Sistema de entrada/salida

Miquel Albert Orenga
Gerard Enrique Manonellas

PID_00177074

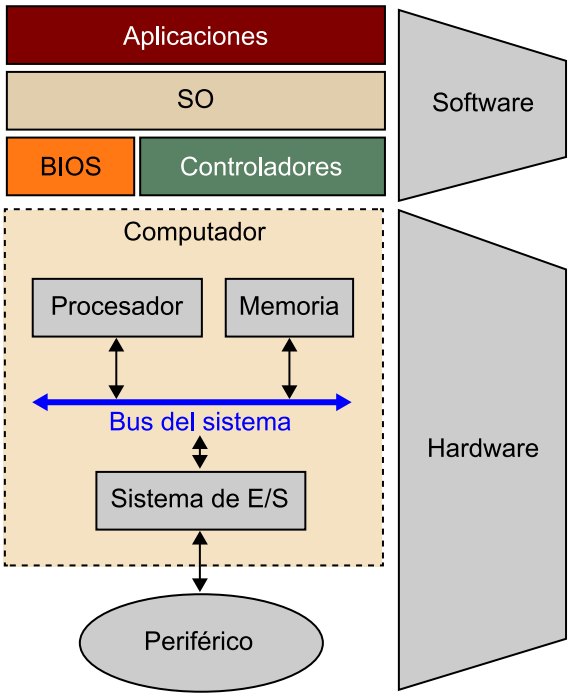
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Todo computador necesita llevar a cabo intercambio de información con personas u otros computadores mediante unos dispositivos que denominamos de manera genérica **dispositivos periféricos**. Para hacer una operación de E/S entre el computador y un periférico, es necesario conectar estos dispositivos al computador y gestionar de manera efectiva la transferencia de datos. Para hacerlo, el computador dispone del **sistema de entrada/salida (E/S)**.

Este sistema de E/S es la interfaz que tiene el computador con el exterior y el objetivo que tiene es facilitar las operaciones de E/S entre los **periféricos** y la **memoria** o los **registros del procesador**. Para gestionar las operaciones de E/S es necesario un hardware y la ayuda de un software.



Dada la gran variedad de periféricos, es necesario dedicar un hardware y un software específicos para cada uno. Por este motivo se ha intentado normalizar la interconexión de los periféricos y el computador mediante lo que se denomina **módulos de E/S** o **controladores de E/S**. Eso nos permite tener, por una parte, una conexión, entre el módulo de E/S y el periférico, específica y con unas características propias que difícilmente se pueden generalizar para utilizarlas en otros dispositivos y, por otra parte, una conexión entre los módulos de E/S y el computador común a todos los controladores, pero estos módulos, además de permitir la conexión de los periféricos al computador, disponen de la lógica necesaria para tener cierta capacidad de procesamiento y gestionar las transferencias de información.

Hay que tener presente que la gestión global del sistema de E/S de un computador la hace el sistema operativo (SO). Las técnicas para controlar este sistema de E/S las utiliza el SO y el programador cuando quieren acceder al periférico, pero en las máquinas actuales, a causa de la complejidad de controlar y gestionar los periféricos, el acceso se lleva a cabo generalmente mediante llamadas al SO, que es quien gestiona la transferencia. El conjunto de rutinas que permiten controlar un determinar periférico es lo que denominamos habitualmente **programas controladores** o ***drivers*** y cuando el SO quiere hacer una operación de E/S con un periférico llama a una de estas rutinas.

Este módulo se centra en el estudio del sistema de E/S y hablaremos de las principales técnicas utilizadas y de qué características debe tener el hardware y el software necesario para gestionar las diferentes maneras de realizar la transferencia de información entre el computador y el periférico.

Objetivos

Con el estudio de este módulo se pretende que el estudiante alcance los objetivos siguientes:

1. Conocer los aspectos básicos del sistema de E/S de un computador.

2. Aprender las técnicas básicas de E/S.

3. Entender las ventajas de cada una de estas técnicas para mejorar el rendimiento del computador.

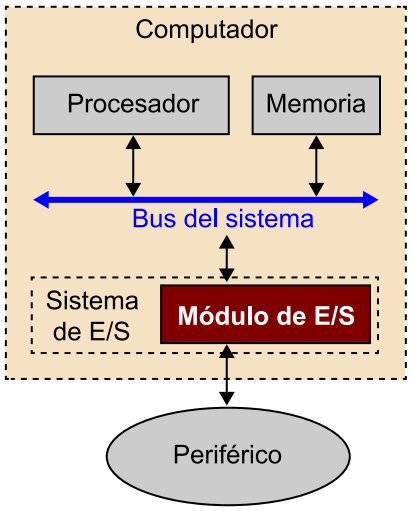
4. Tener unos conocimientos básicos de los tipos de dispositivos que podemos conectar al computador y cómo se comunican con el procesador mediante el sistema de E/S.

1. Aspectos básicos del E/S

Cuando hablamos de E/S de información entre un computador y un periférico lo hacemos siempre desde el punto de vista del computador. Así, decimos que es una **transferencia de entrada** cuando el periférico es el emisor de la información y tiene como receptor el computador (procesador o memoria) y decimos que es una **transferencia de salida** cuando el computador es el emisor de la información y tiene como receptor el periférico.

De manera más concreta, toda operación de E/S que se lleva a cabo entre el computador y un periférico es solicitada y gobernada desde el procesador, es decir, es el procesador quien determina en qué momento se debe hacer y con qué periférico, si la operación es de lectura o escritura, qué datos se han de transferir, y también quién da la operación por acabada.

Para llevar a cabo la operación de E/S, hemos de conectar el periférico al computador. Para hacerlo, es necesario que el computador disponga de unos dispositivos intermedios por donde ha de pasar toda la información que intercambia el computador con el periférico y que nos permite hacer una gestión y un control correctos de la transferencia. Estos dispositivos los llamamos de manera genérica **módulo de E/S**.



Puede parecer lógico conectar el periférico directamente al bus del sistema del computador, pero esta opción no es factible básicamente por dos razones:

- La necesidad de gestionar una gran variedad de periféricos con unas características muy específicas y diferenciadas. Esto hace muy complejo añadir

la lógica necesaria dentro del procesador para gestionar esta gran diversidad de dispositivos.

- La diferencia de velocidad entre sí, en la que, salvo casos excepcionales, el procesador es mucho más rápido que el periférico. Por un lado, hay que asegurar que no se pierdan datos y, por otro, garantizar principalmente la máxima eficiencia del procesador, pero también de los otros elementos del computador.

Así pues, para hacer una operación de E/S, el módulo de E/S nos debe permitir establecer, por una parte, **mecanismos de control** para determinar el inicio y el final de la operación de E/S, la cantidad de información que hay que transmitir, la detección de errores, etc., y, por otra parte, **mecanismos para hacer la transferencia de datos** considerando aspectos como la manera de dirigir el periférico, la conversión serie/paralela de la información, la conversión de códigos, la sincronización, etc. Estos mecanismos se reparten entre la unidad de control del procesador, el módulo de E/S y los programas de E/S.

Cuando queremos hacer la operación de E/S, hemos de diferenciar el caso de una transferencia individual, en la que se transmite un solo dato y el control de la transferencia es muy simple (leer una tecla, mirar si se ha hecho un clic en el ratón), y la transferencia de bloques, que se basa en una serie de transferencias individuales y en la que se necesita un control mayor de todo el proceso (leer un fichero, actualizar el contenido de la pantalla).

Otro aspecto importante que hay que considerar, dado que podemos tener conectados al computador una gran variedad de periféricos, es que si se desencadenan operaciones de E/S de manera simultánea, el sistema de E/S del computador debe disponer de los mecanismos necesarios para gestionarlas sin que se pierdan datos.

1.1. Estructura del sistema de E/S del computador

Los elementos principales que forman el sistema de E/S son los siguientes:

- los periféricos,
- los módulos de E/S,
- los sistemas de interconexión externos y
- el mapa de memoria e instrucciones de E/S.

A continuación haremos una breve descripción de estos elementos y de cómo interactúan entre sí.

1.1.1. Periféricos

Los periféricos son dispositivos que se conectan al computador mediante los módulos de E/S y que sirven para almacenar información o para llevar a cabo un tipo determinado de comunicación con el exterior con humanos, con máquinas o con otros computadores.

La clasificación más habitual es la siguiente:

- Para la interacción con humanos:
  - Entrada.
  - Salida.

- Para la interacción con otros computadores o sistemas físicos (en los que las operaciones que se hacen son generalmente de E/S):
  - Almacenamiento.
  - Comunicación.

En un periférico distinguimos habitualmente dos partes: una parte mecánica y una parte electrónica. La parte mecánica hace funcionar los elementos principales que forman el periférico, como el motor para hacer girar un disco o mover el cabezal de una impresora, el botón de un ratón o el láser de un dispositivo óptico. La parte electrónica nos permite, por una parte, generar las señales eléctricas para gestionar los elementos mecánicos y, por otra parte, hacer la conversión de los datos provenientes del computador a señales eléctricas o al revés.

La conexión física entre un periférico y el computador se lleva a cabo mediante lo que denominamos **sistema de interconexión de E/S**. Este sistema de interconexión de E/S nos permite hacer la gestión de las señales de control, de estado y de datos necesarias para llevar a cabo una transferencia de información que, como veremos más adelante, es gestionada desde el módulo de E/S del computador.

En este módulo nos centraremos en analizar la transferencia de información entre un periférico y el computador mediante los módulos de E/S.

Nota

En este módulo no analizaremos la estructura y el funcionamiento del periférico; habrá suficiente con ver el periférico como un elemento capaz de enviar o recibir una determinada cantidad de información, a una determinada velocidad, mediante el sistema de interconexión de E/S. No analizaremos tampoco las características técnicas que permiten al periférico tener estas prestaciones.

La cantidad de información que puede enviar o recibir el periférico por unidad de tiempo la denominamos **velocidad de transferencia** y generalmente se expresa en bits o bytes por segundo.

La velocidad de transferencia puede ir de unos pocos bits por segundo a gigabytes por segundo, pero hemos de tener presente que un computador puede llegar a trabajar a velocidades bastante superiores y hemos de garantizar que durante una transferencia no se pierdan datos.

1.1.2. Módulos de E/S

Un módulo de E/S es un controlador de uno o varios periféricos que establece una interfaz entre el periférico y el computador (procesador y memoria) para facilitar la comunicación entre el uno y el otro de manera que buena parte de los detalles técnicos del periférico queden ocultos al resto del computador.

Nota

La complejidad de un módulo de E/S puede variar mucho. Por este motivo, aquí haremos una descripción general de las partes y características básicas más importantes. Veremos otras características más específicas cuando analicemos las diferentes técnicas de E/S.

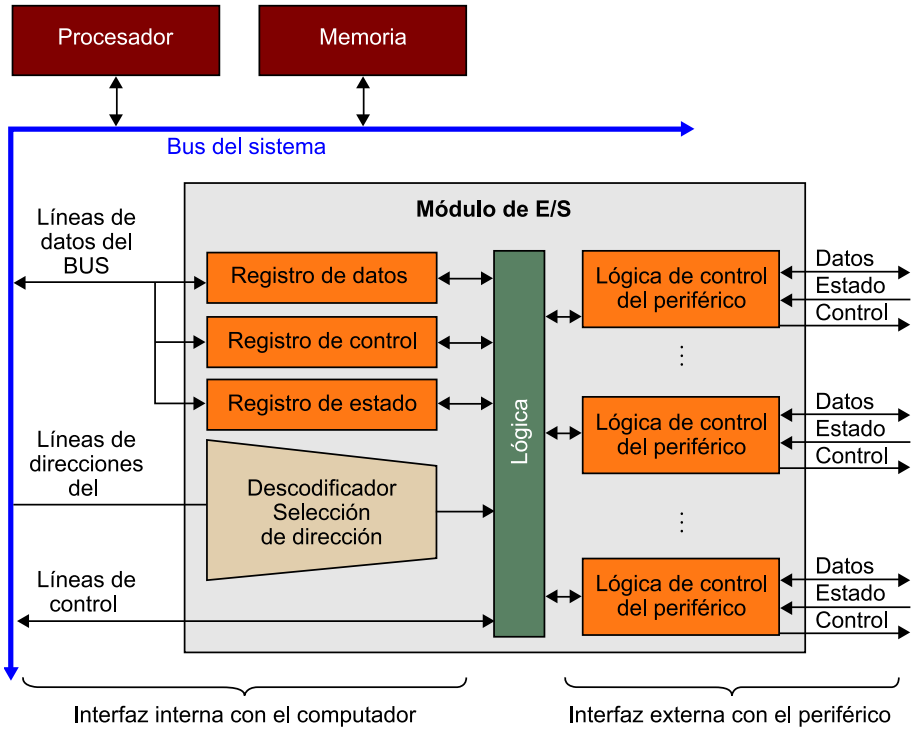
Del módulo de E/S distinguimos tres partes básicas:

1) Una interfaz interna normalizada con el resto del computador mediante el bus de sistema que nos da acceso al banco de registros del módulo de E/S.

2) Una interfaz externa específica para el periférico que controla. Habitualmente la conexión con el periférico se realiza mediante un sistema de interconexión normalizado de E/S.

3) La lógica necesaria para gestionar el módulo de E/S. Es responsable del paso de información entre la interfaz interna y externa.

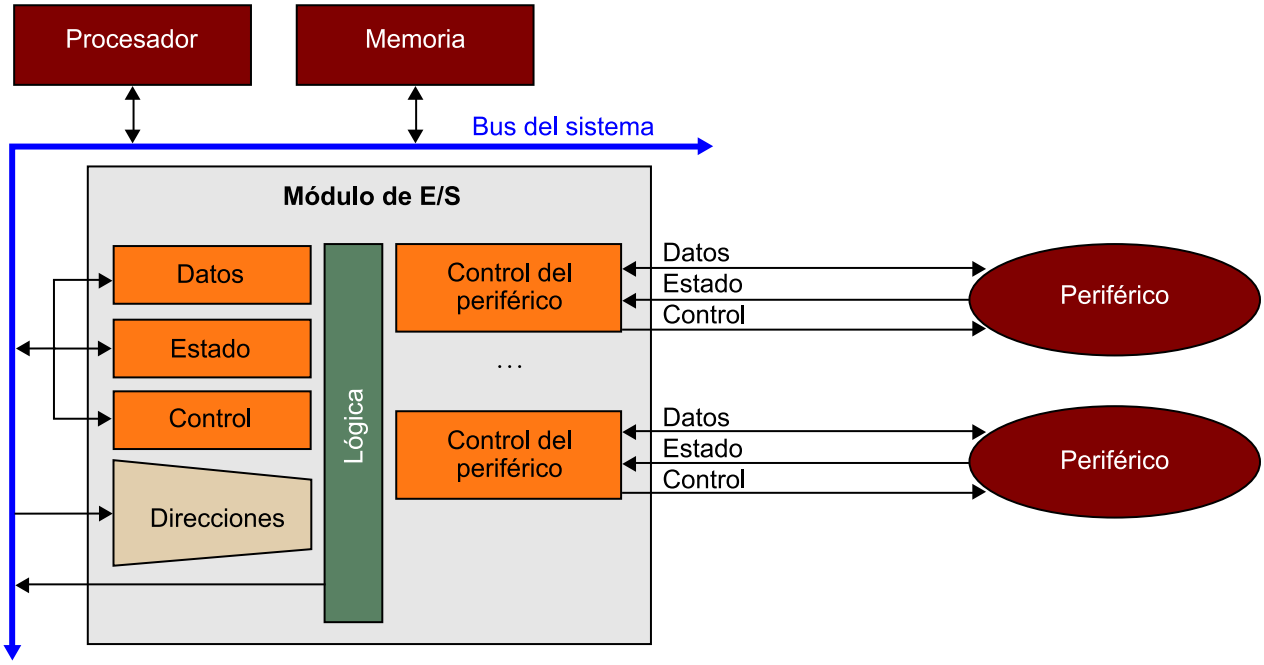
En la siguiente figura podéis ver el esquema general de un módulo de E/S.



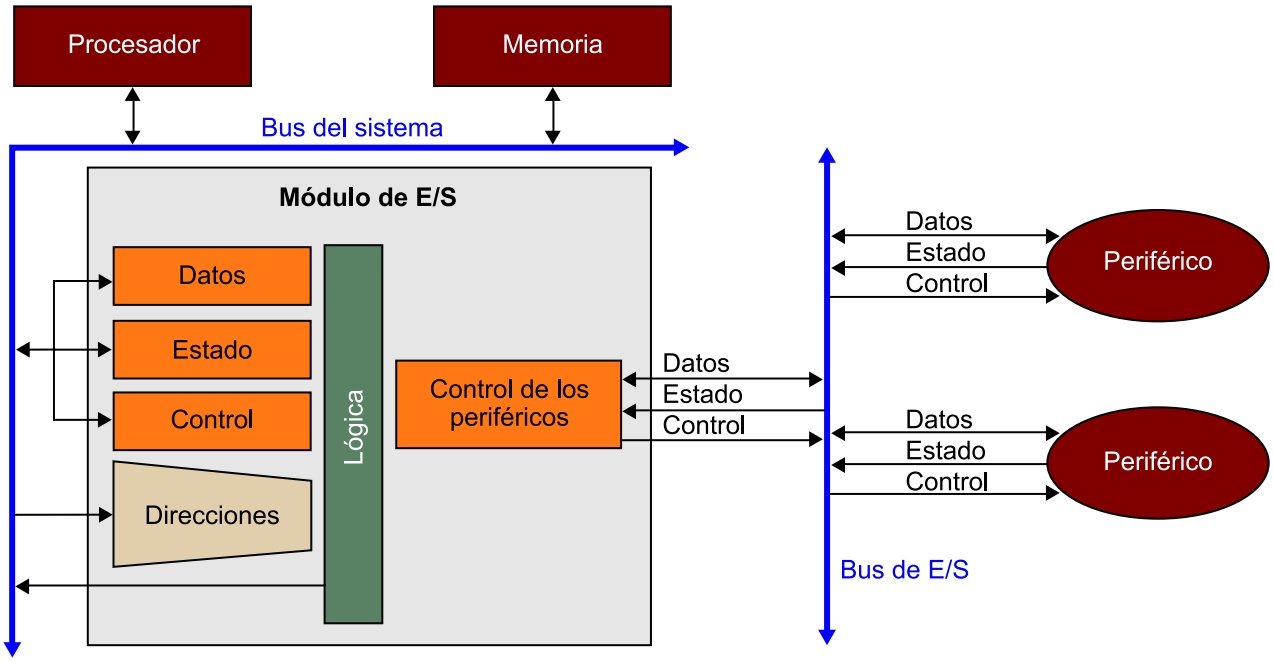
La forma de comunicación entre el módulo de E/S y el periférico es específica para cada periférico. Lógicamente, depende de las características del periférico que queremos controlar, pero también del sistema de interconexión utilizado para comunicarse. Esta conexión tiene habitualmente unas especificaciones normalizadas y adaptadas al tipo de transferencia que se debe realizar y lo denominamos **sistema de interconexión de E/S**. Esto hace que la interfaz externa tenga unas características propias que difícilmente se pueden generalizar.

Cuando un módulo de E/S gestiona más de un periférico, hay dos configuraciones básicas, la conexión punto a punto y la multipunto, aunque las configuraciones que encontramos en máquinas reales son muy variadas. En la conexión punto a punto el módulo de E/S gestiona la comunicación con cada periférico individualmente; no es un bus de E/S, pero sí que tiene unas especificaciones normalizadas de la conexión, de manera parecida a las de un bus normalizado de E/S. En la conexión multipunto el módulo de E/S gestiona la comunicación con los periféricos mediante un bus normalizado de E/S y hay que añadir la lógica para acceder al bus.

Conexión punto a punto entre el módulo de E/S y el periférico



Conexión multipunto entre el módulo de E/S y el periférico



La comunicación entre los módulos de E/S y el computador es siempre la misma para todos los módulos. Esta comunicación se establece mediante el bus del sistema, de modo que el procesador ve el módulo de E/S como un espacio de memoria, pero estas direcciones, físicamente, corresponden (están mapeadas) a cada uno de los registros que tiene el módulo de E/S del computador y se denominan habitualmente **puertos de E/S**. De esta manera conseguimos que la comunicación entre el computador y el módulo de E/S se lleve a cabo mediante instrucciones de transferencia para leer y escribir en sus registros, de una manera muy parecida a como hacemos para acceder a la memoria.

Estos registros se pueden agrupar según el tipo de señales o el tipo de información que necesitamos para hacer una gestión correcta del periférico:

- Registros de control.
- Registros de estado.
- Registros de datos.

Para gestionar la comunicación entre el procesador y el módulo de E/S son necesarios diferentes tipos de señales.

Las **señales de control** las utilizamos generalmente para dar órdenes al módulo de E/S, como empezar o parar una transferencia, seleccionar modos de operación del periférico o indicar acciones concretas que debe hacer el periférico, como comprobar si está disponible. Estas señales se pueden recibir directamente de las líneas de control del bus del sistema o de las líneas de datos del bus del sistema y se almacenan en el *registro de control*.

Las **señales de estado** nos dan información del estado del módulo de E/S, como saber si el módulo está disponible o está ocupado, si hay un dato preparado, si se ha acabado una operación, si el periférico está puesto en marcha o parado, qué operación está haciendo, o si se ha producido algún error y qué tipo de error. Estas señales se actualizan generalmente mediante la lógica del módulo de E/S y se almacenan en el *registro de estado*.

Los **datos** son la información que queremos intercambiar entre el módulo de E/S y el procesador mediante las líneas de datos del bus del sistema y se almacenan en el *registro de datos*.

Las **direcciones** las pone el procesador en el bus de direcciones y el módulo de E/S debe ser capaz de reconocer estas direcciones (direcciones de los puertos de E/S) correspondientes a los registros de este módulo. Para saber si la dirección corresponde a uno de los registros del módulo utilizamos un descodificador. Este descodificador puede formar parte del módulo de E/S o de la misma lógica del bus del sistema.

Hay que tener presente que un computador puede tener definidos diferentes tipos de conexiones normalizadas entre el módulo de E/S y el resto del computador. Tanto el módulo de E/S como el computador se deben adaptar a estos tipos de conexión, de modo que tenemos módulos de E/S adaptados a las diferentes normas, y eso tiene implicaciones con respecto al hardware y a la manera de gestionar las operaciones de E/S, como veremos más adelante cuando analicemos las técnicas básicas de E/S.

1.1.3. Sistemas de interconexión externos

En un computador distinguimos dos tipos básicos de sistemas de interconexión: los internos del computador, que nos permiten conectar el procesador, la memoria y el sistema de E/S y que denominamos **bus del sistema**, y los externos al computador, que nos permiten conectar el sistema de E/S con los diferentes periféricos y que denominamos **sistemas de interconexión de E/S** o **buses de E/S**.

Desde el punto de vista del sistema de E/S, el **bus del sistema** nos permite la comunicación entre los módulos de E/S y el resto del computador. Este bus tiene una estructura jerárquica formada por diferentes tipos de buses para aislar los elementos más rápidos de los más lentos y, de esta manera, mejorar las prestaciones del sistema.

Los **sistemas de interconexión de E/S** o **buses de E/S** nos permiten la comunicación de los módulos de E/S con los periféricos o dispositivos con suficiente autonomía para gestionar una operación de E/S y los módulos de E/S. Las características de estos sistemas se adaptan al tipo de dispositivos que hemos de conectar.

Físicamente, un sistema de interconexión está formado por un conjunto de hilos conductores o líneas que interconectan diferentes dispositivos. Por estas líneas circulan señales eléctricas que los dispositivos que tenemos conectados pueden interpretar como señales binarias. Hay tres tipos de señales básicas: señales de datos, de direcciones y de control.

Las siguientes son las características principales de los sistemas de interconexión externos:

- **Ancho de banda**: la cantidad máxima de información que podemos transmitir por unidad de tiempo. Se expresa en bits o bytes por segundo.

- **Serie/paralelo**: en una interconexión paralela hay varias líneas que conectan el módulo de E/S y el periférico y pueden transmitir varios bits simultáneamente mediante las líneas de datos. En una interconexión serie solo hay una línea para transmitir los datos y los bits se han de transmitir uno a uno. Tradicionalmente las interconexiones de tipo serie eran para dispositivos lentos y las de tipo paralelo, para dispositivos más rápidos, pero con las nuevas generaciones de sistemas de interconexión serie de alta velocidad las paralelas cada vez son menos utilizadas.

- **Punto a punto/multipunto**: una interconexión punto a punto tiene un enlace dedicado entre el módulo de E/S y el periférico. En una interconexión multipunto, que habitualmente se denomina *bus de E/S* y que dispone

de un enlace compartido entre diferentes periféricos y el módulo de E/S, el hecho de tener múltiples dispositivos conectados a un mismo conjunto de líneas hace necesario establecer mecanismos para controlar el acceso.

Otras características típicas de los buses de E/S son:

- **Modo de operación síncrono/asíncrono/semisíncrono**: si el control de los accesos al bus es controlado o no por un reloj.

- **Multiplexación de datos y direcciones**: si las líneas del bus están dedicadas a datos y direcciones o si se comparten las mismas líneas para datos y para direcciones.

- **Arbitraje centralizado y distribuido**: es centralizado cuando un único árbitro o controlador determina quién tiene que acceder al bus en cada momento y es distribuido cuando los dispositivos conectados al bus disponen de capacidad de controlar el acceso al bus.

- **Tipos de operaciones de lectura/escritura**: diferentes maneras de hacer las operaciones de lectura y escritura, como la transferencia de bloques o la combinación de operaciones de lectura y escritura.

- **Esquema de direccionamiento**: hay dos tipos básicos: el *direccionamiento lógico*, que es cuando el espacio de direccionamiento de memoria es común a todos los dispositivos y cada uno dispone de un rango de direcciones único y los dispositivos para descodificar la dirección para saber si esta dirección está dentro de su rango; el *direccionamiento geográfico*, que es cuando cada dispositivo tiene una dirección propia y se separa la identificación del módulo de la selección de la dirección dentro del módulo.

Normalización de un bus

La normalización de un bus consiste en dar una descripción detallada y precisa de las características que tiene a diferentes niveles. Los niveles principales son el mecánico (tamaños y conectores), el eléctrico (tipos de señales eléctricas que circulan por las líneas), el lógico (descripción de todas las señales: direcciones, datos y control) y el cronograma (cuál es la secuencia de señales para hacer la transferencia de un dato o de un bloque).

La normalización de un bus facilita la divulgación y también el diseño de los dispositivos que hemos de conectar. Generalmente, la mayoría de las normalizaciones son definidas por organismos internacionales. El más conocido en el ámbito de la electricidad y la electrónica es el IEEE.

1.1.4. Mapa de memoria e instrucciones de E/S

Tal como hemos explicado, el procesador ve el banco de registros del módulo de E/S como un espacio de memoria dirigible, de manera que cada registro del módulo de E/S tiene asociada (mapeada) una dirección única. Veamos ahora cómo hemos de acceder a estas direcciones, que denominamos **puertos de E/S**.

Para identificar los registros del módulo de E/S hay dos posibilidades:

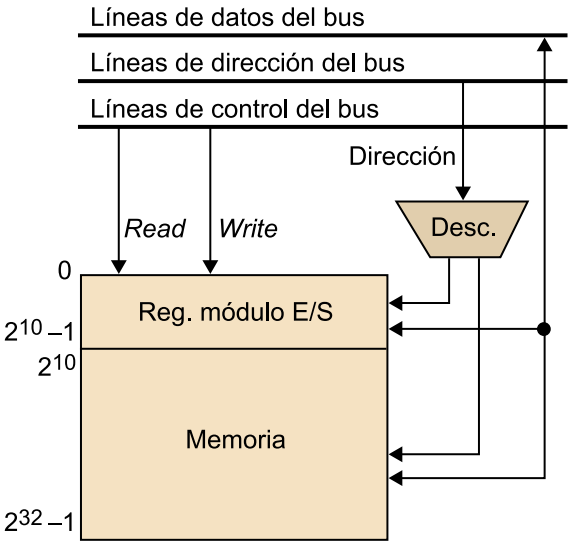
1) Mapa común de memoria y E/S. No hay distinción entre direcciones de memoria y registros de E/S. Para acceder a los registros se utilizan descodificadores que se activan a partir de las líneas del bus de direcciones y se utilizan las mismas señales de control (READ/WRITE) que se emplean para seleccionar la

memoria. Podemos acceder a los puertos de E/S con las mismas instrucciones de transferencia que utilizamos para acceder a memoria (MOV y las variantes que tiene).

Este sistema tiene la ventaja de que nos permite aprovechar el amplio conjunto de instrucciones del que dispone el procesador para acceder a memoria y podemos hacer programas más eficientes. La principal desventaja es que hemos de dedicar una parte del valioso espacio de memoria a las direcciones de E/S y hay que ser cuidadosos con la cantidad de espacio asignado, ya que este espacio va en detrimento del espacio de memoria disponible, pero cada vez este problema es menos importante a causa del incremento del espacio de memoria dirigible.

Ejemplo de mapa común de memoria

En la siguiente figura tenéis un ejemplo de conexión de una memoria de 2^{32} direcciones y 2^{10} puertos de E/S utilizando un mapa común y un mapa independiente de memoria y E/S.

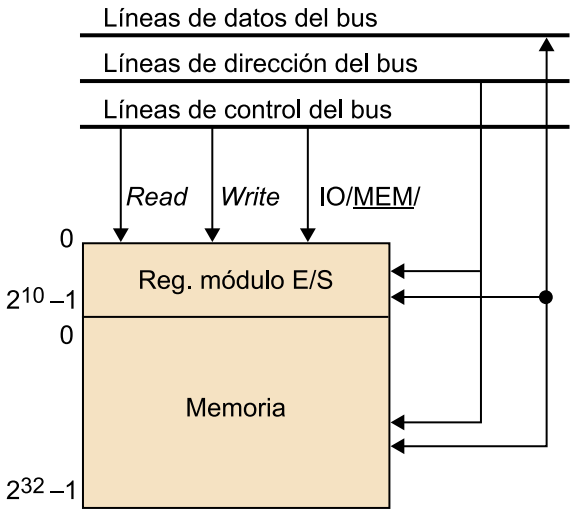


2) Mapa independiente de E/S. Hay distinción entre direcciones de memoria y registros de E/S. Las líneas de direcciones se suelen compartir, pero hay que añadir algunas líneas de control para distinguir si un acceso es a memoria o a un puerto de E/S. También son necesarias instrucciones específicas de E/S. Las instrucciones utilizadas habitualmente son IN (para leer del puerto de E/S) y OUT (para escribir en el puerto de E/S).

Este sistema tiene la ventaja de que la memoria dispone de todo el rango de direcciones y la clara desventaja de que dispone de un reducido número de instrucciones específicas de E/S que solo disponen de los modos de direccionamiento más básicos para acceder a los puertos de E/S.

Ejemplo de mapa independiente

En la siguiente figura tenéis un ejemplo de conexión de una memoria de 2^{32} direcciones y 2^{10} puertos de E/S utilizando un mapa independiente de memoria y E/S.



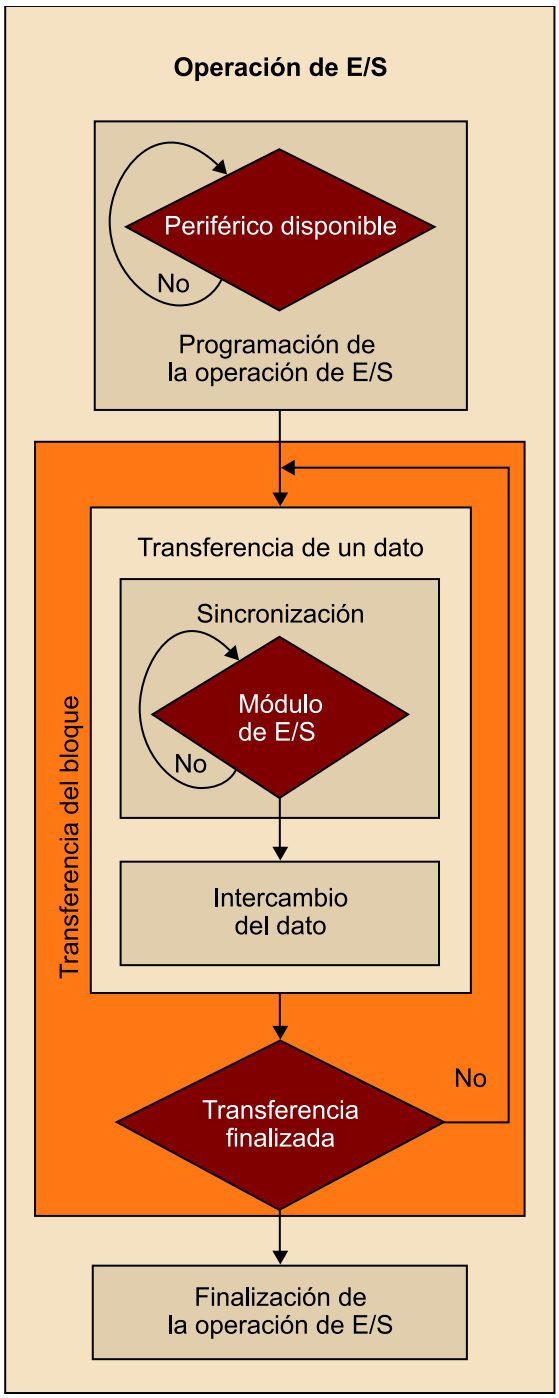
Una cuestión interesante que hay que considerar es que si utilizamos instrucciones específicas de E/S, el procesador puede saber en la fase de descodificación de la instrucción que haremos una operación de E/S. Esto hace más fácil la gestión de las señales de control para llevar a cabo esta operación y también hace más fácil establecer un mecanismo de protección para controlar que estas instrucciones solo se puedan ejecutar en modo supervisor. Si no tenemos instrucciones específicas de E/S, hasta que no se hace el acceso al puerto de E/S (se descodifica la dirección), no podemos saber si aquel acceso se debe permitir o no y eso complica la gestión.

1.2. Operación de E/S

Para garantizar que una transferencia de datos se realiza con éxito, es necesario definir una serie de pasos y establecer una serie de mecanismos que nos permitan controlar en todo momento la operación de E/S.

En el proceso global de una operación de E/S distinguimos los pasos siguientes:

1) Programación de la operación de E/S.
2) Transferencia de un bloque. Transferencia de un dato (para cada dato del bloque):
a) Sincronización.
b) Intercambio del dato.
3) Finalización de la operación de E/S.



1.2.1. Programación de la operación de E/S

La programación de la operación de E/S es el proceso para indicar al módulo de E/S cómo se debe llevar a cabo la transferencia.

Esta programación consiste en ejecutar un pequeño conjunto de instrucciones que verifican si el periférico está disponible para iniciar una transferencia de datos, actualizan los registros del módulo de E/S, principalmente los registros

de control para dar instrucciones al periférico, y también nos puede servir para inicializar las variables o registros que necesite el procesador para llevar a cabo la transferencia de datos.

1.2.2. Transferencia de datos

La transferencia de datos es la fase donde se hace realmente la transferencia de información entre el procesador y el módulo de E/S y es la fase que veremos con más detalle analizando diferentes técnicas.

De manera genérica, en la transferencia de cada dato dentro de un bloque distinguimos dos fases principales: la sincronización y el intercambio. Estas dos fases se repiten para cada dato del bloque.

La **sincronización** es donde se establece un mecanismo para conseguir que el dispositivo más rápido espere que el dispositivo más lento esté preparado para llevar a cabo el *intercambio del dato*.

Este mecanismo nos garantiza que no se dejen datos sin procesar, pero la consecuencia es que la transferencia de datos se realiza a la velocidad del dispositivo más lento.

Durante la fase de sincronización, el procesador (o el elemento que controle la transferencia) debe ser capaz de detectar cuándo está disponible el periférico para hacer el intercambio de un dato. El módulo debe informar de que el periférico está preparado.

Ejemplo del proceso de sincronización

Supongamos que el procesador envía diez datos (D0, D1, D2, D3, D4, D5, D6, D7, D8, D9) a un periférico. El procesador puede enviar un dato cada 2 ms y el periférico tarda 5 ms en procesar cada dato.

	Tiempo (ms)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sin sincronización	Proc. envía	D0		D1		D2		D3		D4		D5		D6		D7		D8		D9		FI
	Perif. lee	↓ D0					↓ D2					↓ D5					↓ D7					↓ FI
Con sincronización	Proc. envía	D0					D1					D2					D3					...
	Perif. lee	↓ D0					↓ D1					↓ D2					↓ D3					↓ ...

Cuando no hay sincronización, observamos que hay datos que se pierden (D1, D3, D4, D6, D8, D9). Cuando el periférico quiere leer el dato siguiente, el procesador ya ha enviado nuevos y los anteriores se han perdido porque se han sobreescrito en el registro de datos del módulo de E/S. Con sincronización, el procesador se espera 3 ms (zona sombreada) y, cuando el periférico está preparado, el procesador envía el nuevo dato. De esta manera, no se pierde ningún dato, pero la transferencia se hace al ritmo que marca el periférico, que es el más lento.

Memorias intermedias

Este problema se puede reducir utilizando una pequeña memoria intermedia (generalmente denominada *buffer*) para almacenar temporalmente la entrada o salida de datos. Pero si la diferencia de velocidad es grande o los bloques de datos son grandes, la única manera de garantizar que no se pierdan datos es tener un mecanismo de sincronización.

El **intercambio del dato** es la fase en la que se envía realmente el dato. Sabiendo que tanto el emisor como el receptor están preparados (se ha hecho la *sincronización*), se verifica que el dato se ha recibido correctamente, se deja todo preparado y se indica que ya se puede iniciar una nueva transferencia.

En una operación de entrada (lectura), el periférico envía el dato al módulo de E/S, que hace una verificación para detectar posibles errores. Generalmente, esta verificación es una validación muy simple, como verificar el bit de paridad, y deja el dato disponible para el procesador en el registro de datos del módulo de E/S para que el procesador lo pueda leer.

En una operación de salida (escritura), el procesador escribe el dato en el registro de datos del módulo de E/S y después el módulo de E/S lo envía al periférico.

Veamos cómo sería una operación de E/S en la que el procesador lee un dato de un periférico:

Nota

Hay que tener presente que el proceso que aquí describimos puede variar ligeramente dependiendo del tipo de periférico con el que hacemos la transferencia o de la técnica de E/S utilizada.

1) El procesador comprueba el estado del periférico, lee el registro de estado del módulo de E/S y mira los bits que indican si el periférico está disponible. Puede ser necesario dar una orden accediendo al registro de control para que el módulo de E/S consulte el estado del periférico y actualice el registro de estado.

2) Si el periférico está disponible, el procesador solicita el dato enviando una orden al módulo de E/S. El procesador debe acceder al puerto que corresponde al registro de control del módulo de E/S para escribir en él y así indicar la operación que queremos que haga. Hemos programado la transferencia.

Acceso a los puertos de E/S

Recordad que para acceder a un puerto de E/S (registro de un módulo de E/S), utilizamos instrucciones de transferencia si el procesador tiene un mapa común de memoria y E/S e instrucciones específicas de E/S si el computador tiene un mapa independiente de memoria y E/S.

En estas instrucciones debemos especificar la dirección de uno de los registros del módulo de E/S donde tenemos conectado el periférico con el que queremos llevar a cabo la operación de E/S.

El módulo, por su parte, ha de ser capaz de reconocer esta dirección que el procesador ha puesto en las líneas de direcciones del bus y leer el dato de las líneas de datos del bus para actualizar el registro correspondiente si se trata de una operación de escritura o poner la información de este registro en las líneas de datos del bus si estamos haciendo una operación de lectura.

3) La lógica del módulo se encarga de identificar la operación que tiene que hacer e inicia la transferencia con el periférico, de manera que el periférico deja de estar disponible y cambia el estado a ocupado hasta que la transferencia se finalice. Empieza la transferencia del dato con la sincronización.

4) El módulo obtiene el dato del periférico, lo guarda en el registro de datos para que el procesador lo pueda leer y actualiza el registro de estado o avisa al procesador para indicar que el dato está disponible, y consigue de esta manera la sincronización con el procesador. Si ha habido errores al obtener el dato del periférico, también lo indica en el registro de estado. Mientras el módulo de E/S obtiene el dato del periférico y valida la información, el procesador se ha de esperar. Se acaba la sincronización y empieza el intercambio del dato.

5) Una vez el dato está preparado en el módulo de E/S y el procesador está enterado de ello, el procesador obtiene el dato y se hace el intercambio del dato. El procesador debe acceder al puerto correspondiente al registro de estado si quiere validar que el dato recibido es correcto y al puerto correspondiente al registro de datos para leer el dato. Se acaba el intercambio del dato.

6) Como solo se tiene que transferir un dato, se indica que el estado del periférico es de disponible para hacer una nueva operación de E/S. Se acaba la operación de E/S.

1.2.3. Finalización de la operación de E/S

La finalización de la operación de E/S es un proceso parecido a la programación de la operación de E/S. También consiste en ejecutar un pequeño conjunto de instrucciones que actualizan los registros del módulo de E/S, pero ahora para indicar que se ha finalizado la transferencia y que el módulo de E/S queda disponible para atender a otra operación de E/S. También se puede consultar el estado del módulo de E/S para verificar que la transferencia se ha hecho correctamente.

Hay que remarcar que tanto la programación como la finalización de la operación de E/S son tareas que siempre efectúa el procesador, ya que es quien solicita la operación de E/S de información, y tienen realmente sentido cuando queremos enviar un bloque de datos y el módulo de E/S tiene autonomía para gestionar la transferencia. En cambio, no tienen tanto sentido para enviar datos unitarios. La complejidad de este proceso depende de la complejidad del periférico y de la técnica de E/S que utilizamos.

1.3. Gestión de múltiples dispositivos

Todos los computadores deben gestionar más de un periférico y estos pueden trabajar al mismo tiempo; por ejemplo, estamos imprimiendo un documento que tenemos guardado en el disco mientras escribimos un texto con el teclado que se muestra por pantalla. Por lo tanto, hemos de prever que nuestro

sistema de E/S pueda gestionar transferencias de E/S con dos o más periféricos simultáneamente. Eso quiere decir que de manera simultánea dos o más módulos de E/S deben estar preparados para hacer la transferencia de datos con el procesador, pero la transferencia no la podemos hacer al mismo tiempo. Por este motivo, hemos de disponer de un sistema que, primero, nos permita determinar cuáles son los módulos a los que tenemos que atender (**identificar los módulos de E/S** que están preparados para la operación de E/S) y, segundo, nos permita decidir a quién atendemos primero, teniendo en cuenta que si ya atendemos a otro periférico o hacemos otra tarea más prioritaria no la podemos interrumpir (**establecer una política de prioridades**).

Tanto la identificación del módulo de E/S que está preparado para transferir un dato como la manera de establecer una política de prioridades dependen de la técnica de E/S que utilicemos para gestionar las operaciones de E/S y lo hemos de analizar con detalle en cada caso.

Las políticas de prioridades básicas son:

- **Prioridad fija**: cuando el procesador está preparado para atender una petición, siempre empieza la consulta por el periférico que tiene más prioridad. El periférico con más prioridad siempre es atendido el primero. Si hay muchas peticiones de periféricos prioritarios, los menos prioritarios puede ser que hayan de esperar mucho tiempo para ser atendidos.

- **Prioridad rotativa**: cuando el procesador está preparado para atender una petición, consulta al periférico que hay a continuación según un número de orden preestablecido. A los periféricos se les asigna un orden pero todos tienen la misma prioridad.

Normalmente las políticas de prioridad se definen según las necesidades de cada dispositivo y las restricciones específicas que muchas veces impone el sistema de E/S utilizado en un computador. Y se pueden utilizar diferentes políticas de prioridades en un mismo sistema.

1.4. Técnicas de E/S

Hemos visto de manera genérica cuáles son los pasos para hacer una operación de E/S. El primer paso y el último, la programación y la finalización de la operación de E/S, siempre son responsabilidad del procesador y la complejidad que tienen depende en gran medida de las características del periférico y del sistema de interconexión de E/S que utilicemos, y no tanto, aunque también, de la técnica de E/S que utilicemos.

Por este motivo, en los apartados siguientes nos centraremos en la fase de **transferencia de un dato** (sincronización e intercambio del dato). Analizaremos qué dispositivos nos permiten liberar o descargar al procesador de las diferentes tareas que se tienen que hacer en este proceso y distinguiremos las técnicas básicas de E/S siguientes:

- E/S programada.
- E/S por interrupciones.
- E/S por DMA.
- Canales de E/S.

En la tabla que hay a continuación vemos cómo quedan repartidas las responsabilidades en una transferencia de E/S según la técnica de E/S que utilicemos.

	Programación	**Sincronización**	**Intercambio**	**Finalización**
E/S programada	Procesador	Procesador	Procesador	Procesador
E/S por interrupciones	Procesador	Módulo de E/S	Procesador	Procesador
E/S por DMA	Procesador	DMA	DMA bloquea el procesador	Procesador
Canales de E/S	Procesador/Canal de E/S	Canal de E/S	Canal de E/S bloquea el procesador	Procesador

2. E/S programada

Para hacer la operación de E/S entre el procesador y el módulo de E/S, el procesador ejecuta un programa que controla toda la operación de E/S (programación, transferencia de datos y finalización).

A continuación, analizamos con más detalle la transferencia de un dato:

1) Sincronización. Durante la sincronización, el procesador, como responsable de la transferencia, ejecuta un programa que mira constantemente el estado del periférico consultando el registro de estado del módulo de E/S. Este programa tiene un bucle que se ejecuta continuamente hasta que detecta el cambio de estado e indica que el periférico está preparado. Este método de sincronización se denomina **sincronización por encuesta** o **espera activa**.

Mientras se lleva a cabo la sincronización, el procesador está dedicado al cien por cien a esta tarea y, por lo tanto, no puede atender a otros procesos o aplicaciones. Si esta espera es muy larga, puede degradar el nivel de prestaciones de todo el sistema. Por lo tanto, es recomendable que las transferencias hechas utilizando esta técnica sean cortas y rápidas.

2) Intercambio del dato. Durante el intercambio del dato, si es una operación de lectura (entrada), el procesador lee el registro de datos del módulo de E/S para recoger el dato enviado por el periférico, y lo guarda en memoria; si es una operación de escritura (salida), el procesador toma de la memoria el dato que queremos enviar al periférico y lo escribe en el registro de datos del módulo de E/S.

Programa para atender a un periférico utilizando E/S programada

Queremos enviar un dato (escritura) a un periférico controlado por un módulo de E/S que tiene mapeados los registros de control, de estado y de datos en las direcciones 0120h, 0124h, 0128h, respectivamente.

SINCRO:	**IN**	R0,[0000 0124h]	0000 0124h: dirección del registro de estado del módulo de E/S. Leemos el estado y lo guardamos en R0.
	AND	R0, 00000004h	00000004h es una máscara para mirar el bit 3 del registro R0; en este caso es el bit que indica si el periférico está disponible para hacer la transferencia de datos.
	JE	SINCRO	Si el bit 3 está a cero, volvemos a SINCRO: y continuamos esperando hasta que el periférico esté preparado para recibir el dato y el módulo active este bit; si vale 1, continuamos y vamos a INTERCAMBIO: para hacer el intercambio del dato.
INTERCAMBIO:	**MOV**	R1, [Dato]	Tomamos el dato que queremos enviar al periférico y lo ponemos en el registro R1.
	OUT	[0000 0128h], R1	Copiamos el dato que tenemos en el registro R1 en el registro de datos del módulo de E/S (0000 0128h) para que este lo envíe al periférico.
	RET		Devolvemos el control al programa que ha llamado a esta rutina para hacer la operación de E/S.

Tanto este ejemplo como la descripción del proceso de sincronización y transferencia de datos se ha hecho siguiendo un modelo simplificado de operación de E/S. Hay que tener presente que para programar una rutina que lleve a cabo una operación de E/S con un periférico concreto hay que conocer las direcciones de los puertos de E/S asociadas a los registros del módulo de E/S que gestiona este periférico y el significado o la utilidad de cada uno de los bits. Dada la gran variedad de periféricos y su creciente complejidad, interpretar toda esta información es una tarea bastante complicada y generalmente son los propios fabricantes quienes desarrollan estas rutinas.

2.1. Gestión de múltiples dispositivos

No es habitual gestionar operaciones de E/S con múltiples dispositivos utilizando E/S programada. En caso de ser necesario, hay que tener presente que durante la sincronización se debe realizar la encuesta de todos los periféricos implicados estableciendo una política de prioridades implementada en forma de programa durante la sincronización. Este sistema de identificar qué periférico necesita ser atendido se denomina **encuesta** o ***polling***.

En el momento en el que se detecta que uno de los periféricos está preparado llamamos a una subrutina para hacer el intercambio del dato y volvemos al bucle de sincronización para seguir haciendo la encuesta según la política de prioridades implementada hasta que se hayan acabado todas las operaciones de E/S que están programadas.

3. E/S con interrupciones

En este apartado veremos el E/S por interrupciones. Esta técnica de E/S pretende evitar que el procesador tenga que estar parado o haciendo trabajo improductivo mientras espera a que el periférico esté preparado para hacer una nueva operación de E/S y pueda aprovechar este tiempo para ejecutar otros programas.

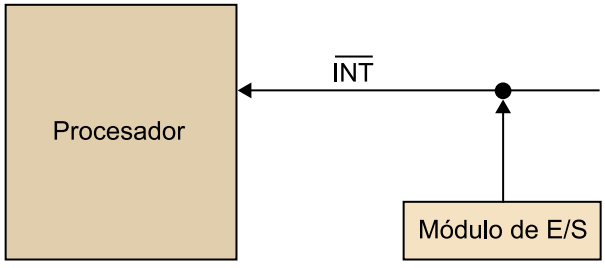
Utilizando la técnica de E/S por interrupciones se descarga al módulo de E/S de la responsabilidad de llevar a cabo la sincronización entre el periférico y el procesador.

Para utilizar esta técnica de E/S en un computador, es necesario considerar tanto aspectos del software como del hardware.

Como parte del hardware, es necesario que el computador disponga de una línea especial que tiene que formar parte del conjunto de líneas de control del bus del sistema y que denominamos **línea de petición de interrupción (INT)**. El módulo de E/S avisa al procesador mediante esta línea e indica que está preparado para hacer la transferencia. La señal INT la activa el módulo de E/S y la recibe el procesador. Es una señal activa a la baja. El procesador debe tener un punto de conexión de entrada por donde llegarán las interrupciones y el módulo de E/S debe tener un punto de conexión de salida por donde generará las interrupciones.

Señal activa a la baja

La señal INT es activa a la baja. Se considera activa si tenemos un 0 en la línea y no se considera activa si tenemos un 1.



Para hacer una operación de E/S utilizando esta técnica se siguen los mismos pasos que en la E/S programada: se programa la operación de E/S, se realiza la transferencia de datos y se finaliza la operación de E/S. La diferencia principal la tenemos durante la transferencia de datos, en la que en la fase de sincronización debemos hacer la gestión de las interrupciones y eso también afecta en cierta medida al intercambio de datos, como veremos más adelante.

Durante la fase de sincronización, una vez hecha la programación de la operación de E/S, el procesador ejecuta otro programa (según la política de gestión de procesos del sistema operativo) de manera que el procesador estará ocupado haciendo trabajo productivo hasta que el módulo de E/S esté preparado y active la señal de petición de interrupción (INT).

De entrada, el procesador no sabe en qué momento se producirá esta petición; por lo tanto, ha de comprobar periódicamente si el módulo de E/S pide la atención del procesador, sin que ello afecte a la dedicación que tiene. Esta comprobación el procesador la hace dentro del ciclo de ejecución de cada instrucción. Es una operación muy rápida que incluso se puede encabalgar con el comienzo de la lectura de la instrucción siguiente para que no afecte al rendimiento del procesador. Los procesadores que han de gestionar interrupciones deben tener en el ciclo de ejecución de instrucción una **fase de comprobación de interrupciones**.

Ciclo de ejecución de instrucción

Recordad que las cuatro fases principales para ejecutar una instrucción son:

1) Lectura de la instrucción.

2) Lectura de los operandos fuente.

3) Ejecución de la instrucción y almacenamiento del operando destino.

4) Comprobación de interrupciones.

En el momento en el que el procesador reconoce que ha llegado una petición de interrupción, empieza un **ciclo de reconocimiento de interrupción** para detener la ejecución del programa actual y transferir el control a la **rutina de servicio de la interrupción (RSI)**, rutina que accede al módulo de E/S correspondiente para llevar a cabo la transferencia de datos y, una vez se acabe la ejecución de la RSI, continuar la ejecución del programa que habíamos detenido haciendo el **retorno de interrupción**.

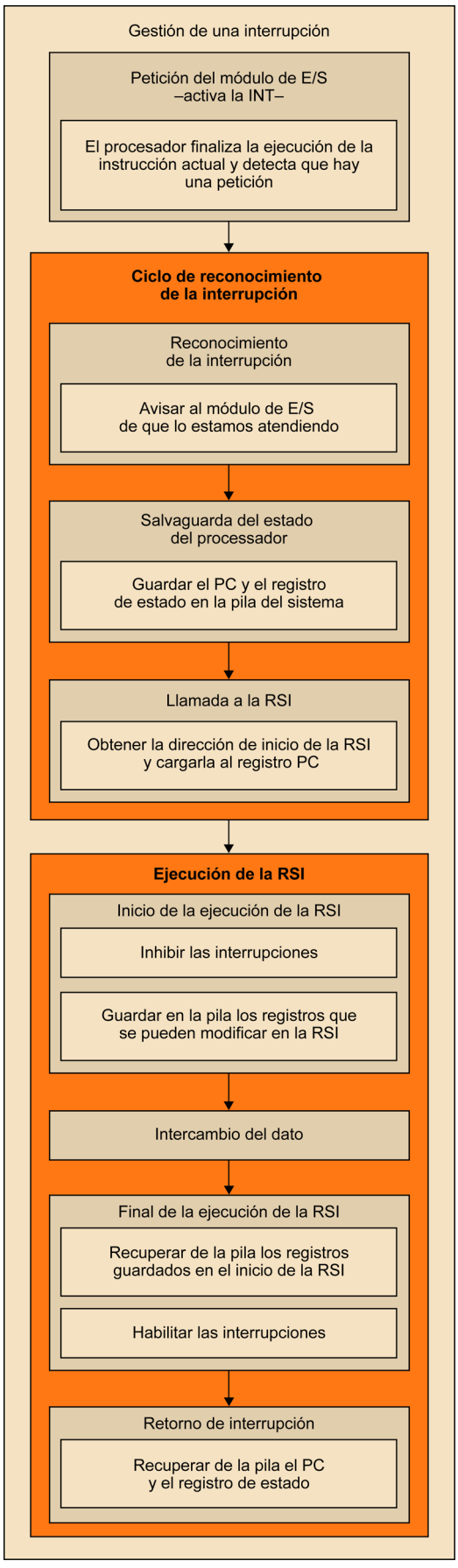
En este apartado nos centraremos en el análisis de interrupciones provocadas por elementos externos al procesador (interrupciones del hardware), pero también hay un tipo de interrupciones producidas por acontecimientos internos al procesador y denominadas **interrupciones de software**, **excepciones** o ***traps***.

Las interrupciones del software se gestionan de una manera muy parecida a las interrupciones externas al procesador, pero sin ser necesario dedicar una línea del bus del sistema para gestionarlas. Para tratarlas, no hay que esperar a la fase de comprobación de interrupción, sino que se pueden tratar en el mismo momento en el que se produce.

Las más habituales son estas: operaciones que la ALU no puede hacer, como dividir por 0, no se reconoce el código de operación de la instrucción que queremos ejecutar, acceso a recursos restringidos o áreas privilegiadas de memoria y también peticiones del propio programador con instrucciones específicas del juego de instrucciones para hacer peticiones al sistema operativo.

3.1. Gestión de una interrupción con un único módulo de E/S

Veamos cuáles son los pasos básicos para la gestión de una interrupción en un sistema con una única línea de interrupción y un único módulo de E/S. Más adelante analizaremos diferentes mejoras de este sistema para gestionar múltiples módulos de E/S y múltiples líneas de interrupción y cómo afecta eso a este proceso. Primero, sin embargo, hay que entender bien el caso más simple.


Gestión de una interrupción
Petición del módulo de E/S
–activa la INT–
El procesador finaliza la ejecución de la instrucción actual y detecta que hay una petición
Ciclo de reconocimiento de la interrupción
Reconocimiento de la interrupción
Avisar al módulo de E/S de que lo estamos atendiendo
Salvaguarda del estado del processador
Guardar el PC y el registro de estado en la pila del sistema
Llamada a la RSI
Obtener la dirección de inicio de la RSI y cargarla al registro PC
Ejecución de la RSI
Inicio de la ejecución de la RSI
Inhibir las interrupciones
Guardar en la pila los registros que se pueden modificar en la RSI
Intercambio del dato
Final de la ejecución de la RSI
Recuperar de la pila los registros guardados en el inicio de la RSI
Habilitar las interrupciones
Retorno de interrupción
Recuperar de la pila el PC y el registro de estado

Una vez el procesador ha solicitado una operación de E/S, ha programado la transferencia y ya ejecuta otro programa mientras espera a que el módulo de E/S esté preparado. En el momento en el que el módulo de E/S pide la atención del procesador (el módulo activa la INT), se produce una secuencia de acontecimientos que el procesador ha de gestionar para atender a esta petición del módulo de E/S, garantizando que después podrá devolver el control al programa cuya ejecución detenemos para atender la petición del módulo de E/S.

Los pasos son los siguientes:

1) Petición del módulo de E/S. El módulo de E/S está preparado para hacer una transferencia y activa la INT. Entonces, cuando el procesador acaba la ejecución de la instrucción actual, en la última fase del ciclo de ejecución de la instrucción, la fase de comprobación de interrupción, detecta que se ha hecho una petición.

2) Ciclo de reconocimiento de la interrupción. Esta es seguramente la parte más compleja de la gestión de interrupciones porque se producen muchos acontecimientos en poco tiempo y, como veremos más adelante, algunos de estos acontecimientos se pueden producir encabalgadamente en el tiempo y hay que estar atento a quién genera la acción, quién la recibe y qué respuesta da.

a) Reconocimiento de la interrupción. Si las interrupciones están habilitadas, el procesador acepta la petición (cuando hay más de un dispositivo se debe determinar si es suficientemente prioritario para ser atendido, situación que analizaremos más adelante, ya que tiene otras implicaciones en el proceso). Entonces, es necesario que el procesador avise al módulo de E/S para que sepa que lo está atendiendo y para que desactive la INT, y también es necesario que el procesador inhiba las interrupciones para evitar nuevas peticiones.

Si las interrupciones están inhibidas, el procesador no atiende la petición y continúa ejecutando el programa. El periférico se tiene que esperar a ser atendido y ha de dejar la INT activa hasta que el procesador las vuelva a habilitar.

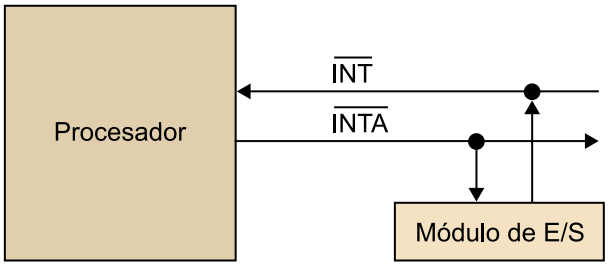
Para **inhibir las interrupciones**, hay básicamente dos maneras de hacerlo: con un hardware específico que las inhiba automáticamente hasta que se acabe la atención de la interrupción o dejar que sea responsabilidad de la propia RSI, que tiene que ejecutar una primera instrucción para inhibir las interrupciones y al acabar ejecutar otra instrucción para volver a habilitarlas.

Inhibir las interrupciones consiste en modificar uno o más bits del registro de la palabra de estado del procesador (algunos procesadores pueden tener registros específicos para la gestión de interrupciones). Estos bits son los que se

consultan en la fase de comprobación de interrupciones durante la ejecución de una instrucción para saber si las interrupciones están habilitadas y se tiene que aceptar o no una petición.

Si el procesador no inhibe las interrupciones, al ejecutar la primera instrucción de la RSI, en la fase de comprobación de interrupción volverá a aceptar la petición y podría entrar en un bucle infinito ejecutando esta primera instrucción indefinidamente, ya que volverá a empezar un ciclo de reconocimiento de la misma interrupción.

De manera análoga a como se hace en la inhibición de las interrupciones, se puede avisar al módulo de E/S de dos maneras: la más habitual es utilizando un hardware específico que incluye una **línea de reconocimiento de interrupción** (INTA o INTACK, del inglés *interrupt acknowlege*) que activa el procesador, igual que la INT es una señal activa a la baja, o dejar que sea responsabilidad de la propia RSI, que ha de ejecutar una o varias instrucciones para acceder a los registros del módulo de E/S y de esta manera indicarle que ya lo atiende.



b) Salvaguarda del estado del procesador. Ahora el procesador ha reconocido que hay una petición de un módulo de E/S, la ha aceptado y tiene que ejecutar la rutina de servicio de interrupción (RSI) para atenderlo, pero recordemos que el procesador está ejecutando otro programa y hay que reanudar la ejecución una vez ha acabado la ejecución de la RSI. Por este motivo se tiene que hacer la salvaguarda del estado del procesador, que consiste en almacenar la información necesaria para garantizar que podremos reanudar la ejecución del programa en las mismas condiciones. Esta información se almacena en la memoria del computador, generalmente en la pila de sistema.

El estado del procesador lo determina el valor de todos sus registros, pero para reanudar la ejecución del programa solo hay que guardar los registros que pueda modificar la RSI. Una RSI puede modificar cualquier registro del procesador, pero realmente suelen ser rutinas relativamente simples que solo modifican algunos de estos registros. Si solo se guardan los registros que se modifican dentro de la RSI, la salvaguarda y la restauración posterior serán mucho más eficientes en tiempo y espacio, ya que reducimos los accesos a la memoria y el espacio necesario para guardar al estado.

La manera más habitual de hacer la salvaguarda es almacenar de manera automática la información mínima necesaria para llamar a la RSI y almacenar el resto de la información dentro de la misma rutina, y es responsabilidad del programador de la RSI determinar qué registros hay que guardar.

La información mínima que se debe guardar es el **registro contador de programa** (PC) y el **registro de estado**. El registro PC tiene la dirección de la instrucción siguiente que queremos ejecutar. El registro de estado contiene información importante de control y de estado del procesador que puede ser modificada por cualquier instrucción.

Nota

Recordad que el valor del PC se actualiza en la fase de lectura de la instrucción del ciclo de ejecución.

Con respecto al resto de la información, como el procesador no puede saber qué registros modificará la RSI, deja que sea responsabilidad de la propia RSI.

c) Llamada a la RSI. Para empezar a ejecutar la rutina de servicio de interrupción (RSI), primero de todo, hay que saber la dirección de memoria donde tenemos que iniciar la ejecución de la RSI. En caso de gestionar un único dispositivo, esta dirección puede ser fija (cuando tenemos más de un dispositivo, en la fase de reconocimiento de la interrupción tenemos que haber identificado qué dispositivo pide atención y utilizaremos esta información para saber la dirección de la RSI correspondiente al dispositivo al que hemos de atender).

Una vez sabemos la dirección de inicio de la RSI, se carga en el registro PC y se inicia el ciclo de ejecución de la primera instrucción de la RSI. La RSI se ejecuta en modalidad supervisor.

3) Ejecución de la rutina de servicio de interrupción

a) Inicio de la ejecución de la RSI. Si no se ha hecho la inhibición de las interrupciones en la fase de reconocimiento de la interrupción, hay que hacerlo al inicio de la RSI. Esta inhibición se efectúa utilizando una instrucción específica del juego de instrucciones.

Después se deben guardar los registros que puedan ser modificados en la pila (el programador de la RSI sabe qué registros se cambian y, por lo tanto, puede determinar qué registros se deben guardar) y antes de acabar la ejecución hay que restaurarlos con los valores originales. Cuando se programa una RSI, se debe ser muy cuidadoso con la salvaguardia de los registros porque puede provocar que otros programas (los que detenemos para ejecutar la RSI) funcionen mal.

b) Intercambio del dato. La parte principal del código de la RSI tiene que ser específico para el periférico al que atendemos. De manera general se debe acceder al registro de datos para hacer el intercambio del dato. Si es necesario, se puede acceder primero a los registros de estado y de control para saber si la operación de E/S se ha hecho correctamente o se han producido errores.

Para programar el código de una RSI, hay que conocer con detalle el funcionamiento del periférico y qué función tiene cada uno de los bits de los registros del módulo de E/S que se utiliza para controlarlo. Generalmente, esta información que hay que conocer es bastante compleja cuando se debe analizar con detalle y puede variar mucho de un dispositivo a otro, incluso, en evoluciones de un mismo tipo de dispositivo.

c) Finalización de la ejecución de la RSI. Hay que recuperar el valor de los registros del procesador que se han guardado al principio de la RSI a la pila (en orden inverso al modo como se ha guardado, ya que lo tenemos guardado en la pila).

Si se han inhibido las interrupciones al principio de la RSI, hay que habilitarlas utilizando una instrucción específica del juego de instrucciones.

d) Retorno de interrupción. Antes de acabar la ejecución de la RSI, hemos de restaurar el estado del procesador para devolver el control al programa que se estaba ejecutando. Para ello, hemos de recuperar de la pila del sistema la información que se ha guardado de manera automática (los registros de estado y el PC). Esta información la recuperamos al ejecutar la última instrucción de la RSI, que es la instrucción de retorno de interrupción, habitualmente denominada *IRET*.

Una vez restaurado el estado del procesador, se reanuda la ejecución del programa que hemos detenido en las mismas condiciones iniciando un nuevo ciclo de ejecución de la instrucción. Y la petición del módulo de E/S ha quedado atendida.

Ejemplo

En este ejemplo se observa cómo se atiende una interrupción. Es especialmente importante fijarse en cómo queda el registro PC después de ejecutar cada fase y la evolución de la pila durante este proceso.

Las abreviaturas que se utilizan son:

- SP: *stack pointer* (puntero en la cima de la pila).
- PC: *program counter* (dirección de la instrucción que estamos ejecutando).
- RSI: rutina de servicio de interrupción.
- INT: señal que activa el módulo de E/S para pedir atención al procesador.


1. Petición del módulo de E/S. Al acabar la ejecución de la instrucción actual el procesador mira si la INT está activa y pasa a atender la petición.
Rutina para atender una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercambio del dato
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Cima de la pila
...
Programa en ejecución
...
PC → Instrucción actual
Instrucción siguiente
...
arriba
← INT
2. Reconocimiento de la interrupción, salvaguarda del estado (guarda PC y registro de estado) y llamada a la RSI (actualiza PC con la dirección de inicio de la RSI).
Rutina para atender una INT
PC → RSI Inhibir INT
PUSH Regs.
...
Intercambio del dato
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → PC (Inst. siguiente)
Reg. STATUS
Cima de la pila
...
Programa en ejecución
...
Instrucción actual
Instrucción siguiente
...
3. Se inicia la ejecución de la RSI (inhibir las interrupciones y guardar los registros que pueden ser modificados por la RSI).
Rutina para atender una INT
RSI: Inhibir INT
PUSH Regs.
PC → ...
Intercambio del dato
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Regs. de l'RSI
PC (Inst. siguiente)
Reg. STATUS
Cima de la pila
...
Programa en ejecución
...
Instrucción actual
Instrucción siguiente
...
4. Intercambio del dato.
Rutina para atender una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercambio del dato
...
PC → POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Reg. de la INT
PC (Inst. siguiente)
Reg. STATUS
Cima de la pila
...
Programa en ejecución
...
Instrucción actual
Instrucción siguiente
...
5. Final de la ejecución del RSI. Recuperar de la pila los registros guardados y habilitar las interrupciones.
Rutina para atender una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercambio del dato
...
POP Regs.
Habilitar INT
PC → IRET
...
Pila del sistema
SP → PC (Inst. siguiente)
Reg. STATUS
Cima de la pila
...
Programa en ejecución
...
Instrucción actual
Instrucción siguiente
...
6. Retorno de interrupción. Recuperar el registro de estado y el PC de la pila, y dejar el procesador en el mismo estado y el PC apuntando a la instrucción siguiente.
Rutina para atender una INT
RSI: Inhibir INT
PUSH Regs.
...
Intercambio del dato
...
POP Regs.
Habilitar INT
IRET
...
Pila del sistema
SP → Cima de la pila
...
Programa en ejecución
...
Instrucción actual
PC → Instrucción siguiente
...

3.2. Gestión de interrupciones con múltiples módulos de E/S

Hasta ahora hemos estudiado el caso más simple, en el que solo tenemos un módulo que funciona por interrupciones, pero la realidad es que un sistema de E/S por interrupciones tiene que gestionar múltiples dispositivos con capacidad de generar interrupciones. Veamos cómo afecta esto a la gestión de interrupciones y cómo lo hemos de implementar.

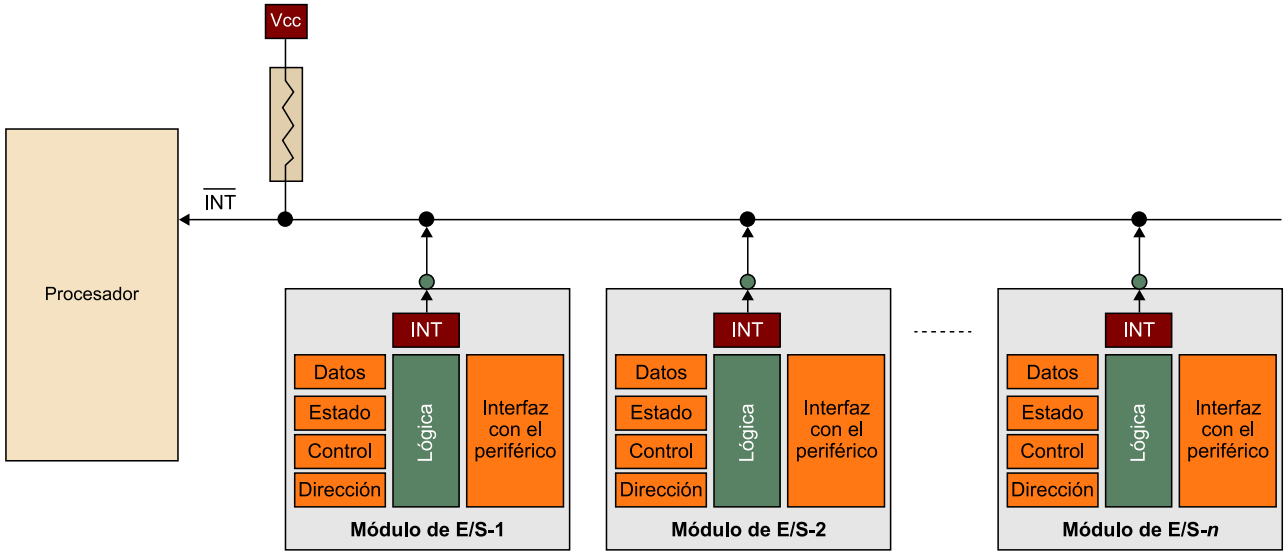
Cuando el procesador recibe una petición, sabe que algún módulo de E/S pide atención pero no sabe cuántos piden atención, ni cuáles son. Para identificar al que pide atención y decidir cuál se tiene que atender, las cuestiones principales que hay que resolver son:

- Identificar cuál es el periférico que pide atención.
- Determinar qué periférico hemos de atender primero en caso de que dos o más periféricos soliciten atención al mismo tiempo.
- Gestionar, si el sistema permite la nidificación, las prioridades para determinar si el periférico que solicita atención es más prioritario que otro al que ya atendemos.
- Obtener la dirección de la RSI correspondiente al periférico al que hemos de atender.

Estas cuestiones afectan principalmente al ciclo de reconocimiento de la interrupción y hacen este proceso bastante más complejo. El objetivo principal de los sistemas que nos permiten gestionar más de un módulo de E/S mediante interrupciones es conseguir que este proceso sea tan eficiente como sea posible, añadiendo mejoras al hardware que da soporte al sistema de atención de interrupciones.

3.3. Sistema con una única línea de petición de interrupción

Este es el caso más simple, en el que todos los módulos de E/S se conectan en colector abierto, utilizando un PULL-UP, a una única línea de petición de interrupción que llega al procesador.



La resistencia conectada a Vcc (tensión alta) sirve para implementar un PULL-UP, que da la funcionalidad de OR-CABLEADA considerando que la línea es activa a la baja. Si tenemos un 0 en la línea, se considera activa y si tenemos un 1, no se considera activa. Esto permite que los módulos de E/S estén conectados en colector abierto a la línea, de manera que si ningún módulo activa la salida, el *pull-up* mantiene la línea a 1 (tensión alta), condición de reposo de todos los dispositivos, y si algún módulo activa la salida, poniendo un 0 (tensión baja), la línea cambia de estado y el procesador puede reconocer que hay una petición pendiente de ser atendida.

La gestión de una interrupción en este sistema es análoga a la gestión de una interrupción con un único módulo de E/S. Ahora bien, para identificar qué periférico pide atención y gestionar las prioridades si más de un módulo de E/S ha pedido atención al mismo tiempo, el procesador ejecuta una única RSI que tiene una dirección de inicio fija. La RSI, mediante código y accediendo a los registros de estado de cada módulo de E/S, determina el módulo de E/S al que debe atender, de manera muy parecida a la encuesta (o *polling*) que se hace en E/S programada cuando tenemos más de un módulo de E/S. El código que utilizamos para atender al periférico es una subrutina de la RSI.

Las ventajas principales que ofrece este sistema son que:

- Como hacemos la encuesta de todos los módulos por programa, es muy flexible determinar las prioridades porque podemos implementar diferentes políticas de prioridades simplemente modificando el código de la RSI.

- No hay que hacer cambios en el hardware.

Las desventajas principales son que:

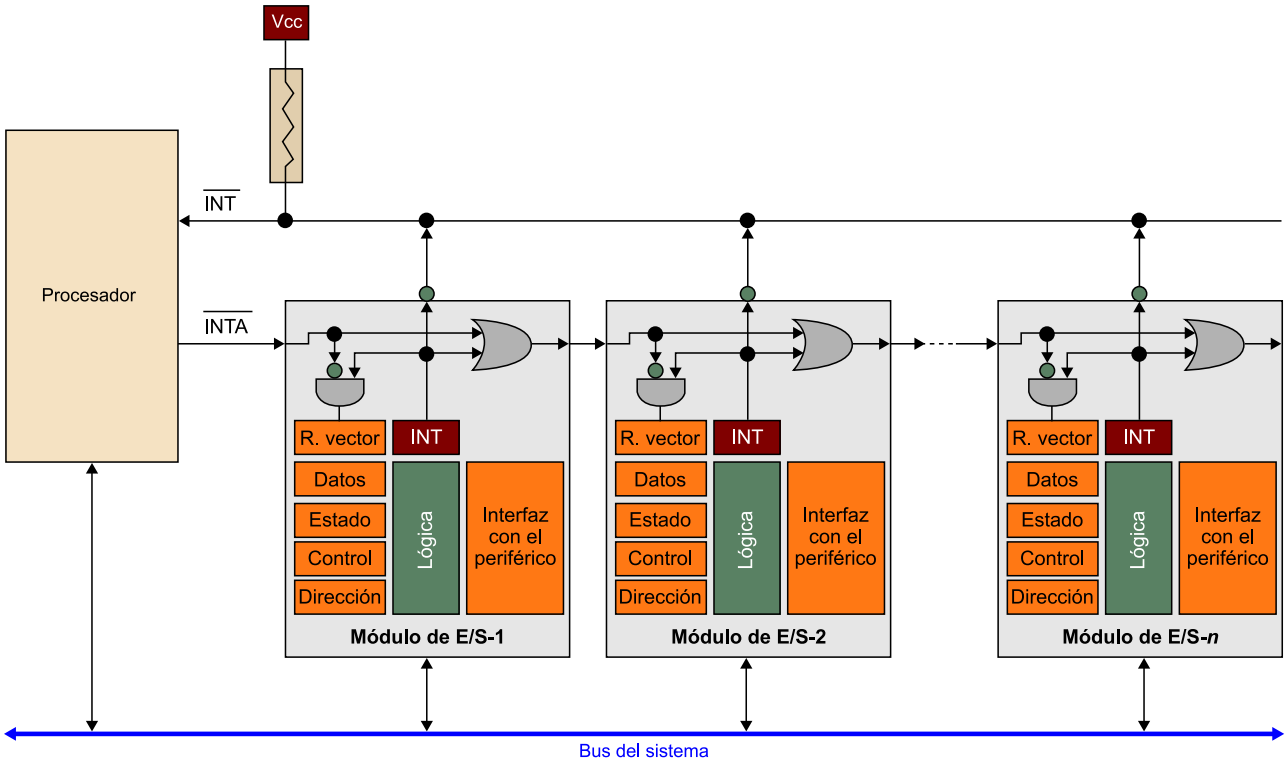
- La ejecución del código para hacer la encuesta de los módulos y la gestión de prioridades es muy costosa en tiempo, ya que estamos ejecutando un pequeño programa que accede a todos los módulos de E/S conectados a la línea.

- Como hay una sola línea de petición de interrupción, se tienen que inhibir las interrupciones y, por lo tanto, mientras atendemos una petición no podemos atender otras aunque sean más prioritarias.

3.4. Sistema con una línea de petición de interrupción y una línea de reconocimiento con encadenamiento

Este sistema con encadenamiento también se denomina ***daisy-chain***. Los módulos de E/S se conectan al procesador con una línea de petición de interrupción (INT) en colector abierto y una línea de reconocimiento de interrupción (INTA) que genera el procesador para indicar al módulo de E/S que se atiende la petición que ha hecho. Esta señal INTA se propaga mediante los módulos.

Es un sistema de conexión que permite, por una parte, reducir el tiempo necesario para gestionar las prioridades si más de un módulo de E/S ha pedido atención al mismo tiempo y, por otra parte, identificar qué periférico pide atención, sin tener que ejecutar un programa de encuesta que accede a todos los módulos de E/S.



Para implementar este sistema, hay que hacer algunos cambios en el hardware: el procesador necesita disponer de un punto de conexión por donde generar la señal de reconocimiento de interrupción (INTA); los módulos de E/S deben disponer de un circuito lógico muy simple para propagar la señal INTA al siguiente módulo de E/S dejando todos los módulos de E/S conectados uno tras otro y disponer de un nuevo registro, que denominamos **registro vector**, donde almacenamos un valor llamado **vector de interrupción** que enviamos al procesador mediante el bus del sistema. Este valor que almacena cada registro vector tiene que ser único, ya que el procesador lo utiliza para identificar al periférico que pide atención.

La gestión de una interrupción en este sistema es análoga a la gestión de una interrupción con un único módulo de E/S, y varía el ciclo de reconocimiento de la interrupción para gestionar las prioridades si más de un módulo de E/S ha pedido atención al mismo tiempo e identificar qué periférico demanda atención.

3.4.1. Interrupciones vectorizadas

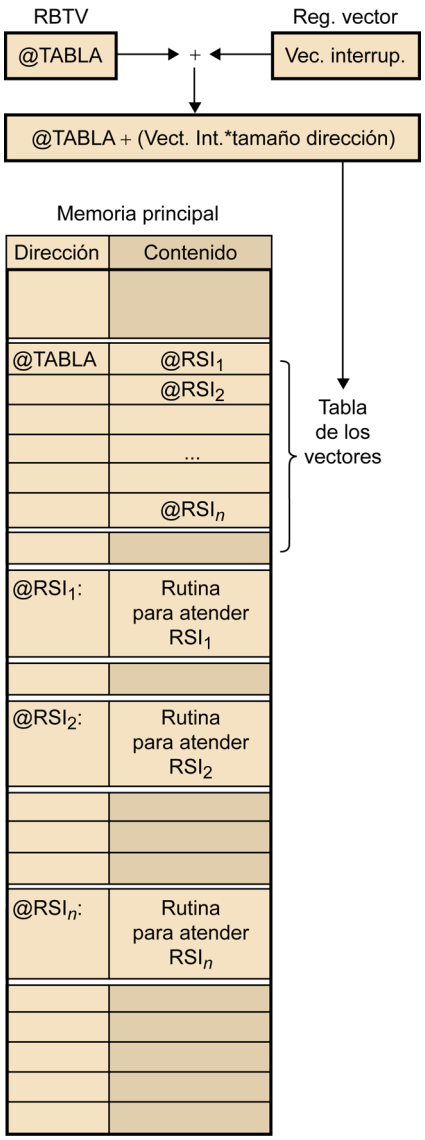
Se denomina **vectorización** a la técnica en la que el procesador identifica al módulo de E/S mediante la información que envía el mismo módulo de E/S. Cuando para tratar las interrupciones utilizamos esta técnica para identificar a quien hace la petición decimos que tenemos las **interrupciones vectorizadas**.

Una vez un módulo de E/S o más de uno han hecho la petición al procesador activando la INT y el procesador acepta la petición, activa la INTA y empieza el proceso para saber a cuál se debe atender.

Tratamiento de la señal INTA

La señal INTA tiene un funcionamiento parecido a la señal READ para leer un dato de memoria. Cuando el módulo recibe la señal INTA y tiene la INT activa, deposita el vector de interrupción en el bus del sistema para que el procesador lo pueda leer.

Esta señal INTA proveniente del procesador, activa a la baja, llega al primer módulo. Si este módulo ha solicitado atención, bloquea la propagación de la señal INTA y deposita en el bus del sistema el vector de interrupción, valor almacenado en el registro vector del módulo de E/S; si no ha solicitado atención (no tiene activa la INT), deja pasar la señal INTA al módulo siguiente, que hace lo mismo hasta que llega al último módulo conectado a esta línea.



Como vemos, los módulos de E/S quedan encadenados por esta señal INTA. El primer módulo de la cadena es el primero en recibir la señal y, por lo tanto, el más prioritario, y la propaga hasta el último, que es el menos prioritario. Si queremos cambiar las prioridades, nos veremos obligados a cambiar el orden de los módulos dentro de la cadena físicamente.

El procesador lee el vector de interrupción del bus del sistema. Con este valor identifica el periférico al que tiene que atender y lo utiliza para obtener la dirección de inicio de la RSI (esta dirección es la que hemos de cargar en el registro PC para hacer la llamada a la RSI).

La manera más habitual de obtener la dirección de inicio de la RSI es utilizar una **tabla de vectores de interrupción** almacenada en la memoria principal, donde tenemos guardadas las direcciones de inicio de las RSI asociadas a cada petición de interrupción a la que tenemos que atender. El vector de interrupción lo utilizamos para acceder a la tabla. Esta tabla puede estar almacenada en una dirección fija de memoria o en una dirección programable almacenada en el **registro base de la tabla de vectores (RBTV)**. Si tenemos la tabla

de vectores en una dirección fija, hemos de determinar la posición dentro de la tabla de vectores, a partir del vector de interrupción. Si tenemos la tabla de vectores en una dirección programable, hemos de determinar la posición dentro de la tabla de vectores sumando el RBTV a un índice obtenido a partir del vector de interrupción.

Desplazamiento dentro de la tabla de vectores

Una manera de obtener el desplazamiento dentro de la tabla de vectores a partir del vector de interrupción, como el vector de interrupción identifica el dispositivo (y no la dirección dentro de la tabla o la dirección de la rutina misma), es multiplicar el vector de interrupción por las posiciones de memoria que ocupa una dirección dentro de la tabla de vectores. Si las direcciones de memoria almacenadas en la tabla de vectores ocupan 4 bytes (32 bits) y cada posición de memoria es de un byte, el vector de interrupción se tiene que multiplicar por 4. Hacer esta operación en binario es muy simple: solo hay que desplazar el vector de interrupción dos posiciones a la izquierda (añadir dos ceros a la parte derecha) para obtener el desplazamiento dentro de la tabla. En sistemas más complejos puede ser necesario hacer otro tipo de operaciones.

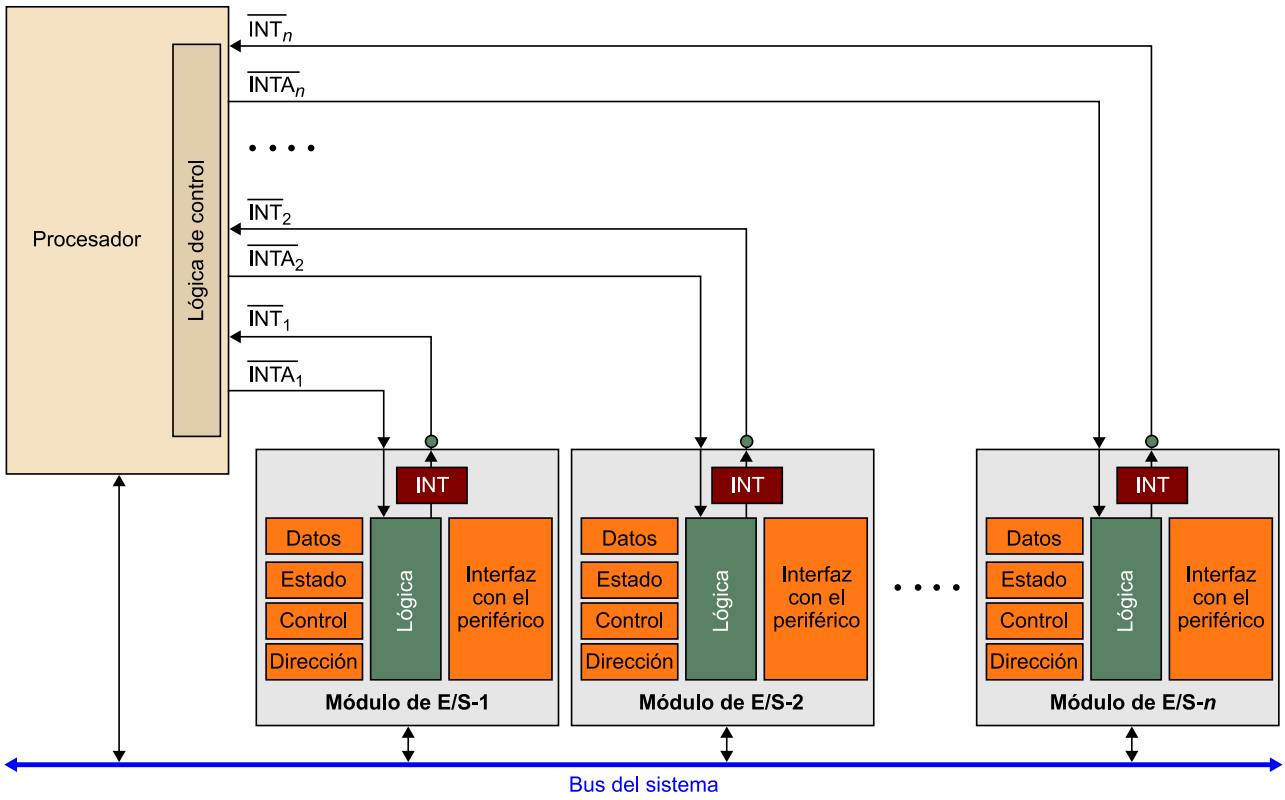
Las principales ventajas de este sistema con encadenamiento son que la identificación se efectúa en un tiempo parecido a la duración de un ciclo de bus, que es mucho menos costoso en tiempo que hacer la encuesta a todos los módulos de E/S y que también resuelve la gestión de prioridad cuando hay peticiones simultáneas.

Las desventajas principales son que el sistema de prioridades es fijo e inalterable –cambiar el orden de prioridades implica modificar el hardware– y que, como tiene una única línea de petición de interrupciones, no permite que otros periféricos más prioritarios sean atendidos si ya atendemos a una petición de interrupción sea cual sea la prioridad que tiene.

3.5. Sistema con líneas independientes de petición de interrupciones y de reconocimiento

En este sistema tenemos una línea de petición de interrupción y una de reconocimiento de interrupción para cada módulo de E/S. De esta manera conseguimos que la identificación del periférico sea inmediata y permitimos la **nidificación de interrupciones**, es decir, una interrupción puede interrumpir la ejecución de una RSI que atiende una petición menos prioritaria que se ha producido con anterioridad.

Para implementar este sistema es necesario que el procesador disponga de los puntos de conexión necesarios para conectar las señales de petición y reconocimiento de cada uno de los módulos de E/S que puedan generar una interrupción hacia el procesador y un circuito específico para hacer el control de estas líneas de petición y reconocimiento de interrupciones. Los módulos de E/S solo deben poder generar una petición de interrupción y recibir la señal de reconocimiento de interrupción generada por el procesador.



La gestión de una interrupción en este sistema es análoga a la gestión de una interrupción con un único módulo de E/S, y varía la fase de reconocimiento de la interrupción para gestionar las prioridades e identificar qué periférico pide atención.

El proceso para hacer el ciclo de reconocimiento de la interrupción es el siguiente: una vez uno o más módulos de E/S han hecho la petición al procesador activando la INT, el procesador decide si acepta la petición, inhibe de manera selectiva las interrupciones, activa la INTA correspondiente y obtiene la dirección de la RSI para atender aquella petición.

En un sistema con una única línea de petición de interrupción hemos visto que cuando aceptamos una petición hay que inhibir las interrupciones y, por lo tanto, no podemos aceptar nuevas peticiones hasta que se ha acabado la atención de esta petición. En un sistema con múltiples líneas de petición y reconocimiento de interrupción también hemos de inhibir las peticiones de interrupción de la línea de la que hemos aceptado la petición, pero hay que tener un mecanismo para decidir si se deben permitir peticiones de las otras líneas para enmascarar las interrupciones de manera selectiva según una política de prioridades.

Hay 2 formas básicas para enmascarar de manera selectiva las interrupciones:

- Enmascaramiento individual.
- Enmascaramiento por nivel.

En el **enmascaramiento individual** disponemos de un **registro de interrupciones**, que tiene 2 bits por cada línea de petición de interrupción: un bit de máscara que nos permite habilitar o inhibir las peticiones de una línea y opcionalmente un bit de interrupción que nos indica si se ha aceptado la petición, es decir, si ha habido una petición del módulo de E/S y están habilitadas las peticiones de interrupciones para aquella línea.

Este sistema de enmascaramiento también dispone de 1 bit para el enmascaramiento general de interrupciones, un bit de máscara que nos permite habilitar o inhibir todas las peticiones de interrupción enmascarables. Para aceptar una petición de una línea, tienen que estar activos los bits de la línea y también el bit general de interrupciones.

Si el procesador en la fase de comprobación de interrupción detecta que hay alguna petición de interrupción, hace un ciclo de reconocimiento de interrupción y se determina qué petición se tiene que atender accediendo a los bits de máscara del registro de interrupciones. Si se acepta una petición, se deben inhibir las interrupciones de aquella línea y de las líneas menos prioritarias modificando los bits de máscara correspondientes. En computadores simples o microcontroladores, el registro de interrupción es visible al programador y hemos de identificar quién pide atención y decidir a quién atendemos primero mediante un programa; en sistemas más complejos generalmente se dispone de un hardware específico.

Esta forma de enmascaramiento se suele utilizar en sistemas con pocas líneas de petición de interrupción porque, a pesar de ser muy flexible, si no se dispone de un hardware específico la gestión de prioridades se debe realizar por programa y eso la hace más lenta. Es una situación parecida a la encuesta que hacemos cuando tenemos una única línea de petición de interrupción, pero en lugar de tener que acceder a los registros de los módulos de E/S mediante el bus del sistema, accedemos a un registro del procesador que es mucho más fácil y rápido.

En el **enmascaramiento por nivel**, a cada línea de petición de interrupción se asigna un **nivel de interrupción**, denominado también **nivel de ejecución**. Si el número de niveles es inferior al número de líneas de petición de interrupción, se agrupan varias líneas en un mismo nivel.

Este nivel de ejecución se utiliza para gestionar las prioridades, de manera que si ya atendemos una petición de interrupción, solo aceptamos una nueva petición y detenemos la atención si esta petición tiene un nivel de ejecución más prioritario.

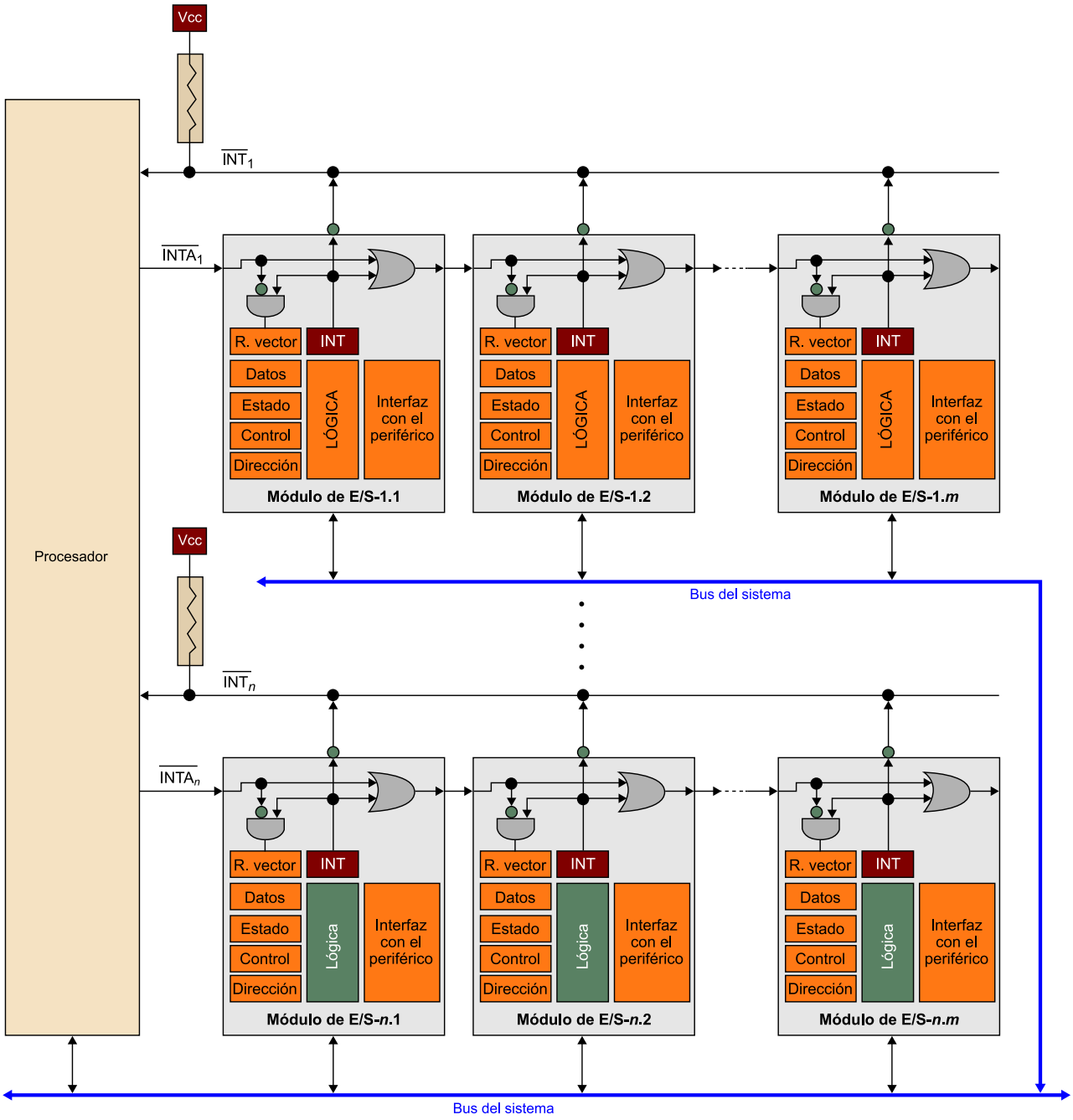
El nivel de ejecución se puede obtener fácilmente conectando las líneas de petición de interrupción a un codificador de prioridad y el código generado se almacena en un registro del procesador, generalmente en el registro de estado del procesador. Para decidir si aceptamos una determinada petición, con un circuito comparador, hemos de comparar el valor obtenido del codificador de prioridad con el nivel de ejecución que tenemos almacenado. Si es más prioritario el valor obtenido del codificador, aceptamos la petición y utilizamos este valor como un vector de interrupción para obtener la dirección de la RSI para atender a esta petición. El nuevo nivel de ejecución se actualiza de manera automática una vez hecha la salvaguarda del estado del procesador (se ha almacenado el PC y el registro de estado en la pila del sistema) y empieza la ejecución de la RSI. De esta manera, cuando acabamos la ejecución de la RSI que atiende esta petición más prioritaria y restauramos el estado del procesador para reanudar la ejecución del programa parado, recuperamos también el nivel de ejecución que tiene, ya que queda guardado con el resto de la información del registro de estado.

Esta forma de enmascaramiento es la más habitual en este tipo de sistemas.

Por otra parte, para garantizar un funcionamiento correcto del computador, el procesador debe disponer de líneas de petición de interrupción que no se puedan enmascarar. Solo están inhibidas en momentos concretos durante el ciclo de reconocimiento de las interrupciones y durante el retorno de interrupción para garantizar la estabilidad del sistema.

Las principales ventajas de este sistema con líneas independientes de petición de interrupciones y de reconocimiento son que permite la nidificación, que la identificación del periférico es muy rápida y que la gestión de prioridades resulta muy flexible.

La desventaja principal es que no se puede aumentar el número de módulos de E/S a los que podemos atender, ya que implica un rediseño del procesador o, como se puede ver en el esquema siguiente, utilizar sistemas híbridos estableciendo por ejemplo un sistema de encadenamiento (*daisy-chain*) en cada una de las líneas de petición y reconocimiento de interrupción, aunque esta opción hace la gestión de las interrupciones más compleja y lenta.



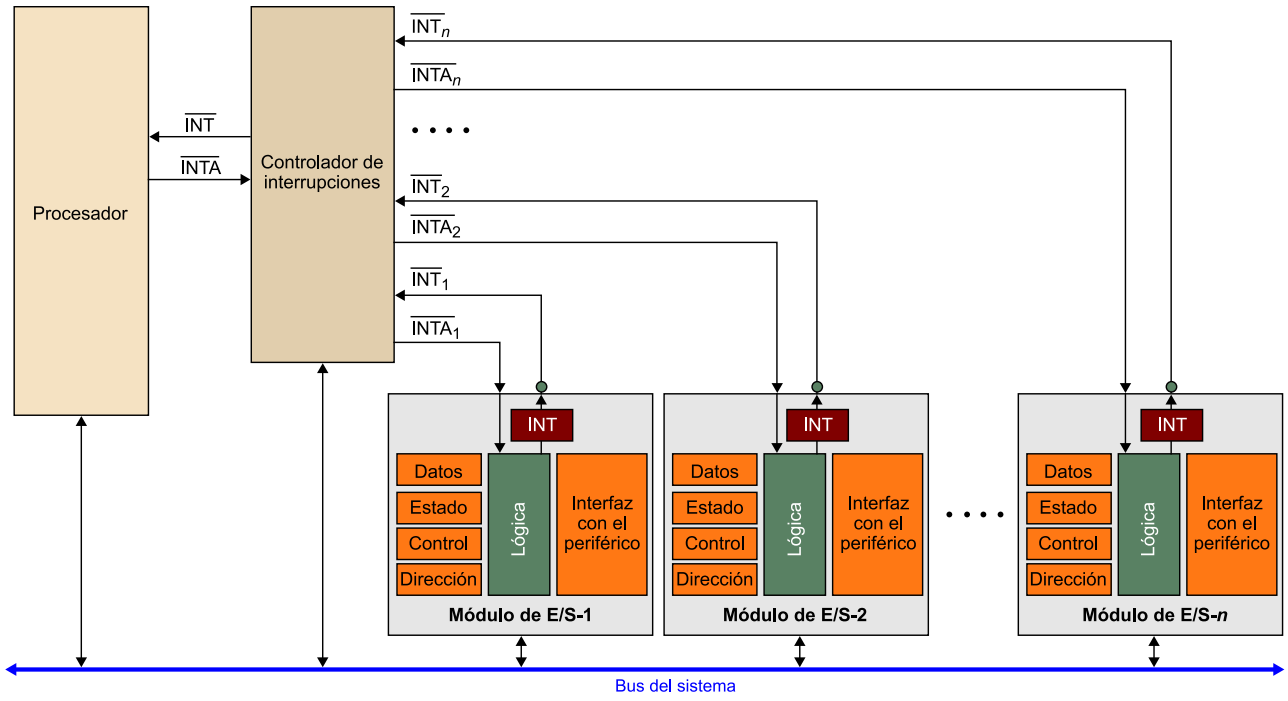
3.6. Sistema con controladores de interrupciones

En un sistema con controladores de interrupciones se añade un nuevo elemento especializado para gestionar las peticiones de interrupción que denominamos **controlador de interrupciones**. Las líneas de petición de interrupción y de reconocimiento de cada módulo de E/S están conectadas al controlador de interrupciones. De esta manera el procesador solo debe tener una línea de petición de interrupción y una de reconocimiento conectadas al controlador de interrupción para gestionar las peticiones provenientes de los módulos de E/S.

Las funciones del controlador de interrupción son las siguientes:

- Definir una política de prioridades para los módulos de E/S conectados al controlador.
- Identificar qué módulo de E/S pide atención e informar el procesador.

La gestión de una interrupción en este sistema es análoga a la gestión de una interrupción con un único módulo de E/S, y varía la fase de reconocimiento de la interrupción para gestionar las prioridades e identificar qué periférico pide atención.



El proceso para hacer el reconocimiento de la interrupción es el siguiente: para identificar el periférico que pide atención y obtener la dirección de la RSI que tiene que atender la petición, utilizamos un sistema de vectorización muy parecido al descrito en el *daisy-chain*, pero ahora no es necesario que el módulo de E/S tenga un registro vector. El controlador de interrupciones dispone de un conjunto de registros vector donde se guardan los vectores de interrupción asociados a cada línea.

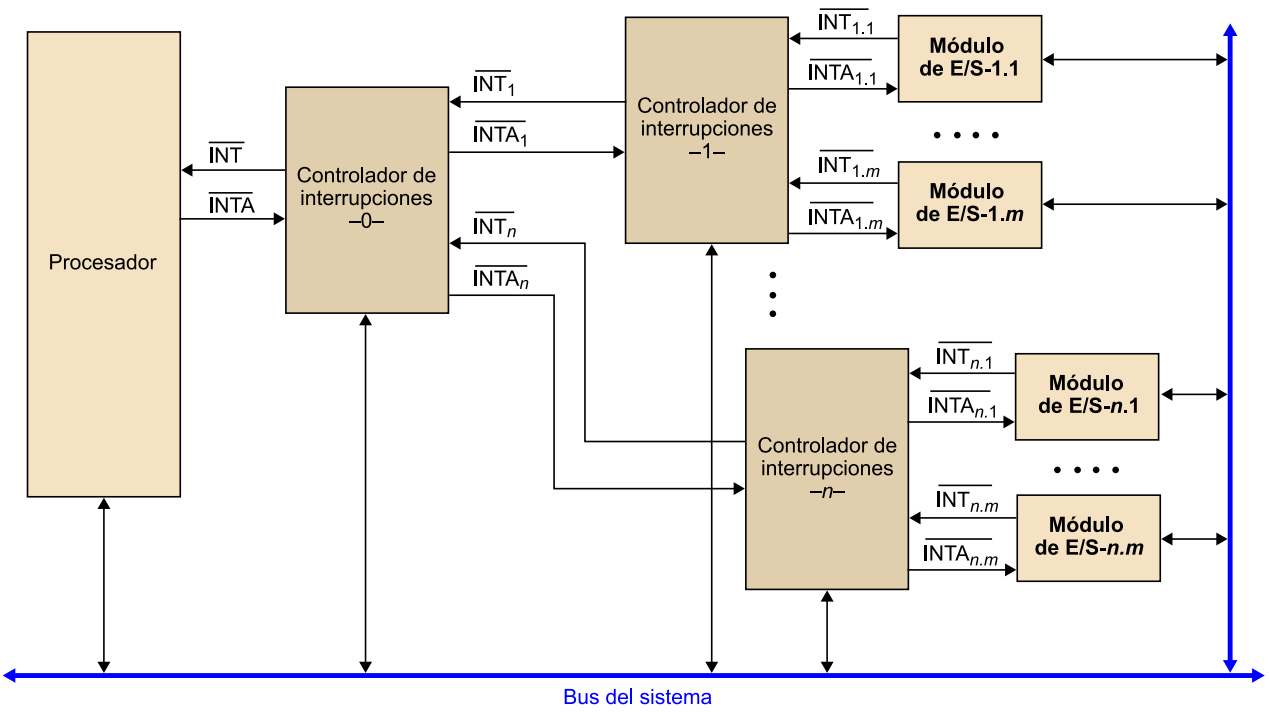
Cuando llegue una petición de interrupción de un módulo de E/S muy prioritario, el controlador de interrupciones hace la petición al procesador activando la señal INT y el procesador contesta activando la señal INTA. Cuando el controlador de interrupciones recibe la señal INTA coloca el vector de interrupción, correspondiente al módulo de E/S que ha hecho la petición, en el bus de datos y el procesador lee el vector. De esta manera, el procesador puede obtener la dirección de la RSI y empezar la ejecución de la rutina para

hacer la transferencia de datos, comunicándose directamente con el módulo de E/S. La transferencia de datos no se gestiona mediante el controlador de interrupciones.

Las principales ventajas de este sistema con un controlador de interrupción son que la identificación del periférico es bastante rápida y la gestión de prioridades flexible, y también que el procesador solo necesita disponer de una línea INT y una INTA para gestionar múltiples módulos de E/S.

La desventaja principal es que a causa de la gestión de prioridades que hace el controlador no se puede hacer la salvaguarda del nivel de ejecución de manera automática, sino que se debe realizar ejecutando un pequeño programa que acceda al controlador de interrupciones y hace el proceso de salvaguarda del estado más lento.

Si tenemos un sistema que utiliza controladores de interrupción y queremos aumentar el número de módulos de E/S, hemos de conectar el controlador de interrupción en cascada como se muestra a continuación, pero es necesario que los controladores estén diseñados específicamente para conectarlos de esta manera.



4. E/S con acceso directo a memoria

Las técnicas de E/S que hemos visto hasta ahora requieren una dedicación importante del procesador (ejecutar un fragmento de código) para hacer simples transferencias de datos. Si queremos transferir bloques de datos, estas técnicas todavía ponen más en evidencia la ineficiencia que tienen. En E/S programada implica que el procesador no pueda hacer nada más y en E/S por interrupciones descargamos el procesador de la sincronización a costa de hacer las rutinas de atención más largas para garantizar el estado del procesador, lo que limita la velocidad de transferencia.

En este apartado describiremos una técnica mucho más eficiente para transferir bloques de datos, el **acceso directo a memoria** (**DMA**). En esta técnica el procesador programa la transferencia de un bloque de datos entre el periférico y la memoria encargando a un nuevo elemento conectado al bus del sistema hacer toda la transferencia. Una vez acabada, este nuevo elemento avisa el procesador. De esta manera, el procesador puede dedicar todo el tiempo que dura la transferencia del bloque a otras tareas. Este nuevo elemento que gestiona toda la transferencia de datos entre el periférico y la memoria principal lo denominamos **módulo** o **controlador de DMA** o también en versiones más evolucionadas **canal** o **procesador de E/S**.

Utilizando la técnica de E/S por DMA se descarga al procesador de la responsabilidad de llevar a cabo la sincronización y el intercambio de datos entre el periférico y la memoria.

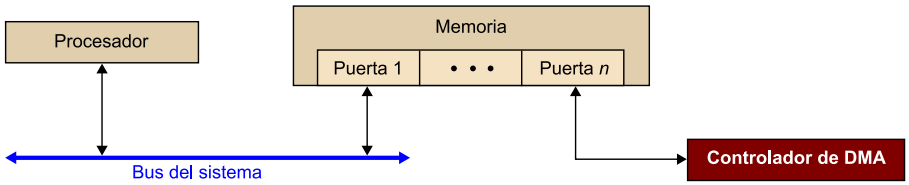
Por otra parte, nos aparece una nueva problemática en el computador, ya que hay dos dispositivos –el procesador y el controlador de DMA– que tienen que acceder de manera concurrente a la memoria y hay que establecer un mecanismo para resolver este conflicto.

4.1. Acceso concurrente a memoria

Básicamente hay dos maneras de resolver el acceso concurrente a memoria:

1) Conexiones independientes, utilizando memorias multipuerta. Hay una conexión independiente para cada dispositivo que tiene que acceder a la memoria. Eso permite al controlador de DMA acceder a la memoria sin que el procesador tenga que intervenir. Aunque la afectación al procesador es muy

baja, esta solución no se suele utilizar porque hace aumentar el coste de la memoria. También aumenta el tiempo de acceso, sobre todo cuando hay más de una puerta que quiere acceder al mismo bloque de información.



2) Conexión compartida, utilizando robo de ciclo. En este caso, la memoria solo necesita una única puerta y tanto el procesador como el controlador de DMA comparten el bus del sistema para acceder a la memoria. En la mayoría de los sistemas, el procesador es quien controla el bus; por lo tanto, hay que establecer un mecanismo para que el procesador pueda ceder el bus al controlador de DMA y este controlador pueda hacer el intercambio de los datos con la memoria. Este mecanismo se denomina *robo de ciclo* y es el que se utiliza más frecuentemente para gestionar el acceso concurrente a la memoria.

Para controlar el acceso al bus, son necesarias dos señales, BUSREQ y BUSACK (parecidas a las señales INT e INTA utilizadas en interrupciones). Con la señal BUSREQ el controlador de DMA solicita el control del bus y el procesador cede el bus activando la señal BUSACK.

> La diferencia más importante entre el acceso directo a memoria y la gestión de interrupciones es que el procesador puede inhibir las interrupciones total o parcialmente, mientras que la cesión del bus no la puede inhibir y está obligado a ceder siempre el control del bus cuando el controlador de DMA lo solicita.

Las principales ventajas de este sistema son que la atención es muy rápida porque no se puede inhibir la cesión del bus y que no es necesario guardar al estado del procesador, que solo queda parado durante un tiempo muy breve mientras el controlador de DMA tiene el control del bus.

La cesión del bus no es inmediata. El procesador solo puede ceder el bus al acabar cada una de las fases del ciclo de ejecución de las instrucciones. Una vez el controlador de DMA libera el bus, de manera que se acaba el robo de ciclo, el procesador continúa la ejecución de la siguiente fase de la instrucción en curso. Por lo tanto, el robo de ciclo se puede producir en diferentes puntos dentro del ciclo de ejecución de una instrucción.

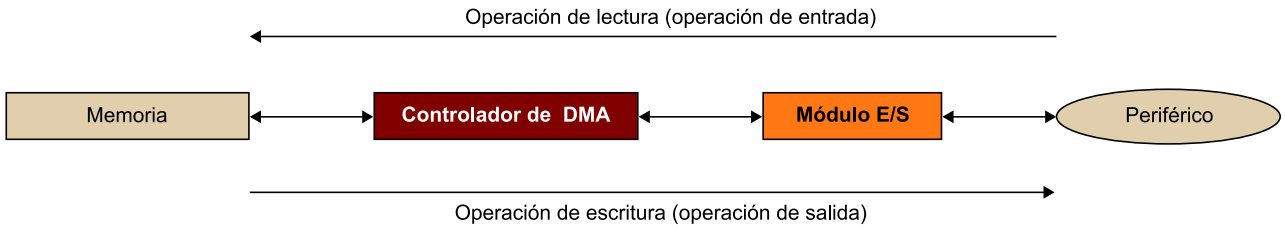
La principal desventaja de este sistema es que se alarga el tiempo de ejecución de las instrucciones a causa de los posibles robos de ciclo que se pueden producir.

4.2. Operación de E/S con acceso directo a memoria

El proceso para hacer una transferencia de E/S utilizando esta técnica es el siguiente:

1) Programación de la operación de E/S: el procesador envía la información necesaria al controlador de DMA para que este controlador pueda gestionar toda la transferencia de datos. Una vez acaba la programación de la operación de E/S, el procesador puede ejecutar otros programas mientras se realiza la transferencia.

2) Transferencia del bloque de datos: las dos operaciones básicas que hace el controlador de DMA son la lectura de un bloque de datos de un periférico y la escritura de un bloque de datos en un periférico, aunque también puede hacer otras operaciones. Todos los datos de la transferencia pasan por el controlador de DMA.



3) Finalización de la operación de E/S: cuando se ha acabado la transferencia del bloque, el controlador de DMA envía una petición de interrupción al procesador para informar de que se ha acabado la transferencia de datos.

Nota

Para utilizar esta técnica de E/S en un computador, es necesario considerar tanto aspectos del software como del hardware.

4.3. Controladores de DMA

En un sistema de este tipo las implicaciones con respecto al hardware y al sistema de conexión de este hardware son diversas.

El procesador debe disponer de un sistema para gestionar interrupciones, como hemos explicado en el apartado anterior. El controlador de DMA avisa al procesador de que se ha acabado la transferencia del bloque de datos mediante una petición de interrupción.

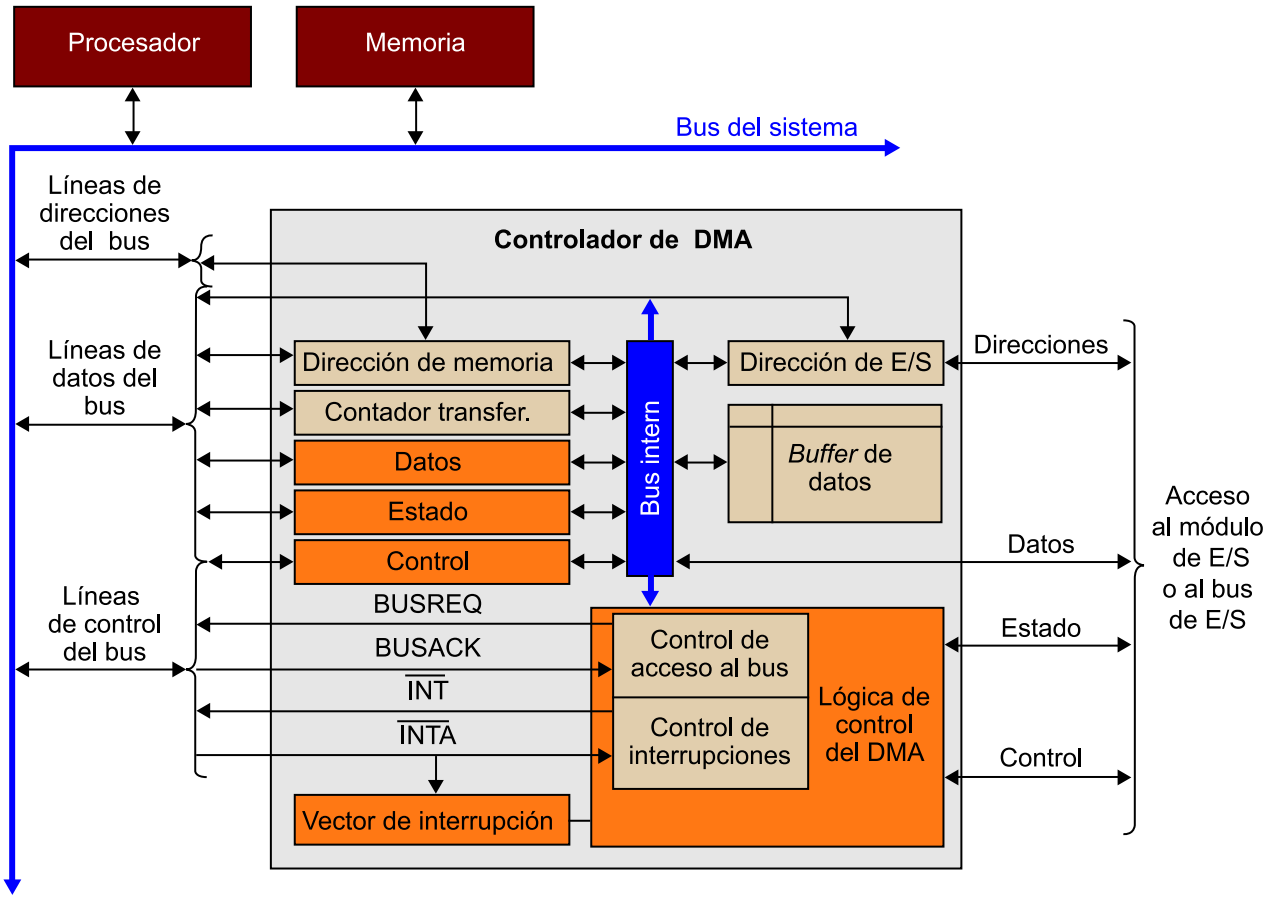
Los elementos básicos que debe tener el controlador de DMA para gestionar una transferencia de datos entre el periférico y la memoria son los siguientes:

- Un banco de registros para gestionar las operaciones de E/S.
- La lógica de control necesaria para gestionar las transferencias entre la memoria y el módulo de E/S.
- La lógica necesaria para gestionar la interrupción de la finalización de la operación de E/S.

- Señales de control para acceder al bus del sistema y para la gestión de interrupciones (BUSREQ, BUSACK, INT, INTA, etc.).

El banco de registros del controlador de DMA está formado por los registros siguientes:

- **Registro de control**: se utiliza generalmente para dar las órdenes al controlador.
- **Registro de estado**: da información del estado de la transferencia de datos.
- **Registro de datos**: almacena los datos que se quieren intercambiar.
- **Registro de direcciones de memoria**: indica la dirección de memoria donde se leerán o se escribirán los datos que se tienen que transferir en cada momento.
- **Registro contador**: indica inicialmente el número de transferencias que se deben hacer, se irá actualizando durante la transferencia hasta que valga cero e indicará en cada momento el número de transferencias que quedan por hacer.
- **Registro de direcciones de E/S**: indica la posición dentro del periférico donde se leerán o se escribirán los datos que se tienen que transferir en cada momento.

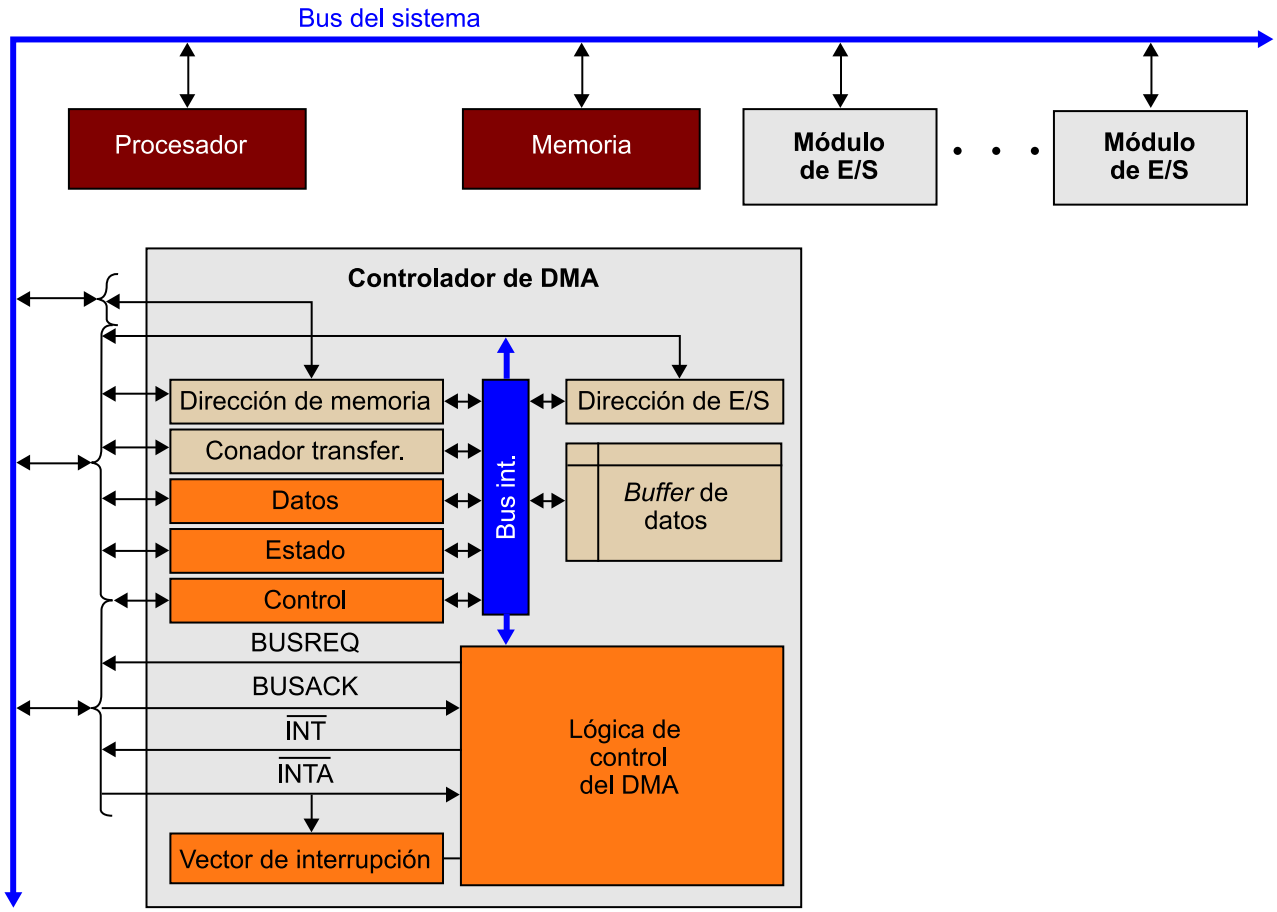


4.3.1. Formas de conexión de los controladores de DMA

A continuación trataremos las configuraciones más habituales del controlador de DMA.

La configuración siguiente es la más simple pero también la más ineficiente porque se tienen que hacer dos accesos al bus: uno para acceder a memoria y otro para acceder al módulo de E/S.

Conexión del controlador de DMA y de los módulos de E/S con un único bus



En caso de tener más de un controlador de DMA conectado al bus, la gestión de prioridades para decidir a quién se cede el bus se hace de manera muy parecida a la gestión de prioridades explicada en la E/S por interrupciones:

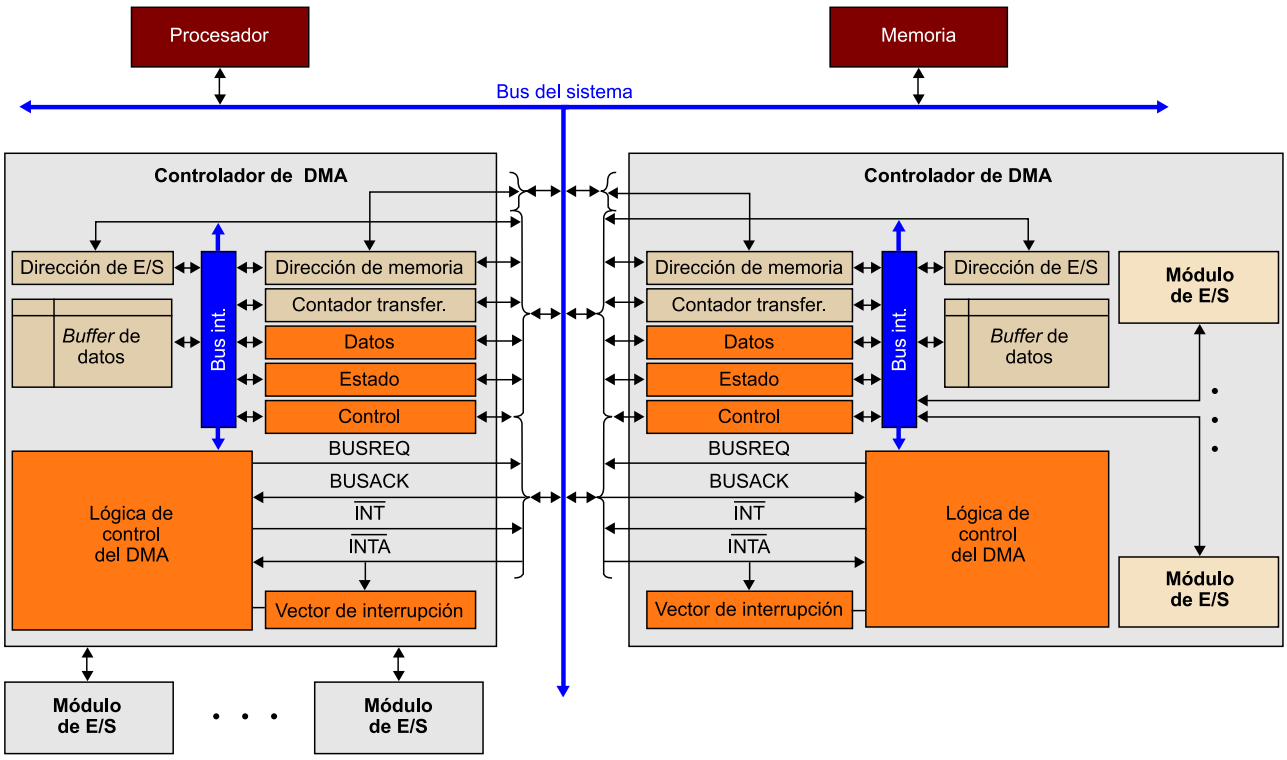
- Sistema con encadenamiento (*daisy-chain*), en el que se hace el encadenamiento de la señal BUSACK.

- Utilizar un controlador de DMA que gestione múltiples transferencias, de manera parecida al controlador de interrupciones, pero ahora la transferencia de datos con la memoria se efectúa mediante el controlador de DMA y no directamente entre el módulo de E/S y el procesador, como sucede cuando utilizamos controladores de interrupciones. En este caso, podemos

encontrar tanto conexiones independientes entre el controlador y cada módulo de E/S, como todos los módulos conectados a un bus específico de E/S.

Una mejora de la configuración anterior consiste en conectar los módulos de E/S directamente al controlador de DMA, de manera que necesitamos solo un acceso al bus del sistema. Esta configuración también permite conectar controladores de DMA que integran la lógica del módulo de E/S.

Conexión punto a punto entre el controlador de DMA y los módulos de E/S

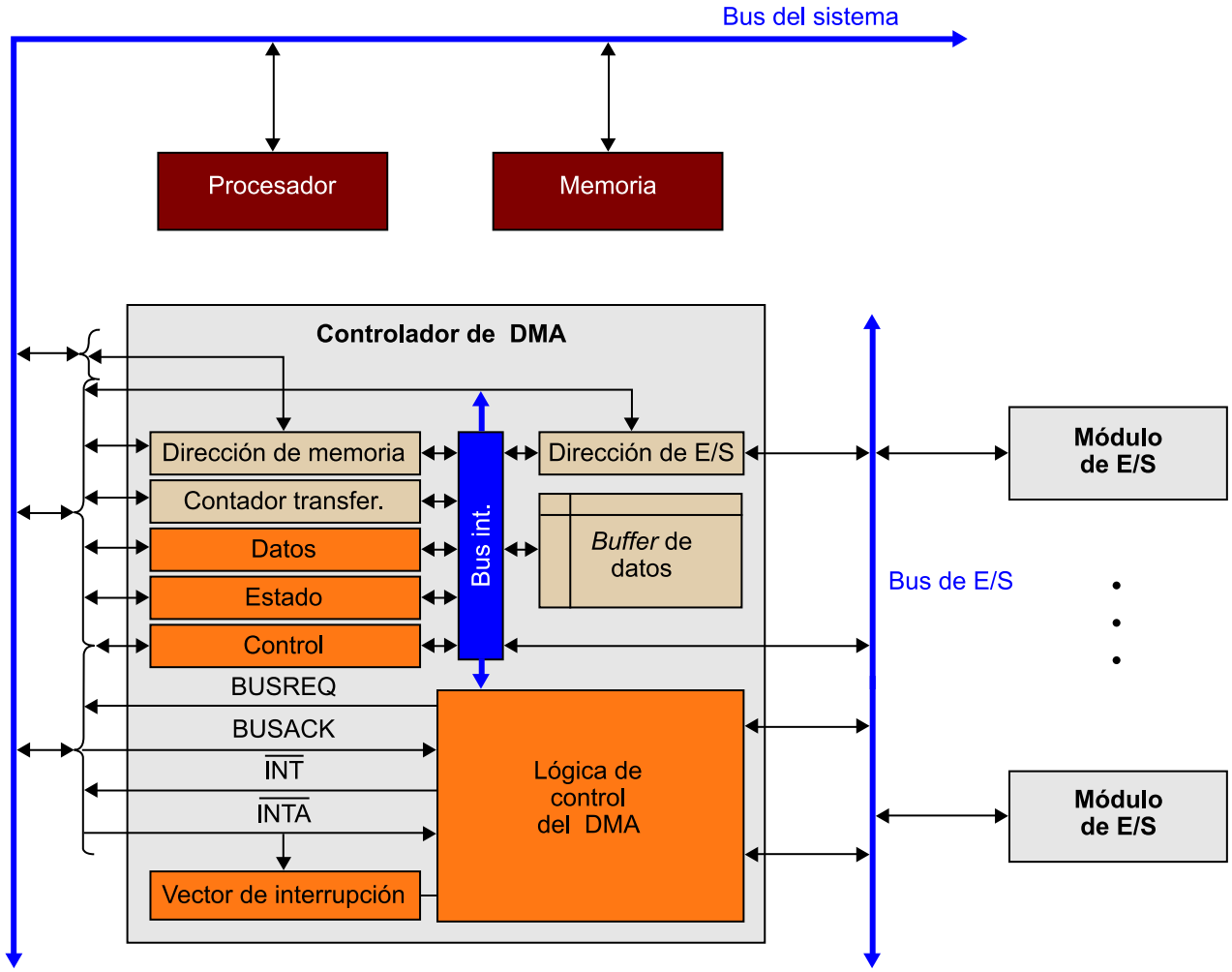


Transferencias simultáneas

Si se tienen que hacer las operaciones de E/S con varios periféricos simultáneamente, hemos de replicar parte del conjunto de registros (direcciones, contador, bits de control, etc.) tantas veces como transferencias de datos simultáneos queramos gestionar.

Otra configuración posible consiste en conectar los módulos de E/S al controlador de DMA mediante un bus de E/S. El flujo de información entre el controlador de DMA y los módulos de E/S no interfiere con los accesos al bus del sistema.

Conexión del controlador de DMA y de los módulos de E/S mediante un bus de E/S



4.3.2. Operación de E/S mediante un controlador de DMA

Para hacer una transferencia de E/S utilizando esta técnica se siguen los pasos siguientes:

1) Programación de la operación de E/S. El procesador tiene que enviar la información necesaria al controlador de DMA para que este controlador pueda hacer la transferencia de manera autónoma. Esta información se debe escribir en el banco de registros del controlador de DMA.

La información imprescindible que se ha de escribir es la siguiente:

a) Operación que hay que hacer. Las operaciones básicas son de lectura o escritura, pero también se pueden hacer otros tipos de operaciones de control del periférico. Esta información se escribe en el registro de control mediante las líneas de control o las líneas de datos del bus de sistema.

b) Dirección de memoria. Dirección inicial de memoria donde se leerán o se escribirán los datos que se han de transferir considerando que el bloque de datos que moveremos está en direcciones contiguas de memoria. Esta información se escribe en el registro de direcciones de memoria.

c) Tamaño del bloque de datos. Número de transferencias que se tiene que hacer mediante el bus del sistema para transferir todo el bloque de datos. Esta información se escribe en el registro contador.

d) Dirección del periférico. Posición inicial dentro del periférico donde se leerán o se escribirán los datos que se han de transferir. Esta información se escribe en el registro de direcciones de E/S. Esta información depende del periférico y en cierta medida del tipo de información que se tiene que transmitir. Si es un dispositivo de almacenamiento, hemos de saber dónde está guardado para recuperar después los datos.

2) Transferencia del bloque de datos. Dado que el procesador ha delegado al controlador de DMA la operación de E/S, el controlador de DMA debe realizar la transferencia de datos de todo el bloque. La transferencia se hace dato a dato accediendo directamente a memoria sin la intervención del procesador.

Los pasos para una operación de lectura y para una operación de escritura son los siguientes:

Después de transferir cada dato, el controlador de DMA decrementa el registro contador y actualiza el registro de direcciones de memoria con la dirección donde tenemos que leer o almacenar el dato siguiente. Cuando el registro contador llega a cero, el controlador de DMA genera una petición de interrupción para avisar al procesador de que se ha acabado la transferencia del bloque.

3) Finalización de la operación de E/S. Una vez el controlador de DMA envía la petición de interrupción al procesador para informar de que se ha acabado la transferencia del bloque de datos, el procesador ejecuta la RSI correspondiente para acabar la operación de E/S.

Cabe señalar que el controlador de DMA se debe dotar del hardware necesario para que pueda generar la petición de interrupción según el sistema de interrupciones que utiliza el procesador; por ejemplo, si utilizamos un sistema con encadenamiento (*daisy-chain*), hemos de poder generar una INT, recibir una INTA y la lógica para propagarla y un registro vector para identificarse.

4.4. Controlador de DMA en modo ráfaga

Una manera de optimizar las operaciones de E/S por DMA consiste en reducir el número de cesiones y recuperaciones del bus. Para hacerlo, en lugar de solicitar y liberar el bus para cada dato que se tiene que transferir, se solicita y se libera el bus para transferir un conjunto de datos de manera consecutiva. Esta modalidad de transferencia se llama **modo ráfaga**.

Para hacer la transferencia de este conjunto de datos, que denominamos **ráfaga**, el controlador de DMA tiene que disponer de una memoria intermedia (*buffer*), de modo que la transferencia de datos entre la memoria y el controlador de DMA se pueda hacer a la velocidad que permita la memoria y no quedando limitada a la velocidad del periférico.

Este modo de funcionamiento no afecta a la programación ni a la finalización de la operación de E/S descrita anteriormente, pero sí que modifica la transferencia de datos.

El funcionamiento de la transferencia del bloque de datos es el siguiente: en el caso de la lectura, cada vez que el módulo de E/S tiene un dato disponible, el controlador de DMA lo almacena en la memoria intermedia y decrementa el registro contador. Cuando la memoria intermedia está llena o el contador ha llegado a cero, solicita el bus. Una vez el procesador le cede el bus, escribe en memoria todo el conjunto de datos almacenados en la memoria intermedia, hace tantos accesos a memoria como datos tenemos y actualiza el registro de direcciones de memoria en cada acceso. Al acabar la transferencia del conjunto de datos, libera el bus.

En el caso de la escritura, el controlador de DMA solicita el bus y, cuando el procesador cede el bus, el controlador de DMA lee un dato de la memoria, lo almacena en la memoria intermedia, decrementa el registro contador y actualiza el registro de direcciones de memoria. Cuando la memoria intermedia está llena o el contador ha llegado a cero, libera el bus. A continuación transfiere todo este conjunto de datos almacenados en la memoria intermedia al módulo de E/S y espera para cada dato a que el módulo de E/S esté preparado.

Una vez acabada una ráfaga, si el registro contador no ha llegado a cero, empieza la transferencia de una nueva ráfaga.

4.5. Canales de E/S

Los canales de E/S son una mejora de los controladores de DMA. Pueden ejecutar instrucciones que leen directamente de memoria. Eso permite gestionar con más autonomía las operaciones de E/S y de esta manera se pueden controlar múltiples operaciones de E/S con dispositivos con una mínima intervención del procesador.

Estos canales todavía se pueden hacer más complejos añadiendo una memoria local propia que los convierte en procesadores específicos de E/S.

La programación de la operación de E/S por parte del procesador se realiza escribiendo en memoria los datos y las instrucciones que necesita el canal de E/S para gestionar toda la operación de E/S. La información que se especifica incluye el dispositivo al que tenemos que acceder, la operación que se debe realizar indicando el nivel de prioridad, la dirección del bloque de datos donde tenemos que leer o escribir los datos que se han de transferir, el tamaño del bloque de datos que se tienen que transferir, cómo se tiene que hacer el tratamiento de errores y cómo se ha de informar al procesador del final de la operación de E/S.

Cuando se acaba la operación de E/S, el canal de E/S informa al procesador de que se ha acabado la transferencia y de posibles errores mediante la memoria. También se puede indicar el final de la transferencia mediante interrupciones.

Las dos configuraciones básicas de canales de E/S son las siguientes:

- **Canal selector**: está diseñado para periféricos de alta velocidad de transferencia y solo permite una operación de transferencia simultánea.

- **Canal multiplexor**: está diseñado para periféricos más lentos de transferencia y puede combinar la transferencia de bloques de datos de diferentes dispositivos.

Las ventajas principales de los canales de E/S respecto a los controladores de E/S son las siguientes:

- Permiten controlar operaciones de E/S simultáneas.

- Se pueden programar múltiples operaciones de E/S sobre diferentes dispositivos o secuencias de operaciones sobre el mismo dispositivo, mientras el canal de E/S efectúa otras operaciones de E/S.

5. Comparación de las técnicas de E/S

Para analizar las prestaciones de un sistema de E/S, compararemos las diferentes técnicas de E/S estudiadas y así profundizaremos un poco más en el estudio del funcionamiento de estas técnicas, y también definiremos los parámetros básicos de una transferencia de E/S para determinar la dedicación del procesador en cada una de las técnicas de E/S.

Para realizar el análisis de las prestaciones, consideramos que hacemos la transferencia de un bloque de datos entre un periférico y el computador.

$$N_{datos} = m_{bloque} / m_{dato}$$

Donde

- m_{dato}: tamaño de un dato expresado en bytes. Entendemos *dato* como la unidad básica de transferencia.
- m_{bloque}: tamaño del bloque de datos que queremos transferir entre el computador y el periférico expresado en bytes.
- N_{datos}: número de datos que forman el bloque que queremos transferir. Nos indica también el número de transferencias que se tienen que hacer.

Periférico

Primero hay que analizar cómo condiciona la transferencia de E/S el periférico. Para hacerlo definimos los parámetros siguientes:

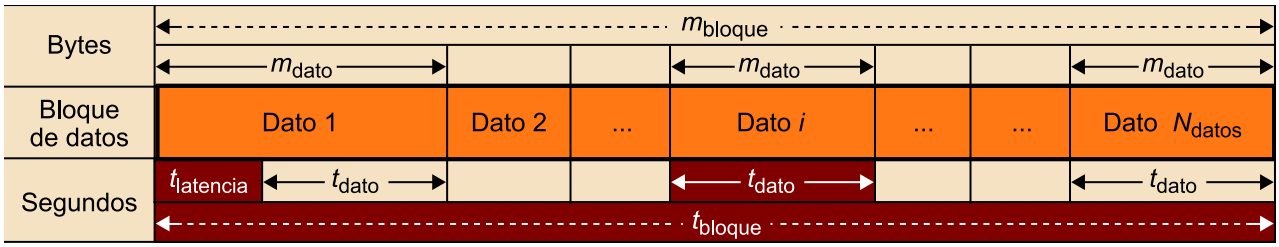
- v_{transf}: velocidad de transferencia media del periférico expresada en bytes por segundo.
- t_{dato}: tiempo de transferencia de un dato (entre el periférico y el módulo de E/S) expresado en segundos.

$$t_{dato} = m_{dato} / v_{transf}$$

- $t_{latencia}$: tiempo de latencia, tiempo que necesita el periférico para iniciar la primera transferencia expresado en segundos. En algunos periféricos este tiempo puede ser significativo.

- t_{bloque}: tiempo de transferencia de los datos de todo el bloque (entre el periférico y el módulo de E/S) expresado en segundos.

$$t_{\text{bloque}} = t_{\text{latencia}} + (N_{\text{datos}} \cdot t_{\text{dato}})$$



Definimos a continuación los parámetros básicos que determinan de alguna manera la transferencia de datos en las diferentes técnicas de E/S.

Procesador

- t_{inst}: tiempo medio de ejecución de una instrucción expresado en segundos.

Bus del sistema y memoria

- $t_{\text{cesión}}$: tiempo necesario para que el procesador haga la cesión del bus expresado en segundos.
- t_{recup}: tiempo necesario para que el procesador haga la recuperación del bus expresado en segundos.
- t_{mem}: tiempo medio para hacer una lectura o una escritura en la memoria principal mediante el bus del sistema expresado en segundos.

Transferencia de E/S

Tal como hemos definido anteriormente en una transferencia de E/S, la programación y la finalización de la transferencia son pasos comunes a todas las técnicas de E/S y son responsabilidad del procesador. Hay que tener presente que en E/S por DMA la programación de la transferencia es estrictamente necesaria y tiene un peso más importante que en el resto de las técnicas.

- t_{prog}: tiempo necesario para hacer la programación de la transferencia de E/S expresado en segundos.

- t_{final}: tiempo necesario para hacer la finalización de la transferencia de E/S expresado en segundos.

Estos tiempos se pueden calcular a partir del número de instrucciones que hay que ejecutar para hacer la programación (N_{prog}) o la finalización de la transferencia (N_{final}), multiplicando el número de instrucciones necesarias por el tiempo medio de ejecución de una instrucción (t_{inst}).

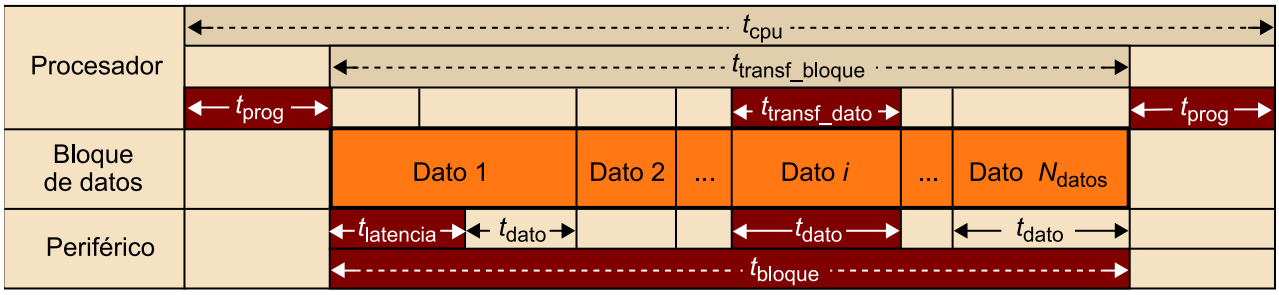
- t_{transf_dato}: tiempo, expresado en segundos, que el procesador está ocupado durante la transferencia de un dato con el módulo de E/S o parado mientras el controlador de DMA hace la transferencia de un dato.

- t_{transf_bloque}: tiempo, expresado en segundos, que el procesador está ocupado durante la transferencia de datos de todo el bloque con el módulo de E/S o parado mientras el DMA hace la transferencia de datos de todo el bloque.

$$t_{transf\_bloque} = N_{datos} \cdot t_{transf\_dato}$$

- t_{cpu}: tiempo que el procesador está ocupado o parado durante la transferencia de E/S expresado en segundos.

$$t_{cpu} = t_{prog} + t_{transf\_bloque} + t_{final}$$



Hasta ahora hemos analizado los parámetros básicos que condicionan los pasos principales en una transferencia de E/S. A continuación, veremos cómo afectan las diferentes técnicas de E/S al tiempo que el procesador está ocupado o parado (t_{transf_bloque} y t_{cpu}).

E/S programada

Cuando utilizamos esta técnica el procesador está dedicado el cien por cien del tiempo a esta tarea; por lo tanto, no tiene tiempo para ejecutar otras instrucciones:

$$t_{transf\_dato} = t_{dato}$$
$$t_{transf\_bloque} = t_{bloque}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloque} + t_{final}$$

Cabe señalar que estos tiempos no dependen del número de instrucciones del bucle de sincronización, ni del código para hacer la transferencia del dato, sino que solo dependen de la velocidad de transferencia del periférico.

E/S por interrupciones

Cuando utilizamos E/S por interrupciones, la sincronización es responsabilidad del módulo de E/S y el procesador solo es responsable de la operación de transferencia de los datos. La transferencia se realiza cuando llega la petición de interrupción (INT) del módulo de E/S y el procesador ejecuta la RSI para atenderla.

Para determinar el tiempo que dedica el procesador a atender la transferencia de un dato definimos los parámetros siguientes:

- t_{rec_int}: tiempo que pasa desde que el procesador reconoce que hay una petición de interrupción hasta que se puede iniciar la ejecución de la RSI expresado en segundos. Este tiempo incluye el reconocimiento de la interrupción y la salvaguarda del estado del procesador.

- t_{rsi}: tiempo de ejecución de la RSI expresado en segundos. Este tiempo incluye inhibir las interrupciones si es necesario, guardar en la pila los registros que se pueden modificar, acceder al módulo de E/S para hacer la transferencia del dato, restaurar de la pila los registros, volver a habilitar las interrupciones y hacer el retorno de interrupción.

- N_{rsi}: número de instrucciones de la RSI.

Tanto el tiempo de reconocimiento de la interrupción y salvaguarda del estado (t_{rec_int}) como el tiempo de ejecución de la RSI (t_{rsi}) dependen de diferentes factores que hay que tener en cuenta si se tienen que calcular.

De manera general decimos que:

$$t_{rsi} = N_{rsi} \cdot t_{inst}$$

El tiempo total que el procesador está ocupado durante toda la transferencia de E/S se puede calcular de la manera siguiente:

$$t_{transf\_dato} = t_{rec\_int} + t_{rsi}t$$
$$_{transf\_bloque} = N_{datos} \cdot t_{transf\_dato}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloque} + t_{final}$$

E/S por DMA

En esta técnica de E/S se descarga el procesador de la responsabilidad de llevar a cabo la transferencia de los datos. Esta responsabilidad se deja al controlador de DMA. Este controlador hace la transferencia de los datos, pero para hacerlo tiene que utilizar el bus del sistema con la técnica del robo de ciclo.

Veamos cuánto tiempo está ocupado o parado el procesador mientras dura la transferencia de los datos. El procesador está ocupado durante la programación y la finalización de la transferencia y está parado mientras el controlador de DMA utiliza el bus del sistema para hacer la transferencia de los datos. Para hacerlo utilizamos los parámetros siguientes:

$$t_{transf\_dato} = t_{cesión} + t_{mem} + t_{recup}$$
$$t_{transf\_bloque} = N_{datos} \cdot t_{transf\_dato}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloque} + t_{final}$$

E/S por DMA en modo ráfaga

En este caso, se considera que la unidad de transferencia del DMA está formada por un conjunto de datos. Cada vez que se solicita el bus, en lugar de transferir un único dato, se transfiere un conjunto de datos. Cada transferencia de un conjunto de datos la denominamos *ráfaga*. Esto se puede hacer si el controlador de DMA dispone de una memoria intermedia (*buffer*) para almacenar el conjunto de datos que transferimos en cada ráfaga.

- $N_{ráfaga}$: número de datos que forman una ráfaga.

Si sabemos el tamaño de la memoria intermedia, podemos obtener el número de datos de la ráfaga de la manera siguiente:

- m_{buffer}: tamaño de la memoria intermedia de datos expresada en bytes.

$$N_{ráfaga} = m_{buffer} / m_{dato}$$

El tiempo que el procesador está ocupado o parado durante la transferencia se puede calcular de la manera siguiente:

$$t_{transf\_ráfaga} = t_{cesión} + (N_{ráfaga} \cdot t_{mem}) + t_{recup}$$
$$t_{transf\_bloque} = (N_{datos} / N_{ráfaga}) \cdot t_{transf\_ráfaga}$$
$$t_{cpu} = t_{prog} + t_{transf\_bloque} + t_{final}$$

Ocupación del procesador

Veamos cuál es la proporción de tiempo que el procesador dedica a la transferencia de E/S respecto al tiempo que tarda el periférico en hacer toda la transferencia, esto es, el porcentaje de ocupación del procesador.

Como el periférico, salvo casos excepcionales, es más lento que el procesador, es el periférico quien determina el tiempo necesario para hacer la transferencia de un bloque de datos.

$$t_{transf\_bloque} < t_{bloque}$$

Considerando toda la transferencia de E/S:

$$\%_{ocupación} = (t_{cpu} \cdot 100) / t_{bloque}$$

- t_{disp}: tiempo que el procesador está libre para hacer otras tareas durante la transferencia de E/S.

$$t_{disp} = t_{bloque} - t_{cpu}$$

Ejemplo

Hacemos una comparación de las tres técnicas de E/S utilizando datos concretos para transferir un bloque de datos.

Definimos primero los parámetros básicos de esta transferencia:

m_{dato}	tamaño del dato (bus de datos y registros)	4 Bytes
m_{bloque}	tamaño del bloque que se quiere transferir	1 MByte

Procesador:

t_{inst}	tiempo de ejecución de las instrucciones	16 ns
	Frecuencia del reloj del procesador	125 MHz
	Ciclos de reloj para ejecutar una instrucción	2

Bus del sistema y memoria:

$t_{cesión}$	tiempo para ceder el bus	2 ns
t_{recup}	tiempo para liberar el bus	2 ns
t_{mem}	tiempo de lectura/escritura en memoria	4 ns

Periférico:

Recordad

Submúltiplos de la unidad de tiempo (segundos):

10^{-3}: mili (m)

10^{-6}: micro (m)

10^{-9}: nano (n)

10^{-12}: pico (p)

Múltiplos de las unidades de datos (bits o bytes).

2^{10}: kilo (K)

2^{20}: mega (M)

2^{30}: giga (G)

2^{40}: tera (T)

Frecuencia

La frecuencia nos indica el número de ciclos de reloj por segundo. Calculando la inversa de la frecuencia obtenemos el tiempo de un ciclo de reloj del procesador.

t_{ciclo} = 1/frecuencia

v_{transf}	velocidad de transferencia	10 MBytes/s
$t_{latencia}$	latencia	7 ms

Técnicas de E/S

E/S programada:

N_{prog}	programar la transferencia	2 instrucciones
N_{final}	finalizar la transferencia	0 instrucciones

E/S por interrupciones:

N_{prog}	programar la transferencia	5 instrucciones
N_{final}	finalizar la transferencia	1 instrucción
t_{rec_int}	tiempo de reconocimiento de la interrupción	48 ns
N_{rsi}	número de instrucciones de la RSI:	12 instrucciones

E/S por DMA:

N_{prog}	programar la transferencia	15 instrucciones
N_{final}	finalizar la transferencia	15 instrucciones
m_{buffer}	tamaño de la memoria interna en modo ráfaga	16 Bytes

Calculamos el número de datos que forman el bloque que queremos transferir, que nos indica también el número de transferencias que se tienen que hacer.

$$N_{datos} = m_{bloque} / m_{dato} = 2^{20}/4$$

Calculamos el tiempo de transferencia de un dato y a partir de esta el tiempo de transferir un bloque.

v_{transf}	10 MBytes/s	$10 \cdot 2^{20}$ Bytes/s
t_{dato}	m_{dato} / v_{transf}	$4 / 10 \cdot 2^{20}$ s
$t_{latencia}$	7 ms	0,007 s
t_{bloque}	$t_{latencia} + (N_{datos} \cdot t_{dato})$	$0,007 + (2^{20}/4) \cdot (4/10 \cdot 2^{20}) = 0,007 + 0,1 = 0,107$ s

Ahora analizamos los tiempos de transferencia para cada una de las técnicas de E/S.

E/S programada:

t_{prog}	$N_{prog} \cdot t_{inst}$	$2 \cdot 16$ ns = 32 ns
t_{final}	$N_{final} \cdot t_{inst}$	$0 \cdot 16$ ns = 0 ns
t_{transf_dato}	t_{dato}	$4 / 10 \cdot 2^{20}$ B
t_{transf_bloque}	t_{bloque} = 0,107 s	107.000.000 ns
t_{cpu}	32 ns + 107.000.000 ns + 0 ns	107.000.032 ns = 107 ms

E/S por interrupciones:

t_{prog}	$N_{prog} \cdot t_{inst}$	$5 \cdot 16$ ns = 80 ns
t_{final}	$N_{final} \cdot t_{inst}$	$1 \cdot 16$ ns = 16 ns
t_{rsi}	$N_{rsi} \cdot t_{inst}$	$12 \cdot 16$ ns = 192 ns
t_{transf_dato}	$t_{rec\_int} + t_{rsi}$	48 ns + 192 ns = 240 ns
t_{transf_bloque}	$N_{dades} \cdot t_{transf\_dada}$	$(2^{20} / 4) \cdot 240$ ns = $2^{20} \cdot 60$ ns = 62.914.560 ns
t_{cpu}	$t_{prog} + t_{transf\_bloc} + t_{final}$	80 ns + 62.914.560 ns + 16 ns = 62.914.656 ns = 62,9 ms

E/S por DMA:

t_{prog}	$N_{prog} \cdot t_{inst}$	$15 \cdot 16$ ns = 240 ns
t_{final}	$N_{final} \cdot t_{inst}$	$15 \cdot 16$ ns = 240 ns
t_{transf_dato}	$t_{cesión} + t_{mem} + t_{recup}$	2 ns + 4 ns + 2 ns = 8 ns
t_{transf_bloque}	$N_{datos} \cdot t_{transf\_dato}$	$(2^{20} / 4) \cdot 8$ ns = $2 \cdot 2^{20}$ ns
t_{cpu}	$t_{prog} + t_{transf\_bloque} + t_{final}$	240 ns + $2 \cdot 2^{20}$ ns + 240 ns = 2.097.632 ns = 2,09 ms

E/S por DMA en modo ráfaga:

$N_{ráfaga}$	m_{buffer} / m_{dato}	16 Bytes / 4 Bytes = 4
$t_{transf\_ráfaga}$	$t_{cesión} + (N_{ráfaga} \cdot t_{mem}) + t_{recup}$	2 ns + (4 · 4 ns) + 2 ns = 20 ns
t_{transf_bloque}	$(N_{datos} / N_{ráfaga}) \cdot t_{transf\_ráfaga}$	$((2^{20} / 4) / 4) \cdot 20$ ns = $2^{16} \cdot 20$ ns = 1.310.720 ns
t_{cpu}	$t_{prog} + t_{transf\_bloque} + t_{final}$	240 ns + 1.310.720 ns + 240 ns = 1.311.200 ns = 1,31 ms

Comparamos el tiempo de ocupación del procesador en cada una de las técnicas de E/S.

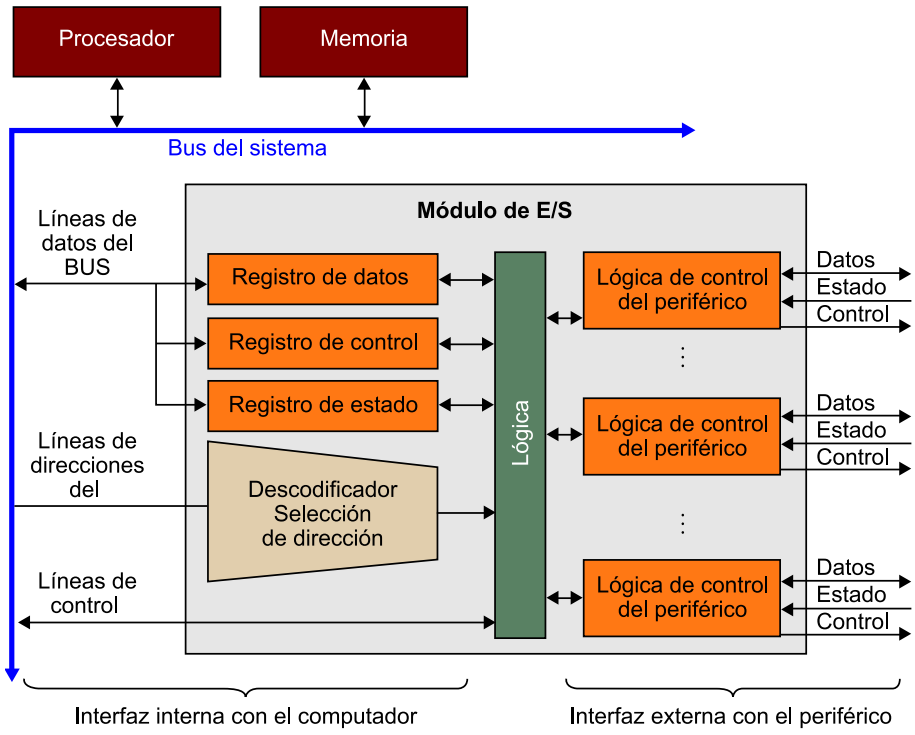
Periférico:

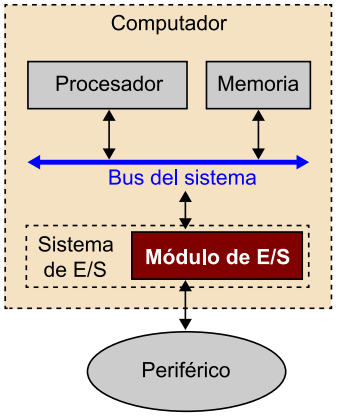
t_{bloque}	107 ms
$\%_{ocupación}$	$(t_{cpu} \cdot 100) / t_{bloque}$
t_{disp}	$t_{bloque} - t_{cpu}$

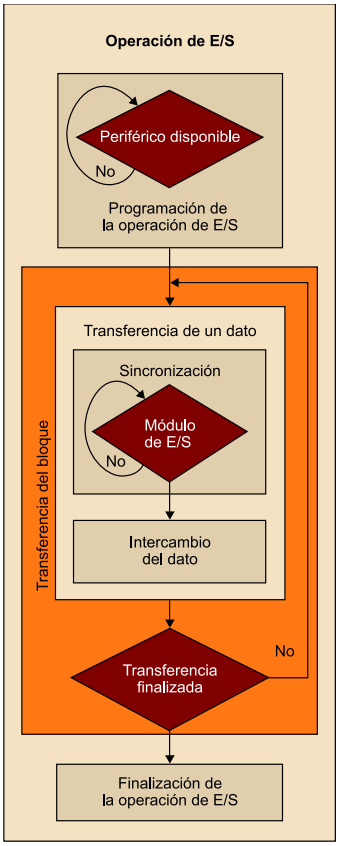
	E/S programada	**E/S por interrupciones**	**E/S por DMA**	**E/S por DMA en modo ráfaga**
t_{cpu}	107 ms	62,9 ms	2,09 ms	1,31 ms
$\%_{ocupación}$	100%	58,78%	1,95%	1,22%
t_{disp}	0 ms	44,1 ms	104,91 ms	105,69 ms

Resumen

En este módulo se han explicado en primer lugar los aspectos básicos del sistema de E/S de un computador. La estructura del sistema de E/S está formada por los periféricos, los módulos de E/S y los sistemas de interconexión externos como elementos principales.







A continuación se han descrito las fases que componen una operación básica de E/S.

1) Programación de la operación de E/S.

2) Transferencia de datos.

3) Finalización de la operación de E/S.

Para hacer la transferencia de datos se han descrito las principales técnicas de E/S:

- E/S programada.
- E/S por interrupciones.
- E/S por DMA.

Para cada una de estas técnicas se ha explicado el hardware necesario, cómo funciona la transferencia de un dato individual y de un bloque entre el computador y un periférico, y también cómo se debe gestionar la transferencia cuando tenemos más de un periférico conectado al computador y se han de gestionar simultáneamente múltiples transferencias, en cuyo caso los problemas principales que hay que resolver son la identificación del periférico con el que se ha de realizar la transferencia y la gestión de prioridades.

Finalmente, se ha hecho una comparación de las diferentes técnicas de E/S para analizar las prestaciones de un sistema de E/S y de esta manera profundizar un poco más en el funcionamiento de estas técnicas.