

ENTRADA / SALIDA

Junto con el procesador y el conjunto de módulos de memoria, el tercer elemento clave de un computador es un conjunto de módulos de E/S.

Cada módulo se conecta al bus del sistema o a un conmutador central y controla uno o más dispositivos periféricos.

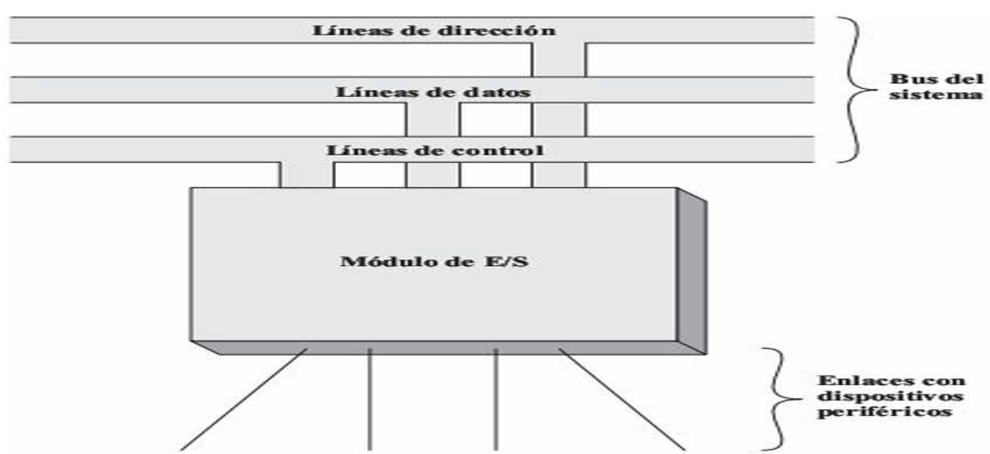
Un módulo de E/S no es únicamente un conector mecánico que permite enchufar el dispositivo al bus del sistema; sino que además está dotado de cierta inteligencia, es decir, contiene la lógica necesaria para permitir la comunicación entre el periférico y el bus.

Los periféricos no se conectan directamente al bus del sistema y las razones son:

- Hay una amplia variedad de periféricos con formas de funcionamiento diferentes. Podría ser imposible incorporar la lógica necesaria dentro del procesador para controlar la diversidad de dispositivos.
- A veces la velocidad de transferencia de datos de los periféricos es mucho menor que la de la memoria o el procesador. Así, no es práctico utilizar un bus del sistema de alta velocidad para comunicarse directamente con un periférico.
- También hay casos donde, la velocidad de transferencia de algunos periféricos es mayor que la de la memoria o el procesador. Esta diferencia daría lugar a comportamientos poco eficientes si no se gestionase correctamente.
- Con frecuencia, los periféricos utilizan datos con formatos y tamaños de palabra diferentes de los del computador a los que se conectan.

En consecuencia, se necesita un módulo de E/S, el cual tiene dos funciones principales :

- Realizar la interfaz entre el procesador y la memoria a través del bus del sistema o un conmutador central.
- Realizar la interfaz entre uno o más dispositivos periféricos mediante enlaces de datos específicos.



DISPOSITIVOS EXTERNOS.

Las operaciones de E/S se realizan a través de una amplia gama de dispositivos que proporcionan una forma de intercambiar datos entre el exterior y el computador. Un dispositivo externo se conecta al computador mediante un enlace a un módulo de E/S. El enlace se utiliza para intercambiar señales de control, estado, y datos entre el módulo de E/S y el dispositivo externo.

Un dispositivo externo conectado a un módulo de E/S frecuentemente se denomina *dispositivo periférico* o simplemente *periférico*.

En sentido amplio, los dispositivos externos se pueden clasificar en tres categorías:

- *De interacción con humanos*: permiten la comunicación con el usuario del computador.
- *De interacción con máquinas*: permiten la comunicación con elementos del equipo.
- *De comunicación*: permiten la comunicación con dispositivos remotos.

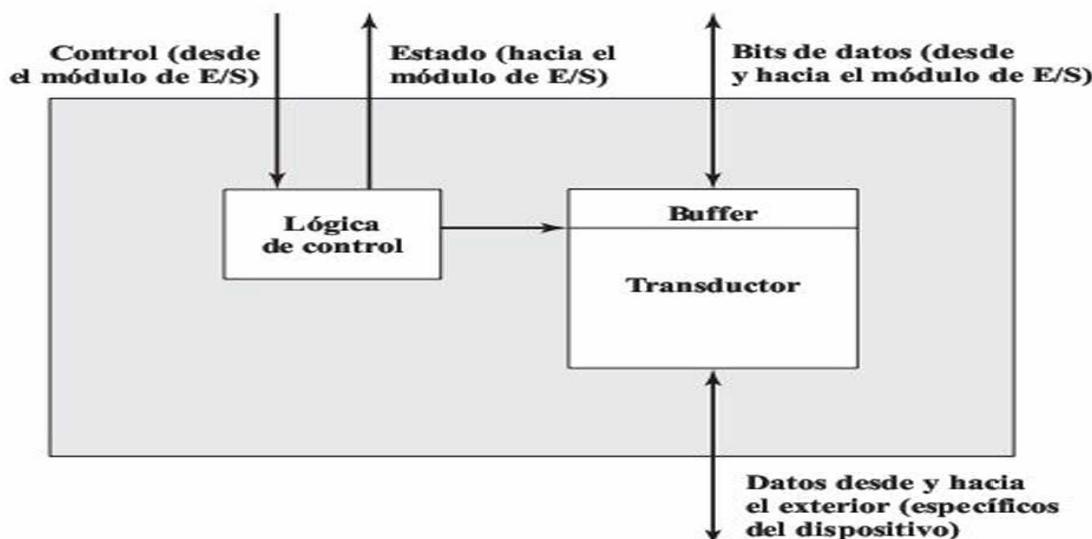
Algunos ejemplos de dispositivos de interacción con humanos son los terminales de video (VDT, *Video Display Termináis*) y las impresoras.

Ejemplos de dispositivos de interacción con máquinas son los discos magnéticos y los sistemas de cinta, y los sensores y actuadores, tales como los que se usan en aplicaciones de robótica.

Los discos y los sistemas de cinta también se consideran como dispositivos de E/S. Desde el punto de vista de su función, estos dispositivos son parte de la jerarquía de memoria y desde un punto de vista estructural, estos dispositivos se controlan mediante módulos de E/S .

Los dispositivos de comunicación permiten que el computador intercambie datos con un dispositivo remoto, que puede ser un dispositivo de interacción con humanos, como una terminal, un dispositivo de interacción con máquinas o incluso otro computador.

La conexión con el módulo de E/S se realiza a través de señales de control, estado y datos.



Los *datos* son el conjunto de bits a ser enviados o recibidos del módulo de E/S.

Las *señales de control* determinan la función que debe realizar el dispositivo, tal como enviar datos al módulo de E/S, ENTRADA (INPUT) o LECTURA (READ), aceptar datos desde el módulo de E/S, SALIDA (OUTPUT) o ESCRITURA (WRITE), indicar el estado o realizar alguna función de control particular del dispositivo (por ejemplo, situar una cabeza del disco).

Las *señales de estado* indican el estado del dispositivo (LISTO/NO-LISTO , READY/NOT-READY) que indica si el dispositivo está preparado para la transferencia de datos.

La *lógica de control* asociada al dispositivo controla su operación en respuesta a las indicaciones del módulo de E/S.

El *transductor* convierte las señales eléctricas asociadas al dato a otra forma de energía en el caso de una salida y viceversa en el caso de una entrada.

Usualmente, existe un *buffer* asociado al transductor para almacenar temporalmente el dato que se está transfiriendo entre el módulo de E/S y el exterior;

TECLADO / MONITOR

Una forma común de interacción computador/usuario se produce a través de la combinación teclado/ monitor. El usuario proporciona la entrada a través del teclado. A continuación esta entrada se transmite al computador y puede verse en el monitor. También el monitor muestra los datos que proporciona el computador.

La unidad básica de intercambio es el carácter.

Asociado con cada carácter hay un código, usualmente de siete u ocho bits de longitud. El código genérico comúnmente utilizado es el IRA (*International Reference Alphabet*)

Cada carácter de este código se representa mediante un único número binario de 7 bits; en consecuencia, se pueden representar 128 caracteres.

Y en particular el código ASCII es uno de los mas utilizados

Los bits de cada carácter se designan desde b_7 , que es el bit más significativo, a b_1 el menos significativo .

Los caracteres son de dos tipos: imprimibles y de control .

Los imprimibles son alfabéticos, numéricos y especiales, que pueden imprimirse en papel o visualizarse en una pantalla.

Algunos de los caracteres de control se utilizan para controlar la impresora o la visualización de los caracteres; un ejemplo es el retorno de carro. Otros caracteres de control están relacionados con los procedimientos de comunicación.

Para la entrada desde teclado, cuando el usuario pulsa una tecla, se genera una señal electrónica interpretada por el transductor del teclado que la traduce al patrón binario del correspondiente código IRA.

Este patrón binario se transmite al módulo de E/S del computador.

En el computador, el texto se puede almacenar utilizando el mismo código IRA.

En la salida, los códigos IRA se transmiten al dispositivo externo desde el módulo de E/S. El transductor del dispositivo interpreta este código y envía las señales electrónicas precisas para mostrar en pantalla el carácter indicado o realice la función de control solicitada.

bit posición				0	0	0	0	1	1	1	1	1	1	
				0	0	1	1	0	0	1	1	1	1	
b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	NUL	DLE	SP	0	@	P	‘	p
0	0	0	0	0	0	0	SOH	DC1	!	1	A	Q	a	q
0	0	0	0	0	0	1	STX	DC2	“	2	B	R	b	r
0	0	1	0	0	1	0	ETX	DC3	#	3	C	S	c	s
0	0	1	1	0	1	1	EOT	DC4	\$	4	D	T	d	t
0	1	0	0	0	0	0	ENQ	NAK	%	5	E	U	e	u
0	1	0	1	0	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	0	0	0	1	BEL	ETB	‘	7	G	W	g	w
1	0	0	0	0	0	0	BS	CAN	(8	H	X	h	x
1	0	0	1	0	0	1	HT	EM)	9	I	Y	i	y
1	0	1	0	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	0	1	1	VT	ESC	+	;	K	L	k	{
1	1	0	0	0	0	0	FF	FS	,	<	L	\	l	
1	1	0	1	0	1	0	CR	GS	-	=	M	J	m	}
1	1	1	0	0	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	0	1	1	SI	US	/	?	O	-	o	DEL

Control de Formato

BS (Backspace, Espacio atrás): indica movimiento de un espacio hacia atrás del mecanismo de impresión o del cursor de la pantalla.

HT (Horizontal Tab, Tabulación Horizontal): indica movimiento del mecanismo de impresión o del cursor de pantalla hasta el siguiente «tabulador» asignado o a la posición de parada.

LF (Line Feed, Avance de Línea): indica movimiento del mecanismo de impresión o del cursor de pantalla hasta el comienzo de la línea siguiente.

VT (Vertical Tab, Tabulación Vertical): indica movimiento del mecanismo de impresión o del cursor de pantalla hasta la siguiente de una serie de líneas impresas.

FF (Form Feed, Avance de Página): indica movimiento del mecanismo de impresión o del cursor de pantalla hasta el comienzo de la siguiente página o imagen de pantalla.

CR (Carriage Return, Retorno de Carro): indica movimiento del mecanismo de impresión o del cursor de pantalla hasta el comienzo de la línea en curso.

Control de Transmisión	
SOH (<i>Start of Heading</i> , Comienzo de Cabecera): usado para indicar el comienzo de una cabecera que puede contener información de dirección o enrutamiento.	NAK (<i>Negative Acknowledgement</i> , Reconocimiento Negativo): carácter transmitido por un dispositivo receptor como respuesta negativa al emisor. Se utiliza como respuesta negativa a los mensajes de sondeo.
STX (<i>Start of Text</i> , Comienzo de Texto): usado para indicar el comienzo de texto y también para indicar el final de la cabecera.	SYN (<i>Synchronous/Idle</i> , Síncrono/Parado): utilizado por un sistema con transmisión síncrona para conseguir la sincronización. Cuando no se envía ningún dato un sistema con transmisión síncrona envía caracteres SYN continuamente.
ETX (<i>End of Text</i> , Final de Texto): usado como fin del texto que se inició con STX.	ETB (<i>End of Transmission Block</i> , Final del Bloque Transmitido): se utiliza en el contexto de las comunicaciones para indicar el final de un bloque de datos. Permite organizar los datos en bloques donde la estructura del bloque no está necesariamente relacionada con el formato de procesamiento.
EOT (<i>End of Transmission</i> , Final de Transmisión): indica el final de la transmisión que puede incluir uno o más textos con sus cabeceras.	
ENQ (<i>Enquiry</i> , Interrogación): petición de respuesta desde una estación remota. Se puede utilizar como una petición de identificación («WHO ARE YOU?») para la estación.	
ACK (<i>Acknowledge</i> , Reconocimiento): carácter transmitido por un dispositivo receptor como respuesta afirmativa al emisor. Se utiliza como respuesta positiva a los mensajes de sondeo (<i>polling</i>).	
Separadores de Información	
FS (<i>File Separator</i> , Separador de Fichero)	Los separadores de información se utilizan de manera opcional si bien están ordenados jerárquicamente desde el FS (el más inclusivo) hasta US (el menos).
GS (<i>Group Separator</i> , Separador de Grupo)	
RS (<i>Record Separator</i> , Separador de Registro)	
US (<i>United Separator</i> , Separador Unido)	
Miscelánea	
NUL (<i>Nul</i> , Nulo): carácter nulo. Se utiliza para consumir tiempo u ocupar espacio en una cinta cuando no hay datos.	DEL (<i>Delete</i> , Borrar): utilizado para borrar caracteres (por ejemplo en una cinta de papel perforando cada posición de bit).
BEL (<i>Bell</i> , Pitido): utilizado para llamar la atención humana. Puede controlar una alarma o dispositivos que requieren llamar la atención.	SP (<i>Space</i> , Espacio): carácter no imprimible utilizado para separar palabras o para mover el mecanismo de impresión, o también para adelantar el cursor una posición.
SO (<i>Shift out</i> , Fuera de Código): indica que los caracteres que siguen no deben interpretarse utilizando el estándar hasta que llegue un carácter SI.	DLE (<i>Data Link Escape</i> , Salir del Enlace de Datos): carácter que puede cambiar el significado de uno o más caracteres consecutivos que lo siguen. Puede proporcionar caracteres de control suplementarios, o permitir el envío de caracteres de datos con cualquier combinación de bits.
SI (<i>Shift in</i> , Dentro de Código): indica que los caracteres que siguen deben interpretarse de acuerdo con el estándar.	

CONTROLADOR DE DISCO.

Un controlador de disco contiene la electrónica necesaria para intercambiar señales de datos, control y estado con un módulo de E/S, más la electrónica necesaria para controlar el mecanismo de lectura/escritura del disco.

MÓDULOS DE ENTRADA / SALIDA.

Las principales funciones y requisitos de un módulo de E/S se encuentran dentro de las siguientes categorías:

- Control y temporización.
- Comunicación con el procesador.
- Comunicación con los dispositivos.
- Almacenamiento temporal de datos.
- Detección de errores.

En cualquier momento, el procesador puede comunicarse con uno o más dispositivos externos en cualquier orden, según las necesidades de E/S del programa.

Los recursos internos, tales como la memoria principal y el bus del sistema, deben ser compartidos entre distintas actividades incluyendo la E/S de datos.

Así, la función de E/S incluye ciertos requisitos de control y temporización, para coordinar el tráfico entre los recursos internos y los dispositivos externos.

Por ejemplo, el control de la transferencia de datos desde un dispositivo externo al procesador podría implicar la siguiente secuencia de pasos:

1. El procesador interroga al módulo de E/S para comprobar el estado del dispositivo conectado al mismo.
2. El módulo de E/S devuelve el estado del dispositivo.
3. Si el dispositivo está operativo y preparado para transmitir, el procesador solicita la transferencia del dato mediante una orden al módulo de E/S.
4. El módulo de E/S obtiene un dato del dispositivo externo.
5. Los datos se transfieren desde el módulo de E/S al procesador.
Si el sistema utiliza un bus, entonces cada una de las interacciones entre el procesador y el módulo de E/S implican uno o más arbitrajes del bus.

Además, el módulo de E/S debe tener la capacidad de establecer comunicación con el procesador y con el dispositivo externo. La comunicación con el procesador implica:

- *Decodificación de órdenes*: el módulo de E/S acepta órdenes del procesador. Estas órdenes generalmente se envían utilizando líneas del bus de control. Por ejemplo, un módulo de E/S para un controlador de disco podría recibir las órdenes: LEER SECTOR (READ SECTOR), ESCRIBIR SECTOR (WRITE SECTOR), BUSCAR número de pista (SEEK *track number*), y EXPLORAR Identificador de registro (SCAN *record ID*).

Cada una de las dos últimas órdenes incluye un parámetro que es enviado a través del bus de datos.

- *Datos*: el procesador y el módulo de E/S intercambian datos a través del bus de datos.
- *Información de Estado*: puesto que los periféricos son lentos, es importante conocer el estado del módulo de E/S. Por ejemplo, si se solicita a un módulo de E/S que envíe datos al procesador (lectura), puede que no esté preparado por encontrarse todavía respondiendo a una orden de E/S previa. Esta situación puede indicarse con una señal de estado. Las señales de estado usuales son BUSY (ocupado) y READY (preparado). También puede haber señales para informar de ciertas situaciones de error.
- *Reconocimiento de Dirección*: igual que cada palabra de memoria tiene una dirección, cada dispositivo de E/S tiene otra. Así, un módulo de E/S puede reconocer una única dirección para cada uno de los periféricos que controla.

Por otra parte, el módulo de E/S debe ser capaz de *comunicarse con el dispositivo*. Esta comunicación implica intercambiar órdenes, información del estado, y datos.

Una tarea esencial para un módulo de E/S es el *almacenamiento temporal de datos (data buffering)*.

Mientras que la velocidad de transferencia desde, y hacia, la memoria principal o el procesador es bastante alta, dicha velocidad puede ser varios órdenes de magnitud menor para la mayoría de los dispositivos periféricos.

Los datos provenientes de la memoria se envían al módulo de E/S en ráfagas rápidas, son almacenados temporalmente en el módulo de E/S y después se envían al periférico a la velocidad de este.

En el sentido contrario, los datos se almacenan para no mantener a la memoria ocupada en una operación de transferencia lenta.

El módulo de E/S debe ser capaz de operar a las velocidades tanto del dispositivo como de la memoria.

Igualmente, si el dispositivo de E/S trabaja a una velocidad mayor que la memoria, el módulo de E/S se encarga del almacenamiento temporal necesario.

Un módulo de E/S a menudo es responsable de la *detección de errores* y de informar de estos errores al procesador. Una clase de errores son los defectos mecánicos y eléctricos en el funcionamiento del dispositivo (por ejemplo papel atascado, pista de disco en mal estado, etc.).

Otra clase está constituida por los cambios accidentales en los bits al transmitirse desde el dispositivo al módulo de E/S. Para detectar estos errores de transmisión frecuentemente se utiliza algún tipo de código de detección de errores.

Un ejemplo sencillo es el uso de un bit de paridad en cada carácter de datos.

Por ejemplo, un carácter IRA utiliza siete de los bits de un byte.

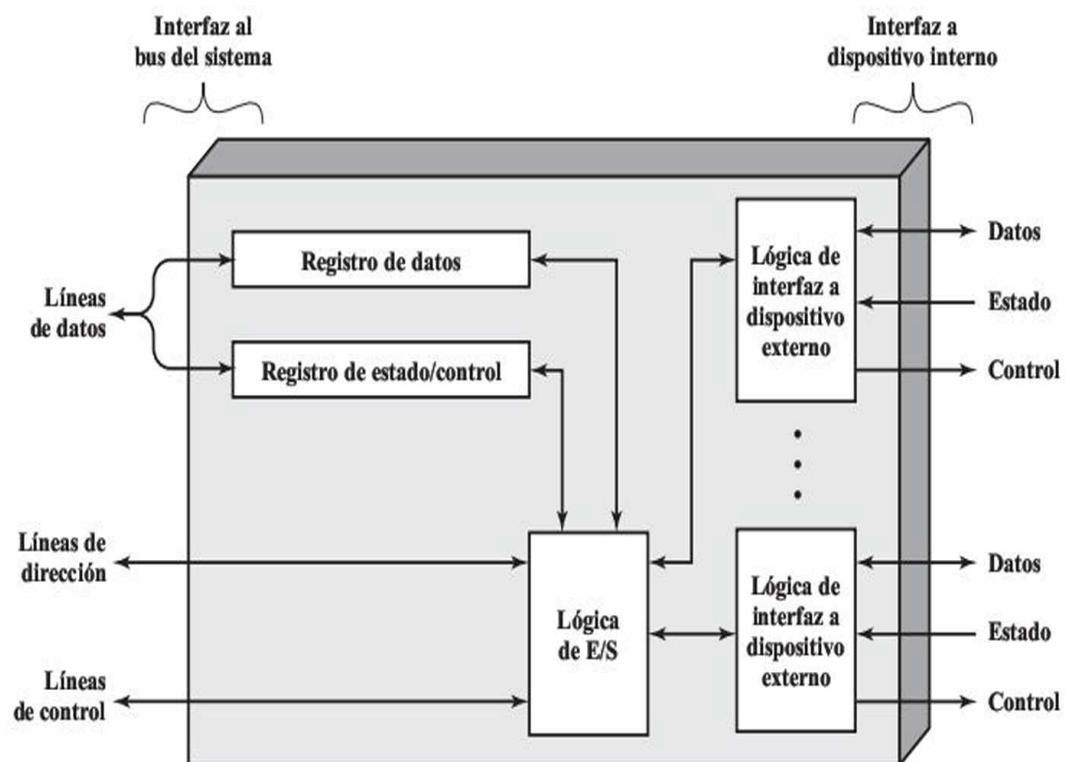
El octavo bit se asigna de manera que el número total de «unos» en el byte sea par (paridad par) o impar (paridad impar).

Cuando se recibe un byte, el módulo de E/S comprueba la paridad para determinar si se ha producido un error.

ESTRUCTURA DE UN MÓDULO DE E / S .

La complejidad de los módulos de E/S y el número de dispositivos externos que controlan varían considerablemente.

Un esquema de bloques genérico podría ser el siguiente :



El módulo se conecta al resto del computador a través de un conjunto de líneas (por ejemplo, líneas del bus del sistema).

Los datos que se transfieren a, y desde, el módulo se almacenan temporalmente en uno o más registros de datos.

Además, puede haber uno o más registros de estado que proporcionan información del estado presente. Un registro de estado también puede funcionar como un registro de control, para recibir información de control del procesador.

La lógica que hay en el módulo interactúa con el procesador a través de una serie de líneas de control. Estas son las que utiliza el procesador para proporcionar las órdenes al módulo de E/S.

Algunas de las líneas de control pueden ser utilizadas por el módulo de E/S (por ejemplo, para las señales de arbitraje y estado).

El módulo también debe ser capaz de reconocer y generar las direcciones asociadas a los dispositivos que controla.

Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un conjunto único de direcciones.

Por último, el módulo de E/S posee la lógica específica para la interfaz con cada uno de los dispositivos que controla.

TÉCNICAS DE ENTRADA / SALIDA .

	Sin interrupciones	Usando interrupciones
Transferencia de E/S a memoria a través de la CPU	E/S Programada	E/S mediante interrupciones
Transferencia directa de E/S a memoria		Acceso Directo a Memoria (DMA)

Son posibles tres técnicas para las operaciones de E/S.

Con la *E/S programada*, los datos se intercambian entre el procesador y el módulo de E/S.

El procesador ejecuta un programa que controla directamente la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de una orden de lectura o escritura y la transferencia del dato.

Cuando el procesador envía una orden al módulo de E/S, debe esperar hasta que la operación de E/S concluya.

Si el procesador es más rápido que el módulo de E/S el procesador desperdicia este tiempo.

Con la *E/S mediante interrupciones*, el procesador proporciona la orden de E/S, continúa ejecutando otras instrucciones y es interrumpido por el módulo de E/S cuando este ha terminado su trabajo.

Tanto con E/S programada como con interrupciones, el procesador es responsable de extraer los datos de la memoria principal en una salida y de almacenar los datos en la memoria principal en una entrada.

La alternativa se conoce como *acceso directo a memoria (DMA)*.

En este caso, el módulo de E/S y la memoria principal intercambian datos directamente, sin la intervención del procesador.

ENTRADA SALIDA PROGRAMADA.

Cuando el procesador está ejecutando un programa y encuentra una instrucción relacionada con una E/S, ejecuta dicha instrucción mandando una orden al módulo de E/S apropiado.

Con E/S programada, el módulo de E/S realizará la acción solicitada y después activará los bits apropiados en el registro de estado de E/S .

El módulo de E/S no realiza ninguna otra acción para avisar al procesador.

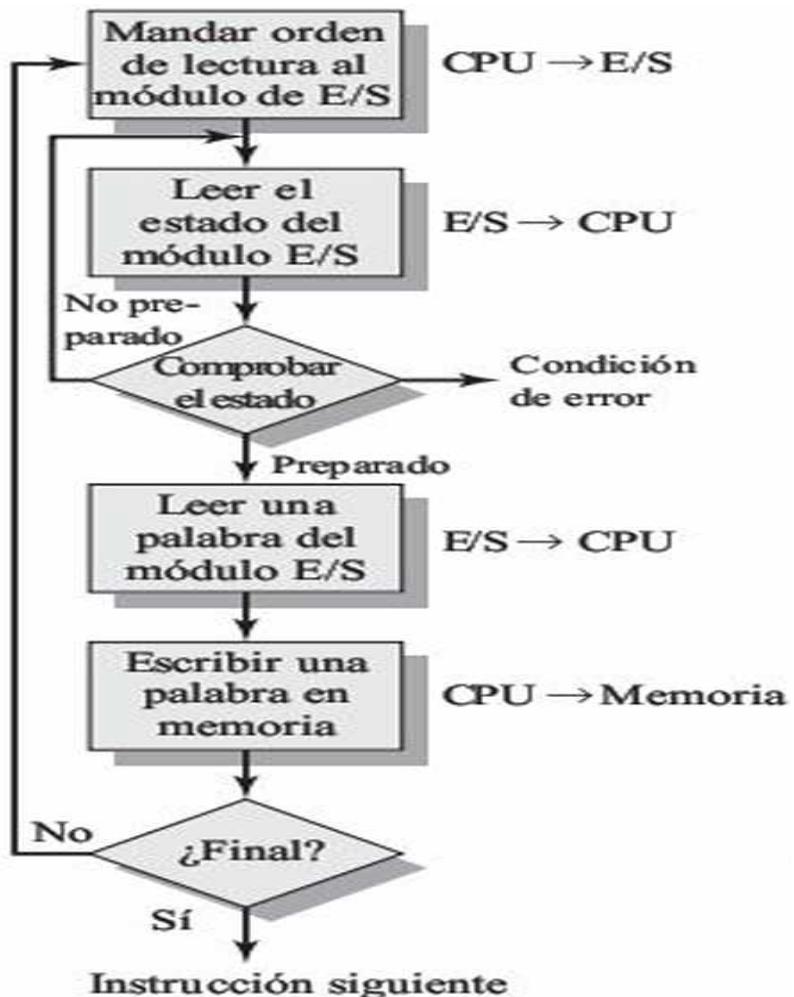
En concreto, no interrumpe al procesador. De esta forma, el procesador es responsable de comprobar periódicamente el estado del módulo de E/S hasta que encuentra que la operación ha terminado.

ÓRDENES DE E/S

Al ejecutar una instrucción relacionada con una E/S, el procesador proporciona una dirección, especificando el módulo de E/S particular , el dispositivo externo y una orden de E/S.

Hay cuatro tipos de órdenes de E/S que puede recibir un módulo de E/S cuando es direccionado por el procesador:

- *Control*: se utiliza para activar el periférico e indicarle qué hacer.
Por ejemplo, puede indicarse a una unidad de cinta magnética que se rebobine o que avance al registro siguiente.
Estas órdenes son específicas del tipo particular de periférico.
- *Test*: se utiliza para comprobar diversas condiciones de estado asociadas con el módulo de E/S y sus periféricos. El procesador podrá comprobar si el periférico en cuestión está conectado y disponible para su uso.
También podrá saber si la operación de E/S más reciente ha terminado y si se ha producido algún error.
- *Lectura*: hace que el módulo de E/S capte un dato de un periférico y lo sitúe en un buffer interno o registro de datos.
Después, el procesador puede obtener el dato solicitando que el módulo de E/S lo ponga en el bus de datos.
- *Escritura*: hace que el módulo de E/S capte un dato (byte o palabra) del bus de datos y posteriormente lo transmita al periférico



Los datos se leen palabra a palabra (16 bits, por ejemplo).

Por cada palabra leída, el procesador debe permanecer en un ciclo de comprobación de estado hasta que determine que la palabra está disponible en el registro de datos del módulo de E/S.

La principal desventaja de esta técnica: es un proceso que consume tiempo y mantiene al procesador innecesariamente ocupado.

INSTRUCCIONES DE E/S.

En la E/S programada, hay una estrecha correspondencia entre las instrucciones de E/S que el procesador capta de memoria y las órdenes de E/S que el procesador envía a un módulo de E/S al ejecutar las instrucciones.

Es decir, las instrucciones se pueden hacer corresponder fácilmente con las órdenes de E/S, y a menudo hay una simple relación de uno a uno.

La forma de la instrucción depende de la manera de direccionar los dispositivos externos.

Normalmente, habrá muchos dispositivos de E/S conectados al sistema a través de los módulos de E/S. Cada dispositivo tiene asociado un identificador único o dirección.

Cuando el procesador envía una orden de E/S, la orden contiene la dirección del dispositivo deseado.

Así, cada módulo de E/S debe interpretar las líneas de dirección para determinar si la orden es para él.

Cuando el procesador, la memoria principal, y las E/S comparten un bus común, son posibles dos modos de direccionamiento: asignado en memoria (*memory-mapped*) y *aislado*.

Con las E/S *asignadas en memoria*, existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S.

El procesador considera a los registros de estado y de datos de los módulos de E/S como posiciones de memoria y utiliza las mismas instrucciones máquina para acceder tanto a memoria como a los dispositivos de E/S.

Así, por ejemplo, con diez líneas de dirección, se puede acceder a un total de $2^{10} = 1024$ posiciones de memoria y direcciones de E/S, en cualquier combinación.

Con las E/S asignadas en memoria, se necesita una sola línea de lectura y una sola línea de escritura en el bus.

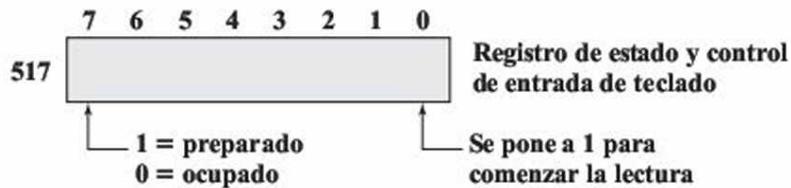
Alternativamente, el bus puede disponer de líneas de lectura y escritura en memoria junto con líneas para órdenes de entrada y salida.

En este caso, las líneas de órdenes especifican si la dirección se refiere a una posición de memoria o a un dispositivo de E/S.

El rango completo de direcciones está disponible para ambos.

Para el caso de *E/S aislada* y de nuevo con diez líneas de dirección, el sistema puede soportar ahora 1024 posiciones de memoria y 1024 direcciones de E/S.

Puesto que el espacio de direcciones de E/S está aislado del de memoria.



DIRECCIÓN	INSTRUCCIÓN	OPERANDO	COMENTARIO
200	Load AC	"1"	
	Store AC	517	Comenzar la lectura de teclado
202	Load AC	517	Obtener el byte de estado
	Branch if Sign = 0	202	Repetir bucle hasta estar preparado
	Load AC	516	Cargar un byte de datos

(a) E/S asignada en memoria

DIRECCIÓN	INSTRUCCIÓN	OPERANDO	COMENTARIO
200	Load I/O	5	Comenzar la lectura del teclado
201	Test I/O	5	Comprobar si se ha acabado
	Branch Not Ready	201	Repetir el bucle hasta acabar
	In	5	Cargar un byte de datos

(b) E/S aislada

Suponer con un dispositivo de entrada sencillo, tal como un teclado cuando se utiliza *E/S asignada en memoria*.

Se asumen direcciones de diez bits, con una memoria de 512 palabras (posiciones 0-511) y hasta 512 direcciones de E/S (posiciones 512-1023).

Se dedican dos direcciones a la entrada de teclado desde un terminal concreto.

La dirección 516 se refiere al registro de datos y la dirección 517 al registro de estado, que además funciona como registro de control para recibir las órdenes del procesador.

El programa que se muestra lee un byte de datos desde el teclado y lo escribe en el registro acumulador del procesador. El procesador ejecuta un bucle hasta que el byte de datos está disponible.

Con *E/S aislada*, los puertos de E/S solo son accesibles mediante una orden específica de E/S, que activa las líneas de órdenes de E/S del bus.

La mayor parte de procesadores disponen de un conjunto relativamente grande de instrucciones distintas para acceder a memoria.

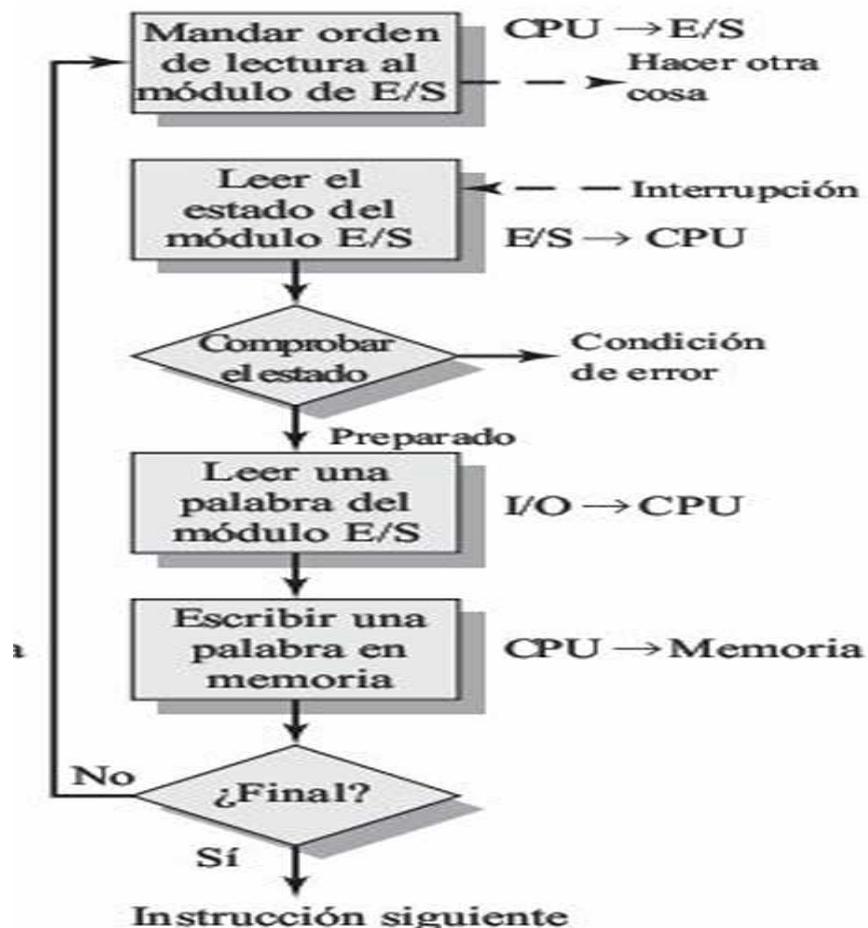
Si se utiliza E/S aislada, solo existen unas pocas instrucciones de E/S.

Por eso, una ventaja de la *E/S asignada en memoria* es que se puede utilizar este amplio repertorio de instrucciones, permitiendo una programación más eficiente.

Una desventaja es que se utiliza parte del valioso espacio de direcciones de memoria.

Tanto la E/S asignada en memoria como la aislada son empleados comúnmente.

E/S MEDIANTE INTERRUPCIONES.



El problema con la E/S programada es que el procesador tiene que esperar un tiempo considerable a que el módulo de E/S en cuestión esté preparado para recibir o transmitir los datos.

El procesador, mientras espera, debe comprobar repetidamente el estado del módulo de E/S. Como consecuencia, se degrada el nivel de prestaciones de todo el sistema.

Una alternativa consiste en que el procesador, tras enviar una orden de E/S a un módulo, continúe realizando algún trabajo útil. Después, el módulo de E/S interrumpirá al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él. El procesador ejecuta entonces la transferencia de datos, como antes, y después continúa con el procesamiento previo.

Desde el punto de vista del módulo de E/S ,para una entrada, el módulo de E/S recibe una orden READ del procesador.

Entonces, el módulo de E/S procede a leer el dato desde el periférico asociado.

Una vez que el dato está en el registro de datos del módulo, el módulo envía una interrupción al procesador a través de una línea de control.

Después, el módulo espera hasta que el procesador solicite su dato.

Cuando ha recibido la solicitud, el módulo sitúa su dato en el bus de datos y pasa a estar preparado para otra operación de E/S.

Desde el punto de vista del procesador, el procesador envía una orden READ de lectura. Entonces pasa a realizar otro trabajo (es decir, el procesador puede estar ejecutando programas distintos al mismo tiempo).

Al final de cada ciclo de instrucción, el procesador comprueba las interrupciones.

Cuando se pide la interrupción desde el módulo de E/S, el procesador guarda el contexto (es decir, el contador de programa y los registros del procesador) del programa en curso y procesa la interrupción.

En este caso, el procesador lee la palabra de datos del módulo de E/S y la almacena en memoria.

Después recupera el contexto del programa que estaba ejecutando (o de otro programa) y continúa su ejecución.

No obstante, las E/S con interrupciones todavía consumen gran cantidad del tiempo del procesador puesto que cada palabra de datos que va desde la memoria al módulo de E/S o viceversa debe pasar a través del procesador.

PROCESAMIENTO DE LA INTERRUPCIÓN.

Cuando se produce una interrupción se disparan una serie de eventos en el procesador, tanto a nivel hardware como software.

Cuando un dispositivo de E/S termina una operación de E/S, se produce la siguiente secuencia de eventos en el hardware:

1. El dispositivo envía una señal de interrupción al procesador.
2. El procesador termina la ejecución de la instrucción en curso antes de responder a la interrupción.
3. El procesador comprueba si hay interrupciones, determina que hay una, y envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción.
4. Ahora el procesador necesita prepararse para transferir el control a la rutina de interrupción.
Para empezar, debe guardar la información necesaria para continuar el programa en curso en el punto en que se interrumpió.
La información mínima que se precisa es :
 - a) el estado del procesador, que se almacena en un registro llamado Palabra de Estado del Programa (*PSW, Program Status Word*).
 - b) la posición de la siguiente instrucción a ejecutar, que está contenida en el contador de programa. Estos registros se pueden introducir en la pila de control del sistema.

5. Después, el procesador carga el contador de programa con la posición de inicio del programa de gestión de la interrupción solicitada.

Según sea la arquitectura del computador y el diseño del sistema operativo, puede haber un solo programa, uno por cada tipo de interrupción, o uno por cada dispositivo y cada tipo de interrupción.

Si hay más de una rutina de gestión de interrupción, el procesador debe determinar a qué programa llamar.

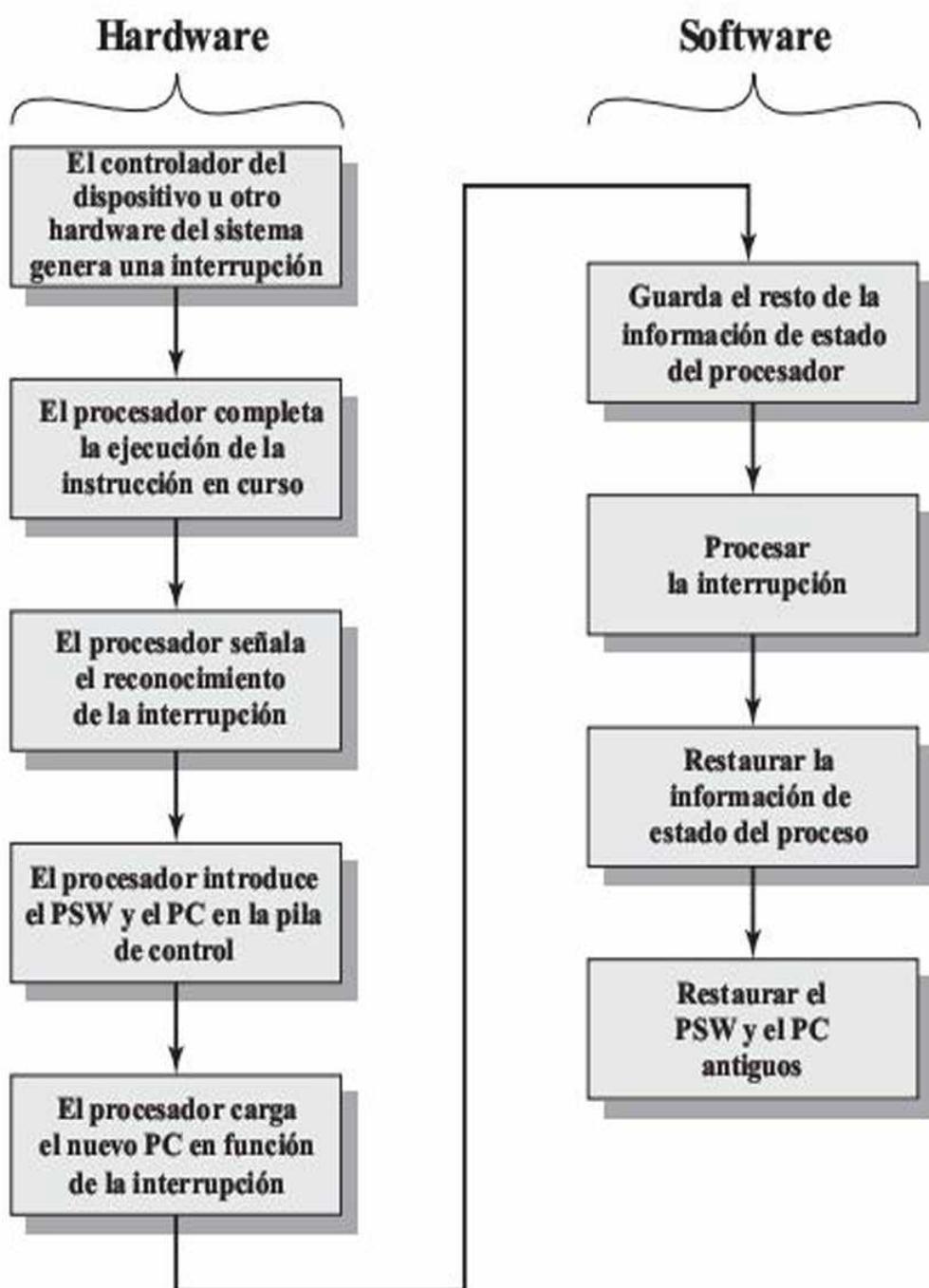
Esta información puede haber sido incluida en la señal de interrupción original, o el procesador puede tener que enviar una solicitud al dispositivo que originó la interrupción para que este responda con la información que se precise.

Una vez que el contador de programa se ha cargado, el procesador continúa con el ciclo de instrucción siguiente, que empieza con la captación de instrucción.

Puesto que la instrucción a captar viene determinada por el contenido del contador de programa, el control se transfiere al programa de gestión de interrupción.

La ejecución de este programa da lugar a las siguientes operaciones:

6. Hasta este momento, se han almacenado en la pila del sistema el contador de programa y el PSW del programa interrumpido. Sin embargo, hay otra información que se considera parte del estado de un programa en ejecución. En concreto, se deben guardar los contenidos de los registros del procesador puesto que estos registros pueden ser utilizados por la rutina de interrupción. Usualmente, la rutina de gestión de interrupción empezará almacenando en la pila los contenidos de todos los registros.
7. La rutina de gestión de la interrupción puede continuar ahora procesando la interrupción.
Esto incluirá el examen de la información de estado relativa a la operación de E/S o a cualquier otro evento que causara la interrupción.
También puede implicar el envío al dispositivo de E/S de órdenes o señales de reconocimiento adicionales.
8. Cuando el procesamiento de la interrupción ha terminado, los valores de los registros almacenados se recuperan de la pila y se vuelven a almacenar en los registros.
9. El paso final es recuperar los valores del PSW y del contador de programa desde la pila.
Como resultado, la siguiente instrucción que se ejecute pertenecerá al programa previamente interrumpido.



Es importante almacenar toda la información del estado del programa interrumpido para que este pueda reanudarse. Esto se debe a que la interrupción no es una llamada a una rutina realizada desde el programa.

En cambio, la interrupción puede producirse en cualquier momento y por consiguiente en cualquier punto de la ejecución del programa de usuario.

Una interrupción es impredecible.

En la implementación de las E/S mediante interrupciones aparecen dos consideraciones.

Primero, puesto que casi invariablemente habrá múltiples módulos de E/S, cómo determina el procesador qué dispositivo ha provocado la interrupción ?

Segundo, si se han producido varias interrupciones, cómo decide el procesador la que debe atender.

Consideremos en primer lugar la identificación del dispositivo.

Hay cuatro tipos de técnicas que se utilizan comúnmente:

- Múltiples líneas de interrupción.
- Consulta software (*software poll*).
- Conexión en cadena, (*Daisy chain*), (Consulta hardware, vectorizada).
- Arbitraje de bus (vectorizada).

La aproximación más directa al problema consiste en proporcionar *varias líneas de interrupción* entre el procesador y los módulos de E/S.

Sin embargo, no resulta práctico dedicar más de unas pocas líneas del bus o terminales del procesador a ser líneas de interrupción.

En consecuencia, incluso si se utilizan varias líneas, es probable que a cada una se conecten varios módulos de E/S.

Por ello, se debe utilizar alguna de las otras tres técnicas en cada línea.

Una alternativa es la *consulta software*.

Cuando el procesador detecta una interrupción, se produce una bifurcación a una rutina de servicio de interrupción que se encarga de consultar a cada módulo de E/S para determinar el módulo que ha provocado la interrupción.

La consulta podría realizarse mediante una línea específica (por ejemplo, TEST E/S). En este caso, el procesador activa TEST E/S y sitúa la dirección de un módulo de E/S en las líneas de dirección.

El módulo de E/S responde positivamente si solicitó la interrupción.

Como alternativa, cada módulo de E/S podría disponer de un registro de estado direccionable. Entonces, el procesador lee el estado del registro de cada módulo de E/S para identificar el módulo que solicitó la interrupción.

Una vez identificado el módulo, se produce una bifurcación para que el procesador ejecute la rutina de servicio específica para ese dispositivo.

La desventaja de la consulta software está en el tiempo que consume.

Una técnica más eficiente consiste en utilizar la *conexión en cadena* (*daisy chain*) de los módulos de E/S que proporciona, de hecho, una consulta hardware.

Todos los módulos de E/S comparten una línea común para solicitar interrupciones. La línea de reconocimiento de interrupción se conecta encadenando los módulos uno tras otro.

Cuando el procesador recibe una interrupción, activa el reconocimiento de interrupción. Esta señal se propaga a través de la secuencia de módulos de E/S hasta que alcanza un módulo que solicitó interrupción.

Normalmente este módulo responde colocando una palabra en las líneas de datos. Esta palabra se denomina *vector* y es la dirección del módulo de E/S o algún otro tipo de identificador específico. En cualquier caso, el procesador utiliza el vector como un puntero a la rutina de servicio de dispositivo apropiada.

Así se evita tener que ejecutar una rutina de servicio general en primer lugar. Esta técnica se conoce con el nombre de *interrupción vectorizada*.

Hay otra técnica que hace uso de las interrupciones vectorizadas, y se trata de el *arbitraje de bus*.

Con el arbitraje de bus, un módulo de E/S debe en primer lugar disponer del control del bus antes de poder activar la línea de petición de interrupción. Así, solo un módulo puede activar la línea en un instante.

Cuando el procesador detecta la interrupción, responde mediante la línea de reconocimiento de interrupción.

Después, el módulo que solicitó la interrupción sitúa su vector en las líneas de datos.

Las técnicas enumeradas arriba sirven para identificar el módulo de E/S que solicita interrupción.

Además proporcionan una forma de asignar prioridades cuando más de un dispositivo está pidiendo que se sirva su interrupción.

Con varias líneas de interrupción, el procesador simplemente selecciona la línea con más prioridad.

Con la consulta software, el orden en el que se consultan los módulos determina su prioridad.

De igual forma, el orden de los módulos en la conexión en cadena (*daisy chain*) determina su prioridad.

Finalmente, el arbitraje de bus puede emplear un esquema de prioridad en base a la jerarquía de buses.

ACCESO DIRECTO A MEMORIA

La E/S con interrupciones, aunque más eficiente que la sencilla E/S programada, también requiere la intervención activa del procesador para transferir datos entre la memoria y el módulo de E/S, y cualquier transferencia de datos debe seguir un camino a través del procesador.

Por tanto, ambas formas de E/S presentan dos inconvenientes :

- La velocidad de transferencia de E/S está limitada por la velocidad a la cual el procesador puede comprobar y dar servicio a un dispositivo.
- El procesador debe dedicarse a la gestión de las transferencias de E/S; debe ejecutar cierto número de instrucciones por cada transferencia de E/S.

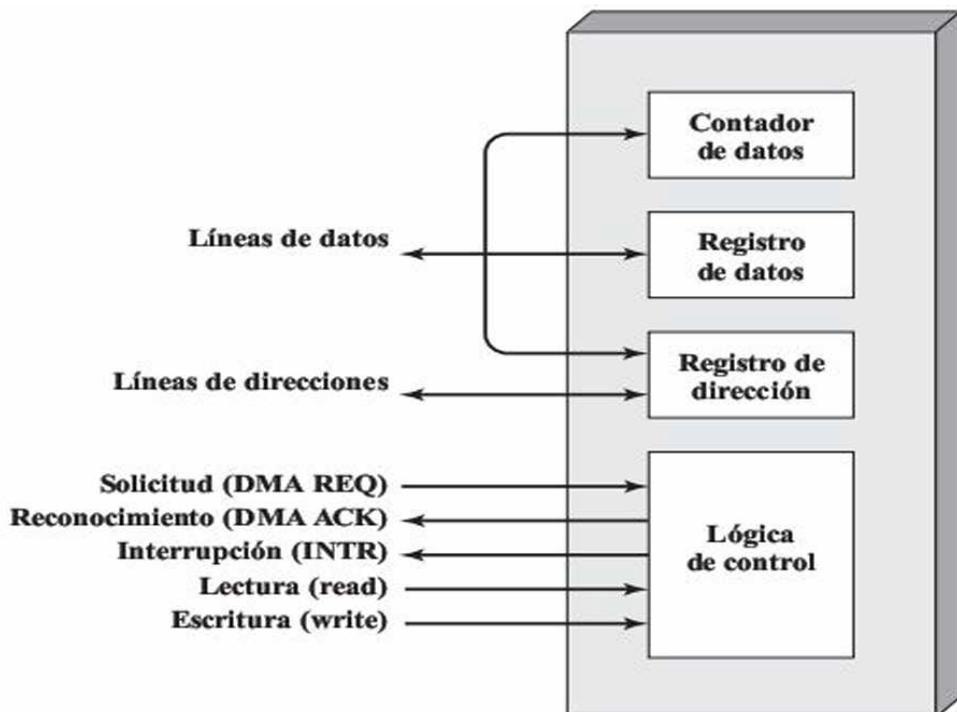
Existe un cierto compromiso entre estos dos inconvenientes.

Suponiendo una transferencia de un bloque de datos, utilizando E/S programada, el procesador se dedica a la tarea de la E/S y puede transferir datos a alta velocidad al precio de no hacer nada más.

La E/S con interrupciones libera en parte al procesador, a expensas de reducir la velocidad de E/S.

No obstante, ambos métodos tienen un impacto negativo tanto en la actividad del procesador como en la velocidad de transferencia de E/S.

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



El DMA requiere un módulo adicional en el bus del sistema conocido como *el módulo o controlador de DMA*.

Este nuevo módulo o controlador DMA es capaz de imitar al procesador y, de hecho, de recibir el control del sistema cedido por el procesador.

Necesita dicho control para transferir datos a, y desde, memoria a través del bus del sistema.

El módulo de DMA debe utilizar bus sólo cuando el procesador no lo necesita, o debe forzar al procesador a que suspenda temporalmente su funcionamiento.

Esta última técnica es la más común y se denomina *robo de ciclo (cycle stealing)*, puesto que, en efecto, el módulo de DMA roba un ciclo de bus.

Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA, incluyendo la siguiente información:

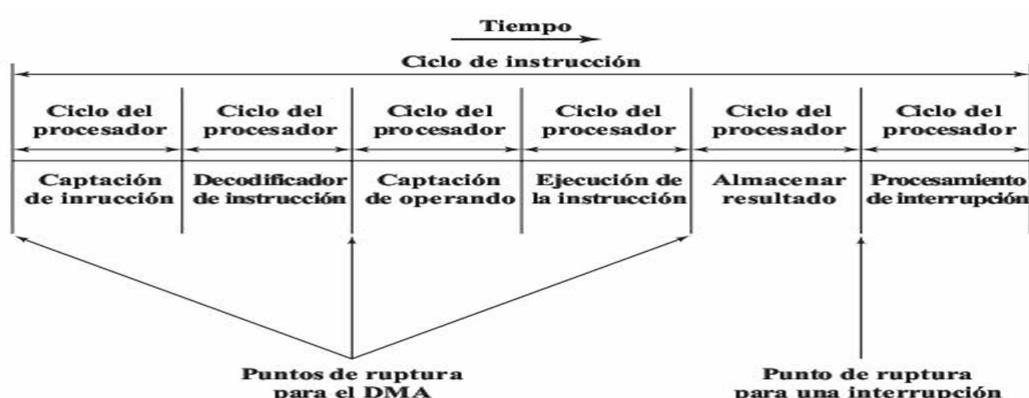
- Si se solicita una lectura o una escritura, utilizando la línea de control de lectura o escritura entre el procesador y el módulo de DMA.
- La dirección del dispositivo de E/S en cuestión, indicada a través de las líneas de datos.
- La posición inicial de memoria a partir de donde se lee o se escribe, indicada a través de las líneas de datos y almacenada por el módulo de DMA en su registro de direcciones.
- El número de palabras a leer o escribir, también indicado a través de las líneas de datos y almacenado en el registro de cuenta de datos.

Después, el procesador continúa con otro trabajo habiendo delegado la operación de E/S al módulo de DMA, que se encargará de ella.

El módulo de DMA transfiere el bloque completo de datos, palabra a palabra, directamente desde o hacia la memoria, sin que tenga que pasar a través del procesador.

Cuando la transferencia se ha terminado, el módulo de DMA envía una señal de interrupción al procesador.

Así pues, el procesador solo interviene al comienzo y al final de la transferencia.

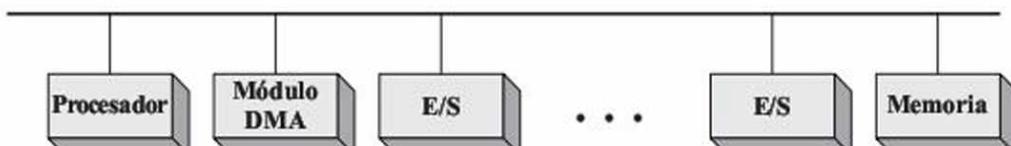


En cada caso, el procesador se detiene justo antes de necesitar el bus. Después, el módulo de DMA transfiere una palabra y devuelve el control al procesador. No se trata de una interrupción; el procesador no guarda el contexto ni hace nada más. En cambio, el procesador espera durante un ciclo de bus.

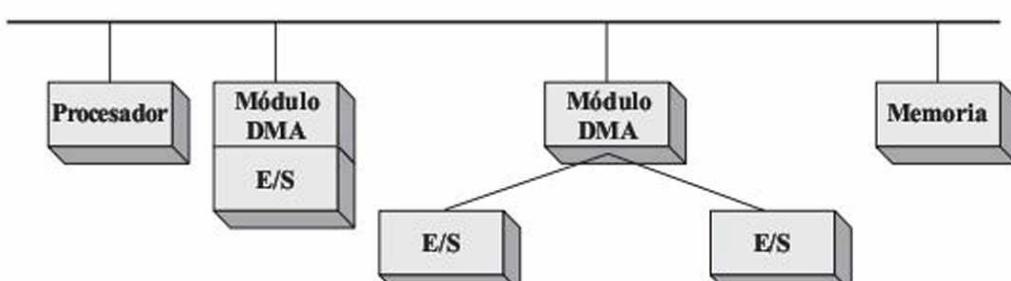
El efecto resultante es que el procesador es más lento ejecutando los programas.

No obstante, para una transferencia de E/S de varias palabras, el DMA es mucho más eficiente que la E/S mediante interrupciones o la E/S programada.

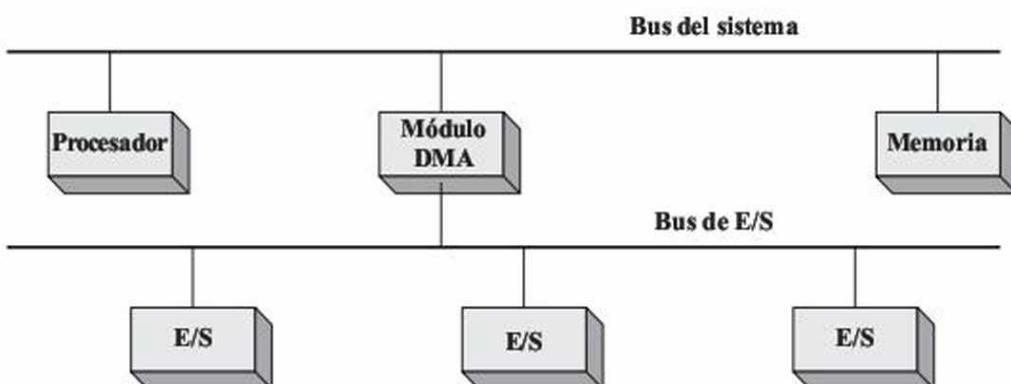
El mecanismo de DMA puede configurarse de diversas formas.



(a) Bus único, DMA independiente



(b) Bus único, DMA-E/S integrados



(c) Bus de E/S

En el primer caso, todos los módulos comparten el mismo bus del sistema. El módulo de DMA, actuando como un procesador suplementario, utiliza E/S programada para intercambiar datos entre la memoria y un módulo de E/S a través del módulo de DMA.

Esta configuración, si bien es la más económica, es claramente ineficiente. Igual que con la E/S programada controlada por el procesador, la transferencia de cada palabra consume dos ciclos de bus.

En el segundo caso los ciclos de bus necesarios puede reducirse si se integran las funciones de DMA y de E/S , esto significa que existe un camino entre el módulo de DMA y uno o más módulos de E/S que no incluye al bus del sistema. La lógica de DMA puede ser parte de un módulo de E/S, o puede ser un módulo separado que controla a uno o más módulos de E/S.

En el tercer caso se conectan los módulos de E/S a un módulo de DMA mediante un bus de E/S . Esto reduce a uno el número de interfaces de E/S en el módulo de DMA y permite una configuración fácilmente ampliable.

En todos estos casos el bus del sistema, que el módulo de DMA comparte con el procesador y la memoria, es usado por el módulo de DMA solo para intercambiar datos con la memoria.

El intercambio de datos entre los módulos de DMA y E/S se produce fuera del bus del sistema.

CANALES Y PROCESADORES DE E/S.

A medida que los computadores han evolucionado, la complejidad y sofisticación de sus componentes se ha incrementado.

Se hace más evidente en el funcionamiento de las E/S y se pueden resumir en la forma siguiente :

La CPU controla directamente al periférico.

Esta situación se observa en los dispositivos simples controlados por microprocesadores.

Se añade un controlador o módulo de E/S.

La CPU utiliza E/S programada sin interrupciones. De esta forma, la CPU se independiza de los detalles específicos de las interfaces de los dispositivos externos.

Se emplean interrupciones.

La CPU no necesita esperar a que se realice la operación de E/S, incrementándose la eficiencia.

Acceso directo a la memoria a través del DMA.

Se puede transferir un bloque de datos a, o desde, la memoria sin implicar a la CPU, excepto al comienzo y al final de la transferencia.

El módulo de E/S se mejora haciendo que se comporte como un procesador en sí.

Tiene un repertorio especializado de instrucciones orientado a las E/S.

La CPU hace que el procesador de E/S ejecute un programa de E/S en memoria.

El procesador de E/S capta y ejecuta sus instrucciones sin intervención de la CPU.

Esto permite que la CPU pueda especificar una secuencia de actividades de E/S y ser interrumpida cuando se haya completado la secuencia entera.

El módulo de E/S tiene una memoria local propia y es, de hecho, un computador en sí.

Con esta arquitectura, se puede controlar un conjunto grande de dispositivos de E/S con la mínima intervención de la CPU.

Un uso común de este tipo de arquitectura ha sido la comunicación con terminales interactivos. El procesador de E/S se ocupa de la mayoría de las tareas correspondientes al control de los terminales.

Siguiendo el camino marcado por esta evolución, cada vez más funciones de E/S se realizan sin la intervención de la CPU.

La CPU se releva de las tareas relacionadas con las tareas de E/S, mejorando las prestaciones. Con las dos últimas etapas, se ha producido un cambio importante al introducir el concepto de un módulo de E/S capaz de ejecutar un programa.

Y el módulo E/S normalmente ya se denomina *canal de E/S*.

CARACTERÍSTICAS DE LOS CANALES DE E/S.

El canal de E/S representa una ampliación del concepto de DMA. Puede ejecutar instrucciones de E/S, lo que le confiere un control completo sobre las operaciones de E/S.

En un computador con tales dispositivos, la CPU no ejecuta instrucciones de E/S.

Dichas instrucciones se almacenan en memoria principal para ser ejecutadas por un procesador de uso específico contenido en el propio canal de E/S.

De esta forma, la CPU inicia una transferencia de E/S indicando al canal de E/S que debe ejecutar un programa de la memoria.

El programa especifica el dispositivo o dispositivos ,el área o áreas de memoria para almacenamiento, la prioridad, y las acciones a realizar en ciertas situaciones de error.

El canal de E/S sigue estas instrucciones y controla la transferencia de datos.

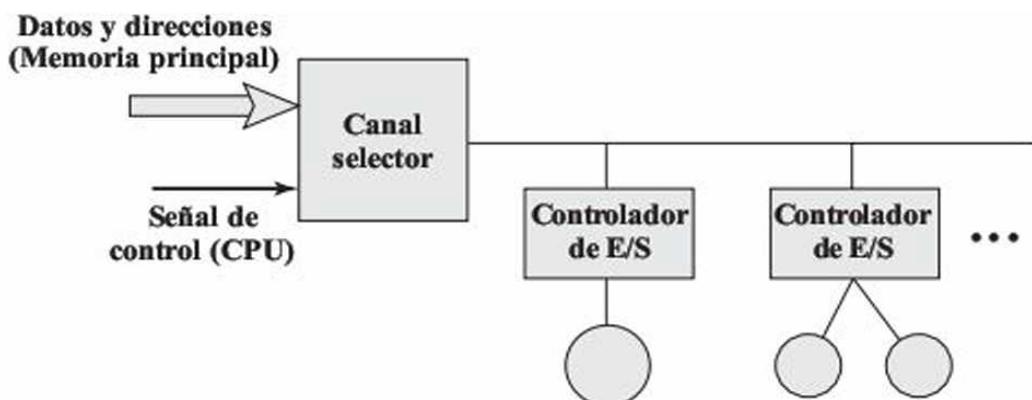
Son comunes dos tipos de canales de E/S :

Un *canal selector* controla varios dispositivos de velocidad elevada y en un instante dado se dedica a transferir datos a uno de esos dispositivos.

Es decir, el canal de E/S selecciona un dispositivo y efectúa la transferencia de datos.

Cada dispositivo, o pequeño grupo de dispositivos, es manejado por un *controlador*, o módulo de E/S, que es similar a los módulos de E/S que se han comentado.

Así, el canal de E/S se utiliza en lugar de la CPU para controlar estos controladores de E/S.



Un *canal multiplexor* puede manejar las E/S de varios dispositivos al mismo tiempo. Para dispositivos de velocidad reducida, un *multiplexor de byte* acepta o transmite caracteres tan rápido como es posible a varios dispositivos.

Por ejemplo, la cadena de caractéres resultante a partir de tres dispositivos con diferentes velocidades y cadenas individuales $A_1, A_2A_3A_4\dots, B_1, B_2B_3B_4\dots$, y $C_1C_2C_3C_4\dots$, podría ser $A_1 B_1 C_1 A_2 C_2 A_3, B_2 C_3 A_4$ y así sucesivamente.

Para dispositivos de velocidad elevada, un *multiplexor de bloque* entrelaza bloques de datos de los distintos dispositivos.

