Sin embargo, debe recordarse que cada carácter en el mensaje tiene un código de 8 bits y que cada bit se transmite en forma serial. El carácter de control ETX significa la terminación de los caracteres de texto. El siguiente carácter después de ETX es una comprobación de redundancia longitudinal (LRC). Cada bit en este carácter es un bit de paridad calculado de todos los bits en la misma columna en la sección de código de la tabla.

El procesador de comunicación de datos recibe este mensaje y avanza a analizarlo. Usa la terminal T4 y almacena el texto asociado con el mensaje. Mientras recibe los caracteres, el procesador comprueba la paridad en cada carácter y también calcula la paridad longitudinal. La LRC calculada se compara con el carácter LRC recibido. Si los dos coinciden, se envía de regreso a la terminal un reconocimiento positivo. Si no coinciden se retorna a la terminal un reconocimiento negativo (NAK) que iniciaría una retransmisión del mismo bloque. Si el procesador encuentra que el mensaje no tiene errores, lo transfiere dentro de la memoria e interrumpe la CPU. Cuando la CPU reconoce la interrupción, analiza el mensaje y prepara un mensaje de texto para responder a la solicitud. La CPU envía una instrucción al procesador de comunicación de datos para que envíe el mensaje a la terminal.

En la tabla 11-6 se lista una respuesta típica del procesador a la terminal. Después de dos caracteres SYN, el procesador reconoce el mensaje previo con un carácter ACK. La línea continua inactiva con un carácter SYN

TABLA 11-6 Trasmisión típica de procesador a terminal

Código	Símbolo	Comentarios
0001 0110	SYN	Primer carácter sincronizado
0001 0110	SYN	Segundo carácter sincronizado
1000 0110	ACK	El procesador reconoce el mensaje previo
0001 0110	SYN	La línea está inactiva
.	.	
.	.	
.	.	
0001 0110	SYN	La línea está inactiva
0000 0001	SOH	Inicio de encabezado
0101 0100	T	La dirección de la terminal es T4
0011 0100	4	
0000 0010	STX	Inicio de transmisión de texto
1100 0010		
1100 0001	El saldo es $ 100.00	El texto enviado es una respuesta de la computadora proporcionando el saldo de la cuenta
.		
.		
.		
1011 0000		
1000 0011	ETX	Fin de transmisión de texto
1101 0101	LRC	Carácter de paridad longitudinal

en espera de la respuesta. El procesador arregla en el formato apropiado el mensaje recibido de la CPU al insertar los caracteres de control necesarios antes y después del texto. El mensaje tiene el encabezado SOH y la dirección de la terminal T4. El mensaje de texto informa a la terminal que el balance es $100. Se calcula un carácter LRC y se envía a la terminal. Si la terminal responde con un carácter NAK, el procesador retransmite el mensaje.

Mientras el procesador cuida esta terminal, también está ocupado procesando otras terminales. Como los caracteres se reciben en forma serial, se necesita cierta cantidad de tiempo para recibir y reunir un carácter de 8 bits. Durante este tiempo el procesador está multiplexando todas las otras líneas de comunicación y atiende a cada una por turno. La velocidad de las terminales remotas es extremadamente lenta en comparación con la velocidad del procesador. Esta propiedad permite multiplexar a muchos usuarios para conseguir una mayor efectividad en un sistema de tiempo compartido. También permite que muchos usuarios operen simultáneamente mientras se atiende a cada uno de ellos a velocidades comparables a la respuesta humana normal.

Transparencia de datos

El protocolo orientado a caracteres se desarrolló originalmente para comunicarse con teclados, impresora y dispositivos de exhibición visual que utilizan

sólo caracteres alfanuméricos. Conforme se amplió el campo de comunicación de datos, se hizo necesario transmitir información binaria que no fuera texto ASCII. Esto sucede, por ejemplo cuando dos computadoras remotas envían programas y datos una a la otra por un canal de comunicación. Un patrón de bits arbitrario en el mensaje del texto se convierte en un problema en el protocolo orientado a caracteres. Esto se debe a que el receptor interpretará de manera errónea cualquier patrón de 8 bits que pertenezca a un carácter de control de comunicación. Por ejemplo, si los datos binarios en la parte del texto del mensaje tiene el patrón de 8 bits 10000011, el receptor interpretará éste como un carácter ETX y considerará que se ha alcanzado el fin del campo de texto. Cuando la longitud de la parte del texto del mensaje es variable y contiene bits que se van a tratar sin ninguna referencia a algún código particular, se dice que contienen datos transparentes. Esta característica requiere que la lógica de reconocimiento de caracteres del receptor se apague para que los patrones de datos en el campo de texto no se interpreten accidentalmente como información de control de comunicación.

carácter DLE

La transferencia de datos se consigue en los protocolos orientados a caracteres al insertar un carácter *DLE* (*data link escape*, escape de enlace de datos) antes de cada carácter de control de comunicación. Por lo tanto, se detecta el comienzo de un encabezado del carácter doble DLE SOH, y el campo de texto se termina con el carácter doble DLE ETX. Si ocurre el patrón de bits DLE 00010000 en la parte de texto del mensaje, el transmisor inserta otro patrón de bits DLE después de él. El receptor quita todos los caracteres DLE y después comprueba el siguiente patrón de 8 bits. Si es otro patrón DLE, el receptor lo considera como parte del texto y continúa recibiéndolo. De otra manera, el receptor toma el siguiente patrón de 8 bits como un carácter de control de comunicación.

Lograr la transparencia de datos mediante el carácter DLE no es eficiente y resulta complicada de implantar. Por lo tanto, se han desarrollado otros protocolos para hacer más eficiente la transmisión de datos transparentes. Un protocolo utilizado por Digital Equipment Corporation emplea un campo de cuenta de bytes que proporciona la cantidad de bytes en el mensaje que sigue. Después el receptor debe contar la cantidad de bytes recibidos para llegar al final del campo de texto. El protocolo que se ha usado con mayor frecuencia para resolver el problema de transparencia (y otros problemas asociados con el protocolo orientado a caracteres) es el protocolo orientado a bits.

Protocolo orientado a bits

El protocolo orientado a bits no utiliza caracteres en su campo de control y es independiente de cualquier código particular. Permite la transmisión de un flujo de bits serial de cualquier longitud sin implicar límites de caracteres. Los mensajes se organizan en un formato especial llamado cuadro. Además

del campo de información, un cuadro contiene los campos de dirección, control y comprobación de error. Los límites del cuadro los determina un número especial de 8 bits llamado bandera. El SDLC (synchronous data link control, control de enlace de datos síncrono) utilizado por IBM. El HDLC (*high-level data link control*, control de enlace de datos de alto nivel) adoptado por la International Standards Organization, y el ADCCP (*advanced data communication control procedure*, procedimiento avanzado de control de comunicación de datos) adoptado por el American National Standards Institute son ejemplos de protocolos orientados a bits.

Cualquier enlace de comunicación de datos implica al menos dos estaciones participantes. La estación que tiene la responsabilidad para el enlace de datos y que emite los comandos para controlar el enlace se llama la estación primaria. La otra es una estación secundaria. Los protocolos orientados a bits consideran la presencia de una estación primaria y una o más estaciones secundarias. Todas las comunicaciones en el enlace de datos son de la estación primaria a una o más de las estaciones secundarias o de la estación secundaria a la primaria.

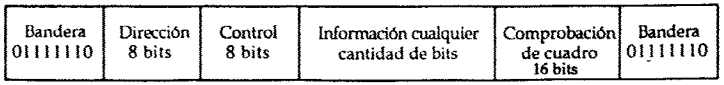
bandera de ocho bits

El formato del cuadro para el protocolo orientado a bits se muestra en la figura 11-26. Un cuadro comienza con una bandera de 8 bits 01111110 a la que le sigue una secuencia de dirección y control. El campo de información no se limita en formato o contenido y puede tener cualquier longitud. El campo de comprobación del cuadro es una secuencia CRC (*cyclic redundancy check*, comprobación de redundancia cíclica) que se usa para detectar errores en la transmisión. La bandera de terminación le indica a la estación receptora que los 16 bits que recibió constituyen los bits CRC. Al final del cuadro puede seguir otro cuadro, otra bandera o una secuencia de números 1 consecutivos. Cuando dos cuadros están uno tras el otro, la bandera que interviene es simultáneamente la bandera final del primer cuadro y la bandera inicial del siguiente. Si no se intercambia información, el transmisor envía una serie de banderas para conservar la línea en estado activo. Se dice que la línea está en estado inactivo cuando ocurren 15 o más números 1 consecutivos. Se envían algunos cuadros con ciertos mensajes de control sin un campo de información. Un cuadro debe tener un mínimo de 32 bits entre dos banderas para alojar la dirección, los campos de comprobación del cuadro, la dirección y el control. La longitud máxima depende de la condición del canal de comunicación y su capacidad para transmitir mensajes largos sin error.

inserción de cero

Para evitar que ocurra una bandera a la mitad de un cuadro, el protocolo orientado a bits utiliza un método llamado *inserción de 0*. Este

Figura 11-26 Formato de marco para protocolo orientado a bits.



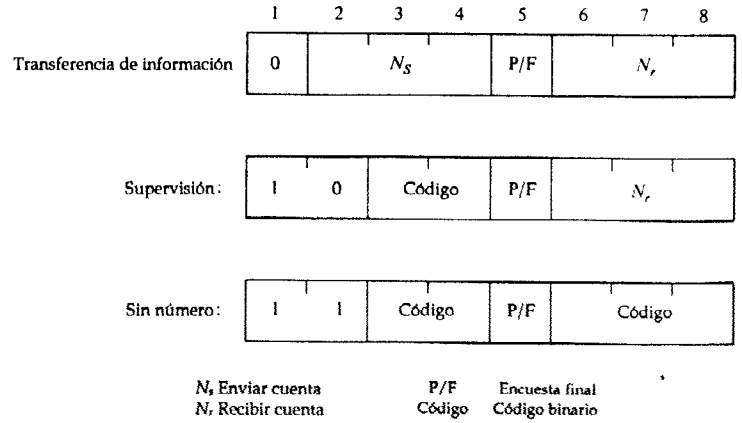
requiere que la estación que transmite inserte un 0 después de que se sucedan cinco números 1 continuos. El receptor siempre quita un 0 después de una sucesión de cinco números 1. Por lo tanto el patrón de bits 0111111 se transmite como 01111101 y el receptor lo restablece a su valor original al quitar el 0 que le sigue al quinto número 1. Como consecuencia, jamás se transmite un patrón 01111110 entre las banderas inicial y final.

Después de la bandera está el campo de dirección que utiliza la estación primaria para designar la dirección de la estación secundaria. Cuando una estación secundaria transmite un cuadro, la dirección le dice a la estación primaria cuál estación secundaria originó el cuadro. Un campo de dirección de 8 bits puede especificar hasta 256 direcciones. Algunos protocolos orientados a bits permiten que se use un campo de dirección ampliado. Para hacer esto, se desactiva en 0 el bit menos significativo de un byte de dirección si le sigue otro byte de dirección. Se usa un número 1 en el bit menos significativo de un byte para reconocer el último byte de dirección.

campo de control

Después del campo de dirección está el campo de control. El campo de control viene en tres formatos diferentes, como se muestra en la figura 11-27. El formato de transferencia de información se utiliza para la transmisión de datos ordinarios. Cada cuadro transmitido en este formato contiene cuentas de envío y recibo. Una estación que transmite cuadros en secuencia cuenta y numera cada cuadro. Esta cuenta la proporciona la cuenta de envío N_s. Una estación que recibe cuadros secuenciados cuenta cada marco sin error que recibe. Esta cuenta la proporciona la cuenta de recibo N_r. La cuenta N_r avanza cuando se comprueba un cuadro y se encuentra que no contiene

FIgura 11-27 Formato de campo de control en protocolo orientado a bits.



errores. El receptor confirma que se han aceptado los marcos de información numerada al retornar su cuenta N_r a la estación que transmite.

La estación primaria utiliza el bit P/F para encuestar a una estación secundaria que solicite que inicie la transmisión. La estación secundaria la utiliza para indicar el cuadro transmitido final. Por lo tanto, el campo P/F se llama P (poll, encuesta) cuando transmite la estación primaria, pero se denomina F (final) cuando transmite una estación secundaria. Cada cuadro que envía la estación primaria a la estación secundaria tiene un bit de P inactivo en 0. Cuando la estación primaria termina y está preparada para que responda a la estación secundaria, el bit P se activa en 1, después la estación secundaria responde con una cantidad de cuadros en la cual el bit F se desactiva en 0. Cuando la estación secundaria envía el último cuadro, activa el bit F en 1. Por lo tanto, se utiliza el bit P/F para determinar cuándo ha terminado la transmisión de datos de una estación.

El formato supervisor del campo de control se reconoce porque los dos primeros bits son 1 y 0. Los dos bits siguientes indican el tipo de comando. A esto le sigue un bit P/F y una recepción de cuenta de cuadro de secuencia. Los cuadros del formato supervisor no contienen un campo de información. Se usan para ayudar en la transferencia de información porque confirman la aceptación de los cuadros anteriores que llevan información, comunican condiciones preparado u ocupado y reportan errores en la numeración de cuadros.

Se reconoce el formato sin numerar porque los dos primeros bits son 11. Los cinco bits de código disponibles en este formato puede especificar hasta 32 comandos y respuestas. La estación primaria utiliza el campo de control para especificar un comando para una estación secundaria. La estación secundaria utiliza el campo de control para transmitir una respuesta a la estación primaria. Los cuadros de formato sin numerar se emplean para inicializar funciones de enlace, reportar errores de procedimiento, colocar las estaciones en modo desconectado y otras operaciones de control de enlace de datos.

PROBLEMAS

11-1. Las direcciones asignadas a cuatro registros de la interface de E/S de la figura 11-2 son iguales al equivalente binario de 12, 13, 14 y 15. Muestre el circuito externo que debe conectarse entre la dirección de E/S de 8 bits de la CPU y las entradas CS, *RS1* y *RS0* de la interface.

11-2. Seis unidades de interface del tipo que se muestra en la figura 11-2 están conectadas a una CPU que utiliza una dirección de E/S de 8 bits. Cada una de las seis entradas de selección de integrado (CS) está conectada a una línea de dirección diferente. Por lo tanto, la línea de dirección de orden superior está conectada a la entrada CS de la primera unidad de interface y la sexta línea de dirección está conectada a la entrada CS de la sexta unidad de interface.

Las dos líneas de dirección de orden menor están conectadas a las *RS1* y *RS0* de las seis unidades de interface. Determine la dirección de 8 bits de cada registro en cada interface.

11-3. Liste cuatro dispositivos periféricos que produzcan una salida aceptable para que la comprenda una persona.

11-4. Escriba su nombre completo en ASCII utilizando 8 bits por carácter y con el bit de la extrema izquierda siempre 0. Incluya un espacio entre los nombres y un punto después de una inicial del segundo nombre.

11-5. ¿Cuál es la diferencia entre E/S independiente y E/S mapeada en memoria? ¿Cuáles son las ventajas y desventajas de cada una?

11-6. Indique si los siguientes constituyen comandos de control, de estado o de transferencia de datos.

a. Omitir la siguiente instrucción si está activa la bandera.
b. Buscar cierto registro en una cinta magnética.
c. Comprobar si está preparado el dispositivo de E/S.
d. Mover papel impreso para iniciar en siguiente página.
e. Leer interface de registro de estado.

11-7. Una unidad de interface comercial utiliza diferentes nombres para las líneas de reconocimiento asociadas con la transferencia de datos del dispositivo de E/S dentro de la unidad de interface. La línea de reconocimiento de entrada de interface se etiqueta STB (habilitación, estroboscopio) y la línea de reconocimiento de salida de interface se denomina IBF (buffer de entrada lleno). Una señal de nivel bajo en STB carga datos del canal de E/S dentro del registro de datos de la interface. Una señal de nivel alto de IBF indica que la interface ha aceptado los datos. IBF va a nivel bajo después de una señal de lectura de E/S por parte de la CPU cuando lee el registro de datos.

a. Dibuje un diagrama de bloques que muestre la CPU, la interface y el dispositivo de E/S junto con las interconexiones pertinentes entre las tres unidades.
b. Dibuje un diagrama de temporización para la transferencia de reconocimiento mutuo.
c. Obtenga un diagrama de flujo de secuencia de eventos para la transferencia del dispositivo a la interface y de la interface a la CPU.

11-8. Una CPU con un reloj de 20 Mhz está conectada a una unidad de memoria cuyo tiempo de acceso es de 40 ns. Formule diagramas de temporización para lectura y escritura utilizando un pulso de habilitación READ y un pulso de habilitación WRITE. Incluya la dirección en el diagrama de temporización.

11-9. La interface de comunicación asíncrona que se muestra en la figura 11-8 está conectada entre una CPU y una impresora. Dibuje un diagrama de flujo que describa la secuencia de operaciones en la parte transmisora de la interface, cuando la CPU envía caracteres que se van a imprimir.

11-10. Proporcione al menos seis condiciones de estado para la especificación de bits individuales en el registro de estado de una interface de comunicación asíncrona.

11-11. ¿Cuántos bits hay en el registro de corrimiento transmisor de la figura 11-8 cuando la interface está conectada a una terminal que necesita un bit de alto? Liste los bits en el registro de paridad cuando se trasmite la letra W utilizando ASCII con paridad par.

11-12. ¿Cuántos caracteres por segundo pueden transmitirse sobre una línea de 1200 baudios en cada uno de los siguientes modos? (Considere un código de caracteres de ocho bits).

a. Transmisión serial síncrona.
b. Transmisión serial asíncrona con dos bits de alto.
c. Transmisión serial asíncrona con un bit de alto.

11-13. Se inserta información dentro de un búffer FIFO a una velocidad de m bytes por segundo. La información se borra a una velocidad de n bytes por segundo. La máxima capacidad de búffer es k bytes.

a. ¿Cuánto tiempo se necesita para llenar un buffer vacío cuando $m > n$?
b. ¿Cuánto tiempo se necesita para que se vacíe un búffer lleno cuando $m < n$?
c. ¿Se necesita el búffer FIFO si $m = n$?

11-14. Los bits en el registro de control de la FIFO que se muestra en la figura 11-9 son $F_1F_2F_3F_4 = 0011$. Proporcione la secuencia de operaciones internas cuando se borra un dato de la FIFO y se inserta después un nuevo dato.

11-15. ¿Cuáles son los valores de entrada preparada, salida preparada y los bits de control del F_1 al F_4 en la figura 11-9 cuando:

a. ¿El búffer está vacío?
b. ¿El búffer está lleno?
c. ¿El búffer contiene dos datos?

11-16. Muestre un diagrama de bloques similar a la figura 11-10 para la transferencia de datos de la CPU a la interface y después a un dispositivo de E/S. Determine un procedimiento para especificar y limpiar el bit de bandera.

11-17. Usando la configuración establecida en el problema 11-16, obtenga un diagrama de flujo (similar a la de la figura 11-11) para el programa de CPU para sacar datos.

11-18. ¿Cuál es la ventaja básica de utilizar transferencia de datos iniciada por interrupción sobre la transferencia bajo el control del programa sin interrupciones?

11-19. En la mayoría de las computadoras se reconoce una interrupción sólo después de la ejecución de la instrucción. Considere la posibilidad de reconocer la interrupción en cualquier momento durante la ejecución de la instrucción. Analice la dificultad que puede surgir.

11-20. ¿Qué sucede en la interrupción de prioridad de lazo de margaritas que se muestra en la figura 11-12 cuando el dispositivo 1 solicita una interrupción, después que el dispositivo 2 ha enviado una solicitud de interrupción a la CPU, pero antes de que la CPU responda con el reconocimiento de interrupción?

11-21. Considere una computadora sin circuitería de interrupción de prioridad. Cualquiera de muchas fuentes puede interrumpir a la computadora y cualquier solicitud de interrupción da como resultado el almacenamiento de la dirección de retorno y la transferencia a una rutina de interrupción común. Explique cómo puede establecerse una prioridad en el programa de servicio de interrupción.

11-22. Utilizando técnicas de diseño de circuitos combinacionales, derive las expresiones booleanas listadas en la tabla 11-2 para el codificador de prioridad. Dibuje el diagrama lógico del circuito.

11-23. Diseñe una circuitería de interrupción de prioridad paralela para un sistema con ocho fuentes de interrupción.

11-24. Obtenga la tabla de verdad de un codificador de prioridad 8×3. Considere que las tres *xyz* del codificador de prioridad se utilizan para proporcionar una dirección de vector de la forma 101*xyz*00. Liste las ocho direcciones de vector, comenzando con la de más alta prioridad.

11-25. ¿Qué debe hacerse en la figura para hacer que los cuatro valores VAD sean iguales al equivalente binario de 76, 77, 78 y 79

11-26. ¿Cuáles pasos de programación se requieren para comprobar cuando una fuente interrumpe a la computadora mientras todavía está siendo atendida mediante una solicitud de interrupción previa de la misma fuente?

11-27. ¿Por qué son bidireccionales las líneas de control de lectura y escritura en un controlador DMA? ¿Bajo qué condiciones y para qué propósito se utilizan como entradas? ¿Bajo qué condiciones y para qué propósito se usan como salidas?

11-28. Es necesario transferir 256 palabras de un disco magnético a una sección de memoria que comienza en la dirección 1230. La transferencia es mediante DMA, como se muestra en la figura 11-18.

a. Proporcione los valores iniciales que debe transferir la CPU al controlador DMA.

b. Proporcione la cuenta paso a paso de las acciones que se toman durante la entrada de las primeras dos palabras.

11-29. Un controlador DMA transfiere palabras de 16 bits a la memoria utilizando el robo de ciclo. Las palabras se ensamblan de un dispositivo que trasmite caracteres a una velocidad de 2400 caracteres por segundo. La CPU recupera y ejecuta instrucciones a una velocidad de un millón de instrucciones por segundo. ¿Cuánto se frenará la CPU debido a una transferencia DMA?

11-30. ¿Porqué el DMA tiene prioridad sobre la CPU cuando ambos solicitan una transferencia de memoria?

11-31. Dibuje un diagrama de flujo similar a la de la figura 11-20, que describa la comunicación de canal CPU-E/S en la IBM 370.

11-32. La dirección de una terminal conectada a un procesador de comunicación de datos consta de dos letras del alfabeto o una letra y uno de diez números. ¿Cuántas direcciones diferentes pueden formularse?

11-33. Liste un procedimiento de línea y la secuencia de caracteres posibles para la comunicación entre un procesador de comunicación de datos y una terminal remota. El procesador pregunta si la terminal es operativa. La terminal responde sí o no. Si la respuesta es sí, el procesador envía un bloque de texto.

11-34. Un enlace de comunicación de datos emplea el protocolo controlado por caracteres con transparencia de datos, utilizando el carácter DLE. El mensaje de texto que envía el trasmisor entre STX y ETX es como sigue:

```
DLE STX DLE DLE ETX DLE DLE ETX DLE ETX
```

¿Cuál es el valor binario de los datos del texto transparente?

11-35. ¿Cuál es la cantidad mínima de bits que debe tener un cuadro en el protocolo orientado a bits?

11-36. Muestre cómo funciona la inserción de ceros, en el protocolo orientado a bits, cuando se trasmite un cero seguido por los diez bits que representan el equivalente binario de 1023.

REFERENCIAS

1. Gorsline, G. W., *Computer Organization: Hardware/Software*, 2a. ed. Englewood Cliffs, NJ: Prentice Hall, 1986.
2. Hays, J. F., *Computer Architecture and Organization*, 2a. ed. Nueva York: McGraw-Hill, 1988.
3. Hill, F. J., y G. R. Peterson, *Digital Systems: Hardware Organization and Design*, 3a. ed. Nueva York: John Wiley, 1987.
4. Hwang, K., y F. A. Briggs, *Computer Architecture and Parallel Processing*. Nueva York: McGraw-Hill, 1984.
5. Lippiatt, A. G., y G. L. Wright, *The Architecture of Small Computer Systems*, 2a. ed. Englewood Cliffs, NJ: Prentice Hall, 1985.
6. Patterson, D. A. y J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, San Mateo, CA: Morgan Kaufmann Publishers, 1970.
7. Pollard, L. H., *Computer Design and Architecture*, Englewood Cliffs, NJ: Prentice Hall,1990.
8. Rafiquzzaman, M., y R. Chandra, *Modern Computer Architecture*, St. Paul, MN: West Publishing, 1988.
9. Toy, W., y B. Zee, *Computer Hardware/Software Architecture*. Englewood Cliffs, NJ: Prentice Hall, 1986.
10. Wakerly, J. F., *Microcomputer Architecture and Programming*. Nueva York: John Wiley, 1981.
11. Ward, S. A., y R. H. Halstead, Jr., *Computation Structures*. Cambridge, MA: MIT Press, 1990.