

# Entrada/salida y buses

Estructura de Computadores

12<sup>a</sup> Semana

## Bibliografía:

- |               |   |
|---------------|---|
| [HAM03] Cap.4 | Organización de Computadores. Hamacher, Vranesic, Zaki. McGraw-Hill 2003<br>Signatura ESIIT/ <a href="#">C.1 HAM org</a>                            |
| [STA8] Cap.7  | Organización y Arquitectura de Computadores, 7 <sup>a</sup> Ed. Stallings. Pearson Educación, 2008.<br>Signatura ESIIT/ <a href="#">C.1 STA org</a> |

# Guía de trabajo autónomo (4h/s)

## ■ Lectura

- Cap. 4 Hamacher
- Cap. 7 Stallings

## Bibliografía:

[HAM03] Cap.4

Organización de Computadores. Hamacher, Vranesic, Zaki. McGraw-Hill 2003

Signatura ESIIT/[C.1 HAM org](#)

[STA8] Cap.7

Organización y Arquitectura de Computadores, 7<sup>a</sup> Ed. Stallings. Pearson Educación, 2008.

Signatura ESIIT/[C.1 STA org](#)

[

# Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

# Objetivo de los sistemas de E/S

- Realizar la conexión del procesador con una gran variedad de dispositivos periféricos, teniendo en cuenta que las características de los dispositivos de E/S suelen diferir notablemente de las del procesador; en especial:
  - la **velocidad** de transmisión de los periféricos
    - normalm. **menor** que la velocidad a la que opera el procesador
    - muy **variable** (pocos bytes/s hasta >100 MB/s)
  - la **longitud de palabra**
  - los **códigos** para representar los datos
- Para compatibilizar las características de los dispositivos de E/S con el sistema procesador/memoria se usan los circuitos de interfaz o controladores de periféricos.

# Interfaces de E/S

- **Circuitos de interfaz o controladores de periféricos:**
  - Circuitos de **adaptación de formato de señales y características de temporización** entre el procesador y los dispositivos de E/S.
  - Proporcionan todas las transferencias de datos necesarias entre el procesador y los periféricos, utilizando un bus de E/S.
  - **Requieren uso de software:**
    - Programas de E/S ejecutados por el procesador que controlan la transferencia de información hacia y desde los dispositivos de E/S.
  - En computadores de altas prestaciones se han utilizado procesadores especializados para las funciones de E/S: **procesadores de E/S (IOP)** o canales.
    - Procesadores cuyos conjuntos de instrucciones se restringen a aquellas que se precisan en las operaciones de E/S.
    - Se hacen cargo de todas las transferencias con los periféricos.

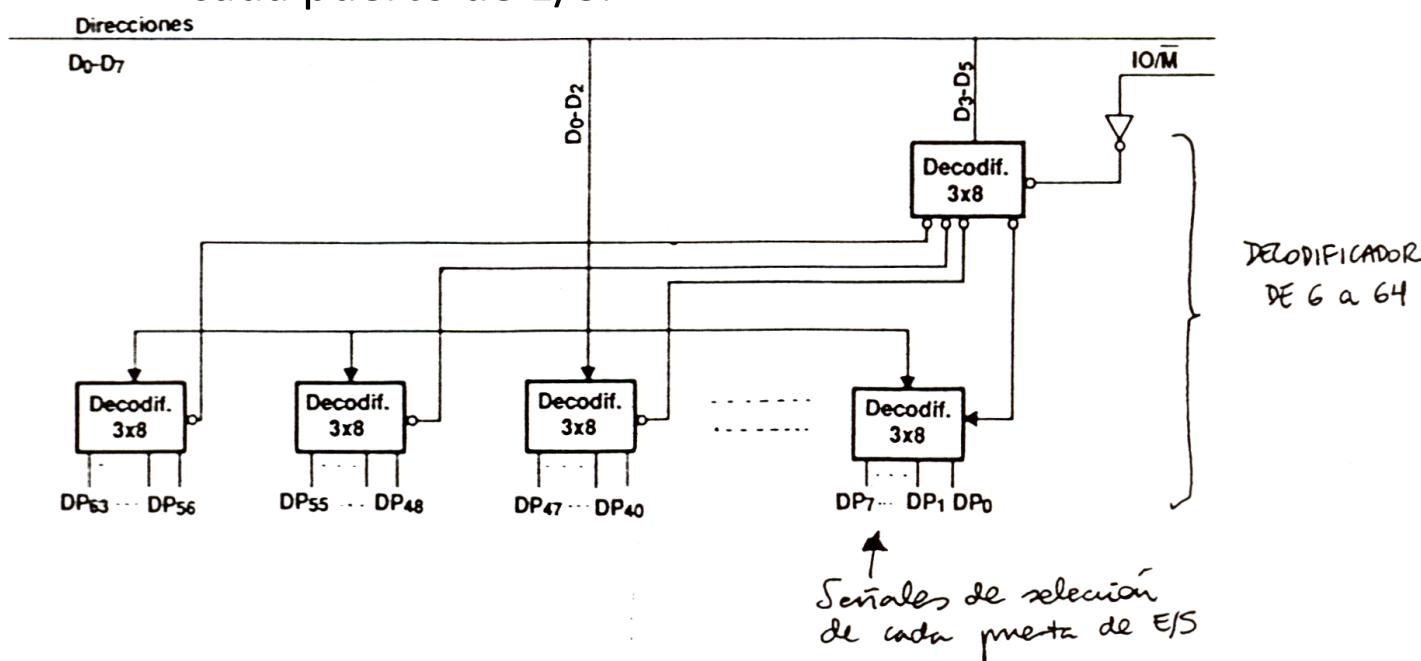
# Funciones que debe incluir el sistema de E/S (I)

## ■ *Direccionamiento o selección del periférico:*

- El procesador sitúa en el bus de direcciones la dirección asociada con el dispositivo.
- Si se conectan varios periféricos debe preverse la forma de que no haya conflictos de acceso al bus.
- Con  $p$  bits  $\Rightarrow$  pueden direccionarse  $2^p$  direcciones distintas (mapa de E/S).
  - Cada dirección especifica uno o dos puertos de E/S:
    - El hardware de cada dirección suele ser único (bien entrada o bien salida).
    - Pero a veces los circuitos de E y de S de una única dirección son independientes (misma dirección  $\Rightarrow$  dos puertos, uno de entrada y otro de salida).
  - Cada interfaz de periférico emplea varios puertos para comunicarse con el procesador.

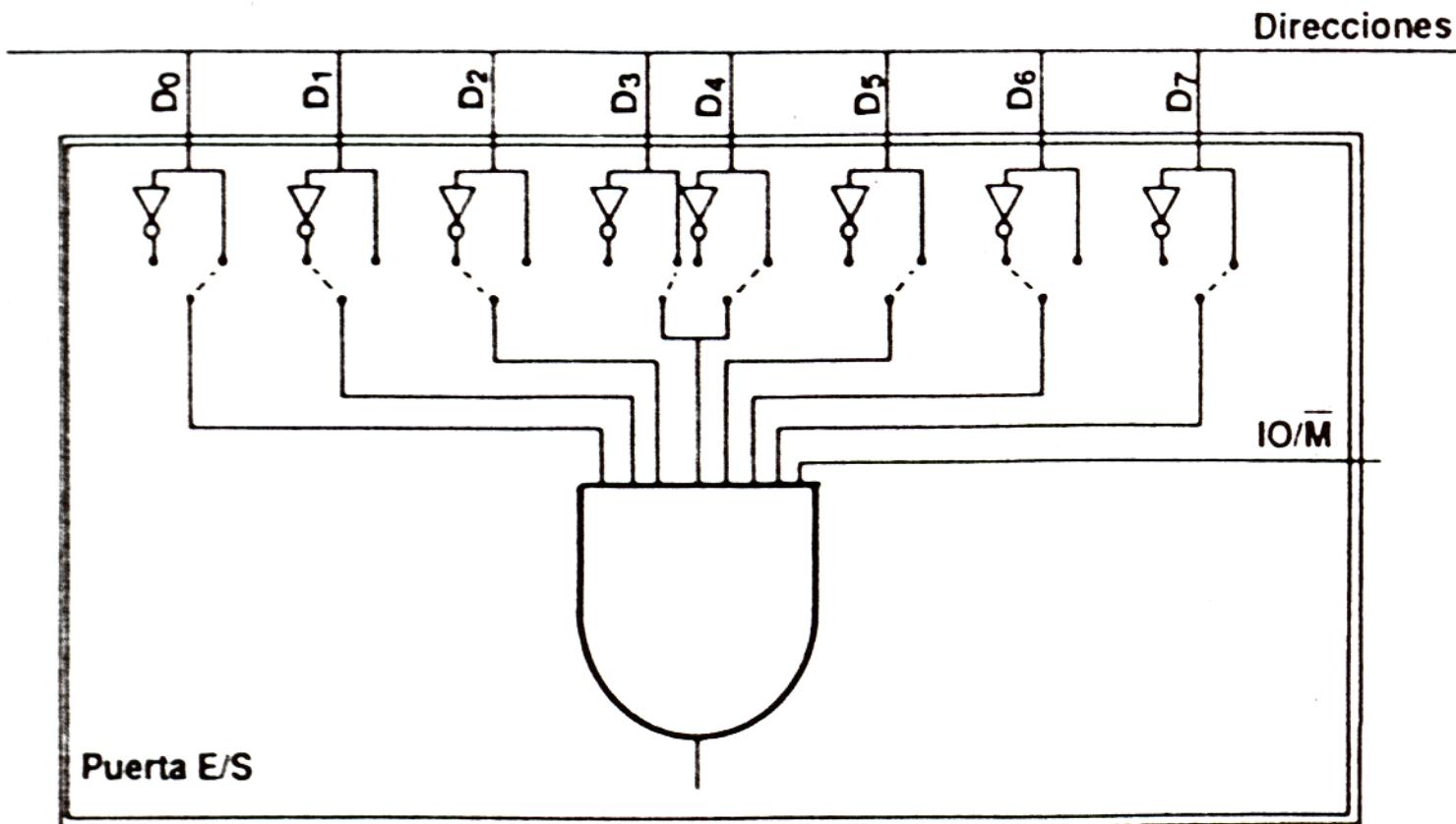
# Funciones que debe incluir el sistema de E/S (II)

- Técnicas de direccionamiento
  - ✗ **Direccionamiento por selección lineal:** asignar un bit del bus de direcciones a cada puerto.
  - ✓ **Direccionamiento por decodificación:** decodificar los bits de dirección para seleccionar un puerto de una interfaz.
    - **Decodificación centralizada:** un decodificador selecciona cada puerto de E/S.



# Funciones que debe incluir el sistema de E/S (III)

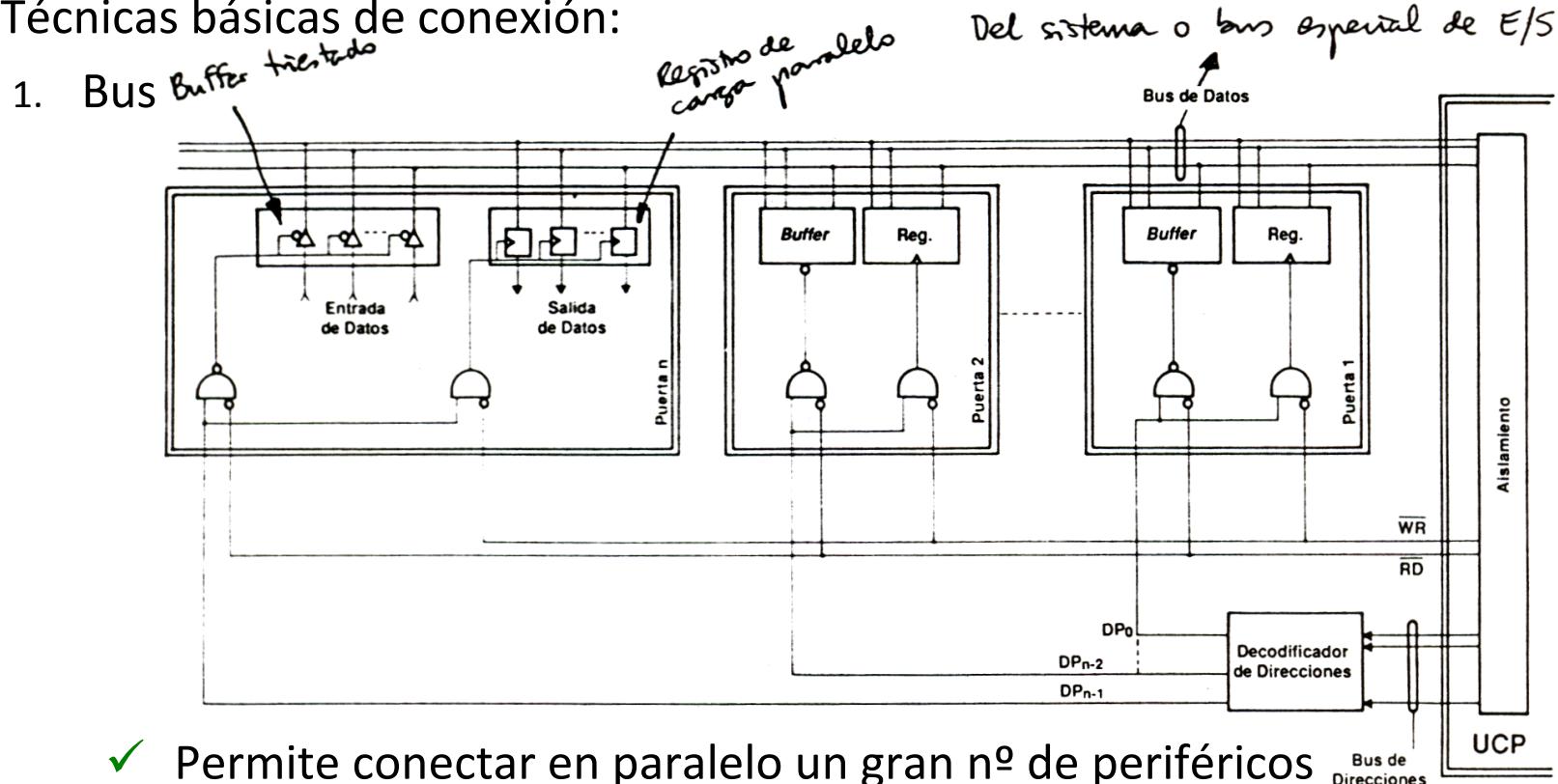
- Decodificación en cada puerto de E/S: cada puerto reconoce su propia dirección.



# Funciones que debe incluir el sistema de E/S (IV)

## ■ Comunicación física entre el periférico y el procesador.

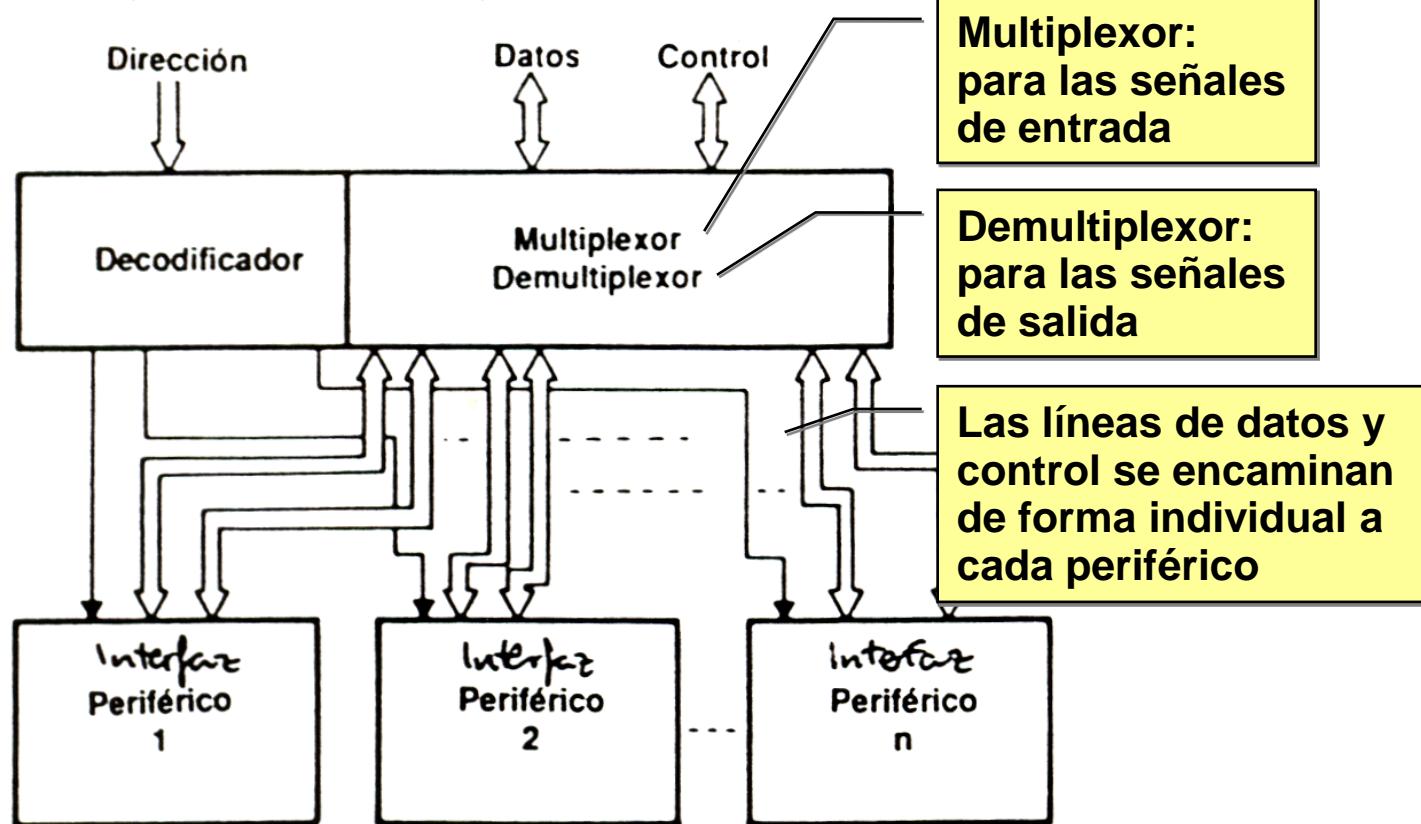
- Técnicas básicas de conexión:



- ✓ Permite conectar en paralelo un gran nº de periféricos
- ✓ Es fácil expandir el sistema (añadiendo más tarjetas o circuitos de interfaz)

# Funciones que debe incluir el sistema de E/S (V)

## 2. Multiplexor / demultiplexor



- ✗ Expansión difícil
- ✗ Mucha circuitería

# Funciones que debe incluir el sistema de E/S (VI)

## ■ *Sincronización:*

- Acomodación de las velocidades de funcionamiento del procesador/MP y los dispositivos de E/S.
- Hay que establecer un mecanismo para saber cuándo se puede enviar o recibir un dato.
- Deben incluirse:
  - Palabras de memoria temporal en la interfaz que sirvan como **búfer**. La entrada o salida se hace sobre este búfer intermedio. La operación de E/S real se realiza sólo cuando el dispositivo está preparado.
  - **Señales de control de conformidad** para iniciar o terminar la transferencia (listo, petición, reconocimiento).
  - La temporización de las transferencias puede ser:
    - Síncrona —————
    - Asíncrona

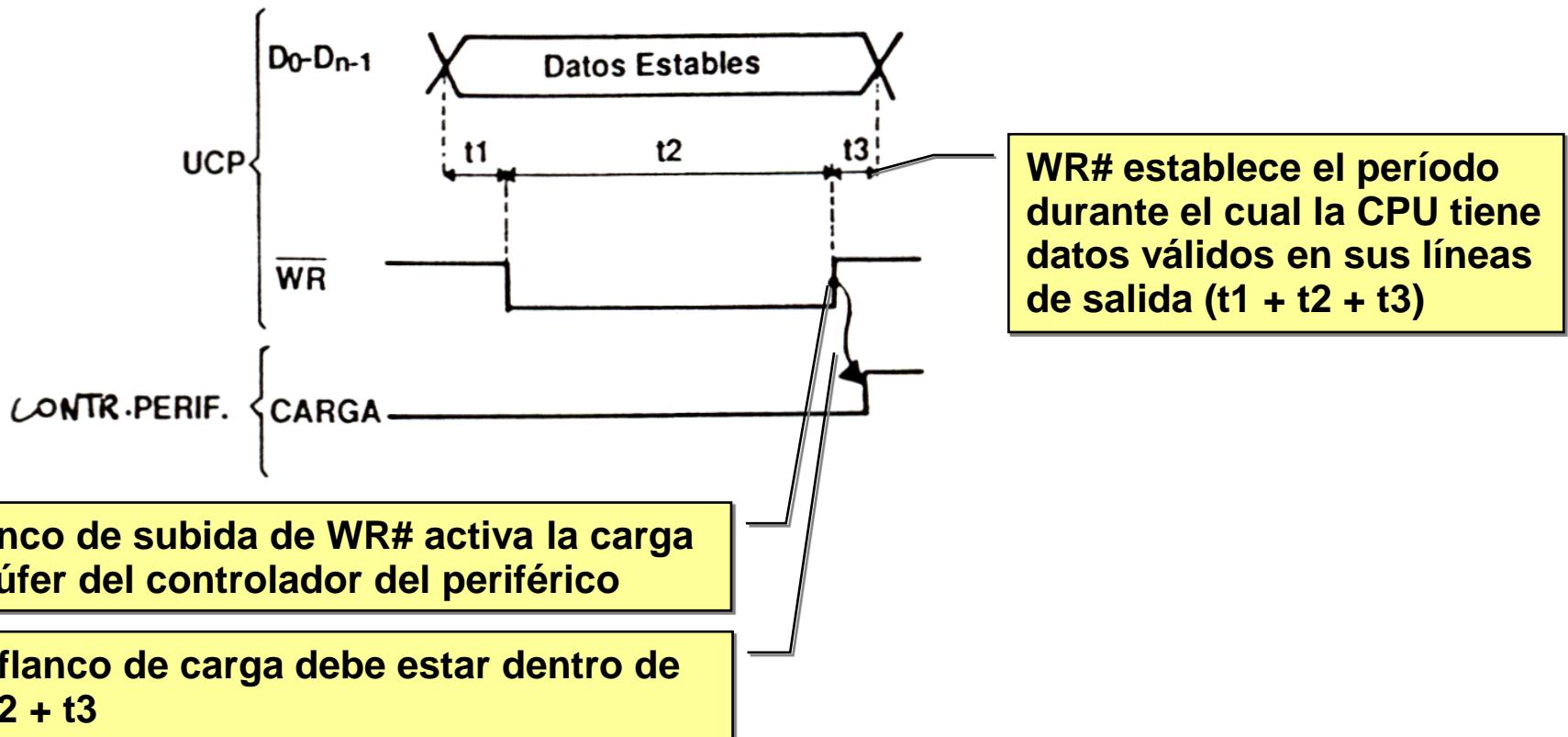
**¡Ojo! concepto confuso,  
depende del autor**

# Funciones que debe incluir el sistema de E/S (VII)

## ■ Temporización síncrona

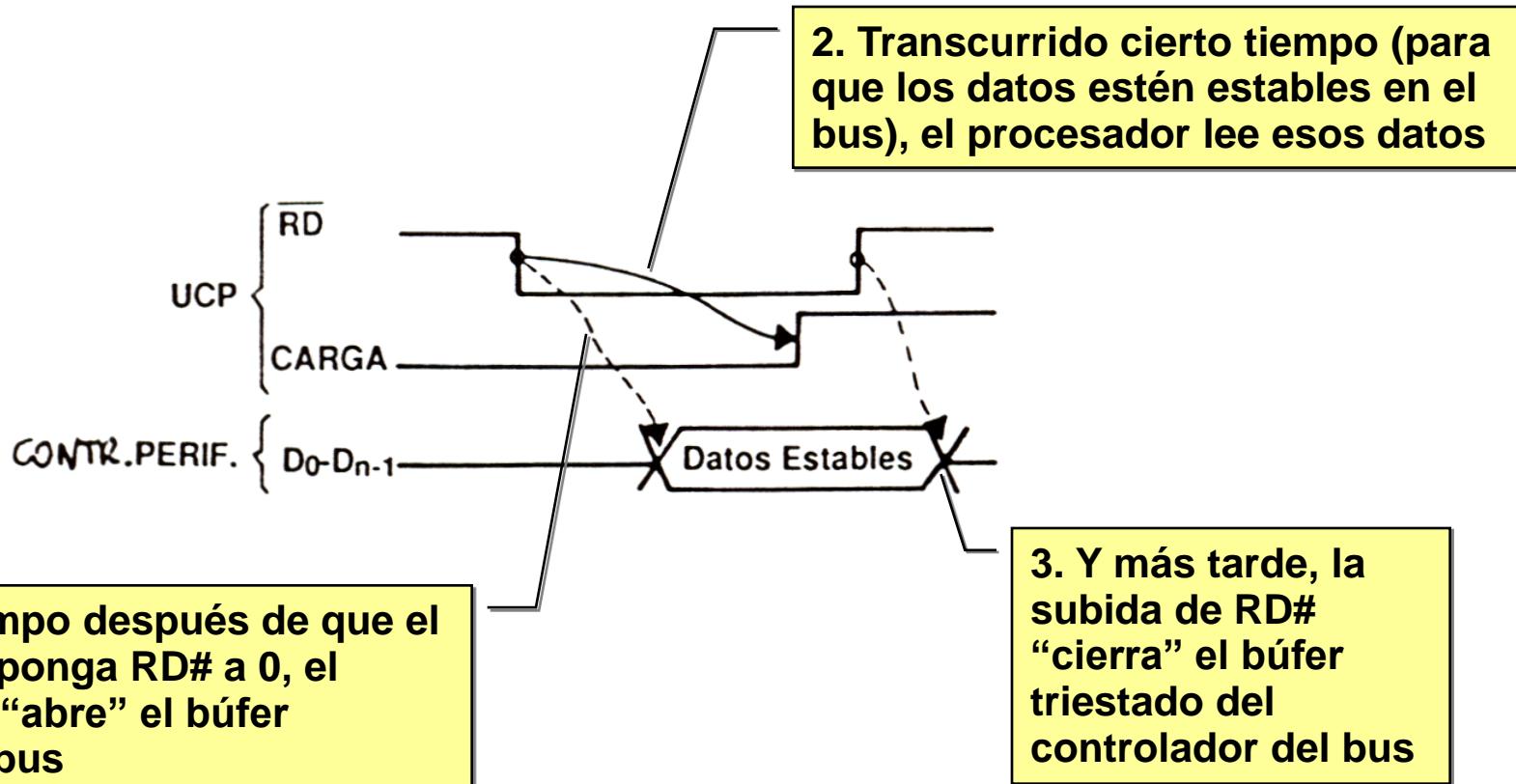
### ▪ Escritura

- El procesador sitúa los datos en el bus de datos y al mismo tiempo genera una señal WR# = 0.



# Funciones que debe incluir el sistema de E/S (VIII)

- Lectura
  - El procesador suministra una señal de lectura RD# = 0. A partir de entonces el controlador del periférico dispone de cierto tiempo para suministrar el dato al bus de datos.



# Funciones que debe incluir el sistema de E/S (IX)

- En la temporización síncrona...
  - El **procesador establece la temporización**, que debe ser seguida por el controlador del periférico.
  - ✖ Si éste no es capaz de leer el bus, o de poner el dato, en el tiempo establecido ⇒ la transferencia fracasa.
    - » Podría haberse direccionado un dispositivo lento, inexistente o apagado.
    - » En lectura el procesador almacenará un dato incorrecto.
    - » En escritura el procesador no sabrá si ha escrito con éxito.

# Funciones que debe incluir el sistema de E/S (X)

- Temporización asíncrona o con “handshaking”
  - Se necesita una nueva señal de aceptación (ACK) con la que el controlador del periférico contesta a la petición de transferencia generada por el procesador.
  - Escritura
- 
- Hasta que no se produce el flanco de subida de ACK, no se cierra la transferencia, subiendo el procesador las señales WR# o RD#
- Lectura
- 
- Hasta que no se produce el flanco de subida de ACK, no se cierra la transferencia, subiendo el procesador las señales WR# o RD#

# Funciones que debe incluir el sistema de E/S (XI)

- En la temporización asíncrona o con “handshaking”...
  - Se establece un **diálogo (*handshaking*)** para adaptar el cronograma a las necesidades de tiempo del periférico.
    - ***Handshaking***: establecimiento de una comunicación sincronizando dos dispositivos mediante acuse de recibo o intercambio de señales de control.
  - Ventajas:
    - ✓ Se pueden **conectar dispositivos con distintos requisitos de tiempo**.
    - ✓ Se tiene una **mayor garantía** de que el **dato sea válido**, puesto que se exige una contestación positiva del periférico.
  - Para evitar que el sistema quede bloqueado si no existe el periférico o éste no contesta, es necesario establecer un **período de espera máximo**, después del cual se considera la transferencia como errónea.

# Funciones que debe incluir el sistema de E/S (XII)

## ■ *Conversión de datos:*

- Acomodación de las características físicas y lógicas de las señales de datos empleadas por el dispositivo de E/S y por el bus del sistema.
  - Conversión de códigos (BCD, ASCII, EBCDIC, UNICODE, ANSI, etc.)
  - Conversión serie / paralelo.
  - Conversión de niveles lógicos para representar 1 y 0.
  - Conversión A/D y D/A.

## ■ *Control de los periféricos:*

- Interrogación y modificación de su estado:
  - Encendido, apagado, disponible...
- Envío de otras señales de control al periférico.

# Funciones que debe incluir el sistema de E/S (XII)

- *Mecanismo que determine la cantidad de información a transmitir en una operación de E/S y cuente el número de palabras / bytes ya transmitidos.*
- *Detección de errores*
  - En el funcionamiento del periférico...
    - ...o en los datos
      - Mediante códigos de paridad, polinomiales, etc.
    - Se repetirá la transferencia en caso necesario.

# Conceptos a diferenciar (I)

## ■ Transferencia elemental de información

- Envío o recepción de una única unidad de información (byte o palabra), ya sea un dato o una palabra de estado o control.



## ■ Operación completa de E/S

- Transferencia de un conjunto de datos
  - Sector de un disco
  - Línea de pantalla

# Conceptos a diferenciar (II)

## ■ Dispositivos de E/S físicos

- Cuando el ordenador carece de SO o *driver* adecuados.
- El programador debe tratar directamente con ellos, asumiendo sus detalles de funcionamiento y características físicas.



## ■ Dispositivos de E/S lógicos

- El programador efectúa las transferencias de datos activando las rutinas de E/S que proporciona el SO.
- Por ejemplo, el programador puede escribir un programa que escriba en una impresora lógica (ésta en realidad puede ser un bloque de espacio en disco). El SO asigna una de las impresoras físicas a la impresora lógica y controla el proceso de impresión (*spooling*).

# Conceptos a diferenciar (III)

## ■ E/S aislada o independiente

- El procesador distingue internamente entre espacio de memoria y espacio de E/S.



## ■ E/S mapeada en memoria

- El procesador no distingue entre accesos a memoria y accesos a los dispositivos de E/S.

# E/S independiente frente a E/S en memoria

## ■ E/S aislada, independiente, o con espacio de E/S

- Emplea la **patilla IO/M#** del procesador
  - **Nivel alto** ⇒ Indica a memoria y a dispositivos de E/S que se va a efectuar una operación de **E/S**.
    - Al ejecutar instrucciones específicas de E/S: IN y OUT.
  - **Nivel bajo** ⇒ Operación de intercambio de datos con **mem.**
    - Al ejecutar instrucciones de acceso a memoria: LOAD, STORE o MOVE.
- **Instrucciones específicas:** IN y OUT (o READ y WRITE), con poca riqueza de direccionamiento.
- Ejemplo:
  - procesadores de 8 bits empleaban dirección de 8 bits para puertos de E/S ⇒ 256 puertos: IN puerto, OUT puerto
  - y disponían de bus de direcciones de 16 bits ⇒ 64 K dir. memoria

# E/S independiente frente a E/S en memoria

- Ventajas:
  - ✓ Diseño más limpio de la decodificación de las direc. de memoria.
  - ✓ Facilita la protección de E/S (por ejemplo, haciendo que las instrucciones IN, OUT,... sean privilegiadas).
  - ✓ Los programas son relativamente más rápidos por la decodificación más sencilla y el menor tamaño de las instrucciones de E/S.
- Desventajas:
  - ✗ Mayor complejidad en el diseño del procesador:
    - Hay que decodificar y ejecutar las instrucciones IN, OUT...
    - Hay que generar la señal IO/M# y se necesita una patilla más del procesador para ella.

# E/S independiente frente a E/S en memoria

## ■ E/S mapeada en memoria

- Se usan algunas **direcciones de memoria** para acceder a los puertos de E/S, tras decodificarlas adecuadamente.
- El **procesador no distingue** entre accesos a memoria y accesos a los dispositivos de E/S.
  - **No** se usa la patilla IO/M#
- **No se dispone de instrucciones especiales**, sino que se usan LOAD, STORE o MOVE.
- Para evitar particionar el mapa dedicado a memoria, se agrupa la E/S en una zona bien definida al principio o final del mapa de memoria.

# E/S independiente frente a E/S en memoria

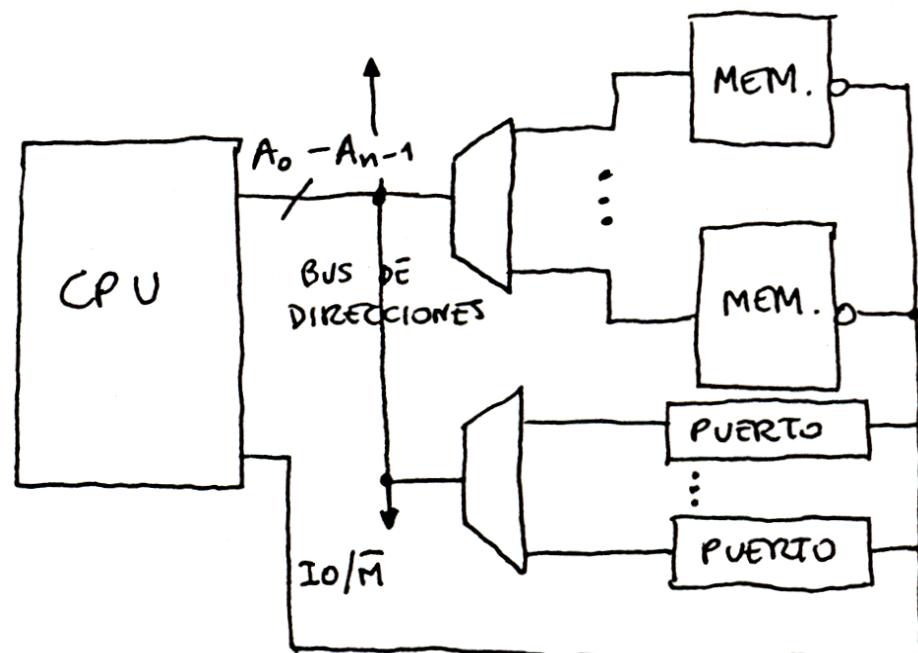
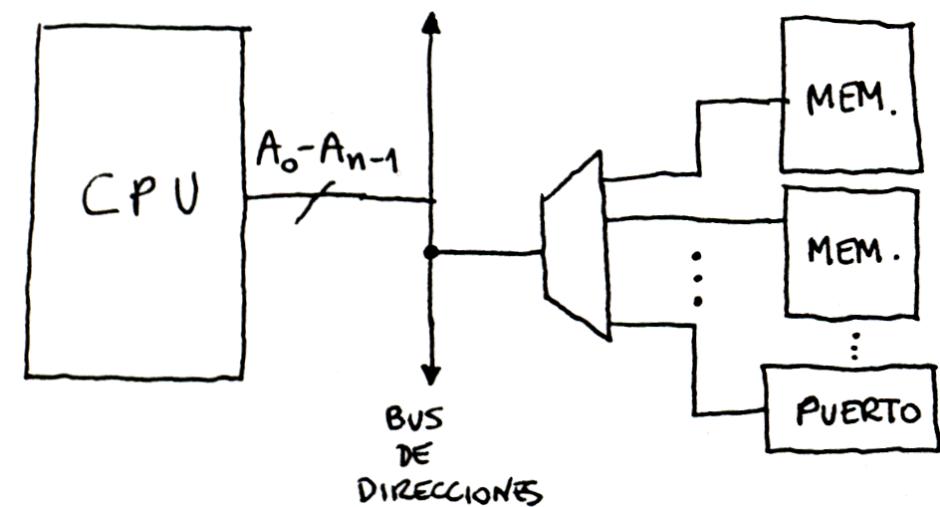
- Ventaja:
  - ✓ Menor complejidad en el diseño del procesador.
- Desventajas:
  - ✗ Cada puerto de una interfaz “ocupa” una dirección que no puede utilizarse para memoria.
  - ✗ Las instrucciones de acceso a memoria suelen ser más largas que las específicas de E/S.
    - Por ej., en los microprocesadores de 8 bits: 3 bytes frente a 2.
    - Puede disminuir la velocidad de procesamiento.
    - Aumentan los requisitos de memoria.

# E/S independiente frente a E/S en memoria

Esquemas:

E/S mapeada en memoria

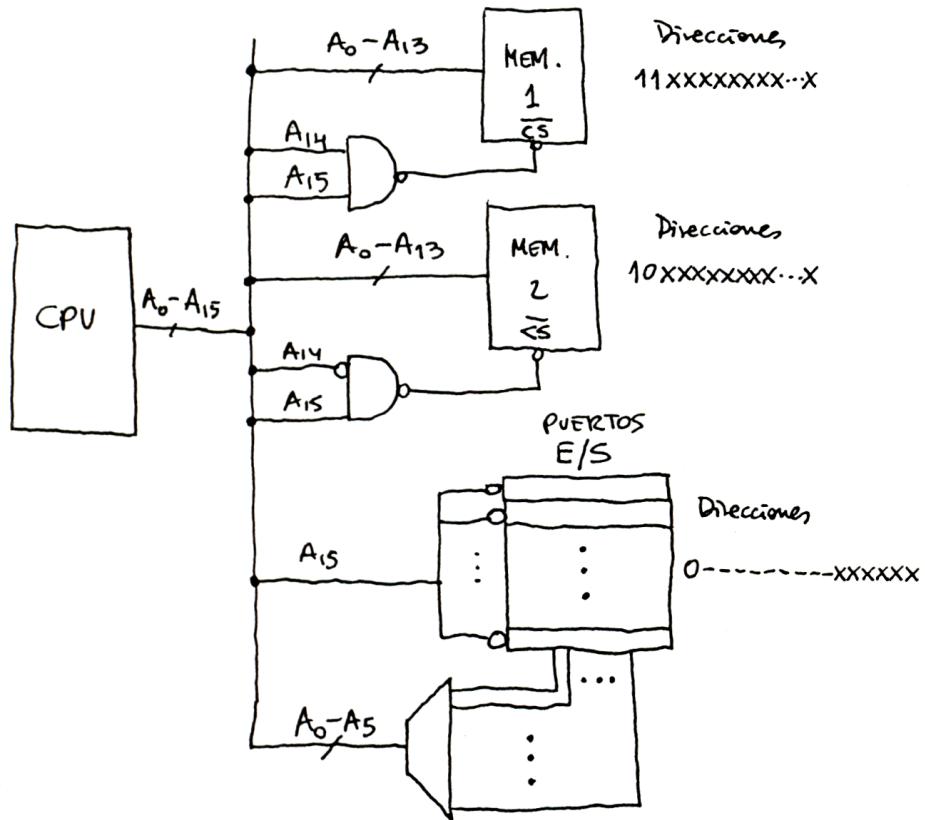
E/S independiente



# E/S independiente frente a E/S en memoria

## Ejemplo: Supongamos:

- 2 módulos de mem. de 16 KB (14 bits para direccionamiento de bytes dentro de un módulo).
- 64 puertos de E/S.
- bus de direcciones de 16 bits.

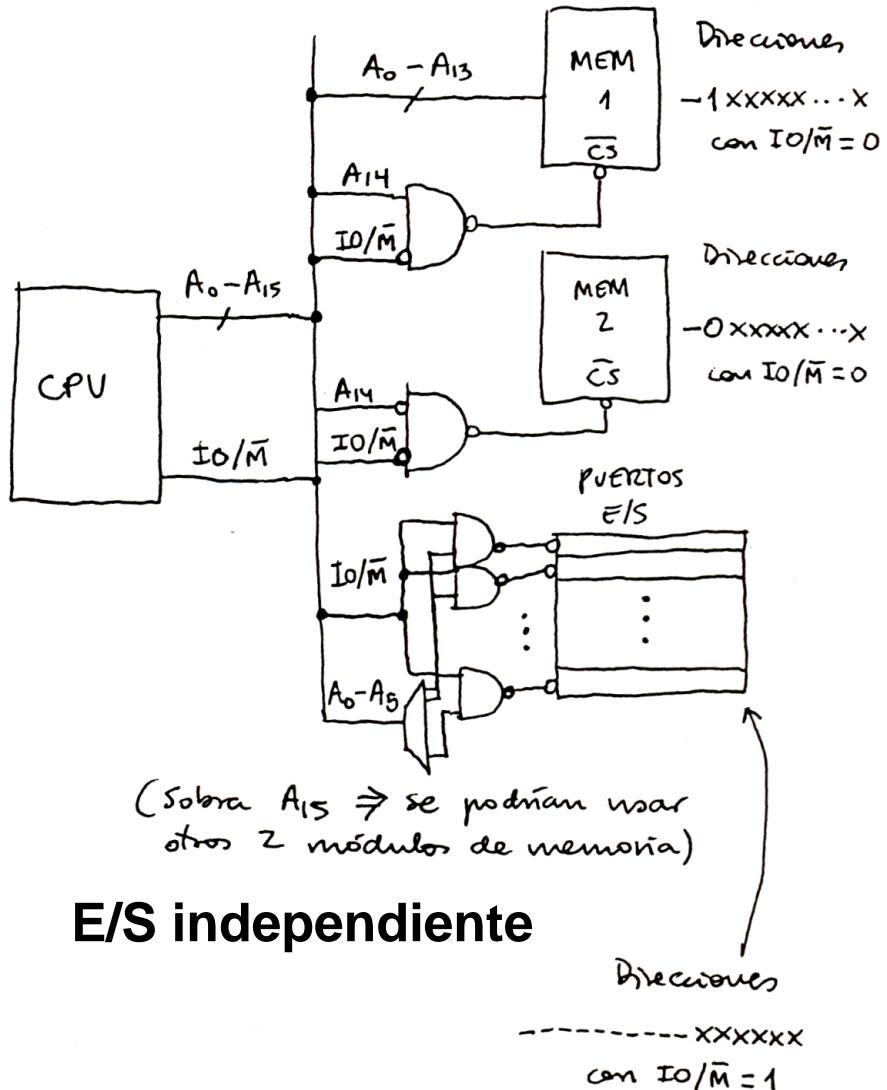


**E/S mapeada en memoria**

# E/S independiente frente a E/S en memoria

## Ejemplo: Supongamos:

- 2 módulos de mem. de 16 KB (14 bits para direccionamiento de bytes dentro de un módulo).
- 64 puertos de E/S.
- bus de direcciones de 16 bits.

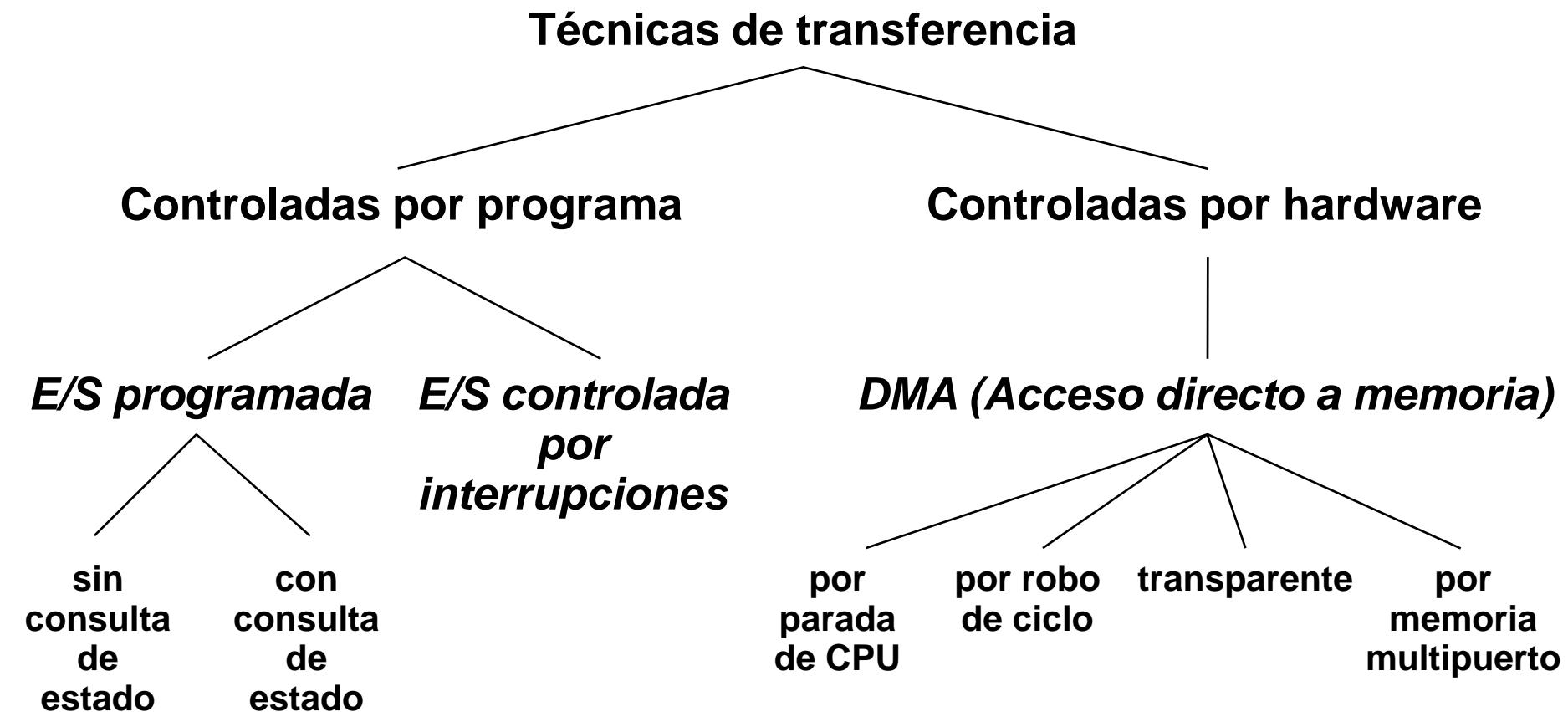


# Técnicas de E/S

## ■ Procedimientos para realizar el intercambio de información entre procesador, memoria y E/S.

- Para realizar la comunicación con los periféricos, consideramos una versión simplificada de los controladores de periféricos:
  - Uno o varios registros (**puertos** o puertas –*ports*– de E/S):
    - datos
    - control
    - estado
  - Lógica de control
    - interpreta las señales de control/estado emitidas por el procesador
    - genera las señales de control/estado que el procesador exige

# Técnicas de E/S



# Técnicas de E/S

- *E/S programada*

- El procesador participa activamente ejecutando instrucciones en todas las fases de una operación de E/S: inicialización, transferencia de datos y terminación.

- *E/S controlada por interrupciones*

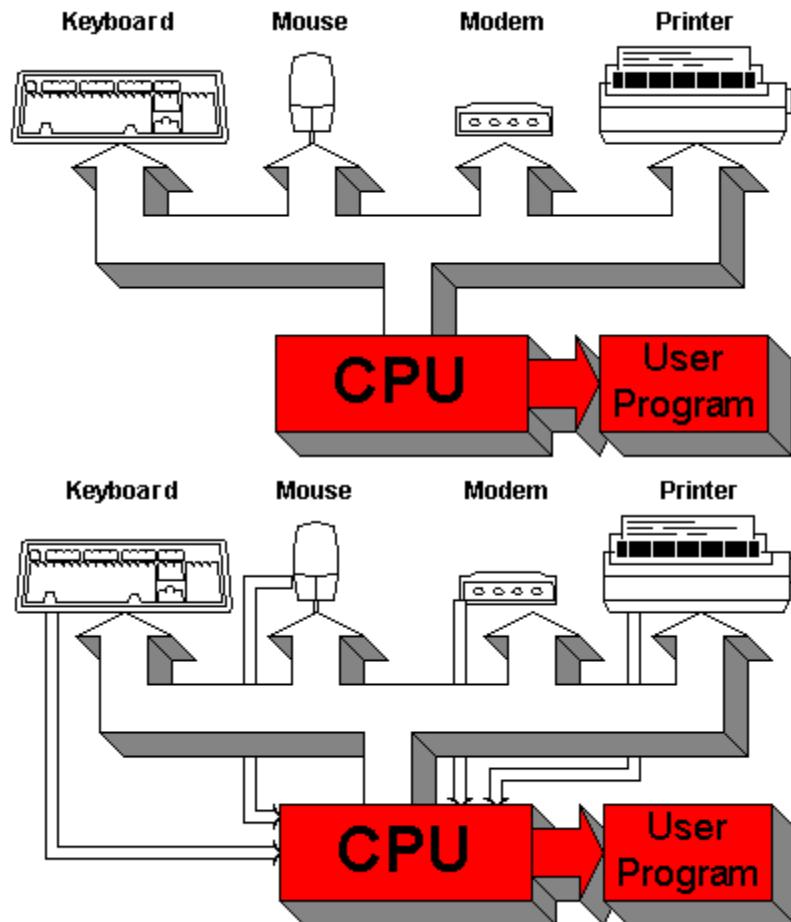
- Los dispositivos de E/S se conectan al procesador a través de líneas de petición de interrupción, que se activan cuando los dispositivos requieren los servicios del procesador.

- En respuesta, el procesador suspende la ejecución del programa en curso y ejecuta un programa de gestión de interrupción para transmitir datos con el dispositivo.

- Como en la E/S programada, los pasos de transferencia de datos están bajo el control directo de programas de control.

- *E/S mediante DMA*

- Requiere la presencia de un controlador DMA, que puede actuar como controlador del bus y supervisar las transferencia de datos entre MP y uno o más dispositivos de E/S, sin intervención directa del procesador salvo en la inicialización.



# Técnicas de E/S

## Método de control de E/S

<i>Función o parámetro</i>	E/S programada	E/S controlada por interrupciones	Acceso directo a memoria
<i>Comienzo de las operaciones de E/S</i>	La CPU lee y comprueba el estado de los dispositivos de E/S (en el caso de consulta de estado).	El dispositivo de E/S envía una petición de interrupción a la CPU. Ésta transfiere el control a una rutina de servicio P.	(Para cada bloque) El dispositivo de E/S envía una petición de interrupción a la CPU. La CPU transfiere el control a la rutina de servicio P'. P' inicializa el control de DMA.
<i>Transferencia de datos de E/S</i>		La CPU ejecuta un programa de transferencia de datos.	El controlador de DMA transfiere bloques de datos por el bus del sistema.
<i>Finalización de las operaciones de E/S</i>		Fin de ejecución del programa de transferencia de datos	(Para cada bloque) La palabra-contador DMA llega a 0. El controlador de DMA envía una petición de interrupción a la CPU.
<i>Complejidad del circuito de interfaz de E/S</i>	La menor	Baja	Moderada
<i>Velocidad de respuesta a una petición de transferencia de datos por el dispositivo de E/S</i>	Lenta	Rápida	La más rápida
<i>Máxima velocidad de transferencia de E/S</i>	Moderada	Moderada	Alta

# Segmentación de cauce

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

# Concepto de E/S programada

- Todos los pasos de una operación de E/S requieren la ejecución de instrucciones por parte del procesador
- La transferencia de un byte se realiza mediante la ejecución de una instrucción de E/S: **instrucción de transferencia de datos en la que:**
  - Un operando es un registro del controlador del periférico (**puerto**).
  - El otro operando es un **registro del procesador** (o posición mem.).
- Al decodificar la instrucción de E/S, la UC del procesador envía al exterior información de:
  - dirección (sobre qué periférico se realiza la transferencia)
  - tipo de operación (lectura / escritura)
  - temporización / sincronización

...y envía el dato o se dispone a recibirlo.

# Concepto de E/S programada

- **SALIDA:**
  - El procesador ejecuta una **instrucción de carga** de una palabra de memoria a un registro.
  - El procesador ejecuta una **instrucción de salida** que transfiere el dato desde el procesador a un puerto de salida.
- **ENTRADA:**
  - El procesador ejecuta una **instrucción de entrada**.
  - El procesador ejecuta una **instrucción de almacenamiento**.
- **Además se pueden necesitar instrucciones adicionales, por ejemplo para actualizar el número de palabras transferidas.**

# E/S programada sin y con consulta de estado

## ■ Sin consulta de estado, o incondicional

- El procesador decide en qué momento se realiza la transferencia.
- El dispositivo externo debe
  - estar siempre dispuesto a recibir datos (salida)
  - o debe tener siempre datos disponibles (entrada)
- Ejemplos:
  - Salida a un display de 7 segmentos
  - Entrada desde un joystick.

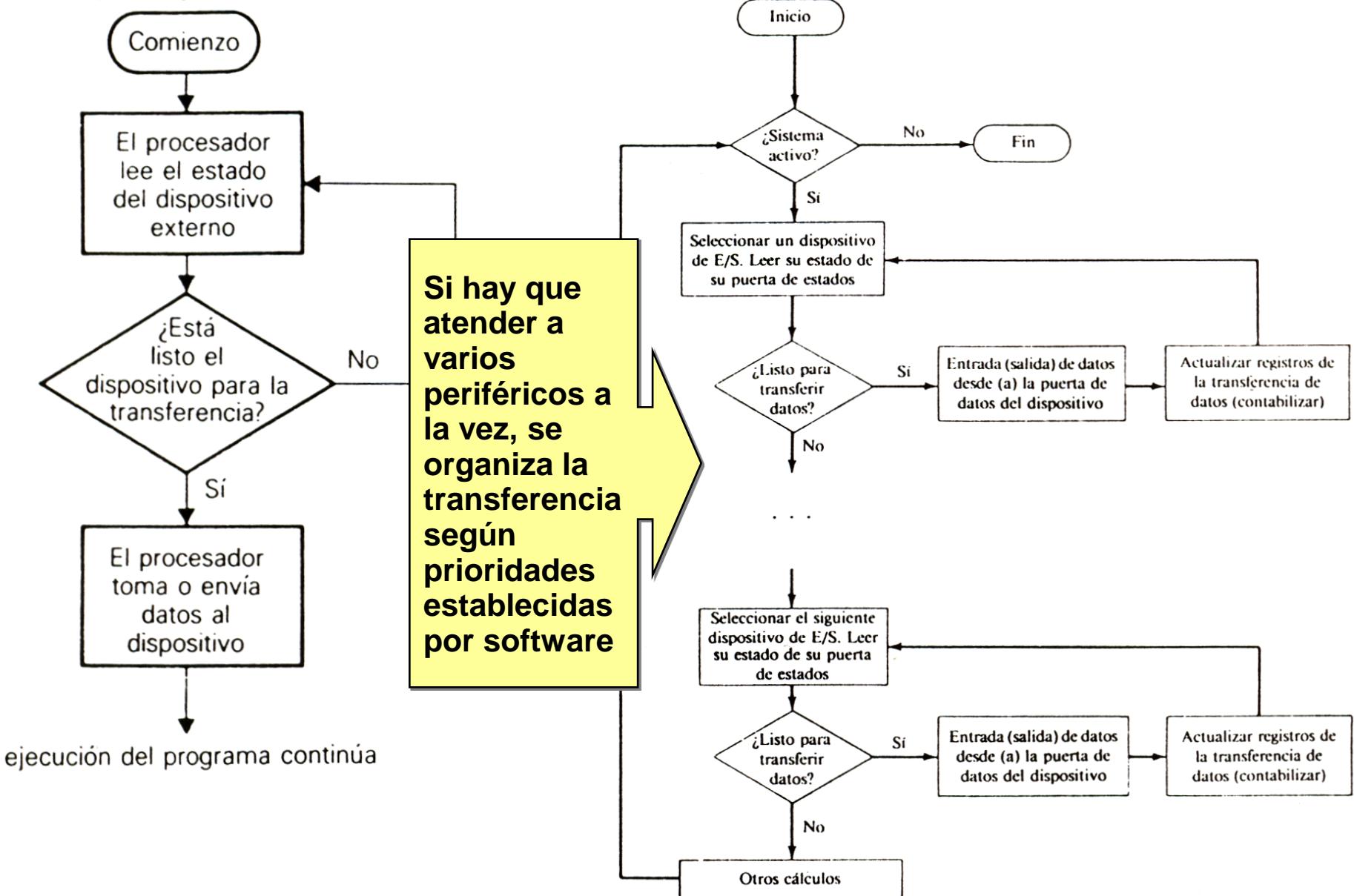
# E/S programada sin y con consulta de estado

## ■ Con consulta de estado, condicional, o con escrutinio

- El procesador pregunta al periférico (a través de la interfaz) si está preparado o no para la transferencia.
- Además de los puertos de E/S, la interfaz tiene un registro de estado con información sobre:
  - Dato listo (si se usa como entrada)
  - Periférico libre (si se usa como salida)

¡Ojo! concepto parecido al *handshaking*, pero por software

# E/S programada sin y con consulta de estado



## Ej. de programa de E/S programada con consulta de estado

- Programa (para el microprocesador Motorola 6800) que consulta un dispositivo de entrada y transfiere un bloque de datos a la MP.
- Puertos de la interfaz:
  - DATA: Datos a leer
  - STATUS: Estado del dispositivo (bit más signif. == 1 ⇒ listo)
- La E/S es mapeada en memoria.
- El número de palabras a transmitir se sitúa en el registro X del 6800, y se decrementa después de cada transferencia de un dato.
- Un dato de E/S se transfiere desde el puerto de E/S al acumulador A, y después a la pila, que sirve como área de memoria intermedia de E/S.



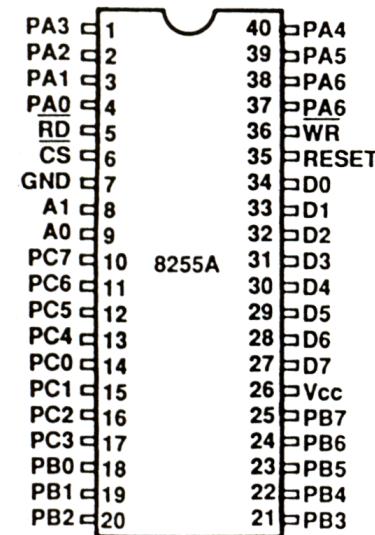
## Ej.de programa de E/S programada con consulta de estado

```
lds    IOBuf          ; Inic. puntero de pila en
                  ; ...área de mem. intermedia
ldx    Count          ; Inicializar X como contador
...
poll  lda    a Status   ; Entrada de palab. de estado
      bpl  next          ; Comprobar signo (N) de A
                  ; ...si N != 1 ==> consultar
                  ; ...otro disp. de E/S
      lda    a Data        ; Entrada de dato en A
      psh    a              ; Transferir dato a pila
      dex              ; Decrementar contador X
      bne    poll          ; Continuar consultando disp.
                  ; ...en curso si X != 0
next   ...            ; Proceder con la siguiente
                  ; ...tarea
```

# Ejemplo de circuito integrado de interfaz paralela: 8255

## ■ Interfaz de periféricos programable 8255

*(Programmable Peripheral Interface, PPI)*



## ■ Permite gestionar tres puertos de E/S de 8 bits.

## ■ Tres modos de funcionamiento:

- E/S programada sin validación:
  - *Modo 0*: E/S básica (programada).
- E/S programada o por interrupciones con validación (handshaking).
  - *Modo 1*: E/S con validación.
  - *Modo 2*: E/S con bus bidireccional, validada.

## ■ Más información en apéndice

# Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

# Concepto de interrupción

## ■ Interrupciones:

excepto en el caso de las interrupciones software

- Bifurcaciones normalmente externas al programa en ejecución, provocadas por muy diversas causas
  - externas (señales que provienen del exterior del procesador), o
  - internas (la interrupción la puede producir el propio procesador)
- ...cuyo objetivo es reclamar la atención del procesador sobre algún acontecimiento o hecho importante
- ...pidiendo que se ejecute un programa específico para tratar dicho acontecimiento,

(el código que se ejecuta en respuesta a una solicitud de interrupción se conoce como rutina de servicio de interrupción o ISR (*Interrupt Service Routine*)).

- ...de manera que el programa en ejecución queda temporalmente suspendido.

# Concepto de interrupción

## ■ Ejemplo:

- Interrupción provocada por un controlador de DMA,
  - ...cuando termina la transmisión de un bloque de datos,
  - ...para poner en conocimiento de los programas que tratan la E/S
  - ...que ha terminado y se encuentra disponible para poder realizar otra operación.

# Concepto de interrupción

## ■ Hay que diferenciar entre:

- Interrupción propiamente dicha:
  - Diálogo de señales necesario para que
    - ...el procesador acepte, de forma correcta, la solicitud de interrupción,
    - ...y salte al programa que se debe ejecutar.
- Tratamiento de la interrupción
  - Ejecución del programa de gestión de la interrupción.
- La frontera entre las dos funciones puede resultar difusa, ya que hay pasos que en algunas soluciones se hacen mediante diálogo de señales y en otras mediante programas.

# Secuencia de eventos

Esquema correspondiente a identificación de la fuente de la interrupción por hardware, sin *polling*:

Dispositivo: efectúa una solicitud de interrupción

CPU: informa al dispositivo de que su solicitud ha sido reconocida

Dispositivo: en respuesta, desactiva la señal de solicitud de interrupción

CPU: interrumpe el programa que se estaba ejecutando en ese momento

Pila  $\leftarrow$  PC

PC  $\leftarrow$  Pila

Se reanuda la ejecución del programa interrumpido

ISR

CPU: desactiva las interrupciones durante la primera instrucción o durante toda la ISR

¿Inhabilitar / habilitar interrupciones?

Guardar en la pila todos los registros que se modifiquen

Cuerpo principal de la ISR (por ej. E/S)

Restaurar todos los registros previamente guardados

Habilitar interrupciones si previamente se habían inhabilitado

Retorno de la ISR

# Causas de las interrupciones

## ■ Clasificación en función de las distintas situaciones que pueden requerir que el procesador sea interrumpido:

### 1. Fallo del hardware

- Los circuitos efectúan comprobaciones para verificar si funcionan correctamente, y si no es así, generan interrupciones
- Ejemplos:
  - Fallo de alimentación:
    - » Interrupción de la más alta prioridad que guarda registros (puede ser necesario conectar baterías).
  - Errores de paridad de memoria:
    - » Interrupción que reintenta la transferencia varias veces.

# Causas de las interrupciones

## 2. Errores de programa

- Situaciones de error introducidas por el programador
- Ejemplos:
  - Error de desbordamiento (overflow)
  - División por cero
    - » En ambos casos la interrupción cancela la ejecución del programa o transfiere el control a una función del usuario
  - Violación de la protección de memoria
    - » El programa de usuario intenta acceder a una dirección en una parte protegida de la memoria ⇒ interrupción que cancela la ejecución y genera mensaje de error (*segmentation fault*)
  - Ejecución de códigos de operación no válidos

# Causas de las interrupciones

## 3. Condiciones de tiempo real

- Se espera que el ordenador responda rápido a una situación
- Ejemplo:
  - Vigilancia de enfermos
    - » Interrupciones para hacer sonar timbres de alarma ante situaciones de peligro
  - Control de herramientas
    - » Interrupción para parar el equipo y evitar daños en la pieza o la máquina cuando algo falla

## 4. Entrada / salida

- Para poder hacer uso del procesador mientras un periférico no está listo para realizar la transferencia (E/S por interrupc.)
- ...o durante el tiempo que el perif. realiza la transfer. (DMA)
- ...es necesario que la interfaz del periférico o el controlador de DMA puedan interrumpir al procesador.

# Tipos de interrupciones

## ■ Clasificación en función de la procedencia de la interrupción:

### 1. Interrupciones externas

- Se inician a petición de dispositivos externos.
- ENMASCARABLES:
  - Se pueden habilitar o inhibir usando instrucciones del tipo EI (*Enable Interrupt*) y DI (*Disable Interrupt*).
  - Ejemplo: **interrupciones de E/S**
- NO ENMASCARABLES:
  - Tienen mayor prioridad que las enmascarables.
  - Ejemplo: para gestionar **fallos del hardware externo** al procesador y **condiciones de tiempo real**.

# Tipos de interrupciones

## 2. Interrupciones internas (o excepciones o traps)

- Se activan de forma interna al procesador mediante condiciones excepcionales, como **errores del programa o fallo del hardware interno.**

## 3. Interrupciones software

- Las instrucciones de interrupción software se emplean para realizar **llamadas al SO.**
- Son instrucciones más cortas que las de llamada a subrutina y no se necesita que el programa llamador conozca la dirección del SO en memoria.

# Determinación de la dirección de la ISR

## ■ ¿Cómo determinar la dirección de comienzo de la rutina de servicio de interrupción?

- Direcciones fijas (o interrupciones no vectorizadas):
  - La dirección o direcciones se fijan y definen en los circuitos del procesador.
  - Ejemplo: una dirección fija para la rutina de servicio de cada interrupción.
- Interrupciones vectorizadas
  - Este término genérico se refiere a todos los esquemas en los cuales el dispositivo que solicita una interrupción suministra de algún modo la dirección de la rutina de servicio.



# Determinación de la dirección de la ISR

- Interrupciones vectorizadas (1):
  - **Direccionamiento absoluto**
    - La interfaz suministra la dirección completa de su ISR
  - **Direccionamiento relativo**
    - La interfaz sólo envía parte de la dirección, que deberá ser completada por el procesador
      - » añadiendo más bits
      - » sumando una cantidad
    - ✓ Se reduce el número de bits que necesita transmitir la interfaz, con lo que se simplifica su diseño
    - ✗ Se limita el número de dispositivos que el procesador puede identificar automáticamente.
  - **Envío de una instrucción de bifurcación completa**
    - en lugar de una simple dirección.

# Determinación de la dirección de la ISR

- Interrupciones vectorizadas (2):
  - ⌚ Con los métodos anteriores, la ISR para un dispositivo determinado siempre debe comenzar en la misma localización
  - 😊 Para conseguir mayor flexibilidad:
    - el programador puede almacenar en esta localización una instrucción de salto a la rutina adecuada
    - Esto lo puede realizar de forma automática el mecanismo de gestión de interrupciones



- **Direccionamiento indirecto (interrupciones vectorizadas propiamente dichas)**
  - La dirección enviada por la interfaz es la **posición relativa en una tabla**, residente en MP, que contiene las direc. de las ISR.
  - Cada posición de la tabla se conoce como **vector de interrupción**.

# Determinación de la dirección de la ISR

- Para sincronizar la transmisión de la dirección, el procesador envía una señal de control:

- INTA# (*Interrupt Acknowledge*, o aceptación de interrupción)
- equivalente a RD# en lectura

**...causando que la fuente de interrupción sitúe la dirección en un bus, siendo entonces leída por el procesador.**

- Las señales de dirección se pueden enviar por un bus especial, por el bus de datos, o por el propio bus de direcciones.

# Identificación de la fuente de interrup.

Los computadores pueden tener conectados una gran variedad de dispositivos con poder de interrupción



La acción requerida al recibir una interrupción dependerá de la causa de ésta  
(de qué dispositivo la produjo)



Es necesario que haya diferentes rutinas de servicio de interrupción, para cada una de las causas



Es necesario identificar la causa o el dispositivo que produjo la interrupción

# Identificación de la fuente de interrup.

## ■ Soluciones:

- Una o múltiples líneas de interrupción con un dispositivo en cada línea
    - Cada fuente de interrupción generará una señal de interrupción en la línea apropiada.
    - La identificación del dispositivo es trivial y la dirección de comienzo de la ISR puede ser fija o vectorizada.
  - Una o múltiples líneas de interrupción, con más de un dispositivo por línea
    - Es normal tener varios dispositivos (fuentes de interrupciones) conectados a la misma línea de interrupción.
    - El dispositivo que solicita la interrupción ha de ser identificado
    - Esto puede realizarse por software o por hardware
- 

# Identificación de la fuente de interrup.

- Una o múltiples líneas de interrupción, con más de un dispositivo por línea
  - Identificación de la fuente de interrupción por **software** (técnica de sondeo o “polling”)
    - Se usa cuando la dirección de salto para todos los dispositivos conectados a una misma línea es única y fija.
    - La ISR debe identificar el dispositivo que solicita la interrupción, examinando de uno en uno los dispositivos de la línea, para transferir el control a la ISR del solicitante.
    - ✖ Método lento
    - ✓ Económico desde el punto de vista hardware
  - Identificación de la fuente de interrupción por **hardware**:
    - Se usan interrupciones vectorizadas.
    - ✓ Método rápido
    - ✖ Más costoso desde el punto de vista hardware

# Sistemas de prioridad en las interrupciones

■ En un computador con más de un dispositivo con capacidad de interrupción, hay que establecer mecanismos de prioridad que resuelvan los problemas:

- Interrupciones simultáneas

- Cuando se produce una interrupción, ésta no se acepta hasta que la instrucción ejecutada termine. Durante ese breve tiempo pueden haberse generado otras interrupciones, que requieren ser atendidas.
- ¿Qué interrupción se atiende primero?

- Interrupciones anidadas

- Se puede producir una interrupción antes de haber atendido completamente la anterior
- ¿Debe terminarse de atender la primera interrupción, o se ha de aceptar y atender inmediatamente la nueva solicitud?

- Inhibición de interrupciones

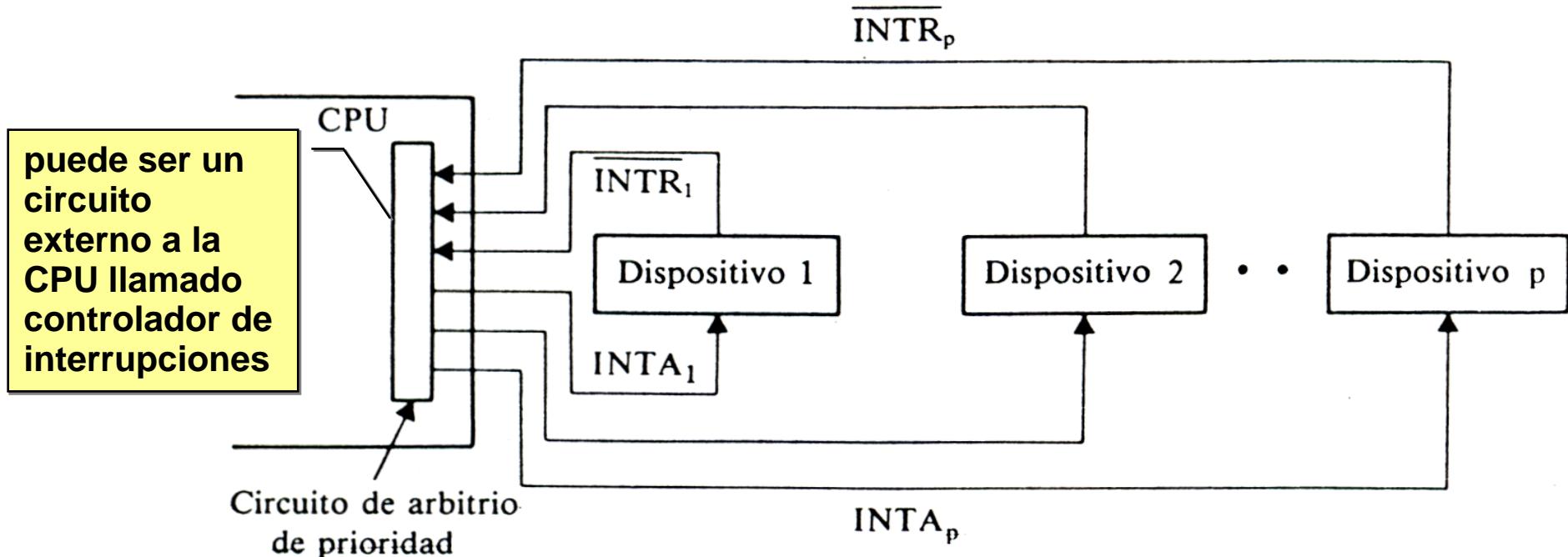
- Mecanismos de prioridad para permitir que cada programa defina los tipos de interrupciones que puede tolerar

# Interrupciones simultáneas (I)

- Si llegan solicitudes de interrupción de dos o más dispositivos, el procesador debe disponer de algún medio para que sólo se dé servicio a una solicitud, y el resto se retrase o no se tenga en cuenta.

- Gestión de prioridades centralizada

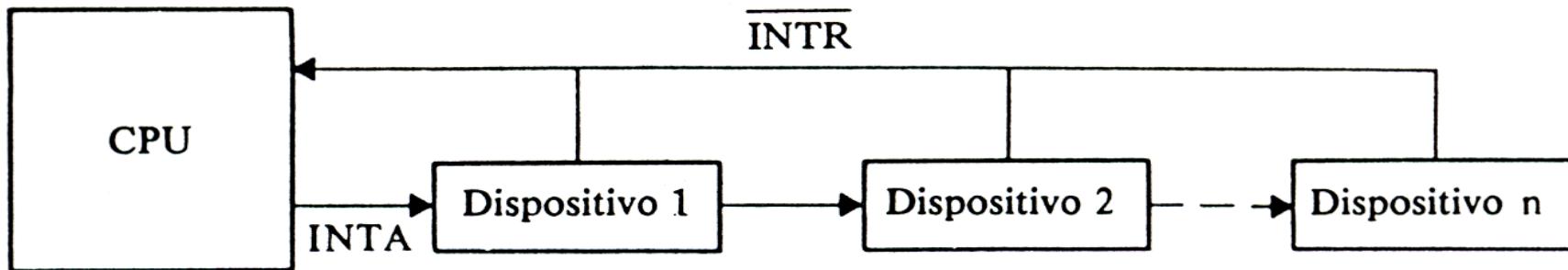
- Cuando hay un solo dispositivo en cada línea de interrupción, la



# Interrupciones simultáneas (II)

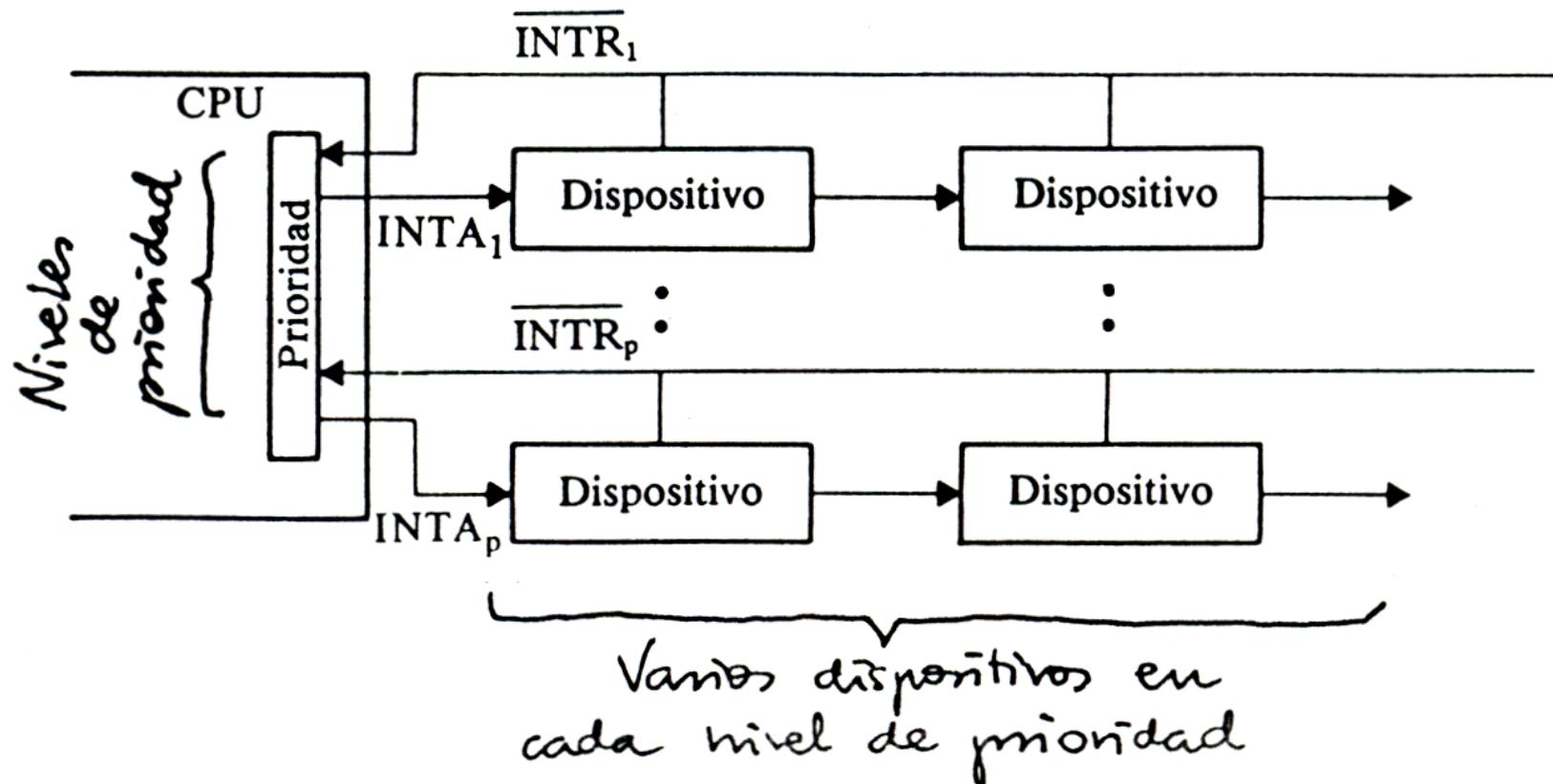
## ■ Gestión de prioridades distribuida

- Cuando varios dispositivos comparten una única línea de solicitud de interrupción es necesario implantar un mecanismo para asignar prioridades a estos dispositivos:
  - **Técnica de sondeo o “*polling*”**, que como hemos visto se usa para determinar por software el origen de una interrupción.
    - » La prioridad se implanta de forma automática según el orden en el que se escrutan los dispositivos.
    - » Las prioridades se pueden cambiar modificando la ISR para que sondee en un orden diferente.
  - **Técnica de encadenamiento o “*daisy-chain*”**



# Interrupciones simultáneas (III)

- Gestión de prioridades híbrida
  - Combinación de esquemas centralizado y distribuido (daisy-chain).

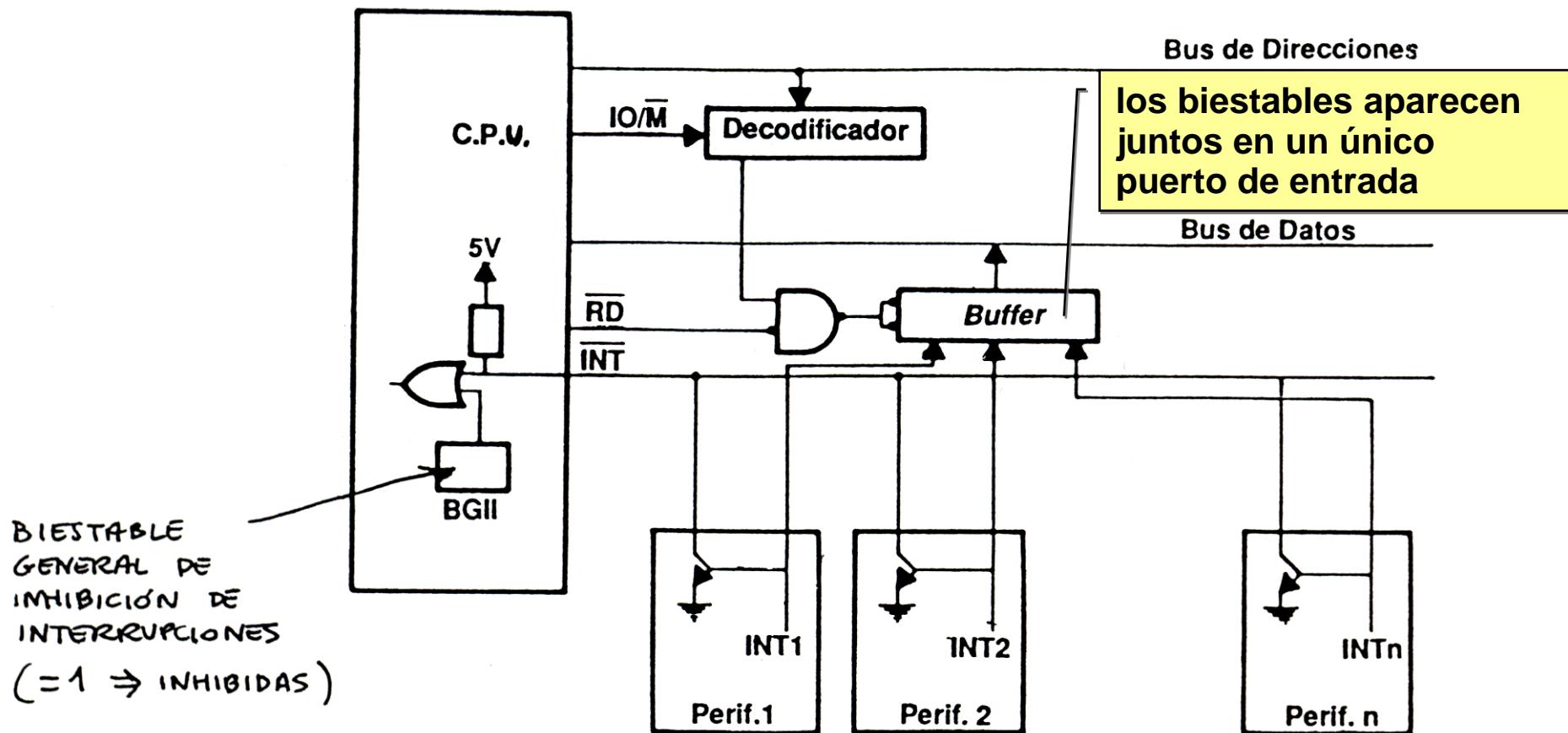


# (POLLING I)

- Se usa para
  - Identificar el origen de una interrupción
  - Establecer un mecanismo software de asignación de prioridades a los dispositivos
- En esta solución, el ordenador dispone normalmente de una única línea de interrupción INT#, que sirve para que cualquier dispositivo solicite una interrupción.
- La línea INT# se organiza en colector abierto (OR cableado)
  - Cualquier dispositivo puede poner INT<sub>i</sub>=1 para solicitar la interrupción, lo que hace que INT# se active (se ponga a 0).
- La ISR está en una posición de memoria fija y se encarga de identificar cuál es el dispositivo que interrumpió, comprobando el valor de los biestables de interrupción de los dispositivos, que estarán a 1 para aquellos dispositivos que solicitan interrupción.

# (POLLING II)

- La asignación de prioridades se hace por el orden en el que la ISR analiza los biestables de interrupción de los periféricos.

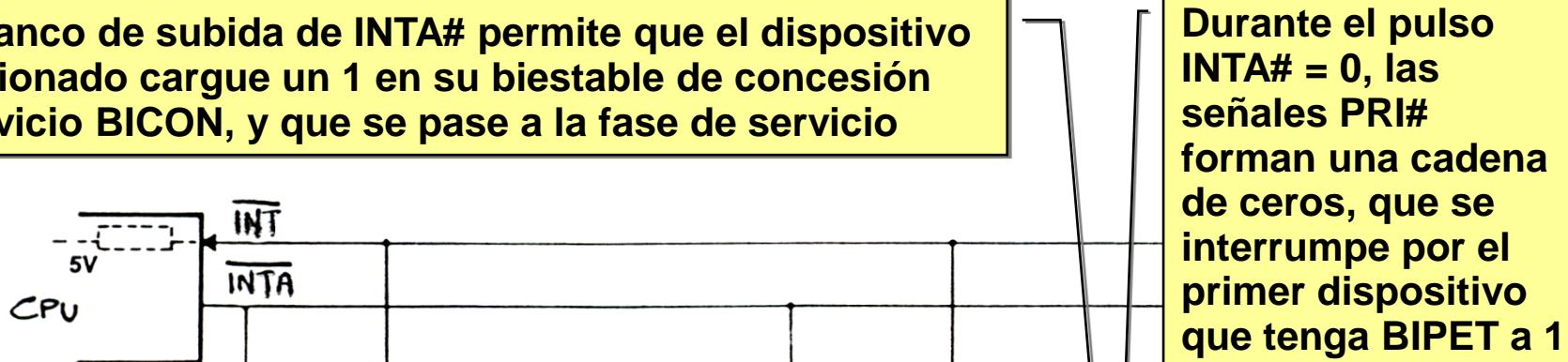


# (DAISY-CHAIN I)

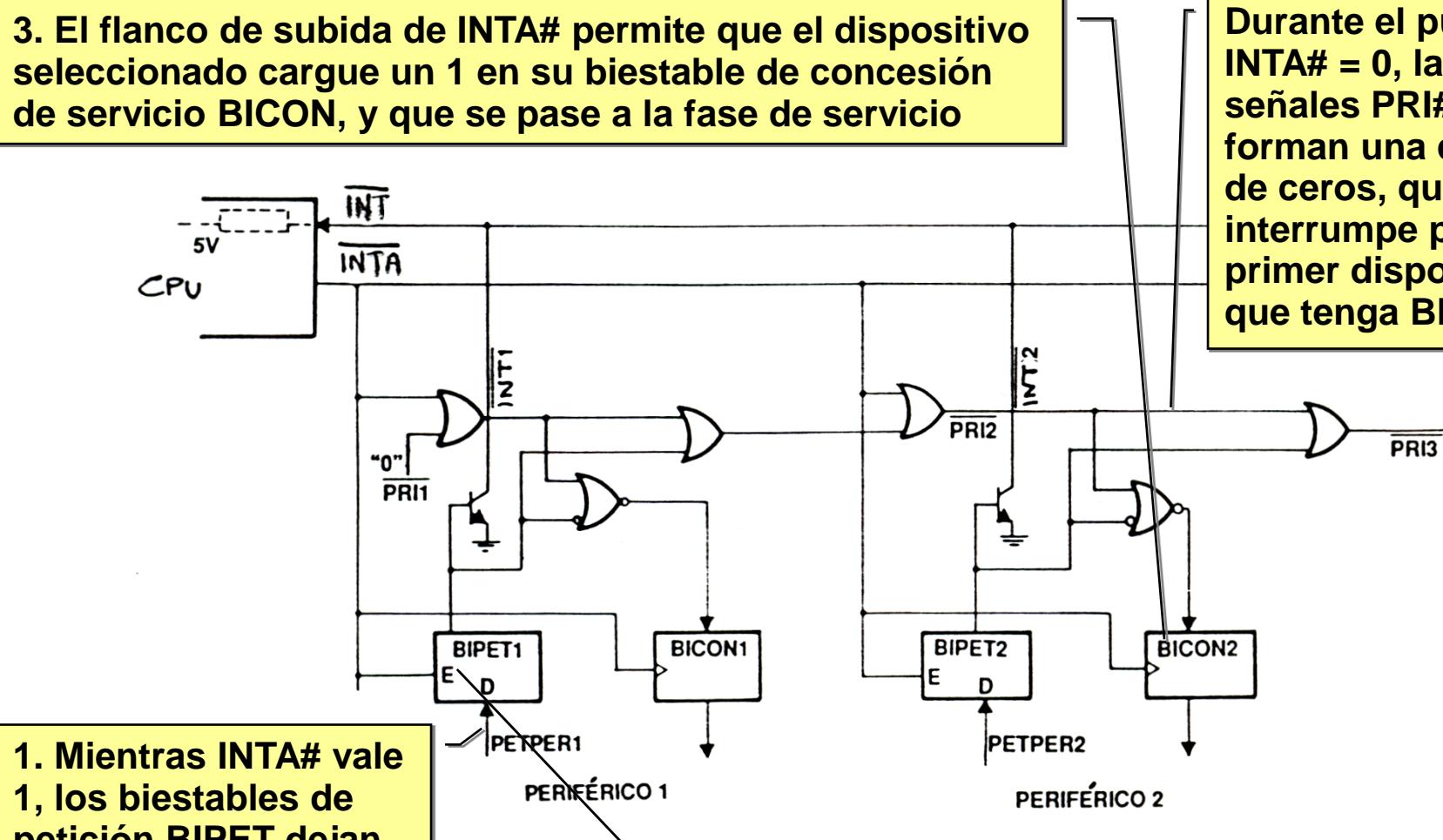
- Se usa para establecer un mecanismo hardware de asignación de prioridades a los dispositivos.
- Se basa en dos señales comunes a todos los peticionarios y a la CPU:
  - INT#: Petición de interrupción
  - INTA#: Concesión o aceptación de interrupción
- Los peticionarios se conectan a INT# en colector abierto.
  - Esto permite que uno o varios dispositivos soliciten simultáneamente la interrupción, poniendo un 0 en INT#
- La señal INTA# sirve, a modo de testigo, para que uno solo de los peticionarios sea atendido.
  - Es un pulso que recorre en serie, uno tras otro, los dispositivos.
  - Es tomado y eliminado por el primero que desea ser atendido.

# (DAISY-CHAIN II)

3. El flanco de subida de INTA# permite que el dispositivo seleccionado cargue un 1 en su biestable de concesión de servicio BICON, y que se pase a la fase de servicio



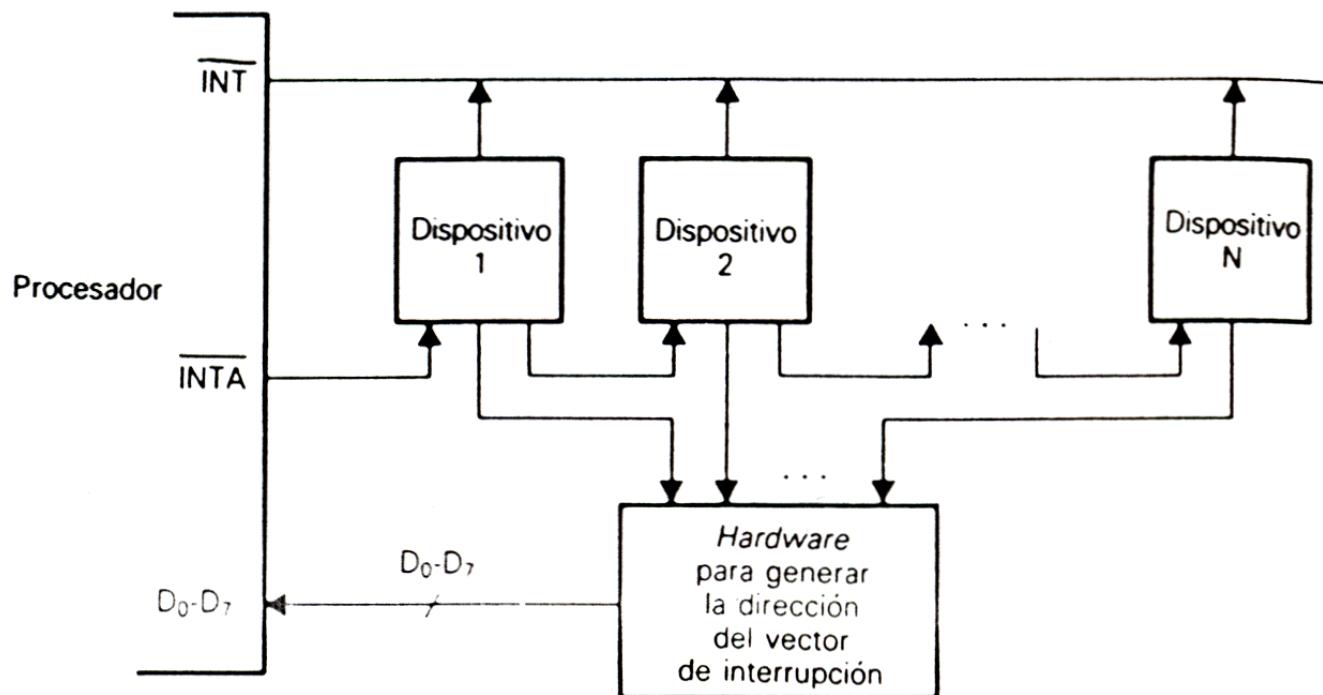
1. Mientras  $\text{INTA}\#$  vale 1, los biestables de petición  $\text{BIPET}$  dejan pasar las peticiones internas  $\text{PETPER}$  de sus respectivos periféricos



2. Cuando llega un pulso de concesión  $\text{INTA}\# = 0$ , se congelan los biestables  $\text{BIPET}$ , de forma que la situación de solicitud de los periféricos no pueda variar

# (DAISY-CHAIN III)

- Identificación de la fuente de interrupción:
  - Alternativa 1: la ISR puede determinar qué dispositivo interrumpió leyendo los biestables BICON.
  - Alternativa 2: se pueden usar interrupciones vectorizadas



# Interrupciones anidadas (I)

## ■ En muchas ocasiones se produce una interrupción mientras se está atendiendo otra. ¿Qué hacer?

- Inhabilitar las interrupciones durante la ejecución de la ISR
  - La ejecución de la ISR, una vez iniciada, siempre continuará hasta su finalización, antes de que la CPU acepte una segunda solicitud de interrupción.
  - Esta solución es válida cuando las ISR sean breves.
    - Dispositivos simples para los cuales el retraso posible en la respuesta a la (segunda) solicitud sea aceptable.
- Permitir que la CPU acepte la segunda solicitud de interrupción durante la ejecución de la ISR, si su prioridad es mayor
  - ya que, para algunos dispositivos, un largo retraso en la respuesta a una solicitud de interrupción los podría llevar a funcionar de forma errónea.

# Interrupciones anidadas (II)

- Ejemplo: Reloj de tiempo real
  - Envía solicitudes de interrupción a la CPU a intervalos regulares ⇒ ejecución de breve ISR que incrementa la hora (contadores de memoria).  
⇒ Necesidad de que se atienda la interrupción antes de que se produzca de nuevo.  
⇒ La CPU debe aceptar la solicitud de interrupción aunque se esté atendiendo a otro dispositivo.
- Durante la ejecución de una ISR se aceptarán solicitudes de interrupción de algunos dispositivos, pero no de otros, según sea su prioridad.
- Cada vez que se acepta una nueva interrupción, el contenido de PC se transfiere a la pila, y una vez atendida se toma de ésta.
  - La pila debe ser lo suficientemente grande para que las interrupciones se puedan anidar a suficiente profundidad.

# Inhibición de interrupciones (I)

- Hay situaciones en las que conviene evitar temporalmente que se produzcan interrupciones.
- Tres niveles de desactivación de interrupciones:
  1. Desactivar todas las interrupciones
    - Para ello se puede usar un Biestable General de Inhibición de Interrupciones
      - Si está a 1 (por ej.), el programa no podrá sufrir interrupciones.
      - Se puede cambiar su estado mediante instrucciones EI (Enable Interrupts) y DI (Disable Interrupts).
    - Independientemente del valor de BGII, la CPU puede desactivar las interrupciones automáticamente:
      - Durante la ejecución de la primera instrucción de la ISR
        - Si es una instrucción DI  $\Rightarrow$  el programador se asegura de que no ocurrirán más interrupciones hasta ejecutar EI.
      - Durante toda la ISR
        - No ocurrirán más interrupciones hasta ejecutar EI o retornar de la interrupción.

# Inhibición de interrupciones (II)

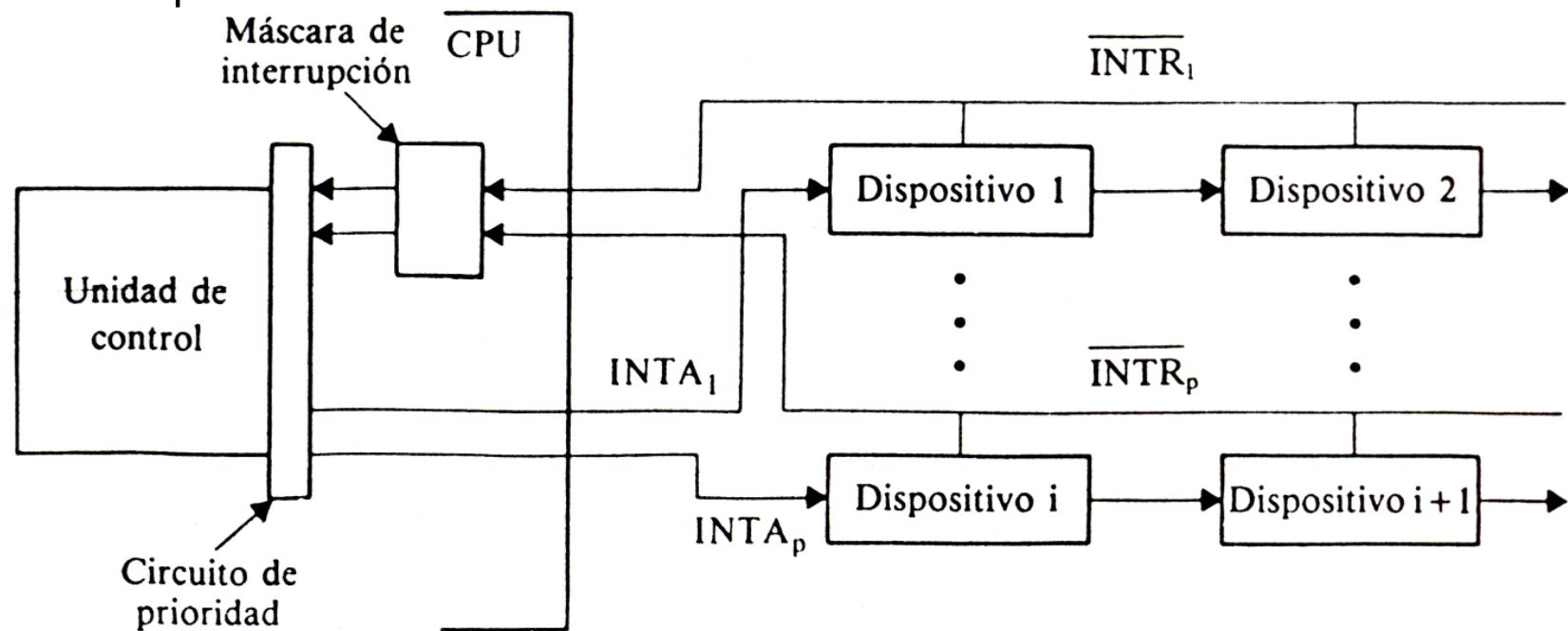
## 2. Desactivar interrupciones de inferior o igual prioridad

- Se puede tener un registro (o varios bits del registro de estado del procesador) con el valor del menor nivel que puede interrumpir.
- Un decodificador se encarga de desactivar todos los niveles de menor prioridad.
- La CPU tendría así un nivel de prioridad (prioridad del programa que se está ejecutando), y sólo aceptará interrupciones de dispositivos con prioridades mayores que la suya.

# Inhibición de interrupciones (III)

## 3. Desactivar de forma selectiva determinados niveles de interrupción

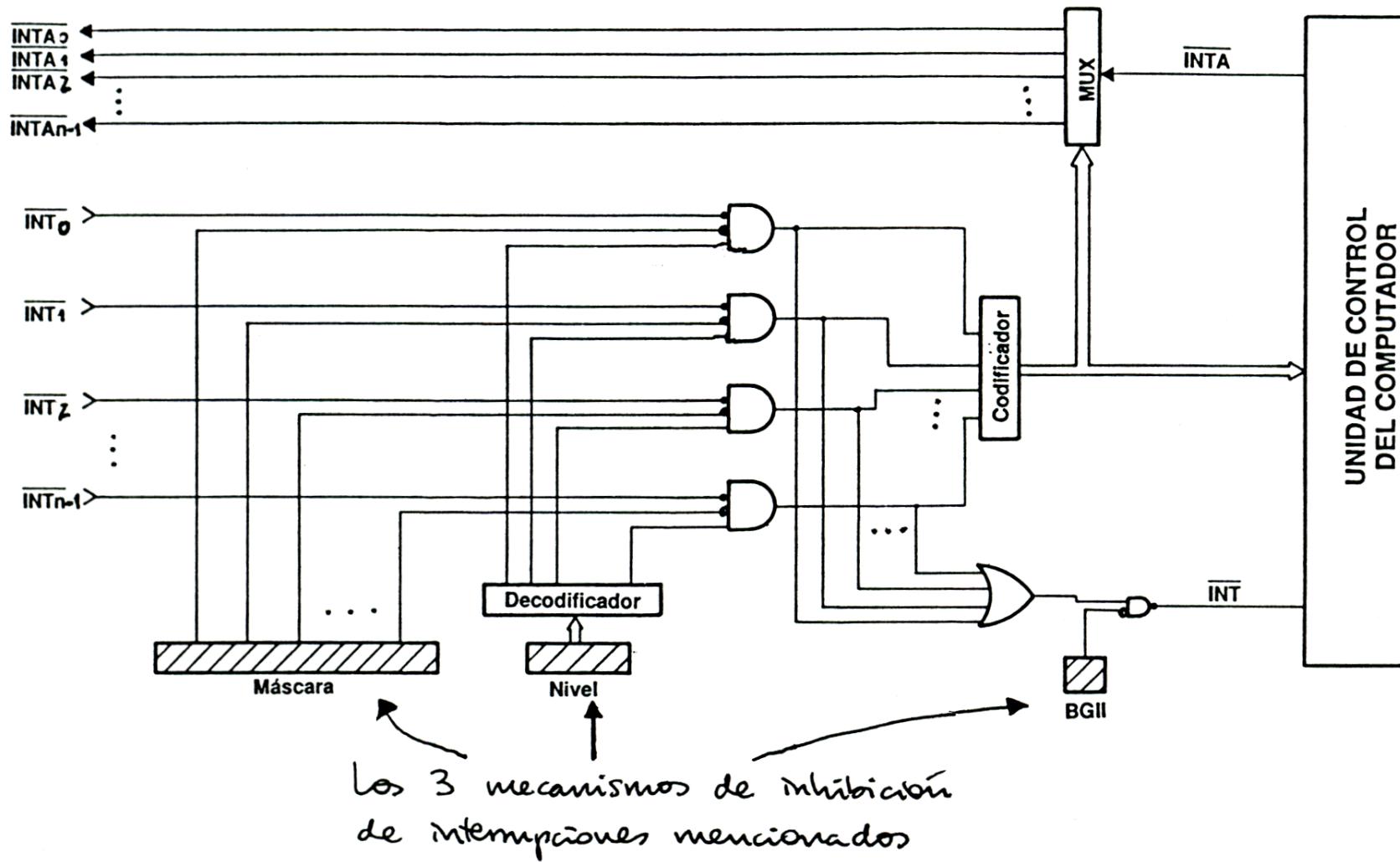
- La señal INTR# de cada nivel se puede enmascarar haciendo la operación AND con el bit correspondiente de un registro máscara de interrupción.
- El programa cargará un 1 en aquellos niveles de interrupción que se deseen inhibir.



# Inhibición de interrupciones (IV)

## Ejemplo:

n  
NIVELES  
DE  
PRIORIDAD



# Inhibición de interrupciones (V)

- Las señales INT<sub>i</sub># deben atravesar un doble filtro:
  - Su correspondiente bit de máscara ( $=1 \Rightarrow$  nivel inhibido)
  - La condición de que  $i \leq \text{Nivel}$
- El decodificador convierte el contenido del registro de nivel en los correspondientes ceros y unos en cada puerta AND para filtrar las peticiones de interrupción.
- La señal INT# es la que realmente produce la interrupción.
  - Se genera si alguna señal INT<sub>i</sub># consigue atravesar su puerta AND correspondiente, y además el biestable general de inhibición de interrupciones está a 0.
- El codificador genera el valor  $i$  de la señal INT<sub>i</sub># de mayor nivel que atraviesa el filtro, para:
  - controlar el multiplexor que encamina la señal INTA# a su nivel correspondiente
  - uso interno de la unidad de control, indicándole cual es el nivel a atender (funcionando como un vector de interrupción)

# Ejemplo de ISR

## ■ ISR “outwd” para 8080:

- Escribe bytes de una posición fija de memoria (data) en un dispositivo de salida (puerto 9).
- Otras interrupciones son inhabilitadas durante la ejecución de “outwd” (lo hace el 8080 automáticamente).

```
outwd:push psw      ; Salvar A e indicadores en la pila
      lda data    ; A <-- M[data]
      out 9       ; Puerto 9 <-- A
      pop psw      ; Restaurar A e indicadores
      ei          ; Habilitar sistema de interrup.
      ret         ; Retornar a programa interrumpido
```

El sistema de interrupciones del 8080 no es habilitado por EI hasta después de ejecutada la instrucción que sigue a EI (RET en este caso). Esto asegura que la ISR se completa antes de que la CPU entre en otro ciclo de interrupción.

# Ej. de circuito controlador de interrupciones: 8259

**La mayoría de las familias de microprocesadores contienen circuitos especiales denominados “controladores de interrupciones”, que**

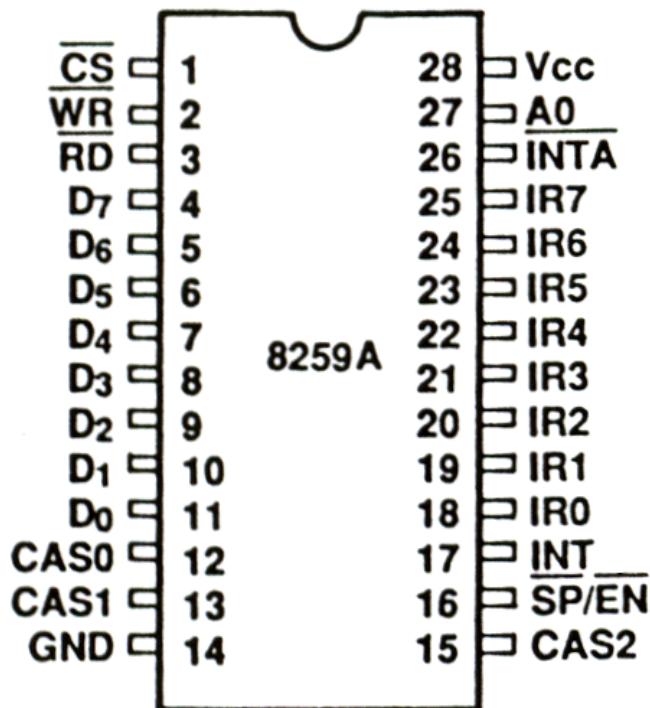
- simplifican las diversas tareas de diseño (prioridades, enmascaramiento,...) asociadas con el control de las interrupciones externas, y
- permiten aumentar el número de líneas de petición de interrupción disponibles.

## Ej. de circuito controlador de interrupciones: 8259

### ■ Programmable Interrupt Controller, PIC

- Permite manejar 8 líneas de interrupciones vectorizadas con prioridad.
- Compatible con las familias 8085 y 8086.
- Prioridades alterables por software.
- Permite enmascaramiento individual de cada línea.
- Es posible conectar 9 controladores para manejar hasta 64 niveles de interrupción.

### ■ Más información en apéndice



# Interrupciones en el PC (modo real)

## ■ Interrupciones vectorizadas:

- Existen 256 interrupciones posibles (0 a 255 = 0xFF), vectorizadas.
- Tabla de vectores de interrupción:
  - primeros 1024 bytes de memoria (0 a 0x3FF).
- Cada vector de interrupción:
  - doble palabra (32 bits, 4 bytes)
  - dirección de la ISR asociada a esa interrupción

Segmento (CS)
Desplazamiento (IP)

## ■ Más información en apéndice

# Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

# Concepto de DMA

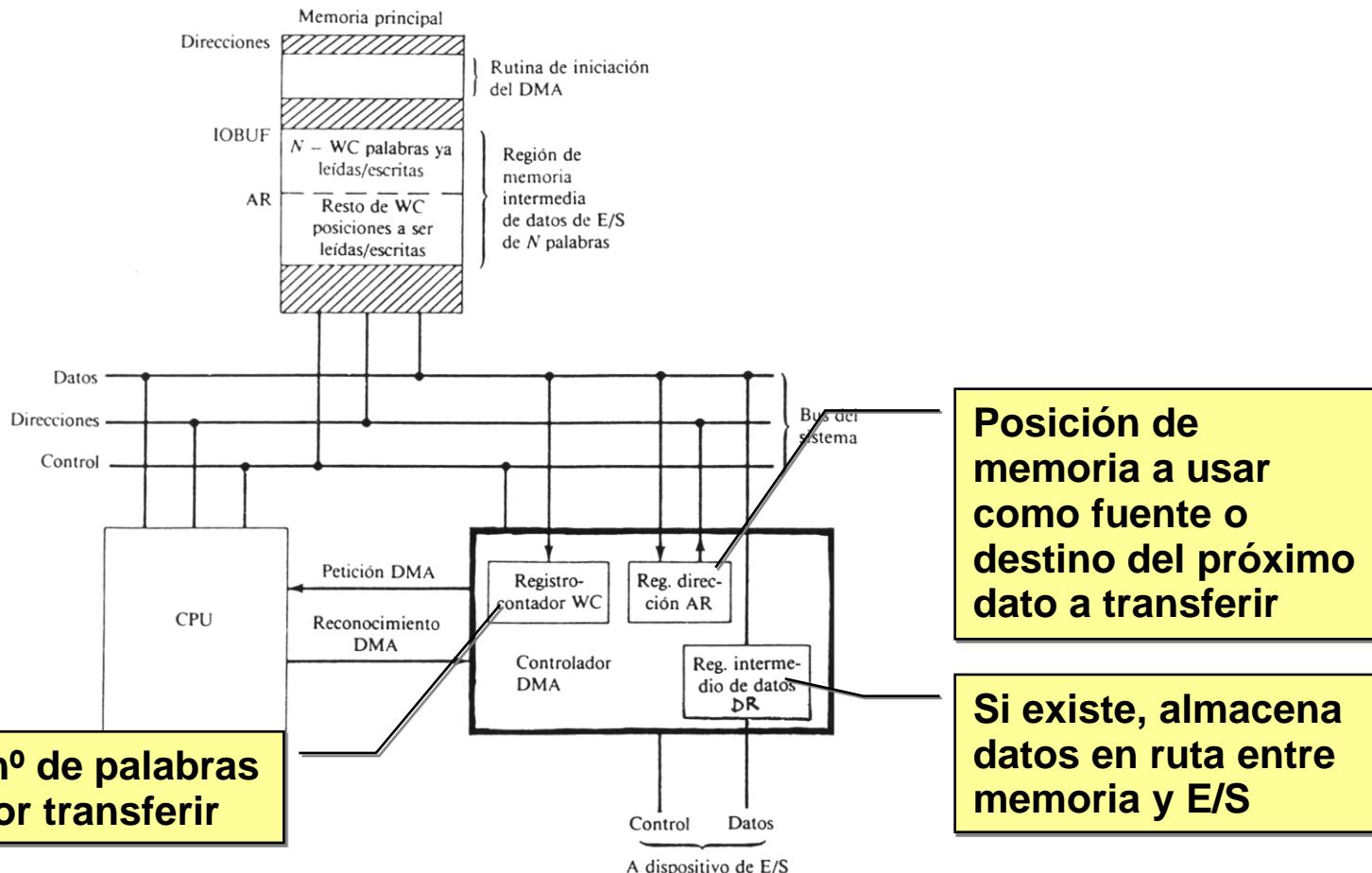
## ■ DMA (*Direct Memory Access*):

- Técnica que permite realizar transferencias de datos **entre la memoria y los dispositivos de E/S sin intervención directa del procesador.**
  - No se ejecutan instrucciones en el procesador para realizar la transferencia.
- Permite transferencias a la **máxima velocidad permitida por el bus del sistema, la memoria y el periférico.**
  - En sistemas con un único bus, esta velocidad es mucho mayor que la máxima posible con E/S controlada por el procesador:
    - DMA: un ciclo del bus por palabra (pocos ciclos de reloj)
    - E/S controlada por procesador: ejecución de varias instrucciones para cada palabra (muchos ciclos de reloj)
- Se utiliza con **dispositivos rápidos:**
  - Discos, tarjetas gráficas, tarjetas de red, etc.

# Controlador de DMA

## ■ El DMA utiliza un controlador de DMA (DMAC).

- Chip que genera las señales de control y direcciones, actuando como maestro del bus.



Especifica el nº de palabras que quedan por transferir

Posición de memoria a usar como fuente o destino del próximo dato a transferir

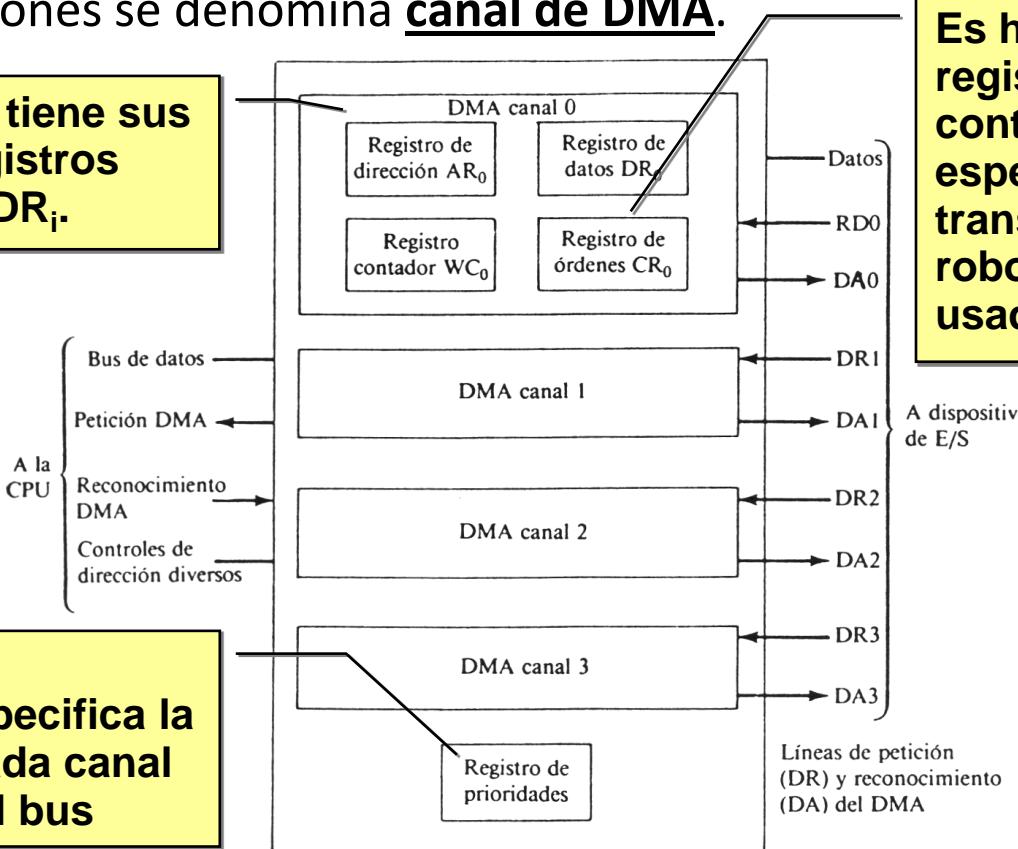
Si existe, almacena datos en ruta entre memoria y E/S

# Controlador de DMA

- Existen DMAC que permiten varias operaciones de DMA independientes.

- La lógica de control y los registros asociados con cada una de esas operaciones se denominan **canal de DMA**.

Cada canal tiene sus propios registros  $AR_i$ ,  $WC_i$  y  $DR_i$ .



El registro de prioridades especifica la prioridad de cada canal para acceder al bus

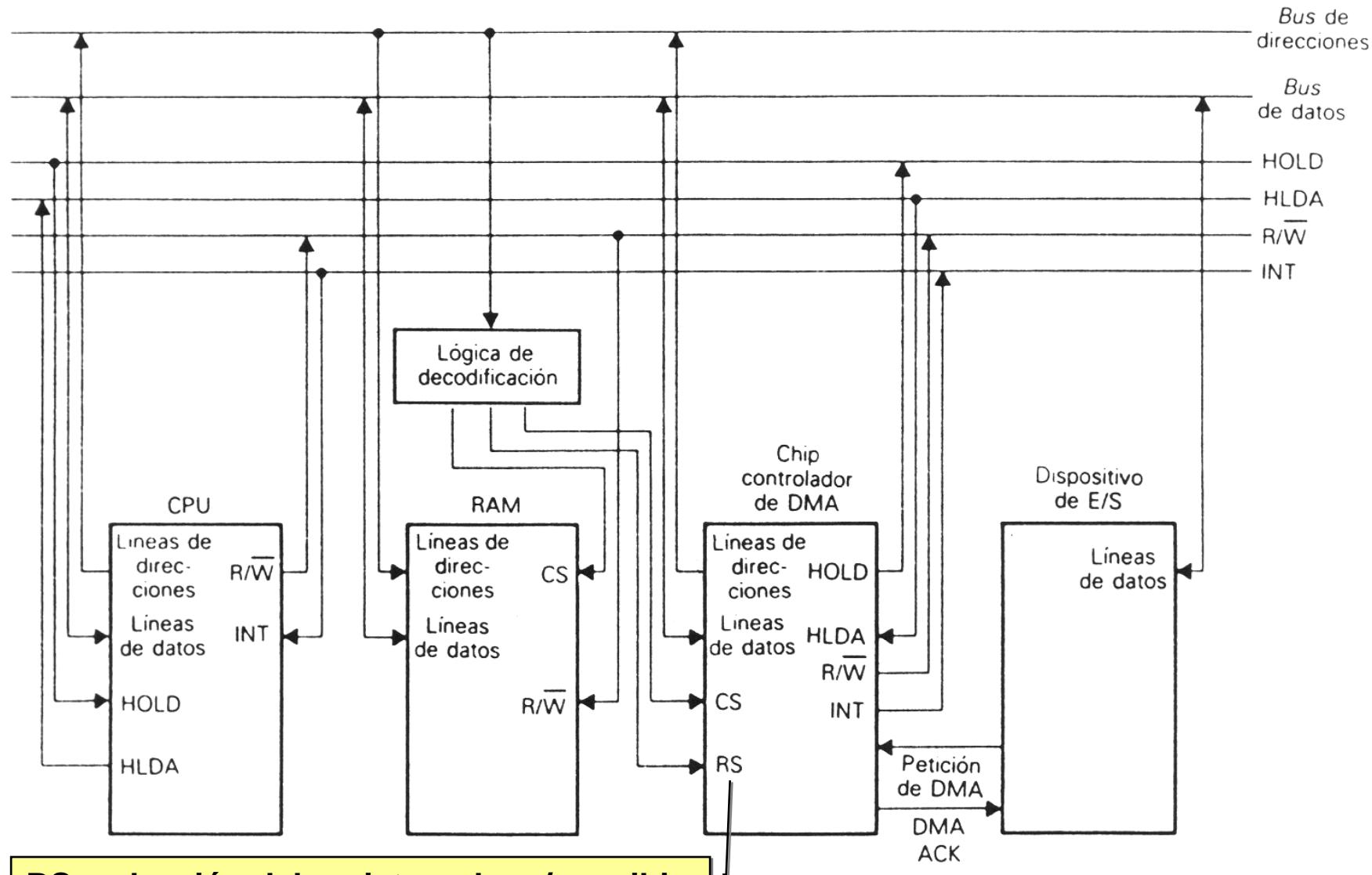
Es habitual que existan registros de órdenes o control ( $CR_i$ ) que especifican el modo de transferencia (bloques, robo de ciclo, etc.) usado por cada canal.

# Señales de control

## ■ El DMA y el procesador deben compartir el bus.

- Necesidad de un mecanismo de control de acceso al bus.
- Usualmente se usan dos líneas de control entre DMA y procesador:
  - Petición / solicitud de DMA (*DMA Request*) o del bus (*Bus Request*):
    - HOLD o BSRQ#
  - Reconocimiento / cesión de DMA (*DMA Acknowledge*) o del bus (*Bus Acknowledge*):
    - HLDA o BSAK#
- Cuando el DMA necesita obtener el control del bus, activa HOLD.
- El procesador responde aislando del bus y activando HLDA, para indicar que ha entregado el bus.
- El DMA realiza la transferencia.
- El DMA se desconecta del bus y desactiva HOLD.

# Señales de control



**RS: selección del registro a leer / escribir**

# Secuencia de eventos

- Una operación de E/S por DMA se establece ejecutando una corta rutina de inicialización:
  - Varias instr. de salida para asignar valores iniciales a:
    - AR: Dirección de memoria de la región de datos de E/S IOBUF
    - WC: Número N de palabras de datos a transferir
- Una vez inicializado, el DMAC procede a transferir datos entre IOBUF y el dispositivo de E/S:
  - Se realiza una transferencia cuando el dispositivo de E/S solicite una operación de DMA a través de la línea de **petición del DMAC**.
  - Después de cada transferencia, se hace **WC--** y **AR++**
  - **La operación termina cuando  $WC = 0 \Rightarrow$**  el DMAC (o el periférico) indica la conclusión de la operación enviando al procesador una **petición de interrupción**.

# Secuencia de eventos detallada

- 
1. El procesador inicializa el DMA programando AR y WC.
  2. El dispositivo de E/S realiza una petición de DMA al DMA.
  3. El DMA activa la línea de petición de DMA al procesador.
  4. Al final del ciclo del bus en curso, el procesador pone las líneas del bus en alta impedancia y activa la cesión de DMA.
  5. El DMA asume el control del bus.
  6. El DMA responde al dispositivo de E/S con aceptación.
  7. El dispositivo de E/S transmite una nueva palabra de datos al registro intermedio de datos del DMA.
  8. El DMA ejecuta un ciclo de escritura en memoria para transferir el registro intermedio a la posición M[AR].
  9. El DMA decrementa WC e incrementa AR.
  10. El DMA libera el bus y desactiva la línea de petición de DMA.
  11. El DMA compara WC con 0:
    - Si  $WC > 0 \Rightarrow$  se repite desde el paso 2.
    - Si  $WC = 0 \Rightarrow$  el DMA se detiene y envía una petición de interrupción al procesador.

# Métodos de control de DMA

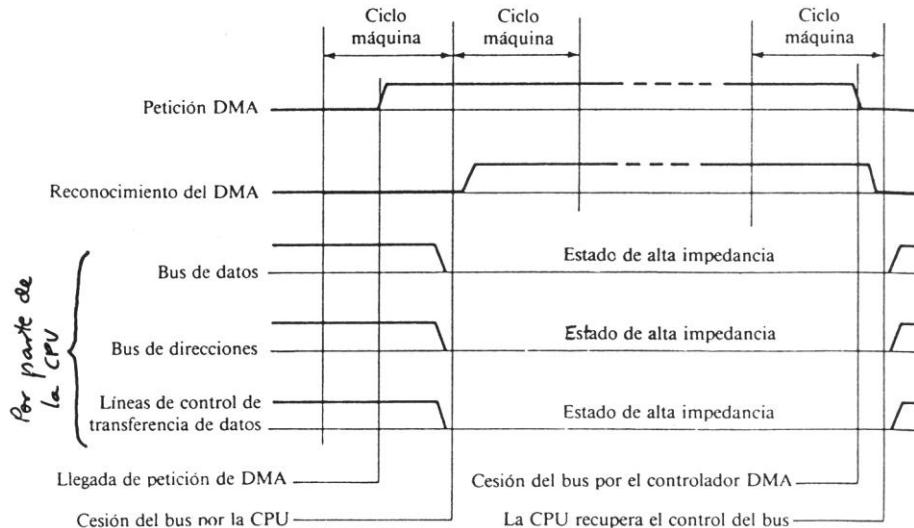
## ■ Formas de realizar el DMA:

- Robo de ciclo (*Cycle Stealing DMA*)
- Transferencia de bloques o parada de CPU (*Block Transfer DMA*)
- DMA intercalado o transparente (*Interleaved DMA*)
- Memoria multipuerto

# Métodos de control de DMA

## ■ Robo de ciclo (*Cycle Stealing DMA*)

- La transferencia de un bloque se produce palabra a palabra.
- El DMAC “roba” periódicamente uno o varios ciclos máquina al procesador, durante los cuales utiliza el bus. El procesador utiliza el resto de ciclos del bus.
- Cuando el DMAC solicita el bus, debe esperar a que el procesador complete el ciclo máquina en curso.
  - Un robo de ciclo puede aceptarse en mitad de una instrucción (hace que la duración de las instrucciones no sea fija).



# Métodos de control de DMA

## ■ Transferencia de bloques o parada de CPU (*Block Transfer DMA*)

- Se transmite un bloque (secuencia de palabras de datos) en una ráfaga continua.
- El DMAC toma el control del bus durante todo el período que dura la transferencia de datos.
- El procesador no tiene acceso al bus hasta que la transferencia termina, lo que le obliga a esperar períodos de tiempo relativamente grandes.

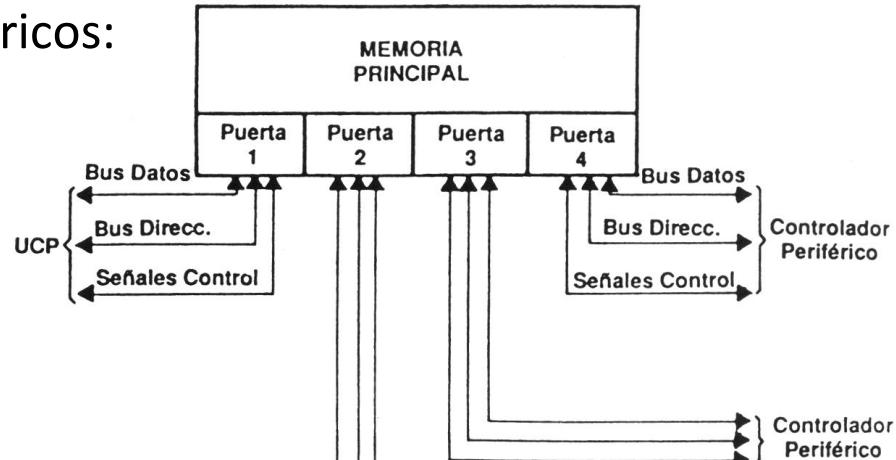
## ■ DMA intercalado o transparente (*Interleaved DMA*)

- El DMAC toma el bus cuando el procesador no lo utiliza.
  - Por ej., mientras está decodificando un código de operación o efectuando operaciones internas de transferencia de registros.
- Para ello, el procesador debe emitir las señales de control adecuadas.

# Métodos de control de DMA

## ■ Memoria multipuerto

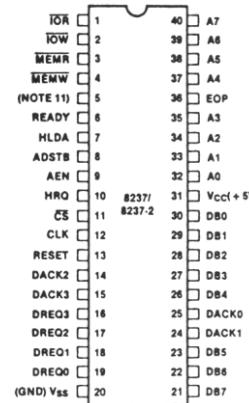
- Existen memorias con varios módulos y varios registros de dirección y de memoria que pueden operar simultáneamente conectados a varios buses, siempre que no direccionen un mismo módulo de memoria.
- Se puede conectar a un puerto el procesador y a los demás puertos varios controladores de periféricos:



- Si hay conflictos de acceso (peticiones simultáneas a un mismo módulo), se concede un acceso y se retardan los demás, según un sistema de prioridades.

# Ejemplo de DMAC: 8237

- ***Programmable DMA Controller***
  - 4 canales de DMA independientes
- **Más información en apéndice**



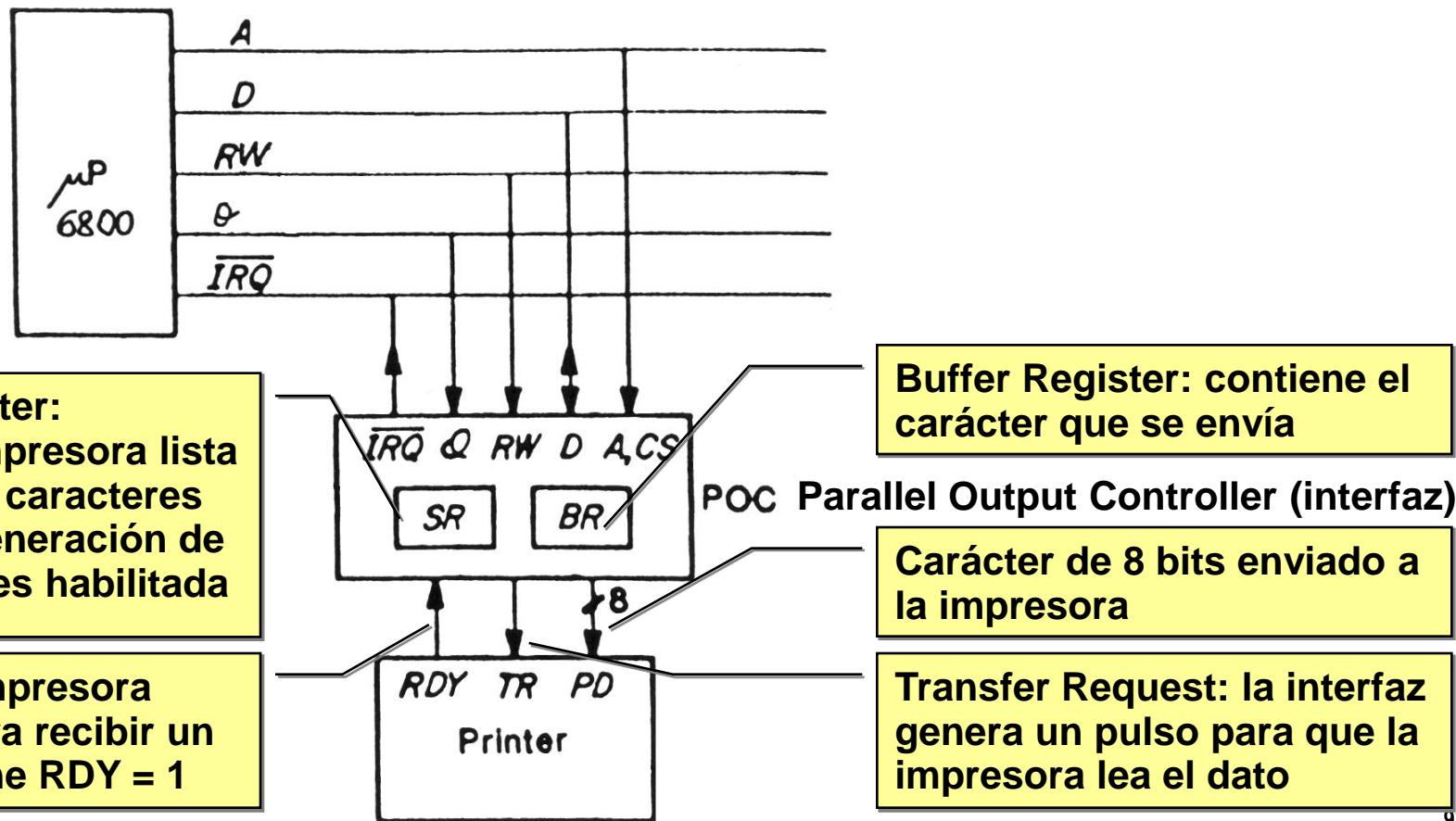
# Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Ejemplos de E/S
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

# Ejemplo de transferencia de E/S usando las tres técnicas

## ■ Objetivo:

- Enviar un bloque a la impresora (200 caract., desde dir. 3000h)
- Configuración para E/S programada y por interrupciones:



# Ejemplo de transferencia de E/S usando las tres técnicas

## ■ E/S por programa:

```
; Inicialización  
LDX    #$3000      ; RegIX<-3000h (bloque  
almacenado          ; a partir de la dir. 3000h)  
LDAB   #200         ; RegB<-200 (200 caract.)  
JSR    PRBLQ  
...
```

**E/S programada**

## Ejemplo de transferencia de E/S usando las tres técnicas

; Imprimir bloque

```

PRBLQ LDAA 0,X          ; RegA <- M[RegIX+0]
JSR    PRCHR
INX
DECB
BNE    PRBLQ           ; Si RegB!=0 => enviar otro car.
RTS

```

E/S programada

```

PRCHR TST   PSR           ; N (bit de signo) <- SR7
      BPL   PRCHR          ; Si N = 0 => esperar
      STAA  PBR           ; BR <- A, impresora pone SR7 a 0
RTS

```

PSR y PBR son etiquetas para los puertos SR y BR,  
mapeados en memoria

# Ejemplo de transferencia de E/S usando las tres técnicas

## ■ E/S por interrupciones:

- Cuando la interfaz está lista para recibir datos (impresora lista), interrumpe al 6800 por su línea IRQ#, forzándole a transferir el control a una ISR

**E/S por interrupciones**

; Inicialización

```
LDX    #$3000      ; RegIX<-3000h (bloque almacenado
                    ; a partir de la dir. 3000h)
STX    BA          ; BA (Byte Address) <- RegIX
LDAA   #200        ; RegA<-200 (bloque 200 caract.)
STAA   BC          ; BC (Byte Count) <- RegA
LDAA   #$01        ; RegA <- 1
STAA   PSR         ; SR0 <- 1 (interfaz generará
                    ; interrupciones)
```

BA	RMB	2	; Reserve Memory Byte (2 bytes)
BC	RMB	1	; Reserve Memory Byte (1 byte)

# Ejemplo de transferencia de E/S usando las tres técnicas

- Puede haber varios dispositivos conectados a IRQ#.
- La ISR comienza chequeando los bits “listo” de cada dispositivo, para ver cuál ha interrumpido. Se supone que la impresora tiene la mayor prioridad en este caso ⇒ es chequeada en primer lugar.

## E/S por interrupciones

```

; ISR
ORG $5000
ISR TST PSR           ; Sondear impresora
      BPL POLLTERM    ; Si SR7 = 0 => saltar a Terminal
      BRA PRINT        ; Si SR7 = 1 => saltar a PRINT
POLLTERM
      TST TSR          ; Sondear otros dispositivos,
      . . .             ; comenzando por el terminal
ORG $FFF8
FDB ISR               ; Form Double Byte
                        ; (Vector de interrupción)

```

**Cuando tiene lugar una interrupción, el 6800 toma la dirección de la ISR de M[FFF8h]**

# Ejemplo de transferencia de E/S usando las tres técnicas

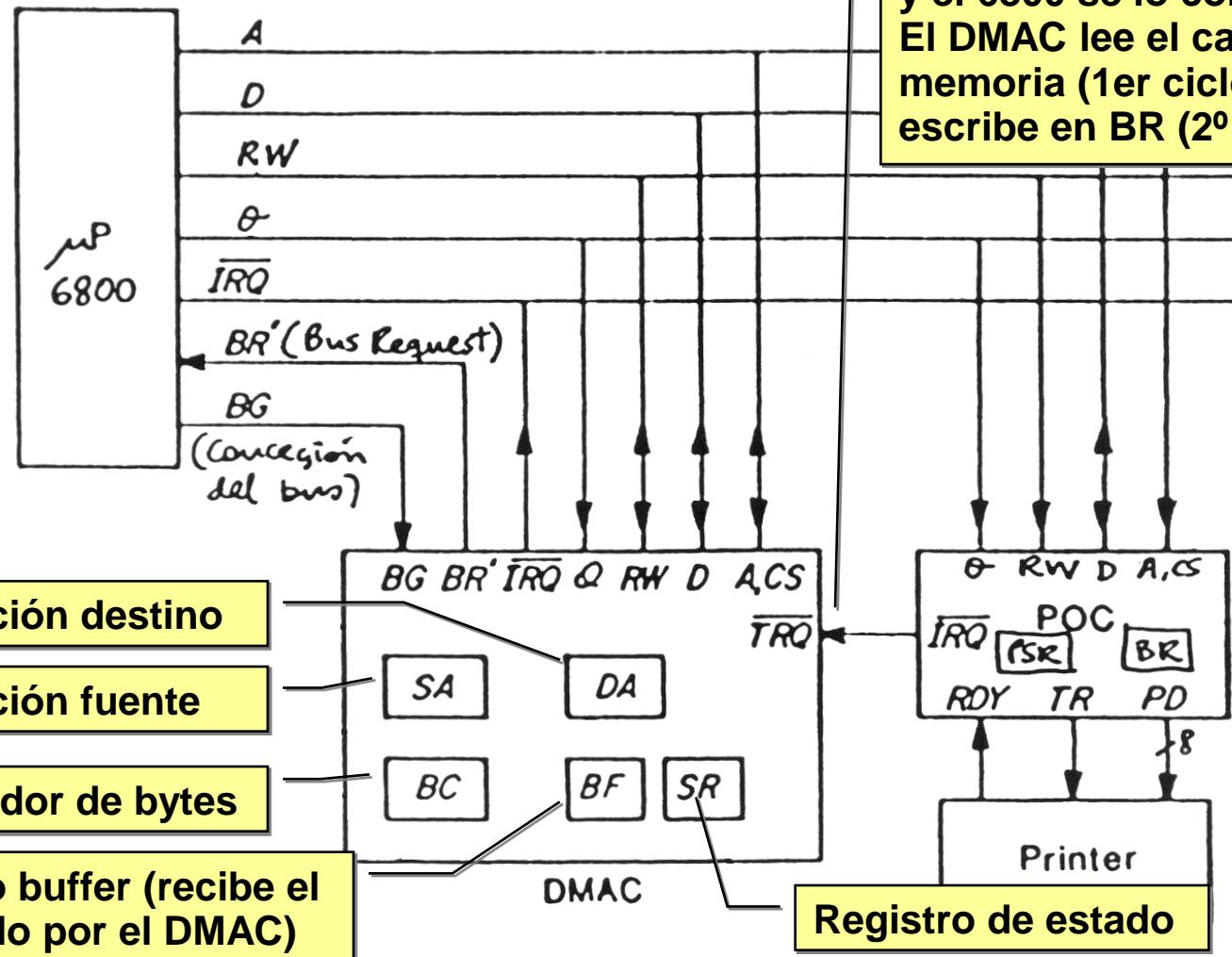
## E/S por interrupciones

```
PRINT LDX BA ; RegIX <- Byte Address
               LDAA 0,X ; RegA <- M[RegIX+0] (sig. car.)
               STAA PBR ; Enviar carácter
               INX   ; RegX++ (apuntar a sig. caráct.)
               STX   BA
               DEC   BC ; RegB-- (actualizar contador)
               BNE   RETURN ; Si BC!=0 => retorno subrutina
               CLR   PSR ; SR0 <- 0 si trans. blq. compl.

RETURN
               RTI ; Return From Interrupt
TERM   ... ; Rutinas de otros dispositivos
```

# Ejemplo de transferencia de E/S usando las tres técnicas

## ■ E/S por DMA:



Petición de DMA: cuando se activa TRQ#, el DMAC pide el bus (BR'=1) y el 6800 se lo concede (BG=1). El DMAC lee el carácter de memoria (1er ciclo del bus) y lo escribe en BR (2º ciclo del bus)

## Ejemplo de transferencia de E/S usando las tres técnicas

- Escribir un 0 en SR7 inicia la transferencia de un bloque. Al completarse, el DMAC pone a 1 SR7 (bit “listo”).
- SR0 = 1 habilita las interrupciones.
- Cuando SR7 = 1 y SR0 = 1, el DMAC genera una interrupción para indicar al 6800 el fin de la transferencia.
- El tipo de transferencia se selecciona escribiendo en SR[2:1]. SR[2:1] = 10  $\Rightarrow$  transferencia de mem. a periférico.

## Ejemplo de transferencia de E/S usando las tres técnicas

; Inicialización del DMA		E/S por DMA
LDX	#\$3000	; SA <- 3000h (bloque en 3000h)
STX	DSA	; 3000h (bloque en 3000h)
LDX	#200	; BC <- 200 caracteres
STX	DBC	; DA <- dirección de BR
LDX	#PBR	; 00000101b (Memoria->periférico,
STAA	DSR	; IRQ permitida)
LDAA	#\$05	; PSR <- Petición del POC
STAA	PSR	; al DMAC permitida

# Ejemplo de transferencia de E/S usando las tres técnicas

## ■ Comparación de tiempos (de procesador):

- E/S por programa
  - Suponiendo una impresora de 200 caracteres/s, el tiempo de procesador será 1 s (**1 000 000 µs**).
- E/S por interrupciones
  - Suponiendo un período de reloj  $\theta = 1 \mu\text{s}$  ( $f = 1 \text{ MHz}$ ), una ejecución de la ISR requiere 59  $\mu\text{s}$ .
  - Además el hardware consume 9 ciclos de reloj cada vez que se atiende la interrupción.
  - En total: **13 600 µs** de tiempo de procesador.
- E/S por DMA
  - Se requieren dos ciclos del bus para transferir cada carácter.
  - Suponiendo un ciclo de reloj por ciclo del bus, el 6800 está inactivo 2  $\mu\text{s}$  por cada carácter.
  - Despreciando el tiempo para inicializar el DMA, **400 µs**.