Estructura de Computadores

Tema 7

Entrada/Salida

Dpto. Arquitectura y Tecnología de Sistemas Informáticos.
Universidad Politécnica de Madrid

Objetivos

- Comprender los problemas que surgen al intercambiar información con el mundo exterior a través de los dispositivos periféricos.
- Conocer los mecanismos que proporcionan los computadores para realizar dicho intercambio
- Conocer los mecanismos que proporcionan los computadores para sincronizarse con los periféricos (mecanismo de interrupciones) o para la transferencia de datos (DMA)

Bibliografía recomendada

- Stallings, W. “Organización y arquitectura de computadores”. Prentice Hall. 7ª Edición. 2006.
- Patterson, D. A., Hennessy, J. L. “Computer Organization and Design”. Morgan-Kaufmann. 4ª edición. 2009.
- de Miguel, P. “Fundamentos de los computadores”. Paraninfo. 9ª edición. 2004

Índice

1. Periféricos
   Ejemplo: disco duro
2. Introducción a E/S. Módulos de E/S. Instrucciones de E/S
3. Técnicas de E/S
   1. E/S programada
   2. E/S por interrupciones
   3. E/S por DMA

Periféricos

Ejemplos:

"Convencionales":

- Ratón, teclado, impresora, monitor
- Disco duro (HDD)
- discos ópticos (CD, DVD, Blu Ray)
- SSD (disco de estado sólido)
- Redes, línea serie (USART), p. paralelo, ethernet…

"Industriales": *computer sensors and actuators*

- Sensor de temperatura
- Motores para la orientación de un telescopio
- Captador de imágenes telescópica

Periféricos

Gran diversidad

- **Modo** de **funcionamiento**
- **Formato** y **tamaño** de los **datos**
- **Velocidad** de **transferencia**
- **Tiempo** de **acceso**

Una posible **clasificación**:

1. **Almacenamiento** ➔ *Parallel I/O*
2. **Comunicación**:

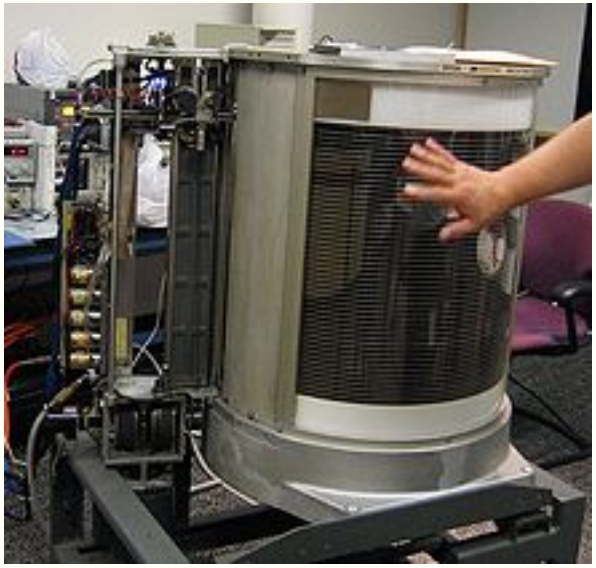
   Humanos ➔ Multimedia, *Brain interfaces*

   Computadores ➔ Redes: *cluster, Grid computing*

   "Medio físico" ➔ Sist. de control o *embedded systems*

Disco Duro (HDD)

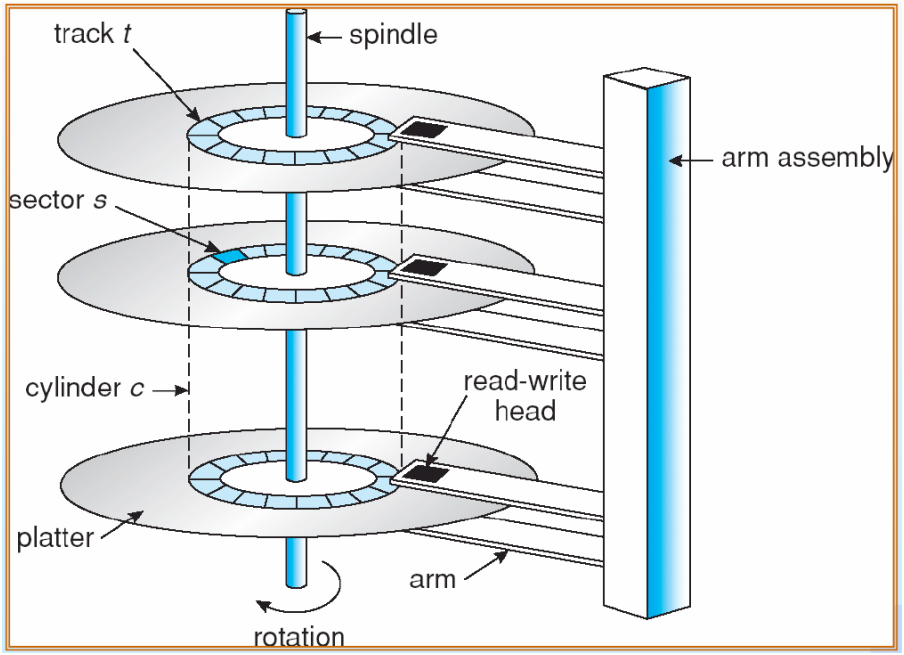
- IBM 1.956: IBM 350
  - 50 discos de 24” (61 cm)
  - 5 MB. T_{acc} 0,5 s



- IBM 1.973: Tecnología “Winchester”. IBM 3340
  - Discos sellados junto con el motor y las cabezas
  - 70 MB. T_{acc} 25 ms

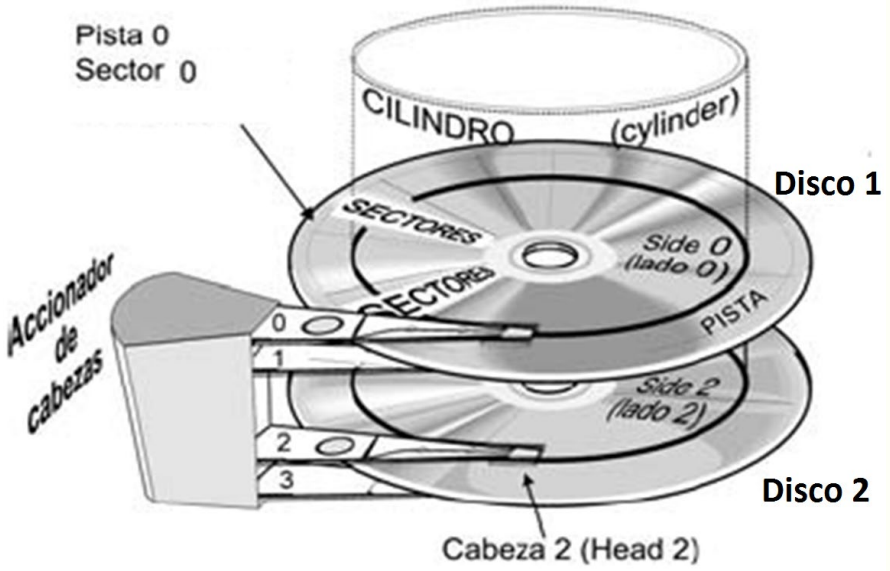
Periféricos: Disco duro (HDD)

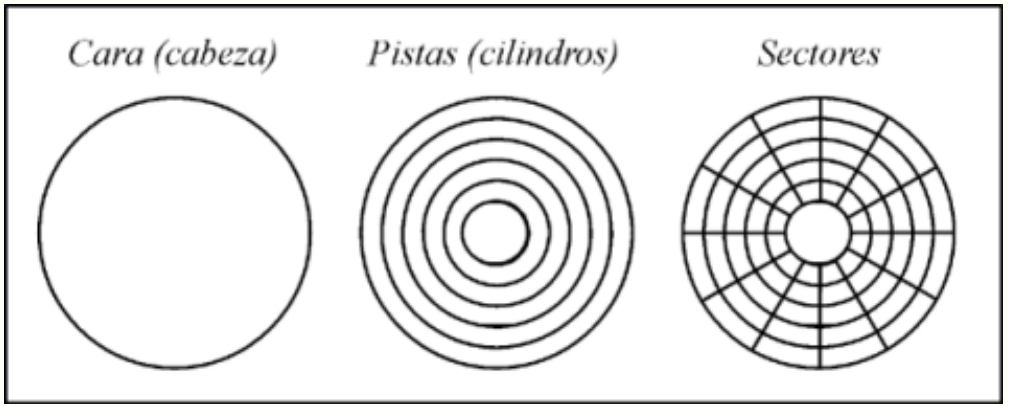
- Dispositivo tipo bloque
- Soporte de la memoria virtual
- Tamaño de los datos: bloque
- V_{transf}
- t_{acc}
- Modo de funcionamiento:
  - Organización: (p/c, sf/h, s)
  - Acceso a un sector
  - Distribución de múltiples sectores



Disco duro (HDD)

- Organización: Geometría. Coordenadas CHS
  - **cilindros** (C) o pistas
  - **superficies** (H) o caras
  - **sectores** (S)





Ejemplo: HDD

Funcionamiento:

- el **motor** gira **siempre** a la misma velocidad de rotación
- el **brazo** se mueve hasta el cilindro destino: $\mathbf{t_{búsqueda}}$
- una vez en cilindro destino, tiempo **giro** hasta el comienzo del sector: $\mathbf{t_{latencia}}$
- el t de transferencia será el de giro del sector: $\mathbf{t_{transf}} = \mathbf{t_{sect}}$

$$t_{opHDD} = t_{acc} + t_{transf}$$
$$t_{acc} = t_{búsqueda} + t_{latencia}$$
$$t_{búsqueda} = t_{posicionamiento} + t_{estabilización}$$

Ejemplo: HDD

Consideraciones sobre los tiempos que se deben tener siempre presentes:

$t_{búsqueda}$:

el motor no se para, luego conforme la cabeza se mueve el disco habrá avanzado un determinado nº de sectores

$t_{latencia}$:

en media es el tiempo de ½ vuelta

depende de la velocidad de rotación y del sector objetivo

t_{transf}:

es el t de giro del sector e igual si es Lectura o Escritura

$t_{transf} = t_{sect} = t_{rev}$/#sect/pista

Ejemplo: HDD

Ejercicio: HDD

Características:

Vel. de giro: 3.000 r.p.m. ➔ 20 ms/rev

Nº de pistas: 500 pistas

Nº de sectores/pista (fijos): 25 sect/pista ➔ $\mathbf{t_{sect}}$ = 0,8 ms/sect

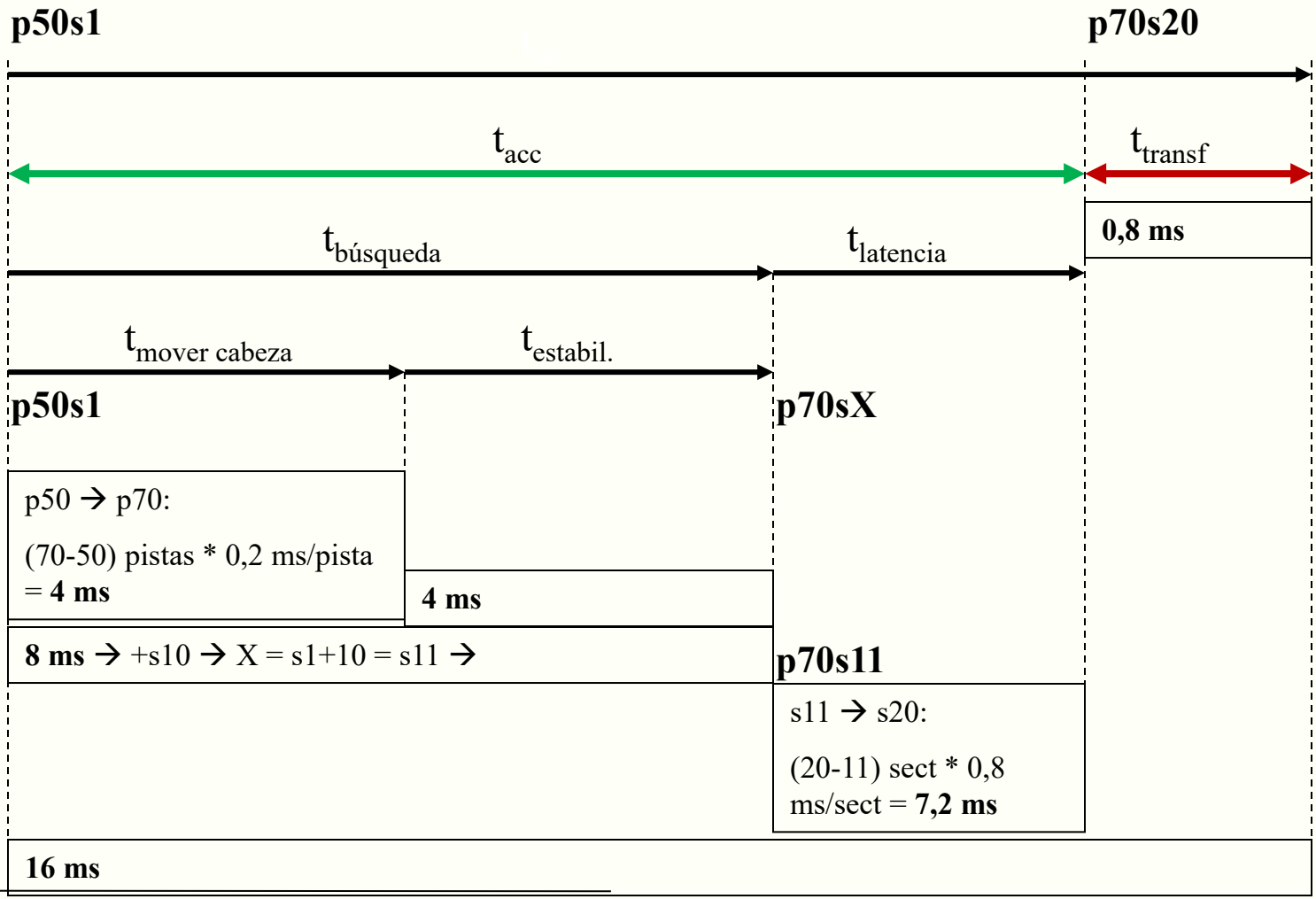
T de pista a pista consecutiva: 0,2 ms/pista

T estabilización al llegar pista destino: 4 ms

Caso:

En t= 0s la cabeza del disco se encuentra al comienzo del sector s1 en la pista p50: ¿en qué instante concluirá la transferencia del sector s20 de la pista p70?

Ejemplo: HDD


p50s1
p70s20
t_{acc}
t_{transf}
0,8 ms
$t_{búsqueda}$
$t_{latencia}$
$t_{mover\ cabeza}$
$t_{estabil.}$
p50s1
p70sX
p50 → p70:
(70-50) pistas * 0,2 ms/pista = 4 ms
4 ms
8 ms → +s10 → X = s1+10 = s11 →
p70s11
s11 → s20:
(20-11) sect * 0,8 ms/sect = 7,2 ms
16 ms

Unidades de Estado Sólido (SSD)

- **SSD, Solid State Drive (o Disk):** unidad de almacenamiento construida con circuitos integrados y elementos de memoria.
- No contiene elementos (electro-)mecánicos (a diferencia de los HDD).
- Ventajas:
  - Son más robustos físicamente: temperatura, golpes, etc.
  - Funcionan silenciosamente
  - Menor tiempo de acceso y de transferencia
  - Más fiables (menor tendencia a contener errores)
  - Menor peso
  - Menor consumo eléctrico (aprox. 1/3 que los HDD).

Índice

1. Periféricos
2. **Introducción a E/S. Módulos de E/S. Instrucciones de E/S**
    1. Tipos
    2. Direccionamiento
    3. Ciclos de bus
3. Técnicas de E/S
    1. E/S programada
    2. E/S por interrupciones
    3. E/S por DMA

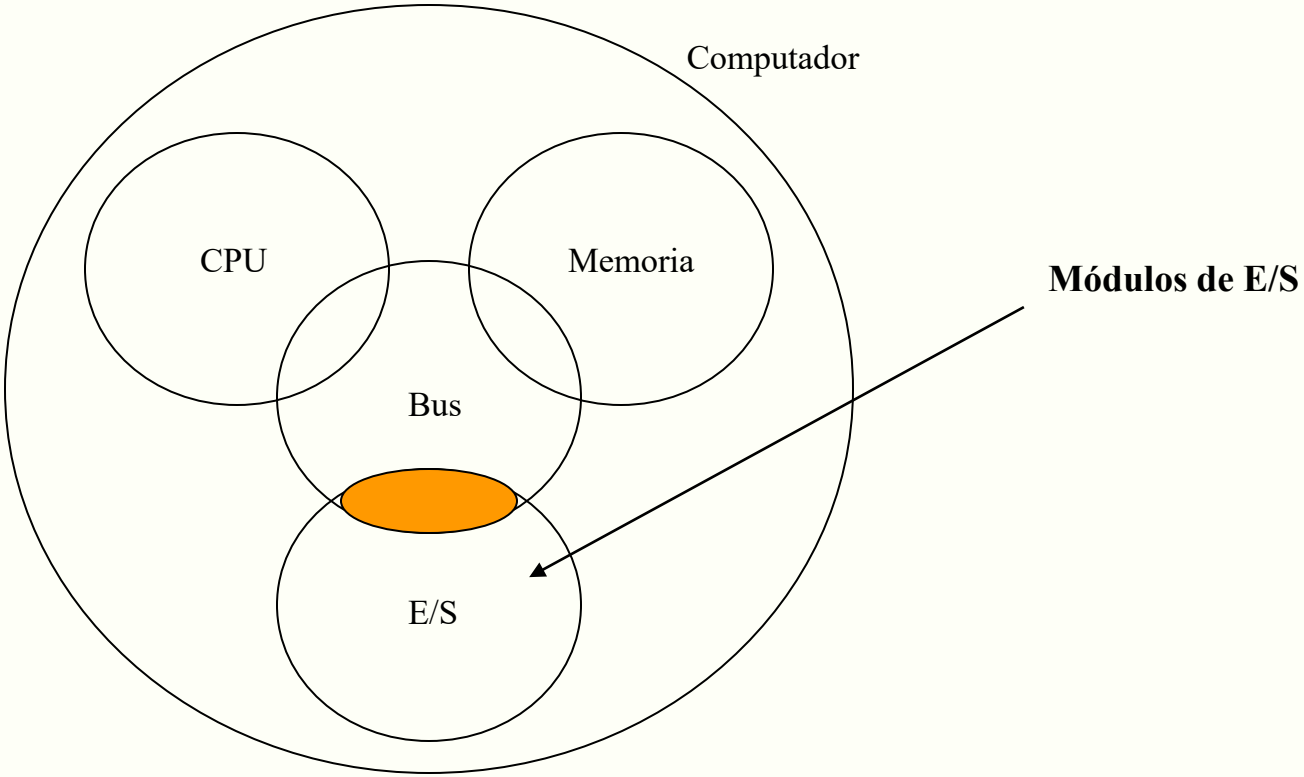
Problemática de la E/S

- Gran diversidad de periféricos con características muy diferentes :
  - Modo de funcionamiento (E, S, E/S, distintas órdenes,..)
  - Formato y tamaño de los datos
  - Velocidad de transferencia
  - Tiempo de acceso

➔ es necesario "unificar" la visión Hw de los periféricos

➔***Módulos de E/S:*** *ocultan las particularidades de cada periférico. La CPU sólo dialoga con los módulos que todos un interfaz semejante.*

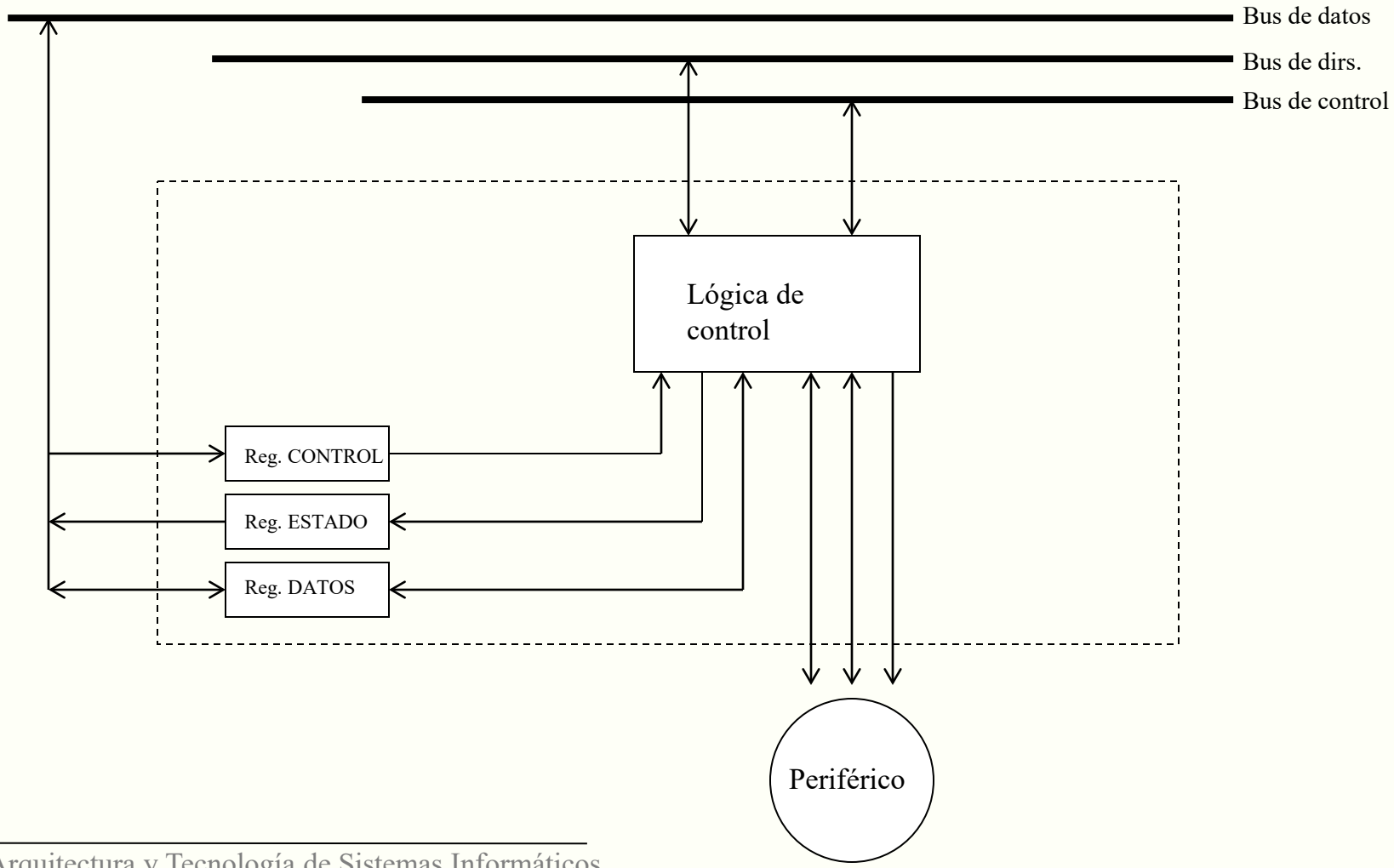
Módulos de E/S



Módulos de E/S

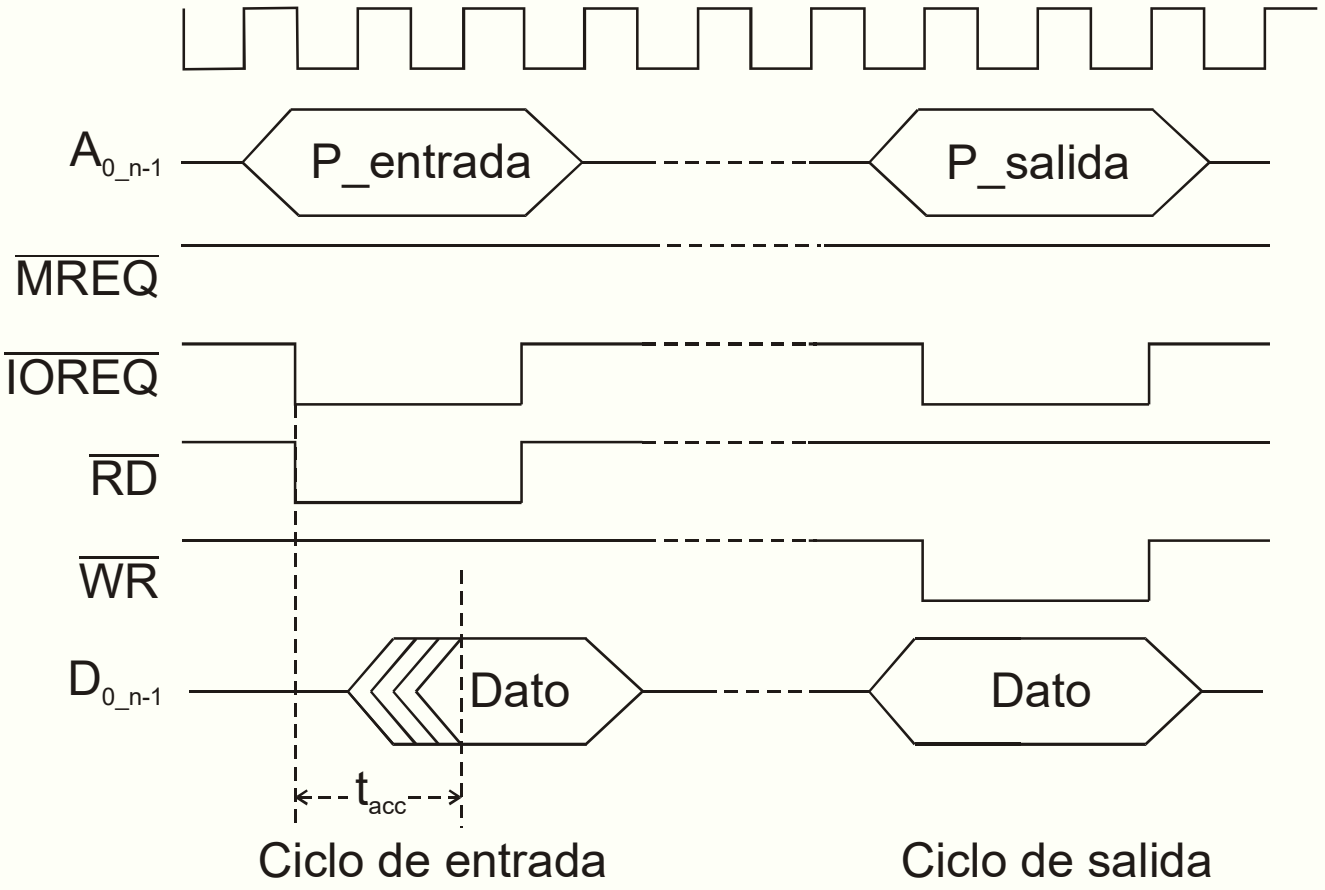
- Tienen **dos interfaces**
  - diálogo con la CPU (bus/memoria), estándar
  - diálogo con el periférico, específico del periférico
- Sus **funciones** son:
  - **Control** y **temporización** de la comunicación
  - Comunicación con la **CPU**
    - CPU → módulo**: CONTROL (órdenes, *commands*)**
    - CPU ← módulo**: ESTADO**
    - CPU ↔ módulo**: DATOS**
  - Comunicación con el **periférico:**
    - señales de control, estado y datos propias del periférico
  - *Buffering,* por la diferencia de velocidades
  - Control de **errores**

Módulos de E/S



Módulos de E/S

Ciclos de bus



Instrucciones de E/S

- Instrucciones de la Arquitectura del computador para la transferencia entre los **regs**. de la CPU y los **regs.** de los Módulos de E/S:

```
OUT .R1, /Dir_RegA_Px; (orig → dest)
IN  .R1, /Dir_RegB_Px; (dest ← orig)
```

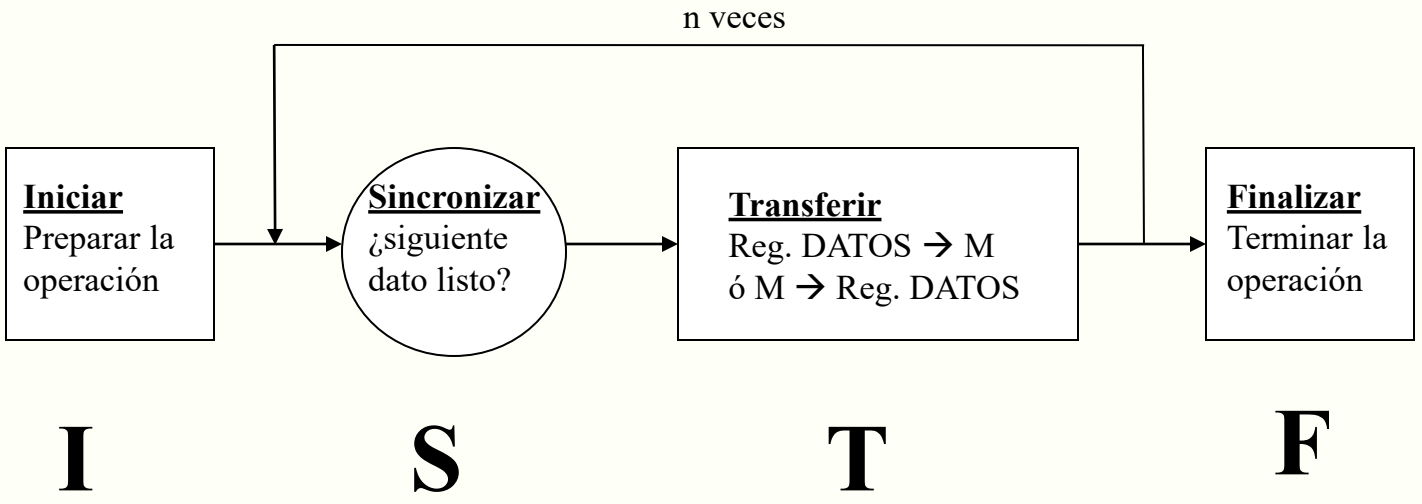
- Algunas arquitecturas no usan instrucciones específicas, usan LD y ST
- Son instrucciones privilegiadas (modo supervisor)

Índice

1. Periféricos
2. Introducción a E/S. Módulos de E/S. Instrucciones de E/S
3. **Técnicas de E/S**
   1. E/S programada
   2. E/S por interrupciones
   3. E/S por DMA

Operación de E/S

Operación de E/S: transferencia de un bloque de *n* datos entre la memoria y el periférico



Técnicas de E/S

Motivación:

*los periféricos son **lentos***

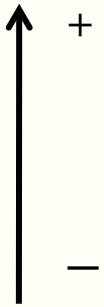
*+ operaciones de transferencia de **bloques** de datos*

Técnicas de E/S: grado de participación de la CPU en las operaciones de E/S

E/S **programada** o **directa**

E/S por **interrupciones**

E/S por **DMA**



Técnicas de E/S

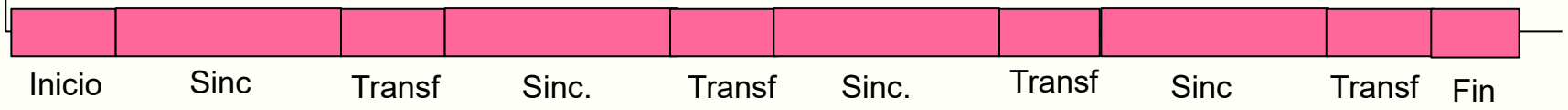
		Fase			
		I	**S**	**T**	**F**
Técnica	**Programada**	**X**	**X**	**X**	**X**
	Interrupciones	**X**		**X**	**X**
	DMA	**X**			**X**

Técnicas de Ent/Sal

Toperación

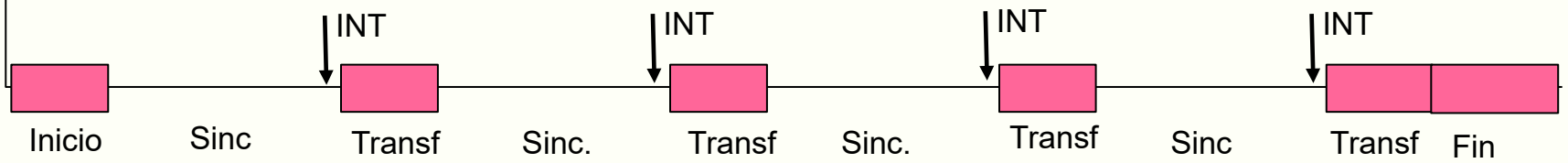
E/S Programada

Tcpu_ocup= Toperación



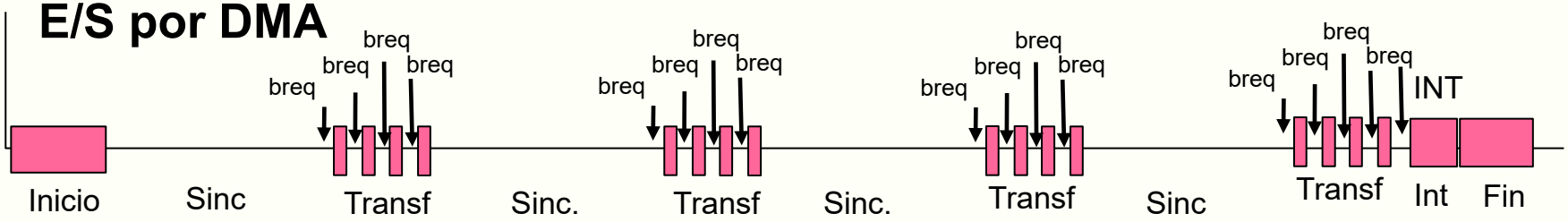
E/S por interrupciones

$Tcpu\_ocup = Tini + N_{INT} \cdot (sri + RTI) + Tfin$



E/S por DMA

$Tcpu\_ocup = Tini + N_{DMA} \cdot (t_{DMA}) + Tint\_fin$



$$Tcpu\_ocup = \sum$$

Índice

1. Periféricos
2. Introducción a E/S. Módulos de E/S. Instrucciones de E/S
3. Técnicas de E/S
    1. **E/S programada**
    2. E/S por interrupciones
    3. E/S por DMA

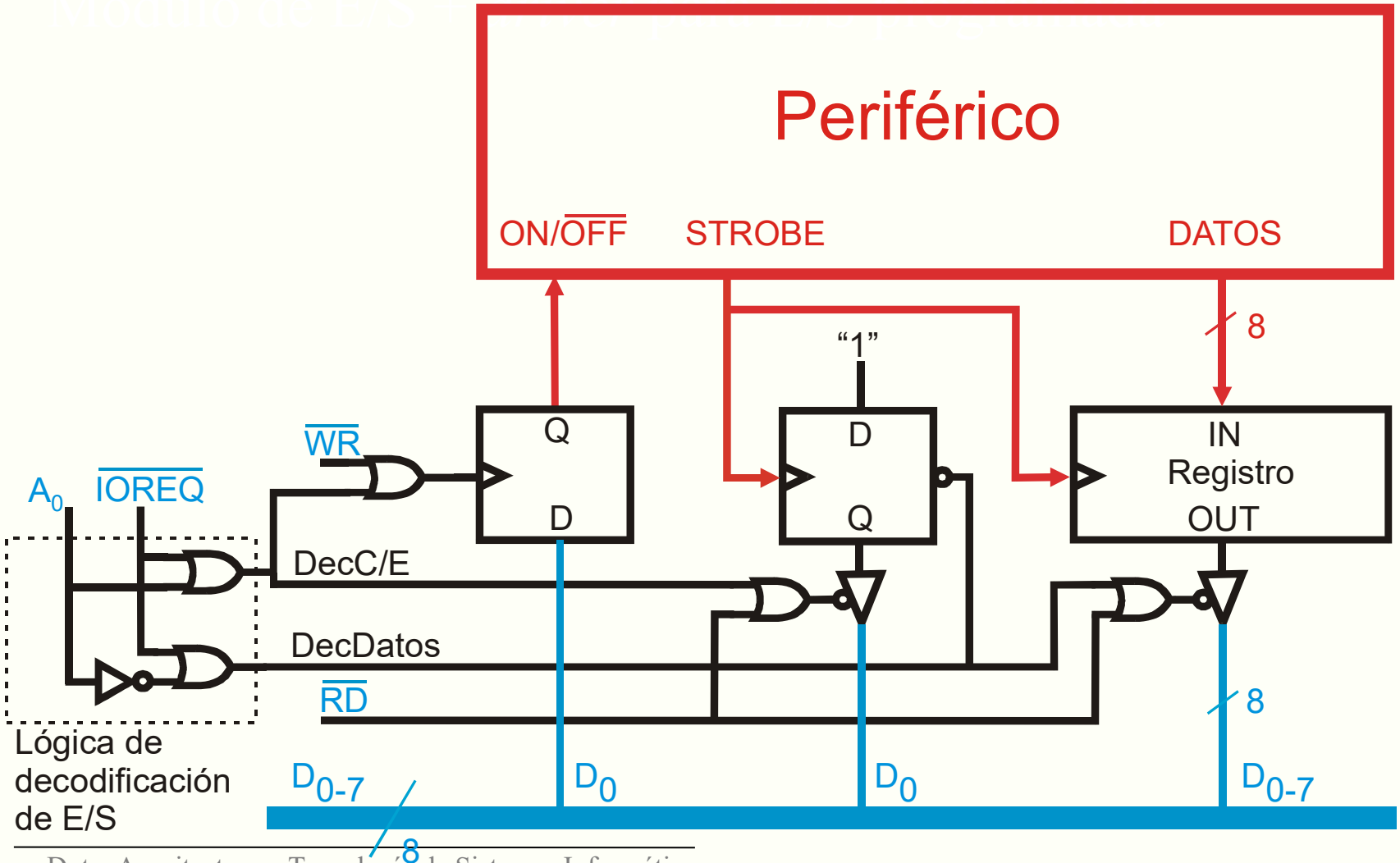
E/S programada o directa

- Todas las fases las realiza la CPU

- La sincronización se realiza mediante instrucciones en un bucle de espera

- Ejemplo con un periférico muy simple, programando su *driver*. Una vez que se enciende empieza a transmitir datos hasta que se le da la orden de parar

E/S programada o directa



Ej. de E/S programada o directa

Parámetros

- Tamaño del bloque: *n* palabras
- Dir. de almacenamiento en memoria: *dir_alm_M*

Direcciones de E/S

- Reg. de Control/Reg. de Estado: *Dir_C_E*
- Reg. de Datos: *Dir_Datos*

Mandatos (o 'comandos')

- Activar: *ON* [xxxxxxx1]
- Desactivar: *OFF* [xxxxxxx0]

Estado

- Nuevo dato listo: *LISTO* [00000001]

E/S programada o directa

```
        LD   .R1, #dir_alm_M
        LD   .R2, #n
        LD   .R0, #H'01 ; ON
        OUT  .R0, /Dir_C_E
```

I

```
sig:    IN   .R0, /Dir_C_E
        CMP  .R0, #LISTO
        BNZ   $sig
```

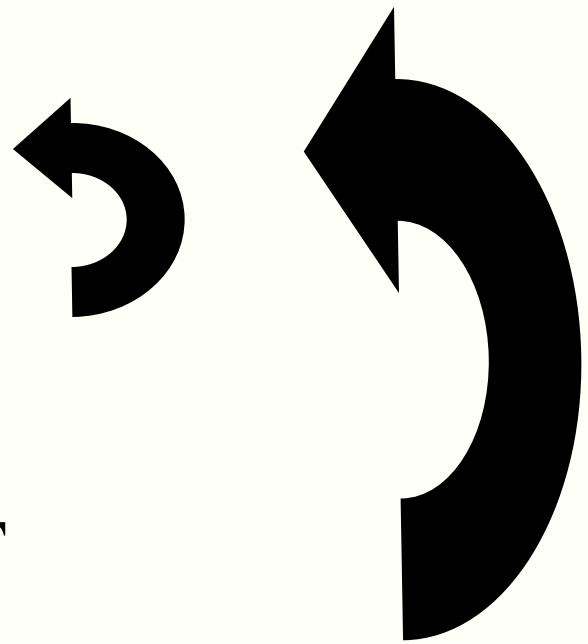
S

```
        IN   .R0, /Dir_Datos
        ST   .R0, [.R1++]
        DEC  .R2
        BNZ $sig
```

T

```
        LD   .R0, #H'00 ; OFF
        OUT  .R0, /Dir_C_E
```

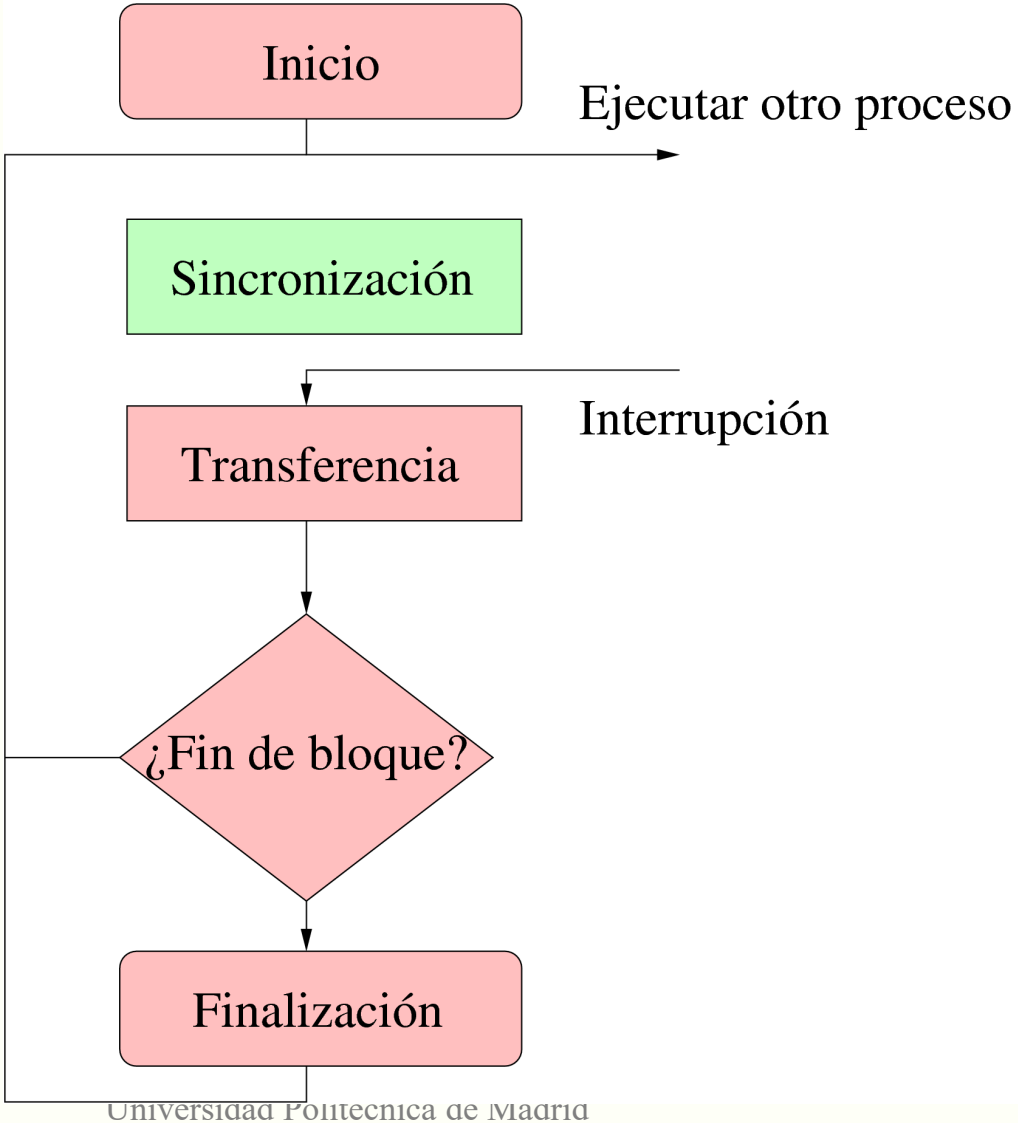
F



Índice

1. Periféricos
2. Introducción a E/S. Módulos de E/S. Instrucciones de E/S
3. Técnicas de E/S
    1. E/S programada
    2. **E/S por interrupciones**
    3. E/S por DMA

E/S por interrupciones



La CPU no se encarga de la sincronización

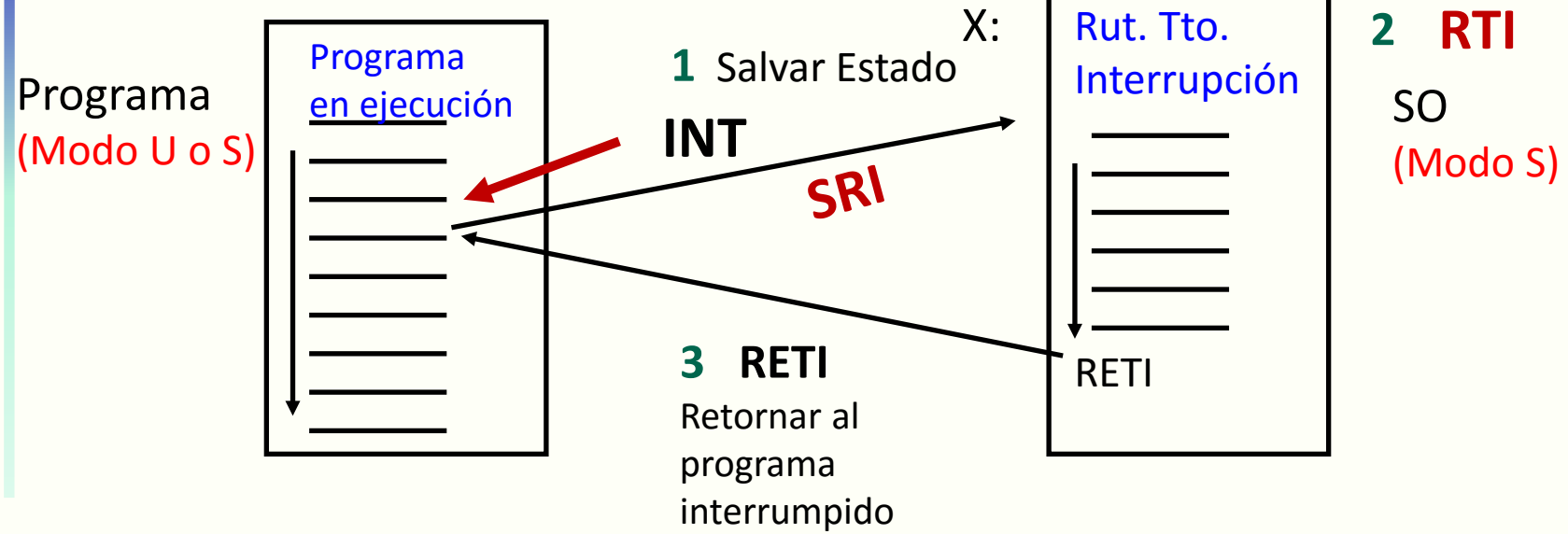
El módulo avisa a la CPU cuando está listo para una nueva transferencia

Se ahorra mucho tiempo de CPU que se usa para ejecutar otros programas

Interrupción

- Suceso **asíncrono** que hace que la CPU no ejecute la instrucción a la que "apunta" el PC
- La CPU pasa a ejecutar la llamada **rutina de servicio** de **interrupción** o **rutina de tratamiento de interrupción (RTI)**
- Al finalizar la RTI, la CPU debe continuar la ejecución del programa interrumpido
- Este mecanismo no puede afectar al comportamiento lógico de los programas cuya ejecución se interrumpe.

Gestión de interrupciones



El sistema (**CPU + SO**) gestiona o trata la interrupción: en general:

SRI

- Guarda la dirección de la instrucción a la que retornar: PC
- Guarda el estado del programa interrumpido: RE
- Pasa a **Modo Supervisor**
- Trata la interrupción: ejecución de la **rutina de tratamiento (**PC<-X)

RTI

- Restituye el estado del programa interrumpido
- Retorna el control al programa interrumpido, restituye PC y RE (**RETI**)

E/S por interrupciones

- Interrupciones
  - Interrupciones, excepciones y subrutinas
  - Solicitud de interrupciones
  - Servicio a peticiones de interrupción
  - Secuencia de reconocimiento de interrupciones (SRI) e instrucción de retorno
  - Rutina de servicio de interrupciones
  - Ejemplo de E/S por interrupciones
  - Sobrecoste de las interrupciones
  - Asignación de prioridades de interrupción

Interrupciones, excepciones y subrutinas

Suceso	Origen	Activación	Tratamiento
Subrutinas	Interno	Síncrona	Continuar
Excepciones	Interno	Síncrona	Continuar/Cancelar
Interrupciones	Externo	Asíncrona	Continuar

Interrupciones:

- El origen es el módulo de entrada/salida, que tiene una temporización propia, determinada por el periférico
- Pueden suceder en cualquier instante de la ejecución de una instrucción

Solicitud de interrupciones

- Mediante una nueva señal a la unidad de control, INT
- El módulo mantiene INT activa hasta que la interrupción es atendida
- La unidad de control secuencia un conjunto de operaciones elementales para servir o tratar la petición de interrupción.
  - Secuencia de reconocimiento de interrupciones (SRI)

Servicio a peticiones de interrupción

- Servicio a interrupciones
  - abandonar la ejecución del programa en curso
  - ejecutar otro programa que dé servicio a la solicitud del módulo de E/S
  - Posteriormente se ha de poder continuar con el programa interrumpido
- La UC muestrea la línea INT al finalizar la ejecución de la instrucción en curso
- La atención a interrupciones puede estar prohibida o inhibida:
  - Habitualmente mediante un bit del RE:
    - biestable de máscara de interrupción (BMI).

Biestable de máscara de interrupciones

- Hay dos programas que se ejecutan concurrentemente:
  - el programa interrumpido y
  - el que da servicio al módulo de E/S → condiciones de carrera
- No se puede impedir que el módulo solicite interrupciones, pero sí que la UC las "vea"
- Existen instrucciones (privilegiadas) para inhibir **DI *(Disable Interrupts)*** y habilitar **EI (*Enable Interrupts*)** la atención a las interrupciones

Secuencia de reconocimiento de Interrupciones (SRI)

- La realiza la UC, al final de cada instrucción, en caso de "ver" que la línea de petición de interrupción está activa:
  - Se hace antes de la secuencia de ***fetch***
- Ha de **salvar el estado** necesario para posteriormente reanudar el programa interrumpido
- La **siguiente instrucción** que se ejecute será la **primera** de la rutina de servicio de interrupciones, **RTI**
- ¿De dónde se obtiene la **dirección** de la RTI (**DRTI**)?
  - **Única.**
  - Vectorización

Secuencia de reconocimiento de Int. (SRI)

```
FETCH: Si INT ∧ RE.BMI entonces:
          Salvar PC
          Salvar RE
          RE.BMI ← 1 (Inhibir interrupciones)
          RE.S ← 1 (Cambiar a modo privilegiado)
          PC ← DRTI (Dir. de la rutina de tratamiento de int.)
          ir a fetch
       Si no
          ir a fetch
```

- El PC y el RE se salvan en la **pila** por sencillez
- Además se "**inhibe**" (prohíbe) las **atención** a las Interrupciones (para no reconocer eternamente la misma)
- Se pasa a **modo supervisor** para poder ejecutar instrucciones de entrada/salida (dentro del manejador del sistema operativo).

Instrucción de retorno de RTI

```
RETI: Restaurar RE
      Restaurar PC
      ir a FETCH
```

- Implícitamente **habilita** las interrupciones
- Esta instrucción deshace lo hecho en la SRI:
  - **restaura** el RE y el PC
- Ejemplos
  - IRET (familia x86)
  - RETT (SPARC)
  - RTE (88110)

Rutina de servicio de interrupciones

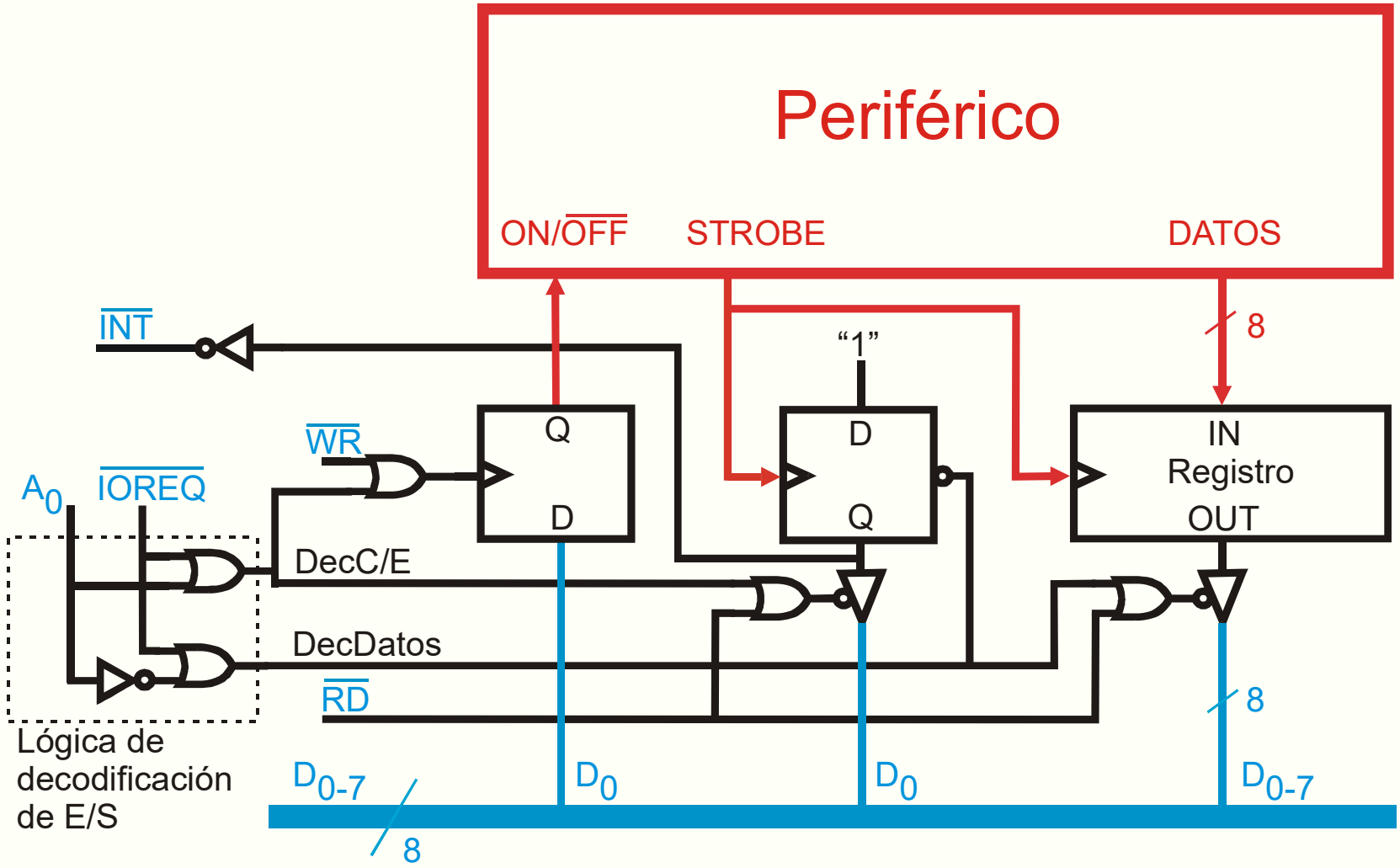
- No debe alterar la lógica del programa interrumpido:

  debe **salvar y restaurar** el estado que altere
- Debe hacer que el módulo de E/S desactive INT
- Como se ejecuta debido a un suceso asíncrono y externo:

  no se le pueden pasar parámetros ni en la pila ni en registros sino en direcciones conocidas de memoria (variables globales).
- Finaliza con una instrucción `RETI`

E/S por interrupciones

Ejemplo con un periférico simple


Periférico
ON/OFF
STROBE
DATOS
INT
"1"
8
WR
Q
D
D
Q
IN
Registro
OUT
A0
IOREQ
DecC/E
DecDatos
RD
8
Lógica de
decodificación
de E/S
D0-7
D0
D0
D0-7
8

Iniciación

```
LD   .R1, #dir_alm_M

LD   .R2, #n

LD   .R0, #H'01 ; ON
OUT  .R0, /Dir_C_E
BR   /Planificador
```

- R1 y R2 serán modificados por el planificador o por el programa a que éste dé paso…

Iniciación

```
LD   .R1, #dir_alm_M
ST   .R1, /Dir_dir_alm_M
LD   .R2, #n
ST   .R2, /Dir_contador
LD   .R0, #H'01 ; ON
OUT  .R0, /Dir_C_E
BR   /Planificador
```

- `Dir_dir_alm_M` y `Dir_contador` contienen la dirección de memoria donde se almacenará el siguiente dato y el número de datos que quedan para completar la operación, parámetros de la RTI (se pasan por variables globales)

Rutina de servicio de interrupciones

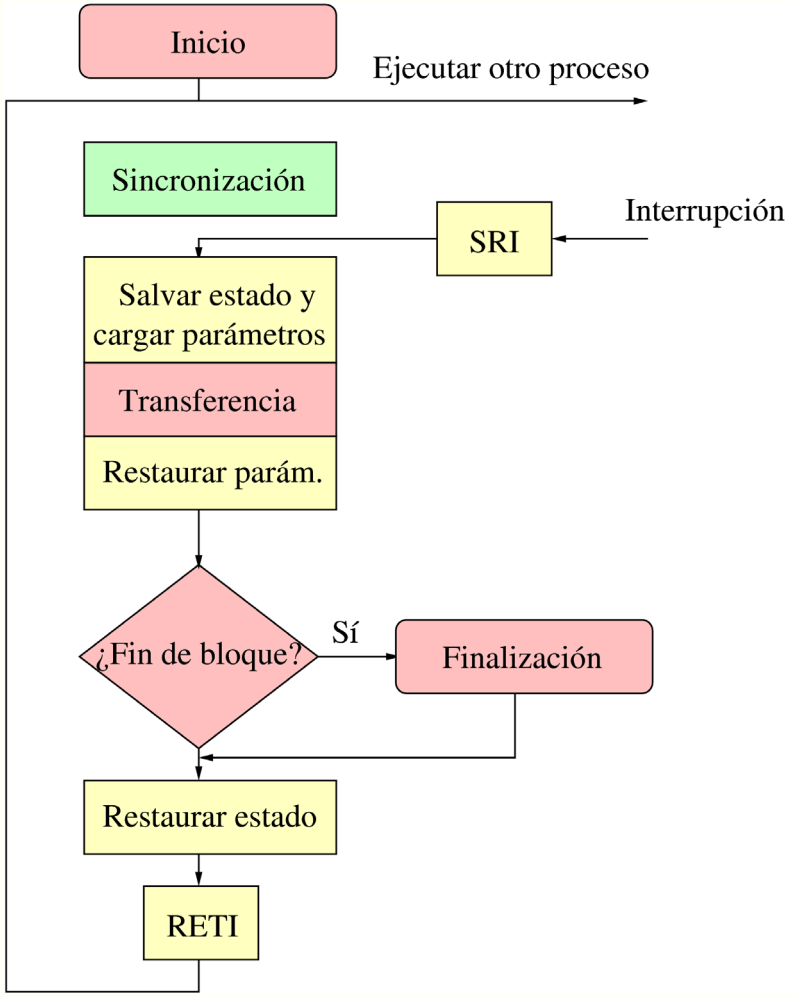
```
                ; en negro sobrecoste de la RTI
PUSH   .R0
PUSH   .R1
PUSH   .R2
LD     .R1, /Dir_dir_alm_M
LD     .R2, /Dir_contador
IN     .R0, /Dir_Datos ;
   Transf.
ST     .R0, [.R1++] ; Transf.
DEC    .R2
CALLZ  $Fin
ST     .R1, /Dir_dir_alm_M
ST     .R2, /Dir_contador
POP    .R2
POP    .R1
POP    .R0
RETI
```

```
Fin: LD   .R0, #H'00 ; OFF
     OUT  .R0, /Dir_C_E
     RET
```

Sobrecoste (*overhead*) de las interrupciones

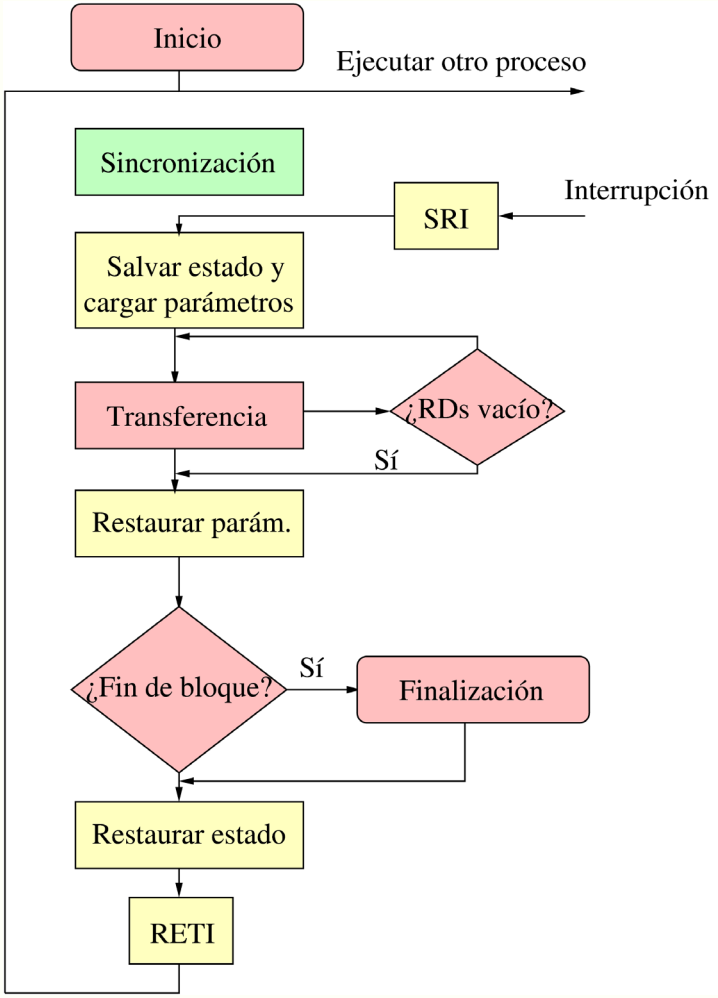
- Secuencia de reconocimiento de interrupciones
- Salvar/restaurar registros
- Paso de parámetros y su posterior actualización
- RETI

Análisis cuantitativo. *Sobrecoste*



- Se evita la sincronización pero se realizan otras operaciones para llevar a cabo la transferencia.
- El total supone mucho menos tiempo de CPU que por programa pero aún existe una sobrecarga inevitable.
- Para minimizar el impacto de esta sobrecarga se puede aumentar el tamaño del registro de datos.

Aumento del tamaño del *buffer* de datos



- Se dota al módulo de entrada/salida de un *buffer* de registros de datos
- Solicita la interrupción cuando el *buffer* está lleno (operación de entrada) o vacío (operación de salida)
- El número de interrupciones por operación se reduce y la sobrecarga por cada dato es mucho menor en función del tamaño del *buffer*

Asignación de prioridades

- El método óptimo consiste en asignar **mayor prioridad al dispositivo que pide interrupciones con mayor frecuencia.**
- La frecuencia depende de la velocidad de transferencia y del tamaño del registro de datos.
- Algunos dispositivos excepcionalmente no siguen esta regla:
  - Consola de operación
  - Temporizadores programables
  - DMA

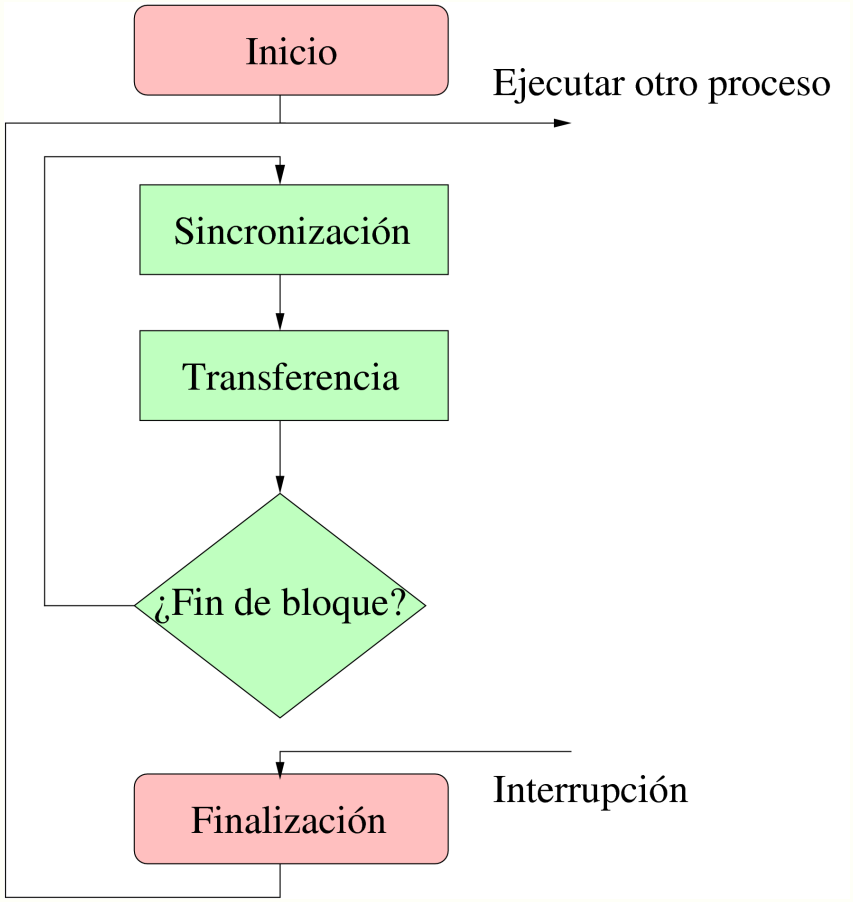
Índice

1. Periféricos
2. Introducción a E/S. Módulos de E/S. Instrucciones de E/S
3. Técnicas de E/S
    1. E/S programada
    2. E/S por interrupciones
    3. **E/S por DMA**

Índice

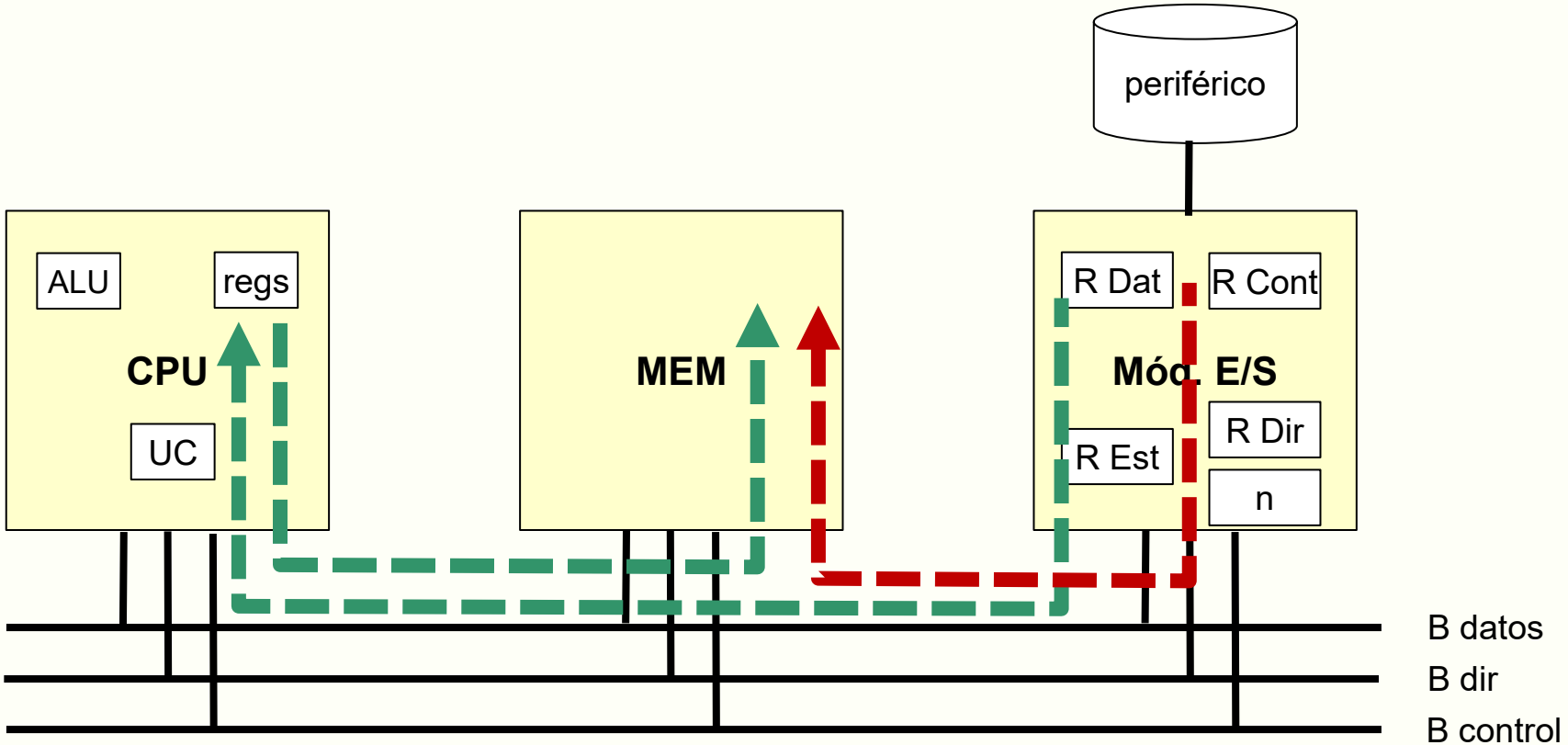
- Entrada/salida mediante acceso directo a memoria *(DMA)*
  - Robo de ciclo **aislado**
  - Robo de ciclo en **ráfagas**
  - Módulo de entrada/salida con *DMA*

E/S mediante DMA

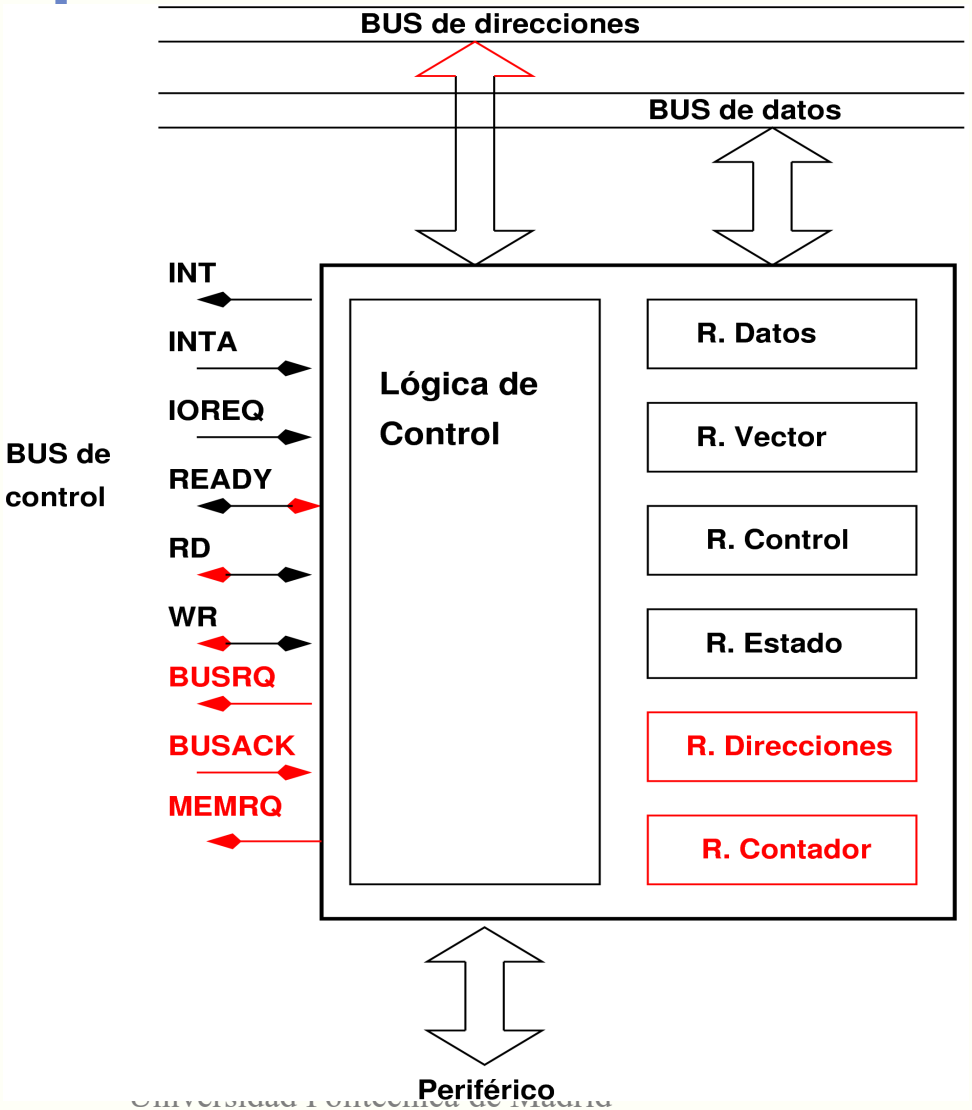


- La CPU se encarga de **iniciar** la operación.
- El módulo de entrada salida se encarga por Hw de la sincronización y transferencia y avisa cuando ha terminado mediante una **interrupción**.
- La CPU **finaliza** la operación.
- Hay una única interrupción por operación: se ahorra mucho tiempo de CPU con dispositivos de bloque.

Esquema básico del computador Von Neumann



Módulo E/S con DMA

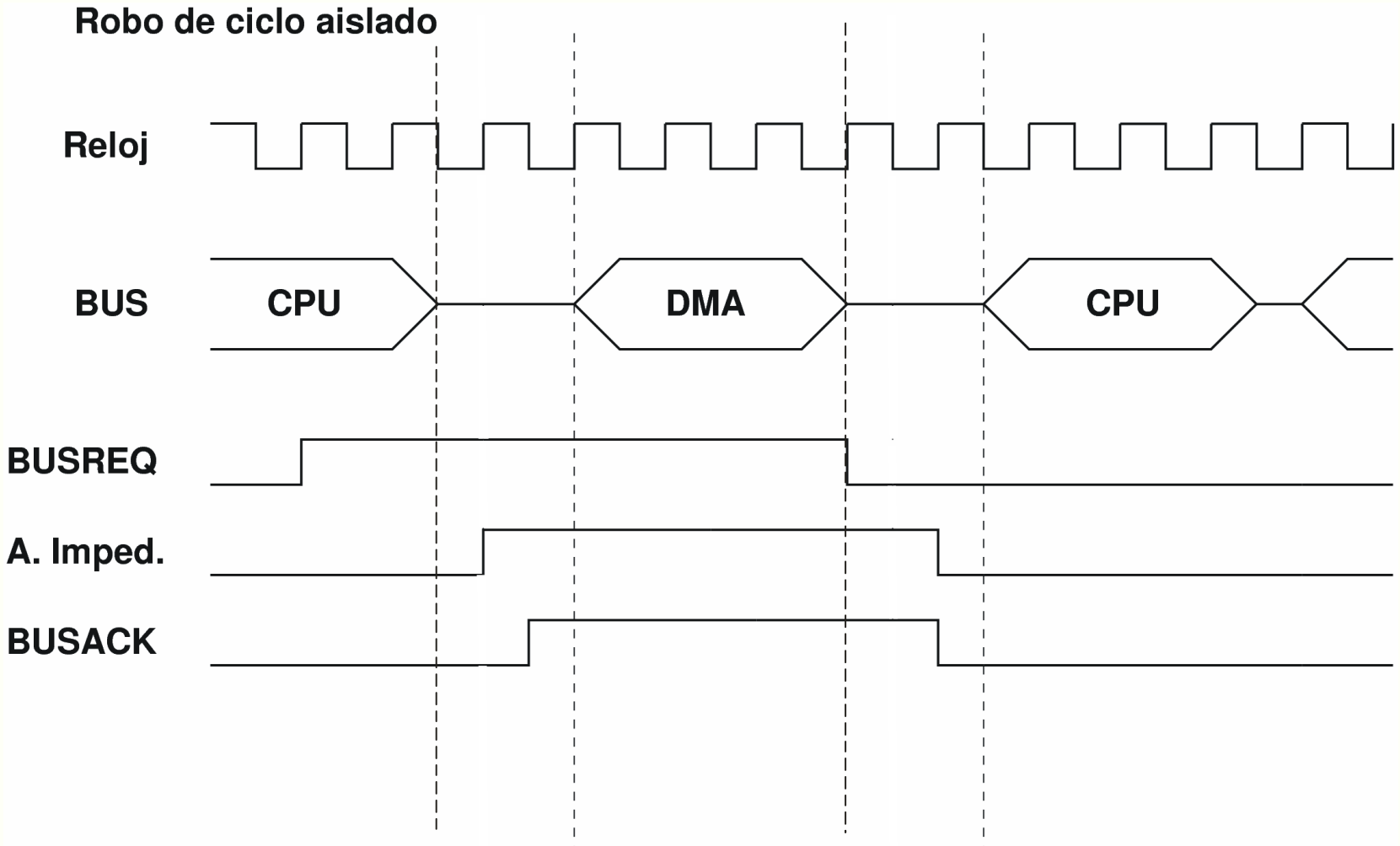


- El módulo es más complejo
- Debe generar las señales de dirección y control, para gobernar la memoria, antes las generaba la UC
- Nuevos **registros** para el **contador** y **Dir Memoria** (antes por Sw en la RTI)
- Necesita incrementador, Dir++, decrementador, cont--, y comparador

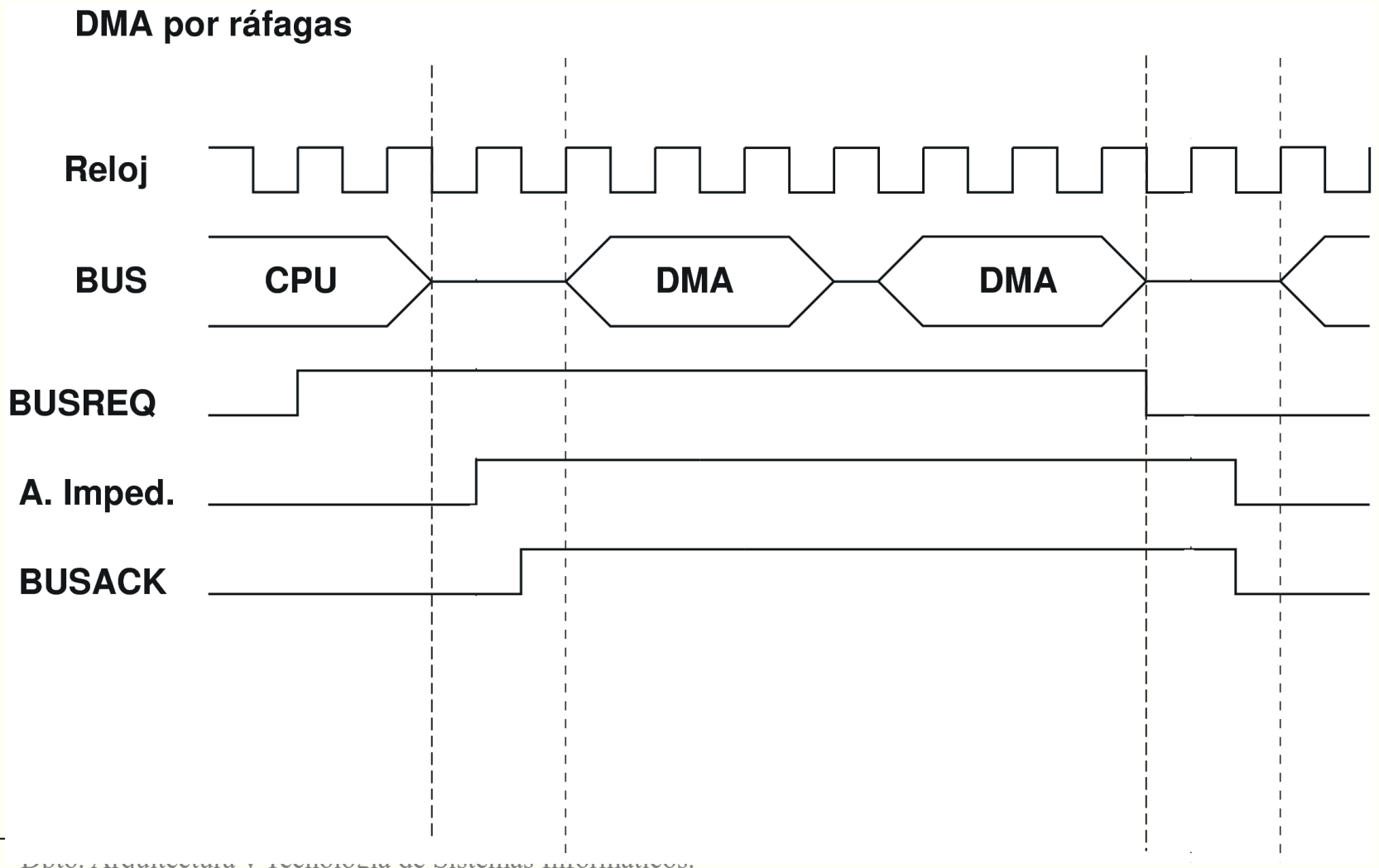
DMA mediante robo de ciclo

- El módulo, cuando hay datos listos, solicita los buses con BUSRQ.
- La CPU los cede al final del ciclo de bus en curso. Para ello, se “desconecta” de los buses colocando sus salidas en alta impedancia.
- La CPU indica al módulo que puede usarlos con BUSACK
- El módulo inicia el ciclo de bus para realizar la(s) transferencia(s) con memoria. Debe activar señales de dirección y control. Actualiza contador y Dir.
- Cuando acaba devuelve el bus desactivando BUSRQ.
- La CPU recupera los buses y desactiva BUSACK.
- Si no hay más datos a transferir (contador = 0) el módulo avisa a la CPU con una interrupción. La CPU finaliza la E/S

Robo de ciclo aislado (sin buffer)



Robo de ciclo en ráfagas (con buffer)



Operación por **DMA**

1.- **Programar** la operación de DMA en el periférico: instrucciones de salida (**OUT**)

1. Dir. Comienzo zona de Memoria: Dir. ⟶ DIR
2. Tamaño del bloque: Nº datos . ⟶ CONT
3. E ó S: E ó S ⟶ E/S

- A partir de ese momento la CPU se dedicará a "sus labores"

2.- Cuando el periférico está **listo** para la transferencia de un dato, **solicita los buses**:

Activa **$\overline{\text{BUSREQ}}$**

- La CPU "**cederá**" los **buses** cuando acabe la "fase" actual de la instrucción que está ejecutando:
  - "Congela" la ejecución de la instrucción (no ejecuta la siguiente Fase).
  - Pone en "alta impedancia" sus conexiones a:
    - Bus de Datos (**DB**)
    - Bus de Direcciones (**AB**)
    - Señales de control de Memoria (**$\overline{\text{RD}}$**, **$\overline{\text{WR}}$**, etc.)
  - Activa **$\overline{\text{BUSACK}}$**

3.- El periférico:

1. Transfiere el dato a/desde M activando las ss. de control y dirección pertinentes y usando el Bus de Datos.
2. CONT ⟵ CONT – 1
3. DIR ⟵ DIR + 1

4.- *Si* CONT ≠0 (* faltan más datos por transferir *)

entonces

Si perif. listo para transferir siguiente dato

entonces (*modo RÁFAGA*)

- Mantiene activa **BUSREQ**
- GO TO **3**

si no (*robo de ciclo aislado*)

- Desactiva **BUSREQ**
- la CPU:
  - desactiva **BUSACK**
  - continúa ejecutando Fase siguiente
- GO TO **2**

Si no (* CONT = 0: bloque de datos transferido *)

- Desactiva **BUSREQ**
- "avisa" a la CPU de que ha terminado la operación de DMA: solicita Interrupción:

Activa **INT**

Diferencias y similitudes con las interrupciones

- Ambos son eventos asíncronos pero los robos de ciclo no alteran el estado de la CPU.
  - Por lo tanto se puede conceder el BUS en “cualquier” momento.
- Un ciclo de bus no es reanudable y muy breve.
  - Por lo tanto no se permite anidamiento.
- Para ambos mecanismos existe una línea para petición y otra para concesión.
  - Aunque la respuesta del procesador es totalmente diferente.

Cuadro resumen técnicas de E/S