Escuela Técnica Superior de Ingeniería Informática

Departamento de Arquitectura y Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

Arquitectura de Computadores

Tema 3: Sistema de Entrada/Salida

Grado en Ingeniería Informática

Contenidos

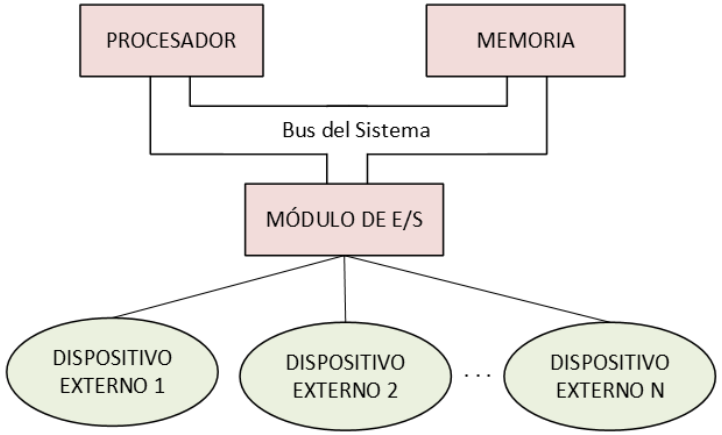
1. Introducción
2. Conexión del subsistema de E/S: común y separado
3. Métodos de Entrada/Salida
   A. E/S por *Polling* (Encuesta)
   B. E/S por interrupciones
   C. Acceso directo a memoria

Bibliografía

W. Stallings, “Organización y Arquitectura de Computadores”:
Capítulo 7 (Entrada/Salida)

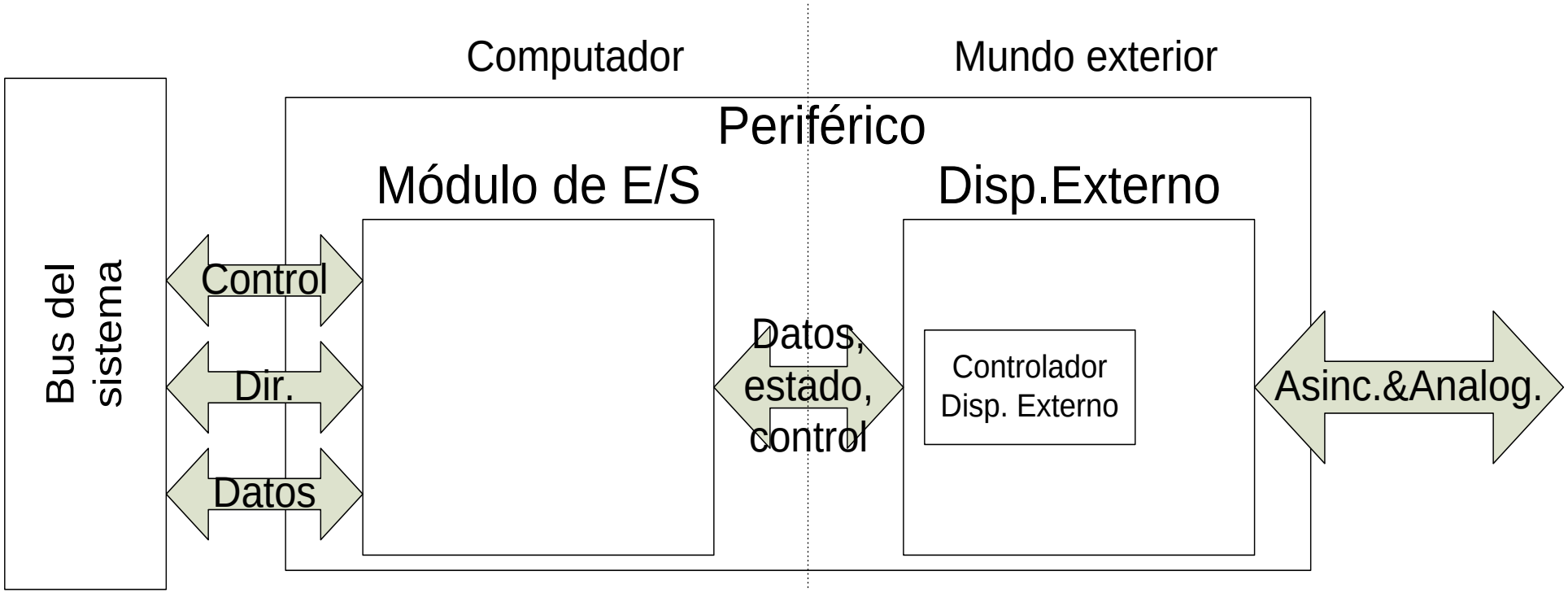
Sistema E/S: Introducción (I)

- Un computador precisa de un sistema de entrada/salida que permita la comunicación con el exterior. Se compone de:
  - **Dispositivos externos**: transfieren información entre el procesador y el exterior a modo de interface transformando la información analógica y asíncrona del exterior en síncrona y codificada del computador.
  - **Módulo de E/S**: permiten la conexión de los dispositivos externos al bus del sistema. Pueden ser:
    - **Controladores de E/S:** módulos muy sencillos (hardware mínimo para que funcione el dispositivo externo) en los que la lógica de control la gestiona parcialmente la CPU.
    - Los **canales o procesadores de E/S**: son auténticos procesadores con un juego especializado de instrucciones orientado a operaciones de E/S, que son programados por la CPU, permitiendo un funcionamiento autónomo.



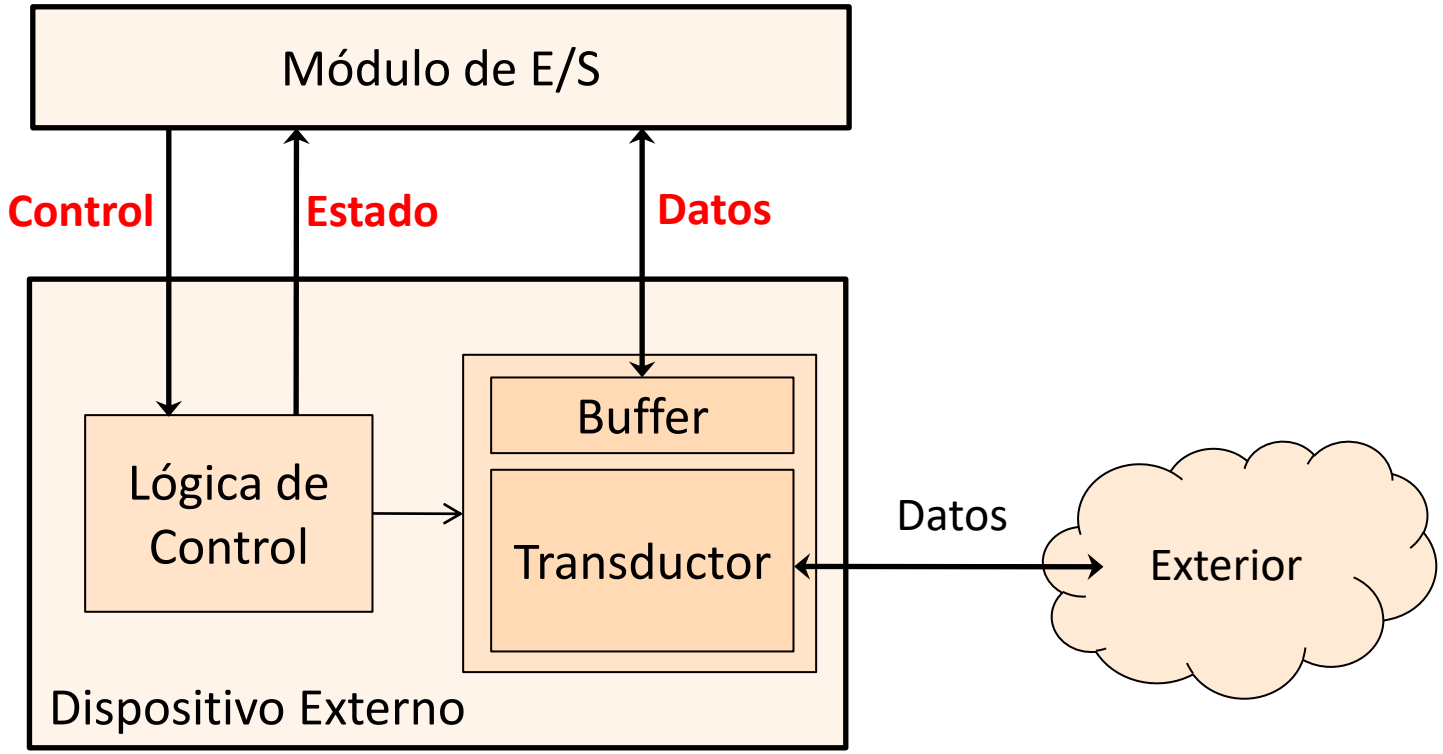
Dispositivo periférico

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



Dispositivo externo

- Diagrama de bloques:

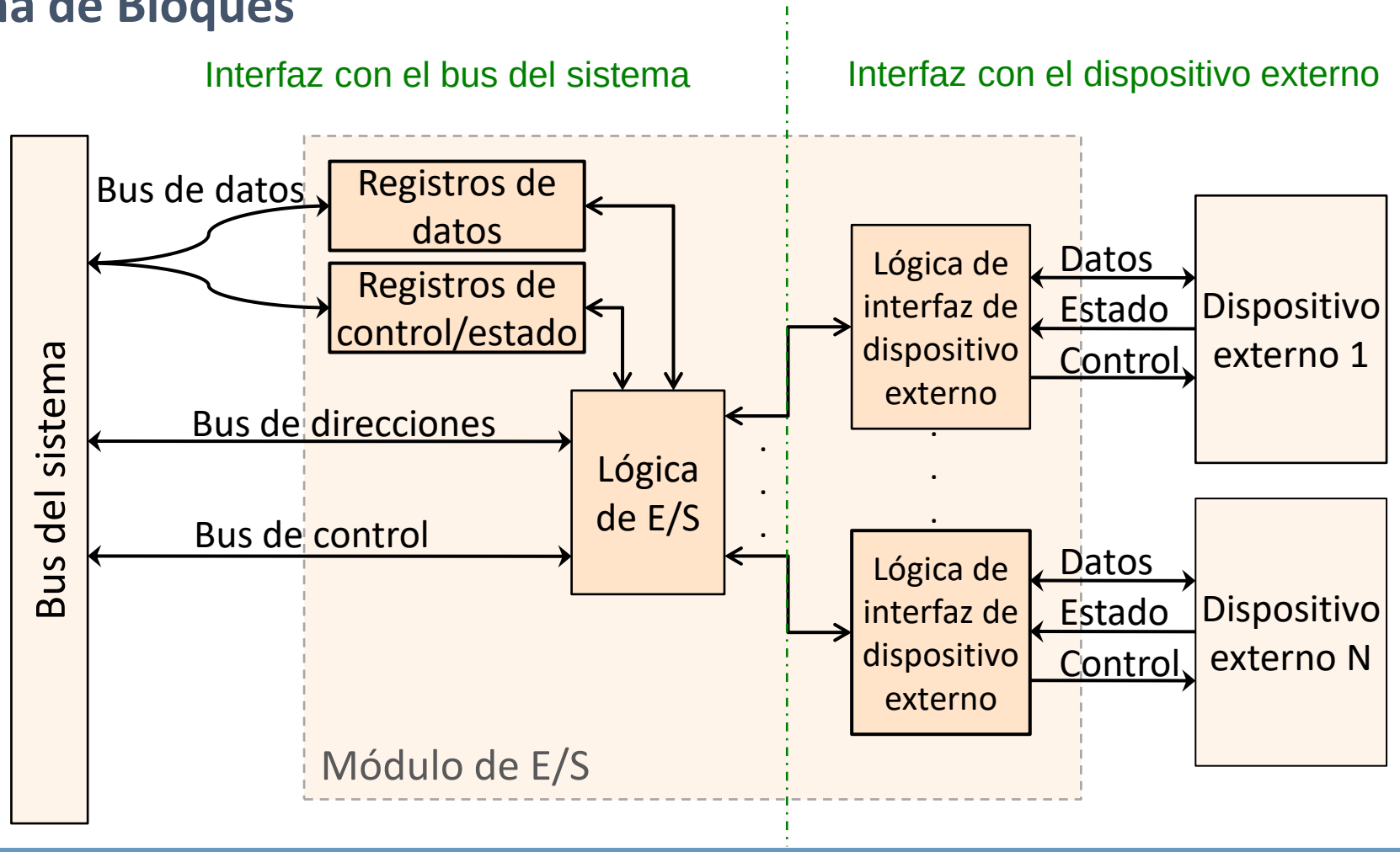


Sistema E/S: Introducción (IV)

Módulo de E/S:

- Su misión principal es la adaptación de los dispositivos externos para su conexión al bus del sistema.

- **Diagrama de Bloques**



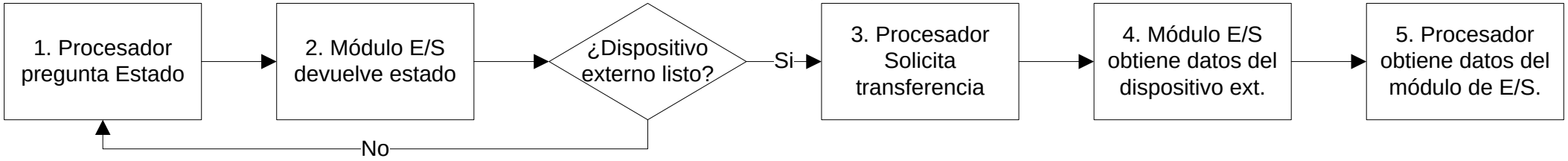
Módulo de E/S: Funciones (I)

- **¿Por qué no pueden conectarse directamente el dispositivo externo al procesador?**
  - **Velocidad:** La velocidad de transferencia en el bus del sistema suele ser distinta a la de los periféricos (adaptar la diferencia de velocidades). La velocidad del bus suele ser mucho mayor.
  - **Comportamiento:** Debido a la gran variedad de dispositivos periféricos existentes, no es posible incorporar toda la lógica necesaria para controlar tal diversidad de dispositivos dentro del procesador.
  - **Codificación:** Los formatos y tamaños de palabra de los datos en los periféricos suelen ser diferentes a los utilizados por el computador.
- **Funciones de un módulo de E/S**
  1. Control y temporización
  2. Comunicación con los dispositivos
  3. Comunicación con el procesador
  4. Almacenamiento temporal
  5. Detección de errores

Módulo de E/S: Funciones (II)

1. Control y temporización

- Necesarios para coordinar el tráfico entre los recursos internos (bus del sistema, memoria principal) y los dispositivos externos.
  - Por ejemplo, el control de la transferencia de datos desde un dispositivo externo al procesador podría implicar la siguiente secuencia de pasos:
    1. El procesador pregunta por el estado del dispositivo
    2. El módulo de E/S devuelve el estado del dispositivo
    3. Si el dispositivo está listo, el procesador solicita la lectura al módulo de E/S (mediante una orden)
    4. El módulo de E/S obtiene los datos
    5. Los datos se transfieren del módulo de E/S al procesador



2. Comunicación con los dispositivos:

- Órdenes
- Información de estado
- Datos

Módulo de E/S: Funciones (III)

3. La comunicación con el procesador:

- Decodificación de órdenes
- Transferencia de datos
- Información de estado
- Reconocimiento de dirección

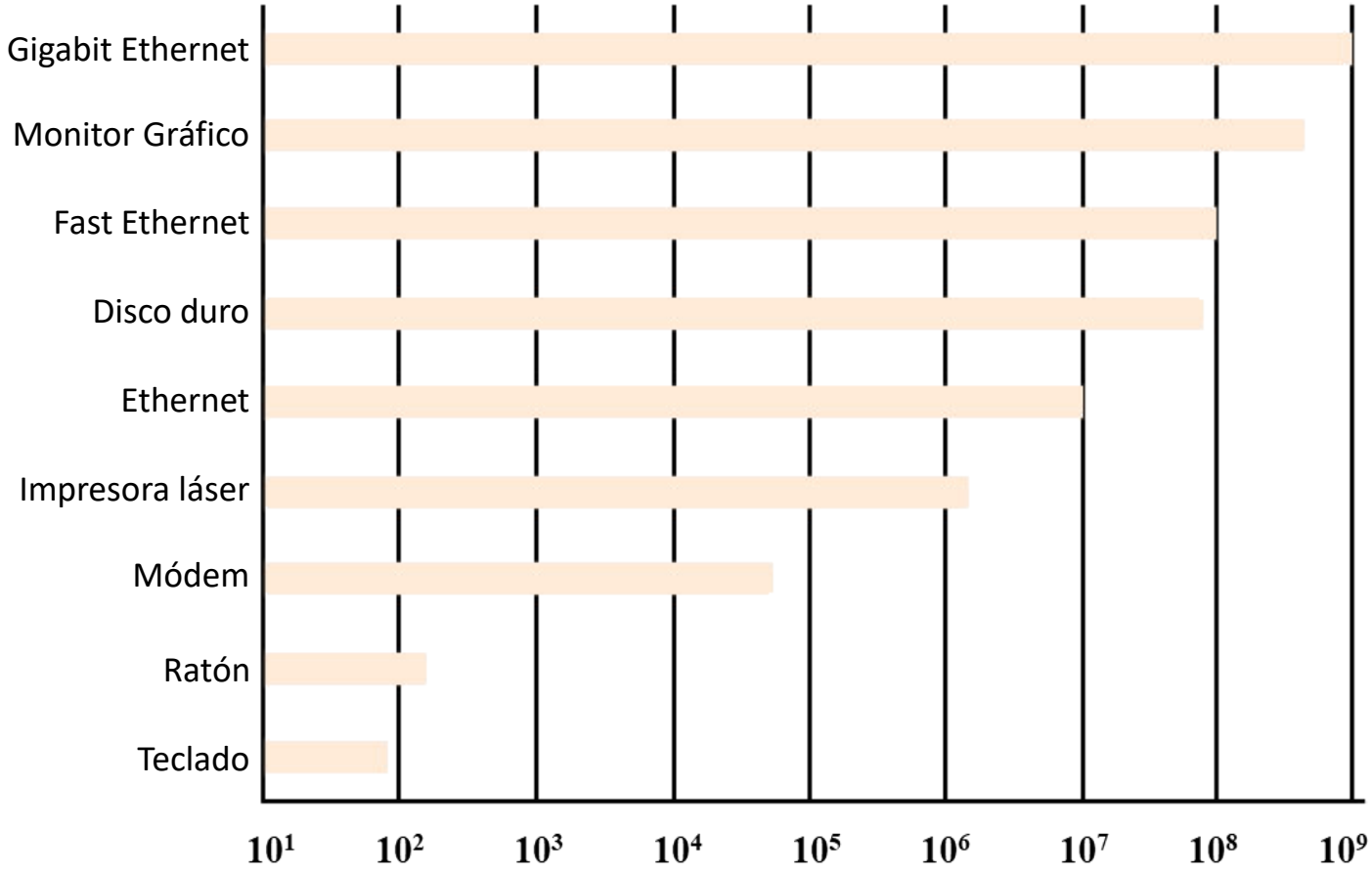
4. Almacenamiento temporal de datos:

- La velocidad de transferencia en el bus del sistema puede ser distinta a la velocidad de los periféricos por lo que es necesario amortiguar la diferencia de velocidades.
- Los datos se envían en ráfagas rápidas desde la memoria o el procesador al módulo de E/S y después se envían al periférico a la velocidad de éste (el proceso inverso es semejante)
- Los datos se almacenan para no mantener ocupada a la memoria en una operación de transferencia lenta (evitar una caída en el rendimiento)

5. Detección de errores:

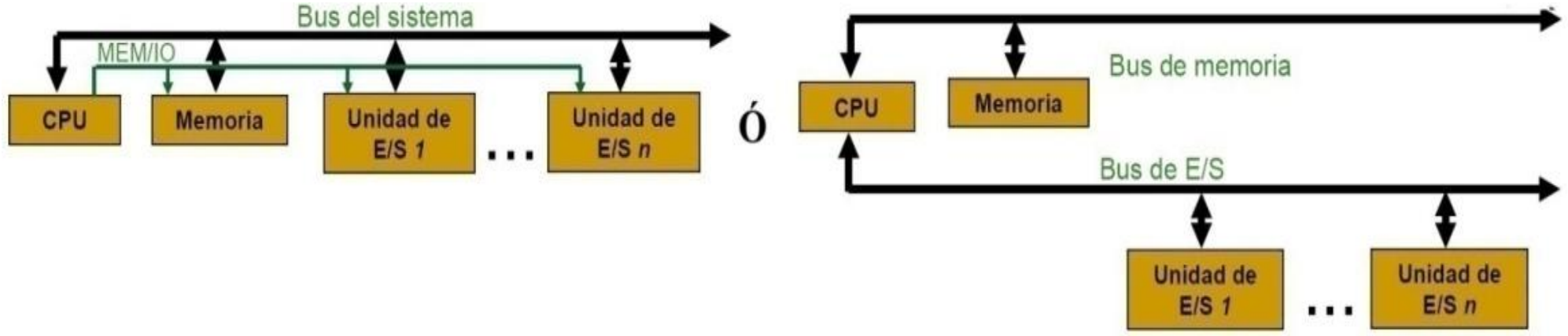
- Errores debidos a defectos mecánicos o eléctricos
- Errores en la transmisión de información (códigos de detección de errores)

Velocidades típicas de transferencia en dispositivos de E/S (bps)

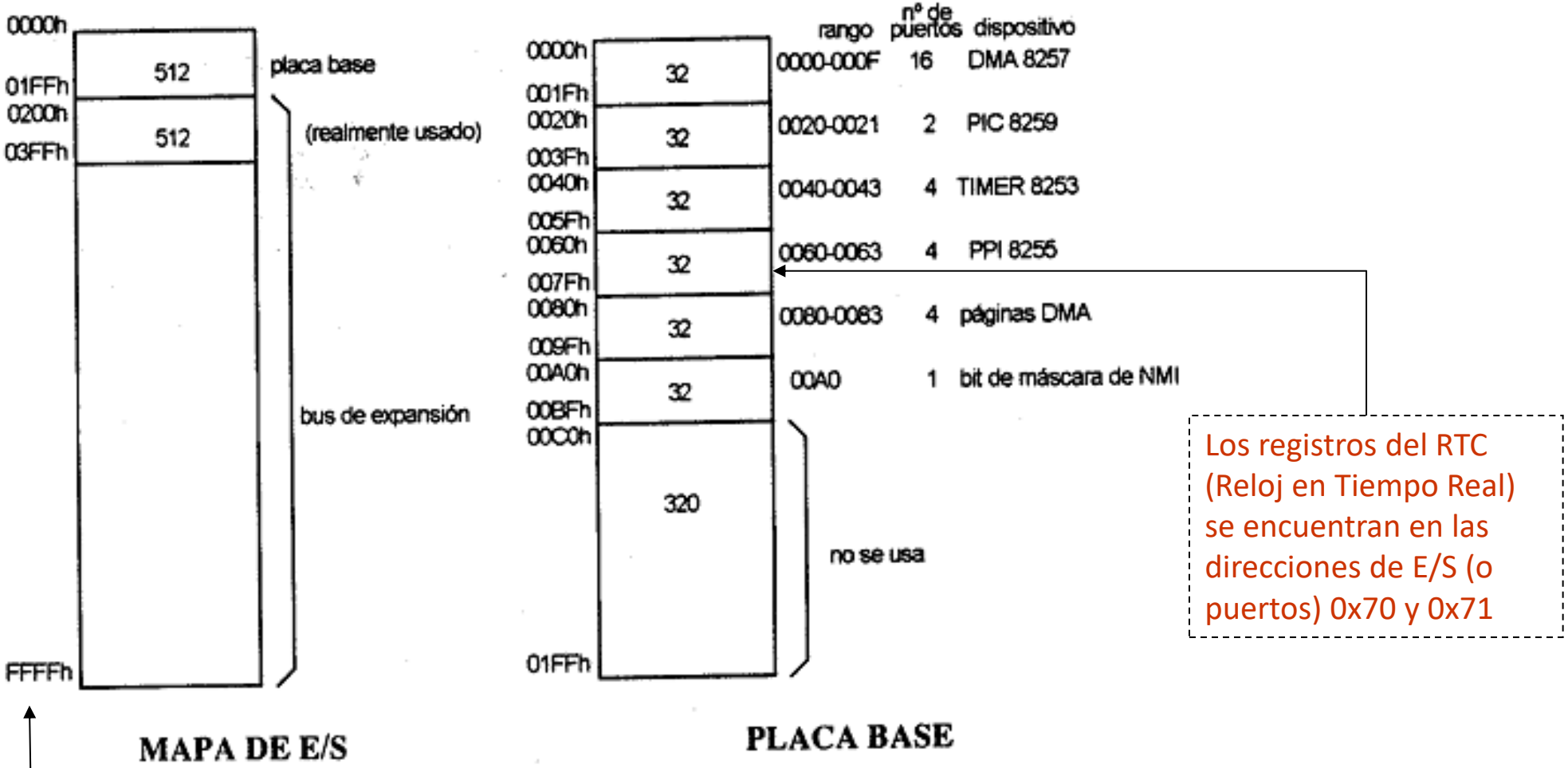


E/S Separada

- A la hora de integrar el subsistema de E/S (selección del dispositivos y acceso a sus registros) en un computador, existen dos posibilidades: E/S **separada** (o aislada) y E/S **mapeada en memoria** (o común)
  - E/S separada: el acceso a la E/S se realiza a través de un espacio de direcciones diferente al espacio de direcciones de memoria, al que llamaremos espacio de direcciones de E/S (puertos de E/S). Por lo tanto, en los sistemas que implementan la E/S separada existen dos espacios de direcciones independientes: uno para memoria y otro para E/S.
    - El acceso a la E/S está contemplado en la arquitectura: se utilizan señales e instrucciones específicas de E/S.
    - Ejemplo: la arquitectura IA32 dispone de un espacio de direcciones de E/S de 16 bits al que se accede mediante las instrucciones IN y OUT



E/S Separada – Ejemplo: mapa de direcciones de E/S en un PC

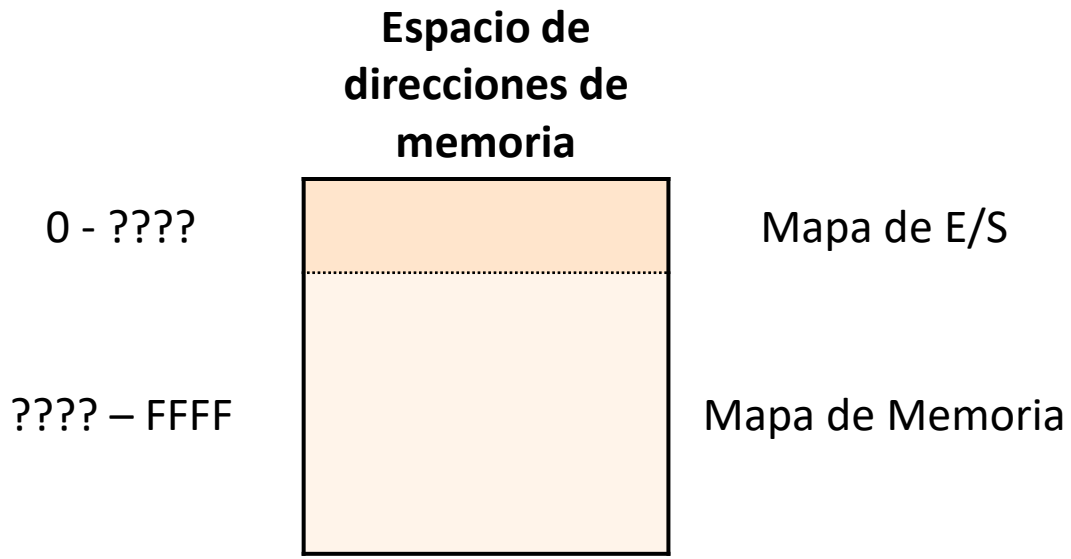


Los registros del RTC (Reloj en Tiempo Real) se encuentran en las direcciones de E/S (o puertos) 0x70 y 0x71

Bus de direcciones de 16 bits

E/S mapeada en memoria

- **E/S común o *mapeada* en memoria**: El acceso a la E/S se realiza a través del espacio de direcciones de memoria; es decir, el acceso a los periféricos se hace como si se accediese a un dato almacenado en la memoria principal.
  - A los periféricos se le asigna posiciones de memoria como si fueran variables.
  - El procesador no contempla el acceso a módulos de E/S.
  - Ejemplo: procesador RISCV, MIPS, Motorola 68000.



Programación básica

- **Ejemplo de E/S separada y mapeada:**
  - **Dispositivo con tres registros de 8 bits**:
    - **Control** (DirBase+0): Operación de lectura → Escribir un 5.
    - **Estado** (DirBase+1): Lectura realizada → Si bit más significativo a 1.
    - **Datos** (DirBase+2): Contiene el último dato leído del dispositivo.
  - **Comportamiento para realizar una lectura:**

    1º - Enviar petición de lectura.

    2º - Verificar si se ha realizado la lectura: Si fuese el caso, obtener el dato leído.
- Ejemplo en C++ : Dirección Base 1000

E/S Mapeada	E/S Separada (Linux)	
char res = 0; *char* v;* *v = (char*)1000;* ***v = 5;*** *v++;* *if(((**v) & 0x80)!=0)* *res = **(++v);*	*char res = 0;* ***outb(5, 1000);*** *if ((**inb(1001)**&0x80)!=0)* *res = **inb(1002);*	**Sintaxis en Linux:** *dato = inb (puerto)* *outb(dato8bits, puerto)* **Sintaxis en Windows:** *dato = inportb (puerto)* *outportb(puerto, dato8bits)*

Programación básica

- Ejercicio anterior suponiendo:
  - En RISCV: mapeados a partir de la dirección base 0xFFFFA000
  - En 8086: mapeados a partir de la dirección base 0x300
- Ensamblador

E/S Mapeada (RISCV)	E/S Separada (8086)	
addi x3,x0,5 *lui x2, 0xFFFFA* *ori x2,x2,0x000* ***sb x3,0(x2)*** ***lbu x4,1(x2)*** *andi x4,x4,0x80* *beq x4,x0,fin* ***lb x5,2(x2)*** *fin:*	*mov al, 5* *mov dx, 300h* ***out dx, al*** *inc dx* ***in al, dx*** *and al,80h* *jz fin* *inc dx* ***in al, dx*** *mov bl, al* *fin:*	Sintaxis de in y out del 80x86: ***in dato, puerto*** ***out puerto, dato*** ***dato*:** · Sólo registro acumulador (AL, AX EAX) · Tamaño del registro según el del puerto ***puerto*:** si es de 8 bits → inmediato si es de 16 bits → Reg. DX

E/S mapeada en memoria: Ventajas

- Simplifica el juego de instrucciones del procesador, ya que no requiere instrucciones de propósito específico.
- Accesos más eficientes a los dispositivos
- Programación más flexible y eficiente:
  - Aprovecha toda la potencia del juego de instrucciones, facilitando la programación ya que el juego de instrucciones específicos de E/S suele ser reducido.
  - El compilador tiene mayor libertad para elegir el lugar y el modo de direccionamiento en el acceso a memoria

    Ejemplo: en IA32, las instrucciones IN y OUT sólo pueden usar los registros EAX, AX o AL para almacenar el dato leído o escrito del puerto de E/S y el registro DX para indicar el puerto.

E/S mapeada en memoria: Inconvenientes

- **Desperdicia** parte del espacio de direcciones de memoria.
- Las optimizaciones que realizan **reordenación de instrucciones** son problemáticas puesto pueden alterar el orden en el que se accede a los registros del dispositivo.
- Los **efectos colaterales** de los accesos a la E/S mapeada pueden afectar negativamente a la Memoria Caché.
  - Una forma de solucionar estos efectos en caché, consiste en configurar el rango de direcciones de memoria utilizadas por el dispositivo como **no *cacheable*** evitando así el uso de la memoria caché para este rango de direcciones.

E/S mapeada en memoria: Inconvenientes

- Problema de la E/S mapeada por los efectos colaterales.
  - Código RISCV del ejercicio anterior:

```
addi x3,x0,5
lui x2, 0xFFFFA
ori x2,x2,0x000
sb x3,0(x2)
lbu x4,1(x2)
andi x4,x4,0x80
beq x4,x0,fin
lb x5,2(x2)
fin:
```

Si la caché es CB-WA la escritura en 0xFFFFA000 no llega al registro de control

Si las direcciones 0xFFFFA0000-0xFFFFA0002 pertenecen al mismo bloque de memoria, siempre se leerán los valores que había memoria en el momento del primer acceso al bloque.

Sistema E/S: Métodos de E/S

- Para realizar una operación de E/S es necesario:
  - Algún mecanismo encargado de **sincronizar la transferencia**, debido a:
    - La diferencia de velocidades entre procesador y periféricos.
    - Los eventos de E/S se producen de forma asíncrona (llegada de datos del dispositivo, dato procesado por el dispositivo, etc.)
  - Realizar la transferencia de los datos.
- Los métodos de E/S suelen clasificarse atendiendo a si la CPU es la responsable de sincronizar las transferencias y/o de realizarlas.

	CPU **sincroniza** las transferencias	CPU **NO sincroniza** las transferencias
CPU **realiza** la transferencia	***Polling (Encuesta)***	***Interrupciones***
CPU **NO realiza** la transferencia	***X***	***Acceso directo a memoria***

E/S por *Polling* (Encuesta)

- La CPU es el responsable único de realizar la operación.
- Ejecutará un programa que verifique cuando el dispositivo está listo para realizar la transferencia (sincronización).
  - **Sondear** (interrogar) el estado del dispositivo leyendo sus registros de estado → **Esperas activas**
- Accederá a los registros de datos para transferir los datos.

```
mientras (no disponible) → leer estado
realizar transferencia.
```

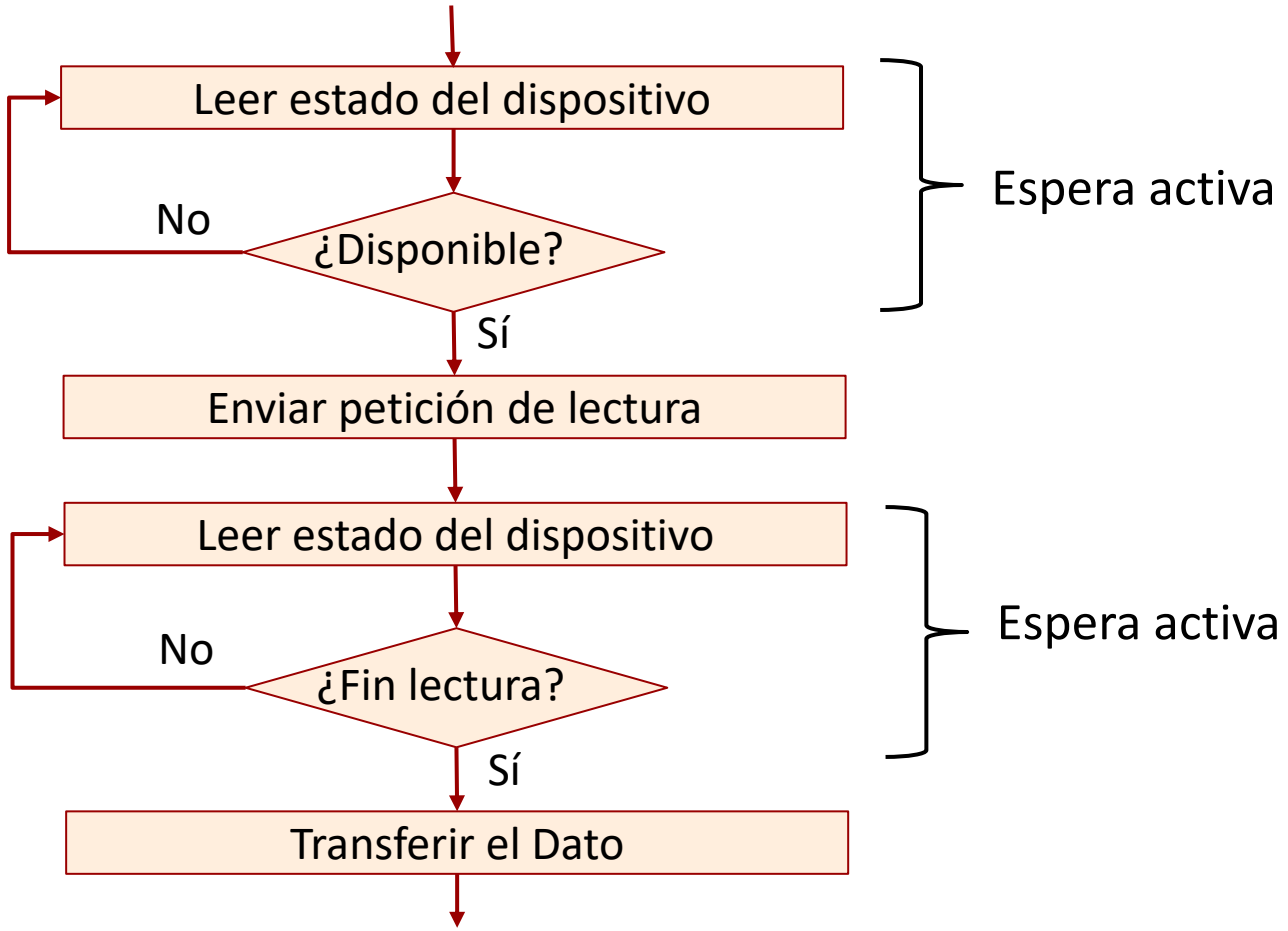
- **Ventajas:**
  - Necesita un hardware de E/S mínimo.
  - Las operaciones se realizan muy rápidamente.
- **Inconvenientes:**
  - Las esperas activas consumen un elevado tiempo de proceso.
  - Las esperas activas son problemáticas en sistemas multiproceso.

E/S por *Polling*: Ejemplo

Realizar una operación de lectura en un dispositivo suponiendo que:

· Antes de enviar una petición de lectura, el dispositivo debe estar 'disponible'.

· De debe esperar a que 'finalice' la lectura antes de transferir el dato al computador.



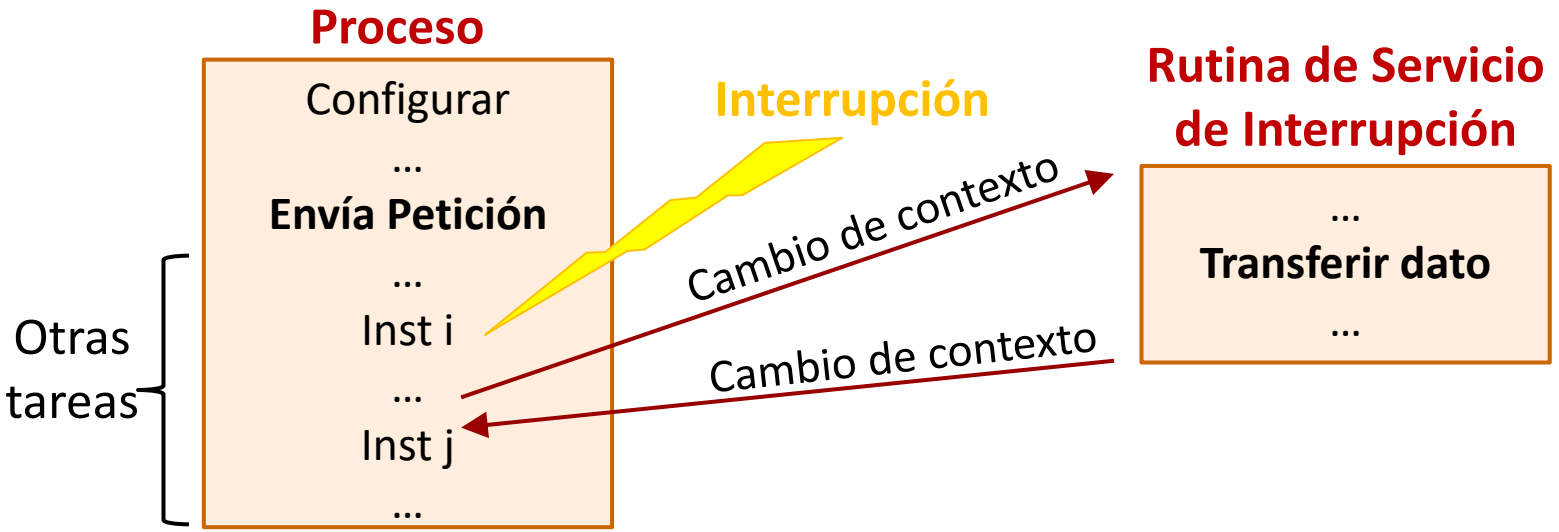
E/S por interrupciones

- La CPU ya no sincroniza la transferencia, sino que el dispositivo avisa a la CPU cuando está preparado.
  - Aunque la CPU sigue realizando la transferencia de datos.
- La CPU dispone de una(s) línea(s) de interrupción, que son activadas por los módulos de E/S cuando el dispositivo notifica que está preparado.
  - El dispositivo debe ser capaz de utilizar interrupciones.

- La CPU realizará la transferencia cuando activen la interrupción.
  - Habrá algún mecanismo para identificar qué dispositivo ha enviado la interrupción y de prioridades.
- **Ventaja**: Mejora el rendimiento de la CPU al eliminar esperas activas.
- **Inconveniente**: es más lenta debido a los cambios de contexto.

Funcionamiento básico

1. La CPU envía una petición y continúa con otra tarea.
2. Cuando el dispositivo finaliza la operación (o cambia su estado), el módulo de E/S activa la línea de interrupción.
3. Tras terminar la instrucción en curso, la CPU envía una señal de reconocimiento al módulo de E/S.
4. La CPU ejecuta la rutina que trata la interrupción (**rutina de servicio de interrupción**) responsable de la transferencia.
5. Al finalizar la rutina, continúa ejecutando su tarea.

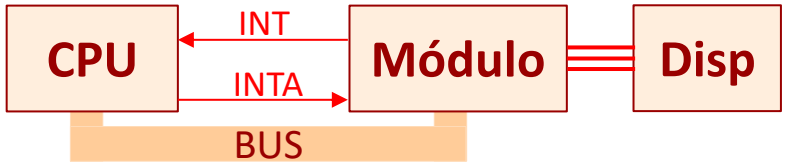


Funcionamiento detallado

1 - Configurar el dispositivo
2 - Configurar el computador
3 - Enviar la petición
4 - El módulo E/S ordena la operación
5 - *Ejecutar otras Instrucciones*
6 - El dispositivo finaliza la petición
7 - El módulo E/S activa línea de interrupción
8 - *Terminar la ejecución en curso*
9 - Gestionar prioridades si existen varias
10 - Reconocer la interrupción
11 - El módulo de E/S desactiva la línea INT
12 - Identificar el dispositivo
13 - Determinar la Rut. Servicio Interrupción
14 - Salvar el estado actual (PSW, PC…)
15 – Cargar en PC la dirección de la RSI

26 - *Continuar la ejecución de Instrucciones*

INT



Rutina de Servicio de Interrupción

16 - *[Deshabilitar interrupciones]*
17 - Guardar los registros seguros
18 - *[Leer registro de estado]*
19 - Procesar la Interrupción *(Transferir)*
20 - *[Enviar nueva petición]*
21 – *[El módulo ordena nueva petición]*
22 - Restaurar los registros seguros
23 - *[Habilitar interrupciones]*
24 - Restaurar el estado anterior (PSW)
25 - Restaurar el PC anterior

Tipos de interrupciones

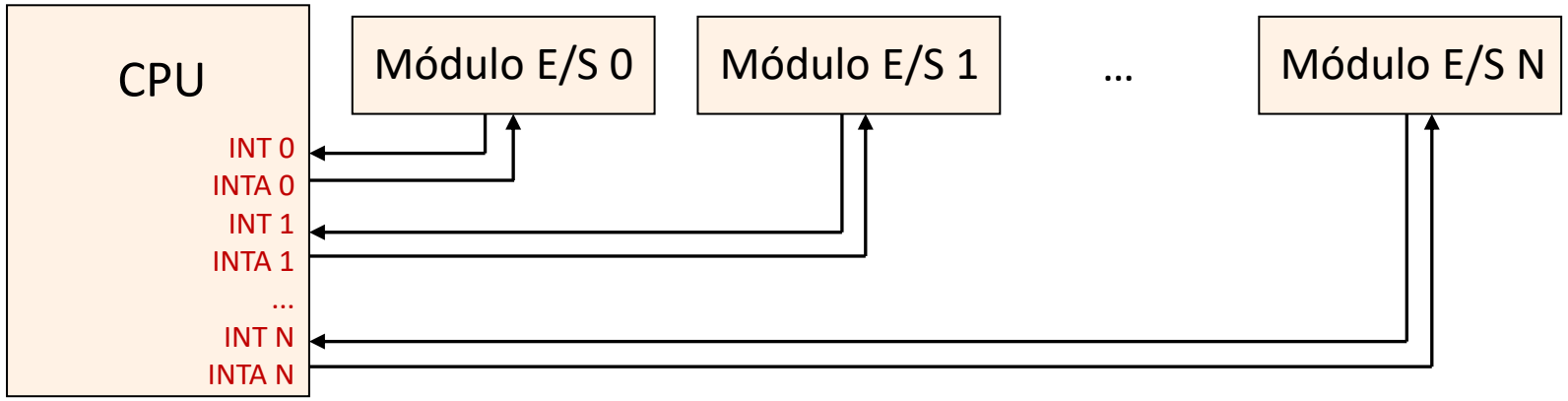
- **Interrupciones Software:**
  - Producidas intencionadamente mediante instrucciones específicas.
- **Interrupciones Hardware:**
  - **Internas (excepciones):** producidas por la CPU.
  - **Externas:** producidas por los dispositivos de E/S.
    - **Vectorizadas:** Identifican la interrupción mediante un **vector**.
      - El vector es utilizado para obtener la rutina de servicio de la interrupción.
    - **No vectorizadas (**autovectorizadas): El procesador identifica el dispositivo por sí mismo.
    - **No enmascarables (NMI):** siempre son atendidas.
      - Suelen ser prioritarias a las no enmascarables.
    - **Enmascarables:** pueden ser desactivadas.

Implementación

- En la implementación de la E/S mediante interrupciones aparecen dos cuestiones:
  - **Identificación del dispositivo:**
    - ¿Cómo determina el procesador qué dispositivo ha provocado la interrupción?
  - **Esquema de prioridades:**
    - Si se producen varias interrupciones simultáneas, ¿cómo decide el procesador la que debe atender?
- Hay cuatro técnicas utilizadas comúnmente para implementar la E/S por interrupciones:
  - **Múltiples líneas de interrupción**
  - **Consulta software** (*software polling*)
  - **Conexión en cadena** (*daisy chain*)
  - **Arbitraje de bus**

Implementación: Múltiples líneas de interrupción

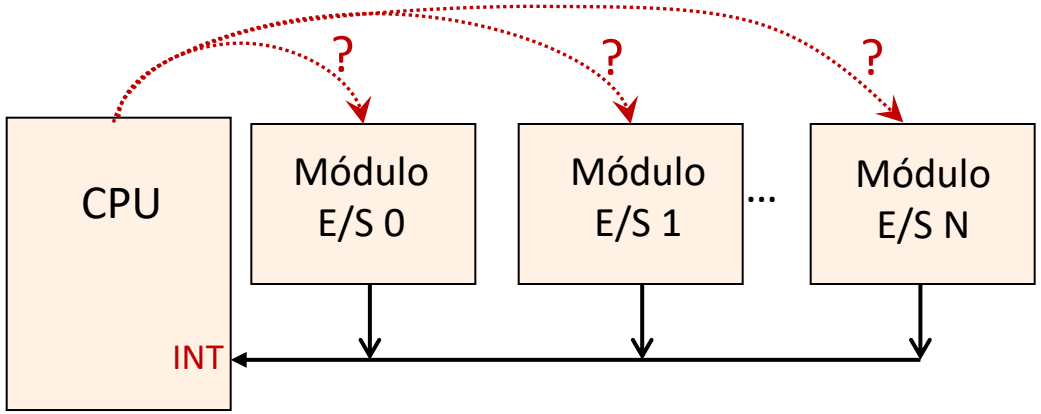
- Consiste en proporcionar varías líneas de interrupción entre el procesador y los módulos de E/S.
- La prioridad viene fijada por el procesador.
- El dispositivo se identifica con la línea activa.

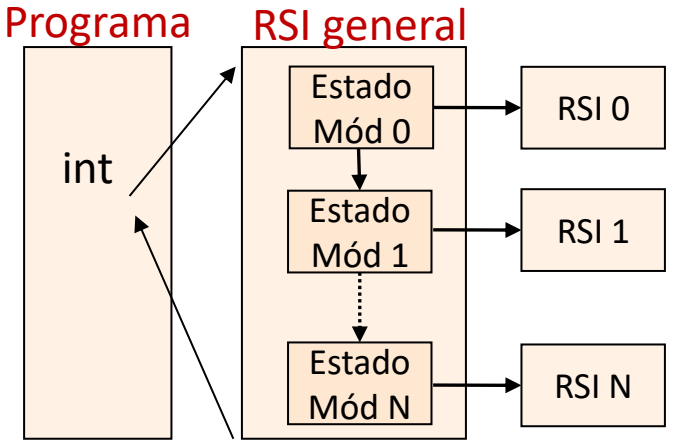


- **Inconvenientes**:
  - Malgasta terminales del procesador
  - De existir más periféricos que líneas no resolvería el problema.

Implementación: Consulta *software*

- Comparten una línea común para solicitar interrupciones.
- Cuando el procesador detecta una interrupción, se produce un salto a una rutina de tratamiento de interrupción que consulta el registro de estado de cada módulo de E/S para determinar cuál ha producido la interrupción.
- **Prioridad**: determinada por el orden en que se hace la consulta

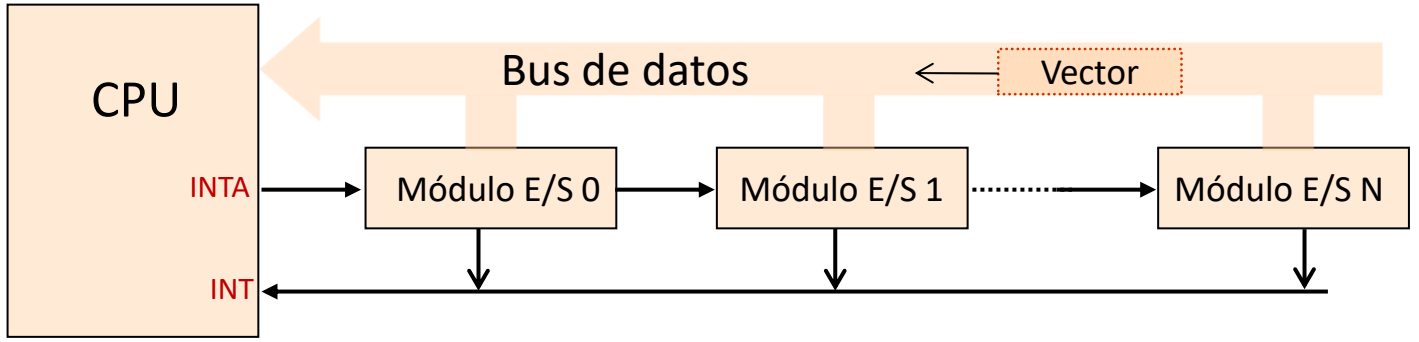




- **Inconveniente**:
  - Consume mucho tiempo consultando en busca del módulo.

Implementación: Conexión en cadena (consulta *hardware*)

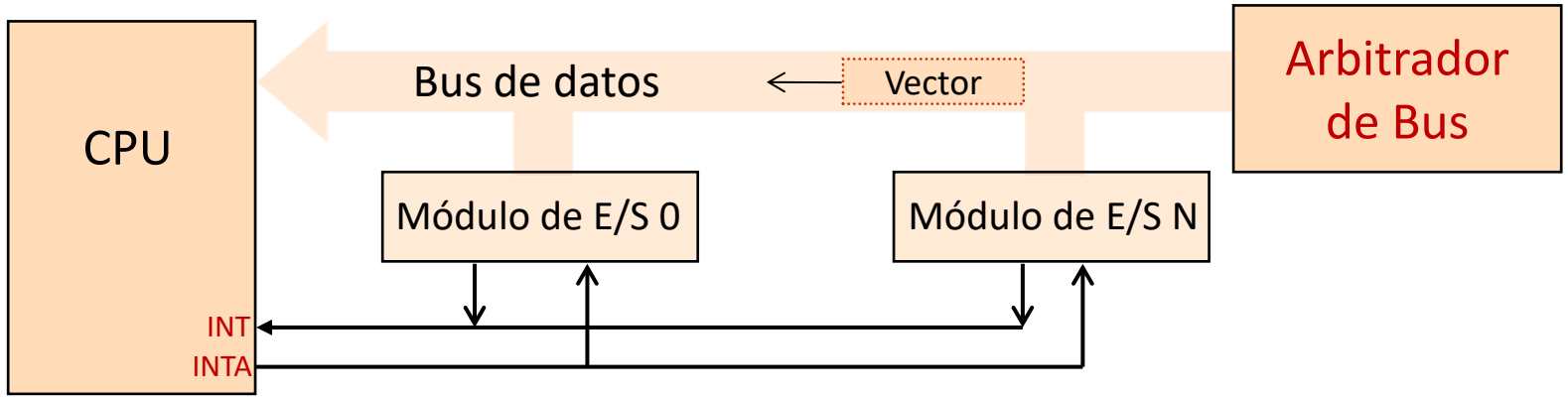
- La línea de reconocimiento de interrupción se conecta encadenando los módulos uno tras otro
- La señal de reconocimiento se propaga por la cadena hasta alcanzar un módulo de E/S que hubo solicitado la interrupción.
- El módulo E/S correspondiente responde colocando un número que lo identifica (vector de interrupción) en las líneas de datos.
- El procesador ejecuta la rutina de interrupción según el vector.



- **Prioridad:** según el orden en que se conectan los módulos.
- **Ventaja :** Evita ejecutar una rutina de servicio general.

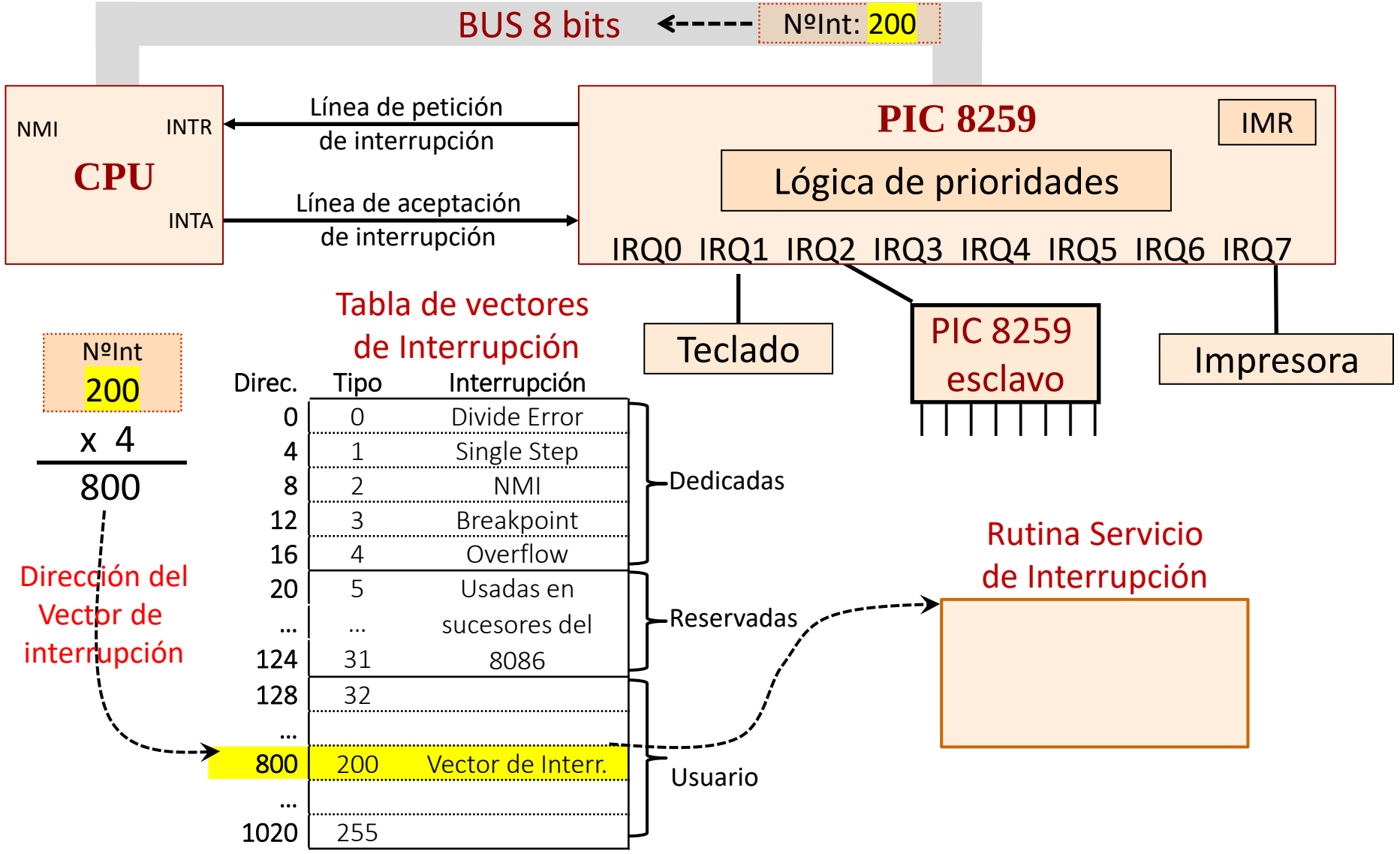
Implementación: Arbitraje de bus

- Un módulo de E/S debe disponer del **acceso del bus** antes de poder activar la línea de petición de interrupción.
- Cuando la CPU acepta la interrupción el dispositivo sitúa el **vector** de interrupción en el bus de datos antes de soltarlo.
- Mediante el **arbitrador** de bus se garantiza que sólo un módulo puede activar la señal de petición en un determinado instante.



- **Prioridad:** viene determinada por el arbitrador.

Ejemplo: Intel 8086



Acceso directo a memoria

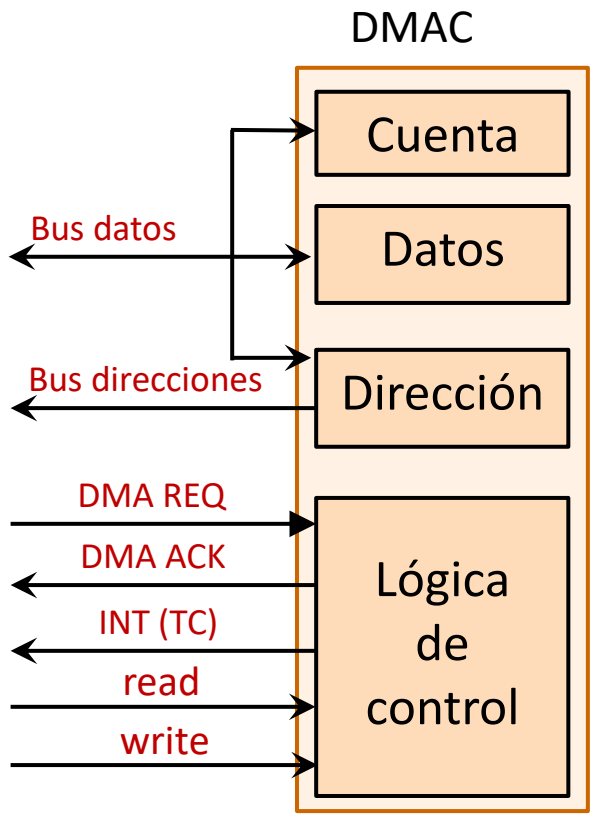
- La E/S por *Polling* y por interrupciones requieren la intervención activa del procesador para transferir datos entre la memoria y los módulos de E/S.
  - Impacto negativo sobre **la actividad del procesador** y la **velocidad de E/S**.
- Utilizando la **E/S por *Polling***, el procesador puede transferir datos a alta velocidad al precio de no hacer nada más
- La **E/S por interrupciones** libera en parte al procesador a expensas de reducir la velocidad de E/S (debido a la sobrecarga que supone el cambio de contexto)
- Cuando hay que transferir grandes volúmenes de datos:

Acceso Directo a Memoria - *Direct Memory Access* (DMA)

La CPU no se encarga ni del inicio ni del control de la transferencia (se libera a la CPU)

Controlador DMAC

- Los módulos de E/S que no tienen capacidad para realizar acceso directo a memoria requieren un **Controlador de DMA (DMAC)** en el bus del sistema.
  - El bus PCI no precisa DMAC, pues ya posee los mecanismos necesarios para que los módulos puedan acceder directamente a memoria.
- **La CPU debe configurar el DMAC, especificando:**
  - Dirección del módulo de E/S
  - Dirección inicial de memoria
  - El tamaño de la transferencia
  - El tipo transferencia: lectura o escritura
- **El DMAC transfiere el bloque palabra a palabra.**
  - Se sincroniza con el dispositivo a través de *DMA Req* y *DMA Ack* en cada palabra.
  - Realiza el acceso a la memoria con su registro de datos, de dirección y las señales de control.
  - Actualiza el registro de dirección y decrementa el registro cuenta de datos.
  - Repite mientras registro “Cuenta de datos” != 0
  - Cuando finaliza, envía una interrupción al procesador (TC, *Terminal Count*)



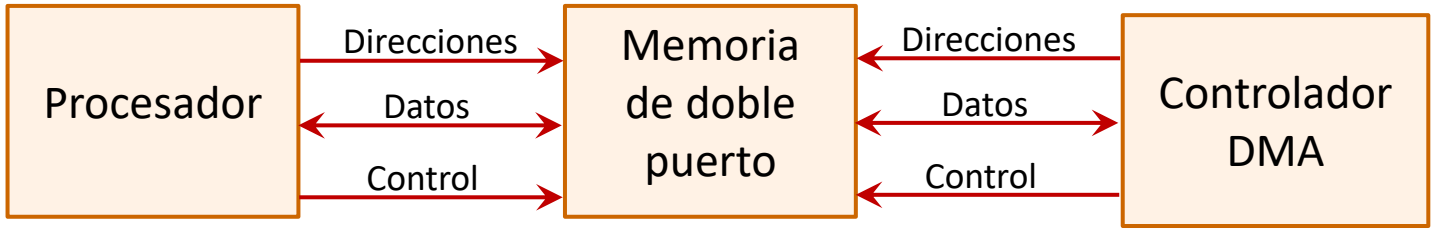
Funcionamiento básico

1. **Inicialización de transferencia**: la CPU envía los parámetros de la transferencia:
   - **Configurar Controlador de DMA**: nº bytes/palabras, tipo de transferencia (R/W), dirección de memoria inicial para transferencia; nº de canal, si DMA tiene más de un canal, etc.
   - **Configurar Dispositivo de E/S**: número de bytes a transferir, tipo de transferencia (R/W), etc.
2. **Transferencia**:
   - **La CPU**: continúa la ejecución de otras tareas, por lo que tendrá que utilizar el bus durante algunos ciclos.
   - **El controlador de DMA**: realiza la transferencia del bloque controlando el acceso a la memoria y accediendo al dispositivo mediante las señales de control… pero:
     - ¿Cómo accede a memoria y al dispositivo exactamente?
     - ¿Accede al bus del sistema? ¿Cómo afecta al rendimiento?
3. **Finalización de la transferencia:**
   - El controlador notifica el resultado de la operación con una interrupción de TC

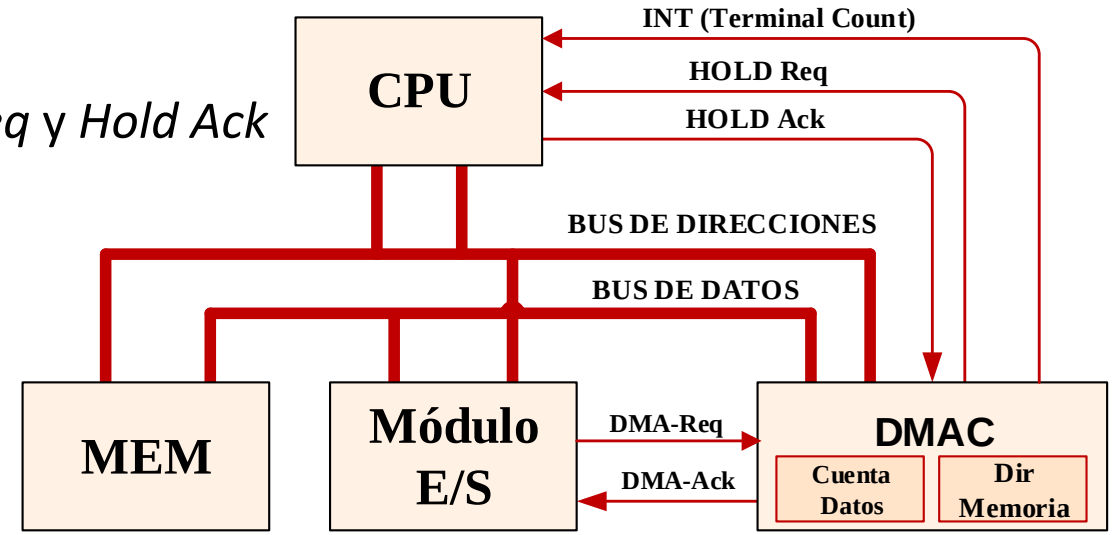
Funcionamiento

¿Cómo accede a Memoria?

- **Memoria multipuerto:** Memoria de dos puertos para CPU y DMAC.
  - Permite realizar transferencias simultáneas de la CPU y el DMAC.
    - Exige mecanismos de control en el acceso concurrente a un mismo bloque



- **Acceso por compartición del bus:**
  - El DMAC pide el control del bus por las líneas *Hold Req* y *Hold Ack*
  - La más utilizada habitualmente.
  - Solución más económica.
  - Tres modos de compartición
    - Modo ráfaga.
    - Modo por robo de ciclos
    - Modo transparente.

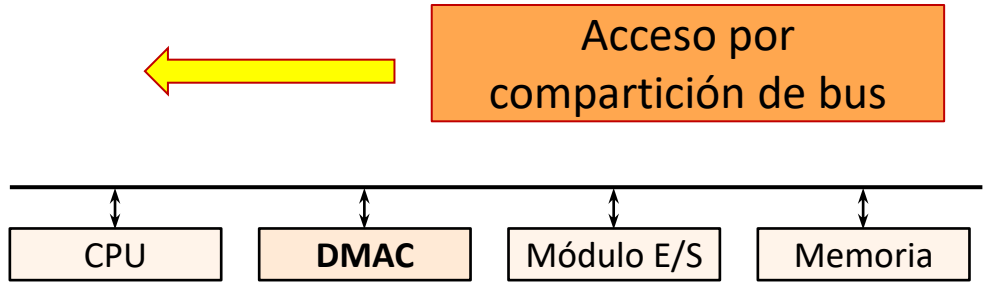


Sistema E/S: Métodos – DMA (V)

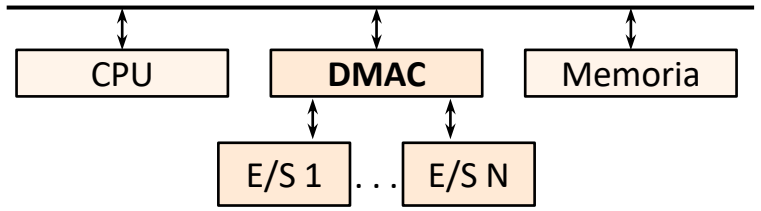
Funcionamiento

¿ Cómo accede a los módulos de E/S?

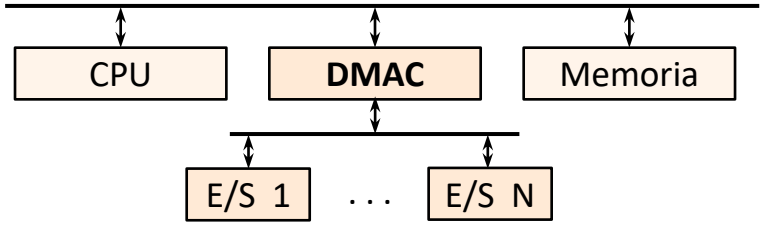
- **Bus único, DMA independiente:**
  - Requiere 2 accesos al bus del sistema:
    1 ciclo para acceder al módulo de E/S y otro a memoria (si no usa Fly-by)
  - Es económica.



- **Bus único, DMA-E/S:**
  - La comunicación entre DMAC y E/S no requiere acceder al bus del sistema.
  - Reduce número de ciclos de bus integrando las funciones de DMA y E/S.

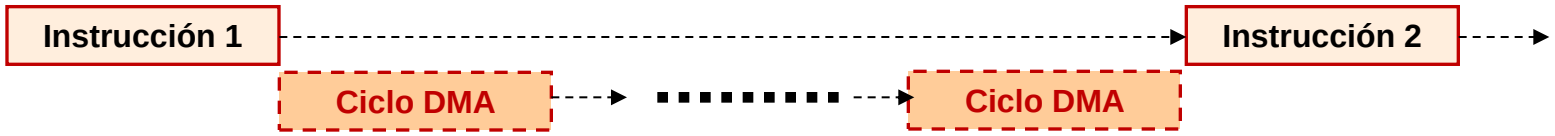


- **Bus de E/S:**
  - Conexión del DMAC a los módulos de E/S mediante un bus de E/S.
  - Se reduce a uno el número de interfaces de E/S y es una configuración fácilmente ampliable.

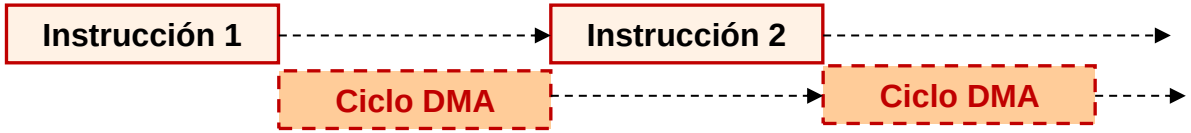


Funcionamiento: Modos de transferencia DMA en acceso por compartición de bus

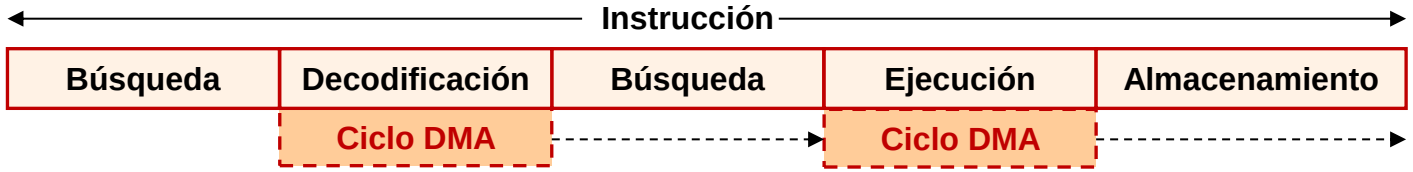
- **Modo Ráfaga:**
  - El DMAC toma control y no lo libera hasta transmitir el bloque de completo.
  - Mayor velocidad de transferencia pero puede dejar sin actividad a la CPU durante periodos grandes de tiempo.



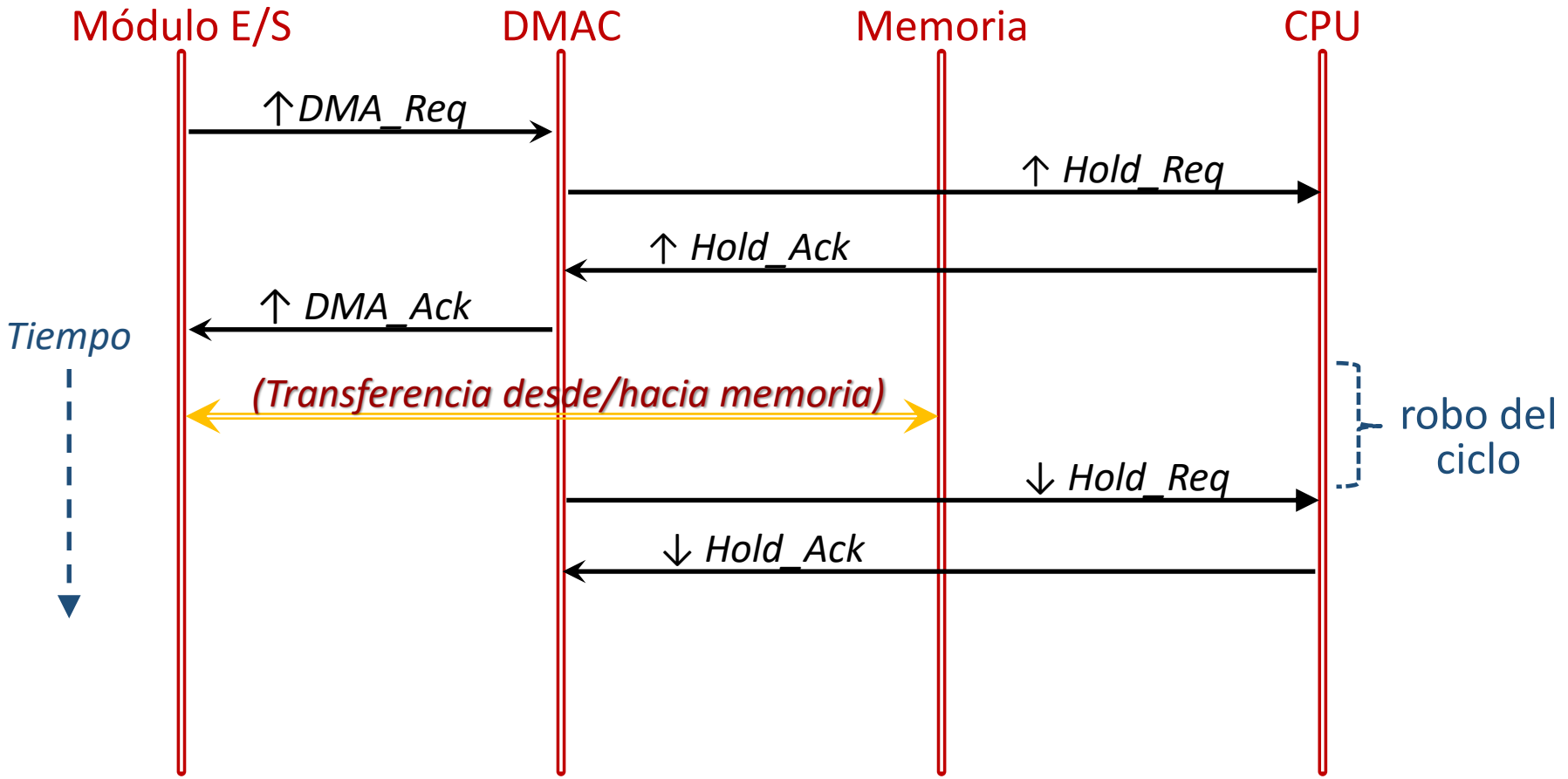
- **Modo Por robo de ciclo (*más usual*):**
  - El DMA toma control durante un solo ciclo. (Transmite una palabra y lo libera).



- **Modo Transparente:**
  - El DMA accede al bus solo en los ciclos en los que la CPU no lo utiliza. (En diferentes fases de ejecución de las instrucciones).
  - La ejecución del programa no se ve afectada en su velocidad de ejecución.

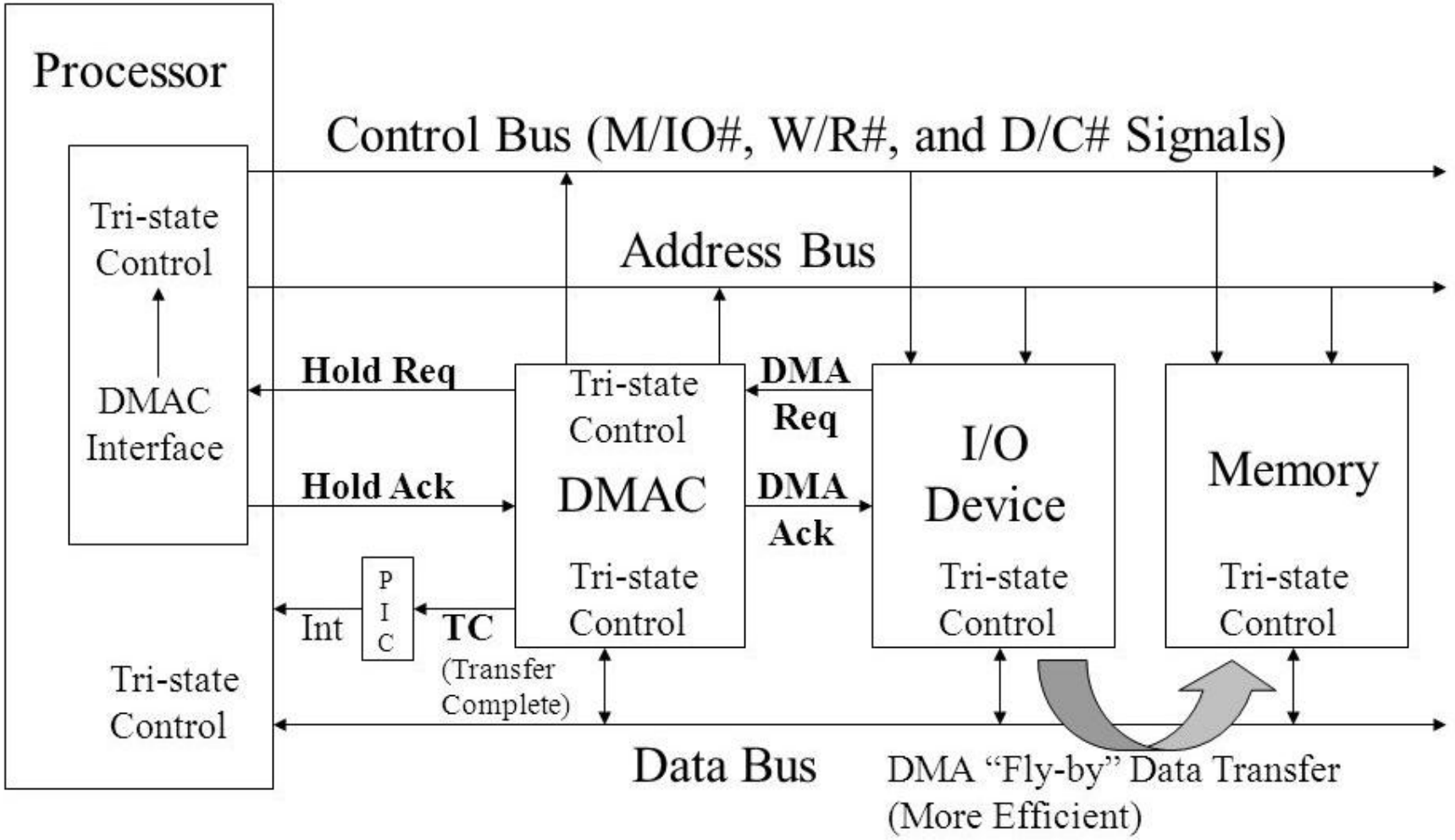


Funcionamiento de compartición de bus por robo de ciclo

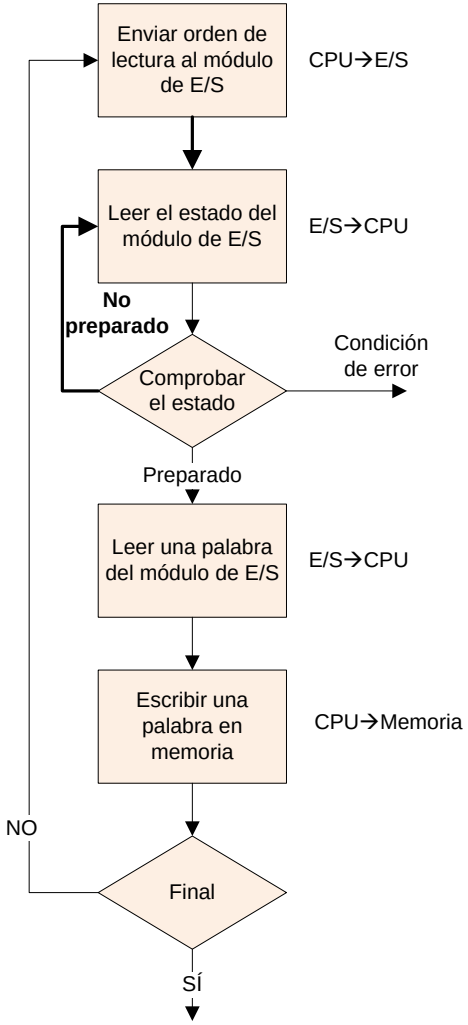


Transferencia DMA "Fly-by"

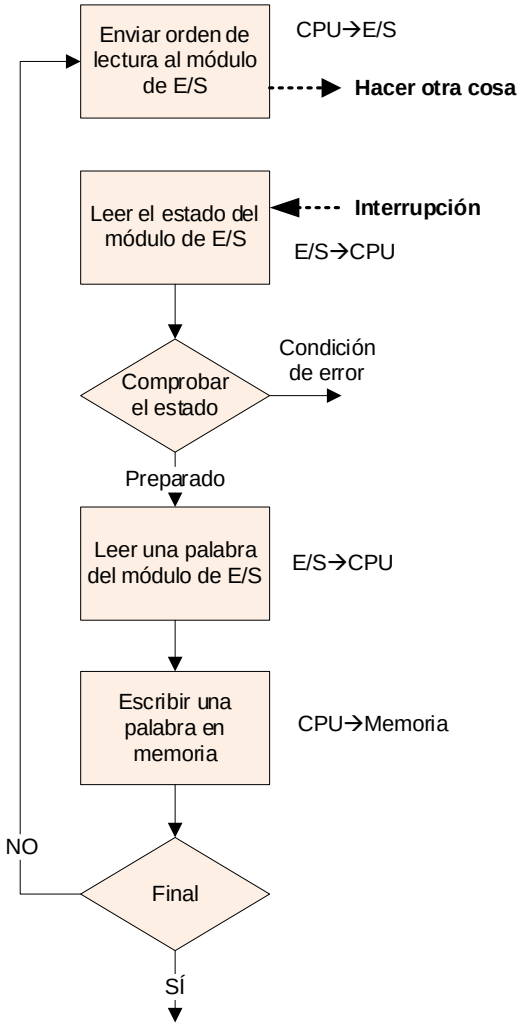
Funcionamiento: Esquema completo por compartición de bus



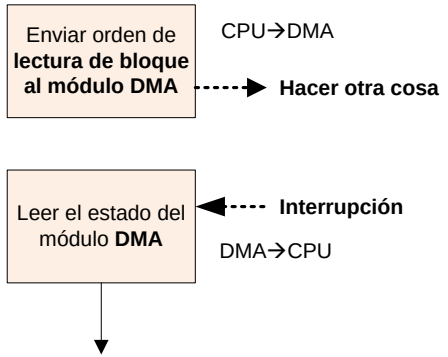
Sistema E/S: Métodos – Resumen (I)



a) E/S por *Polling*



b) E/S por Interrupciones



c) Acceso Directo a Memoria

Sistema E/S: Métodos – Resumen (II)

	CPU para Sincronizar	CPU para Transferir	Velocidad E/S
E/S por Sondeo	Alto (Esp. Activa)	Alto	Alta
E/S Interrupciones	Bajo (No espera)	Alto	Baja (Contexto)
DMA	Bajo (Config & TC)	No (Lo hace otro)	Alta