TEMA 6: COMUNICACIÓN CON EL EXTERIOR

Los sistemas de entrada/salida constituyen uno de los temas de mayor interés y complejidad en el diseño y utilización de los computadores. Abarca desde los cronogramas y mecanismos de sincronismo de las señales empleadas en la interconexión de los diferentes dispositivos periféricos, hasta los procedimientos empleados por el sistema operativo para ejecutar las diversas solicitudes de entrada/salida, pasando por la estructura de los elementos de interconexión y el desarrollo de los programas de accionamiento de los mismos.

En este tema se pretende estudiar los mecanismos básicos de transmisión de información entre CPU y periféricos, y en la organización de las operaciones de E/S. Además se repasan las técnicas para la gestión de prioridades cuando más de un peticionario solicita un mismo recurso y las distintas formas de generar una interrupción. También se repasarán conceptos relacionados con los canales de E/S atendiendo a su clasificación, la intervención del Sistema Operativo en una operación completa de E/S y por último se verá un ejemplo de diseño de un sistema de E/S.

6.1. INTRODUCCIÓN. LAS UNIDADES DE E/S Y LOS DISPOSITIVOS PERIFÉRICOS

Hasta ahora se ha estudiado el sistema computador como una máquina que puede ejecutar un programa inicialmente registrado en la Memoria Principal, sobre datos registrados en la misma Memoria Principal y almacenar los resultados en dicha memoria a medida que se obtienen. Ahora es preciso dotarla de medios para comunicarse con el exterior: este es el papel reservado a las *Unidades de Entrada/Salida* y a los *Dispositivos Periféricos*.

Existen dos grandes clases de *Unidades Periféricas*:

a) Las *Unidades de Comunicación* (impresora, pantalla, teclado, etc.) que permiten el diálogo con el exterior.

b) Las *Unidades de Almacenamiento* (discos, cintas magnéticas, etc.), cuyas capacidades de almacenamiento son muy superiores a la de la Memoria Principal.

Las Unidades Periféricas se conectan a la Unidad Central de Proceso o directamente a la Memoria Principal, a través de unidades especializadas en la gestión de las transferencias de información. Estas unidades de intercambio de información con el exterior se llaman *Unidades de E/S*. La Unidad de Control se comunica con estas Unidades de E/S mediante instrucciones que pueden ser específicas de E/S o generales de transferencia, dependiendo del sistema computador.

En este tema se estudiarán las Unidades de entrada/salida, dejando para un curso posterior el estudio de los dispositivos periféricos.

6.2. UNIDADES DE ENTRADA/SALIDA

6.2.1. Objetivos y necesidades de las unidades de E/S

El objetivo de las Unidades de Entrada/Salida (E/S) también llamadas de intercambio de información, es realizar la conexión y la adaptación de la Unidad Central de Proceso (CPU) del computador con los dispositivos periféricos. Esta conexión presenta unas características muy especiales como son las siguientes:

- Los periféricos tienen unas velocidades de transmisión que abarcan desde unos pocos de bytes por segundo, hasta varios millones de bytes por segundo. Sin embargo esta velocidad es sensiblemente inferior a la de la CPU (10 MIPS a 10 GIPS).
- Los periféricos suelen tener un ancho de palabra de un byte, lo que no coincide con el ancho de palabra de la mayoría de los computadores.

- Algunos periféricos son de lectura, otros de escritura y, por último, otros son simultáneamente de lectura y escritura.

El problema del intercambio de información entre la CPU y los periféricos es relativamente complejo. Por un lado, hay que establecer mecanismos que permitan transmitir la información, mecanismos que deben considerar aspectos tales como el direccionamiento o selección del periférico, la forma de establecer el camino para el envío de datos, la posible conversión serie-paralelo, la posible conversión de códigos, etc. Por otro lado hay que establecer un mecanismo de control que determine el origen y el destino de la información, la cantidad de ella a transmitir, los códigos de protección, etc. Estos mecanismos se reparten entre el controlador del periférico, la unidad de control de la CPU y los programas de entrada/salida que, normalmente, forman parte del sistema operativo.

Por todo ello hay que diferenciar entre lo que es la transferencia elemental de información, la transferencia de bloque y la operación de entrada/salida.

- La TRANSFERENCIA ELEMENTAL de información tiene por objeto el envío o la recepción de una única unidad de información (un byte o una palabra) ya sea ésta un dato o una palabra de control. Suele ser el resultado de la ejecución de alguna instrucción de Entrada-Salida (mapa de memoria no común), por ejemplo INPUT, OUTPUT; o de transferencia de datos (mapa de memoria común), como por ejemplo las instrucciones MOVE, STORE, LOAD.
- La TRANSFERENCIA DE BLOQUE se encarga de enviar o recibir un bloque de información, como puede ser un sector de un disco, un registro de una cinta o una trama de un dispositivo de comunicación hacia o desde posiciones consecutivas de memoria principal. Un bloque de información consiste en un conjunto de unidades de información (byte o palabra) que ocupan posiciones consecutivas.
- La OPERACIÓN DE E/S tiene por objeto el proceso global de transferencia de un conjunto de datos, garantizando que ésta se produzca correctamente. Para ello gestiona la transferencia de bloques y supervisa el estado del periférico, realiza el tratamiento de los errores y lleva la cuenta de los bloques transferidos. También se encarga de la conversión de código en el caso de que los datos estén representados en el periférico en un código diferente al requerido por el programa.

Las funciones más importantes de cada una de estas operaciones están reflejadas en la Figura 6.1.

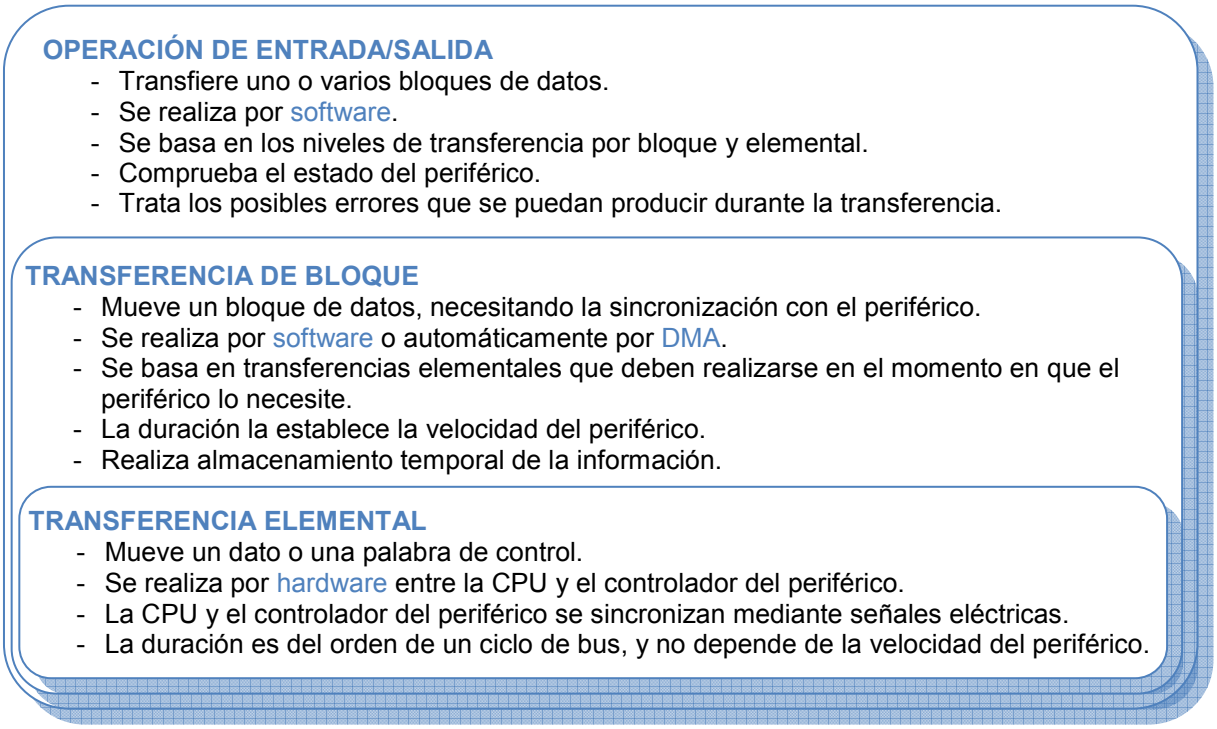

Figura 6.1. *Estructuración de los niveles de transferencia de E/S.*

6.2.2. Técnicas de transferencia

Las operaciones de E/S se basan en transferencias elementales, por tanto, es necesario conocer los procedimientos empleados para transmitir una información elemental entre el periférico y la CPU. Esta información puede ser, según sea el caso, una palabra, un byte o un bit. Existen dos alternativas para hacer una transferencia elemental:

- E/S programada, por ejecución de una instrucción.
- Por acceso directo a memoria (DMA).

6.2.2.1. E/S programada

Las instrucciones de E/S son fundamentalmente instrucciones de transferencia donde uno de los operandos implicados está almacenado en un registro del controlador del periférico y el otro operando en un registro general de la CPU. La CPU suministra información al controlador del periférico para que éste genere las señales necesarias que permiten ejecución de la transferencia. Esta información que la CPU suministra al controlador consiste en:

- Información de dirección para elegir el registro del periférico con el que se realiza la transferencia.
- Tipo de operación, por ejemplo escritura o lectura de un registro de un periférico.

- Temporización. La transferencia exige que se establezca un camino físico entre los dos elementos de almacenamiento y que se genere una señal de flanco cuando los datos estén estabilizados a la entrada del destino. Las señales de la CPU y del periférico han de tener una temporización coherente para que la transferencia se haga correctamente.

Al decodificar la instrucción de E/S, la unidad de control del computador suministra al exterior estas informaciones y envía el dato o se dispone a recibirlo, dependiendo del tipo de acceso. La comunicación se realiza mediante las siguientes señales:

- Señales de dirección para seleccionar el periférico.
- Señales de datos que permiten el intercambio de datos.
- Señales de control que permiten que el controlador del periférico y la CPU se pongan de acuerdo.

SEÑALES DE DIRECCIÓN

La información de la dirección se compone por p bits lo que permite establecer 2^p direcciones distintas, sirviendo cada una de ellas para especificar lo que se llamará una **puerta de entrada/salida**. Estas puertas de entrada/salida sirven para enviar y/o recibir información a tamaño de byte o palabra. Cada periférico empleará una o varias puertas para comunicarse con la CPU. Al conjunto de las 2^p direcciones se le llama **mapa de E/S**.

Es necesario establecer un mecanismo de decodificación que convierta la dirección en la señal de selección correspondiente. Para ello se puede emplear la decodificación centralizada o la decodificación independiente.

- La decodificación centralizada es aquella en la que un decodificador central habilita el respectivo decodificador para seleccionar una determinada puerta de entrada/salida. En la Figura 6.2 se muestra un decodificador de 6 a 64 basado en la decodificación centralizada. Está formado por 9 decodificadores de 3x8, uno de ellos actúa como decodificador central, con lo que se obtienen las 64 señales DP_i. Cada una de ellas sirve para direccionar una puerta de entrada/salida.

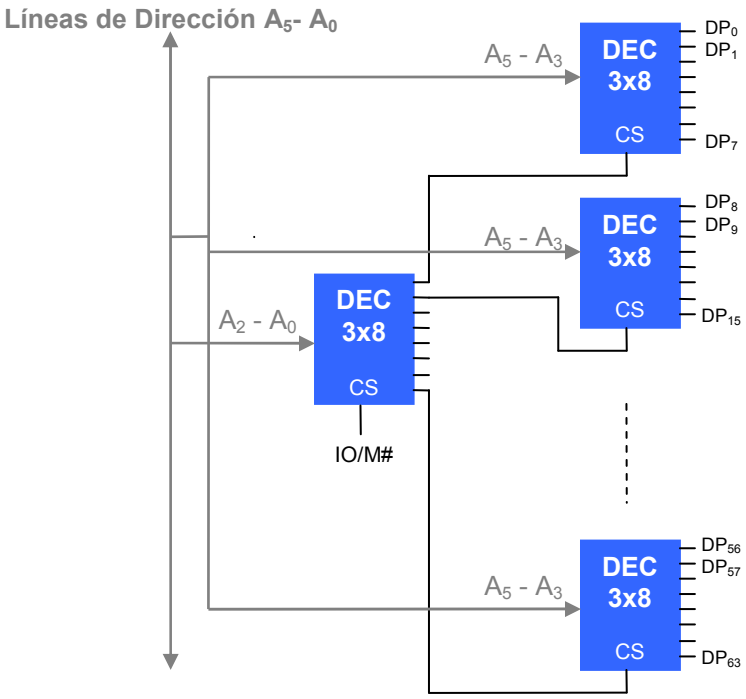

Figura 6.2. *Decodificación Centralizada de Direcciones.*

- La decodificación independiente consiste en hacer que cada puerta reconozca su propia dirección, mediante un detector de dirección particular. En la Figura 6.3 se muestra un mecanismo de decodificación basado en esta técnica. Requiere de inversores para formar la palabra de dirección deseada, y una puerta lógica que active la señal de selección de puerta DP_i en base a la combinación de líneas de dirección correspondiente.

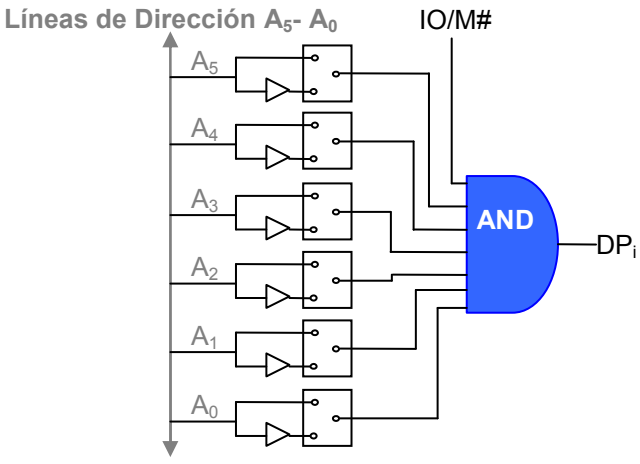

Figura 6.3. *Decodificación Independiente de Direcciones.*

En ambas figuras se ha considerado un conjunto de 6 líneas de dirección, lo que permitirá seleccionar hasta 64 (2^6) puertas diferentes. Se ha añadido la señal IO/M#, que cuando el mapa de memoria es no común, permite acceder a los dispositivos de E/S.

Cuando IO/M# está activa ('1' lógico), se pueden activar las señales de selección de puerta DP_i.

Es importante destacar que operaciones de E/S es frecuente hacer una correspondencia no biunívoca entre direcciones y puertas, con idea de simplificar la decodificación de estas direcciones. Por ejemplo, supóngase que se tiene una máquina de 8 bits de dirección de E/S a la que se quiere conectar 8 periféricos de una puerta cada uno. Las tres posibles soluciones son:

- Solución 1: Establecer una relación biunívoca entre puertas y direcciones.
- Solución 2: Asignar varias direcciones a cada puerta, pero cada dirección sólo corresponde a una puerta.
- Solución 3: Hacer corresponder a cada puerta sólo aquellas direcciones que tienen un único bit a1 y el resto a 0.

Puerta	Direcciones asignadas		
	Solución 1	Solución 2	Solución 3
1	00000000	xxxxx000	00000001
2	00000001	xxxxx001	00000010
3	00000010	xxxxx010	00000100
4	00000011	xxxxx011	00001000
5	00000100	xxxxx100	00010000
6	00000101	xxxxx101	00100000
7	00000110	xxxxx110	01000000
8	00000111	xxxxx111	10000000

Tabla 6.1. *Direcciones por cada puerta de E/S*.

SEÑALES DE DATOS

Los datos pueden transmitirse por un conjunto de líneas bidireccionales o dos conjuntos de líneas de dirección única. Las técnicas básicas de conexión son dos: Conexión a un bus común y Conexión mediante Multiplexor/Demultiplexor.

En el primer caso, conexión a un bus común, representado en la Figura 6.4, puede observarse que un buffer triestado, activado por la combinación simultánea de las señales de dirección de puerta (DP_i) y la señal de lectura (RD#), sirve para que la puerta de E/S se conecte al bus de datos y envíe una palabra a la CPU (puerta de Entrada o Lectura). Por otro lado, un registro activado por la combinación de la señal DP_i y la señal de escritura WR# permiten seleccionar los circuitos de entrada, de forma que la CPU deja un dato sobre el registro correspondiente (Puerta de Salida o Escritura).

Las dos ventajas principales de la conexión al bus común son que permite conectar en paralelo un gran número de periféricos y es fácil expandir el sistema (basta con ir incluyendo más tarjetas con los controladores correspondientes). La desventaja es que

es lenta, ya que conlleva una gestión distribuida de prioridades (ver sección 6.1.4), lo que obliga a establecer un tiempo de espera relativamente grande.

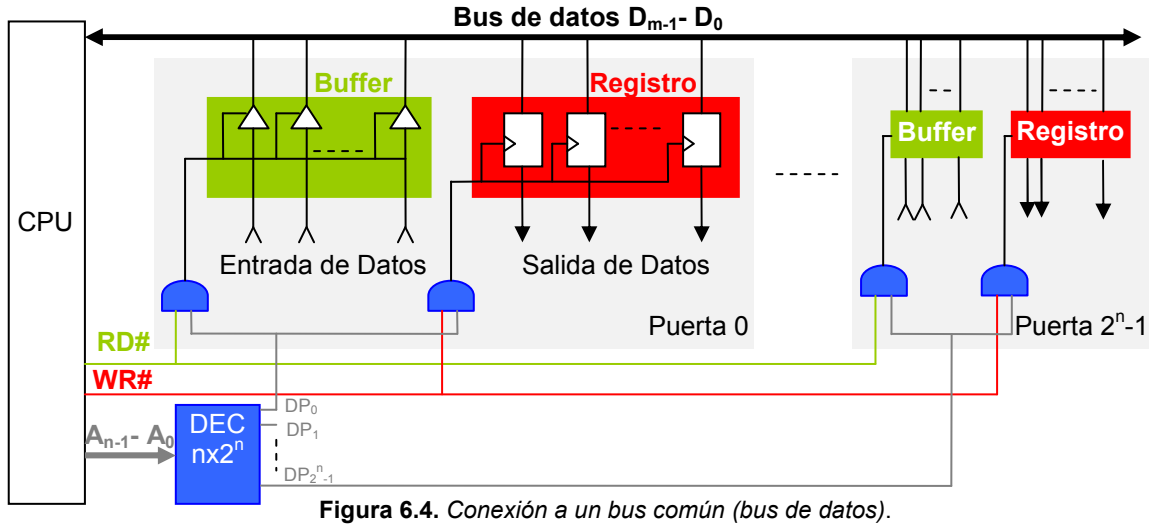

Figura 6.4. *Conexión a un bus común (bus de datos).*

En el segundo caso, en la conexión mediante Multiplexor/Demultiplexor, las puertas de E/S se conectan a las líneas de datos y de control mediante un bloque multiplexor/Demultiplexor (Figura 6.5). El multiplexor se emplea para conectar las señales de las distintas puertas con una única señal de entrada a la CPU, mientras que el demultiplexor es empleado para conectar una sola señal de salida de la CPU con las señales de las distintas puertas. El principal inconveniente de esta conexión es la expansión. Si el multiplexor/demultiplexor está completo, hay que añadir una circuitería adicional para hacer esta expansión, además exige una gran cantidad de líneas de conexión.

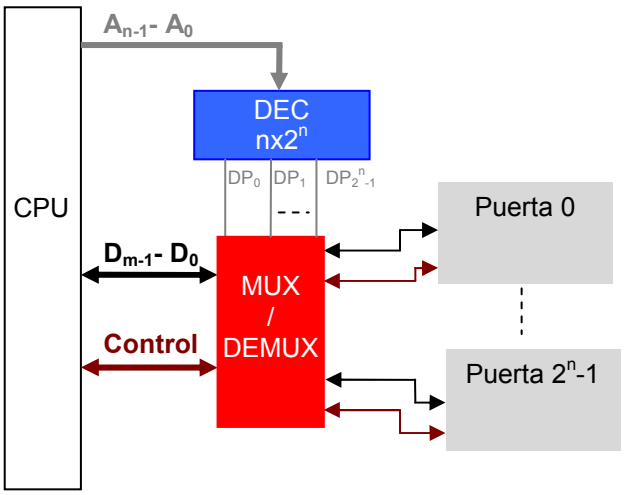

Figura 6.5. *Conexión mediante multiplexor/demultiplexor.*

SEÑALES DE CONTROL

Las señales de control deben especificar el tipo de transferencia, de entrada o de salida, y deben establecer su temporización. Existen dos métodos, denominados síncrono y asíncrono, para establecer esta temporización.

En la transferencia síncrona, la CPU establece el ritmo o temporización que debe ser seguido por el periférico. Si éste no es capaz de leer del bus o de suministrar el dato en el tiempo establecido, la transferencia fracasa. Por otro lado, la CPU no tiene constancia del éxito o fracaso de la transferencia. Podría haberse direccionado un periférico inexistente o apagado y la CPU almacenaría un dato incorrecto, puesto que no se ha puesto ningún valor en el bus.

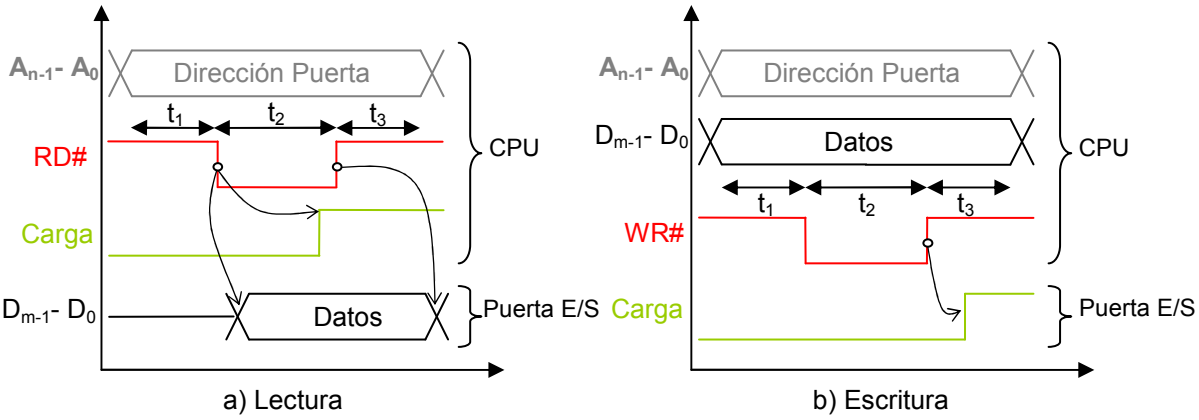

Figura 6.6. *Cronogramas de Transferencia Síncrona*.

Durante la operación de *Lectura* (Figura 6.6a), la señal RD# marca el tiempo del que dispone la puerta de E/S para dejar el dato en el bus de datos. Para esta operación, la señal RD# establece el tiempo mínimo necesario durante el que las líneas de dirección deben mantenerse estables, este tiempo corresponde a la suma $t_1 + t_2 + t_3$. La propia señal RD# se emplea para abrir el buffer triestado al bus de datos (Figura 6.4), de forma que éste contiene el dato estabilizado con el correspondiente retraso y se procede a activar la señal de carga en la CPU. En la operación de *Escritura* (Figura 6.6b), es la señal WR# la que establece el periodo durante el cual la CPU deja los datos válidos sobre el bus de datos. El propio flanco de subida de la señal WR# se emplea para activar la carga del registro de la puerta de E/S. La señal de CARGA de la puerta se genera combinando la señal WR# con la señal de dirección correspondiente DP_i (Figura 6.4), de ahí que se produzca un cierto retraso respecto al propio flanco de WR#. La vinculación de una señal con otra viene indicada sobre la Figura 6.6 mediante las flechas que unen un flanco con otro.

En el segundo de los tipos de transferencia, transferencia asíncrona, es posible conectar periféricos cuyos controladores tengan diferentes requisitos de tiempo, adaptándose la CPU, a las necesidades de cada uno. Además existe una mayor garantía de que el dato sea válido, ya que es necesario una contestación positiva por parte del periférico. El problema es que el sistema puede quedar bloqueado, esperando una señal que no llega nunca, ya sea porque el periférico no existe o porque éste no contesta. Las transferencias asíncronas pueden ser sin interbloqueo o con interbloqueo. En los dos casos existe una nueva señal de aceptación (ACP), con la que la puerta de E/S contesta a la petición de transferencia generada por la CPU.

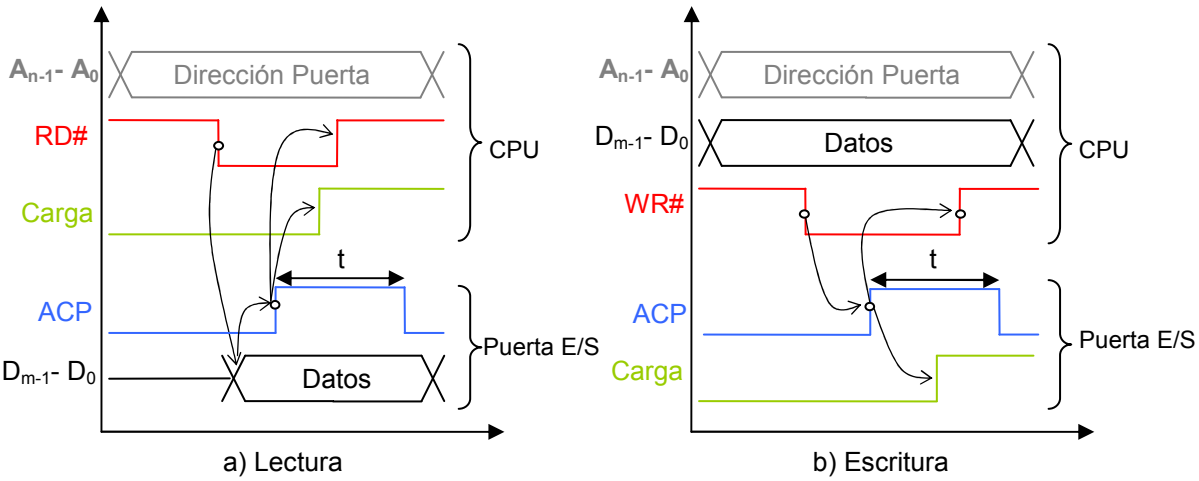

Figura 6.7. *Cronogramas de Transferencia Asíncrona Sin Interbloqueo.*

Esta señal de aceptación, en una transferencia asíncrona sin interbloqueo (Figura 6.7), establece el tiempo durante el que se debe realizar la carga de datos por parte de la CPU, cuando se trate de una operación de lectura, o la carga de datos por parte de la propia puerta cuando sea una operación de escritura. El inconveniente de la transferencia asíncrona sin interbloqueo es que parte de la temporización es fija, por lo que no hay garantía de que el periférico la respete.

En el caso de la transferencia asíncrona con interbloqueo (Figura 6.8), la transferencia entre la CPU y el controlador del periférico se hace correctamente con independencia de los tiempos que tarden en activar sus respectivas señales. Se establece un encadenamiento de las fases de la transferencia que garantizan que ésta se haga correctamente. El encadenamiento se produce mediante una secuencia obligada de eventos. El diálogo establecido entre la CPU y la puerta de E/S se denomina handshaking, que puede ser traducido como interbloqueo. Con él se garantiza que la transferencia se haga correctamente, con independencia de los tiempos que tarden la CPU y el controlador del periférico en activar sus señales.

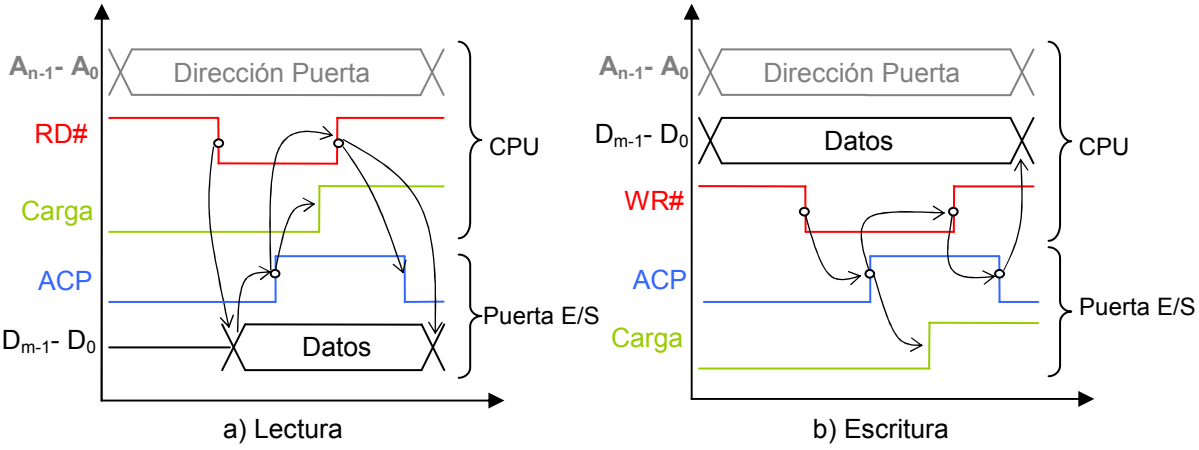

Figura 6.8. *Cronogramas de Transferencia Asíncrona Con Interbloqueo.*

En la operación de lectura por transferencia asíncrona, la puerta de E/S mantiene los datos estables el tiempo necesario para que la CPU los cargue, pero menos del tiempo máximo para no inferir en la transferencia siguiente. En la operación de escritura asíncrona, la puerta de E/S carga los datos en el intervalo existente entre el momento en que se activa la señal de aceptación y el momento en el que la CPU deshabilita los datos. La señal de aceptación se deshabilita a tiempo para que no interfiera en la transferencia siguiente. En ambos casos, operación de lectura o escritura, este intervalo de tiempo del que se habla viene establecido por la señal de aceptación en el caso de transferencia asíncrona sin interbloqueo, mientras que en el caso de la transferencia asíncrona con interbloqueo este tiempo no es fijo y deriva de la comunicación entre las señales de aceptación, carga y RD# o WR#.

6.2.2.2. E/S por acceso directo a memoria (DMA)

En el acceso directo a memoria (DMA: *Direct Memory Access*) el controlador del periférico se comunica directamente con la memoria principal del computador, sin intervención de la CPU. La transferencia elemental se realiza sin que se ejecute ninguna instrucción en la CPU, siendo el controlador del periférico el que se encarga de generar las señales de control. El DMA exige que el computador envíe al controlador del periférico un determinado tipo de información para que éste sepa lo que tiene que hacer. Esta información se denomina bloque de control y consiste en:

- Dirección en memoria principal, para generar las señales de dirección en el acceso a memoria.
- Tipo de operación: lectura o escritura.
- Número de datos a transferir.
- Dirección del periférico.

Para hacer llegar el bloque de control al controlador del periférico se dotará al mismo de una o varias puertas de E/S programada. Basta con ejecutar un programa que contenga tantas instrucciones de salida como palabras tenga el bloque de control. Una vez recibida la información, el controlador de forma independiente, realiza los necesarios ciclos de DMA. Finalizada la operación de E/S, el controlador del periférico envía una señal de interrupción a la CPU. Esto hace que este tipo de transferencia tenga sentido cuando se envíe o se reciba un bloque de datos. En este sentido, el DMA admite dos tipos de transferencia, la transferencia de bloque y la transferencia elemental. Se hablará de operación de DMA para referirnos a la transferencia de un bloque, y de transferencia por DMA, para referirnos al intercambio de una palabra.

Las principales aplicaciones de las operaciones de E/S por DMA son:

- Controladoras de discos y disquetes.
- Operaciones de descanso.
- Adquisición de datos.
- Transferencia de memoria a memoria.
- Búsqueda en memoria.
- Almacenamiento de seguridad.
- Sistema de buses paralelos.
- Conexión con fibra óptica.
- Transferencia de bloques en multiproceso y multiprograma.

La desventaja de realizar operaciones por DMA es que la CPU puede quedar inactiva (no siempre) perdiendo parcial o totalmente el control del sistema de buses, en consecuencia no hay servicio de interrupciones y abandono de refrescos.

Las operaciones de E/S por DMA pueden dar lugar a conflictos de acceso, ya que varios clientes (CPU y controladores) pueden solicitar accesos a memoria simultáneos, que es un recurso único. Por tanto es necesario disponer de un mecanismo de prioridad. Existen dos formas básicas de resolver estos conflictos:

- Mediante Memoria Multipuerta.
- Mediante Robo de Ciclo.

MEMORIA MULTIPUERTA

La memoria multipuerta permite que los periféricos dispongan de la memoria principal sin intervención de la CPU. Al estar dividida en varios bloques de memoria, permite accesos paralelos, es decir puede gestionar peticiones de acceso en paralelo siempre que estas peticiones simultáneas no sean operaciones de escritura o lectura y escritura en un mismo bloque de memoria. El mayor inconveniente de la memoria multipuerta es su

coste, puesto que además de la propia lógica de cada puerta, debe tener un dispositivo de gestión de prioridades para resolver las colisiones, es decir las peticiones simultáneas a un mismo bloque de memoria.

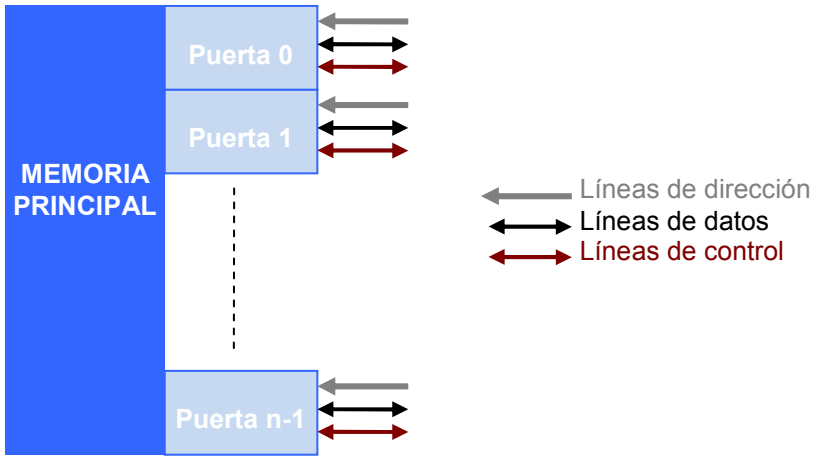

Figura 6.9. *Acceso Directo a Memoria con memoria Multipuerta*.

La información que debe definir el controlador del periférico para comunicarse con la memoria es:

- La Dirección de la Memoria Principal de la posición a la que pretende acceder.
- El Dato, si la operación es de escritura.
- El inicio de ciclo de memoria, para indicar a la memoria el comienzo de la operación.
- La orden de Lectura o Escritura.

Las señales con las que contesta la Memoria Principal son:

- El Dato, si la operación es de lectura.
- El fin de ciclo de memoria, cuando finaliza la operación.

Dado el coste que implica disponer de una memoria multipuerta, se suelen conectar a cada una de las puertas varios dispositivos periféricos, de forma que el tráfico total de información que soporten se acerque a su máximo. Para evitar los conflictos de acceso simultáneo que pueden ocurrir en una misma puerta con varios periféricos conectados, se suele emplear un mecanismo adicional de prioridad.

ROBO DE CICLO

El robo del ciclo es una forma más económica de realizar el acceso directo a memoria, puesto que la memoria va a tener una sola puerta que debe ser compartida por la CPU y por los periféricos. La transferencia exige que la CPU y el controlador del periférico se pongan de acuerdo para obtener el control de los buses y las señales de control de la única puerta de memoria.

Los pasos requeridos son los siguientes:

- El controlador del periférico solicita a la CPU el acceso a la Memoria Principal mediante la señal de control BREQ (Bus Request).
- La CPU contesta mediante la señal BACK (Bus Acknowledge) concediendo la solicitud. Al mismo tiempo pone en estado de alta impedancia los buses y señales de control de acceso a la memoria. La CPU debe esperar a terminar la fase de instrucción en curso para conceder el robo de ciclo solicitado.
- El controlador del periférico realiza el acceso a memoria, activando las señales de control y generando las direcciones correspondientes.
- Cuando el controlador del periférico ha terminado, devuelve a la CPU el control de los buses y demás señales desactivando la señal BREQ.
- La CPU desactiva la señal BACK y recupera el control de los buses.
- Si en el acceso a memoria, se ha finalizado la transferencia de un bloque completo de información, el controlador envía a la CPU la señal INT, generando una interrupción, para la avisarla de que ha finalizado la transferencia del bloque completo.
- A continuación, y una vez que la CPU finaliza la ejecución de la instrucción que se encontraba realizando cuando se produjo el robo de ciclo, ésta responde con la señal INTA para comunicar la aceptación de la interrupción provocada.

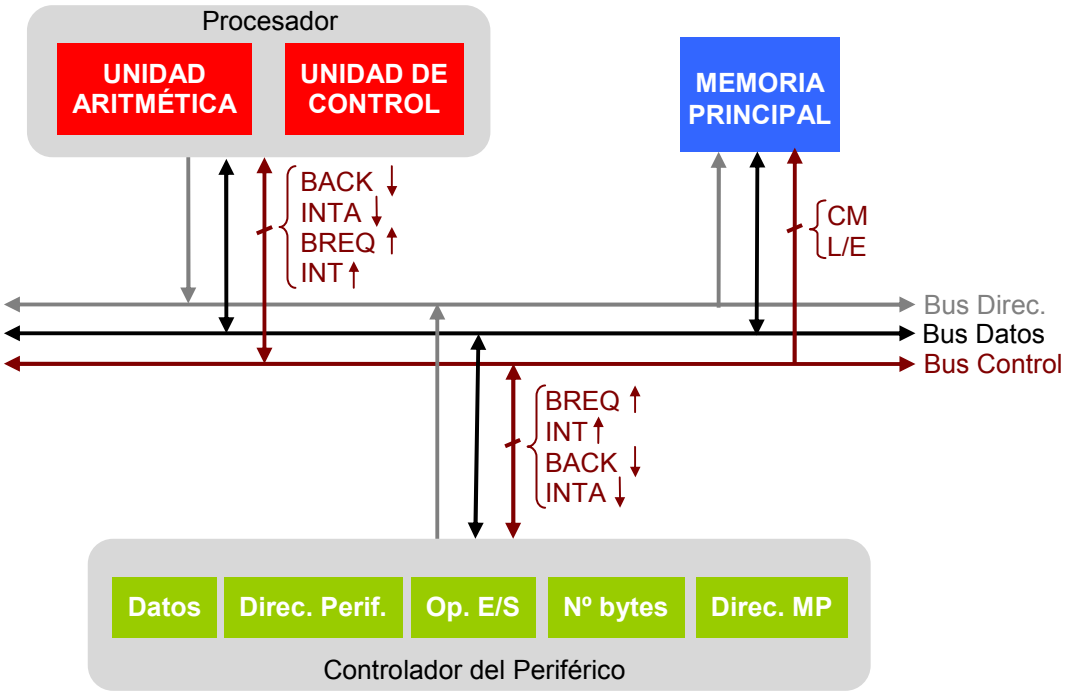

Figura 6.10. *Acceso Directo a Memoria por Robo de Ciclo.*

Las señales que el controlador del periférico ha de enviar a la CPU son las siguientes.

- BREQ#, para solicitar ciclo.

- INT, señal de solicitud de interrupción.

Las señales que el controlador del periférico ha de enviar a la Memoria son:

- CM, señal de inicio de ciclo de memoria.
- L, señal de nivel que indica ciclo de lectura.
- E, señal de nivel que indica ciclo de escritura.

Por otro lado, debe interpretar las señales generadas por la CPU:

- BACK, de concesión de los buses.
- INTA, señal de aceptación de la interrupción.

A su vez, el controlador debe ser capaz de acceder a sus propios registros como son:

- El registro de datos, para cargar/dejar su contenido sobre el bus de datos, según sea una operación de lectura/escritura de memoria.
- El registro que contiene la dirección del periférico implicado en la operación de E/S.
- El registro que indica si la operación es de Entrada o de Salida.
- El registro que indica el número de bytes que se van a transferir durante la operación de E/S. Después de cada acceso a memoria, el controlador decrementa este registro.
- El registro de direcciones, que contiene la dirección de memoria a la que se va a acceder. Además el controlador debe ser capaz de incrementar este registro después de cada acceso a una posición de memoria. En una transferencia de bloque, la información de memoria está almacenada en posiciones consecutivas.

Generalmente el periférico que solicita el robo del ciclo es más lento que la CPU, por lo que va intercalando, esporádicamente, robos de ciclo para hacer transferencias individuales. Sin embargo, puede darse el caso de que una vez obtenido el ciclo, el controlador realice múltiples transferencias, robos de ciclo de tipo ráfaga, antes de desactivar la señal de petición.

Los modos de operación del Controlador de Acceso Directo a Memoria (DMAC) son tres.

- Modo BYTE. El DMAC transfiere un solo byte mientras que el periférico está activo. La CPU toma el control cuando se ha transferido el byte.
- Modo demanda (BURST). El DMA transfiere varios bytes hasta que la CPU retoma el control.
- Modo continuo (BLOCK). El DMA mantiene ocupado los buses todo el tiempo necesario hasta transferir el bloque completo.

El robo del ciclo hace que la duración de la ejecución de la instrucción no sea fija, ya que ésta dependerá de si se realizan robos de ciclo. Este efecto debe tenerse en cuenta si se quieren hacer consideraciones temporales en función de los tiempos de ejecución de las instrucciones.

La sincronización de la transferencia viene reflejada en la Figura 6.11.

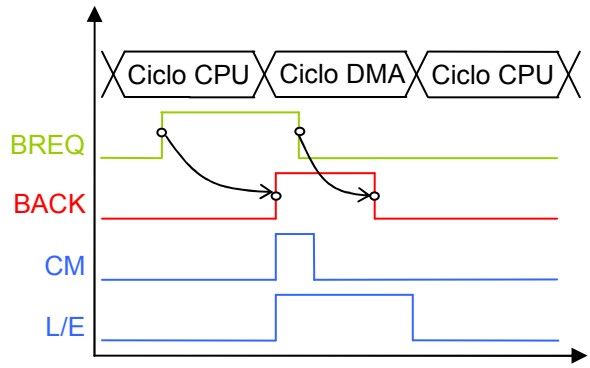

Figura 6.11. *Cronograma Acceso Directo a Memoria por Robo de Ciclo*.

El controlador del periférico activa la señal BREQ, solicitando el robo del ciclo, a lo que contesta la CPU activando la señal BACK. Esta señal la utiliza el controlador para que el registro de direcciones ponga en el bus de direcciones la dirección correspondiente a la posición de memoria deseada. Finalizado el acceso, el controlador del periférico desactiva BREQ y la CPU desactiva BACK. El controlador de DMA debe conocer las direcciones de origen y destino, el sentido de la transferencia y el número de bytes a transmitir.

6.2.3. Señales de control y estado de los periféricos

La mayor parte de los periféricos disponen de un conjunto de señales de control y estado, que permiten conocer su situación interna y simplificar su gobierno.

Señales de control:

- Encender o apagar.
- Seleccionar opciones.
- Saltar de páginas en impresoras.
- Buscar marca en cintas magnéticas.

Señales de estado:

- Periférico encendido o apagado.
- Periférico en funcionamiento.
- Periférico operativo o no.
- Error de operación.
- Error de paridad.

El tratamiento de las señales de control y estado forma parte de las operaciones de E/S.

Para que la CPU pueda acceder a las señales de estado y pueda enviar las señales de control, existen dos procedimientos:

- Manejar las informaciones como si fuesen datos y transmitirlas mediante E/S programadas. Supone tener periféricos con puertas distintas para datos y para control y estado.
- Mediante diferenciación de un mapa de direcciones de control y estado, añadiendo una señal que diferencia entre E/S de datos y E/S de control. Exige instrucciones especiales como TEST (para leer una puerta de control) o CONTROL (para escribir en una puerta de control).

6.2.4. Generalidades sobre prioridades

El problema de prioridades se presenta siempre que dos o más elementos tienen que compartir un mismo recurso, pudiendo existir peticiones simultáneas. Por tanto hay que establecer un mecanismo para determinar cuál de los dispositivos es atendido primero, haciendo esperar a los demás.

Se llamará PETICIONARIOS a los elementos que comparten el recurso común, y FASE DE SERVICIO al tiempo empleado en atender a cada uno de ellos.

El problema se plantea en situaciones tales como:

- Varios periféricos solicitan al mismo tiempo un robo de ciclo. En este caso el robo de ciclo es la fase de servicio.
- Varios periféricos solicitan al mismo tiempo una interrupción.
- Varios periféricos conectados a un mismo bus solicitan de forma simultánea un ciclo de bus.
- Varias puertas de memoria multipuerta tratan de acceder simultáneamente al mismo módulo de memoria.

Existen dos formas de abordar este problema:

- GESTIÓN DISTRIBUIDA DE PRIORIDADES. En este caso los peticionarios han de ponerse de acuerdo entre sí para determinar quién se queda con el recurso.
- GESTIÓN CENTRALIZADA DE PRIORIDADES. En este caso existe un elemento maestro que se encarga de decidir la cesión del recurso.

La política de gestión de prioridades debe definir dos aspectos fundamentales:

- El procedimiento empleado para seleccionar el peticionario. Basado en métodos como:

- Prioridad fija.
- Petición más antigua.
- Selección aleatoria.
- Clases de prioridades fijas.

- Determinar si puede ser suspendida o no la fase de servicio a un peticionario, por la llegada de una petición más prioritaria. La suspensión de un servicio para atender a otro produce un anidamiento de los servicios, siendo el nivel de anidamiento el número de servicios suspendidos para atender al más prioritario.

En general, cuando las fases de servicio son de corta duración, no interesa suspenderlas aunque llegue otra petición más prioritaria. Por otro lado, los mecanismos de prioridad fija normalmente se asignan a las operaciones de E/S cuando el tiempo de servicio para atender a la petición es el mismo e independiente del peticionario (funcionamiento síncrono). En caso contrario habrá que establecer un mecanismo de interbloqueo, que permita a cada peticionario mantener el recurso el tiempo necesario, activando la correspondiente señal.

6.2.4.1. Gestión Distribuida de Prioridades

Este método es recomendado cuando los distintos peticionarios están conectados al recurso mediante un bus. Las técnicas que emplean gestión distribuida de prioridades son:

6.2.4.1.1. Encadenamiento o Daisy-Chain. Exige disponer de un elemento maestro que decide cuándo se concede el recurso, aunque son los peticionarios quienes determinan cuál de ellos se queda con la fase de servicio.

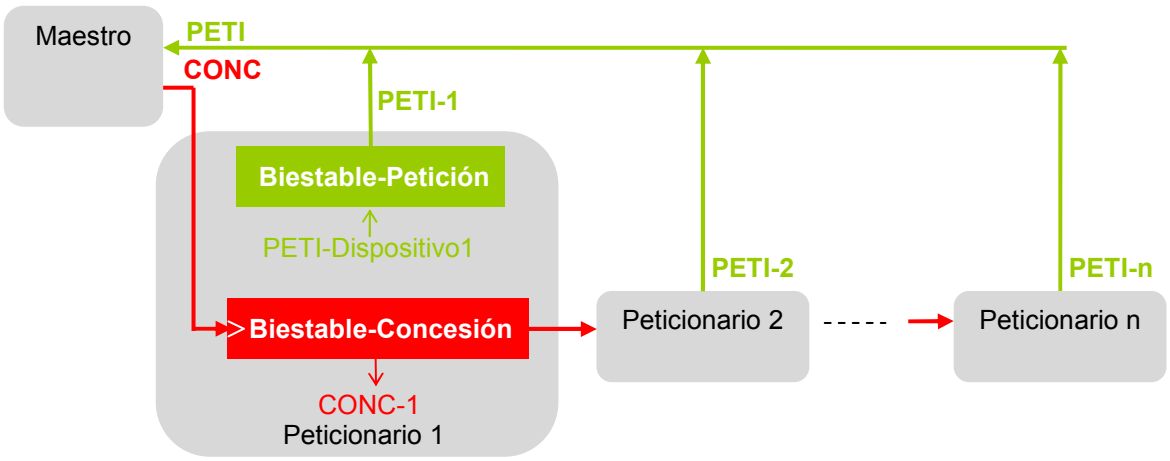

Figura 6.12. *Esquema de Conexión para Encadenamiento o Daisy-Chain.*

Muchas veces será el propio recurso el que actúe como maestro, otras veces el maestro y el recurso solicitado son distintos. Por ejemplo cuando un periférico solicita a la CPU el robo de ciclo para el acceso a la memoria.

Las características fundamentales del *Encadenamiento* son:

- Es muy sencilla y económica.
- Existe una única línea de petición (PETI) a la que se conectan todos los dispositivos. Esta línea debe permitir que se realicen peticiones simultáneas,
- Existe una única línea de concesión (CONC) que asegura la concesión del recurso. La conexión de los dispositivos es en forma de cadena, tal que el pulso de concesión recorre una serie de dispositivos, según sea el orden en que están conectados en la cadena. El pulso de concesión es tomado por el primero de los dispositivos de la cadena que desea el recurso.
- Debe establecerse un mecanismo de enclavamiento para garantizar que solamente un dispositivo tome el flanco y que no pueda conectar una fase de servicio cuando ésta ya ha sido concedida.
- La prioridad es fija y viene definida por el orden en que los dispositivos se conectan a la señal de concesión. Cuanto más próximo en la cadena se está del maestro, se tiene una mayor opción a quedarse con la concesión.
- No es muy rápida, puesto que se debe dejar tiempo para que el testigo recorra el máximo número de dispositivos peticionarios previstos.

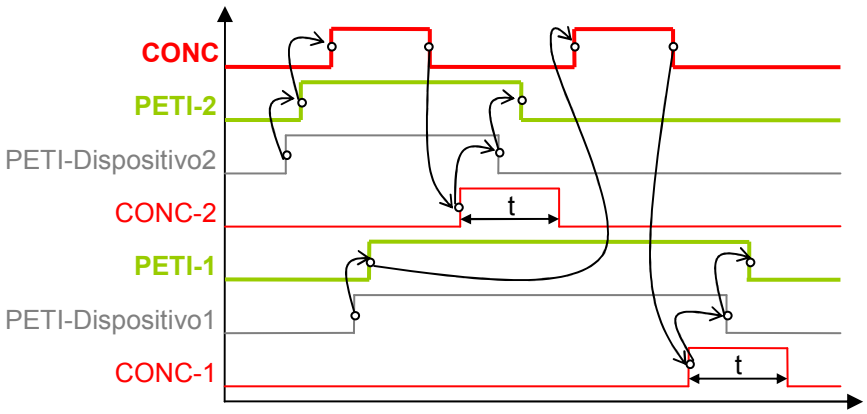

Figura 6.13. *Cronograma de Concesión de Servicio por Encadenamiento.*

La Figura 6.13 se muestra el cronograma correspondiente a la concesión del servicio por encadenamiento en el supuesto de funcionamiento síncrono, es decir los dispositivos peticionarios no intervienen en la duración del servicio.

6.2.4.1.2. Lógica Distribuida. Este método no necesita que el maestro actúe de árbitro en la gestión de la prioridad, por lo que son los propios peticionarios quienes se ponen de

acuerdo para decidir quién se queda con el servicio. Un ejemplo de Lógica Distribuida de Prioridades presenta el bus S-100, empleado para interconectar distintas placas: existen 4 líneas de prioridad (DMA0-DMA3) por lo que pueden existir hasta 16 códigos de prioridad. Cada peticionario coloca su nivel de prioridad al mismo tiempo que observa el código presente. Si éste es mayor que el suyo, se retira.

6.2.4.2. Gestión de Prioridad Centralizada

Este tipo de gestión de prioridad puede ser empleado bien cuando la conexión de los dispositivos es mediante multiplexor/demultiplexor o bien mediante bus. Cada peticionario se comunica con el gestor de prioridades mediante una señal de petición (PETI-n) y concesión (CONC-n). Puesto que sólo existe una lógica de control de prioridades, es más fácil poder establecer políticas más complejas que las de prioridad fija, empleada casi exclusivamente en la gestión distribuida.

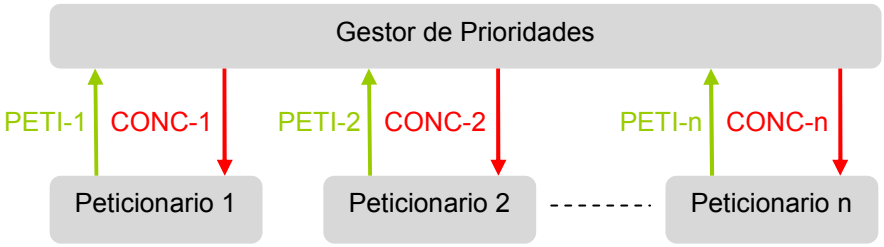

Figura 6.14. *Esquema de Gestión Centralizada de Prioridades.*

6.2.4.3. Gestión de Prioridad Híbrida

Este método emplea simultáneamente los conceptos de gestión distribuida (mediante el empleo de líneas comunes) y de gestión centralizada (mediante líneas individuales). El mecanismo centralizado establece una serie de niveles de prioridad, a cada uno de los cuales se pueden conectar una serie de peticionarios. Las prioridades dentro de cada nivel se gestionan, por ejemplo, mediante el esquema de encadenamiento.

Un ejemplo de aplicación de este método puede encontrarse en las memorias multipuertas, a excepción de la puerta asignada a la CPU, donde cada puerta sirve para conectar varios peticionarios, por lo que se establece un mecanismo de prioridad entre ellos. Los peticionarios de cada puerta suelen conectarse en forma de bus con prioridad por encadenamiento.

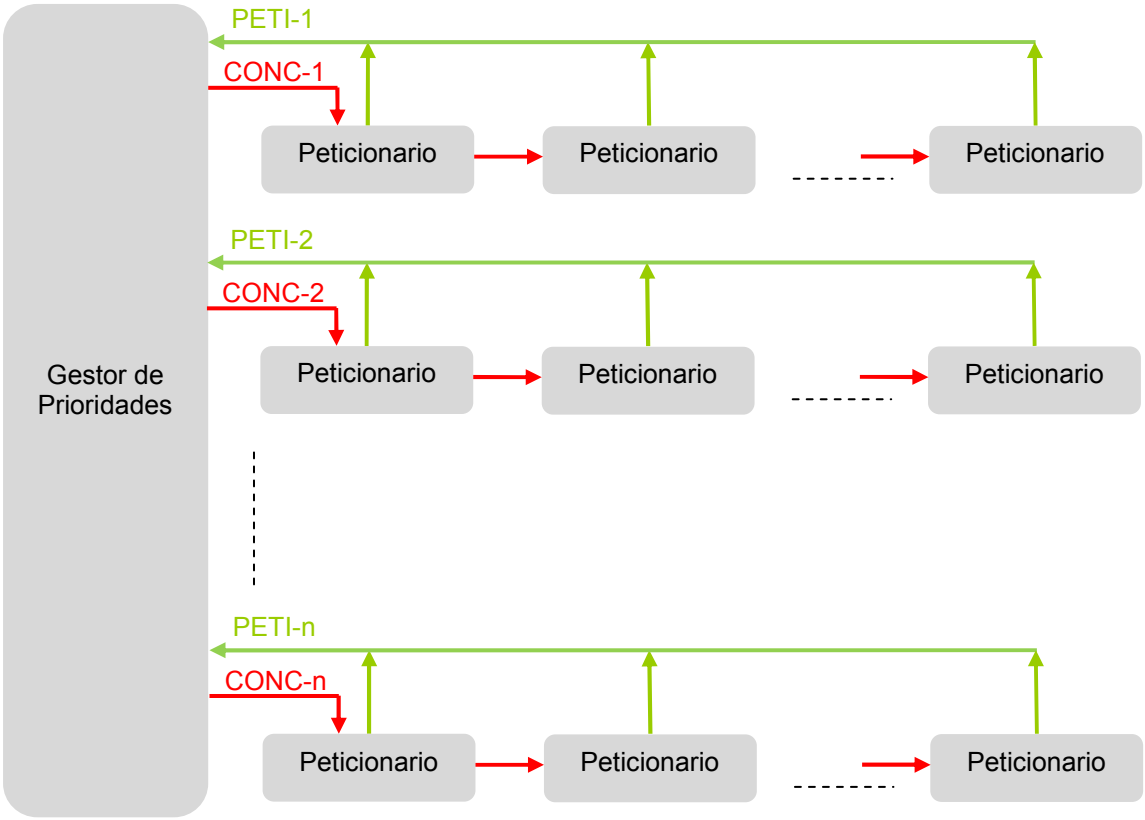

Figura 6.15. *Esquema de Gestión Híbrida de Prioridades.*

6.2.5. Interrupciones. Selección de la rutina de tratamiento de interrupciones

Las interrupciones son rupturas de secuencia no programadas provocadas por señales externas a la CPU. El objetivo que tiene el dispositivo que provoca la interrupción es reclamar la atención de la CPU sobre algún acontecimiento externo, pidiendo que se ejecute un programa específico que trate adecuadamente dicho acontecimiento. A dicho programa se le denomina Rutina de Servicio a la Interrupción (RSI).

Las etapas de una interrupción son dos:

a) Ciclo de concesión de interrupción: Se produce el diálogo entre el dispositivo y la CPU. El dispositivo solicita la interrupción y la CPU acepta dicha interrupción y bifurca a la rutina de servicio a la interrupción. En este sentido, el ciclo de concesión de la interrupción es muy parecido a una instrucción máquina y su duración será equivalente a la de éstas.
b) Tratamiento de la interrupción. Consiste en un programa que realiza las operaciones para atender al dispositivo que ha solicitado la interrupción.

La primera fase de la rutina de servicio a la interrupción consiste en salvar el estado del computador, para luego volver a dar el control al programa interrumpido. Dependiendo del computador, las funciones que realiza directamente el hardware en el ciclo de

concesión de interrupción pueden ser más o menos complicadas. La rutina de servicio deberá completar estas funciones.

La CPU comprueba las solicitudes de interrupción al finalizar la ejecución de cada instrucción. Para evitar ciclos infinitos de peticiones de interrupción, una vez que se acepta una interrupción, se inhiben las interrupciones posteriores. Así, cuando se acepta una interrupción se activa un biestable que impide que se acepte ninguna otra. Más tarde, la rutina de servicio a la interrupción vuelve a desactivar ese biestable, permitiendo otra vez la aceptación de interrupciones.

Los mecanismos básicos de conexión para petición de interrupciones han de resolver los siguientes problemas:

- Identificación del peticionario.
- Selección de la rutina de servicio.
- Desactivación de la solicitud de interrupción.
- Superposición de peticiones. El esquema de conexionado ha de permitir que más de un dispositivo tengan activadas sus solicitudes en un mismo instante.
- Resolución de prioridades. En caso de peticiones simultáneas, hay que seleccionar a quien se atiende.
- Anidamiento de interrupciones. El anidamiento correcto ha de permitir que una rutina de servicio solo se pueda interrumpir por peticiones más prioritarias.

Las formas básicas de conexión de dispositivos para producir interrupción son:

- Línea de interrupción única.
- Líneas de interrupción y aceptación.
- Interrupciones vectorizadas.

LÍNEA DE INTERRUPCIÓN ÚNICA

En este caso, el computador dispone de una única línea de interrupción (INT), que sirve para que cualquier dispositivo solicite una interrupción. Adicionalmente se dispone de un biestable general de inhibición de interrupciones (BGII).

Para permitir la superposición de peticiones de interrupción de varios dispositivos, la línea de interrupción INT, se cablea mediante una puerta OR, con lo que cualquier dispositivo puede generar una interrupción.

Para evitar el bucle infinito de ciclos de aceptación de interrupción, la CPU, al mismo tiempo que acepta las interrupciones las inhibe, desactivando el biestable BGII.

Dado que no se dispone de ninguna otra señal, para ejecutar la rutina de servicio a la interrupción la CPU bifurca a una posición de memoria fija. Esto obliga a tener una única

rutina de servicio que debe comenzar con dicha posición. Esta rutina ha de resolver, por software, los problemas de identificación del peticionario y la gestión de prioridades.

En el caso de que el sistema tenga varios dispositivos que puedan interrumpir, lo primero que hará la rutina de servicio será identificar al dispositivo que interrumpió. Para ello ha de interrogar a todos ellos, por lo que cada dispositivo debe tener un biestable de interrupción. Esta técnica de interrogación se denomina *polling* o muestreo. La asignación de prioridades se hace por el orden en que la rutina de servicio analiza los biestables de interrupción.

El anidamiento de interrupciones no puede resolverse satisfactoriamente, puesto que exige que se puedan inhibir selectivamente ciertas interrupciones.

En la Figura 6.16 puede observarse que se han agrupado los biestables de interrupción de varios dispositivos en la misma puerta de control representada mediante un buffer triestado, de forma que se acelere el proceso de interrogación.

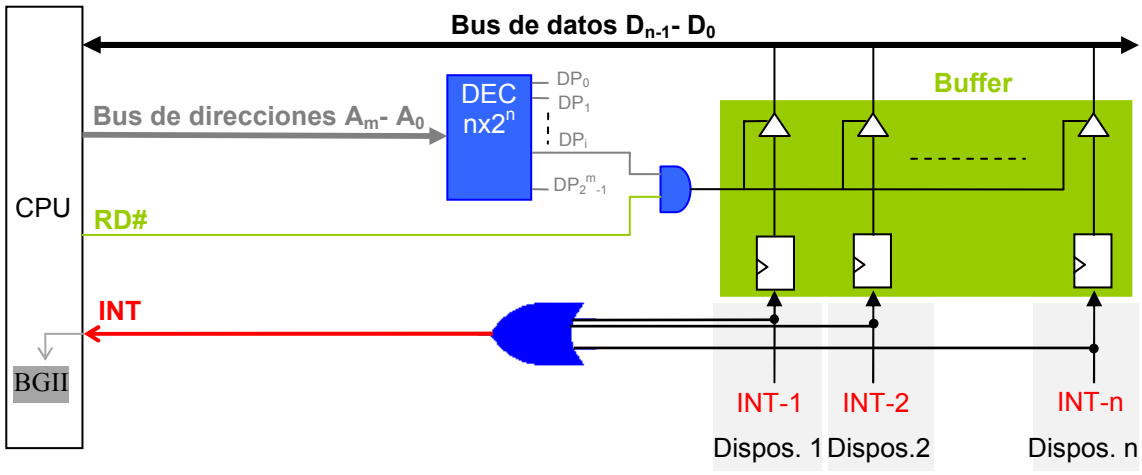

Figura 6.16. *Conexión de dispositivos mediante Línea de Interrupción Única.*

LÍNEAS DE INTERRUPCIÓN Y ACEPTACIÓN

Esta solución se considera como una extensión de la anterior, a la que se le añade una línea de aceptación de interrupción (INTA). Esta línea de aceptación de interrupción tiene origen en la CPU y llega a cada uno de los dispositivos que pueden generar una interrupción. La activación de esta señal actúa a modo de testigo, que se queda el dispositivo más prioritario.

Empleando un mecanismo hardware de encadenamiento (Daisy-Chain), pueden establecerse las prioridades de los dispositivos externos. Sin embargo, se sigue teniendo una única dirección de bifurcación, lo que obliga a una única rutina de servicio que interrogará a los dispositivos mediante la técnica de muestreo o *polling*.

La desactivación de la interrupción puede ser por software, como en el caso anterior, o puede hacerse automáticamente por el propio dispositivo, al detectar que se le acepta la interrupción.

La secuencia queda, por tanto de la siguiente forma:

- El dispositivo que desea interrumpir activa su propia señal de interrupción INT-i, y en consecuencia la CPU detecta la generación de una interrupción al activarse la señal INT.
- La CPU acepta la interrupción activando la señal INTA, al tiempo que inhibe las interrupciones, mediante el biestable BGII.
- El dispositivo de mayor prioridad que desea interrumpir, al reconocer la aceptación de la interrupción, activa su biestable de concesión (Figura 6.12) para evitar que la señal INTA se propague al resto de dispositivos conectados.
- La rutina de servicio lee el buffer que almacena el estado de las líneas INT-i y determina qué dispositivo se quedó con el testigo.
- Seguidamente se envía una instrucción de salida al dispositivo que se ha quedado con el testigo para que desactive el biestable de concesión.

Si a esta solución se le añade una puerta por dispositivo en la línea INT, gobernada cada una por su correspondiente biestable de concesión, el anidamiento queda resuelto.

INTERRUPCIONES VECTORIZADAS

Se habla de interrupciones vectorizadas cuando son los dispositivos que solicitan la interrupción los que proporcionan la dirección de comienzo de la RSI.

Las alternativas de direccionamiento son:

- Direccionamiento absoluto. El periférico suministra la dirección de la RSI. En este caso, el periférico accede al bus de datos o de direcciones para enviar la dirección absoluta de comienzo de la rutina de servicio.
- Direccionamiento relativo. El periférico sólo envía una parte de la dirección de comienzo de la RSI a través del bus de datos. El resto lo completa la CPU o bien se añaden más bits.
- Direccionamiento relativo indirecto. La dirección que envía el periférico a través del bus de datos es la posición relativa en una tabla de direcciones de rutinas de servicio de interrupciones.

Las señales necesarias para realizar el ciclo de interrupción vectorizada son: INT, INTA, para la solicitud y concesión de interrupción, además de la dirección de bifurcación y una señal de reloj para la temporización.

La principal ventaja de las interrupciones vectorizadas es que acceden directamente a la rutina que las debe tratar. Se evita así el tiempo de muestreo o *polling*.

La gestión de prioridades puede resolverse mediante la técnica de encadenamiento (Daisy-Chain), o por prioridad centralizada.

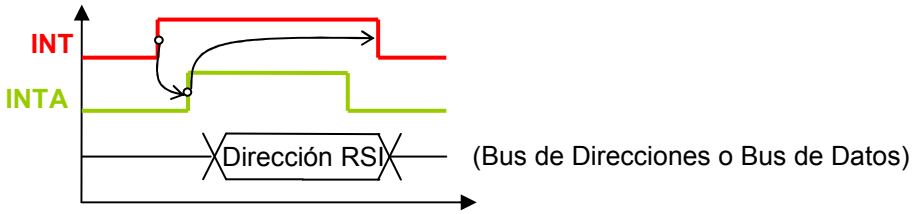

Figura 6.17. *Cronograma de Aceptación de Interrupción Vectorizada*.

En la Figura 6.17 se muestra un ejemplo de cronograma en el que se aprovecha el flanco de bajada de la señal de aceptación de interrupción INTA para que el dispositivo peticionario cancele la señal de petición de interrupción INT.

6.2.5.1. Prioridades y niveles de interrupción

En todo sistema computador con más de un dispositivo capaz de interrumpir debe establecerse un mecanismo de gestión de prioridad. Las funciones que debe resolver este mecanismo son:

- Determinación del dispositivo al que se concede la interrupción cuando existan peticiones simultáneas.
- Permitir que cada programa defina los tipos de interrupciones que puede tolerar, ya que pueden darse las siguientes situaciones:
    a. Que un programa deba ejecutarse sin sufrir ningún tipo de interrupción.
    b. Que un programa pueda ser interrumpido por un cierto número o clase de dispositivos.
    c. Que un programa pueda ser interrumpido por todos los dispositivos.
    d. Que un programa tenga fases de cada una de las tres categorías anteriores.

Para poder realizar, de forma flexible, la inhibición de las distintas interrupciones, se suelen establecer tres mecanismos distintos:

- Mediante el biestable general de inhibición (BGII).
- Asociando a cada nivel de interrupción, un biestable, llamado máscara, que permite inhibir (enmascarar) de forma selectiva cada nivel.

- Mediante un registro, común a todos los niveles, que contiene el valor del menor nivel que puede interrumpir. Un decodificador se encargará de desactivar todos los niveles de menor prioridad.

Es conveniente emplear los tres mecanismos conjuntamente para dar mayor flexibilidad al sistema. Por ejemplo, se puede asignar a cada programa o proceso un nivel de prioridad, que no pueda automodificar. El nivel de un proceso es fijo pero, si necesita eliminar algunas interrupciones puede usar el mecanismo de la máscara o el de registro.

En la Figura 6.18 se contemplan *n* niveles de prioridad, cada uno asociado a la pareja de señales INT-i e INTA-i. Contiene los tres mecanismos de inhibición mencionados anteriormente: el registro de máscara (RM), de nivel (RN) y el biestable de inhibición (BGII). Estos elementos representan la información de prioridad de la máquina y pueden leerse o escribirse mediante instrucciones especiales.

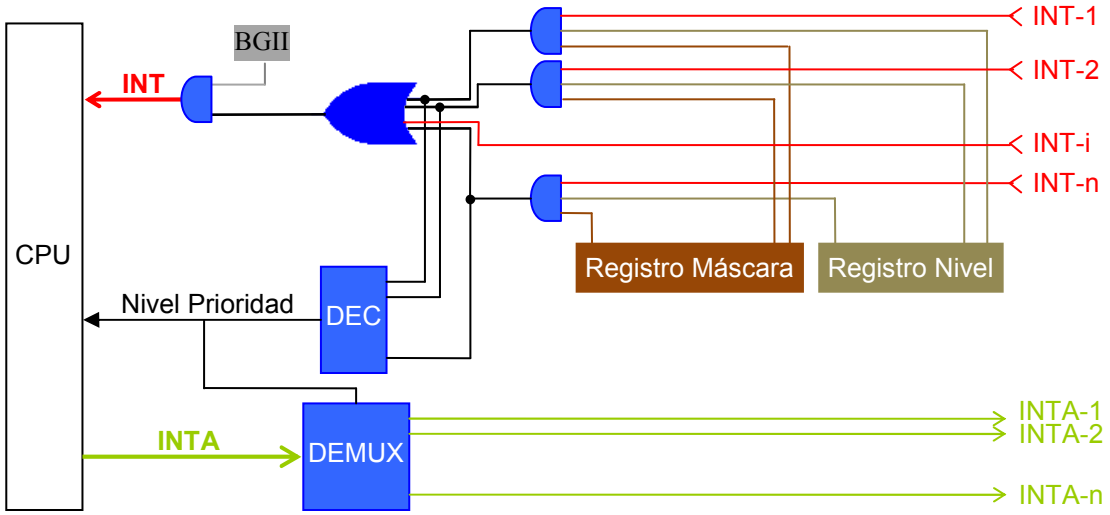

Figura 6.18. *Esquema de Generación de Interrupciones y Gestión de Prioridad.*

Existen interrupciones no enmascarables como por ejemplo la interrupción de fallo de alimentación. Para poder atender este tipo de interrupciones, la línea de interrupción correspondiente INT-i se conectará directamente a la puerta OR sin intervención de los registro RM ni RN (Figura 6.18). No obstante, dicha línea viene afectada por el biestable BGII, que debe mantenerse activo el mínimo tiempo posible para evitar que se produzca un bucle infinito de aceptaciones de interrupción y para permitir que la rutina de servicio salve el estado completo del programa interrumpido.

La asignación de prioridades de interrupción debe tener en cuenta la importancia de la misión y el tiempo de ejecución, ya que mientras menor sea el tiempo de respuesta que requiera el dispositivo, mayor ha de ser su prioridad.

La secuencia que afecta a la CPU al aceptar la interrupción depende de su diseño, pero las etapas más destacables que debe acometer son:

- Detectar la presencia de la solicitud de interrupción mediante la señal INT y la acepta activando la señal INTA.
- Salvar el contenido de su registro de estado, por ejemplo en la pila.
- Salvar el contenido de su registro contador de programa, por ejemplo en la pila.
- Poner el nivel de ejecución en su nivel máximo.
- Leer el vector de interrupciones.
- Desactivar las interrupciones mediante el biestable BGII.
- Leer la dirección de bifurcación, a partir de donde se encuentra la RSI, a través del vector de interrupción.
- Realizar la bifurcación a la RSI.

6.2.6. Canales de entrada/salida, procesadores de entrada/salida o unidad periférica de proceso (PPU)

Un canal es una unidad de E/S automática que funciona por acceso directo a memoria (DMA). En este sentido, el controlador necesario para la lectura de la unidad de disquette por robo de ciclo es un canal. Sin embargo, en esta sección se aplicará el término de canal de forma más restrictiva, puesto que se impondrán las siguientes condiciones:

- Acceso directo a memoria, preferentemente por propia puerta.
- Comunicación con la CPU a través de memoria y no por E/S programada.

El mecanismo de comunicación entre la CPU y el canal suele organizarse de forma que se dispone de un puntero PCANAL, ubicado en una posición fija de memoria, que contiene la dirección de memoria a partir de donde se encuentran las 'órdenes' implicadas en la operación de E/S.

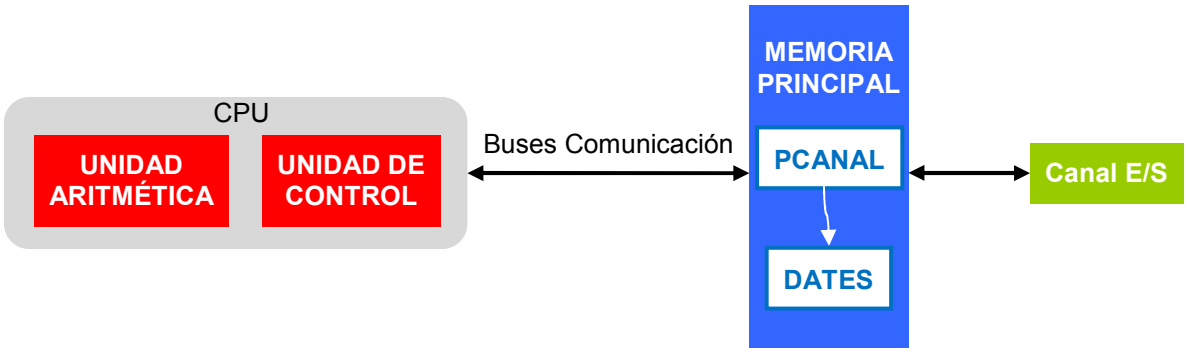

Figura 6.19. *Comunicación CPU-CANAL.*

Estos 'comandos de canal', están ubicados en la zona llamada DATES y consisten en:

- La dirección de memoria principal a partir de donde se leerá/almacenará la información correspondiente a la operación que realiza el canal.
- La dirección asociada al periférico (e indicación del periférico en el caso de existir varios periféricos conectados a un mismo canal).
- El número de bytes implicados en la operación.
- El tipo de operación (entrada o salida).
- El bit de interrupción (si está activado, el canal interrumpe cuando ha finalizado la operación).
- El bit de encadenamiento.
- El balance de la operación, esta información la define el canal al finalizar la operación de E/S correspondiente.

Para ello, los pasos a seguir son:

a. La CPU prepara los 'comandos de canal' que definen la zona de DATES.
b. A continuación, la CPU envía mediante una única instrucción de E/S, una señal al canal para indicarle que tiene que realizar una operación.
c. El canal accede directamente a memoria y, a través de PCANAL, obtiene los 'comandos de canal', DATES, de la operación a realizar.
d. Una vez que el canal ha finalizado la operación, inserta en la zona correspondiente de DATES el resultado de la misma y genera una interrupción para indicar a la CPU que ha terminado. Este balance informa del desarrollo de la operación, indicando si se ha realizado correctamente.

Como puede observarse, el canal es un pequeño procesador que tiene acceso a la memoria principal mediante robo de ciclo, y es capaz de ejecutar un conjunto limitado de instrucciones (comandos de canal). La zona de DATES especifica aquellos parámetros que el canal necesita para controlar los dispositivos de E/S y realizar operaciones de transferencia de datos.

Este mecanismo de comunicación reduce la duración de los programas de iniciación y finalización de operación de E/S, pero no los elimina. Para mejorar esta situación se empleará el encadenamiento de las operaciones de E/S.

ENCADENAMIENTO DE LAS OPERACIONES DE E/S

El encadenamiento se puede realizar sin más que añadir al final de la primera zona DATES1, otra nueva zona DATES2 (Figura 6.20). El canal, al terminar la operación con DATES1 introduce el balance de dicha operación y lee la información almacenada en DATES2. El proceso se repite hasta encontrar la indicación de fin. Las zonas DATES1, DATES2,…, DATESn se encadenan mediante un bit de encadenamiento que está activo

mientras existan nuevas zonas de datos consecutivas. Al conjunto de DATES se le suele llamar PROGRAMA CANAL**.** La CPU va generando dinámicamente el programa de canal, a medida que van surgiendo las necesidades de E/S.

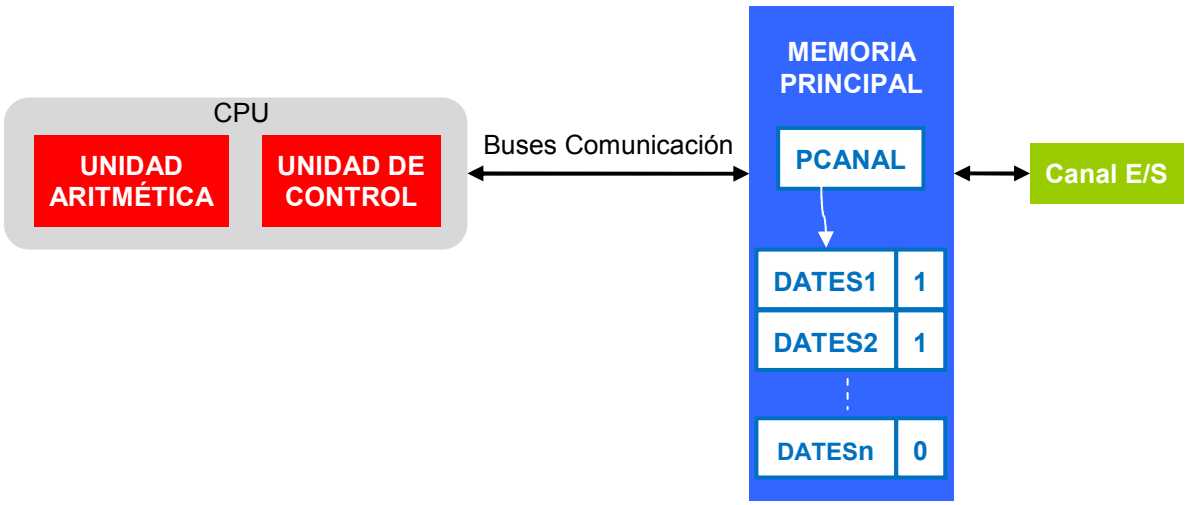

Figura 6.20. *Encadenamiento de órdenes de canal.*

Un ejemplo de aplicación de canal se tiene en la máquina IBM 360/370. La programación de canal en la máquina IBM 360/370 se hace mediante órdenes de 64 bits, cuyo formato se muestra a continuación:

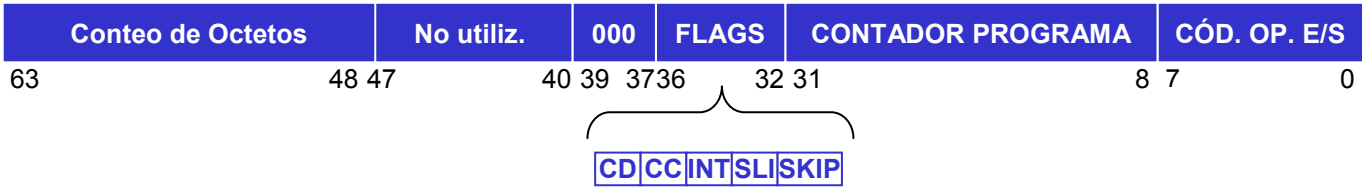

Figura 6.21. *Formato de órdenes de canal del IBM 360/370.*

El código de operación de E/S especifica las siguientes órdenes:

CÓD. OP. E/S	ORDEN
xxxx0000	Inválido
xxxx0100	Test de Periférico
xxxx1000	Bifurcación
zzzz1100	Leer
zzzz0001	Escribir
zzzzzz11	Control

Tabla 6.2. *Interpretación de Códigos de Operación de Instrucciones de Canal.*

Los códigos "z" dependen de cada tipo de periférico y sirven para especificar las condiciones de operación, por ejemplo salto de página, avance de la cinta hasta la próxima marca, etc.

El contador de programa da la dirección de la memoria principal donde se encuentran o se almacenan los datos de la operación.

Los bits correspondientes a las banderas o *flags* permiten las siguientes funciones:

- Bit CD: encadenamiento de datos. Se toman los datos de la siguiente orden pero sin considerar su código ya que se emplea el de la operación anterior.
- Bit CC: encadenamiento de órdenes. Si CD = CC = "0", quiere decir que se ha llegado al final del programa.
- Bit INT: generación de interrupción. Indica que se debe provocar una interrupción interrumpir al final de la operación correspondiente.
- Bit SLI: anulación de información. Suprime la indicación de que no se ha podido transferir el número de bytes solicitados en la orden.
- Bit SKIP: transferencia ficticia. Permite realizar transferencias sin acceder a memoria. Sirve para saltarse una zona de datos.

En el conteo de octetos se cuentan los bytes involucrados en la transferencia.

En esta máquina existen cuatro instrucciones con fines específicos de E/S:

- START I/O, indica el inicio de la operación de E/S.
- HALT I/O, termina la operación de E/S.
- TEST CHANNEL, prueba el estado del canal implicado.
- TEST I/O, prueba el estado del canal implicado, subcanal y dispositivo de E/S.

La operación de E/S mediante canal para esta máquina sigue los siguientes pasos:

- Comienzo de la operación con START I/O.
- Localización del programa de canal por medio de una palabra de dirección de canal (CAW: *Channel Addrres Word*), que está almacenada en la dirección 72 de la memoria principal. La clave es empleada para impedir el acceso a usuarios no desaseados.
- Los tres octetos más significativos de ña palabra de dirección contienen la dirección de la primera palabra de control de canal (CCW: Channel Control Word).

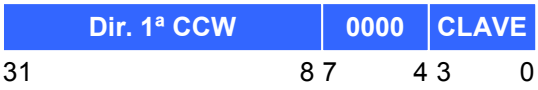

Figura 6.22. *Palabra de Dirección de Canal* (CAW).

Cada vez que el canal completa una operación de E/S, comunica su condición a la CPU

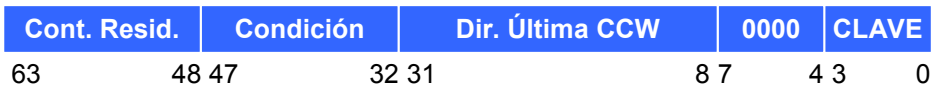

Figura 6.23. *Palabra de Estado de Canal* (CSW).

mediante la palabra de estado de canal (CSW: *Channel Status Word*).

La CPU puede terminar en cualquier momento la operación de canal con la instrucción HALT I/O.

Dado que un computador puede disponer de más de un canal, habrá que hablar de una tabla de punteros y de una tabla de dates.

ESTRUCTURA DE LOS CANALES

Cada canal suele estructurarse mediante un bus al que se conectan diversos periféricos. Se habla en este caso de canal especializado, al que se le pueden asignar uno o varios periféricos del mismo tipo de un conjunto de periféricos.

Sin embargo esta estructura presenta el inconveniente de que pueden existir, simultáneamente, canales parados y canales con lista de espera, lo que redunda en una pobre eficiencia del sistema de E/S. Por ello, algunos fabricantes han introducido el concepto de canal flotante. El canal flotante consiste en un canal que puede ser asignado a cualquier periférico. Las peticiones de operaciones de E/S se asignan a un conjunto de canales que se reparten el trabajo. Sin embargo los canales flotantes exigen que todos los periféricos estén conectados a todos los canales, lo que supone una circuitería cara y compleja.

Dependiendo del tipo de periféricos para el que está diseñado el canal se puede hablar de: Canal Multiplexado, Canal Simple o Selector y Canal Multiplexado por Bloques.

Canal Multiplexado

El Canal Multiplexado es empleado para conectar periféricos de velocidad baja y media, por ejemplo una impresora. Como la velocidad de operación del canal es mucho mayor que la de los dispositivos, es posible operar en forma simultánea con varios dispositivos y un mismo canal, es decir el canal queda dividido en subcanales que trabajan de forma cíclica atendiendo cada uno a un periférico.

Existen dos modos de operación, entre los que el canal cambia automáticamente.

- Entresacado de bytes: la operación de canal se divide en breves segmentos temporales.
- Ráfaga: la conexión lógica con el dispositivo de E/S se mantiene hasta que todas las transferencias de caracteres solicitadas por el dispositivo se completen.

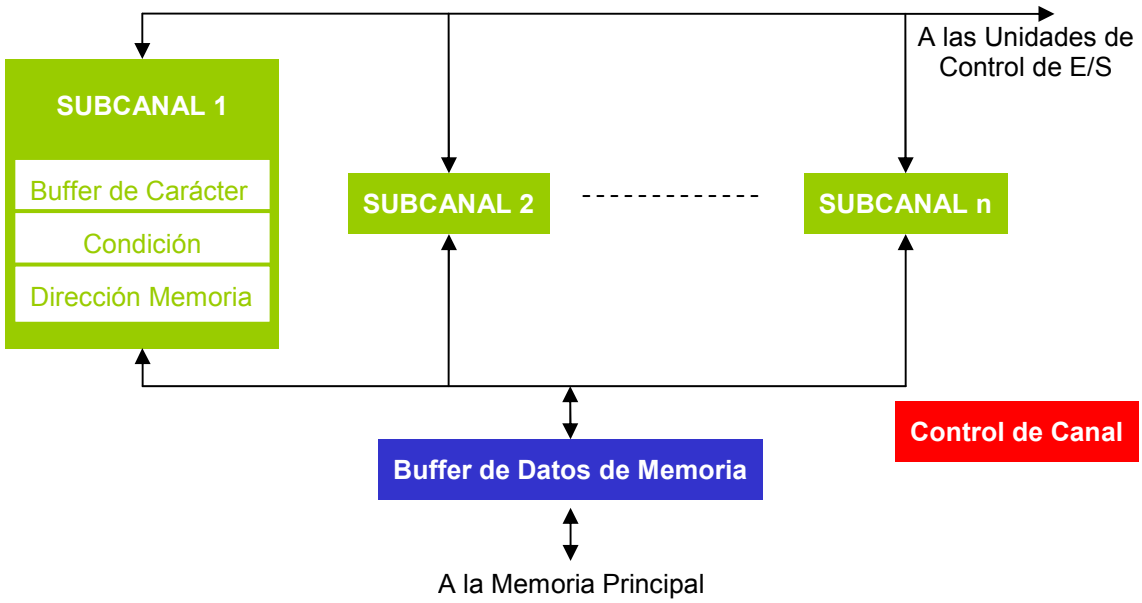

Figura 6.24. *Organización de un Canal Multiplexado.*

Canal Simple o Selector

Este tipo de canal es empleado para conectar periféricos de alta velocidad de transferencia, por ejemplo discos y cintas. Solo permite una operación al mismo tiempo. Cuando una operación de E/S se inicia por medio de un canal selector, el canal se dedica a la tarea mientras dure toda la operación. No puede utilizarse para otra operación de E/S hasta que la anterior no haya terminado.

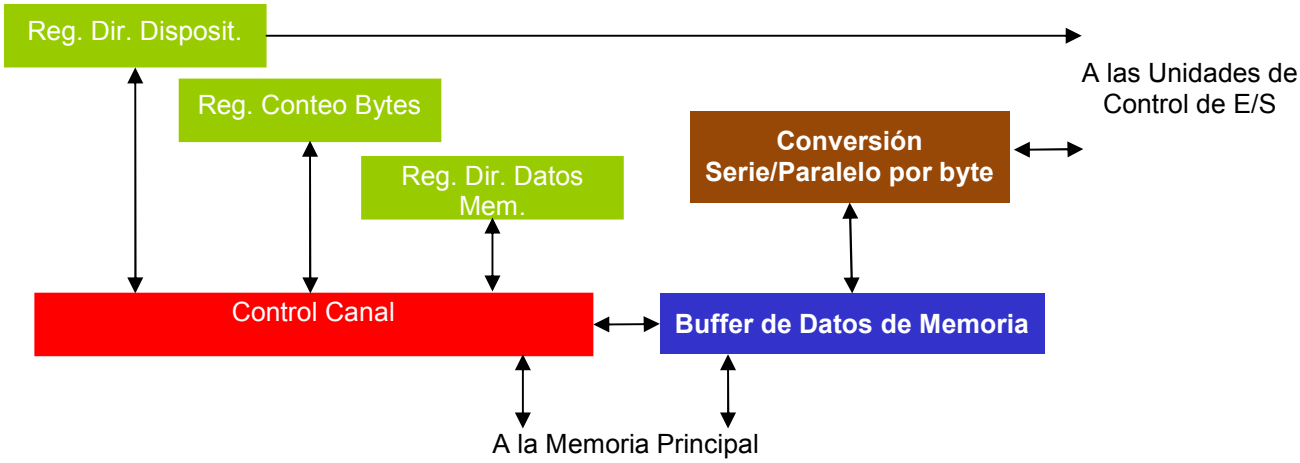

Figura 6.25. *Organización de un Canal Selector.*

Canal Multiplexado por Bloques

Permite la multiplexación de varias operaciones de E/S de alta velocidad, bloque por bloque. Es decir se trata de un canal multiplexado de alta velocidad que opera solamente en modo ráfaga.

6.2.7. Entrada/salida y sistema operativo

En este apartado se destacan aspectos que revisten la ejecución y control de una operación completa de E/S.

Supóngase que se trata de un fichero de datos con los que se pretende trabajar. El programador se limita a escribir una sentencia de apertura del fichero y, seguidamente, programará las instrucciones de lectura o de escritura. Las sentencias de alto nivel, que expresan operaciones de E/S, se compilan como llamadas al sistema operativo, siendo éste el responsable de llevar a cabo estas operaciones. Para ello, el sistema operativo ha de ser capaz de realizar las siguientes funciones:

- Manejar los directorios de los periféricos.
    a. Convertir el nombre simbólico en direcciones físicas.
    b. Insertar nuevos ficheros.
    c. Borrar un fichero.
- Generar las órdenes a los periféricos, elaborando en su caso los programas de canal.
- Gobernar los periféricos, enviándoles órdenes y atendiendo a sus necesidades y problemas.
- Ordenar las peticiones de E/S, manteniendo las colas necesarias y generando mensajes a los programas peticionarios.
- Tratar las situaciones de error, repitiendo las operaciones en caso necesario y avisar al usuario.
- Asignar las zonas de memoria necesarias para estas operaciones, realizar la conversión de código y agrupación de octetos en palabra (y viceversa).

Las rutinas de E/S del sistema operativo son, entre otras:

- Leer el sector *x* del dispositivo *y*.
- Escribir en el sector *x* del dispositivo *y*.
- Leer un carácter del teclado.
- Leer un carácter de una puerta serie.
- Enviar un carácter a una puerta serie.
- Borrar una pantalla.
- Rebobinar una cinta magnética.

El usuario debe construirse sus programas de E/S con las rutinas que facilita el sistema operativo. A la rutina de manejo de cada periférico se le suele llamar manejador o driver.

Por último, indicar que el sistema operativo debe ser capaz de realizar la función “SPOOL” (Simultaneus Peripherical Operation On Line). Esta función se emplea en las

operaciones con periféricos lentos. Consiste en usar un almacenamiento intermedio o buffer ubicado en el disco, con el objeto de hacer un desacoplo entre las velocidades de operación de E/S y el programa que la solicita.

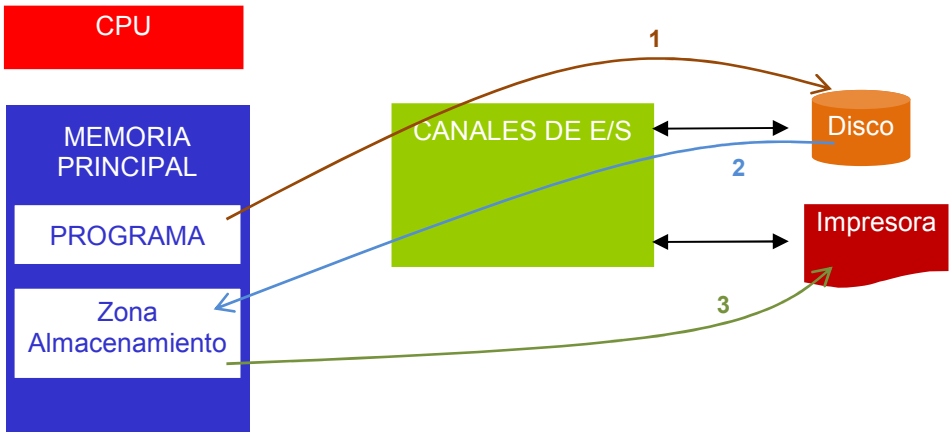

Figura 6.26. *Función SPOOL del Sistema Operativo.*

6.2.8. Ejemplo de diseño de un sistema físico de E/S

Supóngase que se desea diseñar la sección de E/S de un sistema basado en el microprocesador Intel 8085, con las siguientes especificaciones de diseño:

- Ha de tener 3 puertas de salida tipo 74LS374 de un byte cada una.
- Ha de tener 2 puertas de entrada tipo 74LS244 de un byte cada una.
- Ha de conectar una patilla del dispositivo de E/S programable 8255.
- Ha de llevar una unidad de disquete que se gobernará mediante el controlador 8272. A su vez, este dispositivo se conectará por DMA, para lo que se empleará el controlador de DMA 8257 y el controlador de interrupciones 8259.

Para formar el mapa de E/S, hay que analizar cuáles son las necesidades de direcciones de cada uno de los dispositivos involucrados. Así, por cada componente, se requiere:

- Puerta de salida 74LS374: 1 dirección, admite sólo operación de escritura (se necesitan 3 puertas).
- Puerta de entrada 74LS244: 1 dirección, admite sólo operación de lectura (se necesitan 2 puertas).
- Dispositivo de E/S programable 8255: 4 direcciones, admite operaciones de lectura/escritura.
- Controlador de disquete 8272: 2 direcciones, admite operaciones de lectura/escritura.
- Controlador de DMA 8257: 16 direcciones, admite operaciones de lectura/escritura.

- Controlador de interrupciones 8259: 2 direcciones, admite operaciones de lectura/escritura.

En consecuencia, serán necesarias un total de 29 direcciones; luego se podría plantear una distribución de direcciones de acuerdo a la Tabla 6.3:

Dirección	Dispositivo para Lectura	Dispositivo para Escritura
0	74LS244	74LS374
1	74LS244	74LS374
2	libre	74LS374
3 a 6	8255	8255
7 a 8	8272	8272
9 a 24	8257	8257
25 a 26	8259	8259

Tabla 6.3. *Asignación de Direcciones para el Sistema de E/S*.

Esta solución reservaría las primeras posiciones de memoria de forma consecutiva, sin dejar huecos, pero su implementación física implicaría un esquema de decodificación complejo. Dado que el microprocesador 8085 dispone de un bus de direcciones de 16 líneas (permite direccionar 2^{16} posiciones de memoria), y el dispositivo que más direcciones requiere es el 8257 (16 direcciones), se puede dividir el mapa de E/S en bloques de 16 direcciones, lo que equivale a reservar grupos de cuatro bits para su utilización en cada uno de los dispositivos. En base a esto, una nueva propuesta de distribución de direcciones sería la que contempla la Tabla 6.4:

Dirección	Dispositivo para Lectura	Dispositivo para Escritura
0 a 15	74LS244	74LS374
16 a 31	74LS244	74LS374
32 a 47	libre	74LS374
48 a 63	8255	8255
64 a 79	8272	8272
80 a 95	8257	8257
96 a 111	8259	8259

Tabla 6.4. *Reasignación de Direcciones para el Sistema de E/S (bloques de 16 posiciones)*.

La decodificación de estas direcciones es ahora más sencilla (Figura 6.27). De las 16 líneas del bus de direcciones (A_{15}-A_0, empleando las líneas de menos peso del bus de direcciones para el mapa de E/S), se eligen las líneas A_6-A_0 (empleando las líneas de menos peso del bus de direcciones para el mapa de E/S) para reservar las posiciones destinadas al sistema de E/S. De estas 7 líneas, se dedican 3 (A_6-A_4) como entradas para el decodificador (DEC 3x8) que permitirá seleccionar un de los dispositivos implicados. A continuación, se conectan los distintos dispositivos a cada una de las salidas del decodificador, en base a la asignación de espacio propuesta en la Tabla 6.4. Observar que se ha empleado la combinación de direcciones para seleccionar tanto un buffer tipo

74LS244 como un registro tipo 74LS374, diferenciándose entre ellos mediante las señales RD# y WR#. Finalmente, como de las 128 direcciones disponibles (7 líneas de dirección), sólo se han empelado 112, queda libre una de las salidas del decodificador, que en este caso corresponde a la última salida (128 posiciones disponibles – 112 posiciones utilizadas = 16 posiciones libres = tamaño de cada bloque seleccionado con cada salida del decodificador).

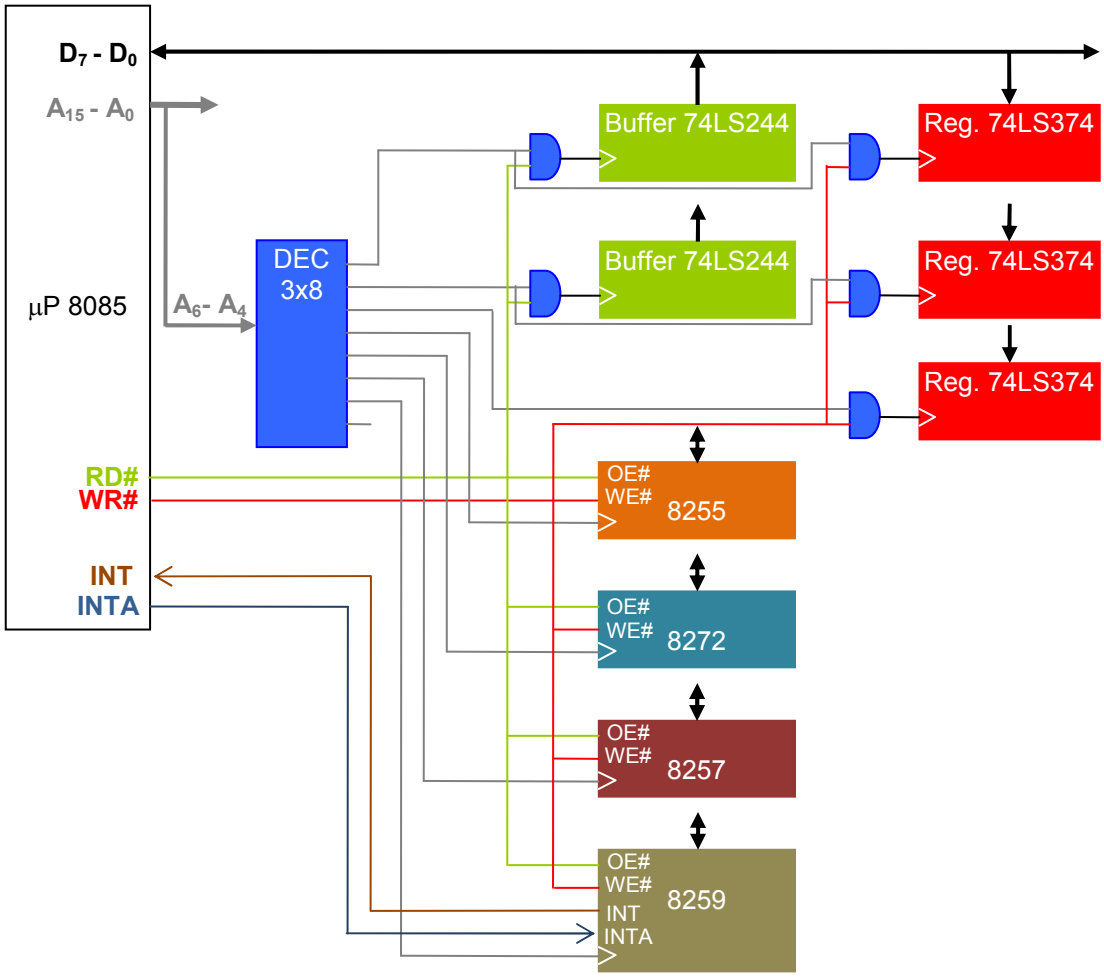

Figura 6.27. *Ejemplo de Diseño de Sistema de E/S.*