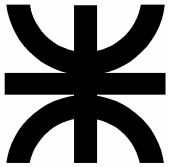

UNIVERSIDAD TECNOLOGICA NACIONAL

FACULTAD REGIONAL BUENOS AIRES

Departamento de Electrónica

Cátedra: Técnicas Digitales II - Plan 1995

CAPITULO III: Entradas Y Salidas – Versión 2006

Autores: Ings. Marcelo E. Romeo y Alejo Rubín Aymá

1 Introducción

Si recordamos la introducción en el Capítulo 1, las partes constitutivas de una microcomputadora eran: el microprocesador (Cap. 1 y 2), las memorias (Cap. 3) y los dispositivos de Entrada / Salida.

El microprocesador necesita interactuar con el mundo exterior. Los dispositivos de Entrada / Salida ofrecen la interfaz con los periféricos y son los que proveen el puente de unión entre el microprocesador y el sistema que lo rodea.

En una configuración típica se ingresan datos desde un dispositivo de ENTRADA, se procesan y luego se envían los resultados a un dispositivo de SALIDA. En un sistema de microcómputo las operaciones de E/S son particularmente importantes, ya que, en la mayoría de las aplicaciones el microprocesador emplea una buena parte de su tiempo interactuando con los dispositivos de E/S.

La operación de los dispositivos de E/S es bastante independiente de la del microprocesador, y se debe adoptar un procedimiento para sincronizar dicha operación con la ejecución del programa del mismor, durante la transmisión de datos.

Hay cuatro tipos básicos de E/S de acuerdo al método empleado para controlar y sincronizar la transferencia de datos :

a) E/S controladas por programa ***(Flag***).

b) E/S controladas por interrupciones ***(Handshake***).

c) E/S controladas por la línea ***Ready.***

d) E/S controladas por acceso directo a memoria ***(DMA).***

Estos métodos difieren en la forma en la que el microprocesador inicia y controla la transferencia de datos. En operaciones de E/S controladas por programa, la transferencia de datos está bajo el control completo del programa del microprocesador; es decir, una operación de E/S tiene lugar sólo cuando se encuentra una instrucción de transferencia de E/S durante la ejecución del programa. En muchos casos es necesario determinar la disponibilidad del dispositivo antes que ocurra la transferencia de datos. Esto implica la comprobación de uno o más indicadores externos o bits de estado asociados con el dispositivo de E/S, siendo necesaria la transferencia de estados al microprocesador (una operación adicional de E/S).

En contraste con el método de la interrupción de programa, un dispositivo externo indica directamente al microprocesador su disposición para transferir datos, activando una entrada de interrupción del mismo. La mayoría de las entradas de interrupción del microprocesador pueden inhibirse bajo el control del programa. Se ignoran las interrupciones que ocurran desde el momento en que se ha inhabilitado la entrada de interrupciones hasta que se la vuelva ha habilitar.

Cuando se interrumpe un microprocesador, se suspende la ejecución del programa en curso y se transfiere el control a una subrutina de atención (o servicio) de interrupción (ISR en inglés). Esta subrutina realiza la transferencia de datos y devuelve el control al programa en el punto en que fue interrumpido y el proceso continúa. De este modo, en una operación de E/S controlada por interrupción del programa, la transferencia se inicia mediante un hardware externo y se controla mediante la subrutina de atención de la interrupción.

Para transferir bloques de datos a *alta velocidad* se utiliza generalmente el acceso directo a memoria ***(DMA).*** En este método, controlado por hardware, la transferencia se efectúa en forma directa entre el dispositivo de E/S y la memoria: el dato no se dirige del dispositivo de E/S a uno de los registros del microprocesador y luego a la memoria principal, o viceversa, sino que se transfiere directamente entre el dispositivo externo y la memoria. El microprocesador sólo interviene en la inicialización indicando la dirección de inicio en la memoria principal y la extensión del bloque de datos a transferir. El hardware asociado al dispositivo de DMA inicia y controla la transferencia de datos propiamente dicha.

Puede ocurrir que los periféricos que intervienen en la transferencia de E/S de información sean algo más lentos que el procesador. Para adecuar la velocidad del procesador a la de dichos periféricos se puede emplear la línea ***READY***. Para sincronizar su operación con el procesador un periférico lento sólo debe aplicar un "0" a la línea ***READY*** a partir del momento en que recibe las señales /RD o /WR que los habilita debiéndola mantener en ese estado el lapso que su escasa velocidad requiera.

El tipo de E/S a utilizar en una determinada aplicación dependerá de tres factores principales :

a) La velocidad de transmisión de datos.

b) El máximo retardo que se pueda aceptar entre el momento en que un dispositivo de E/S esté listo para transmitir o recibir datos y el momento en que se realice la transferencia.

c) La factibilidad de entrelazar las operaciones de E/S con otras actividades del microprocesador.

Cada uno de los cuatro tipos de E/S enumerados convendrá utilizarlo, según el tipo de periférico, de acuerdo con la siguiente guía:

a) Con periféricos de mediana velocidad conviene trabajar por ***FLAG*** (control por programa).

b) Con periféricos rápidos utilizando el esquema de la línea ***READY***.

c) Con periféricos de Actuación Esporádica con el esquema de ***Handshake e Interrupciones.***

d) Con periféricos Muy Rápidos mediante acceso directo a memoria ***(DMA).***

Por supuesto que deberíamos definir y cuantificar que entendemos por periféricos rápidos, muy rápidos, etc. Siendo en definitiva un parámetro subjetivo y de permanente evolución tecnológica.

Esta guía no implica que deberá adoptarse un determinado esquema único según el tipo de dispositivo a emplear. En muchos casos convendrá realizar una combinación de dos o más tipos de control de E/S.

En teoría la transferencia de información con un dispositivo de E/S no es muy diferente de la transferencia de información con la memoria. De hecho, podríamos considerar a la memoria como otro periférico más, pero la memoria es un periférico particular por las siguientes razones:

a) En general opera a la misma velocidad que la CPU.

b) Utiliza las mismas tensiones que la CPU.

c) Utiliza solamente las señales RD/ y WR/.

d) Memoriza en forma automática los datos que se le direccionan.

La mayoría de los periféricos no cumplen con las características anteriores. Pueden operar a velocidades muy inferiores, por ejemplo, una impresora puede transferir 10 caracteres por segundo mientras que el procesador lo puede hacer a centenares de millones de caracteres por segundo. La gama de velocidades es muy amplia. Un transductor puede entregar una información por minuto mientras que una unidad controladora de disco puede transferir millones de bits por segundo.

Podemos, groseramente, separar a los periféricos en tres categorías de acuerdo a su velocidad de transmisión:

a) Periféricos lentos, en los que se realiza un cambio de estado enno más de una vez por segundo, y este cambio requiere algunos milisegundos. Los letreros luminosos, los interruptores, los relevadores y la mayoría de los transductores y accionadores de mecanismos pertenecen a este grupo.

b) Periféricos de mediana velocidad. Realizan una operación en algunos centenares de instrucciones del procesador.

c) Periféricos rápidos: su operación se realiza como máximo en el tiempo de algunas instrucciones (menos de 5). Ejemplos son los discos magnéticos, sistemas de adquisición de datos, sistemas de redes y comunicaciones de alta velocidad.

2 E/S Controladas Por Programa.

2.1 Control/Estado y Datos

La intervención del procesador es preponderante, ya que tiene la iniciativa y el control total de la transferencia de entrada o de salida. El programa determina en qué momento debe realizarse: de allí los nombres de modalidad *programada* o *con control del procesador* o *iniciada por programa*, usadas por diferentes autores. Existen dos posibilidades en cuanto a este tipo de transferencias programadas, ya que pueden ser sincrónicas o asincrónicas. Cuando el periférico tiene un tiempo de respuesta conocido, y puede estar listo para comunicarse con el sistema para recibir o emitir datos en intervalos compatibles con los ciclos de instrucción del procesador, se dice que la transferencia es sincrónica. En cambio, si el periférico responde empleando lapsos no fácilmente conocidos o muy variables, el procesador deberá probar el estado de aquel antes de realizar una operación de E/S, que entonces será asincrónica por depender de una prueba de *dispositivo listo* cuyo resultado positivo no se sabe cuánto tardará en ocurrir.

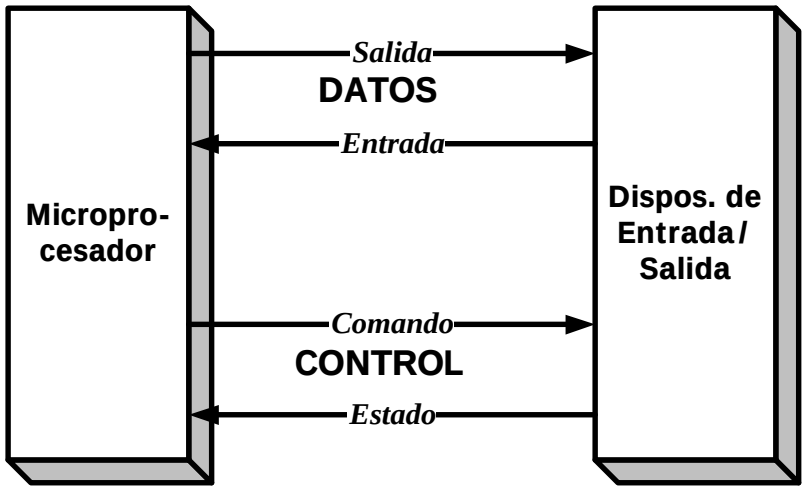

Fig. 1. Flujo de información entre el dispositivo y el microprocesador.

En la figura 1 se muestran los dos tipos básicos de información que se transmite entre el microprocesador y el sistema de E/S: mensajes de ***Datos*** y mensajes de ***Control.*** Los mensajes de control se emplean para sincronizar la operación del dispositivo con la ejecución del programa, antes de realizarse la transmisión de los datos. Los mensajes de control de entrada se denominan. *Palabra De Estado Del Dispositivo.* Los mensajes de control de salida se denominan *Palabra De Comando Del Dispositivo*. Con E/S controladas por programa, las instrucciones de E/S se utilizan para iniciar y controlar la transferencia de todo tipo de mensajes.

El microprocesador lee la palabra de estado para determinar el estado actual del dispositivo. Cada bit de la palabra de estado le indicará una condición particular del dispositivo como:

a) Datos listos para transmitir.

b) Dispositivo ocupado.

c) Dispositivo no conectado o error de transmisión.

El microprocesador envía al dispositivo la palabra de comando para controlar su operación. Cada bit o grupo de bits de la palabra de comando tiene una función particular como parar el motor, cambiar la velocidad de transmisión (baud rate), alimentar papel o cinta, etc.

2.2 Organización de las instrucciones de E/S.

Las instrucciones de E/S pueden organizarse de tres formas distintas

1. Hay una única instrucción para cada tipo de transferencia de mensajes de E/S utilizando una ***Dirección De Dispositivo*** simple para definir a cada uno. Las cuatro operaciones típicas son:

    - Leer dato
    - Escribir dato
    - Enviar comando
    - Recibir estado

2. Dos instrucciones de E/S: una para ingreso y otra para salida, ya sea para datos o palabras de control. Se utilizan por lo menos dos direcciones de dispositivo de E/S distintas: una para datos y otra para control.

   Las dos instrucciones típicas son:

   - Leer mensaje (ya sea dato o palabra de estado)
   - Escribir mensaje (ya sea dato o palabra de comando)

3. No hay instrucciones de E/S separadas. Las instrucciones de transferencia de datos de memoria se utilizan también para comunicarse con los dispositivos de E/S mediante la asignación de bloques de direcciones de memoria libres como dirección del dispositivo. A esta disposición se la denomina *E/S Mapeadas En Memoria*. Si bien esta disposición reduce el área de memoria disponible también puede reducir la extensión de los programas y su tiempo de ejecución ya que permite utilizar instrucciones de operación con memoria, las que generalmente son más poderosas y rápidas que las que trabajan con las entradas y salidas.

Los microprocesadores de las familias de Motorola, son ejemplos de E/S mapeadas en memoria, ya que no poseen instrucciones específicas de E/S, mientras que los de Intel y AMD permiten utilizar las E/S mapeadas en memoria, o mapeadas como E/S en forma separada.

También se hallan disponibles instrucciones que permiten barrer en forma secuencial una porción del mapa de E/S transfiriendo datos desde un periférico a direcciones consecutivas de memoria o viceversa. Es una función análoga al acceso directo a memoria.

Con este tipo de instrucciones se logra, utilizando sólo dos bytes, realizar tareas que en otro tipo de microprocesador demandarían varias líneas de programa.

2.3 Dispositivos de E/S en el mapa de memoria.

Una operación de E/S integrada en memoria se ejecuta como una operación de lectura o escritura en memoria. Sin embargo, no existe memoria en esa dirección y en cambio la posición se decodifica como la de una puerta de E/S. Este método de E/S integradas en memoria tiene la ventaja de permitir el uso de cualquier registro de la CPU para transferencia de datos (una puerta de E/S siempre utiliza el registro Acumulador AL para operaciones de bytes o AX para operaciones de palabras). Esto puede eliminar la necesidad de una transferencia al registro Acumulador.

En el caso de que el segmento reservado a la memoria de datos se halle completo con 64K ó mas de memoria física, los dispositivos de E/S mapeados como memoria deberán ubicarse en otro segmento,. En tal caso, se deberá inicializar un registro de segmento (habitualmente ES o DS) para apuntar hacia él.

Por ejemplo, supongamos que se ubicó una puerta de entrada en la dirección 2000:0000H y una puerta de salida en 8000:3456H y se desean transferir un byte de la puerta de entrada a la puerta de salida, tendremos:

```
MOV       BX,2000H        ; Podría ser cualquier registro de 16 bits
MOV       DS,BX
MOV       DX,8000H        ; Podría ser cualquier registro de 16 bits
MOV       ES,DX
MOV       AL,[0]          ; Podría ser cualquier registro de 8 bits.
                          ; Lee un byte de la posición DS:0
MOV       ES:[3456H],AL   ; Escribe ese byte en 8000:3456
```

Además, diferentes secciones de un programa pueden utilizar distintos registros de la CPU para E/S. Por ejemplo las instrucciones de incrementar/decrementar registros de 8088 tratan a la memoria como si fuera un registro y modifica el contenido de la posición apuntada por el registro de dirección. De aquí que estas rápidas instrucciones de un byte puedan modificar también un registro de un dispositivo de E/S (una puerta de E/S necesita varias instrucciones de dos bytes para hacer esto).

Una desventaja de este método es que las direcciones de memoria usadas para operaciones de E/S no se pueden utilizar también para direccionar memoria real. Además como en el acceso de dispositivos de E/S habitualmente no se realiza decodificación completa, el uso de una sola dirección para E/S impide el uso de un bloque completo de varios kilobytes para el direccionamiento de RAM.

En aplicaciones de controladores dedicados, en los que los programas son usualmente pequeños esto puede no ser de importancia. Sin embargo en el caso de microprocesadores que corren compiladores como C, Pascal, etc., los que requieren grandes espacios de memoria y sistemas operativos residentes, la indisponibilidad de memoria puede ser seria.

Para el 8088, si se desea conocer el contenido de dos puertas de entrada adyacentes y volcar su contenido en el registro BX, si las puertas se hallan mapeadas como memoria, basta con:

```
MOV     AX,[PORTLOW]          ;dirección de 16 bits
```

Realizándose la lectura de DS:PORTLOW y almacenándoselo en AL e inmediata y automáticamente, la lectura de PORTLOW + 1, guardándosela en AH.

Si se hallan ambas puertas mapeadas como E/S, se debe hacer:

```
PUSH          AX            ; a fin de no alterar el contenido
                            ; del acumulador
IN            AX,PORTLOW    ; dirección de 16 bits
MOV           BX,AX
POP           AX
```

Las instrucciones de Entada / Salida típicas equivalen a:

a) Cargar el mensaje (ingreso de datos o palabra de estado)

b) Almacenar el mensaje (salida de un dato o palabra de comando)

En algunos microprocesadores ambos (datos y control) se envían o reciben a través del acumulador o algún otro registro de trabajo. Algunos sistemas tienen registros especiales para manejarse con los mensajes de control o utilizan algún registro de estado del procesador principal con dicho fin. En este último caso la prueba del estado del dispositivo se simplifica ya que las instrucciones de salto condicionado pueden verificar directamente los bits del registro de estados en forma individual.

Los mensajes de control se utilizan para sincronizar la transferencia de datos bajo el control del programa, de la siguiente manera:

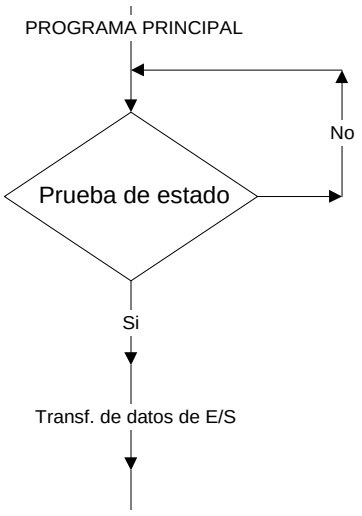

Fig. 2. Diagrama de Flujo de una operación de E/S con prueba de estado

a) Se escribe una palabra de comando en un dispositivo de E/S para solicitar la transferencia de datos.

b) Se lee la palabra de estado desde el dispositivo de E/S.

c) Se verifican los bits de estado correspondientes para saber si se pueden transferir los datos.

d) Si el dispositivo no está listo se repiten los pasos b) y c) hasta que esté listo para transferir los datos.

e) Se efectúa la transferencia de datos (desde o hacia) el dispositivo de E/S.

Esta operación inicializará el estado del dispositivo.

El paso a) se omite en las aplicaciones donde la decisión de iniciar la transferencia la genera el propio dispositivo. En este caso el dispositivo indicará su condición de listo para la transferencia mediante los bits de estado apropiados.

En un programa simple, como:

```
          mov     DX, dir_estado
Vuelta:   in      AL,DX
          test    AL,00100000b
          jz      vuelta

          mov     dx,dir_datos
          in      al,dx
```

la prueba de estado (pasos b) y c)) se repite continuamente hasta que el dispositivo esté listo, como se muestra en la **¡Error! No se encuentra el origen de la referencia.**. El lazo de prueba de estado detiene la ejecución del programa y puede producir una pérdida de tiempo de ejecución de programa que resulte inaceptable.

En la situación límite en la que el dispositivo no responda indefinidamente, se detendría la operación del sistema en forma permanente. Por al motivo se suele colocar una prevención de tal situación denominada exceso de tiempo o ***timeout.*** La misma consiste en realizar la verificación de estado una determinada cantidad de veces o bien por un cierto lapso. En caso de que el sistema no responda con un estado afirmativo, se asumirá una falla del mismo y se generará una condición de error.

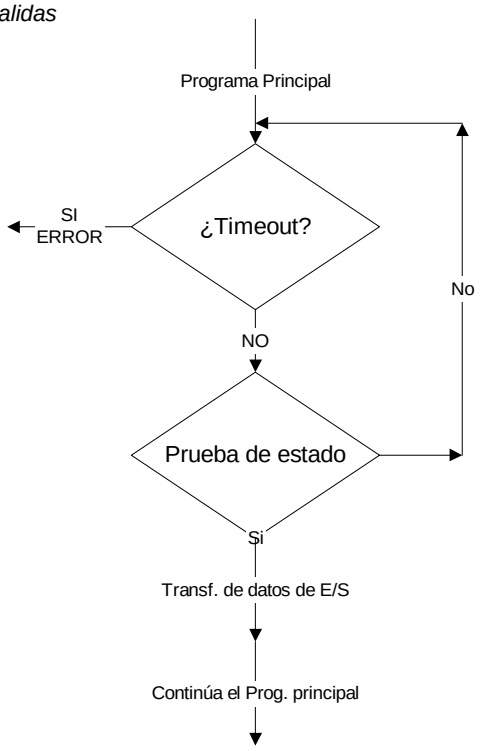

Fig. 3. Introducción del error por exceso de tiempo o timeout

En la Fig. 3 se muestra un esquema más complejo donde la prueba del estado se entrelaza con otras operaciones del microprocesador permitiendo proceso más eficiente en el empleo del tiempo.

```
            Mov     DX, dir_estado
            mov     cx,cuenta
Vuelta:     in      AL,DX
            Test    AL,00100000b
            jnz     listo
            loop    vuelta

            jmp     ERROR

Listo:      mov     dx,dir_datos
            in      al,dx
```

El problema es particularmente importante en los sistemas donde el microprocesador se comunica con varios dispositivos de E/S, en cuyo caso la prueba deberá efectuarse con cada uno de los dispositivos. Esta operación de encuesta de dispositivos puede introducir también un considerable retraso de tiempo entre la disposición del periférico para transferir datos y la prueba por parte del programa sobre su estado.

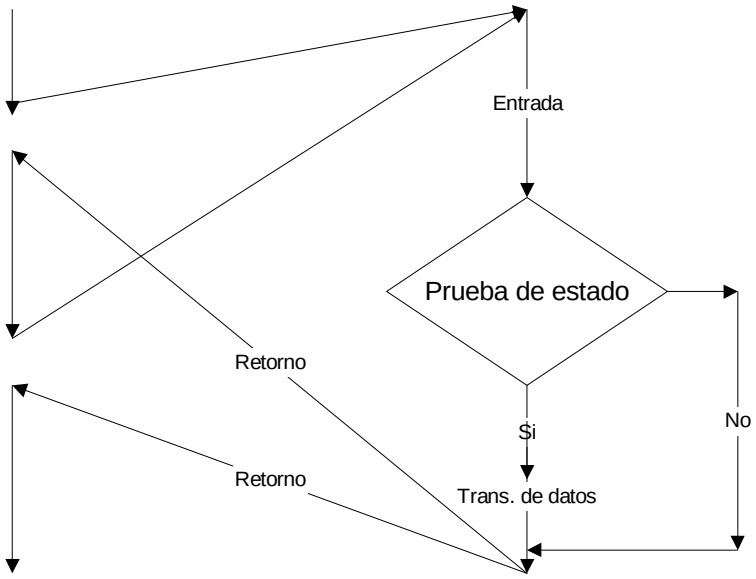

Fig. 4. E/S controladas por programa (operación entrelazada)

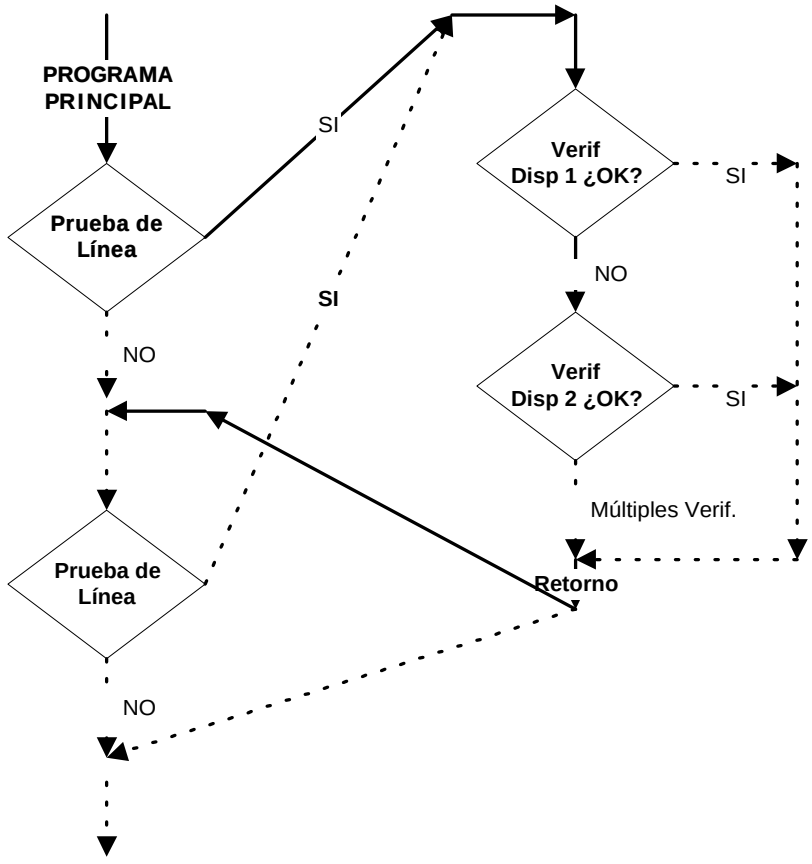

Fig. 5. E/S controladas por programa (línea de prueba)

En algunos microprocesadores el tiempo empleado en comprobar el estado de los dispositivos se reduce al emplear una única línea. Esta línea es común a todos los dispositivos y se utiliza para pedir la atención de cualquiera de ellos.

En la Fig. 5 se observa que el microprocesador puede probar periódicamente en forma rápida una línea, y en caso de haber un pedido, hará una encuesta serie (un dispositivo por vez) hasta encontrar el pedido de atención. El tiempo de espera antes de atender a un dispositivo puede ser aún considerable.

2.4 Ejemplo De E/S Controlada Por Programa

En el esquema de la Fig. 6 se observa un microprocesador con sus buses de direcciones y datos a los que se vinculan: Una puerta de entrada (P2), y dos de salida (P1 y P3), todas de ocho bits. El decodificador activará una de las líneas de selección (CS1/, CS2/, CS3/), según la dirección emitida por el microprocesador. También figuran las líneas de control de lectura (RD/) y de escritura (WR/). Se sobrentiende la existencia de memorias no indicadas en la figura. Mediante este simple esquema se ejemplificarán las dos alternativas de E/S programada.

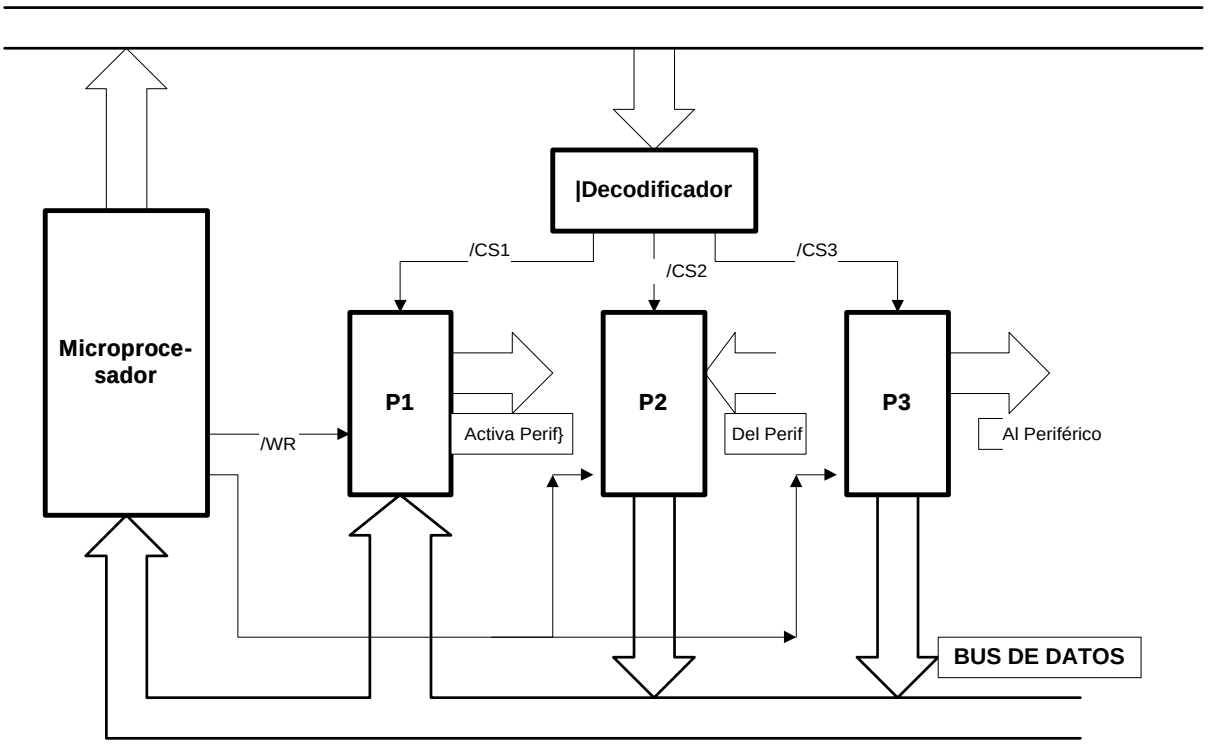

Fig. 6. E/S controlada por programa (ejemplo)

2.4.1 E/S Programada Sincrónica

Se tiene un periférico que debe proveer datos de entrada al microprocesador por la puerta de entrada P2, después de haber sido puesto en marcha (activado) a través de la puerta P1.

Se sabe que dicho dispositivo habrá de responder poniendo los datos en P2, con un retardo de tiempo conocido, que a lo sumo será un cierto valor *dt* de un orden de magnitud compatible con los tiempos de ejecución de instrucciones.

Entonces, tal como puede verse en el diagrama de la Fig. 7, el programa deberá comenzar por activar al periférico, para lo cual el microprocesador tendrá que escribir una cierta palabra sobre P1; se supone que bastará con tener un "1" lógico en el bit de orden 2, sin importar los restantes bits. A partir de allí habrá que esperar un tiempo mayor o igual que *dt*, antes de direccionar y leer la puerta P2. Esto puede lograrse de las siguientes maneras:

1) Intercalar instrucciones útiles que permitan al microprocesador seguir adelante con el programa mientras llega el momento de leer la información de P2.

2) Si no hay posibilidad de realizar tal procesamiento, habrá que perder el tiempo antedicho, ya sea con instrucciones de no-operación (NOP), cuando el intervalo es breve, o mediante un lazo de espera que decremente un contador.

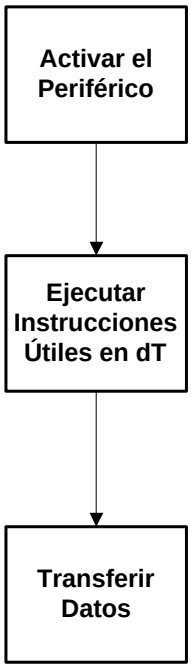

Fig. 7. E/S programada sincrónica

El programa correspondiente para el 8088, con puertas mapeadas en forma separada de la memoria será:

```
MOV     AL,04          ;Cargar código de activación del periférico.
OUT     P1,AL          ;Activar el periférico.
........               ;instrucciones para lograr una demora dT
IN      AL,P2          ;leer información
```

En el ejemplo se usa al acumulador (registro AL) para transferir la constante 00000100B a la puerta P1, y después del retardo se hará la lectura de P2 llevando su contenido al mismo acumulador.

En la

Fig. 8 se muestra un diagrama de tiempos simplificado (no correspondiente a microprocesador alguno en particular), para dar una idea de la secuencia de eventos:

a) El microprocesador, como consecuencia de la instrucción OUT P1,AL, activa la línea /CS1 y escribe (/WR) sobre la puerta de activación del periférico.

b) Se espera el tiempo de respuesta dt o mayor.

c) El microprocesador activa /CS2 y /RD para leer la puerta P2, todo ello por medio de la instrucción

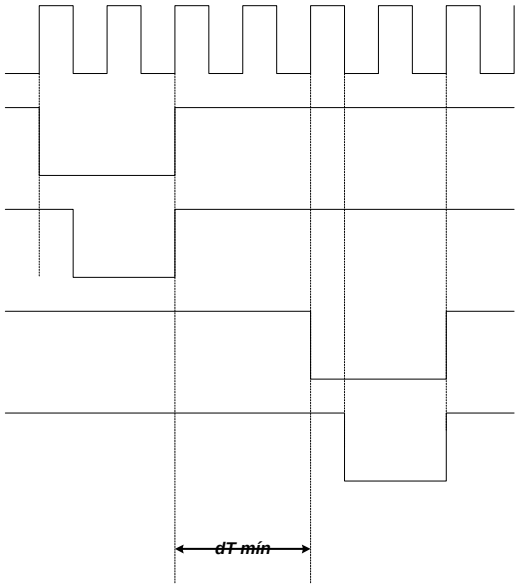

Fig. 8. Diagrama esquemático de tiempos.

El intervalo dT, en caso de usarse lazos de espera, podría lograrse de la siguiente manera:

```
MOV       CX,NUM
LOOP      $
```

Donde NUM un número cualquiera entre 1 y FFFFH y es quien determina el retardo deseado. Este ejemplo supone la transferencia de una sola palabra desde el periférico. El mismo se puede extender al caso de la transferencia de un bloque de palabras, con una velocidad conocida (dada en palabras por segundo). En este caso la instrucción de lectura (IN AL,P2) deberá ser repetitiva mediante un lazo que ajuste la frecuencia de lectura a dicha velocidad de transmisión, y finalice al agotarse el número N de palabras a transmitir.

Para este caso el programa resulta:

```
MOV       SI, offset TABLA       ;apunta a zona de almacenaje en memoria
MOV       AL,4                   ;carga código activación periférico
MOV       CX,NUM                 ;carga cuenta lazo de espera dt
MOV        BL,N                  ;carga contador palabras a transferir
OUT       P1,AL                  ;activa periférico
LOOP      $                      ;lazo de espera dT
LEER:     IN AL,P2               ;lee puerta P2
MOV       [SI],AL                ;almacena en memoria
MOV       CX,NUM1                ;carga cuenta lazo espera dT1[1]
LOOP      $                      ;lazo de espera dt1
INC       SI                     ;apunta siguiente posición de memoria
DEC       BL                     ;decrementa contador palabras
JNZ       LEER                   ;si hay más palabras volver a LEER
……                               ;continúa programa
```

[1] Como se dijo antes, el registro CX se carga con un número adecuado al retardo dt necesario para que el periférico comience a transmitir las N palabras que se cuentan en el registro BL. A partir de la primera lectura y almacenaje en memoria, en la dirección inicial de una tabla señalada por el puntero SI, se incluye un retardo dt1 para que las sucesivas lecturas se realicen a la frecuencia necesaria. Por ejemplo, si el periférico transmitiese a razón de 1000 palabras por seg. el lazo de lectura dt1 debería tener una duración de 1 ms. para cada ejecución del conjunto de instrucciones que lo forman.

Puede decirse también que la transferencia contemplada en este caso es *sincrónica* en el sentido que el microprocesador debe efectuar las lecturas en sincronismo con la velocidad de transmisión del periférico, siempre que éste transmita en forma regular.

2.4.2 E/S Programada Asincrónica

En el caso de la transferencia programada sincrónica se entiende que el periférico está siempre listo, o bien que una vez activado tarda siempre un tiempo fijo y conocido en llegar a esa condición. Pero en muchas situaciones prácticas es imposible prever con alguna exactitud tales intervalos o bien los mismos varían entre límites demasiado amplios, por lo cual la programación de retardos de *caso peor* determinaría una excesiva pérdida de tiempo. Entonces el periférico deberá indicar su estado de *listo* mediante un bit de estado *(Flag)* cualquiera, que será verificado periódicamente por el programa.

Consideremos como ejemplo, para este caso, un sistema mecánico de precisión controlado por el microprocesador de la Fig. 6.El mismo recibe órdenes de comando desde la puerta de salida P1, indica su estado mediante una palabra que coloca en la puerta de entrada P2 y recibe datos de posicionamiento desde la puerta de salida P3. La palabra de comando le indica al dispositivo un determinado modo de trabajo (rápido o lento) mediante los bits de menor peso. La palabra de estado indica si el dispositivo está ocupado, si hubo un error en la transmisión, o si está fuera de servicio.

Como se trata de un sistema mecánico de posicionamiento, donde los recorridos son totalmente aleatorios, también lo serán los tiempos empleados en efectuarlos y, por lo tanto, no se puede establecer una demora determinada entre cada uno de los sucesivos desplazamientos, por lo que resulta conveniente el manejo en forma asincrónica.

Los resultados del procesamiento de una información se cargan primero en un buffer (zona de la memoria principal) y luego se envían al dispositivo bajo el control de un programa.

Dicho programa, antes de enviar los datos de los sucesivos posicionamientos, verificará si el dispositivo está en condiciones de operar (en servicio) mediante la lectura de la palabra de estado a través de la puerta P2, debiendo encontrar en "0" el bit 2,y enviará una palabra de comando de modo de operación a través de la puerta P1.

En la Fig. 9 se muestra un diagrama de flujo correspondiente a la operación de dicho tramo de programa, y las correspondientes palabras de comando y estado del dispositivo.

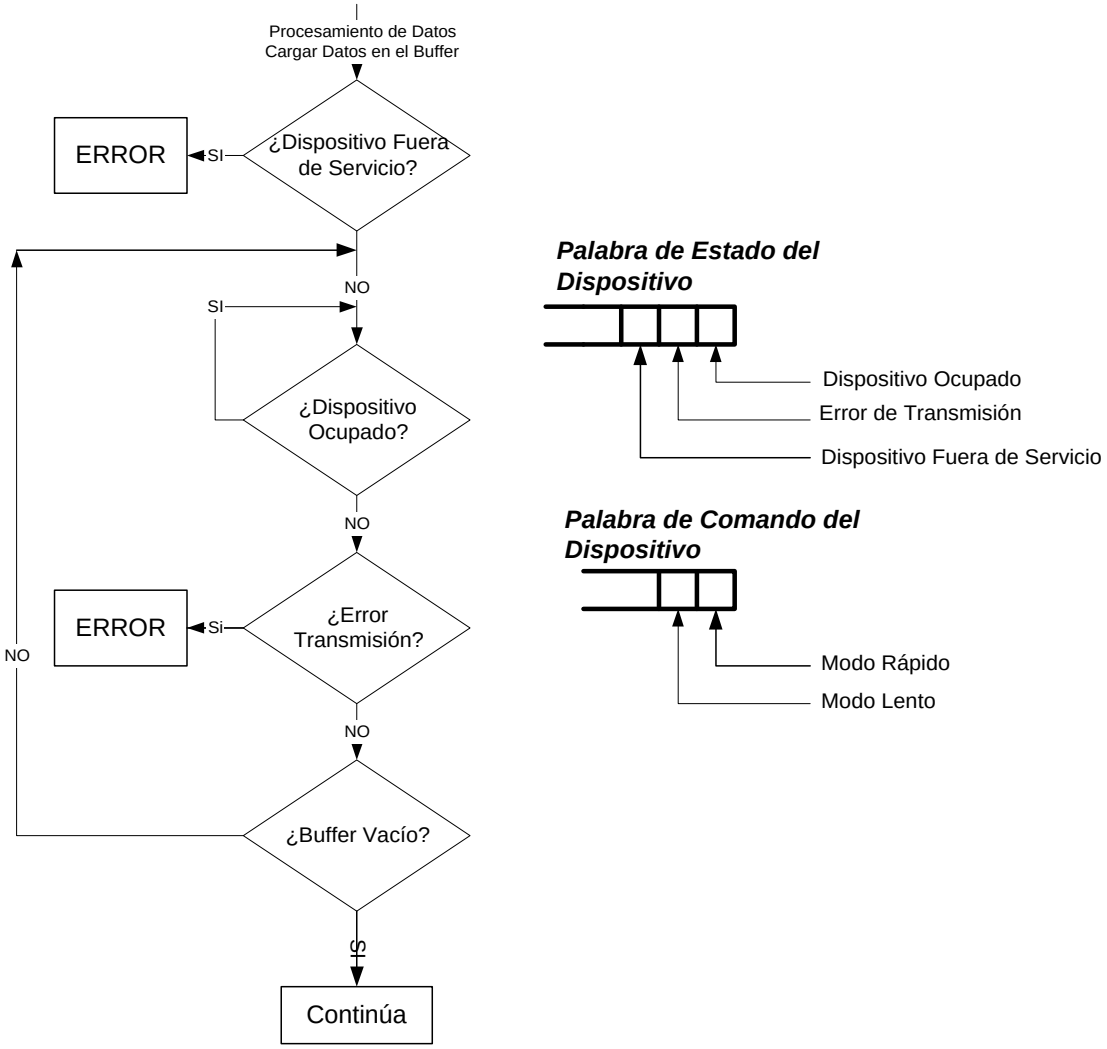

Fig. 9. E/S programada asincrónica

El programa correspondiente al diagrama de la Fig. 9, para el 8088 será:

```
 ;PROCESAMIENTO DE DATOS
;cargar datos en buffer
 MOV   SI, OFFSET BUFFER ;apunta al comienzo del buffer (bloque de datos)
  MOV   CX,NUM           ;carga longitud bloque de datos
  IN    AL,P2            ;lee palabra de estado del periférico
  TEST  AL,00000100B     ;verifica dispositivo en servicio
  JNZ   ERROR            ;si periférico fuera de servicio: salta a ERROR
  MOV   AL,01H           ;carga comando modo rápido
  OUT   P1,AL            ;envía comando
ESP:    IN AL,P2         ;lee estado
  TEST  AL,00000001B     ;verifica si está ocupado
  JNZ   ESP              ;si está ocupado salta a ESPera
  MOV   AL,BYTE PTR [SI] ;obtiene dato
  OUT   P3,AL            ;envía dato
  IN    AL,P2            ;lee estado
  TEST  AL,00000010B     ;verifica error transmisión
  JNZ   ERROR            ;si hubo error salta a ERROR
  INC   SI               ;apunta al siguiente dato
  LOOP  ESP              ;decr. contador de datos. Si quedan datos salta
                         ;a Espera
  ......                        ;continúa proceso de datos
```

Nótese que si bien, por distintas causas, se salta a una misma dirección de ERROR, dicha causa se puede reconocer analizando el contenido del registro AL.

3 E/S Controladas Por Interrupciones

Las microcomputadoras se comunican con el exterior por medio de los dispositivos de entrada y salida. Estos dispositivos pueden ser lentos en comparación con la elevada velocidad del microprocesador.

La mayor desventaja de las E/S controladas por programa surge de la necesidad de abandonar periódicamente la sección de procesamiento de datos en el programa principal para comprobar si algún dispositivo está esperando una transferencia de datos. Este procedimiento de prueba, que puede ocurrir aunque no haya dispositivo alguno preparado, puede malgastar un valioso tiempo de procesamiento.

El problema se resuelve en muchos microprocesador incorporando un ***Sistema De Interrupciones***, el cual permite a los dispositivos de E/S interrumpir la ejecución del programa principal cuando, y sólo cuando, estén preparados para la transferencia de datos En el más simple de los sistemas de interrupción se conecta solamente un dispositivo de E/S a una única ***Línea De Pedido De Interrupción***. La aparición de una señal en esta línea hace que el microprocesador inicie automáticamente la siguiente secuencia de operaciones :

Un ejemplo típico puede ser el teclado: entre las pulsaciones de cada tecla hay un lapso impredecible y dependiente del usuario. Sería impensable que el microprocesador se quedara esperando una nueva opresión sin realizar alguna actividad.

Para tratar de entender la operatoria nos vamos a permitir una serie de simpáticas analogías didácticas y veremos su correlato con nuestra actividad.

3.1 La señora y la torta.

En el Hogar	La microcomputadora
Imaginemos una ama de casa que esta preparando una torta para agasajar a unos amigos que llegarán como visitas a tomar el té. La señora esta con su receta de cocina mezclando y batiendo una torta. Periódicamente se acerca a la puerta a ver si ya han llegado las visitas. Si se acerca a la puerta muy frecuentemente desatiende la preparación de la torta. Si se preocupa fundamentalmente por la torta, desatiende la puerta y las visitas pueden llegar, esperar un tiempo y como no son atendidas, se retiran.	La microcomputadora esta realizando su programa principal. Periódicamente lee las líneas de estado de los periféricos a ver si hay alguno que necesite atención. Si nos dedicamos demasiado al programa principal, puede ser que algún periférico solicite atención pero no sea detectada o bien si le prestamos demasiada atención a los requerimientos de los periféricos, desatendamos las obligaciones del programa principal.
Solución: Colocamos un timbre, que las visitas emplearán para avisar que llegaron. Al producirse el timbrazo, la señora terminará la operación que estaba realizando marcará en la receta el próximo paso a efectuar.	**Solución:** El periférico es quien avisa al microprocesador que requiere atención. Al recibir el pedido de interrupción, el microprocesador termina la instrucción que estaba ejecutando y guardará en la pila la dirección física (CS:IP) de la próxima instrucción a ejecutar y los flags.
Problema: En el vecindario hay una gran cantidad de vendedores ambulantes que harán perder mucho tiempo a la dueña de casa con timbres que no corresponden al objetivo original de avisar la llegada de las visitas. **Solución:** Colocar un interruptor en serie con el pulsador del timbre que inhabilite el timbre hasta el entorno horario en que se espera que lleguen los visitantes. En ese momento se activará el timbre.	Las interrupciones se habilitarán y deshabilitarán por medio del flag I de la palabra de flags (que adrede no fue analizado en el capítulo 2). Se habilitarán por medio de la instrucción STI y se deshabilitarán por medio de la instrucción CLI. Después del Reset amanecen deshabilitadas.
A fin de no perder el control de la torta, al sonar el timbre, la dueña de casa hará ingresar a las visitas, las saludará y con las disculpas de cortesía del caso volverá a su	La rutina encargada de atender a las interrupción deberá ser lo más breve posible. Por ejemplo en el caso del teclado, la rutina

actividad principal (la torta)	de atención de ese pedido de interrupción solo almacenará fila y columna de la tecla oprimida. La identificación exhaustiva de la tecla y la acción a tomar con su opresión será definida por el programa principal.
La casa tiene varias puertas. Cada una se identificará con una campanilla con distinto sonido. De esa manera la dueña de casa sabrá si debe atender la puerta principal, la de servicio o abrir el portón del garage.	Cada periférico se identificará por medio de algún método. Esa identificación dará lugar a lo que se llamarán interrupciones vectorizadas. El periférico solicitante de interrupción colocará un byte que denominaremos tipo de interrupción.
Si la dueña de casa lo desea, mientras esta atendiendo a las visitas, el sodero puede llamar por la puerta de servicio. La señora ordenadamente dejará a las visitas y atenderá al sodero. Terminado lo cual volverá a las visitas y luego a la torta.	Las interrupciones adecuadamente habilitadas pueden anidarse, es decir que una fuente de interrupción interrumpa a otra rutina de atención de interrupción. Al terminar de ejecutarse la última, se volverá a la interrumpida y luego al programa principal.
Si suenan dos ó más timbres simultáneos, la señora podrá identificar cuales son de dos maneras distintas: a) Preguntando puerta por puerta si hay alguien allí ó b) Escuchando simultáneamente a todos los timbres y decidiendo luego cual atender.	En caso de que existan dos ó más fuentes de interrupción activas simultáneamente, se podrá interrogar periférico por periférico cuál pidió interrupción y el orden de pregunta marca la prioridad de las interrupciones (encuesta serie) o bien se leerán simultáneamente todas las líneas de pedido de interrupción (encuesta paralelo). La primera requiere menos hardware de implementación pero es más lenta de actuar. La segunda es más rápida pero el hardware es más complejo.

3.2 Secuencia del pedido de interrupción.

3.2.1 Pedido de interrupción.

El periférico por medio de una señal activa alta conectada a INTR, solicita atención al microprocesador.

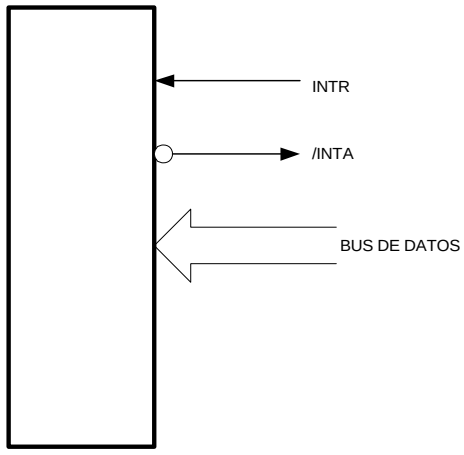

Fig. 10. Pedido de interrupción.

Si las interrupciones no están habilitadas, el pedido de interrupción no es detectado y por ende no se realiza acción alguna.

Si están habilitadas, el microprocesador termina de ejecutar la instrucción corriente (recordemos que una instrucción, una vez que comenzó a ejecutarse no puede ser interrumpida) y guarda en la pila IP,

CS correspondiente a la próxima instrucción a ejecutar y los flags. En total 6 bytes. Automáticamente s deshabilitan las interrupciones.

Nada impide que en la rutina de atención de interrupción se habiliten las interrupciones, permitiendo que otros dispositivos puedan, a su vez (si las prioridades lo permiten) interrumpir a la rutina en ejecución.

3.2.2 Identificación del periférico.

En respuesta a ese pedido de interrupción aceptado, el microprocesador generará un reconocimiento de interrupción por medio de la señal /INTA. Esta servirá para avisarle al periférico que su solicitud fue aceptada.

El periférico deberá colocar sobre el bus de datos un byte de identificación que se denominará "tipo" de la interrupción. Al tratarse de un byte, se dispondrán 256 tipos distintos de interrupciones.

La señal /INTA habilitará un buffer en cuyas entradas se ha cableado el tipo de interrupción que se desea generar.

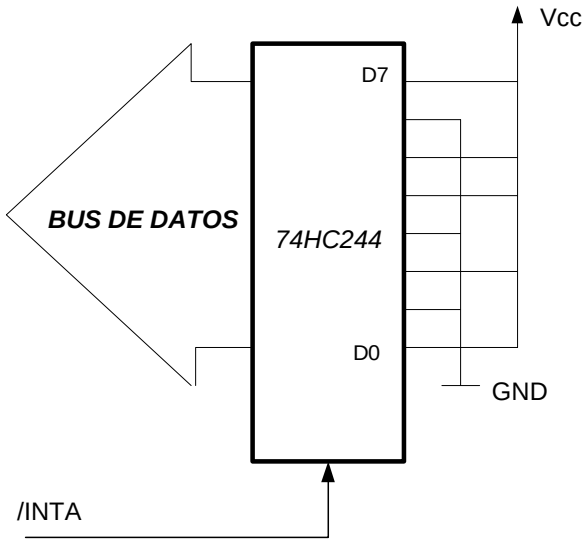

Fig. 11. Generación del tipo de interrupción.

En el ejemplo planteado en la Fig. 11, el tipo generado será B5H.

3.2.3 Ciclos de máquina INTA.

Lo descrito anteriormente se representa con dos ciclos de máquina de reconocimiento de interrupción en los que no se generan /RD pero si se generan /INTA.

En el segundo ciclo de /INTA se deberá colocar el tipo de interrupción. Si también se coloca en el primero, ese valor no es tenido en cuenta (lo que valida la utilización del circuito anterior).

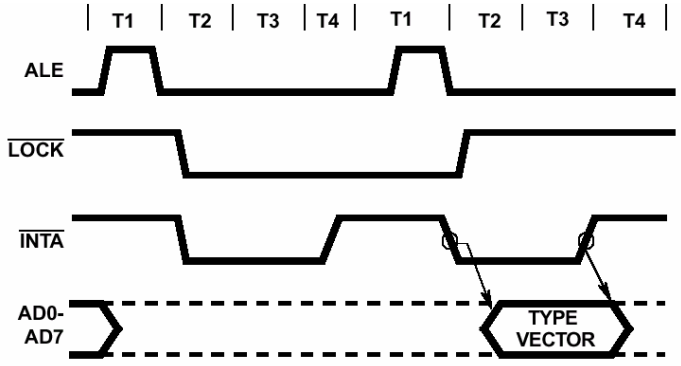

Fig. 12. Diagrama temporal del reconocimiento de interrupción.

3.2.4 Obtención de la dirección de la rutina de atención de interrupción.

En el proceso de inicialización del microprocesador, se deberá guardar en la ubicación adecuada la dirección de la rutina de atención de cada interrupción[2]. Dicha dirección estará compuesta por cuatro bytes que contendrán el CS (2 bytes) y el IP (2 bytes) de la dirección de la misma.

Como cada dirección ocupa 4 bytes, el microprocesador recibirá del periférico el tipo de interrupción, lo multiplicará por cuatro y retirará de una tabla ubicada a partir de la dirección 0000:000H la dirección de inicio de la rutina de atención de interrupción.

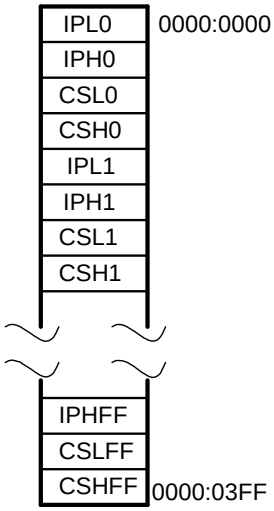

Fig. 13. Tabla de vectores de interrupción

Como disponemos de 256 tipos distintos de interrupción, la tabla de vectores de interrupción ocupará 256 x 4 bytes = 1kbyte.

En los procesadores posteriores (80386 y subsiguientes), la ubicación de esta tabla de vectores es programable.

3.2.5 Ejecución y Regreso de la Rutina de atención de interrupción.

En la rutina de atención de interrupción se podrán realizar operaciones de pila con libertad, pero debe garantizarse que en el momento de retorno, el puntero a la pila se halle en la ubicación adecuada para poder restaurar la dirección de la próxima instrucción que había sido guardada en el momento de la interrupción.

Si pensamos que la interrupción es asincrónica e imprevisible, los resultados de una rutina de atención de interrupción deben ser cuidadosamente planificados. No deberán afectar en forma permanente registros de propósito general que serían utilizados por el programa que estaba ejecutándose en el momento que se produjo la misma.

Imaginemos que un programa estuviera a punto de guardar una variable en memoria y una rutina de atención de interrupción inadecuadamente instrumentada modificara el puntero que estaba a punto de ser utilizado para tal acción.

En resumen:

- Se deberán resguardar todos los registros que sean utilizados en la rutina de interrupción y restaurarlos al valor original antes de retornar al programa original.
- Se deberá garantizar que en el momento de retorno el puntero a la pila se halla en el mismo lugar que estaba en el inicio de la rutina de atención de interrupción. O sea que deberán implementarse igual cantidad de POPs como de PUSHes que se hicieron.

Para retornar al programa que estaba en ejecución cuando apareció la interrupción, deberán recuperarse CS, IP y los flags (6 bytes en total). Ninguna de las instrucciones de retorno vistas hasta

[2] Lo que se implementó en Informática II como setvec()

ahora lo hacían (como máximo, los retornos FAR rescataban CS e IP de la pila), por lo que deberá implementarse una nueva instrucción de retorno llamada justamente retorno de interrupción **IRET.**

3.2.6 Distintos tipos de interrupción.

De los 256 vectores de interrupción posibles, algunos fueron utilizados por Intel desde la creación del 8088, otros fueron reservados para futuras aplicaciones y otros fueron declarados como disponibles por los usuarios.

- **Dentro de los utilizados por Intel, podemos destacar:**

- **Tipo 0:** Ocurre cuando se divide por cero o el cociente es mayor que el valor máximo que permite el destino.

- **Tipo 1:** Ocurre después de ejecutar una instrucción si TF (Trap Flag) vale 1. Esto permite la ejecución de un programa paso a paso, lo que es muy útil para la depuración de programas.

- **Tipo 2:** Ocurre cuando se activa la pata NMI (interrupción no enmascarable).

- **Tipo 3:** Existe una instrucción INT que ocupa un sólo byte, que es la correspondiente a este tipo. En los programas depuradores (debuggers) (tales como *Debug*, *CodeView*, *Turbo Debugger*, etc.), se utiliza esta instrucción como punto de parada (para ejecutar un programa hasta una determinada dirección, fijada por el usuario del depurador,. El depurador inserta esta instrucción en la dirección correspondiente a la parada y se comienza la ejecución. Cuando el CS:IP apunte a esta dirección se ejecutará la INT 3, lo que devolverá el control del procesador al depurador).

- **Tipo 4:** Ocurre cuando se ejecuta la instrucción de interrupción condicional INTO y el flag OF (Overflow Flag) vale 1.

- **Dentro de las reservadas:**

Los tipos 5 a 31 (1F en hexadecimal) están reservados para interrupciones internas (también llamados "excepciones") de futuros microprocesadores.

- **Disponibles para el usuario**

Sin restricciones, desde la 20H a FFH.

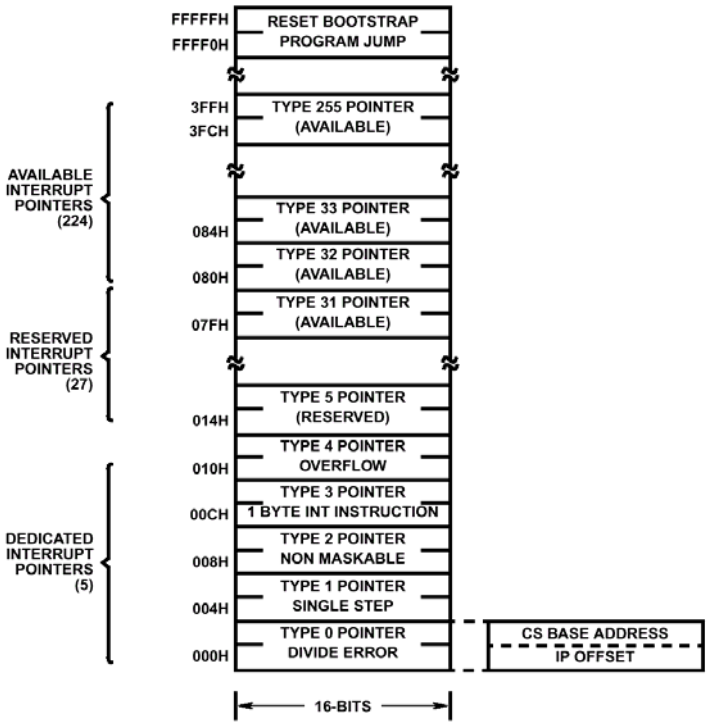

Fig. 14. Distintos tipos de interrupción.

3.3 Recapitulación y empleo de las interrupciones.

1. Cuando se produce la interrupción, el microprocesador observa si las interrupciones estan habilitadas.
2. En caso afirmativo, y si aparece un pedido de interrupción[3] completa la ejecución de la instrucción en curso y genera dos ciclos de INTA y obtiene del bus de datos el tipo de interrupción.
3. Salva en la pila el contenido de la dirección física CS:IP y los flags.
4. Automáticamente inhabilita las interrupciones
5. Multiplica el tipo de interrupción recibido en 2) por cuatro y apunta a una tabla en memoria a partir de la 0000:0000 donde se hallan las direcciones de las rutinas de atención de interrupción. Toma los nuevos valores de CS:IP de dicha tabla.

De este modo, el reconocimiento de una señal de pedido de interrupción provoca el salto, desde una línea del programa principal, a una predeterminada dirección en la memoria de programa (dirección de atención de la interrupción).

En esta dirección comenzará la subrutina que atenderá la correspondiente transferencia de datos entre el microprocesador y el dispositivo que le solicitó su atención. Dicha dirección podrá suministrarla el periférico, o generarse internamente en el microprocesador. Al finalizar la rutina de atención se deberán rehabilitar las interrupciones. Algunos microprocesadores lo realizan en forma automática con la instrucción de retorno.

Debe notarse que en algunos casos puede ser posible que se tarde más tiempo en atender un dispositivo mediante el sistema de interrupciones que por control por programa, debido a la ejecución de varias instrucciones (salto y salvaguarda de registros) antes de comenzar efectivamente la atención del mismo.

Las interrupciones se inhiben poniendo en "1" un bit de la máscara de interrupciones, interna al microprocesador. Este enmascaramiento se suele utilizar para evitar la interrupción de ciertas secciones de programa que se deben ejecutar en forma ininterrumpida (por ejemplo el procesamiento de un conjunto de datos que se debe completar antes de efectuar la siguiente operación de E/S).

Se puede notar aquí la similitud entre la secuencia correspondiente a la atención de una interrupción y la ejecución de una instrucción de salto a subrutina. En ambos casos se salvará el contenido del PC para poder regresar al programa principal una vez completada la correspondiente subrutina; pero hay, entre ambos casos, una diferencia importante: mientras el salto a subrutina por programa sucede en un momento previsto, con condiciones conocidas al momento de la programación, el salto a una subrutina de atención de una interrupción ***puede suceder en cualquier momento***, en forma totalmente asincrónica con la ejecución del programa principal. Por esto, en este último caso, la subrutina de atención de la interrupción deberá salvar los contenidos de los registros de trabajo del microprocesador (acumulador, registros índice, registro de flags...) que necesite emplear, para devolverlos luego, al programa principal, en las mismas condiciones en que se encontraban al momento de atender la interrupción, para no alterar el funcionamiento de dicho programa principal.

El 8088 conjuntamente con los registros IP y CS, salvan en forma automática, en la pila, el registro de flags de 16 bits, por lo que resulta salvaguardar en la pila 6 bytes[4]. Este almacenamiento, automático o no, hará que aumente el tiempo de respuesta de atención de la interrupción (tiempo entre el pedido de interrupción y la ejecución de la primer instrucción efectiva de atención).

---

[3] El pedido de interrupción consiste en que la pata de interrupción este activa desde el último ciclo de reloj de la instrucción previa hasta que

[4] Por tal motivo, la instrucción de retorno de la rutina de atención de interrupción no podrá ser la tradicional RET de las subrutinas estándares, sino que se deberá emplear una especial ***IRET***, que restaurará apropiadamente CS, IP y los flags.

Algunos microprocesador permiten obtener una respuesta mas rápida, mediante una arquitectura de registros especial: disponen de dos juegos de registros, uno para uso del programa principal y otro para las rutinas de interrupción. Otros utilizan como registros una zona de memoria, y disponen de un registro que funciona como puntero del espacio de trabajo, para definir la zona de memoria a utilizar.

La señal de interrupción carga en el registro puntero, antes de pasar a la subrutina, una dirección que define un nuevo grupo de registros de trabajo en la memoria. Al retornar de la subrutina se carga en dicho registro el valor original.

3.4 Software interrupts

Existen códigos de operación llamados software interrupts o INT. Los mismos consisten en intercalar una instrucción y luego un tipo de interrupción (valor de 00 a FFh) en forma similar a lo que sucedía con las interrupciones externas.

El comportamiento posterior es similar (se toma el tipo de interrupción y se lo multiplica por 4, se carga la dirección de la rutina de atención de interrupción, guarda CS, IP y flags en la pila, se deshabilitan las interrupciones, etc), salvo que no se generan externamente los ciclos de /INTA, sino que se continúa con la ejecución del programa, leyendo el tipo de interrupción desde la memoria de programa y no desde el periférico.

Imaginemos que desde el programa deseamos invocar a la interrupción tipo 21h

.....	**.....**
INT 21H	**CD**
....	**21**
	.....

La memoria de programa quedará como se ve a la derecha. Al encontrar el código de operación de INT (CD en hexadecimal), el procesador buscará ***de la memoria de programa*** (y no del periférico) el tipo de interrupción.

Se suele decir que las interrupciones de software son interrupciones sincrónicas pues siempre que se ejecute el tramo de programa se invocará a la rutina de atención de interrupción para diferenciarlas de las interrupciones de hardware en las que su aparición es imprevisible, ya que dependen del mundo exterior y por ende se las denomina asincrónicas.

Una de las principales aplicaciones de las interrupciones de software es la intercepción de las interrupciones en las que se emplea para invocar a la vieja rutina de atención de interrupción.[5]

El retorno de la rutina de atención de interrupción de software, se hará por medio de IRET.

3.5 Interrupciones no enmascarables (NMI)

Existe una pata del 8088 que fue pensada para situaciones catastróficas en las que no puede esperarse a que las interrupciones estén habilitadas o a que el periférico se identifique.

Estos casos de atención excepcional pueden ser error de paridad en la memoria o en una comunicación, detección de la caída de la tensión de alimentación, etc.

Cuando se produce una señal exterior de NMI (Non Maskable Interrupt) todo ocurre como si se generase una interrupción por software tipo 2, es decir que no se generan /INTAs.

La NMI es una interrupción que se activa por flanco creciente, es decir que mantenerla indefinidamente en estado alto NO vuelve a generar NMI y solo actúa por flanco.

[5] O a la nueva según la metodología utilizada.

3.6 Operación En Tiempo Real

En muchas aplicaciones se requiere una operación de E/S en un determinado instante, o en forma periódica con un intervalo definido. En estos casos la operación de los dispositivos de E/S y el control de transferencia de datos por parte del programa principal se deberá sincronizar en tiempo real.

La sincronización se implementa conectando un generador de pulsos externo, de frecuencia constante y conocida, a una línea de pedido de interrupción.

Estos pulsos interrumpen, a intervalos conocidos, la ejecución del programa principal. La operación de los dispositivos de E/S se puede sincronizar en tiempo real, contando los pedidos de interrupción, y controlando el flujo del programa en concordancia. A un oscilador, utlizado de este modo, se lo denomina **Reloj De Tiempo Real** (RTC). Generalmente consta de una cadena divisora alimentada por un oscilador basado en un cristal de alta frecuencia (por ej. 1 MHz).

Existen circuitos integrados que contienen contadores programables y se pueden utilizar como relojes de tiempo real, como por ejemplo el 82C54 de Intel o alguno de los que se emplean en las PC para llevar la fecha (Generadores de timer ticks).

3.6.1 Ejemplo De E/S Utilizando Un Reloj De Tiempo Real

Un sistema de adquisición de datos está basado sobre un conversor analógico-digital (CAD) multicanal, controlado por un microprocesador. Se utiliza un reloj de tiempo real de 50 Hz para controlar el período de muestreo. En esta aplicación se muestrean las señales analógicas de los canales 2 y 4 cada dos segundos, almacenándose en memoria los valores digitalizados.

3.7 Handshake

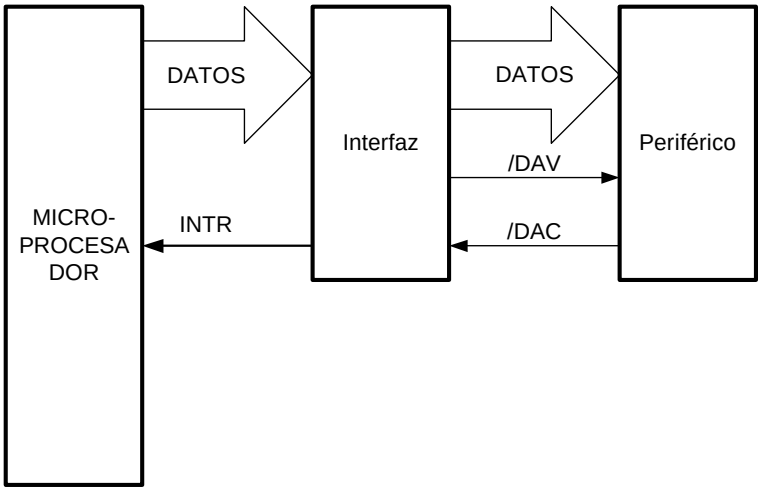

Fig. 15. Handshake

Handshake es un concepto directamente vinculado con la interrupción y consiste en que el procesador y el periférico se sincronizan en la transferencia de datos. Existirán handshake de entrada y de salida.

En la Fig. 15 vemos handshake de salida. El mismo consiste en colocar sobre un bus de baja velocidad (no es el bus de datos del procesador) datos a transferir a un periférico. Se avisa al periférico que los datos están disponibles por medio de la señal /DAV. En el momento en que el periférico tomó los datos, le informará a la interfaz ese hecho por medio de la señal /DAC. La interfaz le avisará al procesador que puede colocar un nuevo dato, generándole un pedido de interrupción. Habitualmente en la rutina de atención de interrupción se escribirá un nuevo dato sobre la interfaz que reiniciará el proceso.

3.8 El Controlador De Interrupciones 8259.

Las interrupciones añaden cierta complejidad al diseño del hardware: en principio, es necesario jerarquizarlas de alguna manera para decidir cuál se atiende en el caso de que se produzcan dos simultáneamente. También es importante el control de prioridad para el caso de que se produzca una interrupción mientras se está procesando otra: sólo se la atenderá si es de mayor prioridad.

Este circuito integrado está especialmente diseñado para controlar las interrupciones en sistemas basados en el 8080/8085 y en el 8086. Puede controlar hasta 8 interrupciones vectorizadas. Además, a un 8259 se le pueden conectar en cascada un máximo de 8 chips 8259 adicionales, lo que permite gestionar sistemas con hasta 64 interrupciones, como veremos.

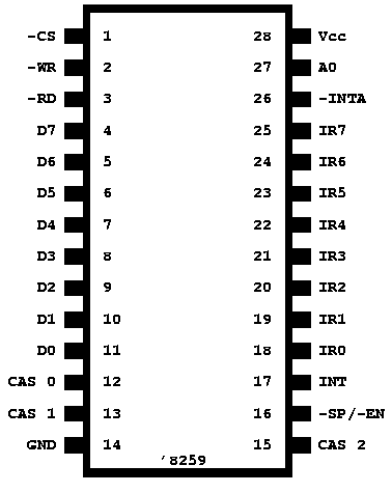

Fig. 16. Diagrama de Conexionado del 8259.

El significado e interpretación de las señales es el siguiente:

/CS:	Habilita la comunicación con el microprocesador.
/WR:	Permite al 8259 aceptar comandos del microprocesador.
/RD:	Permite al 8259 dejar la información en el bus de datos.
D7..D0:	Bus de datos bidireccional, por el que se transmite la información de control/estado y el número de vector de interrupción.
CAS0..CAS2:	Líneas de cascada, actúan como salida en el 8259 maestro y como entrada en los 8259 esclavos, en un sistema con varios 8259 interconectados, constituyendo un bus local.
/SP/EN:	Pin de doble función: en el *buffered mode* del 8259 actuará como -EN, para habilitar los buffers del bus; en el modo normal indicará si el 8259 es maestro o esclavo (-SP).
INT:	Conectado a la patilla INT de la CPU para producir la interrupción cuando llegue el momento.
IR0..IR7:	Líneas asíncronas de petición de interrupción. Una petición de interrupción se ejecuta manteniendo IR en alto hasta que se recibe el reconocimiento (modo por flancos) o simplemente poniendo en alto la línea IR (modo por niveles).
/INTA:	Línea de reconocimiento de interrupción, por medio de esta línea se fuerza al 8259 a depositar en el bus la información del vector de interrupción. INTA es independiente de -CS.
A0:	En conjunción con -CS, -WR y -RD es empleada para enviar las palabras de comando al 8259 y para solicitar información al mismo. Suele ir conectada a la línea A0 de la

CPU.

3.8.1 Descripción Funcional

El diagrama funcional del 8259, con la estructura interna de las diversas partes que lo componen, es el siguiente:

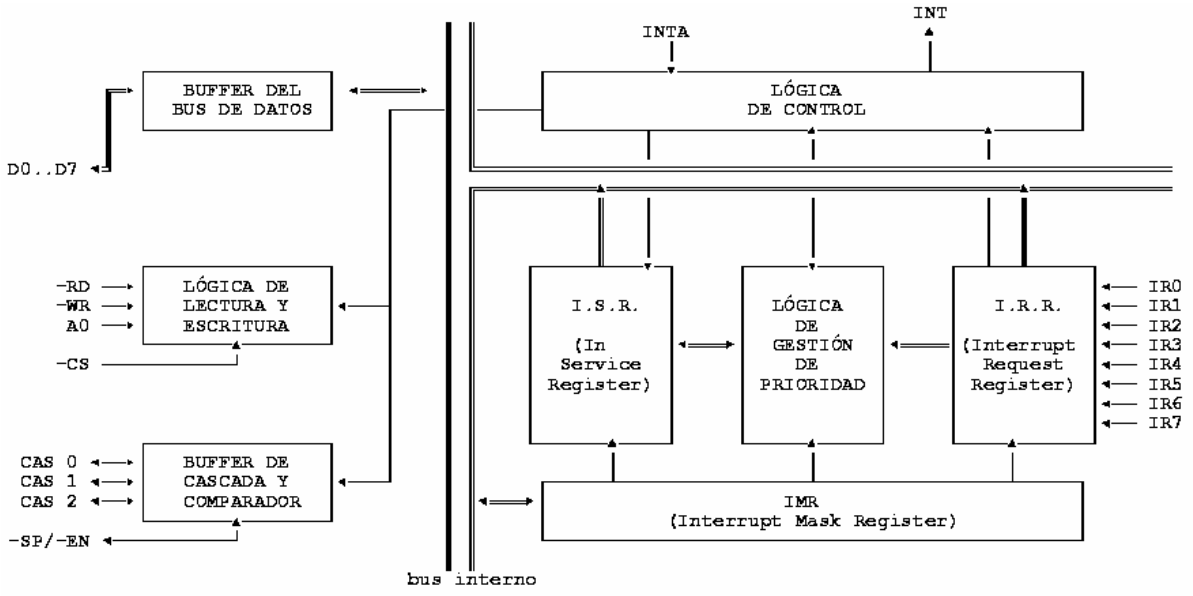

Fig. 17. Diagrama funcional del 8259.

Los principales registros internos del 8259 son el **IRR** (Interrupt Request Register) y el **ISR** (In Service Register). El IRR almacena todas las peticiones de interrupción pendientes; el ISR almacena todas las interrupciones que están siendo atendidas en un momento dado. La **lógica de gestión de prioridad** determina qué interrupción, de las solicitadas en el IRR, debe ser atendida primero: cuando lleguen las señales INTA dicha interrupción será la primera procesada y su bit correspondiente se activará en el ISR. El **buffer del bus de datos** conecta el 8259 con el bus de datos de la placa principal del ordenador: su diseño en 3 estados permite desconectarlo cuando sea necesario; a través de este bus circulan las palabras de control y la información de estado. La **lógica de lectura y escritura** acepta los comandos que envía la CPU: aquí hay registros para almacenar las palabras de inicialización y operación que envía el procesador; también sirve para transferir el estado del 8259 hacia el bus de datos. El **buffer de cascada/comparador** almacena y compara las identificaciones de todos los 8259 que posea el sistema: el 8259 maestro envía la identificación del 8259 esclavo en las líneas CAS, los 8259 esclavos la leen y el implicado en la operación coloca en el bus de datos la dirección (vector) de la rutina que atenderá la interrupción en los 2 próximos (o el próximo) ciclos INTA.

3.8.2 Funcionamiento Del 8259

Analizaremos ahora la operación del 8259 en conexión con un 8088.

1) Una o más líneas IR son activadas por los periféricos, lo que pone a 1 el correspondiente bit del IRR.
2) El 8259 evalúa la prioridad de estas interrupciones y solicita la interrupción a la CPU (línea INT) si es necesario.
3) Cuando la CPU reconoce la interrupción, envía la señal -INTA.
4) Nada más recibida la señal -INTA de la CPU, el 8259 activa el bit correspondiente a la interrupción de mayor prioridad (la que va a ser procesada) en el ISR y lo borra en el IRR. En este ciclo, el 8259 aún no controla el bus de datos.
5) Cuando la CPU envía un segundo ciclo -INTA, el 8259 deposita en el bus de datos un valor de 8 bits que indica el número de vector de interrupción del 8086, para que la CPU lo pueda leer.
6) En el modo AEOI del 8259, el bit de la interrupción en el ISR es borrado nada más acabar el

segundo pulso -INTA; en caso contrario, ese bit permanece activo hasta que la CPU envíe el comando EOI al final de la rutina que trata la interrupción (caso más normal).

Si en el paso (4), no está presente el pedido de interrupción (por ejemplo, porque ha sido excesivamente corta) el 8259 envía una interrupción de nivel 7 (si hubiera un 8259 conectado en IR7, las líneas CAS permanecerían inactivas y la dirección de la rutina de servicio de interrupción sería suministrada por el 8259 maestro).

3.8.3 Programación Del 8259

El 8259 acepta dos tipos de comandos generados por la CPU: los **ICW** (Inicialization Command Word) que inicializan el 8259, y los **OCW** (Operation Command Word) que permiten programar la modalidad de funcionamiento. Antes de que los 8259 de un sistema comiencen a trabajar deben recibir una secuencia de ICW que los inicialice. Los ICW y OCW constan de secuencias de 2 a 4 comandos consecutivos que el 8259 espera recibir secuencialmente, unos tras otros, a través del bus de datos, según sea necesario (el propio 8259 se encarga de contarlos midiendo los pulsos de la línea -WR). Los OCW pueden ser enviados en cualquier momento, una vez realizada la inicialización.

La comunicación con el 8259 emplea las líneas -WR y -RW, así como A0. El hecho de que exista una sola línea de direcciones implica que el 8259 sólo ocupa dos direcciones de puerto de E/S en el espacio de entrada y salida del ordenador.

3.8.3.1 ICWS (Inicialization Command Words).

ICW1: Cuando un comando es enviado con A0=0 y D4=1, el 8259 lo interpreta como la primera palabra de la inicialización (ICW1) e inicia dicha secuencia de inicialización, lo que implica lo siguiente:

- Se resetea el circuito sensible a los niveles, lo que quiere decir que hasta nueva orden las líneas IR serán sensibles por flancos de transición bajo-alto.
- Se limpia el IMR.
- A la línea IR7 se le asigna un nivel de prioridad 7.
- Se desactiva el *Special Mask Mode*. Se queda listo para devolver IRR en la próxima lectura OCW3.
- Si IC4 (bit D0) es 0, todas las funciones seleccionadas en ICW4 serán puestas a 0 (*non buffered mode*, no AEOI, sistema 8080/85) e ICW4 no será necesaria.

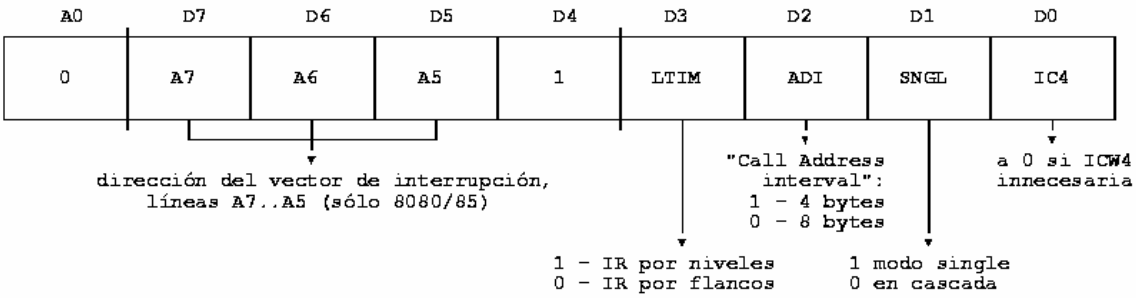

Fig. 18. ICW1

Notas: Si SNGL es 1 significa que el 8259 es único en el sistema y no será enviada ICW3. Si IC4 es 0, tampoco será enviada ICW4. En el 8080/85, las diversas interrupciones generan CALL's a 8 direcciones adyacentes separadas 4 u 8 bytes (según indique ADI): para componer la dirección, el 8259 inserta A0..A4 (o A0..A5) convenientemente, según la interrupción que se trate. En el 8086, A7..A5 y ADI son ignoradas.

ICW2: Se envía con A0=1, para diferenciarlo de ICW0 (hacer OUT a la siguiente dirección de puerto).

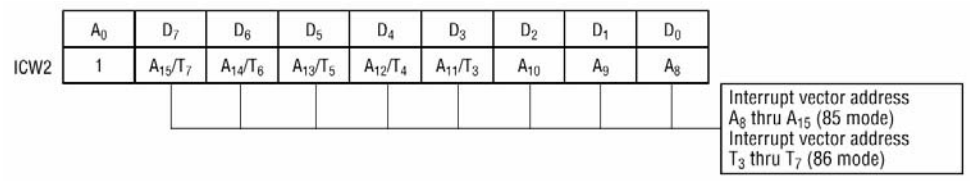

Fig. 19. ICW2

Notas: En el 8080/85, A15..A8 completan la dirección de la rutina de servicio; en el 8086, T7..T3 determinan los cinco bits más significativos del número de vector de interrupción a invocar (los 3 bajos los suministra el 8259 según la interrupción que se trate).

ICW3: Se envía sólo en el caso de que haya más de un 8259 en el sistema (bit SNGL de ICW1 a cero), en caso contrario en su lugar se enviaría ICW4 (si procede).

Formato de ICW3 a enviar a un 8259 maestro:

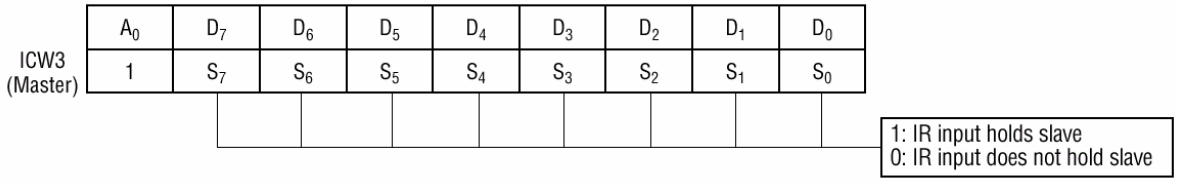

Fig. 20. ICW3 para el maestro.

Formato de ICW3 a enviar a un 8259 esclavo para que memorice de qué línea IR del maestro cuelga:

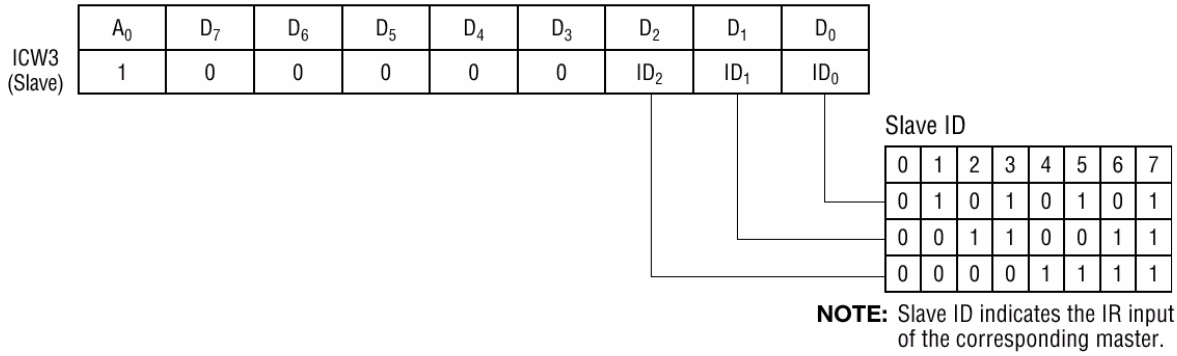

Fig. 21. ICW3 para el esclavo.

ICW4: Se envía sólo si IC4=1 en ICW1, con objeto de colocar el 8259 en un modo de operación distinto del establecido por defecto (que equivale a poner a cero todos los bits de ICW4).

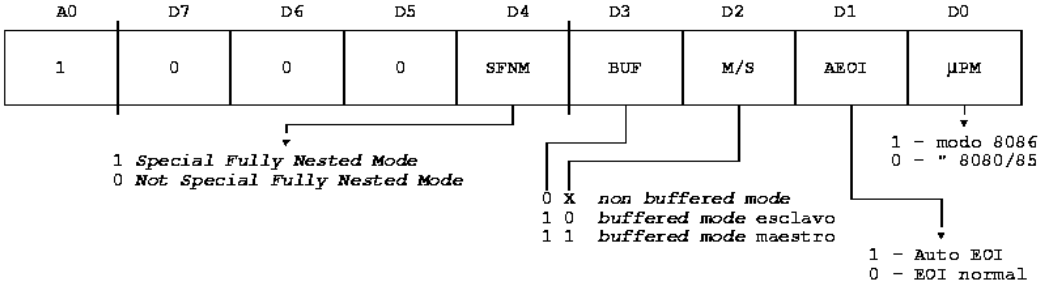

Fig. 22. ICW4

Notas: El Special Fully Nested Mode, el buffered mode y la modalidad AEOI serán explicadas más tarde. Nótese que con el 8086 es obligatorio enviar ICW4 para seleccionar esta CPU.

3.8.3.2 OCWS (Operation Command Words).

Una vez inicializado, el 8259 está listo para procesar las interrupciones que se produzcan. Sin embargo, durante su funcionamiento normal está capacitado para recibir comandos de control por parte de la CPU.

OCW1:

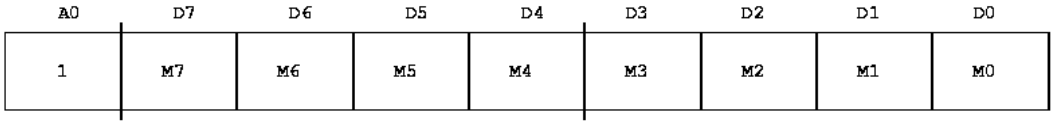

Fig. 23. OCW1

Este comando activa y borra bits en el IMR (Interrupt Mask Register). Los bits M0..M7 de OCW1 se corresponden con sus correspondientes bits del IMR. Un bit a 1 significa interrupción enmascarada (inhibida) y a 0, interrupción habilitada.

OCW2:

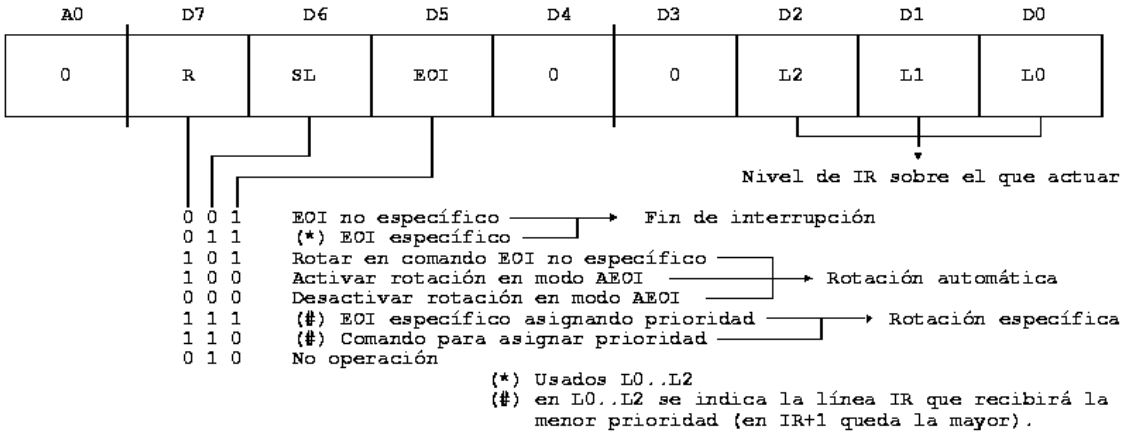

Fig. 24. OCW2.

OCW3:

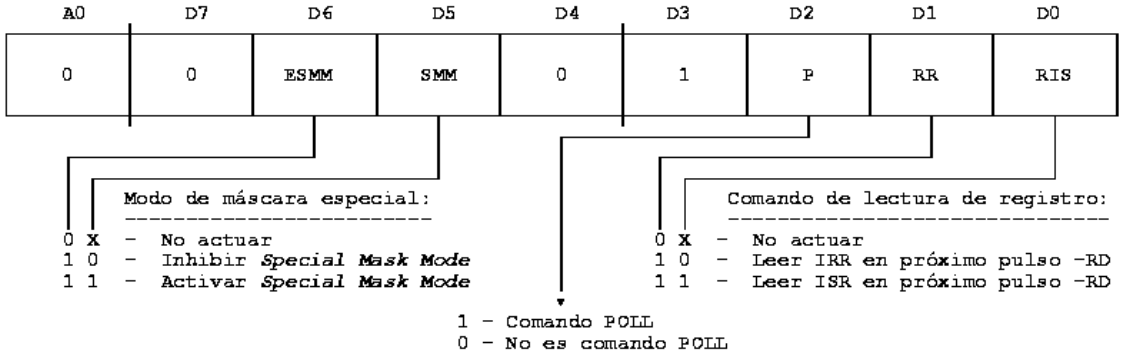

Fig. 25. OCW3

3.8.4 **Trabajando Con El 8259**

En las ICW y, sobre todo, en las OCW, se han introducido un aluvión de elementos nuevos que serán explicados a continuación.

3.8.4.1 Fully Nested Mode.

Por defecto, el 8259 opera en esta modalidad (modo de anidamiento completo), a menos que se le programe de otra manera. En este modo las interrupciones quedan ordenadas, por prioridades, de 0 (máxima) a 7 (mínima). Cuando se produce un reconocimiento de interrupción por parte de la CPU, el 8259 evalúa cual es la interrupción pendiente de mayor prioridad, coloca su número de vector en el bus y activa su bit correspondiente en el ISR. Este bit permanece activo hasta que el 8259 recibe el comando EOI (situación más normal); sin embargo, en el modo AEOI, ese bit se bajaría inmediatamente después del último -INTA. Mientras el bit del ISR esté activo, todas las interrupciones de igual o menor prioridad que lleguen permanecen inhibidas; sin embargo, las de mayor prioridad podrán interrumpir. En el caso del 8086, cuando comienza el tratamiento de la interrupción, un bit del registro de estado de la CPU mantiene inhibidas todas las interrupciones: lo normal es que el programa de control comience con STI para permitir que el 8086 envíe nuevas señales INTA al 8259, así el 8259 podrá enviar las interrupciones de mayor prioridad que le lleguen. Tras la secuencia de inicialización, las interrupciones quedan ordenadas de mayor (IR0) a menor prioridad (IR7), aunque este orden puede modificarse en la modalidad de prioridad rotatoria o con el comando de asignación de prioridad. Nótese que cuando se utiliza el modo AEOI o el *Special Mask Mode* no se respeta el modo *Fully Nested Mode* (debido a que una interrupción de menor prioridad podría interrumpir a una rutina que gestiona otra de mayor prioridad).

3.8.4.2 Special Fully Nested Mode.

Se emplea en sistemas que tienen varios 8259 conectados. Sólo el 8259 maestro es programado en este modo, lo que implica las siguientes diferencias respecto al *Fully Nested Mode* normal:

- Cuando se atiende una interrupción de un 8259 esclavo, si viene otra de mayor prioridad de ese mismo 8259 esclavo, se provoca una interrupción al maestro (normalmente, el 8259 esclavo estaría enmascarado mientras se procesa una de sus interrupciones).

- Cuando acaba la rutina de servicio de interrupción, hay que enviar un EOI no-específico al 8259 esclavo; además hay que leer a continuación su ISR y comprobar si es cero: en ese caso, hay que enviar además otro EOI al 8259 maestro (si no es cero significa que aún hay interrupciones en proceso en el 8259 esclavo).

3.8.4.3 Modos de EOI.

El EOI (End Of Interrupt) sirve para bajar el bit del ISR que representa la interrupción que está siendo procesada. El EOI puede producirse automáticamente (AEOI) al final de la última señal INTA que envía la CPU al 8259 para una interrupción dada (tercer ciclo INTA en el 8080/85 y segundo en el 8086); sin embargo, la mayoría de los sistemas requieren una gestión de prioridades en las interrupciones, lo que significa que es más conveniente que EOI lo envíe el propio procesador al 8259, a través de OCW2, cuando acabe la rutina de gestión de interrupción, para evitar que mientras se gestiona esa interrupción se produzcan otras de igual o menor prioridad. En un sistema con varios 8259, el EOI debe ser enviado no sólo al 8259 esclavo implicado sino también al maestro. Hay dos modalidades de EOI: la específica y la no-específica. En el EOI no específico, el 8259 limpia el bit más significativo que esté activo en el ISR, que

se supone que es el correspondiente a la última interrupción producida (la de mayor prioridad y que está siendo procesada). Esto es suficiente para un sistema donde se respeta el *Fully Nested Mode*. En el caso en que no fuera así, el 8259 es incapaz de determinar cuál fue el último nivel de interrupción procesado, por lo que la rutina que gestiona la interrupción debe enviar un EOI específico al 8259 indicándole qué bit hay que borrar en el ISR.

3.8.4.4 Rotación de prioridades.

Hay sistemas en que varios periféricos tienen el mismo nivel de prioridad, en los que no interesa mantener un orden de prioridades en las líneas IR. En condiciones normales, nada más atender una interrupción de un periférico, podría venir otra que también se atendería, mientras los demás periféricos se cruzarían de brazos. La solución consiste en asignar el menor nivel de prioridad a la interrupción recién atendida para permitir que las demás pendientes se procesen también. Para ello se envía un EOI que rote las prioridades: si, por ejemplo, se había procesado una IR3, IR3 pasará al menor nivel de prioridad e IR4 al mayor, quedando las prioridades ordenadas (de mayor a menor): IR4, IR5, IR6, IR7, IR0, IR1, IR2, IR3. Existe también una rotación específica de prioridades, a través de OCW2, que puede realizarse en un comando EOI o independientemente del mismo (comando para asignar prioridad).

3.8.4.5 Special Mask Mode.

Hay ocasiones en las que mientras se ejecuta una rutina de servicio de interrupción es necesario permitir que se produzcan ciertas interrupciones de menor prioridad en algunos momentos, o prohibirlo en otros, sin ser quizá interesante enviar el EOI antes de tiempo. Esto implica alterar la estructura normal de prioridades. La manera de realizar esto es activando el *Special Mask Mode* a través de OCW3 durante la rutina de servicio de interrupción (es más que conveniente inhibirlo de nuevo al final). Una vez activado este modo, el IMR indica qué interrupciones están permitidas (bit a 0) y cuáles inhibidas (bit a 1). Por ello, suele ser conveniente activar el bit del IMR correspondiente a la IR en servicio (para evitar que se produzca de nuevo cuando aún no ha sido procesada). Al final hay que enviar un EOI específico, ya que este modo de trabajo altera el *Fully Nested Mode* habitual.

3.8.4.6 Comando POLL.

En esta modalidad poco habitual, habilitada a través de OCW3, no se emplea la salida INT del 8259 o bien el microprocesador trabaja con las interrupciones inhibidas. El servicio a los periféricos es realizado por software utilizando el comando POLL. Una vez enviado el comando POLL, el 8259 interpreta la próxima lectura que se realice como un reconocimiento de interrupción, actualizando el ISR y consultando el nivel de prioridad. Durante esa lectura, la CPU obtiene en el bus de datos la palabra POLL que indica (en el bit 7) si hay alguna interrupción pendiente y, en ese caso, cuál es la de mayor prioridad (bits 0-2).

3.8.4.7 Lectura de información del 8259.

El IMR puede ser leído a través de OCW0; para leer el contenido del IRR y el ISR hay que emplear OCW3. Para estos dos últimos registros hay que enviar una OCW3 que elija el IRR o el ISR; a continuación se puede leer el bus de datos (A0=0) sin necesidad de enviar más OCW3 (el 8259 es capaz de recordar si tiene que leer el IRR o el ISR). Esto último no es así, evidentemente, en el caso de utilizar el comando POLL (tras enviarlo, la próxima lectura se interpreta como un INTA). Tras inicializarse, el 8259 queda preparado por defecto para devolver IRR a la primera lectura.

3.8.4.8 Buffered Mode.

Al emplear el 8259 en grandes sistemas, donde se requieren buffers en los buses de datos, si se va a emplear el modo cascada existe el problema de la habilitación de los buffers. Cuando se programa el modo buffer, la patilla -SP/-EN del 8259 actúa automáticamente como señal de habilitación del los buffers cada vez que se deposita algo en el bus de datos. Si se programa de esta manera el 8259 (bit BUF de ICW4) será preciso distinguir por software si se trata de un 8259 maestro o esclavo (bit M/S de ICW4).

3.8.4.9 El 8259 Dentro De La Microcomputadora.

Las PC/XT vienen equipados con un 8259 conectado a la dirección base E/S 20h; este controlador de interrupciones es accedido, por tanto, por los puertos 20h (A0=0) y 21h (A0=1). En los AT y máquinas superiores, adicionalmente, existe un segundo 8259 conectado en cascada a la línea IR2 del primero. Este segundo controlador es accedido a través de los puertos 0A0h y 0A1h. La BIOS del ordenador, al arrancar la máquina, coloca la base de interrupciones del primer controlador en 8, lo que significa que las

respectivas IR0..IR7 están ligadas a los vectores de interrupción 8..15; el segundo 8259 de los AT genera las interrupciones comprendidas entre 70h y 77h. La asignación de líneas IR para los diversos periféricos del ordenador es la siguiente (por orden de prioridad):

```
            IRQ 0           Temporizador                        (INT 08h)
            IRQ 1           Teclado                             (INT 09h)
            IRQ 2           En los PC/XT: canal E/S             (INT 0Ah)
                 IRQ  8     Reloj de tiempo real                (INT 70h) -+
                 IRQ  9     Simulación de IRQ2                  (INT 71h)  |
                 IRQ 10     Reservado                           (INT 72h)  |
                 IRQ 11     Reservado                           (INT 73h)  |
                 IRQ 12     Reservado                           (INT 74h)  |>
Sólo AT y PS/2
                 IRQ 13     Coprocesador aritmético             (INT 75h)  |
                 IRQ 14     Controlador de disco duro           (INT 76h)  |
                 IRQ 15     Reservado                           (INT 77h) -+
            IRQ 3           COM2                                (INT 0Bh)
            IRQ 4           COM1                                (INT 0Ch)
            IRQ 5           Disco duro PC/XT (LPT2 en el AT)    (INT 0Dh)
            IRQ 6           Controlador de disquetes            (INT 0Eh)
            IRQ 7           LPT1                                (INT 0Fh)
```

En las AT, la línea IR2 del 8259 maestro es empleada para colgar de ella el segundo 8259 esclavo. Como la línea IR2 está en el slot de expansión de 8 bits, por razones de compatibilidad los AT tienen conectado en su lugar la IR9 que simula la IR2 original. Cuando se produce una IR9 debido a un periférico de XT que pretendía generar una IR2, el AT ejecuta una rutina de servicio en INT 71h que salta simplemente a la INT 0Ah (tras enviar un EOI al 8259 esclavo).

La colocación de IRQ0-IRQ7 en el rango INT 8-INT 15 fue bastante torpe por parte de IBM, al saltarse la especificación de Intel que reserva las primeras 32 interrupciones para el procesador. En modo protegido, algunas de esas excepciones es estrictamente necesario controlarlas. Por ello, los sistemas operativos que trabajan en modo extendido y ciertos extensores del DOS (como las versiones 3.x de WINDOWS) se ven obligados a *mover* de sitio estas interrupciones. En concreto, WINDOWS 3.x las coloca en INT 50h-INT 57h (por software, las máquinas virtuales 8086 emulan las correspondientes INT 8-INT 15). Además, en el modo protegido del 286/386 (o el virtual-86 del 386) la tradicional tabla de vectores de interrupción es sustituida por otra de descriptores, aunque el funcionamiento global es similar.

La interrupción no enmascarable del 80x86 no está controlada por el 8259: es generada por la circuitería que controla la memoria si se detecta un error de paridad. La interrupción no enmascarable puede ser enmascarada en los ordenadores compatibles gracias a la circuitería de apoyo al procesador, aunque no es frecuente; en los AT el bit 7 del puerto 70h controla su habilitación (si es cero, la NMI está habilitada) sin embargo también se podría inhibir el control de paridad directamente (activando los bits 2 y 3 de la dirección E/S 61h, respetando el resto de los bits de ese puerto por medio de una lectura previa). En los PC/XT, es el puerto 0A0h el que controla la habilitación de la NMI, también con el bit 7 (con la diferencia de que debe estar a cero para inhibirla).

Durante la inicialización del ordenador, la BIOS envía sucesivamente al 8259 las palabras ICW1 a ICW4 de la siguiente manera (listado extraído directamente de la BIOS):

```
                            ; Inicialización del 8259 maestro (XT/AT)
        MOV     AL,10001b   ; funciona por flancos, cascada, ICW4 necesaria
        OUT     20h,AL      ; enviar ICW1
        JMP     SHORT $+2   ; estado de espera para E/S
        MOV     AL,8        ; base de interrupciones en INT 8
        OUT     21h,AL      ; enviar ICW2
        JMP     SHORT $+2
        MOV     AL,4        ; hay un esclavo en IR2 (S2=1)
        OUT     21h,AL      ; enviar ICW3
```

```
        JMP     SHORT $+2
        MOV     AL,1            ; modo 8086, EOI normal, not buffered mode
        OUT     21h,AL          ; enviar ICW4: completa inicialización 8259-1
        JMP     SHORT $+2
        MOV     AL,255
        OUT     21h,AL          ; enmascarar todas las interrupciones

        JMP     SHORT $+2       ; Inicialización del 8259 esclavo (sólo AT)
        MOV     AL,10001b       ; funciona por flancos, cascada, ICW4 necesaria
        OUT     0A0h,AL         ; enviar ICW1
        JMP     SHORT $+2
        MOV     AL,70h          ; base de interrupciones en INT 70h
        OUT     0A1h,AL         ; enviar ICW2
        JMP     SHORT $+2
        MOV     AL,2            ; es el esclavo conectado a IR2
        OUT     0A1h,AL         ; enviar ICW3
        JMP     SHORT $+2
        MOV     AL,1            ; modo 8086, EOI normal, not buffered mode
        OUT     0A1h,AL         ; enviar ICW4: completada la inicialización
                                ; del 8259-2
        JMP     SHORT $+2
        MOV     AL,255
        OUT     0A1h,AL         ; enmascarar todas las interrupciones
```

Como se puede observar, la rutina de arriba enmascara todas las interrupciones a través del IMR. El objetivo de esta medida es evitar que se produzcan interrupciones antes de desviar los correspondientes vectores, pudiendo incluso mientras tanto estar habilitadas las interrupciones con STI.

Cuando se produce una interrupción de la CPU (bien por software o por hardware), el indicador de interrupciones del registro de estado del 8086 se activa para inhibir otra posible interrupción mientras se procesa esa (la instrucción IRET recuperará los flags del programa principal devolviendo las interrupciones a su estado previo). Lo normal suele ser que las rutinas que gestionan una interrupción comiencen por un STI con objeto de permitir la generación de otras interrupciones; las interrupciones sólo deben estar inhibidas en brevísimos momentos críticos. Sin embargo, cuando se procesa una interrupción hardware, el registro de interrupciones activas (ISR) indica qué interrupción en concreto está siendo procesada; si en ese momento llega otra interrupción hardware de menor o igual prioridad le será denegada la petición, si es de mayor prioridad le será concedida (si la rutina comenzaba por STI). Cuando acaba de procesarse la interrupción hardware, la instrucción IRET no le dice nada al 8259, por lo que el programador debe preocuparse de borrar el ISR antes de acabar. Si, por ejemplo, se gestiona la interrupción del temporizador sin limpiar al final el ISR, a partir de ese momento quedarán bloqueados el teclado, los discos ... Conviene aquí señalar que una rutina puede apoyarse en una interrupción hardware sin necesidad de reprogramarla por completo.

Ejemplo:

```
        STI
        PUSH    AX
        IN      AL,puerto_teclado
        CALL    anterior_int9
        ;
        ;       procesar tecla
        ;
        POP     AX
        IRET
```

Al producirse la INT 9 se lee el código de barrido de la tecla y luego se llama a la rutina que gestionaba con anterioridad a ésta la INT 9: ella se encargará de limpiar el ISR que, por tanto, no es tarea de nuestra rutina. Si hubiera que limpiar el ISR, bastaría con un EOI no específico (OCW2: enviar un valor 20h al puerto 20h para el 8259-1 y al puerto 0A0h para el 8259-2; en las IRQ8-IRQ15 hay que enviar el EOI a ambos controladores de interrupción).

Aviso: Aunque el funcionamiento del 8259 es suficientemente lógico como para pasar casi inadvertido, hay veces en que hay que tenerlo en cuenta. Por ejemplo, al utilizar el servicio 86h de la INT 15h del AT (con objeto de hacer retardos) desde una interrupción hardware comprendida entre IRQ 0 e IRQ 7, conviene limpiar el ISR antes de llamar: no basta con hacerlo al final de la rutina. La causa es que la BIOS utiliza las interrupciones asociadas al reloj de tiempo real para hacer el retardo, y en algunas máquinas es poco precavido y no limpia el ISR al principio, lo que deja totalmente bloqueado el ordenador.

3.9 Ejemplo: Cambio De La Base De Las Interrupciones.

La siguiente utilidad reprograma el 8259 maestro para desviar las INT 8-INT 15 a los nuevos vectores INT 50h-INT 57h (que invocan a los originales, para que el sistema siga funcionando con normalidad). Esta nueva ubicación no ha sido elegida por capricho, y es la misma que emplea WINDOWS 3.x. La razón es que el 386 trabaja normalmente en modo virtual-86 bajo MS-DOS 5.0; cuando se produce una interrupción se ejecuta una rutina en modo protegido. El EMM386 del MS-DOS 5.0 no está preparado para soportar las IRQ0-IRQ7 en otra localización que no sea la tradicional INT 8-INT 15 ó en su defecto INT 50h-INT 57h (por compatibilidad con WINDOWS). Con el QEMM386 o, simplemente, sin controlador de memoria expandida instalado, no habría problemas y se podría elegir otro lugar distinto. Por cierto: si se entra y se sale de WINDOWS, la nueva localización establecida, ya sea en 50h o en otro sitio, deja de estar vigente: esto significa que WINDOWS reprograma la interrupción base al volver al DOS. Personalmente he comprobado que aunque IRQDEMO fuera más elegante (empleando funciones de la especificación VCPI), nuestro querido WINDOWS no lo sería: ¡para qué molestarse!. Sin embargo, IRQDEMO sí se toma la molestia de comprobar si la máquina es un XT o un AT para enviar correctamente la ICW3 del 8259.

```
; ********************************************************************
; *  IRQDEMO.ASM   -  Utilidad residente de demostración, que desvía *
; *                   las interrupciones hardware INT 8-INT 15 hacia *
; *                   los vectores INT 50h a INT 57h.                *
; ********************************************************************

irqdemo         SEGMENT
                ASSUME CS:irqdemo, DS:irqdemo

                ORG  100h
inicio:
                JMP   main

; ------------ Area residente

irq0:           INT   8                 ; simular IRQ's normales (se
                IRET                    ; podría aprovechar también
irq1:           INT   9                 ; para hacer algo más útil).
                IRET
irq2:           INT   10
                IRET
irq3:           INT   11
                IRET
irq4:           INT   12
                IRET
irq5:           INT   13
                IRET
irq6:           INT   14
                IRET
irq7:           INT   15
                IRET

tam_resid       EQU   ($-OFFSET inicio+256+15)/16

; ------------ Código de instalación

main            PROC
                LEA   BX,tabla_ints
                MOV   AL,50h            ; nueva base para IRQ's 0-7
otra_int:       PUSH  AX
                PUSH  BX
                MOV   AH,25h
                MOV   DX,[BX]
                INT   21h               ; desviar INT 50h-57h
                POP   BX
                ADD   BX,2
                POP   AX
                INC   AL
                CMP   AL,58h
```

```
                JB    otra_int
                CALL  es_AT?
                MOV   BL,4
                MUL   BL
                MOV   BL,AL             ; BL = 4 en AT y 0 en PC/XT
                CALL  inic_8259
                LEA   DX,texto_txt
                MOV   AH,9
                INT   21h               ; mensaje de instalación
                MOV   ES,ES:[2Ch]
                MOV   AH,49h
                INT   21h               ; liberar entorno
                MOV   AH,31h
                MOV   DX,tam_resid
                INT   21h               ; terminar residente
main            ENDP

; ------------ Subrutinas de apoyo a la instalación.

inic_8259       PROC                    ; Inicialización 8259 maestro
                MOV   AL,0FFh
                OUT   21h,AL            ; enmascarar todas las IRQ
                JMP   SHORT $+2
                MOV   AL,10001b         ; flancos, maestro, sí ICW4
                OUT   20h,AL            ; enviar ICW1
                JMP   SHORT $+2         ; estado de espera E/S
                MOV   AL,50h            ; base interrupciones INT 50h
                OUT   21h,AL            ; enviar ICW2
                JMP   SHORT $+2
                MOV   AL,BL             ; 4 en AT y 0 en PC/XT
                OUT   21h,AL            ; enviar ICW3
                JMP   SHORT $+2
                MOV   AL,1              ; modo 8086, EOI normal
                OUT   21h,AL            ; enviar ICW4
                JMP   SHORT $+2
                MOV   AL,0
                OUT   21h,AL            ; permitir todas las IRQ
                RET
inic_8259       ENDP

es_AT?          PROC                    ; comprobar si es XT ó AT
                PUSHF
                POP   AX
                AND   AX,0FFFh
                PUSH  AX
                POPF
                PUSHF
                POP   AX
                AND   AX,0F000h
                CMP   AX,0F000h
                MOV   AX,1              ; indicar AT
                JNE   es_AT
                DEC   AX                ; indicar PC/XT
es_AT:          RET
es_AT?          ENDP

tabla_ints      DW    irq0, irq1, irq2, irq3, irq4, irq5, irq6, irq7
texto_txt       DB    13,10,"Las interrupciones 8-15 son ahora 50-57h."
                DB    13,10,"$"

irqdemo         ENDS
                END   inicio
```

3.9.1 Sistemas Con Interrupciones Múltiples

El sistema descripto en el ejemplo anterior es apropiado para el caso en que hay un solo dispositivo de E/S capaz de generar interrupciones. Muchos sistemas requieren varias fuentes de interrupción, las que a su vez pueden generar interrupciones de distinta naturaleza. Se pueden definir tres tipos básicos de interrupciones:

1. Interrupciones externas generadas por uno o más dispositivos de E/S.

2. Interrupciones internas generadas por el mismo sistema para indicar la ocurrencia de alguna condición particular (error en la transmisión, falla de la alimentación).

3. Interrupciones simuladas generadas por software (como ayuda en la depuración de programas o pruebas de rutinas de interrupción).

En general las diferentes fuentes de interrupción requieren distintas rutinas de atención. Además algunas pueden requerir una atención inmediata, en cambio otras podrán o deberán esperar que se complete la tarea en curso. Podrán, si su atención no es estrictamente impostergable y deberán si la tarea en curso debe completarse antes de la atención (ej. cálculos de resultados a transferir). El procedimiento de atención de interrupciones deberá :

1. Diferenciar, entre varias, a la fuente de interrupción.

2. Determinar el orden en que se atenderán las interrupciones, en el caso de haber varios dispositivos que pidan atención simultáneamente.

3. Salvar y reponer el contenido de los registros del microprocesador para asegurar la continuidad del programa durante las atenciones de interrupciones múltiples.

Reconocimiento de la fuente de interrupción: algunos microprocesador tienen varias líneas de interrupción, cada una de las cuales lleva asociada una dirección de atención distinta. Si se asigna una única fuente de interrupción a cada línea el problema queda resuelto. Además permite diferenciar entre interrupciones internas, externas y simuladas. Sin embargo en muchos casos varios dispositivos comparten una misma línea. Para identificar, en este último caso, la fuente de interrupción se utilizan tres procedimientos básicos :

3.9.2 Encuesta de dispositivos (polling).

Si podemos implementar un sistema con Interrupciones vectorizadas se conectarán varios dispositivos a una misma línea de interrupción, a través de una lógica externa. Cuando un dispositivo pide atención, dicha lógica genera el pedido de interrupción, suministrando al procesador una dirección que corresponde a la rutina de atención de quien lo solicitó. Las direcciones correspondientes a cada dispositivo suelen ubicarse en forma secuencial, en la memoria, formando lo que se denomina ***vector de interrupción***. En cada una de estas posiciones hay una instrucción de salto a la dirección de comienzo de la correspondiente subrutina. Si hay, simultáneamente más de un pedido de atención, la prioridad la establecerá, por hardware, la lógica externa. Esto puede realizarse mediante un codificador con prioridad, quien indicará al microprocesador el código (tipo) del dispositivo que pidió atención.

Supongamos que disponemos de un sistema de interrupciones sencillo en el que no hemos podido asignar distintos tipos de interrupciones a cada periférico. Al producirse un pedido de interrupción se deberá analizar quien fue el gestor de ese pedido de interrupción.

Existen dos alternativas:

3.9.2.1 Encuesta serie.

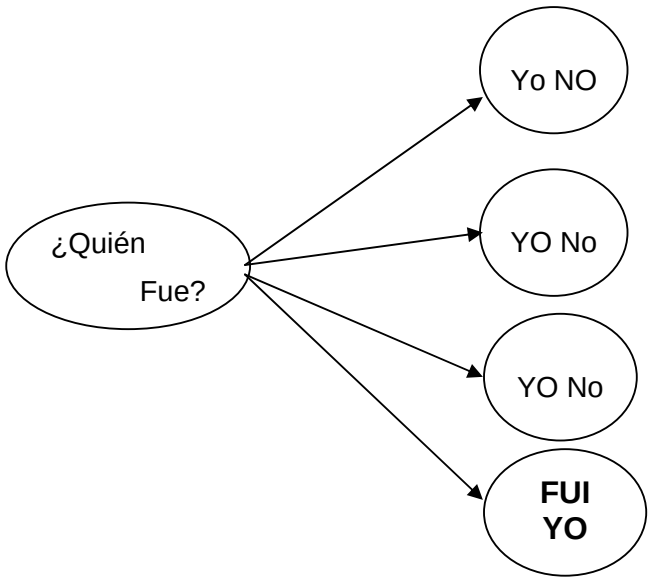

Fig. 26. Encuesta Serie

En la rutina de atención de interrupción se irá encuestando en forma sucesiva a cada uno de los periféricos para que de alguna manera informen si fue ese dispositivo quien pidió interrupción.

Normalmente por medio de una línea de estado se irá informando si ese dispositivo fue quine pidió interrupción. Al ir preguntando de a uno por vez, se genera la idea de encuesta serie (uno a continuación del otro) y la prioridad de las interrupciones estará dada por el orden en que se interroguen los dispositivos. El dispositivo de mayor prioridad será interrogado en primer lugar.

3.9.2.2 Encuesta paralelo

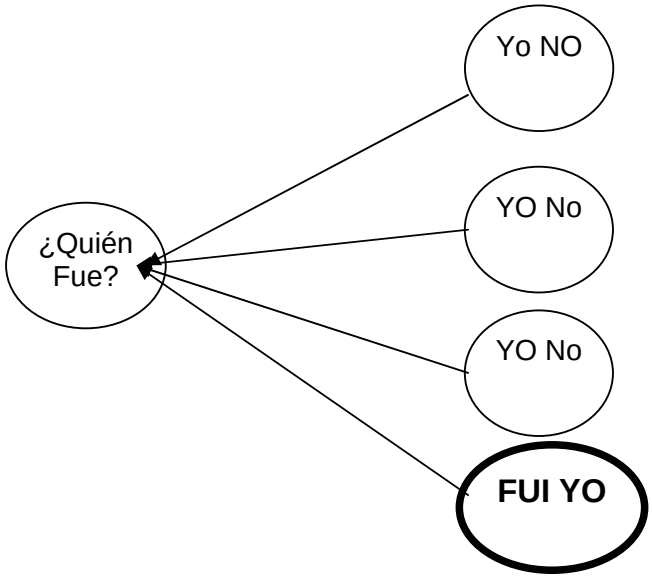

Fig. 27 Encuesta Paralelo

En este caso, se envía en forma simultánea el pedido de que todos los dispositivos coloquen su estado de interrupción de manera que inmediatamente el procesador pueda analizar quien pidio interrupción.

En este caso, habitualmente cada dispositivo pone en una línea del bus de datos su estado de interrupción y el procesador irá rotando el valor leído, en el orden de las prioridades decrecientes (primero se analizará al dispositivo de mayor prioridad y luego y sucesivamente a los de menor).

El sistema con encuesta paralelo es más rápido pero algo más complejo de implementar.

1. Priorización por hardware (DAISY CHAIN): en este caso el sistema de control de interrupciones del microprocesador envía una señal para controlar la lógica de pedido de interrupción de cada uno de los dispositivos de E/S. Esta señal pasa, ***por turno***, por cada uno de los dispositivos. Cuando la señal llega a un dispositivo que no ha solicitado atención, pasa directamente al siguiente. Cuando llega a uno que está esperando atención, éste bloquea el paso de la señal a los siguientes (de menor prioridad),y genera su propio pedido de atención. La posición de un dispositivo a lo largo de la línea de control (cadena), determina su prioridad. Cuando varios aguardan atención, el primero en recibir la señal será el primero que se atienda.

Estos métodos permiten establecer una prioridad de atención antes de entrar en la rutina de atención, pero no permiten que la rutina de atención de una interrupción, una vez comenzada, pueda ser interrumpida por una de mayor prioridad, y no por otra de menor prioridad.

4 Uso De La Línea Ready

La línea READY tiene como finalidad sincronizar la operación del procesador con la de periféricos de menor velocidad. El periférico maneja la línea, "frenando" al procesador para adecuarlo a su menor velocidad de transferencia de información.

En el ejemplo anterior, el procesador ordena al CAD iniciar la conversión, enviándole una palabra de comando, y luego entra en un lazo de espera, hasta que el CAD coloque su palabra de estado diciendo: fin de conversión.

Para el 8088 esta secuencia se podría implementar con las instrucciones:

```
        OUT     CONV,AL
LAZO:   IN      AL,ESTADO
        TEST    AL,MASC
        JNZ     LAZO
        IN      AL,DATO
```

donde ESTADO es la dirección de una puerta sobre la cual el CAD indica al procesador su estado y MASC una máscara que se corresponde bit a bit con el estado del CAD cuando indica fin de conversión. Mientras la palabra de estado sea distinta de MASC se repetirá el lazo de espera. El método es útil con periféricos relativamente rápidos, donde el lazo mencionado no se repetirá más de dos o tres veces. De lo contrario se desaprovecharía demasiado tiempo de procesador.

Otro método, más rápido y por lo tanto más eficiente, para implementarlo es mediante el uso de la línea READY. En este caso el procesador ejecutará

```
OUT     CONV,AL
IN      AL,DATO
```

donde CONV es la palabra de comando para iniciar la conversión y DATO el resultado de la medición. En este caso el CAD deberá detener al procesador hasta finalizar la conversión.

En la Fig. 28 se puede observar el caso en que el procesador es frenado hasta que el conversor termine su operación

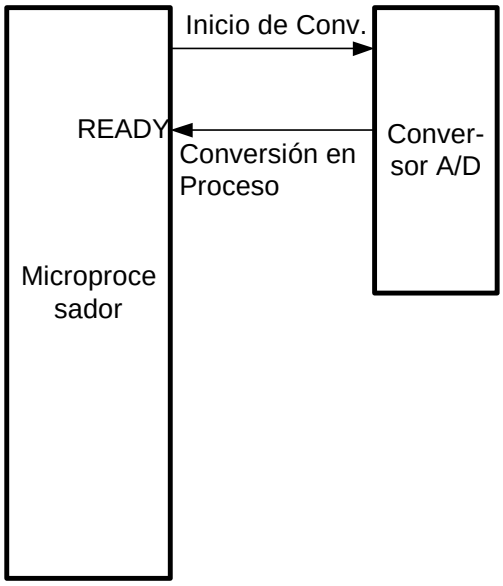

Fig. 28. **Circuito de sincronización por Ready**

5 E/S Con Acceso Directo A Memoria

Algunos dispositivos de E/S requieren una velocidad de transferencia de datos mayor que la que permite cualquier método de E/S controlado por el procesador. En estos casos la información se puede transferir directamente, entre el dispositivo y la memoria del procesador, sin la intervención de éste. A esta técnica se la denomina acceso directo a memoria o **DMA** *(direct memory access).*

Una lógica externa de alta velocidad (controlador de DMA) se encarga de manejar la transferencia de datos entre el dispositivo y la memoria. En una transferencia de datos por DMA el procesador no debe acceder a la memoria para permitir que lo haga el controlador. Este pide un DMA al procesador mediante una línea (HOLD para el 8088 en modo mínimo). El procesador completa el ciclo de máquina en curso y emite una señal de reconocimiento del pedido (/HLDA en el 8088) deteniendo su actividad, y dejando sus buses en estado de alta impedancia, hasta que el controlador retire el pedido. Durante este tiempo se efectuará la transferencia de datos(lectura o escritura de memoria).

En los sistemas donde se utilizan memorias dinámicas se debe tener en cuenta el máximo tiempo que se pueda detener al procesador para mantener el adecuado funcionamiento de la lógica de refresco.

5.1.1 Ejemplo De E/S Por Acceso Directo A Memoria.

Una microcomputadora está conectada a un gran sistema a través de un canal de comunicación de alta velocidad. Entre la microcomputadora y el canal de comunicación se transfieren bloques de datos mediante acceso directo a memoria.

El programa principal se comunica con el controlador de DMA, mediante E/S controladas por programa, indicando el bloque de datos a transferir e iniciando la operación de DMA. Entonces el controlador detiene al procesador y efectúa la transferencia entre la memoria y el canal.

El acceso directo a memoria es una técnica de diseño del hardware que permite a los periféricos conectados a un sistema realizar transferencias sobre la memoria sin la intervención del procesador. De esta manera, las lentas operaciones de entrada y salida de bloques de datos, se pueden realizar *en la sombra* mientras la CPU se dedica a otras tareas más útiles. Como la memoria del ordenador sólo puede ser accedida a un tiempo por una fuente, en el momento en que el DMA realiza las transferencias el microprocesador se desconecta de los buses, cediéndole el control. El funcionamiento del controlador de DMA se basa en unos registros que indican la dirección de memoria a ser accedida y cuántas posiciones de memoria quedan aún por transferir. La transferencia de datos entre los periféricos y la memoria por DMA no suele efectuarse de golpe, sino más bien poco a poco, robándole algunos ciclos a la CPU. Los controladores de DMA suelen disponer de varias líneas de petición de DMA, pudiendo atender las necesidades de varios periféricos que soliciten una transferencia, quienes deben haber sido diseñados expresamente para soportar el DMA.

5.2 Controlador de DMA 8237

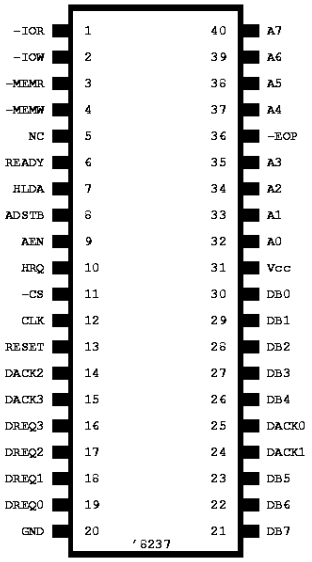

Fig. 29 . Esquema circuital del 8237

El 8237 es un controlador de DMA de 4 canales programables en 3 modos diferentes, con posibilidad de ser conectado en cascada con otros de su misma especie. Además de las funciones tradicionales, el 8237 soporta también transferencias memoria-memoria, incluyendo la posibilidad de rellenar un área de la memoria con cierto dato. La arquitectura es de 16 bits, tanto para direcciones como datos, por lo que está especialmente diseñado para sistemas basados en el Z80 y 8085; aunque puede operar también con procesadores más avanzados, como la serie 80x86, pero sin alcanzar a aprovechar todas sus posibilidades.

5.3 Descripción funcional

Los modos de operación del 8237 están diseñados para soportar transferencias de una sola palabra de datos y flujos de datos discontinuos entre la memoria y los periféricos. El controlador de DMA es realmente un circuito secuencial generador de señales de control y direcciones que permite la transferencia directa de los datos sin necesidad de registros temporales intermedios, lo que incrementa drásticamente la tasa de transferencia de datos y libera la CPU para otras tareas. Las operaciones memoria-memoria precisan de un registro temporal intermedio, por lo que son al menos dos veces más lentas que las de E/S, aunque en algunos casos aún más veloces que la propia CPU (no es el caso de los ordenadores compatibles).

El 8237 consta internamente de varios bloques: un bloque de control de tiempos que genera las señales de tiempo internas y las señales de control externas; un bloque de gestión de prioridades, que resuelve los conflictos de prioridad cuando varios canales de DMA son accedidos a la vez; también posee un elevado número de registros para gestionar el funcionamiento. Los registros internos del 8237 están resumidos en la figura de la derecha

5.4 Operación del DMA

En un sistema, los buses del 8237 están conectados en paralelo al bus general del ordenador, siendo necesario un latch externo para almacenar la parte alta de la dirección de memoria. Cuando está inactivo, el 8237 está desconectado de los buses; cuando se produce una petición de DMA pasa a controlar los buses y a generar las señales necesarias para realizar las transferencias. La operación que realiza el 8237 es consecuencia de la programación realizada previamente en los registros de comando, modo, base de dirección y contador de palabras a transferir.

Para comprender mejor el funcionamiento del 8237 es conveniente considerar los estados generados por cada ciclo. El DMA opera básicamente en dos ciclos: el **activo** y el **inactivo** (o ***idle***). Tras ser programado, el DMA permanece normalmente inactivo hasta que se produce la solicitud de DMA en algún canal o vía software. Cuando ésta llega, si ese canal no estaba enmascarado (es decir, inhibido) el 8237 solicita los buses a la CPU y se pasa al ciclo activo. El ciclo activo se compone de varios estados internos, en función de la manera en que sea programado el chip.

El 8237 puede asumir 7 diferentes estados, cada uno de ellos compuesto de un ciclo de reloj completo. El estado 1 (**S1**) es el estado inactivo o *idle*. En él se entra cuando no hay pendiente una petición de DMA válida, al final de la secuencia de transferencia, o tras un reset o un *Master Clear* (que se verá más adelante). En S1 el DMA está inactivo pero puede ser programado por el microprocesador del sistema. El estado 0 (**S0**) es el primer estado de servicio DMA. El 8237 ha solicitado los buses a la CPU a través de la línea HRQ pero la CPU aún no ha respondido a través de HLDA. En esta situación, el 8237 puede aún todavía ser programado. Una vez que la CPU responde, la labor del 8237 puede comenzar: los estados **S2**, **S3** y **S4** se suceden entonces para realizar el servicio. Si se necesitara más tiempo, está prevista la posibilidad de insertar estados de espera entre S2 ó S3 y S4 a través de la patilla READY.

Téngase en cuenta que los datos son pasados directamente de la memoria hacia/desde los periféricos, por lo tanto no cruzan a través del DMA (las líneas -IOR y -MEMW, o -IOW y -MEMR, son activadas al mismo tiempo). El caso de las operaciones memoria-memoria es especial, ya que para cada palabra a mover hay que realizar la operación de lectura (en unos estados denominados S11, S12, S13 y S14) y después la de escritura (estados S21, S22, S23, S24).

5.5 Conexión en cascada

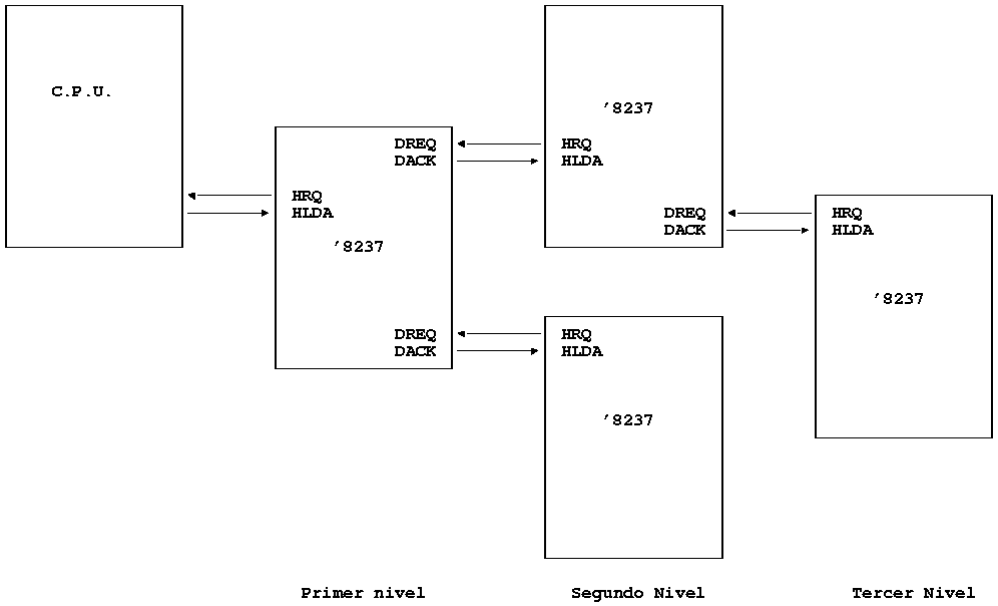

Fig. 30. Cascada de 8237

Esta conexión es empleada para conectar más de un 8237 en el sistema. La línea HRQ de los 8237 hijo es conectada a la DREQ del 8237 padre; la HLDA lo es a la DACK. Esto permite que las peticiones en los diversos 8237 se propaguen de uno a otro a través de la escala de prioridades del 8237 del que cuelgan. La estructura de prioridades es por tanto preservada. Teniendo en cuenta que el canal del 8237 padre es empleado sólo para priorizar el 8237 adicional que cuelga (hijo), no puede emitir direcciones ni señales de control por sí mismo: esto podría causar conflictos con las salidas del canal

activo en el 8237 hijo. Por tanto, el 8237 padre se limita en el canal del que cuelga el 8237 hijo a controlar DREQ, DACK y HRQ, dejando inhibidas las demás señales. El -EOP externo será ignorado por el 8237 padre, pero sí tendrá efecto en el 8237 hijo correspondiente.

Cuando de un 8237 cuelga otro, estamos ante un sistema DMA de dos niveles. Si del DMA hijo cuelga a su vez otro, sería un sistema DMA de tres niveles, como el visto en la Fig. 30

Al programar los 8237 en cascada, se debe empezar por el primer nivel. Tras un Reset, las salidas DACK son programadas por defecto para ser activas a nivel bajo y son colocadas en alto. Si están conectadas directamente a HLDA, el segundo nivel de 8237 no puede ser programado hasta que la polaridad de DACK no se cambie para que sea activa a nivel alto. Los bits de máscara de canales del 8237 padre funcionan como cabría esperar, permitiendo inhibir 8237's de niveles inferiores.

5.6 Modos de Transferencia

Cada uno de los 3 modos de transferencia puede realizar 3 tipos distintos de transferencias: lectura, escritura y verificación. La lectura pasa datos de la memoria al dispositivo E/S (activando -IOW y -MEMR); la escritura mueve datos desde los dispositivos E/S a la memoria (activando -IOR y -MEMW). Las transferencias de tipo verificación son pseudotransferencias: el funcionamiento es similar a la lectura o escritura pero sin tocar las líneas de control de la memoria ni de los periféricos; durante el modo de verificación se ignora la línea READY; este modo no es permitido en las operaciones memoria-memoria.

Autoinicialización.

Cualquier canal puede ser programado para incluir esta característica. En el momento de programar el chip, los registros base de dirección y base contador de palabras son cargados a la vez y con el mismo valor que los registros de dirección en curso y contador de palabras en curso. Los registros base permanecen inalterados en todo momento, por lo que al final del servicio sirven, en este modo de trabajo, para recargar de nuevo los registros en curso. Esto sucede justo tras la señal -EOP, quedando el 8237 listo para repetir de nuevo la misma transferencia (cuando se solicite a través de la línea DREQ o por software). En esta modalidad, los bits de máscara están a 0.

Memoria-Memoria.

En este tipo de transferencia se emplean siempre los canales 0 y 1. La transferencia comienza activando la línea DREQ del canal 0, bien por hardware o por software. El 8237 solicita entonces un servicio de DMA ordinario, con el que lee el byte de la memoria a través de 4 estados y empleando el Block Transfer Mode visto con anterioridad. El registro de dirección en curso del canal 0, que indica la dirección *origen* en la memoria, es incrementado/decrementado (según haya sido programado) y el dato es almacenado en el registro temporal del 8237. En otros 4 estados más, el dato es pasado del 8237 de nuevo a la memoria, usando la dirección del registro de dirección en curso del canal 1, que indica la dirección *destino* en memoria, el cual es también incrementado/decrementado según proceda. Además, se decrementa el registro contador de palabras en curso del canal 1: si al decrementar se desborda (pasa de 0 a 0FFFFh) se activa el bit TC del registro de estado (Terminal Count, fin de cuenta) y se genera un pulso -EOP, finalizando el proceso. En el caso de que el valor del registro contador de palabras del canal 0 pase de 0 a 0FFFFh, sin embargo, no se actúa sobre TC ni sobre EOP (no finaliza el proceso) aunque este canal se autoinicializa si así estaba programado.

Si se desea una autoinicialización total en este tipo de transferencias, los registros contadores de palabras del canal 0 y 1 han de ser programados con el mismo valor inicial; de lo contrario, sólo uno de
El canal 0 puede ser también programado para retener siempre la misma dirección durante todas los dos canales se autoinicializará (el que primero desborde su registro contador de palabras). las transferencias, lo que permite copiar un mismo byte en todo un bloque de la memoria.

El 8237 puede responder a señales -EOP externas durante este tipo de transferencias, pero sólo cede el control de los buses después de completar la transferencia de la palabra que tenga entre manos. Los circuitos para comparar datos en búsquedas de bloques pueden emplear -EOP para terminar la operación tras encontrar lo que buscan. Las operaciones memoria-memoria se pueden detectar por hardware como una combinación de AEN activo sin que al mismo tiempo se produzcan salidas DACK.

5.7 Prioridad

El 8237 tiene dos maneras de codificar la prioridad, seleccionables por software. La primera es la prioridad fija, basada en el número del canal (0-máxima, 3-mínima). Una vez que un canal es atendido, los demás esperan hasta que acabe. La segunda modalidad es la prioridad rotatoria: el último canal servido pasa a tener la menor prioridad y el que le sigue la máxima. La rotación de prioridades se produce cada

vez que se devuelven los buses a la CPU. Esta última modalidad de prioridad asegura que un canal sea atendido al menos después de haber atendido los otros 3, evitando que un solo canal monopolice el uso del DMA. Con independencia del tipo de prioridad programada, ésta es evaluada cada vez que el 8237 recibe un HLDA.

5.8 Generación de direcciones

Para reducir el número de pines, el 8237 tiene multiplexada la parte alta del bus de direcciones. En el estado S1, los 8 bits más significativos de la dirección son depositados en un latch externo a través del bus de datos. La línea AEN indica a la circuitería externa que debe habilitar el latch como parte alta del bus de direcciones cuando llega el momento (la parte baja la suministra directamente el 8237). En el Block Transfer Mode y en el Demand Transfer Mode, que implican múltiples transferencias, el 8237 es suficientemente inteligente como para generar estados S1 sólo cuando hay acarreo en la parte baja del bus de direcciones (1 de cada 256 veces) evitando acceder al latch externo cuando no es necesario modificarlo y ahorrando tiempo.

6 Cuestionario y Ejercicios.

1) Indique los parámetros a tener en cuenta para elegir uno u otro método de entrada salida. Describir brevemente los distintos tipos.
2) Escriba un programa que mueva 1024 bytes de una puerta de entrada ubicada en el mapa de memoria en 2000:0000H a una puerta de salida en 8000:3456H. Debe tener en cuenta que la palabra de estado de la puerta de entrada se halla en 2000:0003H y cuando el bit 2 esté en 1, indicará la llegada de un dato. En la puerta de salida la palabra de estado se encuentra en 8000:3458H y en la misma, cuando el bit 6 esté en 1 indicará que esta en condiciones de transmitir un dato.
3) Repetir el ejercicio anterior para mover 1024 palabras (de 16 bits).
4) Repetir el último ejercicio suponiendo que se desean mover 64K palabras.
5) ¿Cómo se genera la tabla de vectores de interrupción?. ¿Qué contiene?.
6) En la Fig. 11. Generación del tipo de interrupción., suponer que las tres líneas más bajas del buffer están conectadas a tres fuentes de interrupción de manera que el tipo de interrupción generado dependa que cual (o cuáles) interrupciones se hayan generado. Se supone que el periférico conectado a la pata D2 es el de mayor prioridad y el conectado a D0 es el de menor prioridad. Escriba las 8 rutinas de atención de interrupción de manera que de la forma más simple se atiendan todas las interrupciones en el orden de las prioridades.
7) Suponga tener un 8259 Maestro en la dirección de E/S 380h y generando una interrupción tipo base en 50H. Un 8259 esclavo se encuentra en la dirección de E/S 382h y genera una interrupción base en 58h. La salida de interrupción del esclavo se conecta en la entrada IR3 del maestro. Suponga que se genera una IR3 en el esclavo. Describa detalladamente todo el proceso que culmina en la ejecución de la rutina de atención de interrupción.
8) Describa la operación del handshake de entrada. Genere in circuito y analicelo.
9) Busque información respecto de la intercepción de interrupciones. Aplique las interrupciones de software para ejecutar la rutina de intercepción de una interrupción.