

Estructura de Computadores

Grado de Ingeniería Informática
ETSINF

Tema 9: Técnicas de transferencia de Entrada/Salida



Curso 2018-2019



Objetivos

- Comprender los requerimientos para las transferencias de datos en las operaciones de E/S.
- Estudiar los distintos mecanismos de transferencias existentes: PIO (Programmed Input/Output) y DMA (Direct Memory Acces).
- Entender cómo se gestionan las transferencias de datos mediante mecanismos de acceso directo a memoria.

Contenido

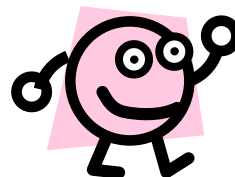
- 1 – Técnicas de transferencia de datos
- 2 – Transferencias por programa: PIO
- 3 – Transferencias por acceso directo a memoria: ADM

Bibliografía

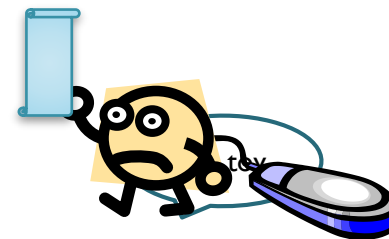
- Patterson, D.A., Hennessy, J.L.
 - ✓ Estructura y diseño de computadores. La interfaz hardware-Software (4ª ed.). Ed. Reverté, 2011
 - Cap 6
- Stallings, W.
 - ✓ Organización y arquitectura de computadores (7ª ed.). Ed. Prentice Hall, 2006
 - Cap. 7
- Hamacher, V.C., Vranesic, Z.G., Zaky, S.G.
 - ✓ Organización de computadores (5ª ed.). Ed. McGraw Hill, 2003

I – Técnicas de transferencia de datos

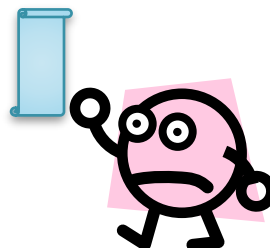
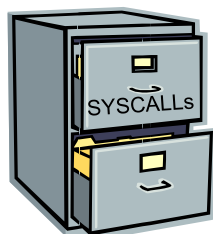
U.C. Pito



Mou Sito



Entradas / Salidas



La transferencia de datos

- Es la operación por la que se mueven los datos desde el periférico a la memoria principal (entrada) o de la memoria principal al periférico (salida).
 - ✓ La naturaleza del periférico impone la cantidad de datos a transferir y la velocidad (ancho de banda mínimo) a la que hay que realizar las transferencias.
 - ✓ Según el volumen de información a transferir, tenemos:
 - Periféricos de caracteres
 - Periféricos de bloques
 - ✓ Según la velocidad:
 - Periféricos de velocidad baja – media
 - Periféricos de velocidad alta – muy alta

La transferencias de caracteres

- Hay que transferir unos pocos bytes.
 - ✓ Las transferencias se realizan mediante lecturas/escrituras en los registros de la interfaz.
 - ✓ Es típico en periféricos sencillos, con requerimientos de velocidad bajos o medios.
 - ✓ Ejemplos:
 - Teclado: 3 bytes a 10 Kbps
 - Ratón, joystick: 4-8 bytes a 10 Kbps
 - Periféricos E/S en entornos industriales: sensores, fotocélulas, actuadores, motores etc.. 1 – 100 bytes a velocidades de KBps

La transferencia de bloques

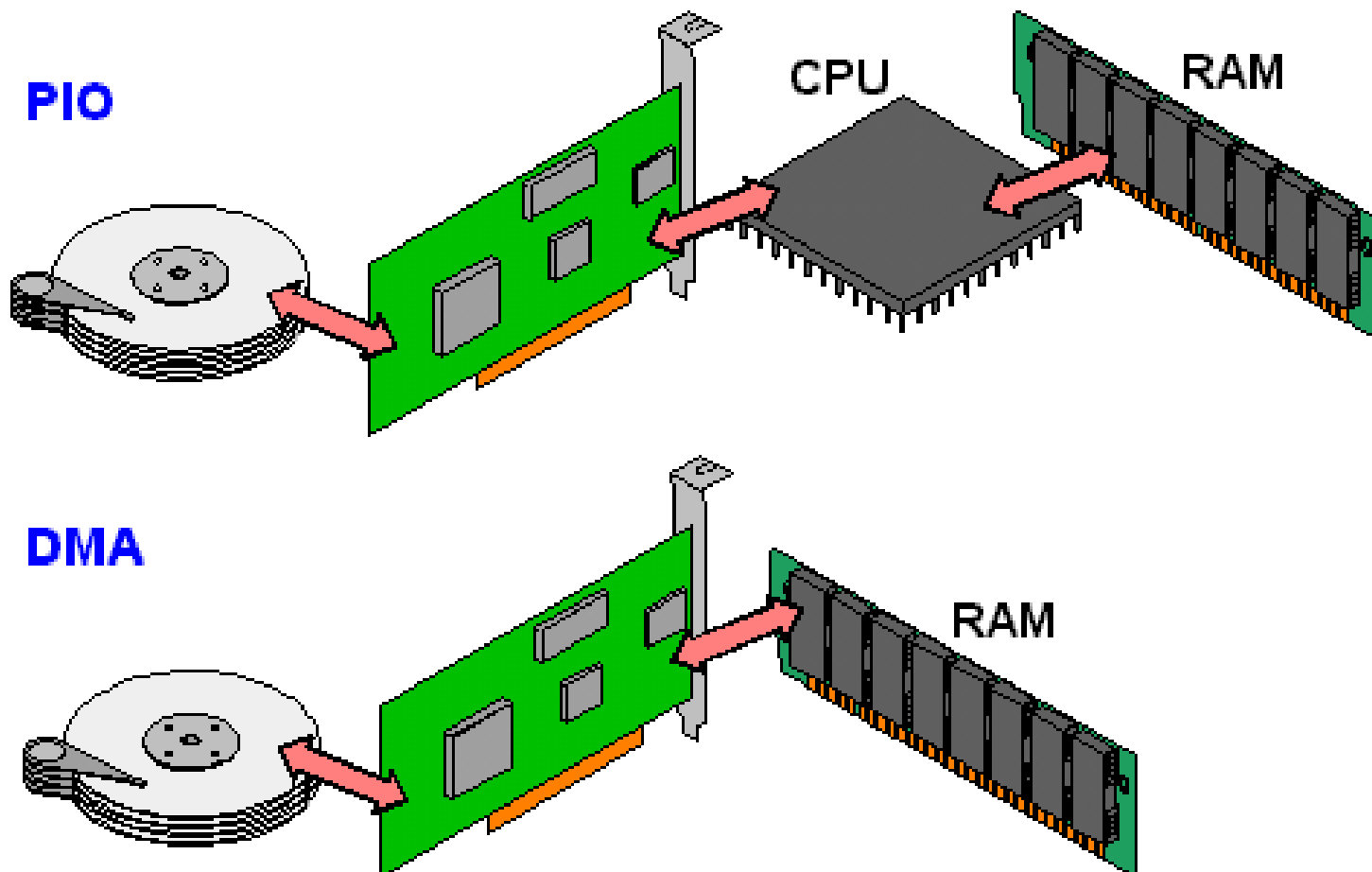
- Hay que transferir bloques de datos de tamaños muy superiores a la palabra del procesador.
 - ✓ Los bloques tienen un significado concreto en el contexto del periférico. P.E un sector de un disco, o una pista de un CD-A.
 - ✓ Para transferir un bloque, hay que hacer muchas transferencias elementales (de un byte o de una palabra) seguidas.
 - ✓ La naturaleza del periférico impone un mínimo ancho de banda para este tipo de transferencias.
 - ✓ Ejemplos
 - Disco: sectores de 512 bytes, a 100 MBps
 - Tarjeta de red: bloques de 1500 bytes a 10/100/1000 Mbps
 - Tarjeta de sonido: bloques de 1024 bytes, a 200 KBps

Técnicas de transferencia

- Transferencia por programa: (**PIO: Programmed Input/Output.**)
 - ✓ La UCP lleva a cabo la transferencia mediante la ejecución de instrucciones IN/OUT o Load/Store, osea, por programa:
 - `IN dato, puerto` `OUT dato, puerto`
 - `lw $8, puerto($t0)` `sh $8, puerto($t0)`
 - ✓ Se accede a los registros o memoria de la interfaz del periférico.
- Transferencia por acceso directo a memoria (**DMA: Direct Memory Access**)
 - ✓ La transferencia se hace sin la intervención de la UCP.
 - ✓ Es la propia interfaz del periférico la que gestiona las transferencias a la memoria.
 - ✓ Se dice que la interfaz tiene capacidad de maestro (BUS Mastering)

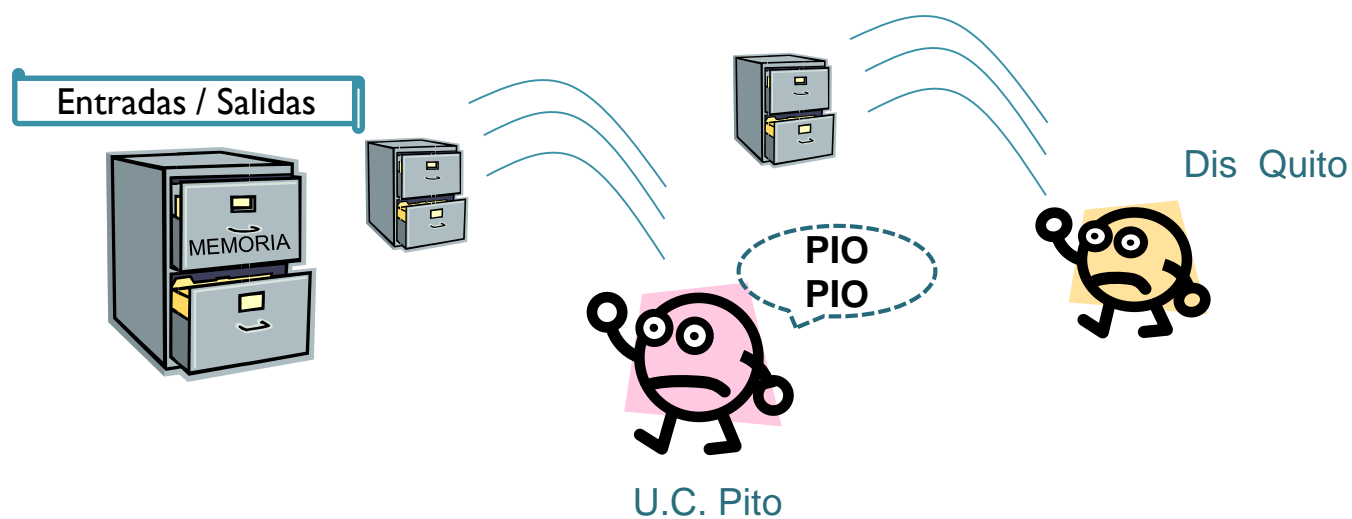
PIO versus DMA

From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.



2 – Transferencias por programa

PIO: Programmed Input / Output



Tipos de transferencias PIO

Tipo	Nº datos	Sincronización	Modo	Velocidad de Transferencia	Factor limitante
Simple	I	Prueba de estado		SDTR	Latencia del periférico
		Interrupciones		SDTR	Latencia del periférico
Bloque	N	Prueba de estado		SDTR	Latencia del periférico
		Interrupciones	Palabra-a-palabra	SDTR	Latencia del periférico
		Interrupción	Bloque	IODTR	CPU + BUS

SDTR: Sustained Data Transfer Rate

Latencia del periférico: Lo que tarda en periférico estar preparado para cada transferencia

IODTR: Input/Output Data Transfer Rate

CPU +BUS: Lo que cuesta a la CPU en leer el dato de la interfaz y escribirlo en memoria más la actualización de punteros y contadores.

Transferencias simples

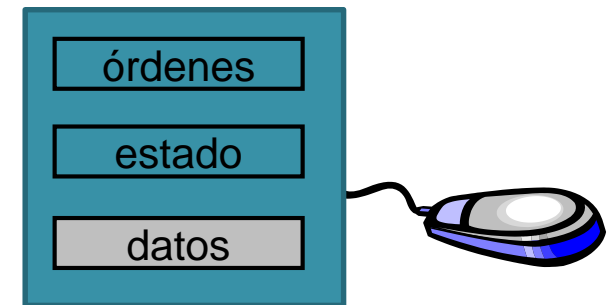
- ✓ Se transfieren uno (o unos pocos bytes/palabras) por programa
 - Se lee / escribe algún registro de Datos.
- ✓ La sincronización puede ser por prueba de estado o por interrupciones
- ✓ La velocidad la marca el periférico
- ✓ Se emplea con periféricos de velocidad baja o media.

```
LeeRaton: la $t0,DB_raton

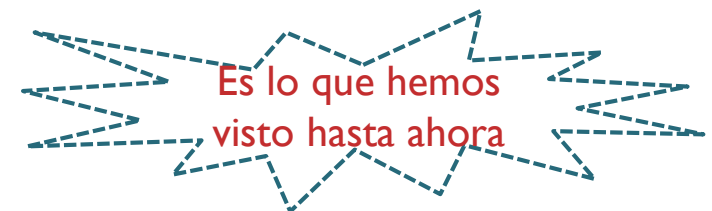
CdE:      { 1b $t1,estado($t0)
Sincronización { andi $t1,$t1,bit_R
              { beqz $t1,CdE

              1b $t1,datos($t0) }
              sb $t1,0($a0)      } Transferencia
                                simple

Cancelación { li $t1,bit_cancel
             { sb $t1,ordenes($t0)
             { jr $ra
```



Interfaz ratón

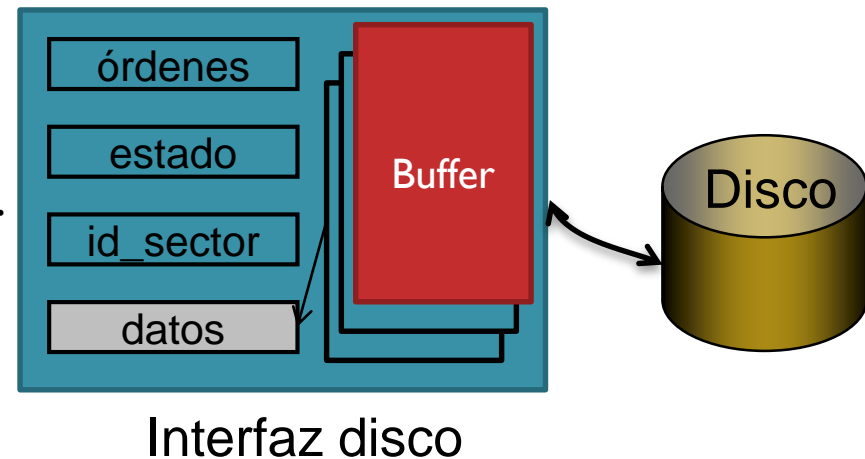


Transferencia de bloques: El uso de *buffers*

- Los periféricos de bloques suelen hacer uso de '*buffers*' internos

✓ Un *buffer* es una memoria de estado sólido (a veces llamada *cache*) situada en la interfaz y que hace de elemento de almacenamiento temporal de los datos.

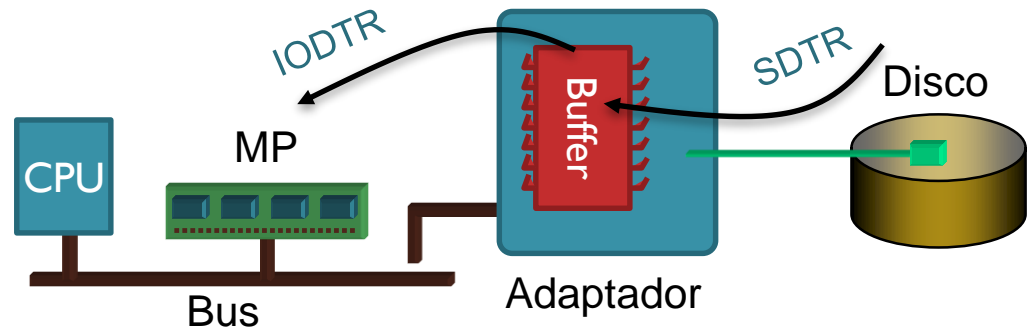
- ✓ El acceso al *buffer* se realiza:
- A través de registro/os de datos.
 - Como memoria de la interfaz.



- Las transferencias se realizan entre el *buffer interno* y unos *buffers de memoria* declarados en el programa.

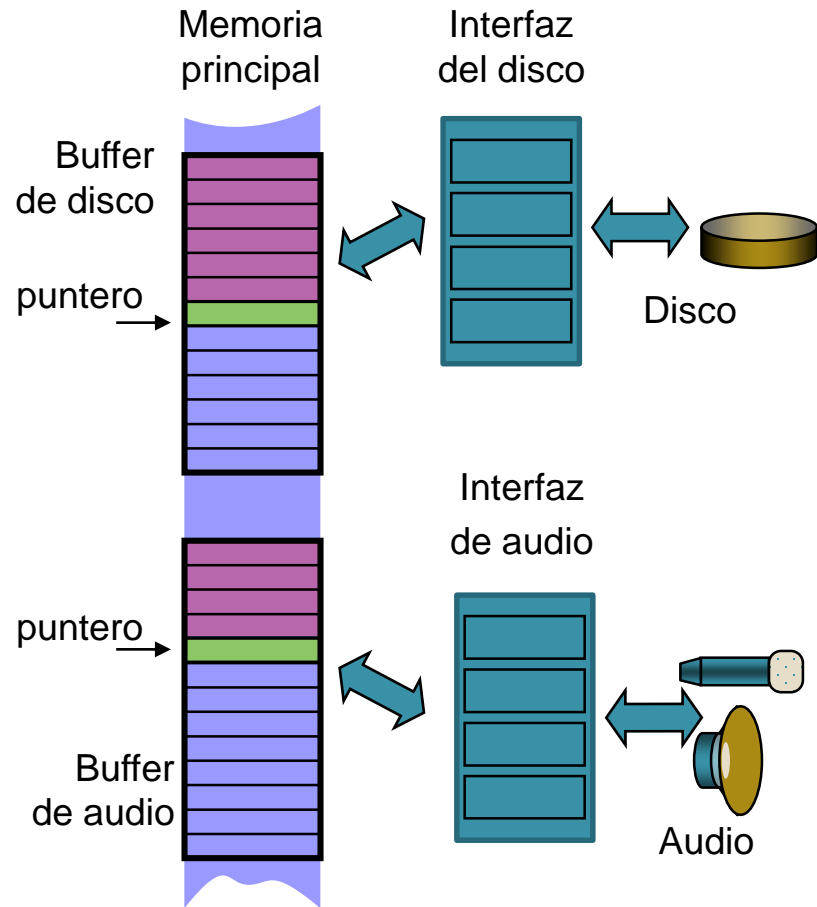
El *buffer* en el periférico

- ✓ Las transferencia de un bloque datos requiere una orden inicial al periférico.
 - Por ejemplo;; Leer sector 23
- ✓ Al ejecutar la orden el periférico transfiere los datos al **buffer interno**.
 - La velocidad de transferencia depende del periférico concreto: (DTR : Data Transfer Rate). El fabricante suele dar un valor sostenido (SDTR: Sustained Data Transfer Rate).
- ✓ Las transferencias entre el buffer interno y la MP se hacen por ADM o por PIO a la velocidad que permita el bus.
 - IODTR: Input/Output Data Transfer Rate



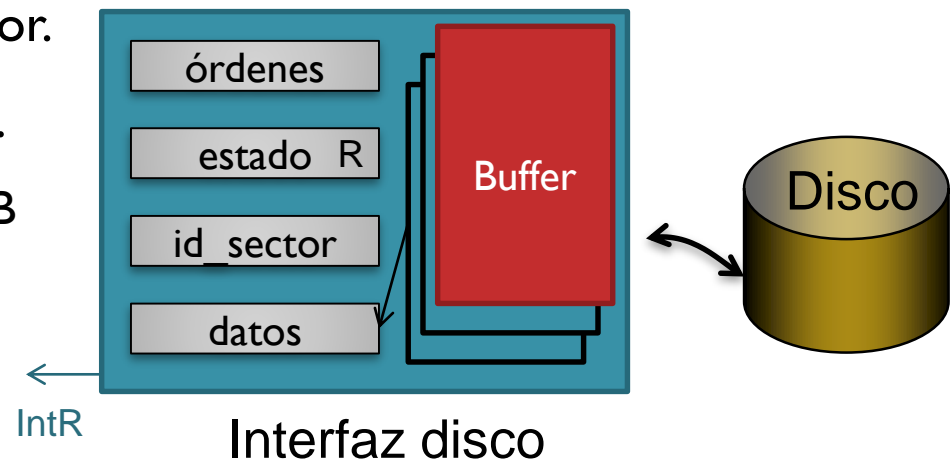
El *buffer* en la Memoria Principal

- ✓ La transferencia se hace entre el dispositivo (buffer interno) y un **buffer de memoria** con capacidad para uno o más bloques.
- ✓ Generalmente, cada dispositivo de bloques tiene reservado un buffer exclusivo en la MP.
- ✓ Cada buffer tiene una **dirección inicial** y un **puntero** que indica el punto actual de la transferencia.
- ✓ Hasta que no se transfiere todo un bloque, no se da por terminada la operación de entrada/salida.



Interfaz ejemplo

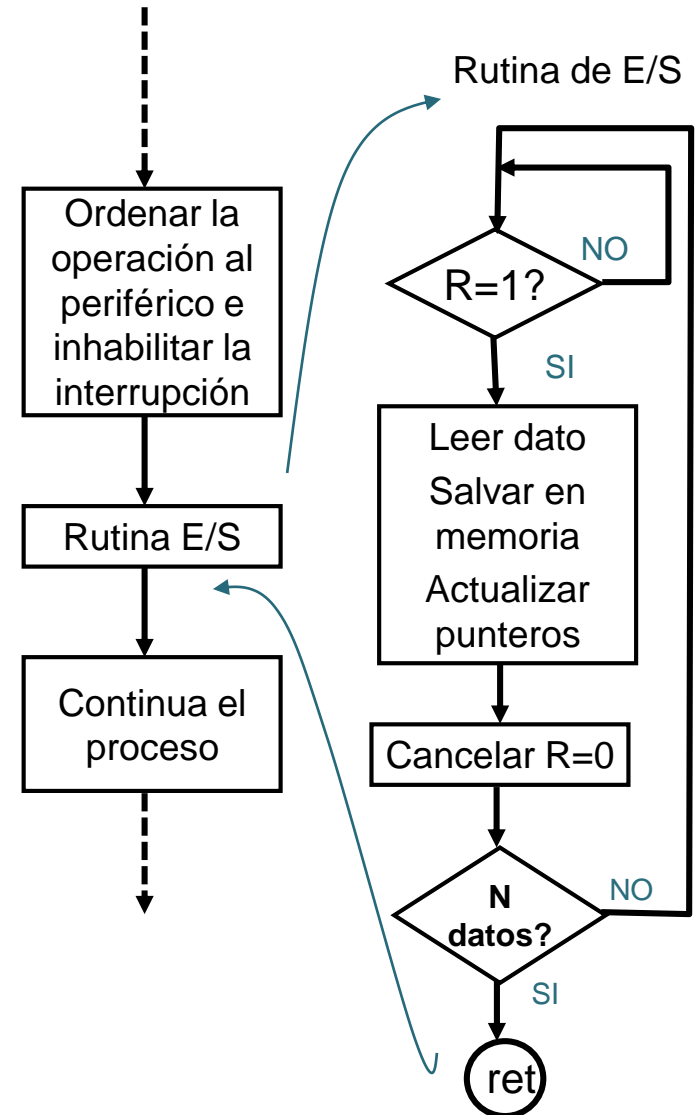
- Controlador básico de disco
 - ✓ Permite la lectura y escritura de sectores de hasta 512 bytes.
- Interfaz
 - ✓ *estado*: registro que informa de la disponibilidad del periférico.
 - ✓ *órdenes*: registro de órdenes con bits que determinan el sentido de la transferencia (lectura: del controlador a la memoria o escritura: de la memoria al controlador) y otros detalles.
 - ✓ *id_sector*: coordenadas del sector.
 - ✓ *datos*: registro de datos (8 bits).
 - Acceso al *Buffer* interno de 512 B



Transferencia de bloque por programa

- Sincronización por consulta de estado: lectura de bloque

- ✓ Primero hay que ordenar la operación al periférico e inhabilitar la interrupción.
- ✓ La CPU queda a la espera de que el periférico esté preparado.
- ✓ El periférico ejecuta la orden y transfiere los datos a su buffer interno.
- ✓ Se lee el bloque sincronizando cada palabra mediante consulta de estado.
 - Entre cada dos lecturas la interfaz coloca un nuevo dato en el registro de datos y vuelve a activar el bit R.
 - La rutina de servicio debe leer todo el bloque de N datos.



Lectura de bloque por programa

- Sincronización por consulta de estado:

\$a0 : Puntero al buffer de memoria

\$a1: Contador de palabras a transferir (N)

```
.data
buffer:.space 512
...
.text
# preparación del bucle
    la $a0,buffer
    li $a1,512 # long. bloc
# programación del periférico
    la $t0,DirBase_disco
    li $t1,coorden_sector
    sw $t1,id_sector($t0)
    li $t1,bits_de_control
    sw $t1,ordenes($t0)
    jal Leer_Bloque
    ....
    ....
```

```
Leer_Bloque: la $t0, DirBase_disco
```

```
    bucle: { lb $t1, estado($t0)
              Sincronización  { andi $t1, $t1, 0x01 # bit R
                                { beqz $t1, bucle

                                lb $a2,datos($t0) } Transferencia
                                sb $a2,0($a0)

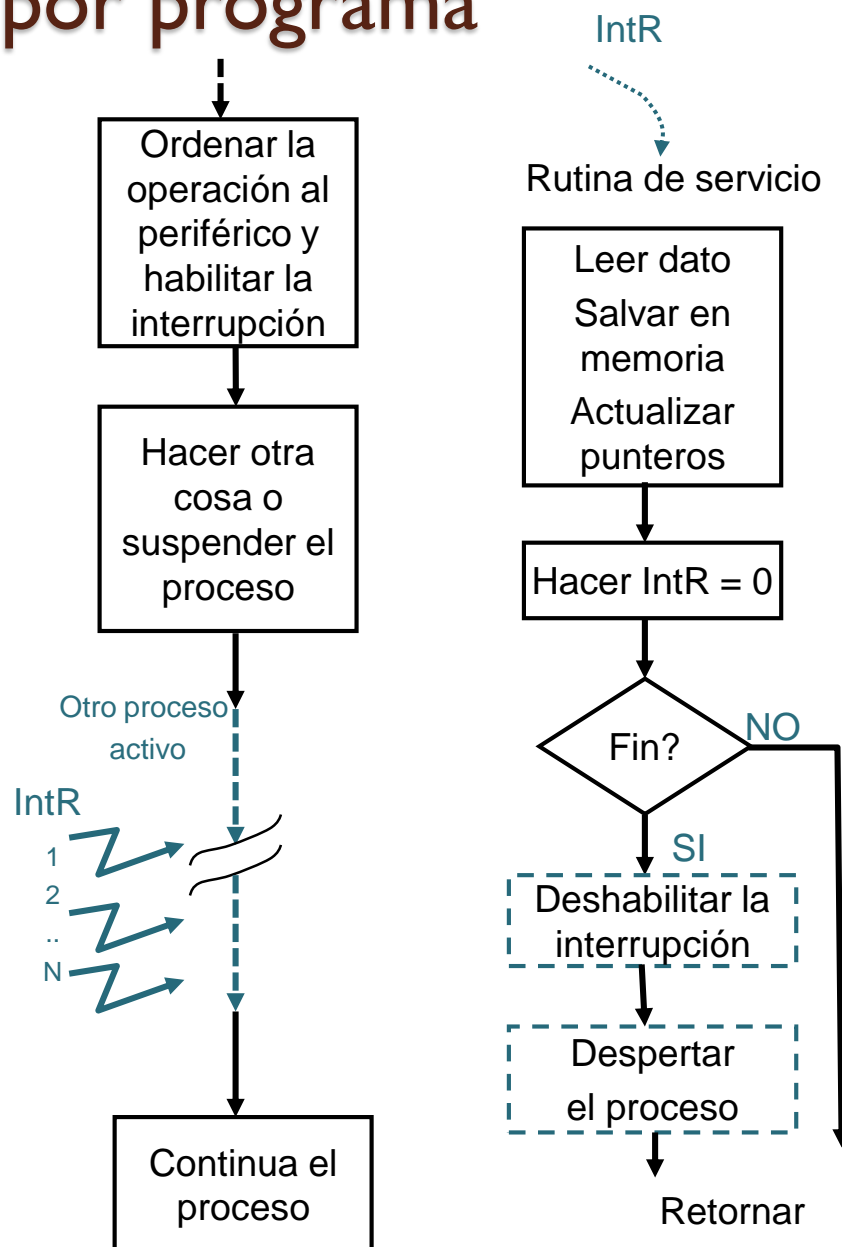
                                addi $a0,$a0,1
                                addi $a1,$a1,-1 } Actualización
                                bnez $a1,bucle   punteros
                                jr $ra
```

N

Transferencias de bloque por programa

- Sincroniz. por interrupciones:
Modo palabra-a-palabra

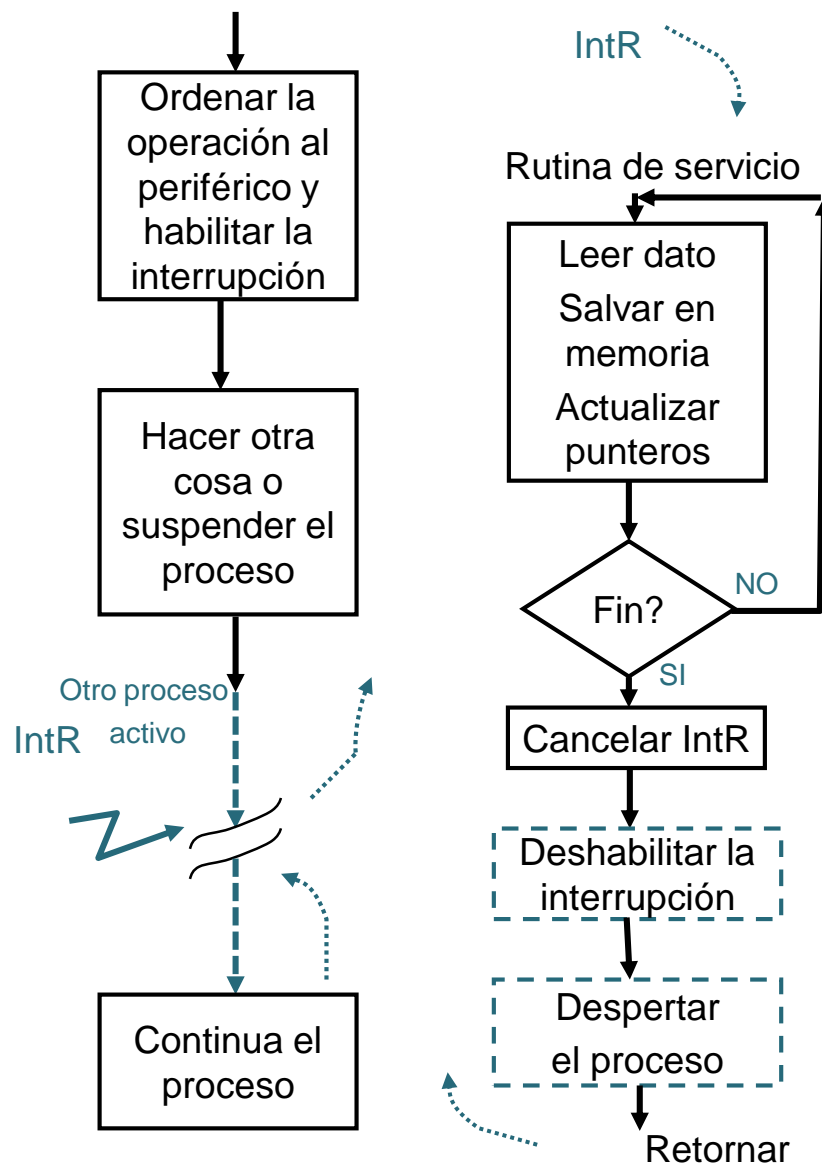
- ✓ Primero hay que ordenar la operación al periférico y habilitar la interrupción.
- ✓ El periférico ejecuta la orden y activa la interrupción (IntR) con cada palabra a transferir: **Sincronización por palabra**
- ✓ Transferencia:
 - La rutina de servicio lee cada dato y los transfiere al buffer de memoria, actualizando los punteros. Se DEBE cancelar la interrupción (IntR=0) tras la transferencia de cada palabra.
 - La interrupción se repite N veces, hasta completar el bloque deseado.
 - La velocidad la marca el periférico
- ✓ Al finalizar hay que deshabilitar la interrupción y despertar al proceso.



Transferencias de bloque por programa

• Sincroniz. por interrupciones: Modo bloque

- ✓ Primero hay que ordenar la operación al periférico y habilitar la interrupción.
- ✓ El periférico ejecuta la orden y transfiere los datos a su buffer interno. En ese momento activa la interrupción.:
Sincronización de bloque
- ✓ Transferencia:
 - La rutina de servicio debe leer TODO el bloque de datos y cancelar la interrupción (IntR=0).
 - Tras leer cada dato, la interfaz coloca automáticamente el siguiente dato en el registro de datos.
 - La velocidad de lectura es la máxima posible (depende de la velocidad de ejecución de la CPU y del bus: IODTR)
- ✓ Al finalizar hay que deshabilitar la interrupción y despertar al proceso.



Ejemplo:

- Lectura bloque: Interrupciones y modo bloque

```
.kdata
buffer: .space 512
...
.ktext
.....

# programación del periférico
la $t0,DirBase_disco
li $t1,coorden_sector
sw $t1,id_sector($t0)
li $t1,bits_de_control
sw $t1,ordenes($t0)
jal suspende_esto_proceso
.....
.....
```

\$a0 : Puntero al buffer de memoria

\$a1: Contador de palabras a transferir (N)

```
Int_R:  la $a0, buffer
        li $a1, 512  #long bloque
        la $t0, DirBase_disco

        Bucle:  lb $a2, datos($t0)
                sb $a2, 0($a0)

                addi $a0,$a0,1
                addi $a1,$a1,-1
                bnez $a1,bucle

        li $t1, cancel_noint
        sw $t1, ordenes($t0)

        jal desierto_proceso_en_espera

        b retexc
```

Transferencia lectura

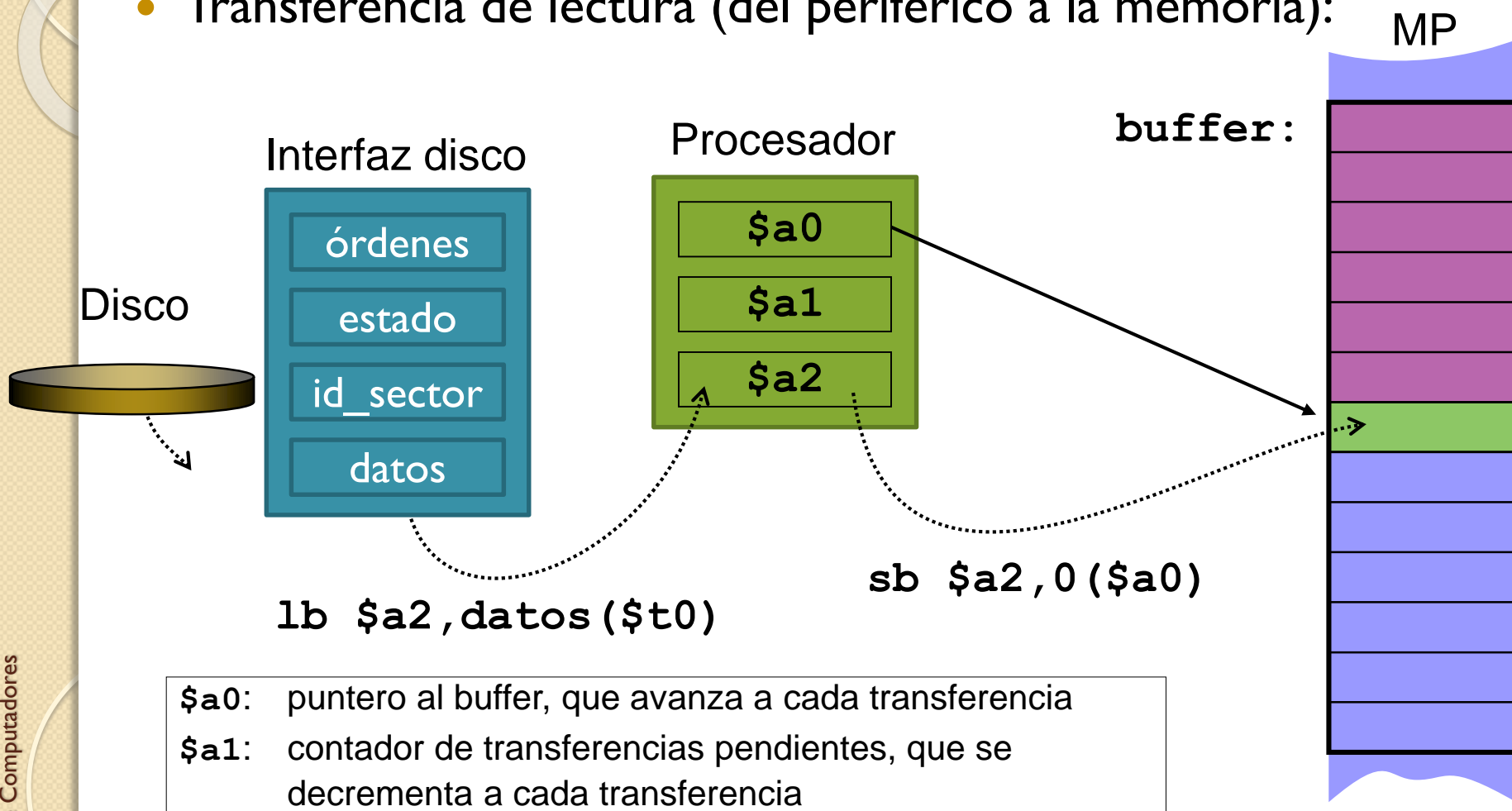
Actualización punteros

Cancela e Inhabilita intR

N

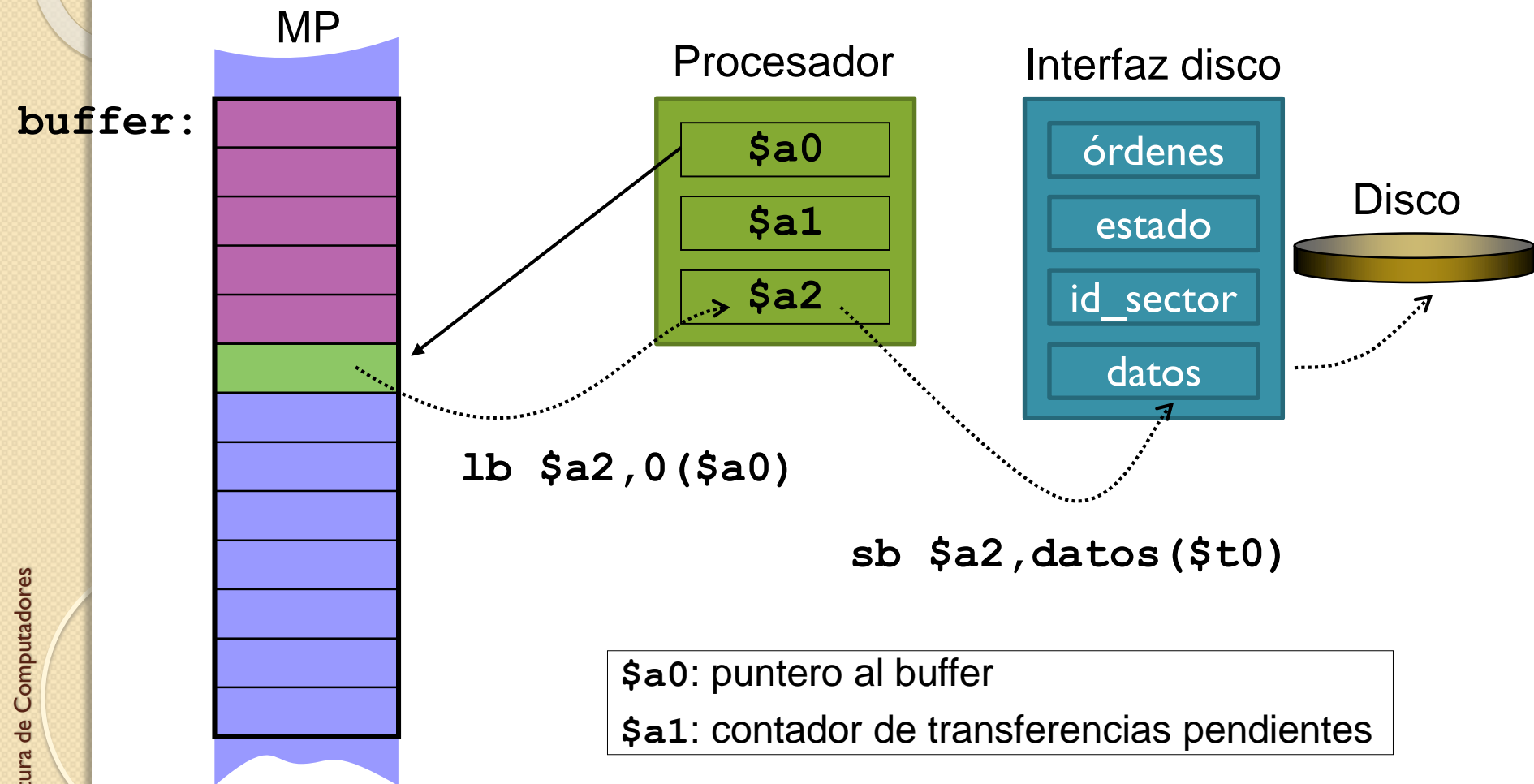
Uso de punteros y contadores

- Transferencia de lectura (del periférico a la memoria):



Uso de punteros y contadores

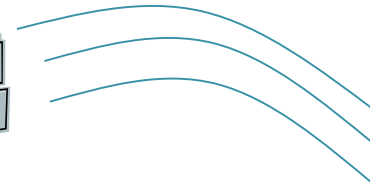
- Transferencia de escritura (de la memoria al periférico)



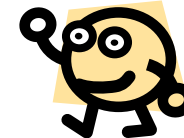
3 – Transferencias por acceso directo a memoria

DMA: Direct Memory Acces

Entradas / Salidas



Dis Quito

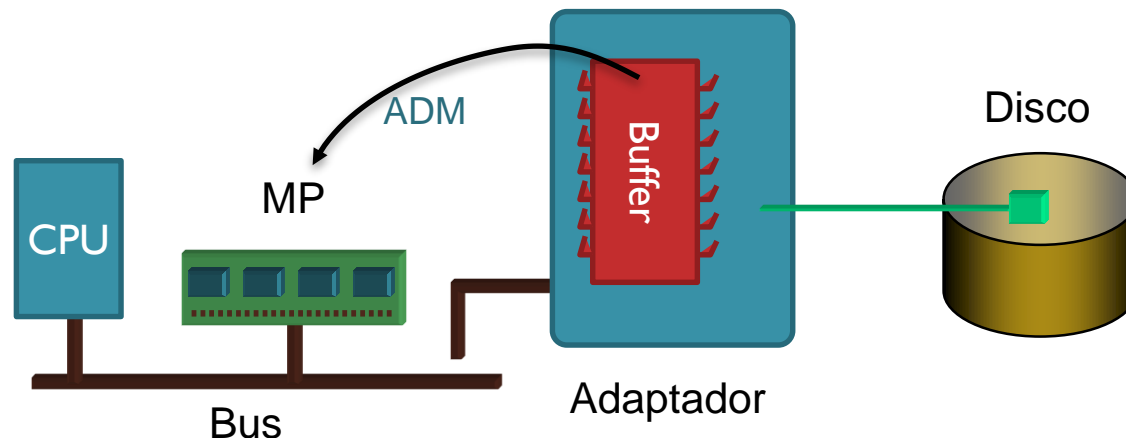


$$12+3 \times 45/3+2-12 \times 3-2+5/23 \times 14 \\ +2-34/24 \times 8-12+\text{sqrt}(23/4)+12=$$



Transferencias por ADM

- ADM: Acceso Directo a Memoria
 - ✓ Se trata de que la interfaz del periférico sea la que se encargue de las transferencias directas de los datos a la memoria principal.
 - ✓ La operación se realiza por hardware
 - ✓ La CPU no interviene (tampoco la M. cache)
 - ✓ Sólo tiene sentido para transferencia de bloques a alta velocidad.



Soporte para transferencias por ADM

- Nuevas funciones de la interfaz del periférico
 - ✓ El controlador del periférico puede acceder a la memoria para leer y escribir manejando las líneas del bus, como si fuera el procesador (Bus Mastering).
 - ✓ La interfaz incorpora dos nuevos registros:
 - Puntero de memoria (para hacer el papel de \$a0)
 - Contador de bytes o palabras (para hacer de \$a1).
 - ✓ Sincronización de fin de transferencia
 - El controlador del periférico indica que la transferencia ha terminado mediante una interrupción.

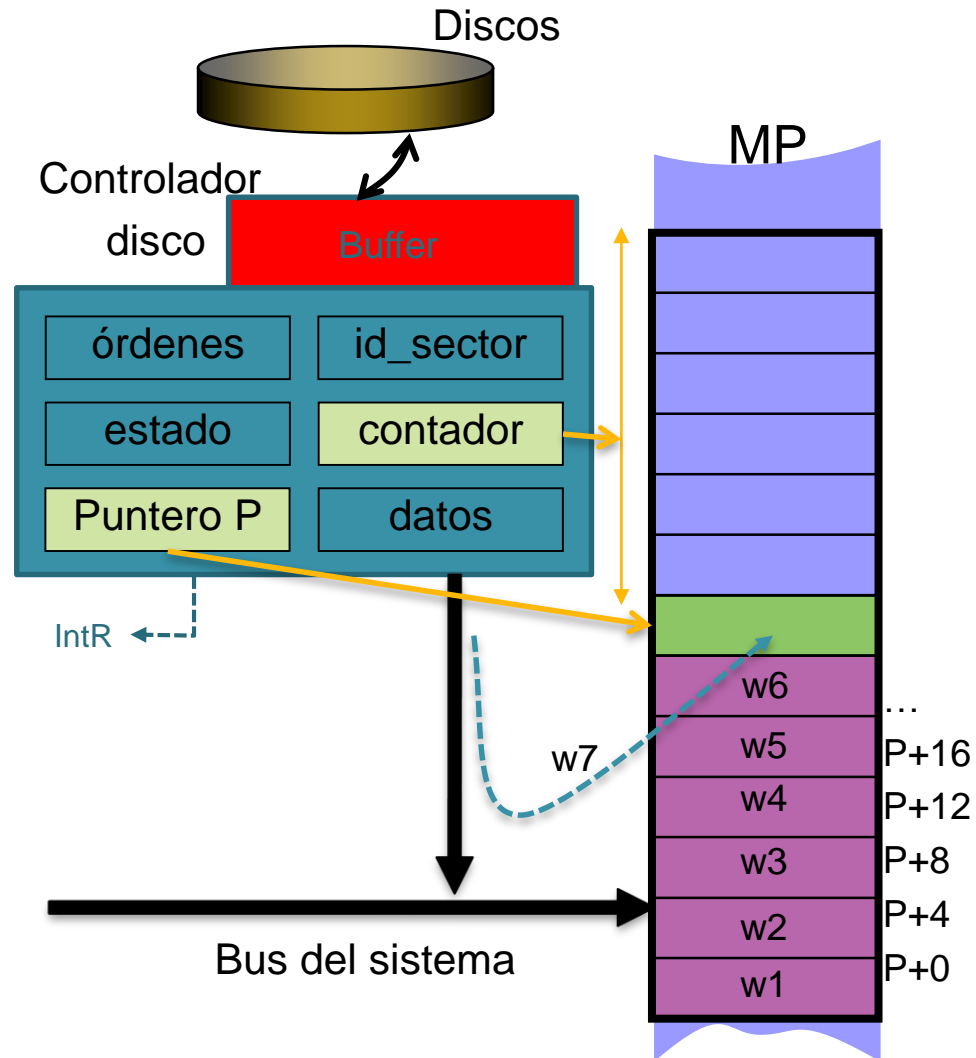
¿Cómo funciona el ADM?

- ✓ Primero se programa la transferencia por ADM en la interfaz del periférico.
- ✓ El periférico ejecuta la operación y transfiere los datos al buffer interno.

✓ Transferencia de Entrada:

- La interfaz transfiere cada palabra de buffer a la dirección de memoria apuntada por el Registro Puntero.
- Con cada transferencia la dirección se autoincrementa (+4) y el registro Contador se autodecrementa (-1).
- La operación se repite palabra por palabra.
- Cuando el Contador llega a cero se ha terminado la operación y se emite la interrupción de fin de transferencia.

- ✓ Para Salidas, primero se transfiere por ADM de memoria al buffer interno y luego se transmite al periférico.



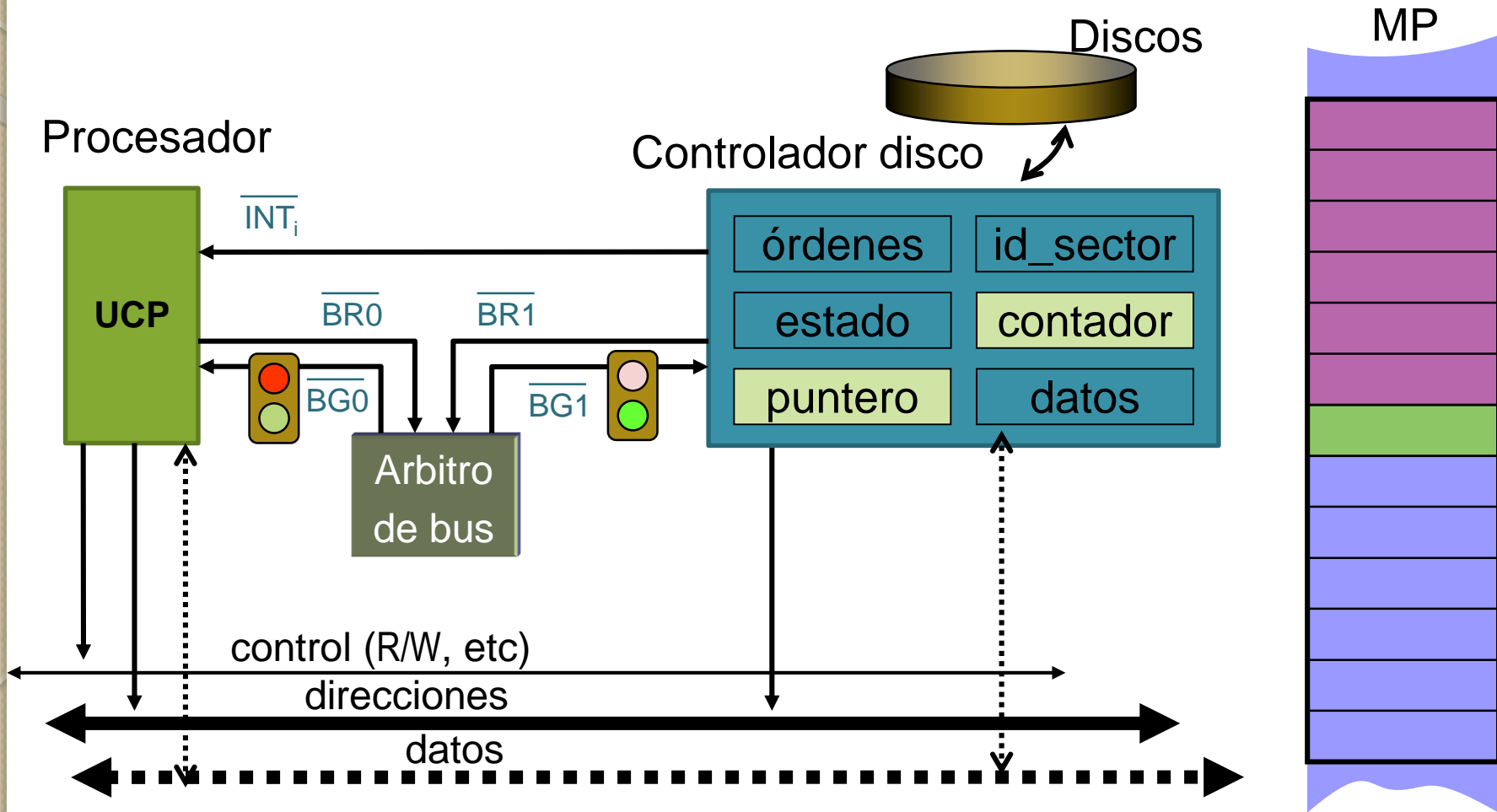
¡Ojo! El Contador cuenta palabras transferidas (no bytes)

Compartición del bus

- ✓ Hasta ahora el único dispositivo que accedía a la memoria era la CPU (o el controlador de cache de la CPU)
- ✓ Ahora aparecen otros dispositivos capaces de conducir el bus (Bus Masters): Los periféricos con capacidad ADM
- ✓ El sistema necesita un recurso para distribuir el uso del bus entre los distintos maestros de bus → Arbitro de bus
- ✓ ¿Dónde está el árbitro del bus?
 - En el chipset de la CPU → Arbitraje centralizado
 - En cada interfaz de periférico → Arbitraje distribuido
- ✓ ¿Qué hace un árbitro de bus?
 - Recibe peticiones para usar el bus (BusRequest)
 - Concede el bus a cada contendiente según un esquema de prioridades (BusGrant)

Conexionado

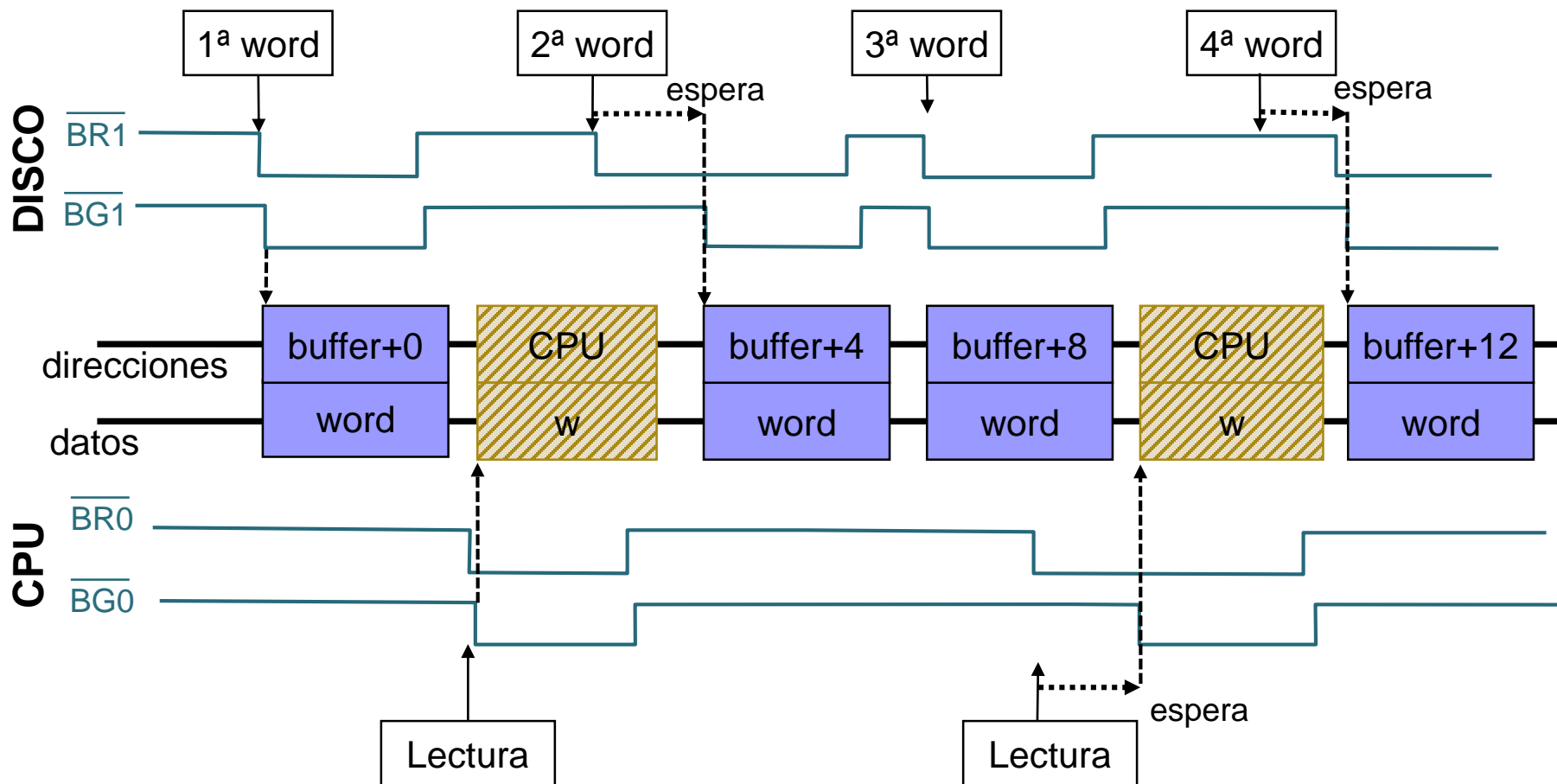
- Conexión de un controlador de disco con ADM:



Arbitraje del bus

- Cronograma ejemplo

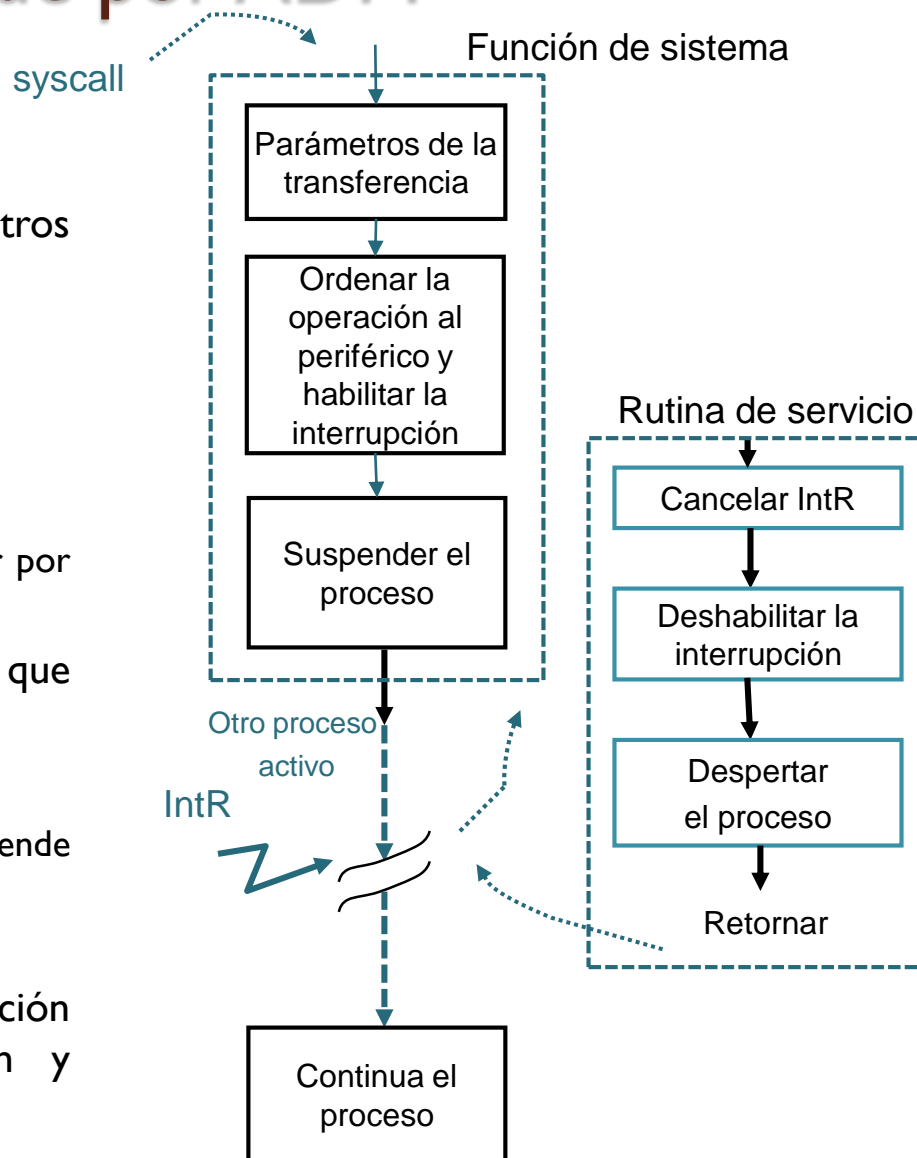
- ✓ Acceso del controlador de disco al bus mientras el procesador hace accesos de lectura



Transferencias de bloque por ADM

• Esquema general:

- ✓ Primero hay que programar los parámetros de la transferencia en la interfaz:
 - Tamaño del bloque (Contador)
 - Dirección de memoria (Puntero)
 - Otros parámetros
- ✓ Por último dar la orden y habilitar INT:
 - En el registro de órdenes (ej. Leer sector por ADM)
- ✓ El proceso se suspende a la espera de que termine la transferencia.
- ✓ Transferencia:
 - Se gestiona por ADM. La velocidad depende del ancho de banda del bus.
 - Cuando termina se activa la interrupción.
- ✓ En la rutina de servicio de la interrupción hay que deshabilitar la interrupción y despertar al proceso.



Programación del ADM

- ✓ Ejemplo: Lectura / Escritura de un sector de disco (`Id_sector`) de 512 bytes y transferencia a memoria por ADM (puntero a `buffer`)

Función	Código	Argumentos	Resultado
<code>Read_Disk</code>	<code>\$v0 = 666</code>	<code>\$a0 =</code> puntero al <i>buffer</i> <code>\$a1 = Id_sector</code>	
<code>Write_Disk</code>	<code>\$v0 = 667</code>	<code>\$a0 =</code> puntero al <i>buffer</i> <code>\$a1 = Id_sector</code>	

Programa
de usuario

```
                .data
buffer:         .space 512
                ...
                .text
                ...
# petición de lectura del disco
                li $v0,666
                la $a0,buffer
                li $a1,Id_sector
                syscall
```

Buffer de memoria
del usuario

Funciones del sistema

- Tratamiento de las funciones en el manejador

```
fun666:      .ktext
            la $t0, dir_base_disc
            # preparación de punteros
            sw $a0,puntero($t0)
            li $t1,128      #512B = 128W
            sw $t1,contador($t0)

            # parámetros de la operación
            sw $a1,id_sector($t0)

            # Orden
            li $t1,bits_de_control
            sw $t1,ordenes($t0)

            jal suspende_esto_proceso
            b retexc

Fun667:      .....    #idem
            .....
            b retexc
```

Para fun666:
Orden LEER_SECTOR
+ habilita interrupción

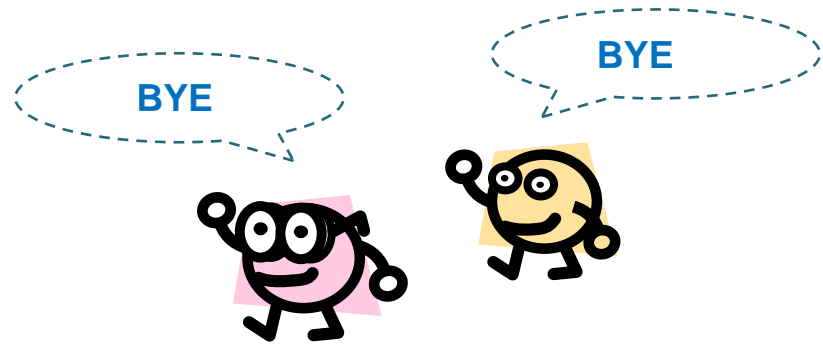
Para fun667:
Orden ESCRIBIR_SECTOR
+ habilita interrupción

Rutina de interrupción

- Rutina de servicio de la interrupción IntR* de finalización de la transferencia

```
intR:      .ktext  
          jal activar_proc_en_espera  
          j  retexc
```

Tema 9



FIN