ESTRUCTURA DE COMPUTADORES

Tema 2. Entrada/salida

Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

ÍNDICE

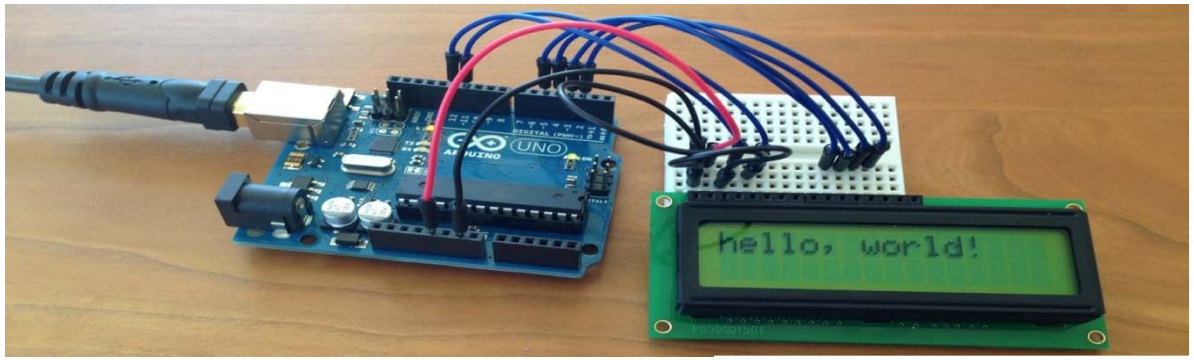
- Introducción
- Estructura y funciones del sistema de E/S
- E/S Programada — Práctica 1
- E/S por interrupciones — Práctica 2
- Controlador de interrupciones
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

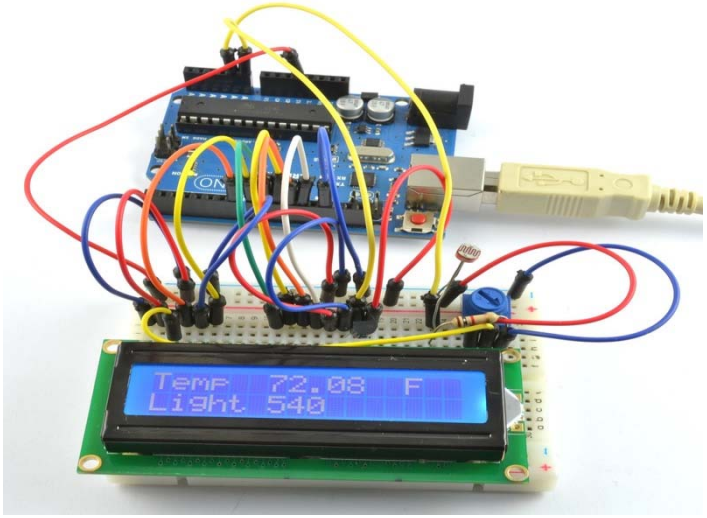
- Bibliografía:
  - W. Stallings; Computer Organization and Architecture, 10ed. Prentice Hall 2016. Cap 3 y 7
  - D. A. Patterson & J. L. Hennessy; Computer Organization and Design. The Hardware/Software Interface, 5ª ed. Morgan Kaufmann 2014.
  - Manuales del ARM7TDMI (procesador) y de la placa S3CEV40. (En el CV.)

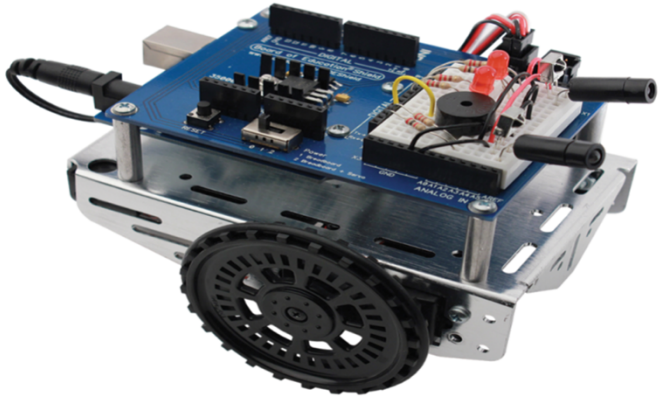
ÍNDICE

- **Introducción**
- Estructura y funciones del sistema de E/S
- E/S Programada
- E/S por interrupciones
- Controlador de interrupciones
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

CONEXIÓN CPU <-> MUNDO EXTERIOR

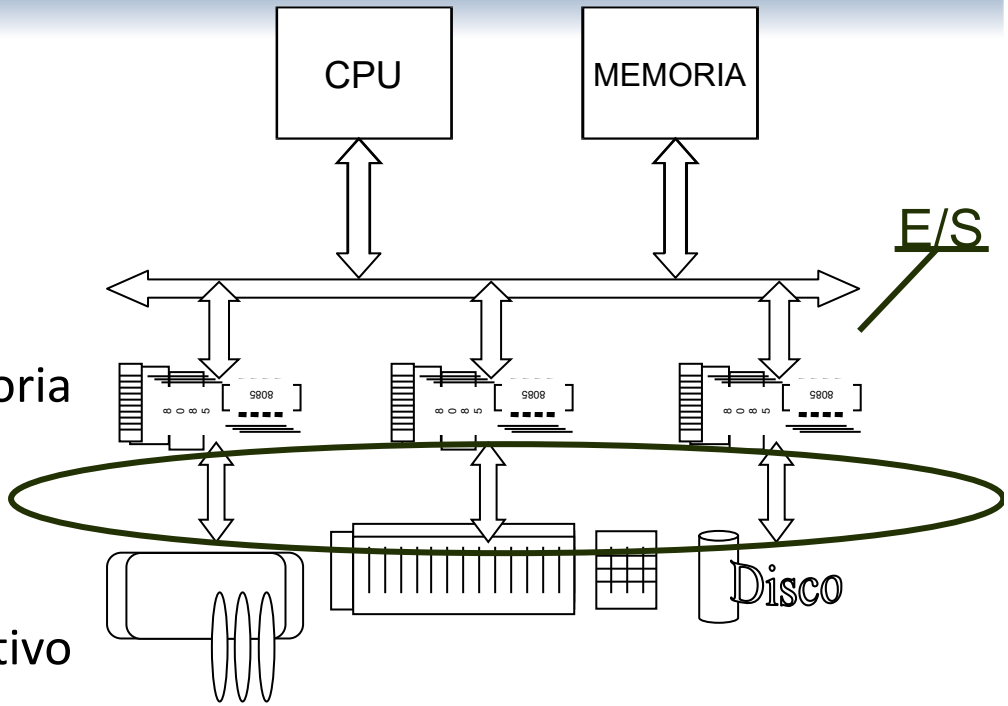




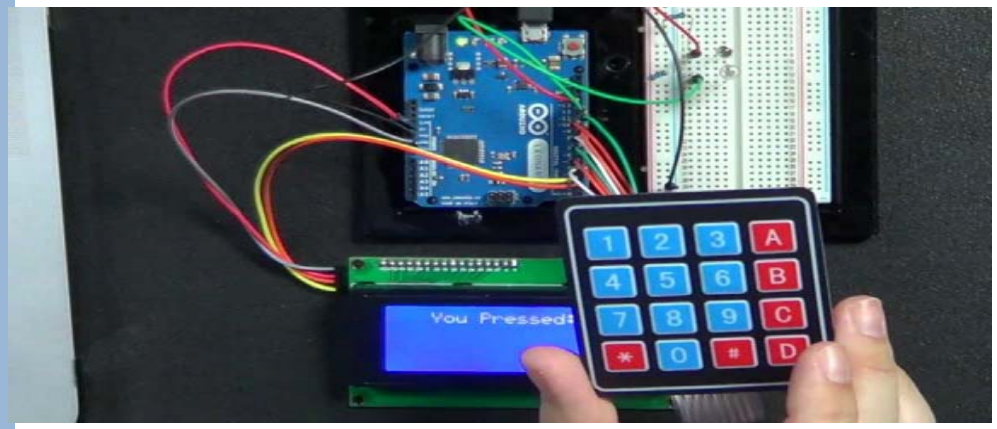


FUNCIONES DEL SISTEMA DE E/S

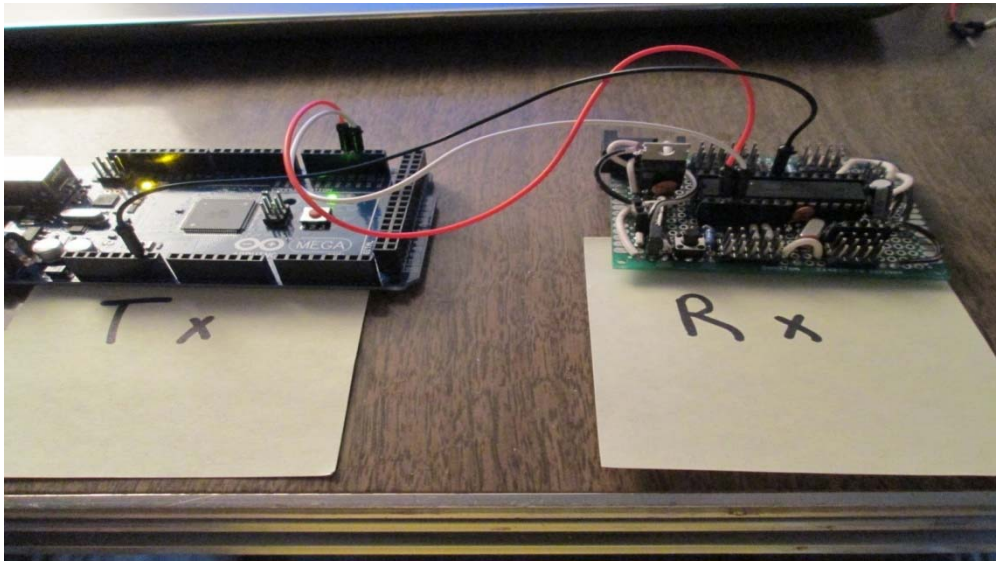
- Interacción CPU <-> mundo
- Entrada de datos
  - Desde dispositivo a CPU
  - Desde dispositivo a Memoria
- Salida de datos
  - Desde CPU a dispositivo
  - Desde Memoria a dispositivo
- En muchas ocasiones, el rendimiento del sistema de E/S determina el rendimiento global del sistema



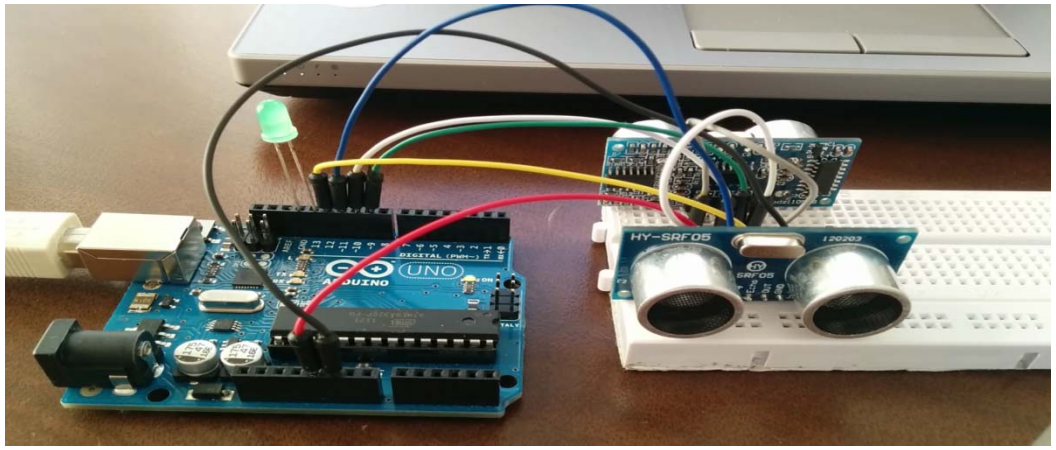
- Dispositivos de **presentación de datos**
  - Interaccionan con los usuarios, transportando datos entre estos y la máquina
  - Ratón, teclado, pantalla, impresora, etc



- Dispositivos de **comunicación** con otros procesadores
  - Permiten la comunicación con procesadores remotos a través de redes
  - Tarjeta de red, módem...



- Dispositivos de **adquisición de datos**
  - Permiten la comunicación con sensores y actuadores que operan de forma autónoma.
  - Se utilizan en sistemas de control automático de procesos por computador
  - Suelen incorporar conversores de señales A/D y D/A.



- Dispositivos de **almacenamiento** **de datos**
  - Forman parte de la jerarquía de memoria: interactúan de forma autónoma con la máquina
  - Discos magnéticos y cintas magnéticas…

DISTINTAS CARACTERÍSTICAS

- **Tipo de datos** que se transmiten
  - Bytes, bloques, tramas, etc
- **Tasa de transferencia**
  - **Ancho de banda:** cantidad de datos que se pueden transferir por unidad de tiempo (Mbit/s, MB/s…)
    - Dispositivos de almacenamiento/red -> gran ancho de banda
  - **Latencia:** tiempo requerido para obtener el primer dato (s, ms, µs, ns)
    - Generalmente alta en comparación con velocidad de la CPU

DISTINTAS CARACTERÍSTICAS

- Los dispositivos de E/S se pueden caracterizar según:
  - **Comportamiento**: entrada, salida, almacenamiento
  - **Destinatario**: humano o máquina
  - **Tasa de transferencia**: bytes/seg, transfers/seg

Dispositivo	Comportamiento	Destino	Transferencia de datos (KB/seg)
Teclado	Entrada	Humano	0,01
Ratón	Entrada	Humano	0,02
Impresora láser	Salida	Humano	100
Gráficos	Salida	Humano	100000
Red LAN	Comunicación	Máquina	10000
Disco óptico	Almacenamiento	Máquina	10000
Disco magnético	Almacenamiento	Máquina	30000

CONCLUSIONES

- Necesitamos
  - Un interfaz distinto para cada periférico
    - Controlador de E/S del dispositivo
  - Distintas formas de interaccionar con ellos
    - Técnicas de E/S

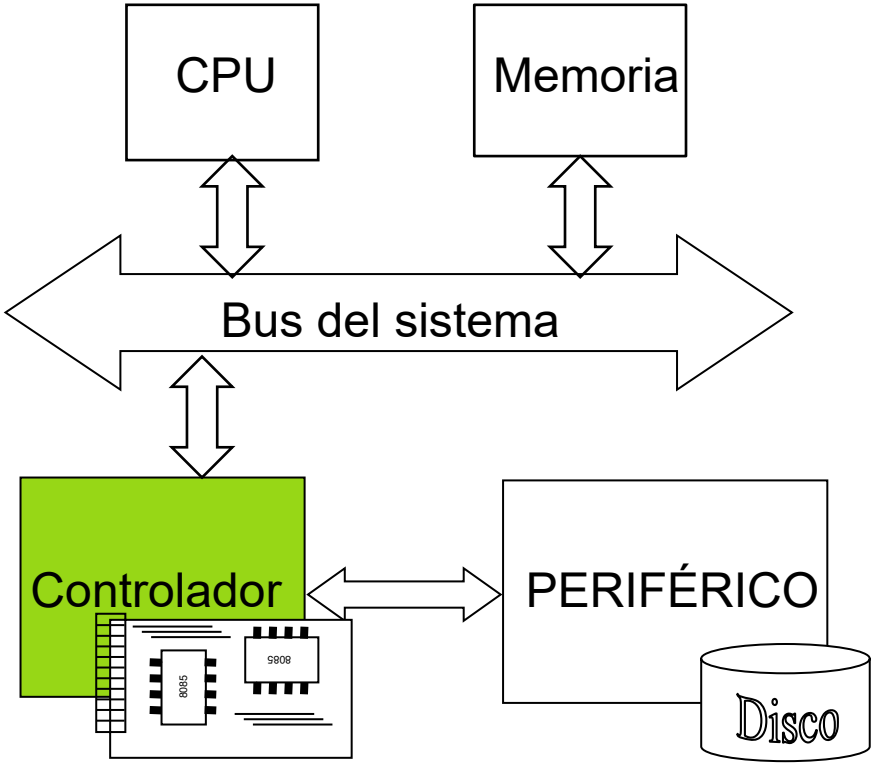
INDICE

- Introducción
- **Estructura y funciones del sistema de E/S**
- E/S Programada
- E/S por interrupciones
- Controlador de interrupciones
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

ESTRUCTURA DEL SISTEMA DE E/S

- Componentes:
  - Dispositivo periférico
  - Módulo de E/S o controlador (hardware)
    - Tarjeta
  - Controlador software
    - Driver S.O.
  - Canal de comunicación:
    - Bus

- Organización sencilla del sistema de E/S



COMPONENTES DEL SISTEMA DE E/S

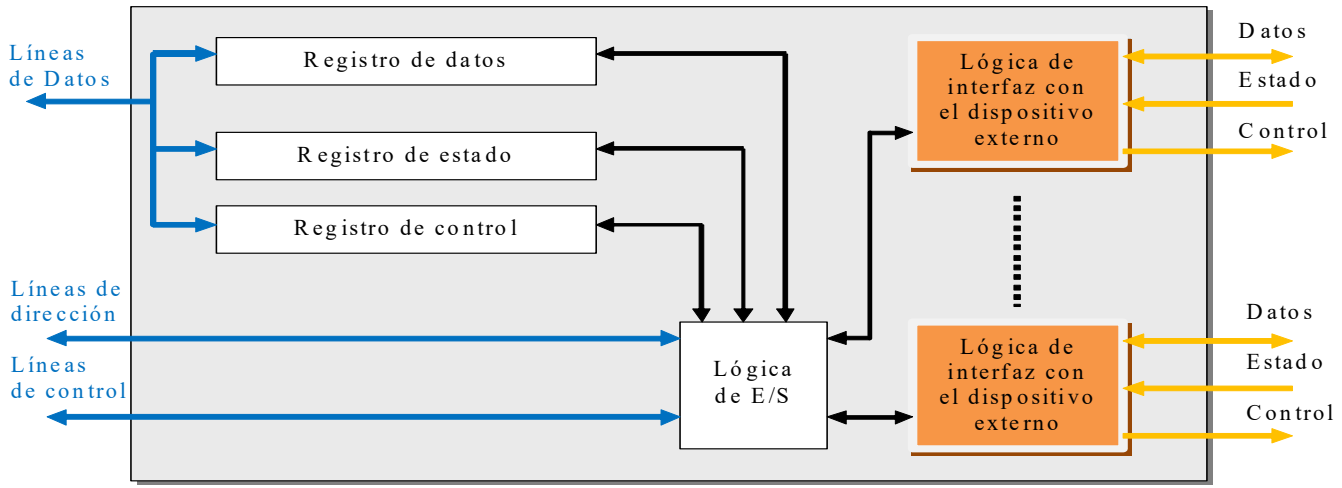
- **Dispositivo periférico**
  - Dispositivo en sí, de carácter mecánico, magnético…. (teclado, platos del disco..)
- **Controlador del dispositivo (hardware)**
  - Interfaz lógica que ofrece el dispositivo al sistema
  - Se *entiende* con el bus y conoce las características físicas del dispositivo
- **Controlador software (*driver*)**
  - Código encargado de programar el controlador HW del dispositivo concreto
  - Forma parte del sistema operativo (suele proporcionarlo el fabricante)
- **Canal de comunicación (p. ej. bus)**
  - Interconexión entre la CPU, memoria y el controlador HW del dispositivo
  - Puede ser compartido por varios dispositivos
    - Necesidad de arbitraje

CONTROLADOR DEL DISPOSITIVO

- Interfaz entre el dispositivo y el procesador
- Funciones:
  - Control y temporización
    - Adapta la velocidad de transferencia
  - Comunicación con el procesador
  - Comunicación con el periférico
  - Buffering o almacenamiento intermedio
  - Detección de errores

Tiene dos interfaces

ESTRUCTURA DE UN CONTROLADOR E/S



- **Registro de datos**: intercambio de datos entre CPU y periférico
- **Registro de estado:** estado del dispositivo, ej. preparado, error, ocupado…
- **Registro de control:** actuación sobre el dispositivo (imprime un carácter, lee un bloque…)

CUIDADO: NO son registros de la CPU. ¡¡¡Pueden estar en otro chip!!!

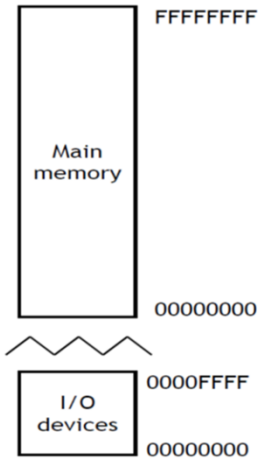
OPERACIÓN DE E/S

- Una operación de entrada/salida se divide en:
  - **Petición**: enviar el comando (y dirección)
    - La CPU indica al dispositivo la operación que quiere hacer escribiendo en su *registro de control*
      - **¿Cómo seleccionar el dispositivo de E/S deseado?** Depende de que sea E/S aislada o mapeada en memoria
  - **Envío**: transferencia de los datos
    - La CPU leerá/escribirá de/al *registro de datos* del controlador del periférico deseado
      - **¿Cómo saber cuándo? SINCRONIZACIÓN**

DIRECCIONAMIENTO DE CONTROLADORES

E/S aislada

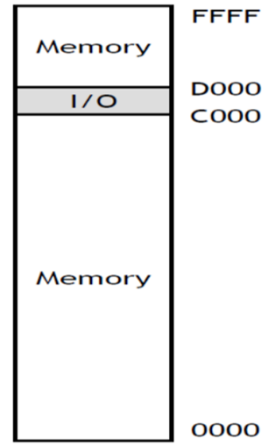
- Espacio de direcciones diferente al de la memoria principal (existen dos mapas de direcciones)
- Existen instrucciones específicas de E/S
  - **IN dir_E/S, Ri** **(CPU ← Periférico)**
  - **OUT Ri, dir_E/S** **(Periférico ← CPU)**
- Ejemplo: Intel x86
  - Cada registro de un módulo de E/S es un puerto



DIRECCIONAMIENTO DE CONTROLADORES

E/S localizada en memoria

- La memoria y los dispositivos de E/S comparten el espacio de direcciones
- Podemos usar instrucciones tipo load/store (ldr/str)
  - **LDR Ri, dir_E/S (CPU ← Periférico)**
  - **STR Ri, dir_E/S (Periférico ← CPU)**
- Ejemplo: ARM
  - Cada registro de un módulo de E/S es una dirección de memoria dentro de un rango reservado



EJEMPLO:

- Controlador GPIO (General Purpose Input/Output), gestiona pines multifunción del chip. **El GPIO** tiene un circuito electrónico que:
  - Permite dar al pin la tensión Vcc o GND en función de lo que se haya escrito en el registro de datos, sólo si en el registro de control se ha configurado como pin de salida
  - Permite leer el valor del pin
- Para encender un led conectado a uno de los pines:
  - Conocer las direcciones de memoria asignadas a sus registros (Puerto B):
    - Registro de control PCONB: 0x01D20008
    - Registro de datos PDATB: 0x01D2000C
  - Configurar los pines del controlador como salidas para poder escribir en los leds :
    - Escribir en el registro de control (PCONB) un '0'
  - Encender los leds :
    - Escribir en un registro de datos (PDATB)del GPIO el valor de la salida
      - 0 enciende el led,
      - 1 apaga el led

```
mov r0,#0
ldr r1,=PCONB    @0x01D20008
str  r0,[r1]
```

```
mov r0,#0
ldr r1,=PDATB    @0x01D2000C
str  r0,[r1]
```

SINCRONIZACIÓN

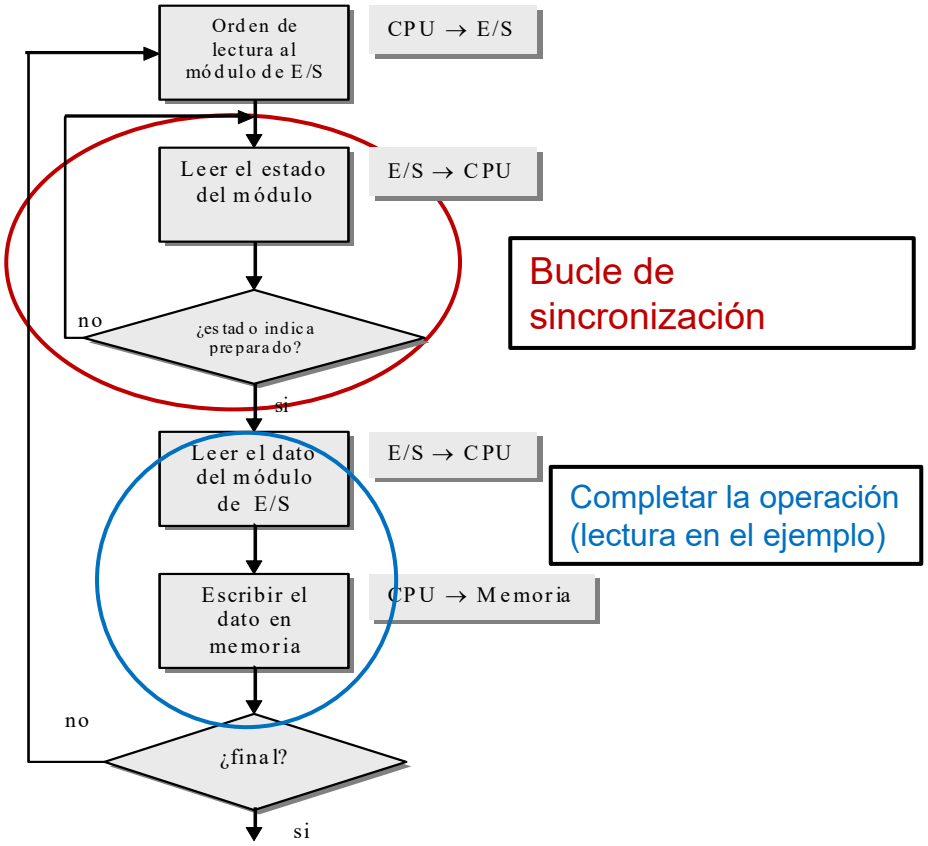
- Exigida por la diferencia de velocidad entre la CPU y los dispositivos de E/S
- Antes de enviar/recibir datos a/desde un periférico hay que asegurarse de que el dispositivo está preparado para realizar la transferencia
  - ¿Cómo sabe la CPU cuándo está lista/ha terminado la transferencia?
  - ¿Cómo sabe si todo ha ido bien?
- Existen dos mecanismos básicos de sincronización de la E/S
  - **E/S programada con espera de respuesta** (polling)
  - **E/S por interrupciones**

ÍNDICE

- Introducción
- Estructura y funciones del sistema de E/S
- **E/S Programada**
- E/S por interrupciones
- Controlador de Interrupciones
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

Operación de E/S Programada

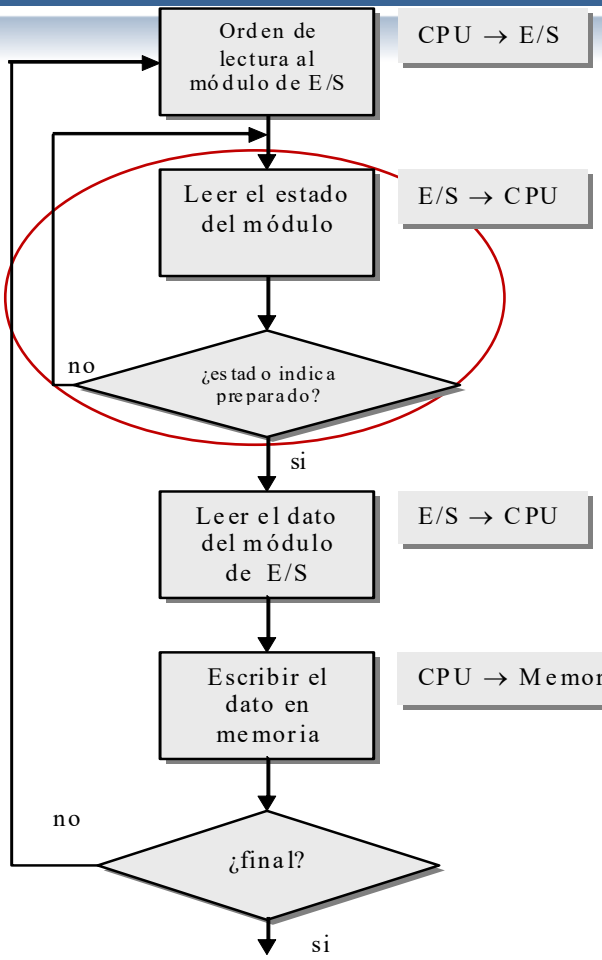
- Cada vez que la CPU quiere realizar una transferencia:
  - Lee una y otra vez el **registro de estado** del periférico (encuesta o "polling") hasta que esté preparado para realizar la transferencia
  - Realiza la transferencia



PROBLEMAS DE LA E/S PROGRAMADA

- La CPU no hace trabajo útil en el bucle de espera activa
  - Con dispositivos lentos el bucle podría repetirse miles/millones de veces
  - La dinámica del programa se detiene durante la operación de E/S
    - Ejemplo: imaginar que en un vídeojuego se detuviese la dinámica del juego a la espera de que el usuario pulse una tecla
- Dificultades para atender a varios periféricos
  - Mientras se espera a que un periférico esté listo para transmitir, no se puede atender a otro

EJEMPLO: LECTURA DE LOS PULSADORES



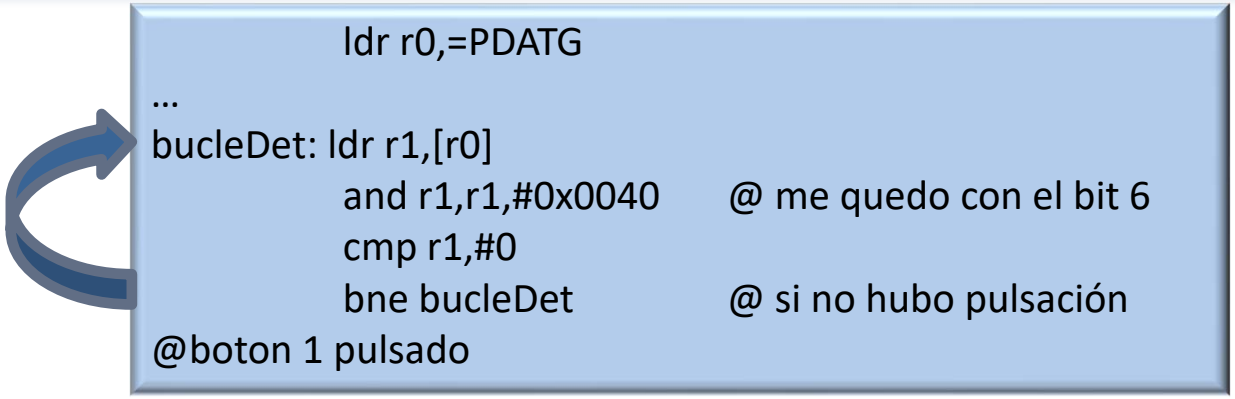
El bit 6 del registro PDATG es:
0 si el botón 1 ha sido pulsado y
1 si no ha sido pulsado

pseudocódigo

```
int  reg
reg = rPDATG
If (( bit 6 de reg) == 0)
                pulsado=SI
else
                pulsado=NO
```

```
                ldr r0,=PDATG
...
bucleDet:   ldr r1,[r0]
                and r1,r1,#0x0040       @ me quedo con el bit 6
                cmp r1,#0
                bne bucleDet            @ si no, hubo pulsación
@boton 1 pulsado
```

EVALUACIÓN DEL EJEMPLO

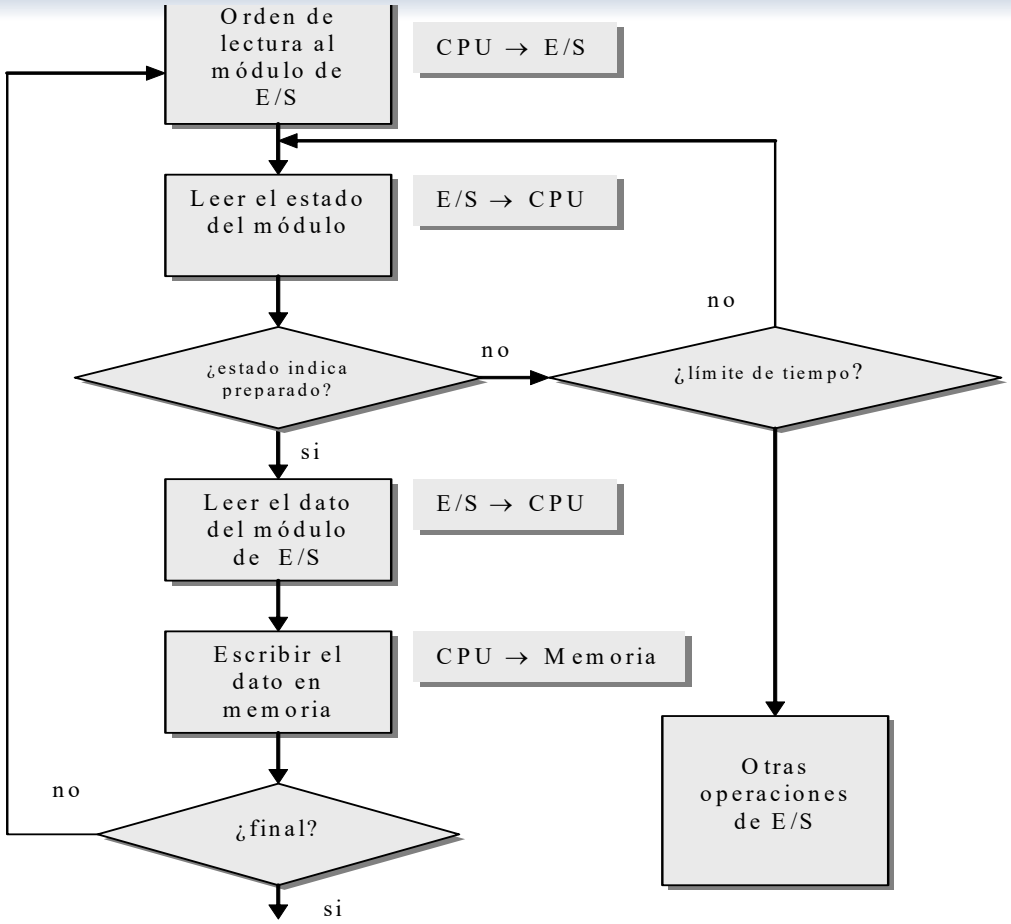


```
                ldr r0,=PDATG
…
bucleDet: ldr r1,[r0]
                and r1,r1,#0x0040      @ me quedo con el bit 6
                cmp r1,#0
                bne bucleDet           @ si no hubo pulsación
@boton 1 pulsado
```

Ejercicio: ¿cuántas veces se ejecuta el bucle suponiendo que soy capaz de pulsar el botón 1 vez por segundo, que el procesador tiene una frecuencia de reloj de 40MHz y cada instrucción tarda 3 ciclos en ejecutarse?

40 x 1000000 ciclos/s / 12 ciclos/iteración = 3,3 millones de iteraciones/s

SOLUCIÓN PARCIAL



Fijar un límite de tiempo en el bucle de espera

¿SE PUEDE MEJORAR?

- **El bucle de sincronización ejecuta instrucciones inútiles**
- ¿Y si el **dispositivo avisase a la CPU cuando haya terminado** su operación?
  - La CPU podría hacer trabajo útil mientras el periférico hace la operación solicitada
  - Sólo tendría que retomar el control de la operación cuando el periférico hubiese terminado
- Este mecanismo se denomina **E/S por interrupciones**

ÍNDICE

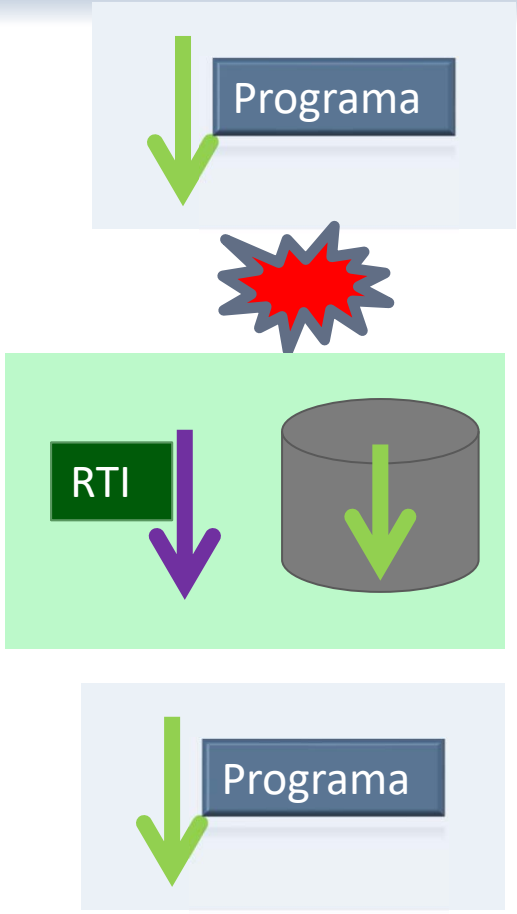
- Introducción
- Estructura y funciones del sistema de E/S
- E/S Programada
- **E/S por interrupciones**
  - **Concepto de excepción. Gestión de excepciones.**
  - **Operación de E/S mediante interrupciones**
- Controlador de interrupciones
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

EXCEPCIÓN

- **Evento inesperado que provoca un cambio en el flujo de ejecución normal del programa.**
- **Excepciones hardware**
  - **Internas:** producidas por la CPU (división por cero, desbordamiento, instrucción ilegal, dirección ilegal, raíz cuadrada de negativos, etc.)
  - **Externas:** producidas por los dispositivos de E/S (**Interrupciones**)
- **Excepciones software** (trap, swi):
  - Producidas por la ejecución de instrucciones especiales
  - Usadas por el SO para ofrecer servicios

EXCEPCIÓN

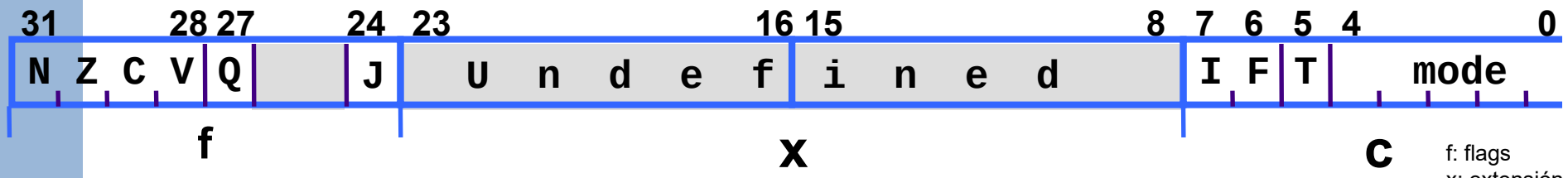
- **Flujo de control cuando se produce una excepción**
  1) La CPU **interrumpe la ejecución** de instrucciones
  2) **Guarda información** para poder retomar la ejecución correctamente más tarde
  3) Cambia a un ***estado especial*** para tratar la excepción
  4) Pasa a ejecutar una **Rutina de Tratamiento de Interrupción/Excepción** (RTI o *ISR*)
  5) El retorno de la RTI **retomaría la ejecución del programa original (o no)**



EXCEPCIONES Y MODOS DE EJECUCIÓN

- Todos los procesadores tienen, al menos, dos modos de ejecución:
  - **Usuario**: permisos limitados
  - **Privilegiado**: acceso a todos los recursos.

Registro de estado



- ¿Cómo cambiar de modo?
  - De modo privilegiado a modo usuario: escribiendo en el registro de estado
  - De modo usuario a privilegiado: cuando se produce una excepción

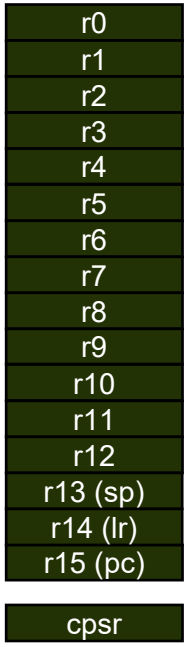
ARM: MODOS DE EJECUCIÓN

- **User (usr):** estado normal de ejecución
- **System :** modo privilegiado para el sistema operativo, usando los mismos registros que en el modo usuario
- **FIQ :** manejo de interrupciones rápidas para transferencias de datos
- **IRQ :** manejo de interrupciones de propósito general o lentas
- **Supervisor (svc):** modo protegido para el sistema operativo
- **Abort (abt):** usado para gestionar las fallos de acceso a datos e instrucciones (prefetching o accesos convencionales)
- **Undef (und):** manejo de fallos por instrucción no definidas

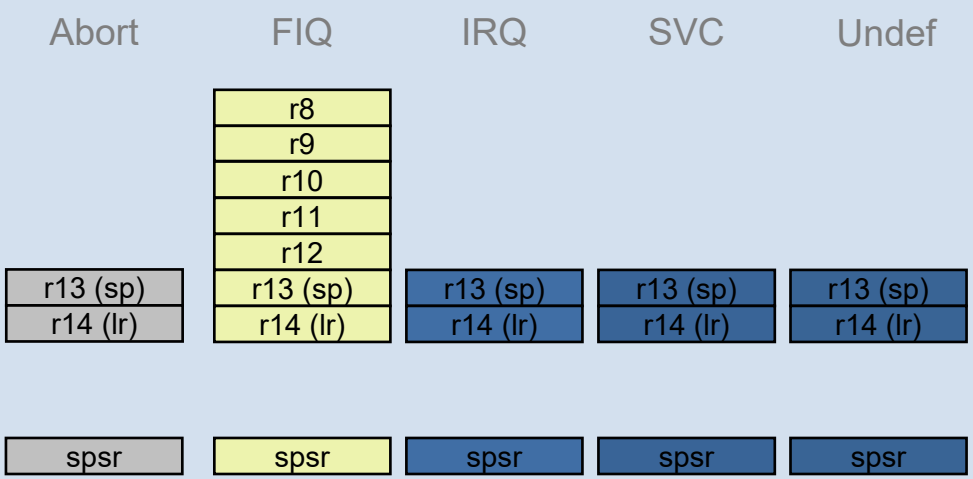
ARM: REGISTROS Y MODOS DE EJECUCIÓN

Registros visibles

User Mode (System)



Banked out Registers

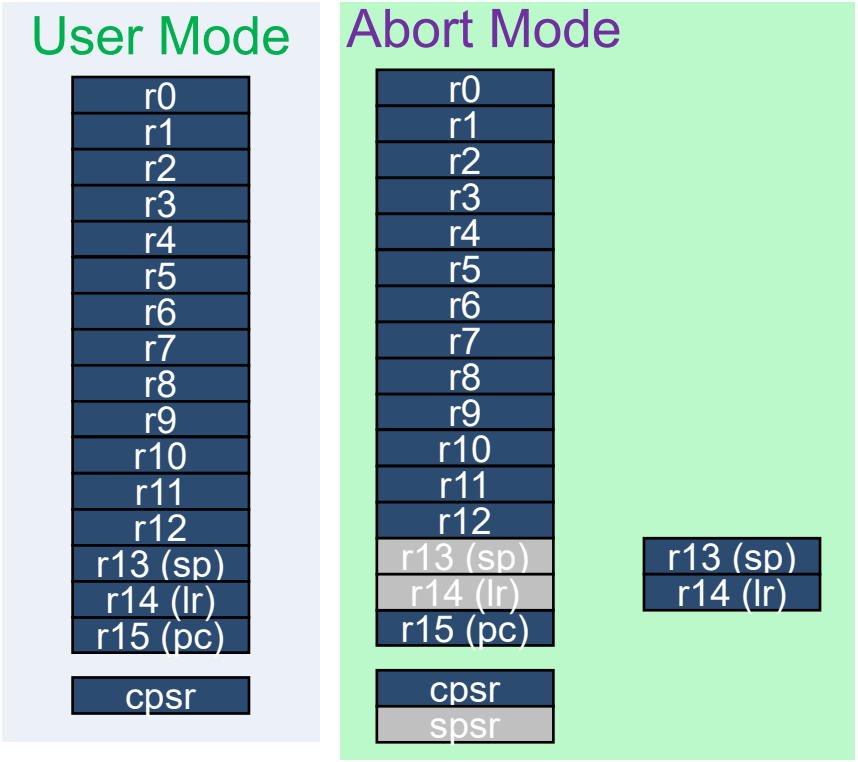


GESTIÓN DE EXCEPCIONES EN EL ARM

- **Cuando se produce una excepción:**

1) La CPU **interrumpe la ejecución** de instrucciones
2) La CPU cambia al **modo de ejecución asociado a** la excepción (p.e. Abort)
3) **Guarda información** para poder retomar la ejecución correctamente más tarde:
   1) **CPSR en SPSR_abort**
   2) **Dirección de retorno en LR_abort**
   3) **Resto de registros en pila (diferentes pilas para diferentes modos de ejecución)**

Inicializar todos los punteros de pila al arrancar el sistema.



GESTIÓN DE EXCEPCIONES EN EL ARM

- **Cuando se produce una excepción:**

4) La CPU pasa a ejecutar la **Rutina de Tratamiento de Excepción** (*ISR*)

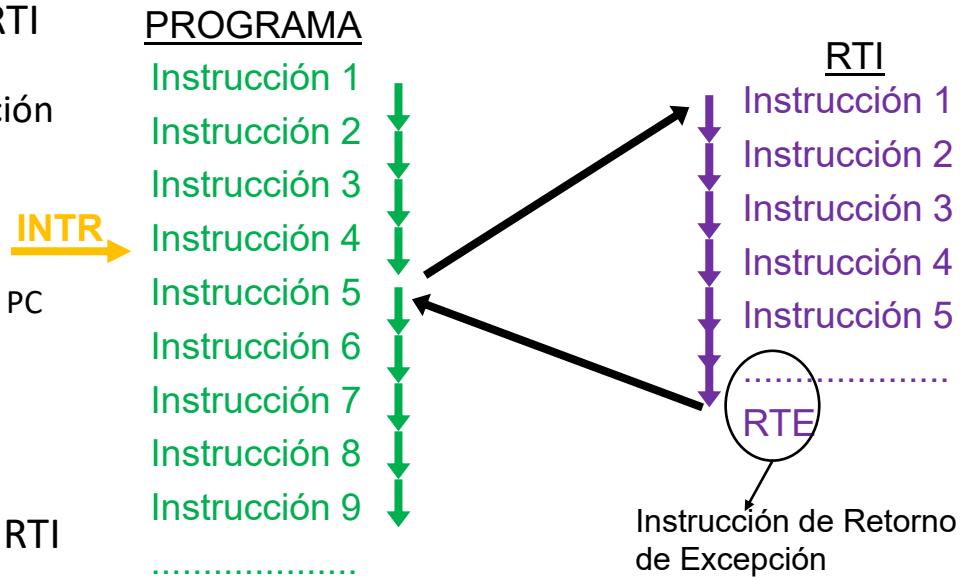
   En una posición fija de memoria denominada Vector de excepción.

5) El retorno de la RTI **retomaría la ejecución del programa original (o no)**

Excepción	Vector
Reset	0x00
Data Abort	0x10
FIQ	0x1C
IRQ	0x18
Prefetch Abort	0x0C
Undef	0x04
SWI	0x08

SUBRUTINAS VS RTI

- **Analogías** entre una subrutina y una RTI
  - Se rompe la secuencia normal de ejecución
  - Cuando terminan de ejecutarse se debe retornar al punto de ruptura
    - Por eso el HW guarda automáticamente el PC

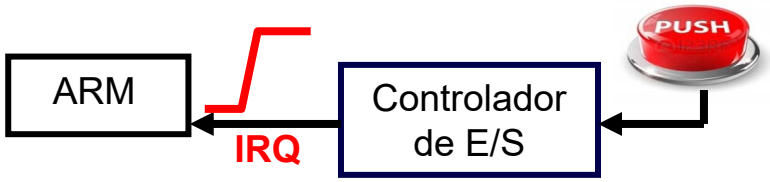


- **Diferencias** entre una subrutina y una RTI
  - En una subrutina el programador sabe en qué punto exacto se rompe la secuencia
  - Una RTI puede ejecutarse en cualquier momento, sin control del programador
  - Necesario guardar el registro de estado y todos los registros que usa la RTI y restaurarlos al retornar de la RTI

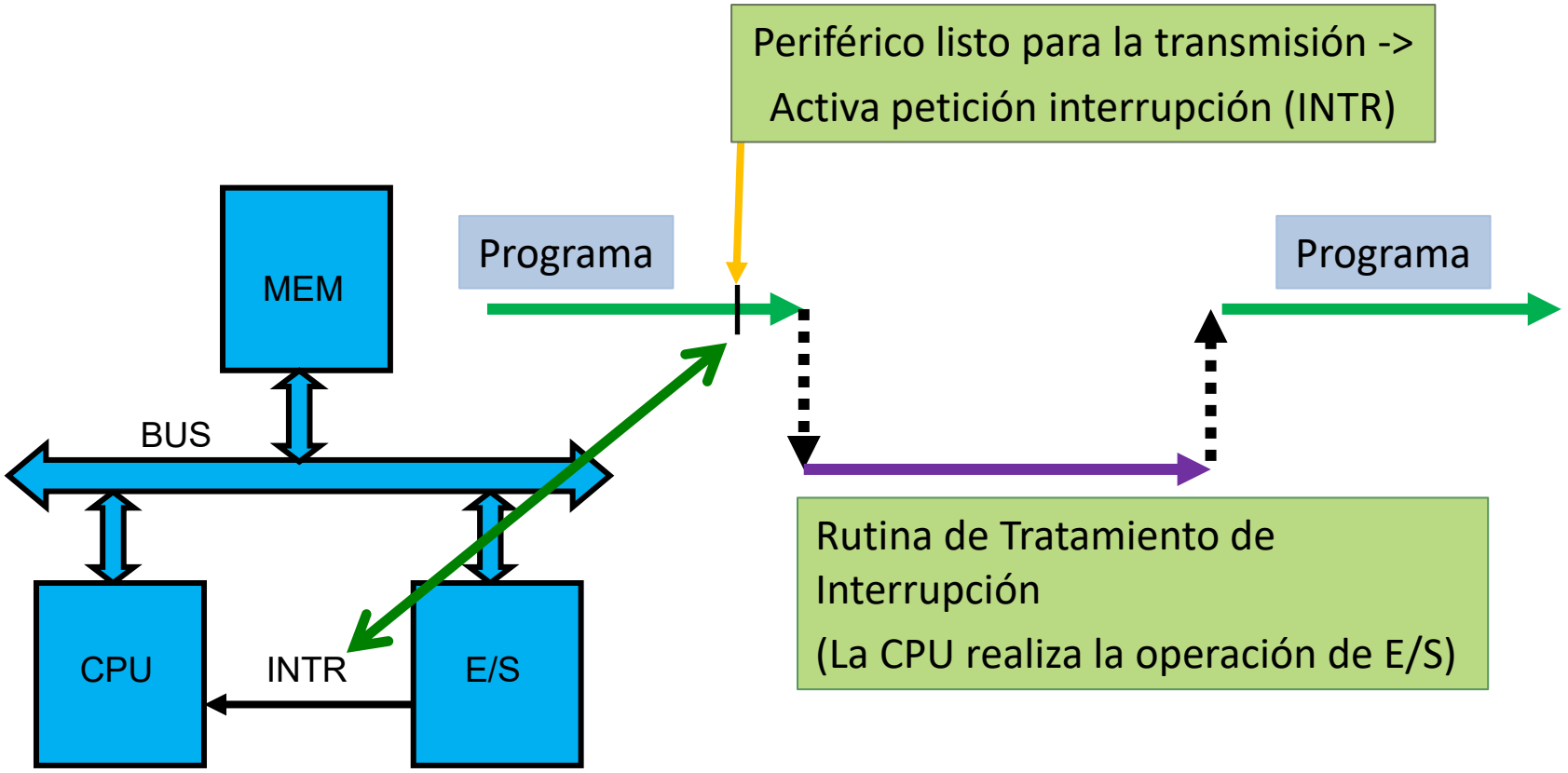
EXCEPCIÓN VS INTERRUPCIÓN

- Una **excepción interna** se produce debido a la ***ejecución (incorrecta) de instrucciones del programa***
  - Siempre que se ejecute esa instrucción se producirá la excepción
  - Ejemplo del ARM:
    - Si se intenta acceder a un dato de tamaño palabra usando una dirección que no es múltiplo de 4 se produce un DATA ABORT.
    - El procesador bifurca a la subrutina asociada a esa excepción ISR_Dabort .

- Una **interrupción** se produce debido a una ***señal externa al procesador***
  - Es un evento asíncrono, que avisa al procesador de que necesita su atención
  - Ejemplo: En la lectura de pulsadores se podría programar el controlador para que cada vez que se pulse el botón genere una interrupción por IRQ.

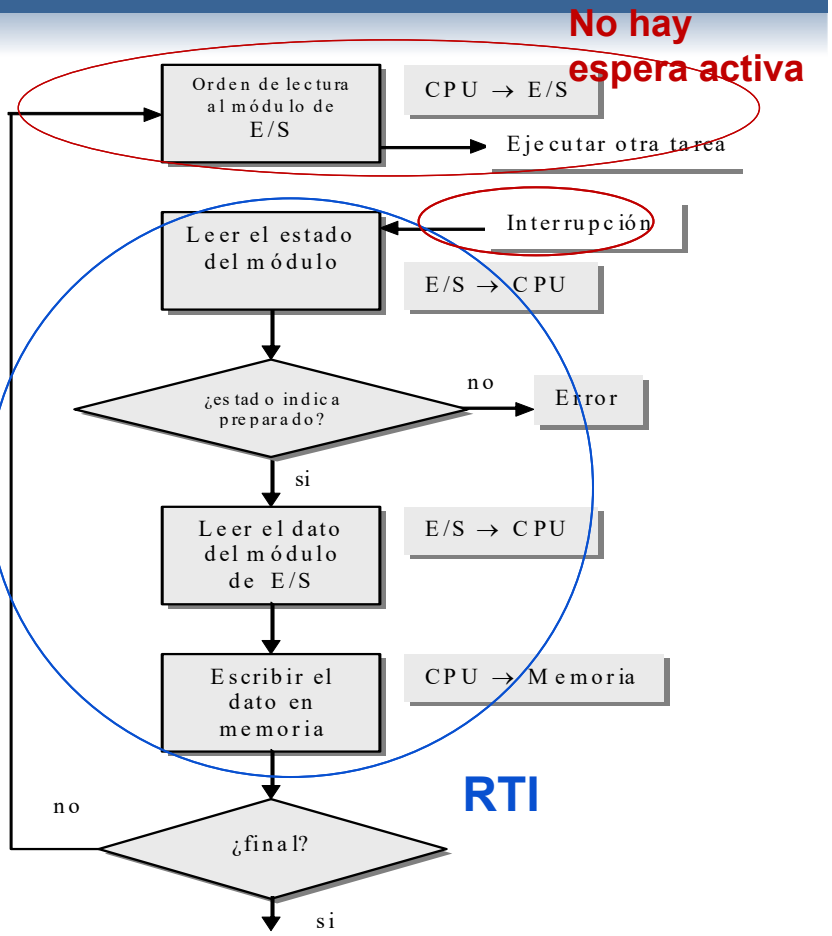


OPERACIÓN DE E/S MEDIANTE INTERRUPCIONES

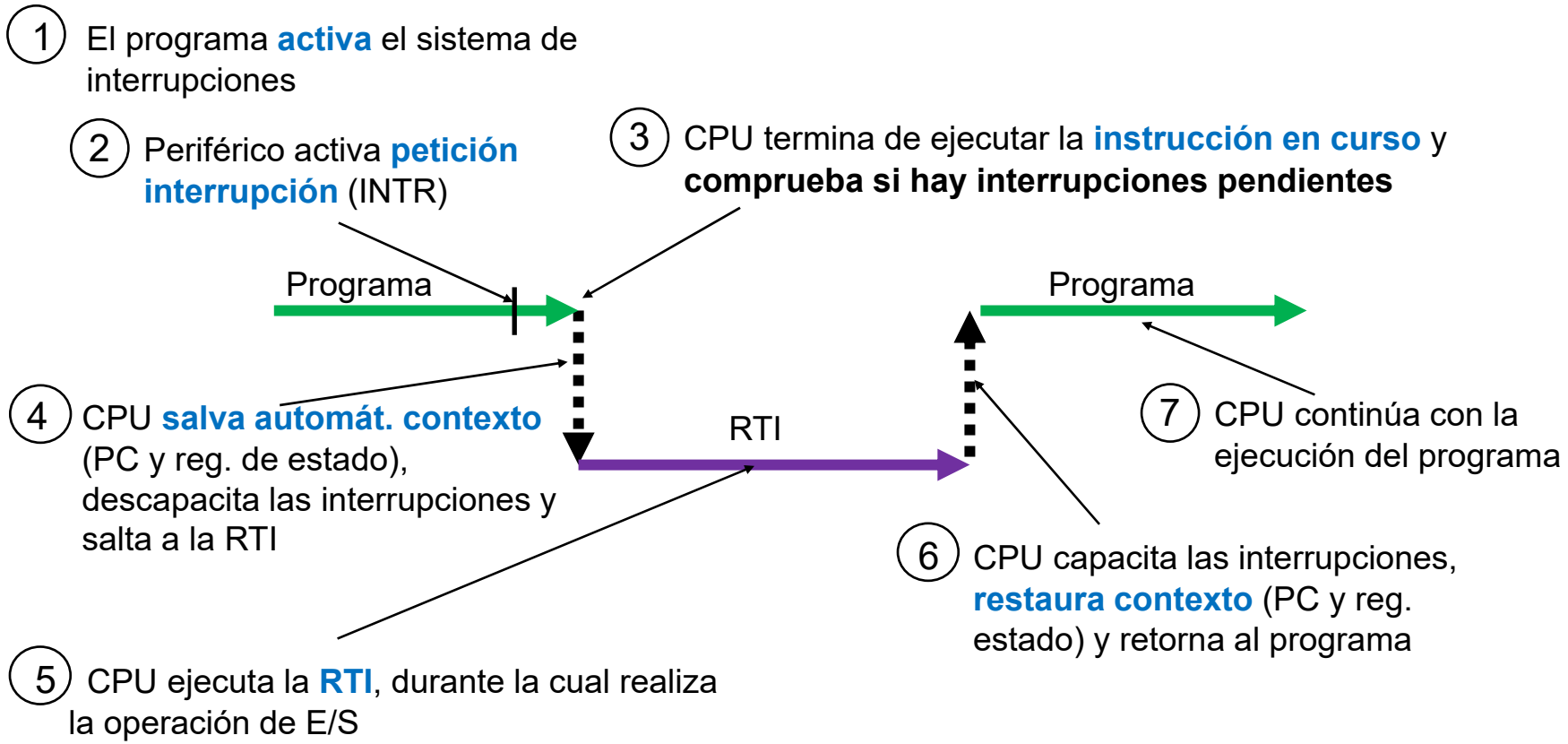


OPERACIÓN DE E/S MEDIANTE INTERRUPCIONES

- La CPU hace la petición de operación de E/S y pasa a ejecutar otros programas.=>No existe bucle de espera
- Cuando un periférico está listo para transmitir se lo indica a la CPU activando una LÍNEA DE PETICIÓN INTERRUPCIÓN
- Cuando la CPU recibe una señal de petición de interrupción salta a una RUTINA DE TRATAMIENTO DE INTERRUPCIONES (RTI), que se encarga de atender al periférico que interrumpió y realizar la operación de E/S



LA GESTIÓN DE INTERRUPCIONES



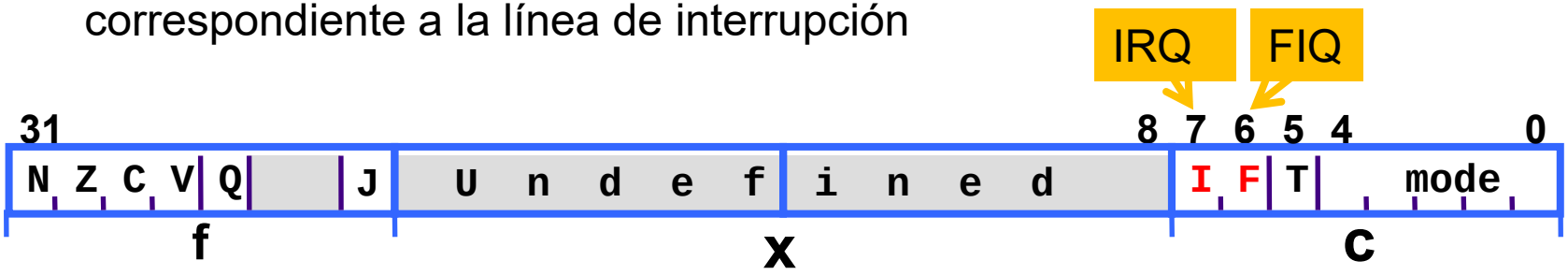
HABILITACIÓN DE INTERRUPCIONES

1 El programa **activa** el sistema de interrupciones

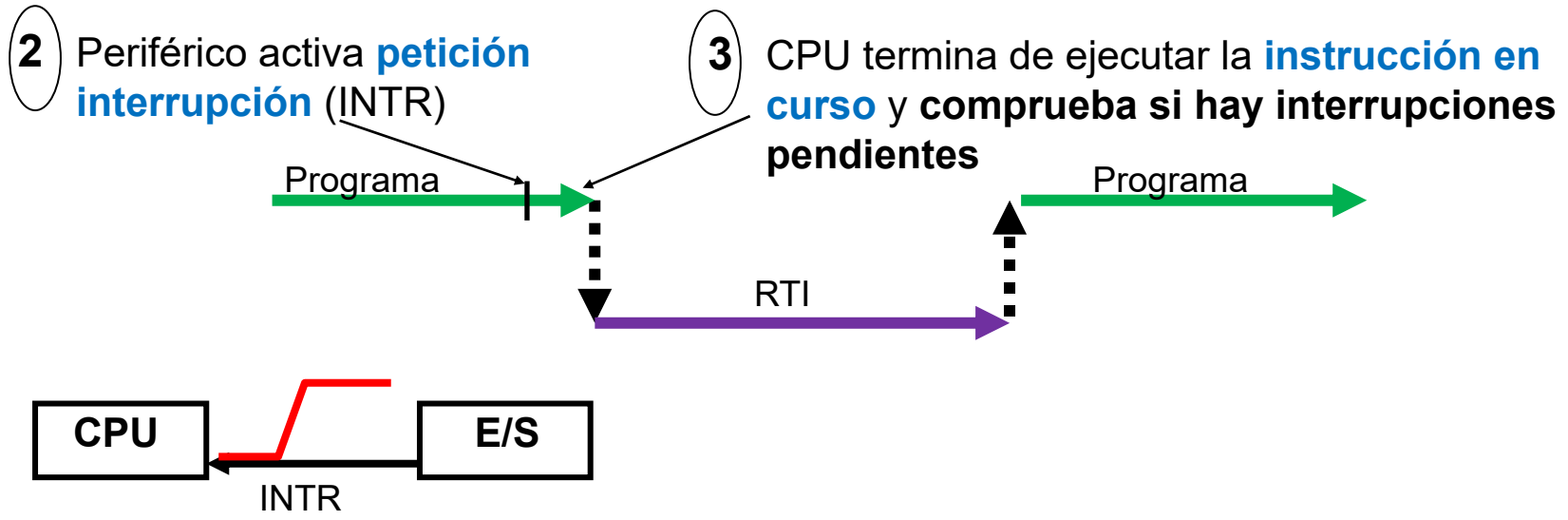
Para que pueda producirse una interrupción la CPU debe permitir que un dispositivo le interrumpa, **habilitando las interrupciones**:

Localmente: se indica al controlador del dispositivo que puede generar una señal de interrupción

Globalmente: en el registro de estado de la CPU se activa el bit correspondiente a la línea de interrupción



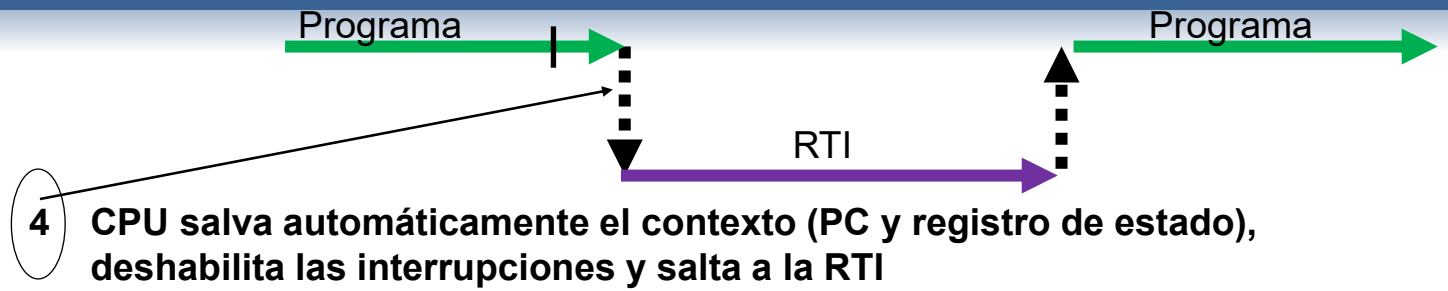
COMPROBACIÓN DE PETICIÓN DE INTERRUPCIÓN



COMPROBACIÓN DE PETICIÓN DE INTERRUPCIÓN

- La CPU **comprueba si hay interrupciones pendientes** (línea INTR activada) al final de la ejecución de cada instrucción
  - **Motivo:**
    - Sólo es necesario guardar el PC, el registro de estado y los registros accesibles por programa (registros de datos y/o direcciones)
    - Si se interrumpiese una instrucción en mitad de la ejecución sería necesario guardar el valor de todos los registros internos/ estado completo de la CPU
      - Registro de instrucción, registros de dirección de datos, registros de datos de memoria, etc.
  - **Salvo para:**
    - Instrucciones de larga duración
      - Por ejemplo, en instrucción de movimiento múltiple (STM), se comprueba si hay interrupciones pendientes después de mover cada una de las palabras

SALTO A LA RTI



Salvar el estado:
El procesador guarda **automáticamente** el contexto del programa en ejecución (En el ARM el PC y registro de estado) en una zona de la pila distinta de la del programa o en registros especiales

Deshabilitar (enmascarar) las interrupciones:
En el registro de estado se enmascaran (deshabilitan) **automáticamente** las interrupciones por la línea INTR

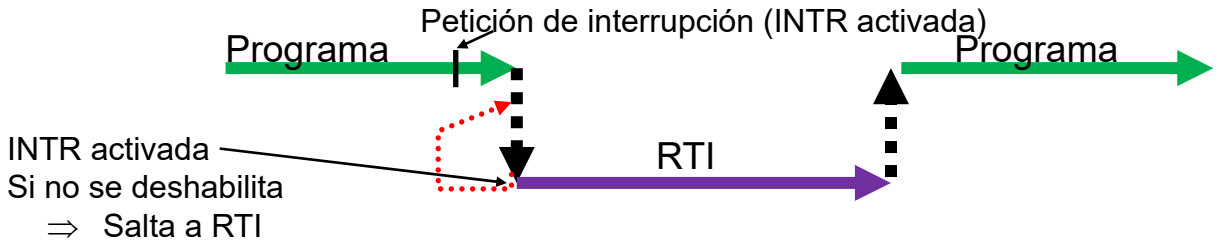
¿Por qué?

Saltar a RTI:
Se obtiene la dirección de la RTI que corresponda al periférico que ha interrumpido

¿Cómo se identifica al periférico?

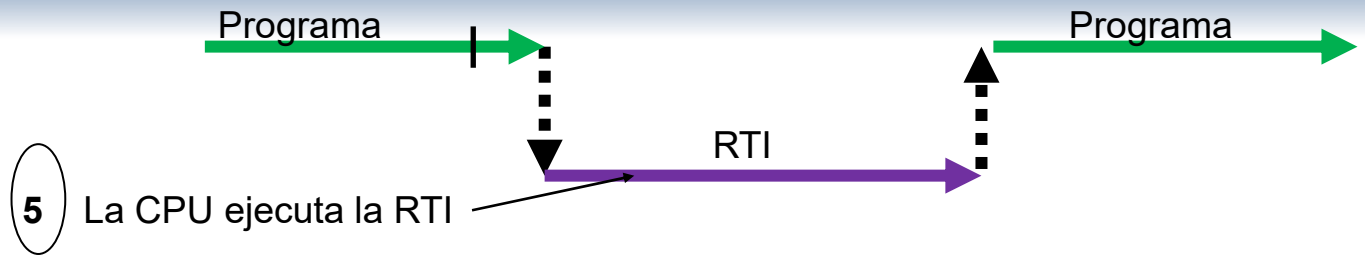
DESHABILITAR LAS INTERRUPCIONES

- **Antes de saltar a la RTI es necesario enmascarar las interrupciones**
- **Motivo: Si no se enmascaran la CPU puede entrar en un bucle infinito**
  - Cuando se entra en la RTI el periférico todavía no ha desactivado su petición
  - Si las interrupciones están habilitadas ⇒ la CPU detecta una interrupción pendiente y vuelve a saltar a la RTI una y otra vez
  - Antes de finalizar la RTI hay que asegurarse de que el periférico ha desactivado la línea de petición INTR



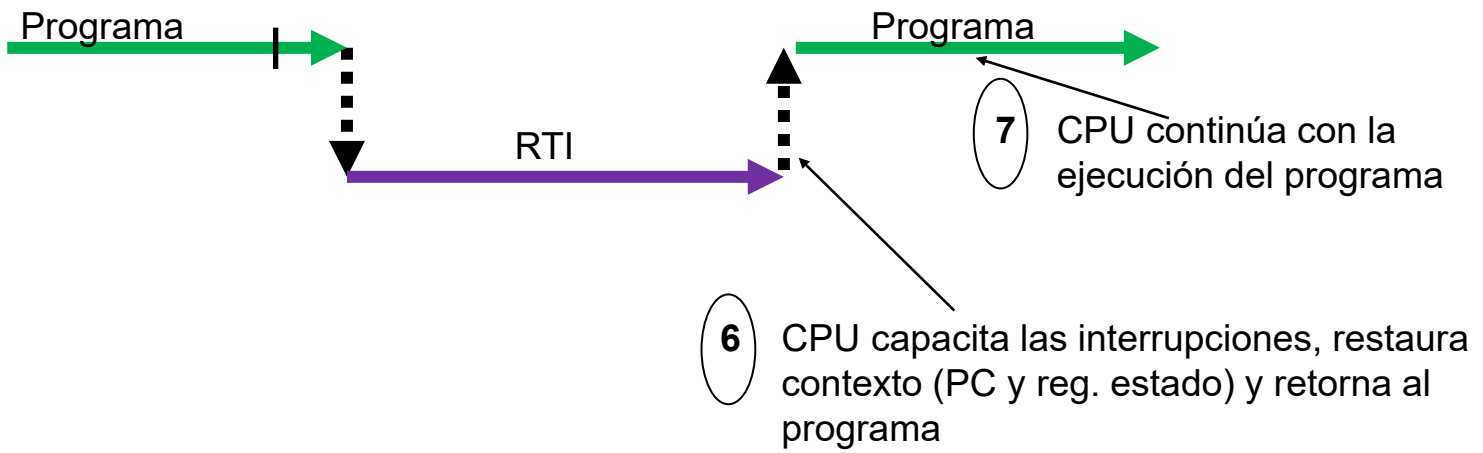
- **Alternativas**
  - **Enmascaramiento global**
    - Se deshabilitan todas las interrupciones ⇒ ningún otro periférico podrá interrumpir durante la ejecución de la RTI
  - **Enmascaramiento selectivo**
    - Cuando hay varios niveles de interrupción se pueden deshabilitar las interrupciones por el nivel que interrumpe, pero no necesariamente por el resto de niveles Ej: IRQ y FIQ

EJECUCIÓN DE LA RTI



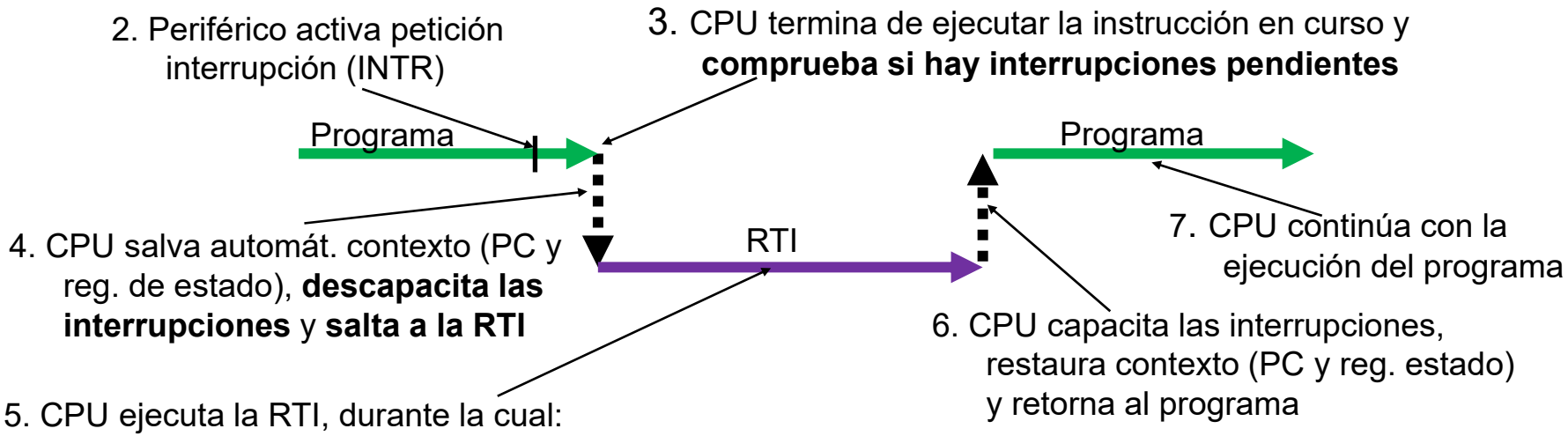
- Responsabilidades de la RTI:
  - Salvar en pila todos los registros que utiliza (manual)
    - Si va a invocar a subrutinas, tiene que cumplir el estándar de subrutinas
  - Informar al periférico de que se ha reconocido su interrupción. **¿Cómo?**
    - Por **software**: borrar flag de petición en un registro de control
    - Por **hardware:** activando una señal de reconocimiento de interrupción (INTA)
      - **Una vez informado el periférico desactiva INTR**
  - Realizar la operación de E/S con el periférico
  - Restaurar el contexto arquitectónico salvado
  - Realizar el retorno de interrupción

AL FINAL DE LA EJECUCIÓN DE LA RTI



GESTIÓN DE INTERRUPCIONES

1. El programa activa el sistema de interrupciones



- Informa al periférico que se ha reconocido su interrupción (por software o por hardware) => El periférico desactiva INTR
- Salva en pila todos los registros de datos y/o direcciones utilizados por la RTI (manual)
- Realiza la operación de E/S con el periférico
- Restaura el contexto arquitectónico salvado
- Realizar el retorno de interrupción correcto

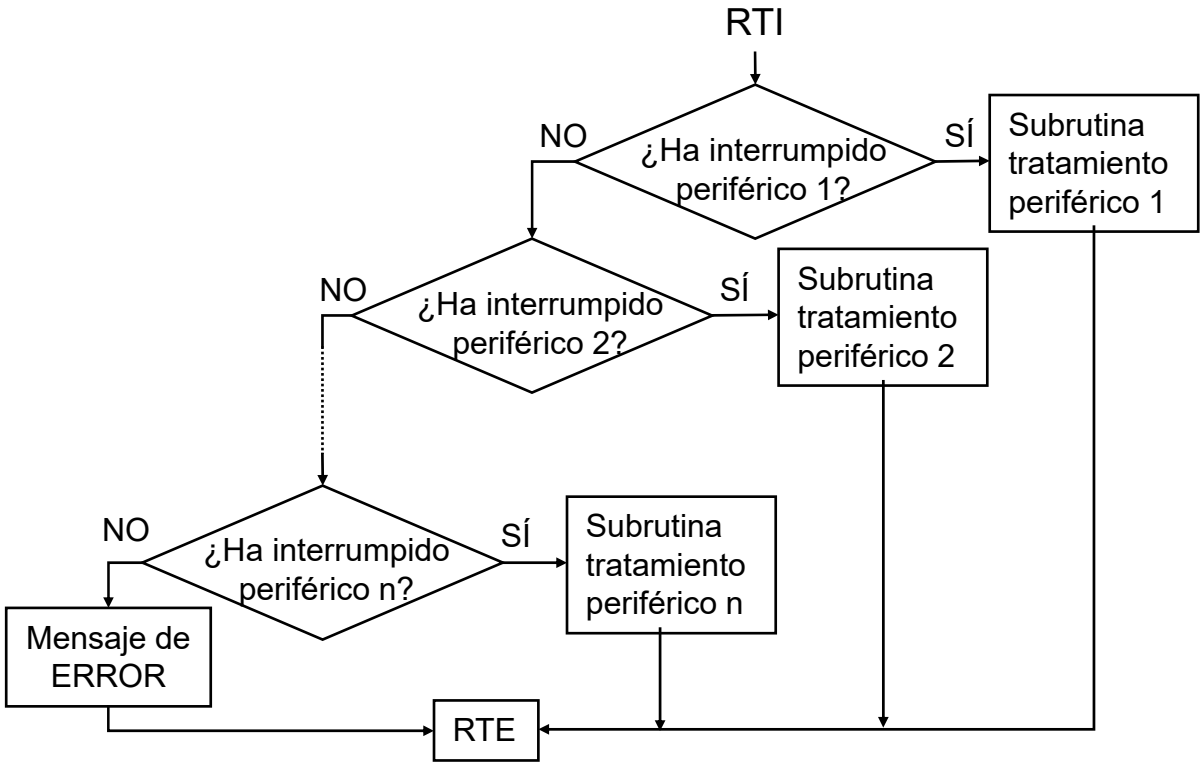
CUESTIONES PENDIENTES

- Muchos procesadores (ej ARM del lab) tienen **una única línea de petición de interrupciones (IRQ)**, por lo que a una misma línea tienen que conectarse varios periféricos. Es necesario un **mecanismo para identificar** al dispositivo que interrumpió
  - Identificación software: por encuesta (polling)
  - Identificación hardware: controlador de interrupciones
- ¿Qué ocurre si se produce una segunda interrupción durante la ejecución de la RTI?
  - Interrupciones multinivel y anidamiento de interrupciones

IDENTIFICACIÓN SOFTWARE: ENCUESTA

- Ante cualquier petición se carga la instrucción de una posición fija **única para todos los dispositivos (ejecuta una RTI general)**
  - La RTI debe identificar por software qué dispositivo solicitó la interrupción
    - Consulta en un orden los registros de control de los controladores HW de los dispositivos (**Encuesta**)
    - El orden de la encuesta determina la prioridad de los dispositivos (**prioridad software**)
    - Se atiende al primer dispositivo que se encuentre solicitando la interrupción (**se ejecuta su RTI específica**)
    - La RTI sólo **borra el flag de petición de ese dispositivo**

IDENTIFICACIÓN SOFTWARE: ENCUESTA



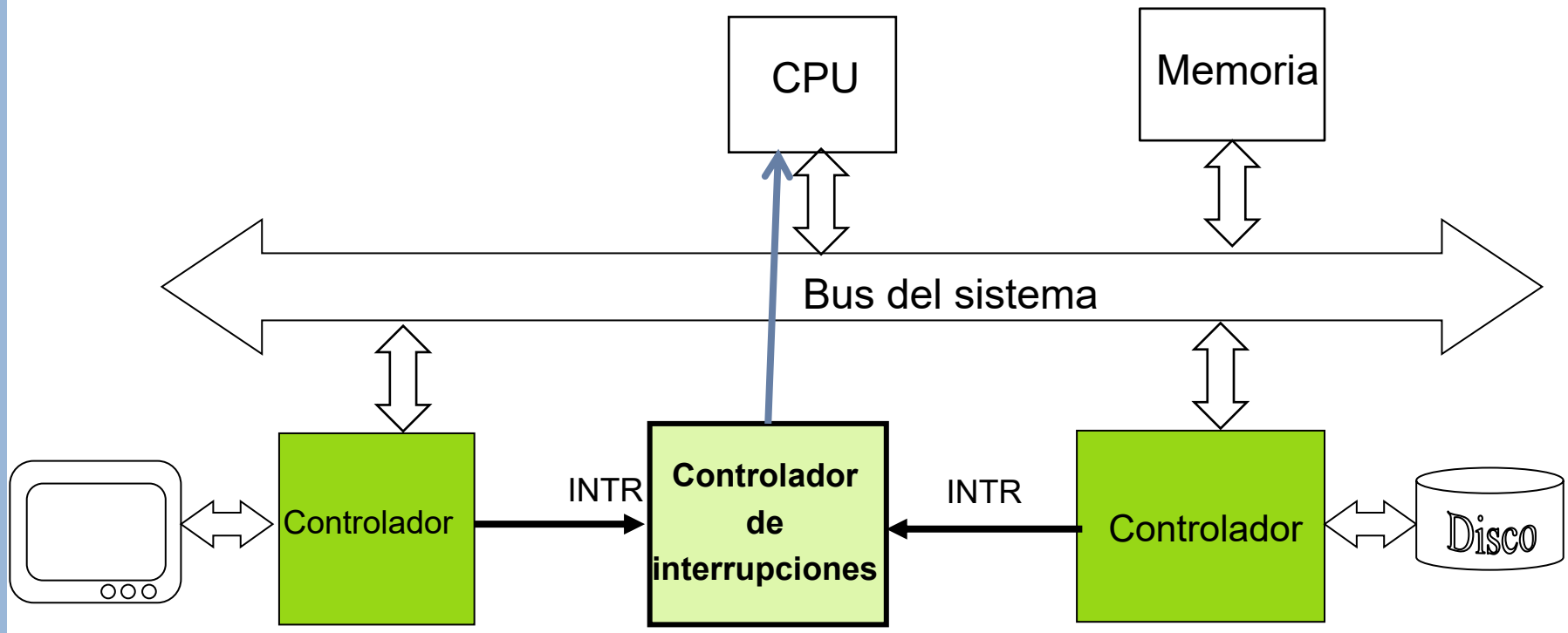
IDENTIFICACIÓN SOFTWARE: ENCUESTA

- **Ventajas**
  - Mecanismo sencillo
  - Fácil de extender
- **Desventajas**
  - RTI tiene que invertir tiempo en identificar la fuente
    - Puede ralentizar mucho el tratamiento de la interrupción (normalmente esto es crítico)
    - El problema se incrementa con el número de dispositivos conectados

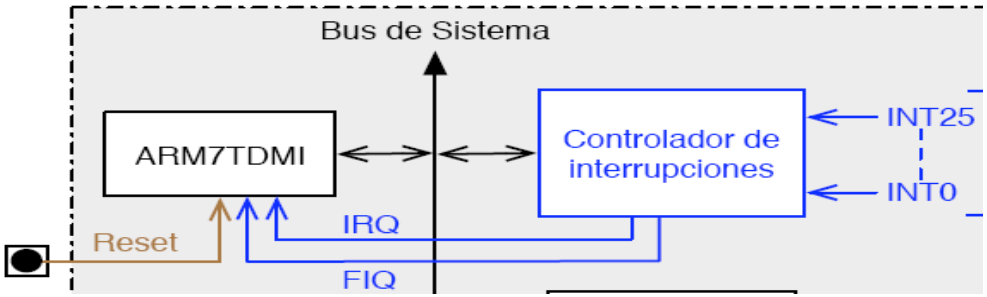
INDICE

- Introducción
- Estructura y funciones del sistema de E/S
- E/S Programada
- E/S por interrupciones
  - Concepto de excepción. Gestión de excepciones.
  - Operación de E/S mediante interrupciones
- **Controlador de interrupciones**
- Sistema de interconexión. Buses
- Entrada/salida por Acceso Directo a Memoria

SISTEMA DE E/S CON CONTROLADOR DE INTERRUPCIONES

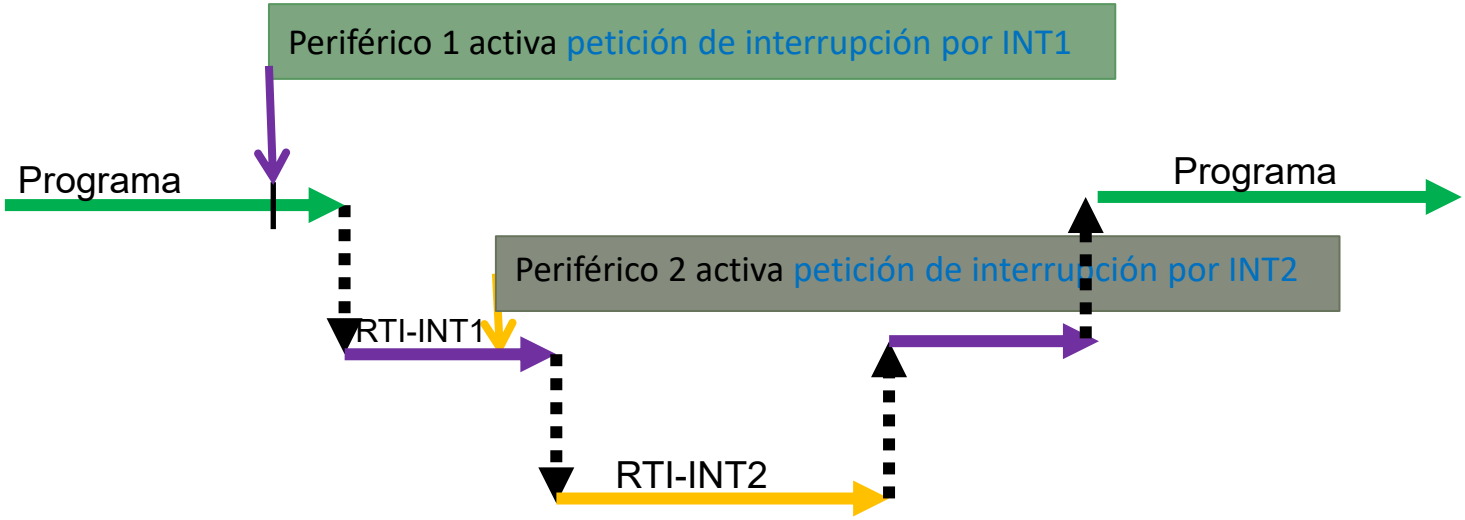


INTERRUPCIONES MULTINIVEL



- Varias líneas o niveles de petición de interrupción
  - Cada nivel tiene asignada una prioridad distinta
  - A cada línea de interrupción se pueden conectar uno o varios dispositivos
- Resolución de conflictos de peticiones de interrupción simultáneas
  - Peticiones simultáneas por la misma línea
    - Encuesta (software)
  - Peticiones simultáneas por líneas distintas
    - Se atiende a la línea más prioritaria

ANIDAMIENTO DE INTERRUPCIONES

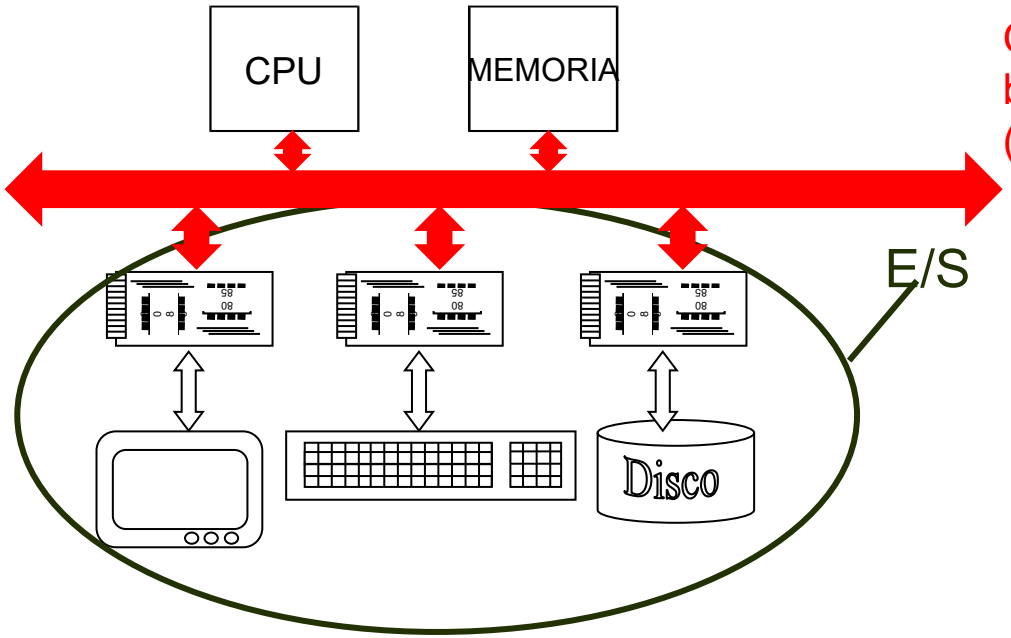


INDICE

- Introducción
- Estructura y funciones del sistema de E/S
- E/S Programada
- E/S por interrupciones
  - Concepto de excepción. Gestión de excepciones.
  - Operación de E/S mediante interrupciones
- Controlador de interrupciones
- **Sistema de interconexión. Buses**
- Entrada/salida por Acceso Directo a Memoria

BUSES DE INTERCONEXIÓN

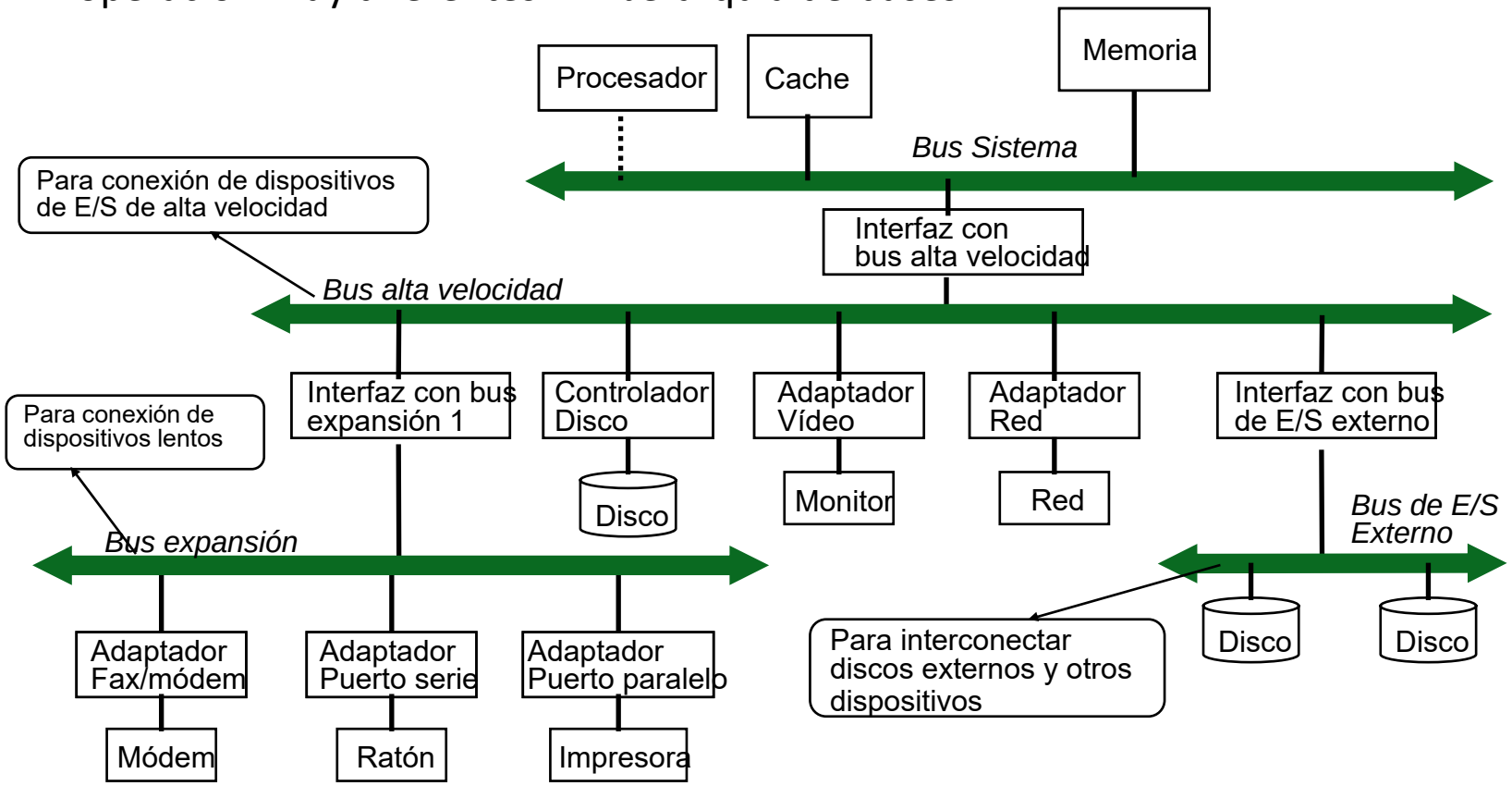
- Sistema de Interconexión. Necesidad de los buses
  - Para que la CPU pueda intercambiar información con el sistema de memoria y los dispositivos de E/S es necesario un medio de comunicación denominado BUS



Originalmente se usaba un bus compartido entre todos (cuello de botella)

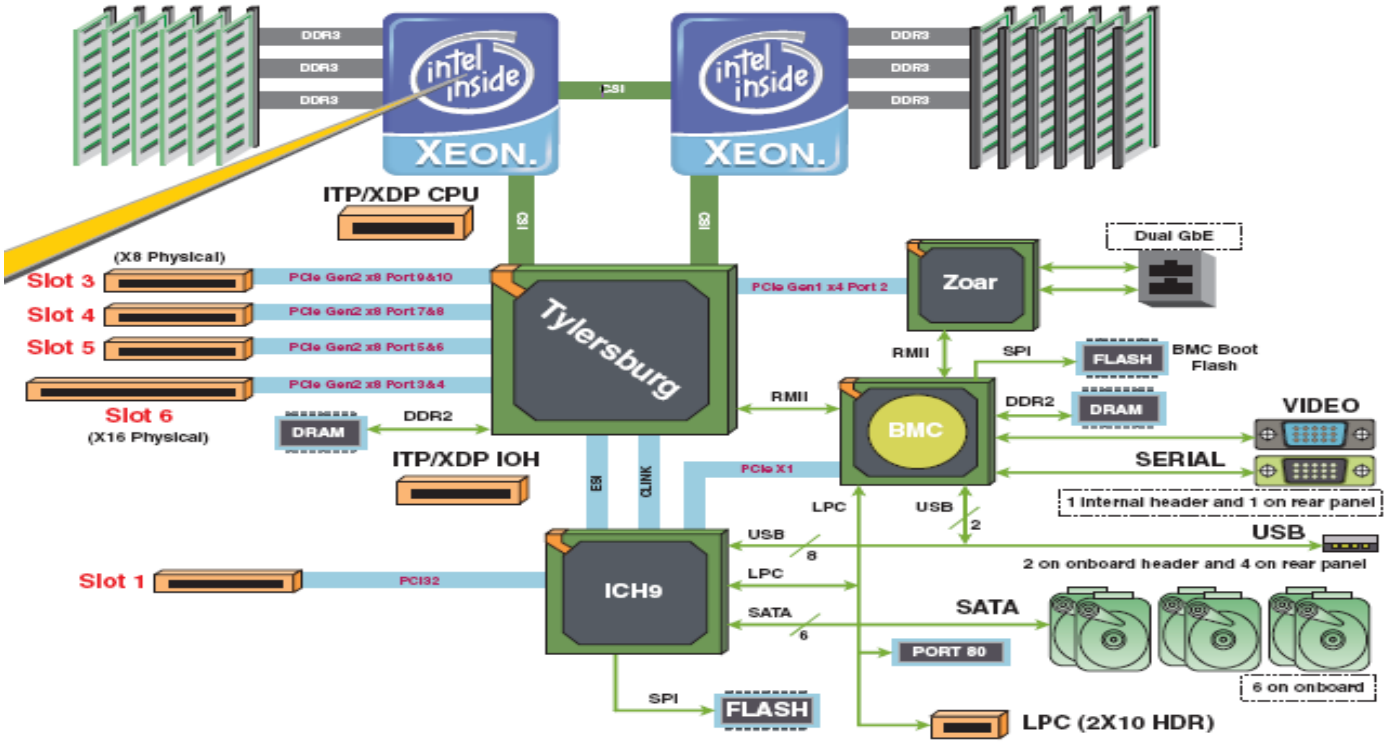
BUSES DE INTERCONEXIÓN

- El uso de un único bus es inadecuado con dispositivos con velocidades de operación muy diferentes. => Jerarquía de buses

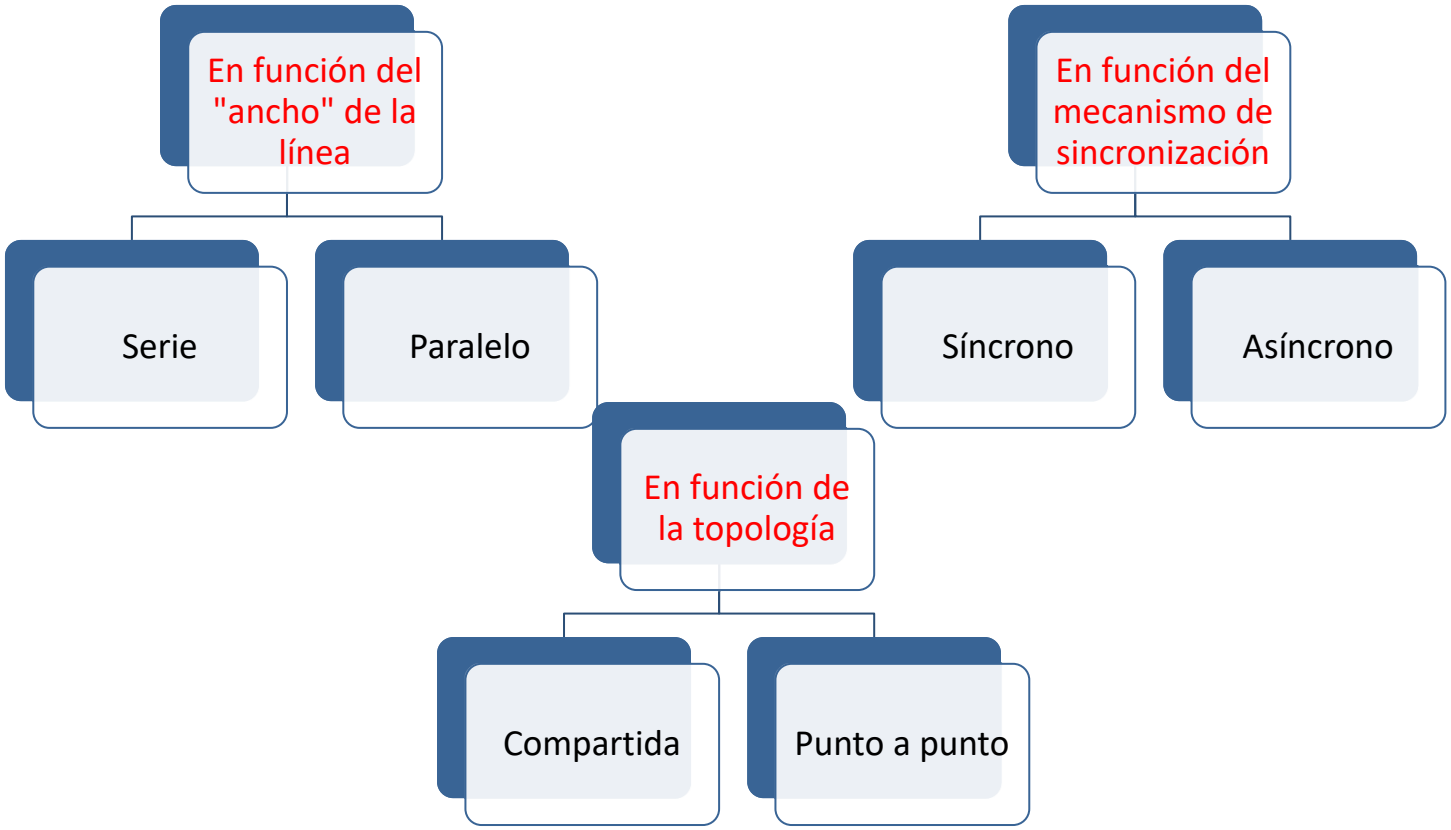


BUSES DE INTERCONEXIÓN

- Sistema actual más complejo, con conexiones punto a punto (QPI, PCIe) y otras conexiones tipo bus.

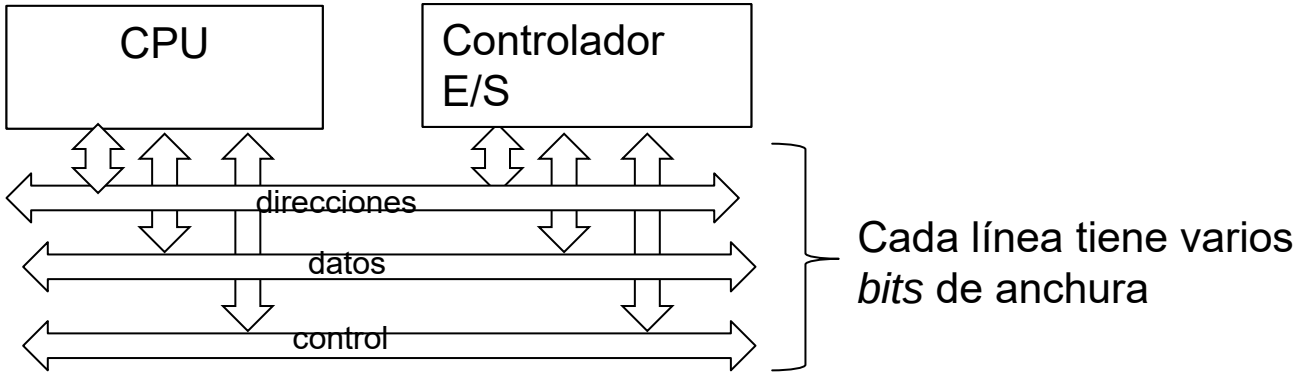


Taxonomía de la líneas de comunicación



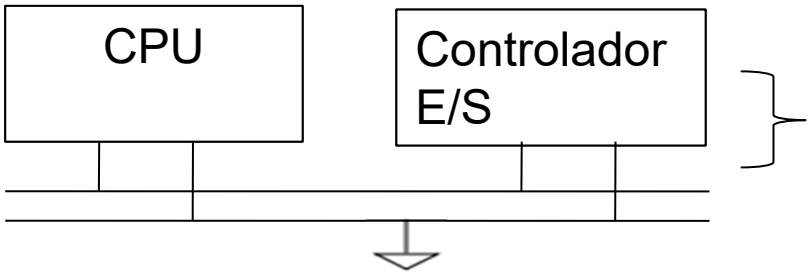
Transmisión paralela

- Utiliza varias líneas de comunicación (cables) a través de las cuales **se envían varios bits de información de forma simultánea**
  - Ancho de banda elevado (en teoría…)
  - Para conectar dispositivos a distancias medias o largas resulta muy costosa
  - Frecuencia de funcionamiento limitada por factores físicos
  - Los dispositivos de baja velocidad no aprovechan el potencial de la transmisión paralela (ratón, módem…)



Transmisión serie

- Utiliza una única línea de comunicación (2 cables: dato y tierra) a través de la cual **se envían los bits de información de forma secuencial**
  - Es menos costosa que la E/S paralela
  - Permite frecuencias de funcionamiento mayores
  - A igualdad de frecuencia que el caso paralelo, el ancho de banda es menor



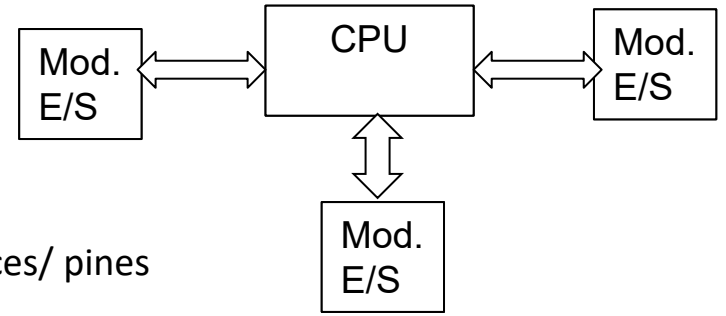
Toda la información (datos, control…) se emite secuencialmente

Es posible **aumentar el ancho de banda** entre dos dispositivos **usando varias conexiones serie**

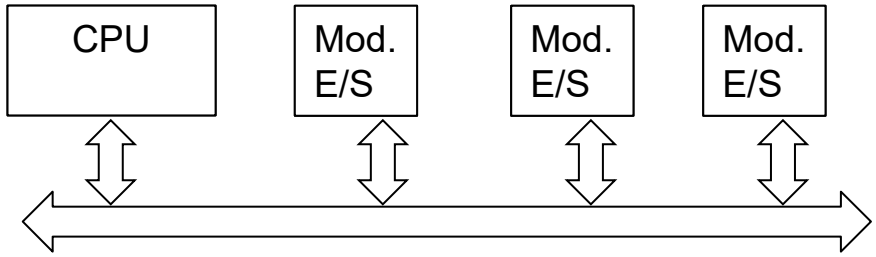
Comunicación síncrona vs asíncrona

- ¿Cuándo empieza/acaba el envío de un dato?
  - **Asíncrona**
    - La señal de reloj NO se envía por la línea de comunicación
    - Emisor y receptor utilizan sus propias señales de reloj.
    - Es necesario establecer un protocolo para la sincronización entre ambos
    - No es imprescindible que emisor y receptor funcionen a la misma frecuencia
  - **Síncrona**
    - La señal de reloj viaja con el resto de señales
    - Más sencillo y, en general, más eficiente
    - Exige que emisor y receptor funcionen a la misma frecuencia
    - Debe ser corto si queremos que sea rápido (para que no le afecte el clock skew)

Punto a punto vs. Compartida



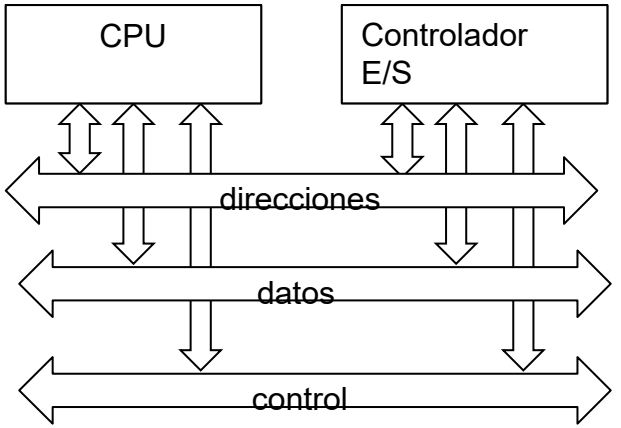
- **Comunicación Punto a punto**
  - Gran rendimiento
  - Exige un gran número de interfaces/enlaces/ pines



- **Compartida**
  - Más versátil y barato
  - Cuello de botella en la comunicación
  - Exige arbitraje entre dispositivos para evitar escrituras simultáneas

Organización general de un bus

- **Líneas de control**
  - Utilizadas para implementar el protocolo de comunicación entre componentes
- **Líneas de datos**
  - Transmite la información entre emisor y receptor
  - Anchura de bus: número de bits que se puede transmitir en un ciclo de bus
- **Líneas de dirección**
  - Contienen la información de direccionamiento
  - Pueden ser diferentes o compartir líneas con los datos (multiplexado)



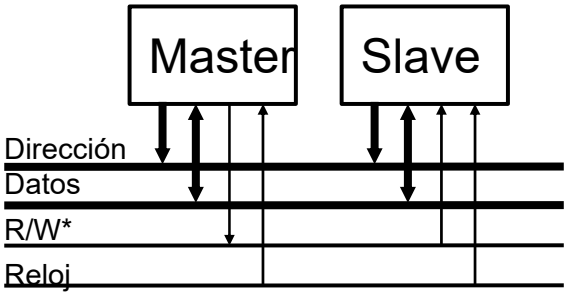
Transferencia de datos en un bus

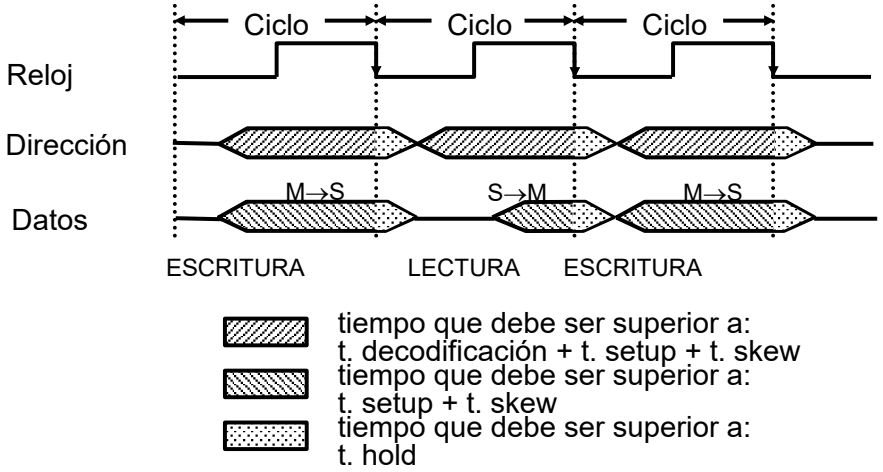
- **Tipos básicos de transferencia**
  - Escritura
  - Lectura
- **Fases de una transferencia**
  1. Direccionamiento (identificación) del slave
  2. Especificación del tipo de operación (lectura o escritura)
  3. Transferencia del dato
  4. Finalización del ciclo de bus
- **Control de la transferencia**
  - Sincronización: determinar el inicio y el final de cada transferencia
  - Arbitraje: control del acceso al bus en caso de varios masters

Sincronización: protocolo síncrono

- **Existe un reloj que gobierna todas las actividades del bus**, las cuales tienen lugar en un número entero de ciclos de reloj (1 en el ejemplo)
- Los **flancos del reloj** (de bajada en el ejemplo) determinan el comienzo de un nuevo ciclo de bus y el final del ciclo anterior
  - ***Ventajas***:
    - Simplicidad (de diseño y de uso)
    - Sólo se necesita una señal (reloj) para llevar a cabo la sincronización
    - Mayor velocidad (en relación a protocolo asíncrono)
  - ***Desventajas***:
    - El periodo de reloj se tiene que adaptar a la velocidad del dispositivo más lento (por lo que suele usarse para conectar dispositivos homogéneos)
    - No existe confirmación de la recepción de los datos
    - La necesidad de distribuir la señal de reloj limita la longitud del bus

Sincronización: protocolo síncrono



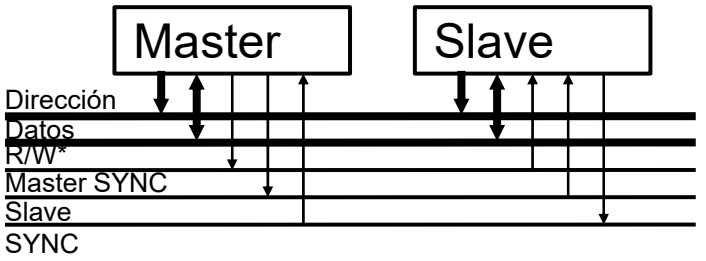


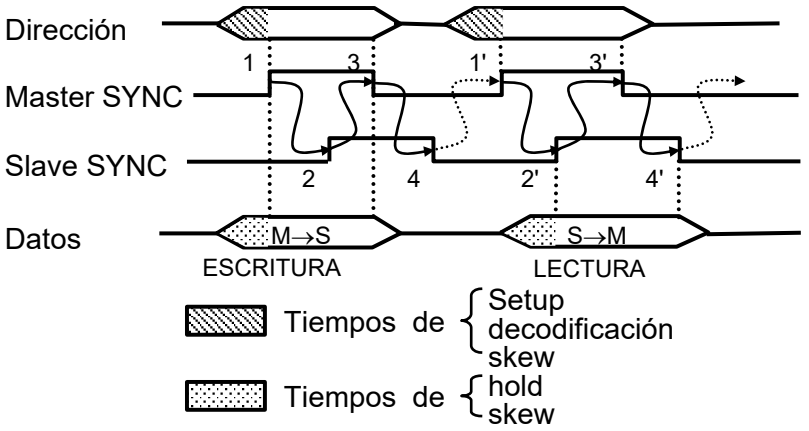
Sincronización: protocolo asíncrono

- **No existe señal de reloj**
- Los dispositivos implicados en la transmisión fijan el comienzo y el final de la misma mediante el intercambio de señales de control (**handshake**)
- Se utilizan 2 señales de sincronización:
  - Master SYNC (procedente del master)
  - Slave SYNC (procedente del slave)
- Protocolo completamente interbloqueado:
  - A cada flanco de master le sigue uno del slave

Sincronización: protocolo asíncrono

- **Ciclo de escritura:**
  - 1: (M a S) Hay un dato en el bus
  - 2: (S a M) He tomado el dato
  - 3: (M a S) Veo que lo has tomado
  - 4: (S a M) Veo que lo has visto (Bus libre)
- **Ciclo de lectura:**
  - 1': (M a S) Quiero un dato
  - 2': (S a M) El dato está en el bus
  - 3': (M a S) He tomado el dato
  - 4': (S a M) Veo que lo has tomado (Bus libre)





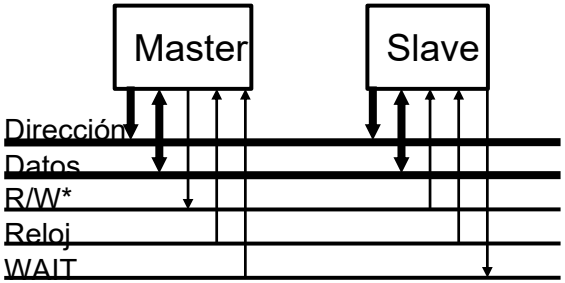
Sincronización: protocolo asíncrono

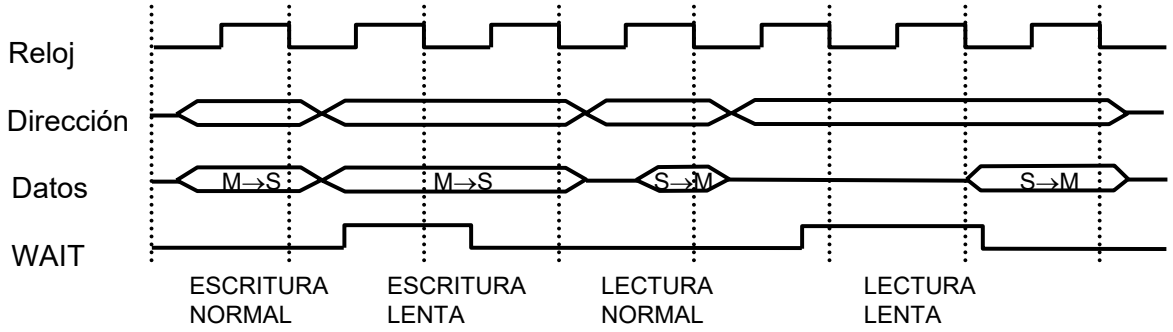
- ***Ventajas:***
  - Facilidad para conectar elementos de diferentes velocidades
  - Fiabilidad: la recepción siempre se confirma.
- ***Desventajas:***
  - El intercambio de señales de control introduce retardos adicionales
  - A igualdad de velocidades de los dispositivos, menos eficiente que el síncrono

Sincronización: protocolo semisíncrono

- Las transferencias se rigen por una **única señal de reloj**
- Cada transferencia puede ocupar uno o varios periodos de reloj
- Señal de WAIT. Puede activarla cualquier Slave si no es capaz de realizar la transferencia en 1 solo ciclo. La desactiva cuando es capaz de finalizar con el siguiente flanco de reloj.
  - **Dispositivos rápidos**: operan como en un bus síncrono (una transferencia por ciclo)
  - **Dispositivos lentos**: activan la señal de WAIT y congelan actuación del Master (una transferencia puede ocupar varios ciclos)

Sincronización: protocolo semisíncrono





BUSES DE INTERCONEXIÓN: ARBITRAJE

Conflictos en el acceso: arbitraje

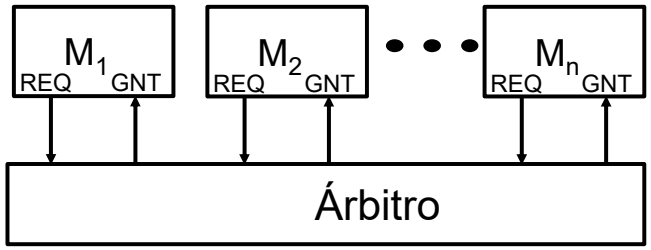
- Si puede haber varios emisores (Masters), es necesario que el acceso al bus sea *libre de conflictos*
  - Se debe evitar que dos módulos escriban simultáneamente en el bus
  - Cada dispositivo solicita permiso para poder tomar el control del bus

- **Protocolos centralizados**: existe un **árbitro** del bus o **máster principal** que controla el acceso al bus
  - En estrella
  - Daisy-chain de 3 y 4 hilos

- **Protocolos distribuidos**: el control de acceso al bus se lleva a cabo entre todos los posibles masters de una forma cooperante
  - Protocolo de líneas de identificación

BUSES DE INTERCONEXIÓN: ARBITRAJE

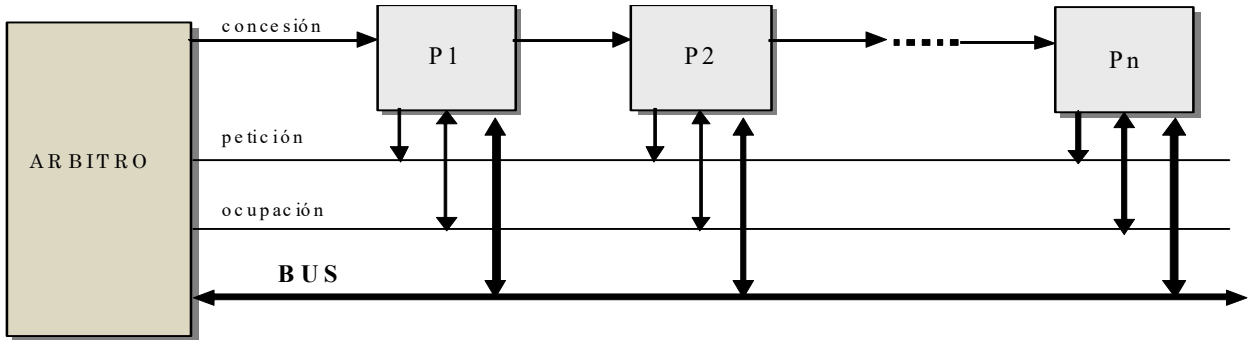
Protocolos centralizados: en estrella



- Cada máster se conecta al árbitro mediante dos líneas individuales:
  - **BUS REQUEST** (REQ): línea de petición del bus
  - **BUS GRANT** (GNT): línea de concesión del bus
- Varias peticiones de bus pendientes: el árbitro puede aplicar distintos algoritmos de decisión:
  - FIFO, Prioridad fija, Prioridad variable
- **Ventajas:**
  - Algoritmos de arbitraje simples
  - Pocos retardos de propagación de las señales (en comparación con daisy-chain)
- **Desventajas:**
  - Número elevado de líneas de arbitraje en el bus (dos por cada posible máster)
  - Número de másteres alternativos limitado por el número de líneas de arbitraje
- Ejemplo: PCI

Protocolo de encadenamiento (daisy chaining) de 3 señales

- **Utiliza tres líneas para gestionar el arbitraje**: **petición, ocupación y concesión**.
  - **La línea de petición es compartida** por todos los procesadores a través de una entrada al árbitro con capacidad de O-cableada.
  - El árbitro activa la concesión cuando recibe una petición y la de ocupación está desactivada.
  - Cuando un procesador toma el control del bus activa la línea de ocupación.
  - Si un procesador recibe la concesión y no ha solicitado el bus, transmite la señal al siguiente.
  - Un procesador toma el control del bus si tiene una petición local pendiente, la línea de ocupación está desactivada y recibe el flanco de subida de la señal de concesión.

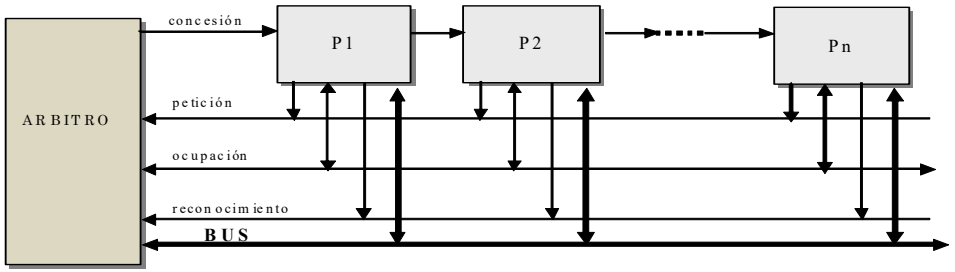


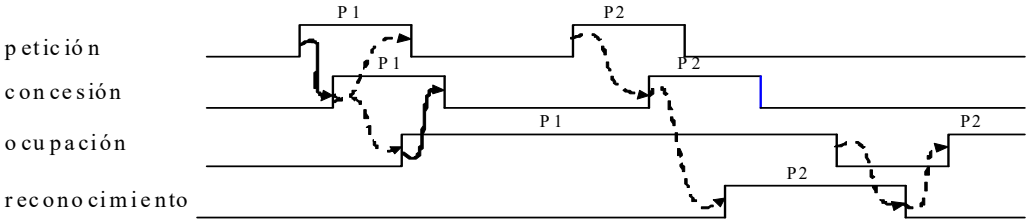
BUSES DE INTERCONEXIÓN: ARBITRAJE

Protocolo de encadenamiento (daisy chaining) de 4 señales

- Permite simultanear el uso del bus por un procesador con el arbitraje para seleccionar el procesador que utilizará el bus en la siguiente transacción.
- Cuando el primer procesador libera el bus, el arbitraje para el siguiente ya se ha realizado.
- La línea nueva de **reconocimiento** la activa un procesador que solicitó el bus (activó petición) y recibió la concesión pero la línea de ocupación estaba activa (bus ocupado).
- Cuando el árbitro recibe la activación de reconocimiento inhibe su actuación, es decir, deja de atender la señal de petición y generar la de concesión.
- El procesador queda en espera de ocupar el bus cuando lo abandone su actual usuario desactivando la señal ocupación.
- Cuando esto ocurre, el procesador ocupa el bus y desactiva la señal de reconocimiento.

Protocolo de encadenamiento (daisy chaining) de 4 señales

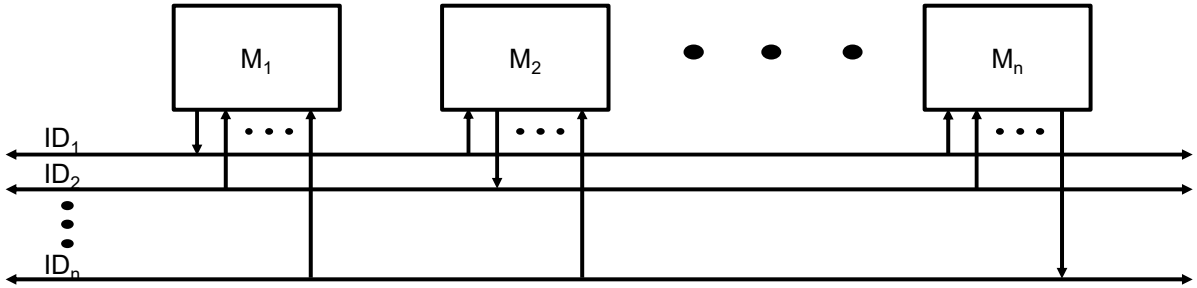




Diálogo de señales:
- Ocupación del bus por P1,
- Arbitraje a favor de P2 mientras P1 realiza su transacción.
- P2 ocupa el bus cuando P1 finaliza

BUSES DE INTERCONEXIÓN: ARBITRAJE

- **Protocolo de líneas de identificación**
  - Cuando un master quiere tomar el control del bus activa su línea de identificación
  - Cada línea de identificación tiene asignada una prioridad: $(ID_1)<(ID_2)< ....< (ID_n)$
  - Si varios masters activan simultáneamente sus líneas de identificación, gana el de mayor prioridad
  - Funcionamiento alternativo: las prioridades pueden ser variables
- *Desventajas:*
  - Número de dispositivos limitado por el número de líneas de arbitraje
- **Prioridad variable**: DEC 70000/10000 AXP, AlphaServer 8000
- **Prioridad fija**: VAX SBI, SCSI

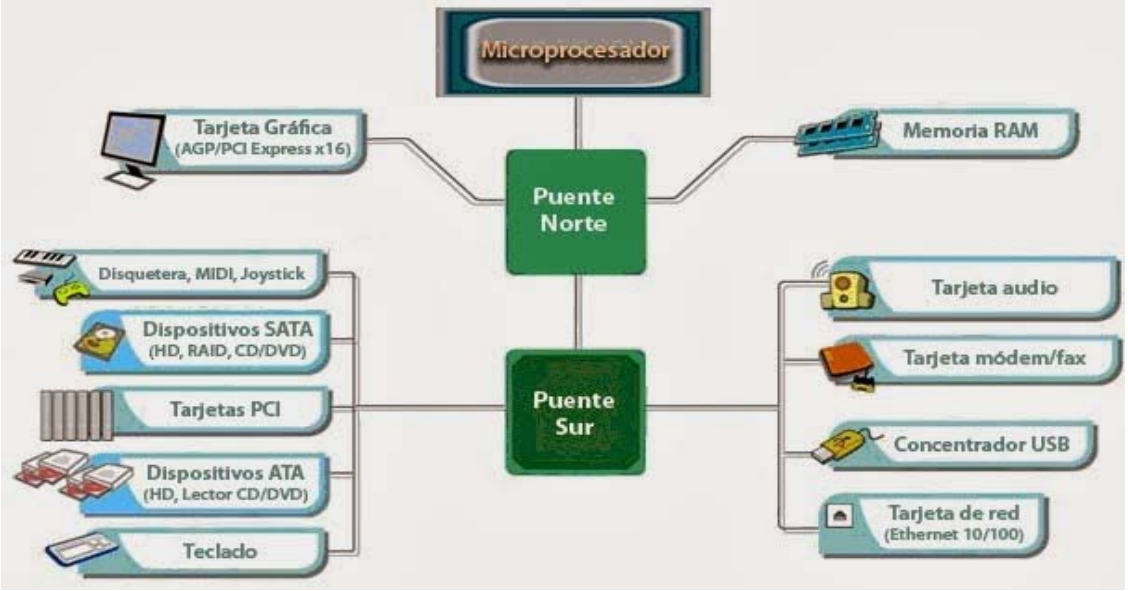


Puentes de comunicación

- **El Puente norte** controla la conexión de la CPU con los componentes de alta velocidad del computador:
  - Memoria RAM
  - Buses de alta velocidad
  - Puente sur

- **El Puente sur** se encarga de coordinar los diferentes dispositivos de entrada/salida y algunas otras funcionalidades de baja velocidad.
- Se comunica indirectamente con la CPU a través del puente norte.
- Un Puente sur actual puede incluir soporte para:
  - Bus PCI
  - Bus ISA
  - Controladores de interrupción
  - Ethernet
  - RAID
  - USB
  - Codec de Audio

Puentes de comunicación



EJEMPLOS I/O BUS

	Firewire	USB 2.0	PCI Express	Serial ATA	Serial Attache SCSI
Intended use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Data width	4	2	2/lane	4	4
Peak bandwidth	100MB/s or 400MB/s	1.5MB/s USB1, a 5GB/s USB 3	250MB/s/lane 1x, 2x, 4x, 8x, 16x, 32x	300MB/s	300MB/s
Hot pluggable	Yes	Yes	Depends	Yes	Yes
Max length	4.5m	5m	0.5m	1m	8m
Standard	IEEE 1394	USB Implementers Forum	PCI-SIG	SATA-IO	INCITS TC T10

INDICE

- Introducción
- Estructura y funciones del sistema de E/S
- E/S Programada
- E/S por interrupciones
  - Concepto de excepción. Gestión de excepciones.
  - Operación de E/S mediante interrupciones
- Controlador de interrupciones
- Sistema de interconexión. Buses
- **Entrada/salida por Acceso Directo a Memoria**

ACCESO DIRECTO A MEMORIA (DMA)

Necesidad del *DMA*

- **E/S controlada por programa**: la *CPU* está dedicada exclusivamente a la *E/S*, transfiriendo los datos a la velocidad determinada por el periférico.
- **Interrupciones**: liberan en buena medida a la *CPU* de la gestión de las transferencias, pudiéndose dedicar a ejecutar otras tareas. La velocidad del periférico se puede ver disminuida si las interrupciones no se atienden con la frecuencia suficiente.
- **En las dos alternativas las transferencias de datos deben pasar a través de la *CPU***.

  => velocidad de transferencia limitada por la velocidad a que la *CPU* atiende el periférico
- Ambos procedimientos manifiestan, pues, limitaciones que afectan a la actividad de la *CPU* y a la velocidad de transferencia.

ACCESO DIRECTO A MEMORIA (DMA)

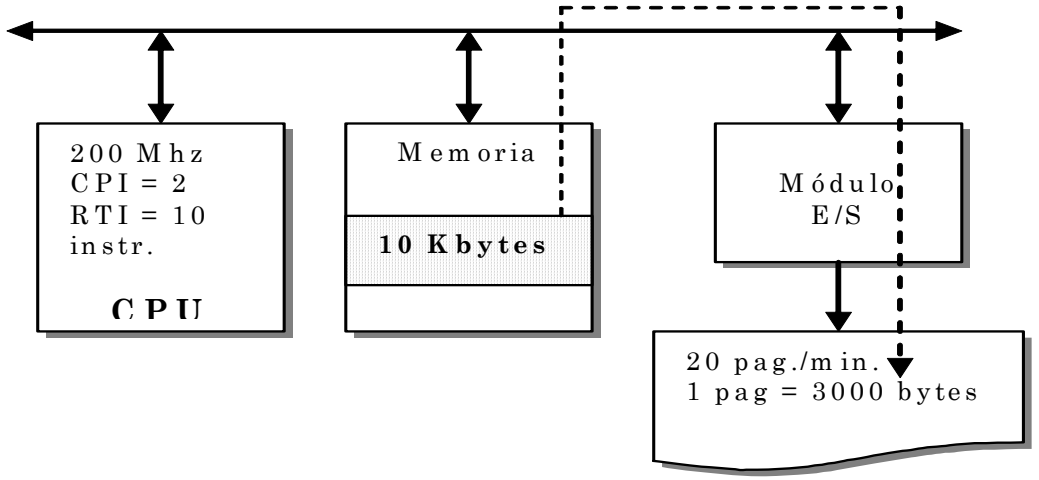
- **Ejemplo 1: Conexión de un periférico lento (impresora láser)**

  La *CPU* opera a *200 Mhz*. y su *CPI* vale *2*.

  RTI = 10 instrucciones

  La impresora opera a *20* páginas/minuto, con *3.000* caracteres (*bytes*) por página.

  Se trata de imprimir un bloque de *10 Kbytes* ubicado en la memoria.

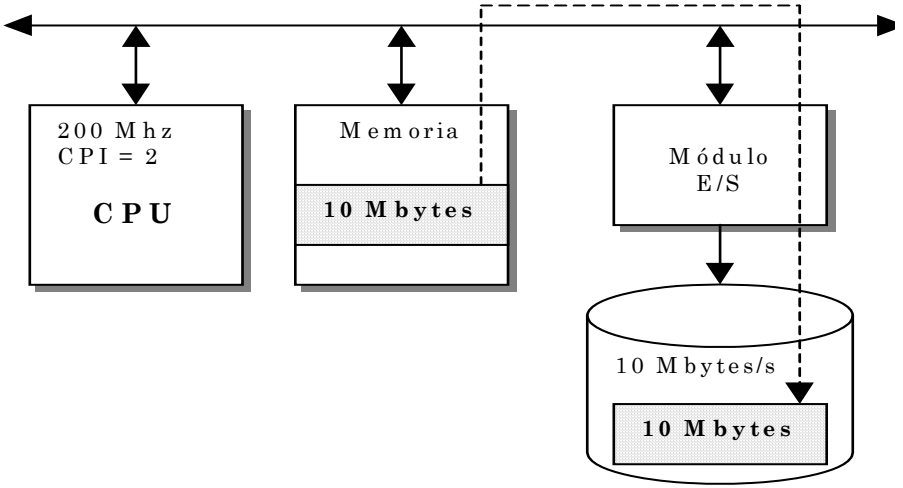


ACCESO DIRECTO A MEMORIA (DMA)

- **Ejemplo 1: Tiempo de CPU dedicado a la E/S**
  - **Conexión por E/S programada**
    - *Tc = 1/Frecuencia = 1/200*10^6 seg. = 5 ns*.
    - Una instrucción tarda en ejecutarse *2 * 5 ns = 10 ns*
    - La impresora opera a 20*3.000 = 60.000 caract./min. = 1.000 caract./seg. = 1 KB/s.
    - Los 10 KB los imprimirá en 10 segundos. ➔ **La *CPU* está ocupada durante *10* seg.**
  - **Conexión por interrupción**
    - La impresora genera una interrupción cuando está preparada para recibir un carácter.
    - Al transferir *10 KB* se producen *10.000* interrupciones➔ejecuta *100.000* instrucciones.
    - Luego **la *CPU* se ocupa durante** *10^5*10 ns = 10^6 ns= 0,001 s.* = ***1 miliseg***.
  - **La *E/S* por interrupción ha reducido en 10.000 veces el tiempo que la CPU** está ocupada en atender la impresora.
  - Sin embargo, **la velocidad de la operación de E/S no ha cambiado**, como era de esperar al estar dominada por la velocidad del periférico.

ACCESO DIRECTO A MEMORIA (DMA)

- **Ejemplo 2: Conexión de un periférico rápido (disco magnético)**
  - La *CPU* opera igual que en caso anterior, es decir, a *200 Mhz* y su *CPI* vale *2*.
  - El disco opera a una velocidad de transferencia de *10 Mbytes/s*.
  - Se trata en este caso de transmitir un bloque de *10 Mbytes* de la memoria al disco



ACCESO DIRECTO A MEMORIA (DMA)

- **Ejemplo 2: Tiempo de CPU dedicado a la E/S**
  - **E/S programada**
    - A la velocidad de *10 Mbytes/s* el disco tarda *1* segundo en recibir los *10 Mbytes*,
    - En *E/S* programada la *CPU* envía un byte cada vez que el disco está preparado para recibirlo ➔ **la *CPU* está ocupada en la transferencia durante *1 segundo***.
  - **E/S por interrupción**
    - Seguimos suponiendo que la rutina de tratamiento ejecuta *10* instrucciones.
    - Como ahora se transmiten 10^7 *bytes* se producirán otras tantas interrupciones ➔ CPU ejecuta en toda la operación $10^7 * 10 = 10^8$ instrucciones.
    - Tiempo de **ocupación *CPU*** = 10^8 instrucciones * *10* ns/instrucción = 10^9 *ns* = ***1 seg.***

ACCESO DIRECTO A MEMORIA (DMA)

- **Conclusiones**
  - En el supuesto 2 las interrupciones no ahorran tiempo de CPU frente a la *E/S* programada porque se ha llegado al límite de velocidad de la línea de interrupción (*10 Mbytes/s*), por encima de esta velocidad la *E/S* por interrupción perdería datos
  - La E/S programada todavía admitiría más velocidad puesto que no tendría que ejecutar las instrucciones de guarda y restauración del contexto.
  - **Para dispositivos rápidos se hace necesario un procedimiento de E/S con menor intervención de la CPU**

ACCESO DIRECTO A MEMORIA (DMA)

- ¿En qué consiste el Acceso Directo a Memoria (DMA)?

  - La técnica de DMA permite la transferencia de datos entre un periférico y la memoria sin intervención de la CPU (salvo en la fase de inicialización de los parámetros de la transferencia)
    - Con interrupciones se evita el bucle de espera pero la transferencia la lleva a cabo el procesador
      - Para un bloque de N bytes, se generan N interrupciones
    - Con DMA toda la transferencia la realiza el controlador de E/S
      - Solo una interrupción al final

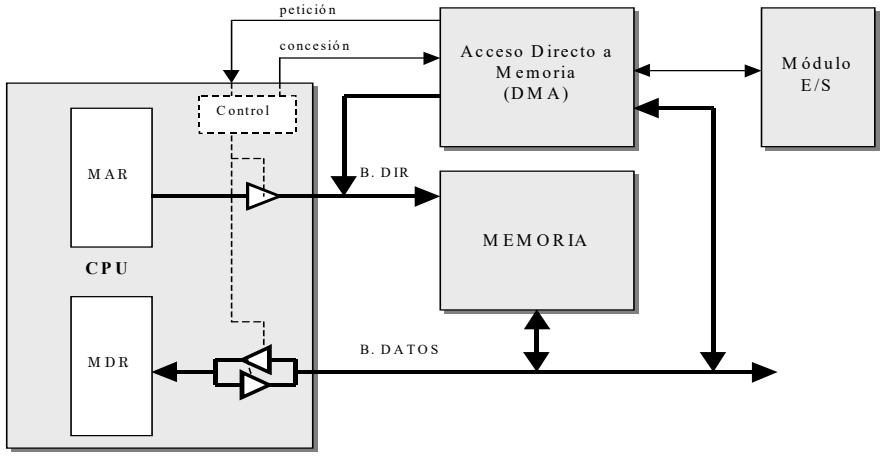
ACCESO DIRECTO A MEMORIA (DMA)

- Se trata de un **módulo con capacidad para leer/escribir directamente en la memoria** los datos procedentes/enviados de/a los dispositivos periféricos.

- **Un DMA es inicializado por la CPU cuando tiene que realizar una operación de E/S.**

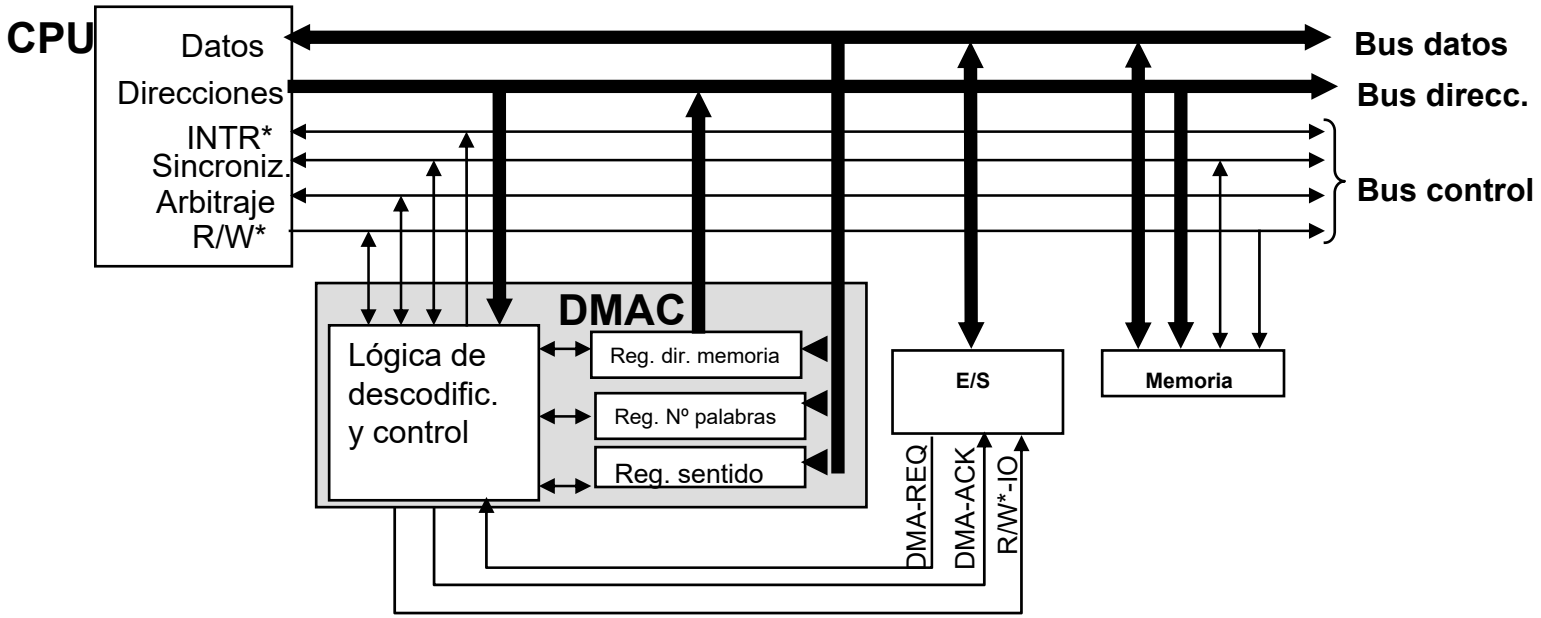
Una vez inicializado opera de la siguiente manera:

1. Solicita a la CPU el permiso para acceder a memoria.
2. La *CPU* pone en estado de alta impedancia su conexión a los buses del sistema.
3. El DMA accede a la memoria
4. Cuando el *DMA* finaliza la operación la *CPU* vuelve a tomar control
5. La velocidad de transferencia sólo está limitada por el ancho de banda de la memoria.



Estructura de un Controlador de DMA

- **Registro de dirección de memoria:** almacena la dirección inicial de memoria y se incrementa/decrementa después de transferir cada palabra
- **Registro de Nº palabras:** almacena el número de palabras a transferir y se decrementa después de transferir cada palabra
- **Registro de sentido:** almacena el sentido de la transferencia (lectura o escritura)



CONTROLADOR DE DMA: SEÑALES

- DMA-REQ: solicitud de servicio DMA
  - La activa el periférico para indicar al DMAC que está listo para transmitir/recibir
- DMA-ACK: Concesión del servicio DMA
  - La activa el DMAC para indicar al periférico que puede realizar la transferencia
  - Antes de activar esta señal el DMAC debe estar en posesión del bus
- R/W*-IO: Sentido de la transferencia para el periférico
  - El periférico no puede utilizar la misma señal de R/W* que la memoria, ya que la transferencia tiene sentidos opuestos desde el punto de vista de uno y otro dispositivo
  - Operación DMA de lectura (memoria $\rightarrow$ periférico)
    - Para la memoria es una lectura: R/W* = 1
    - Para el periférico es una escritura: R/W*-IO = 0
  - Operación DMA de escritura (periférico $\rightarrow$ memoria)
    - Para la memoria es una escritura: R/W* = 0
    - Para el periférico es una lectura: R/W*-IO = 1

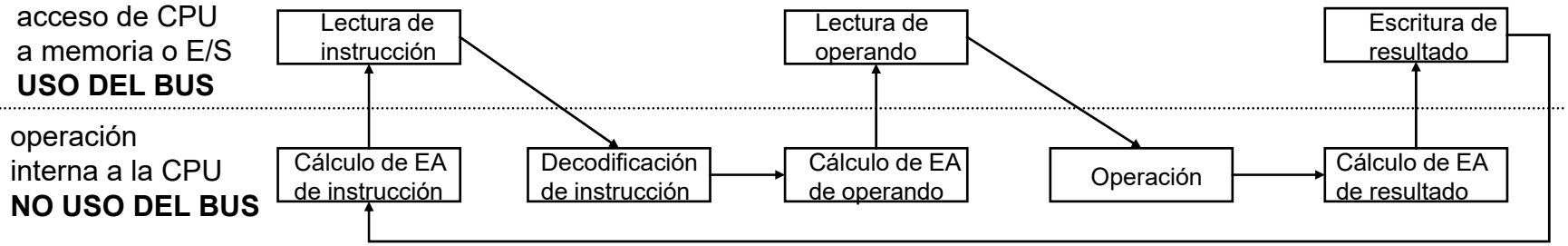
ACCESO DIRECTO A MEMORIA (DMA)

Problema que puede presentar el DMA

- Puede degradar el rendimiento de la CPU si el DMAC hace uso intensivo del bus
  - Si el bus está ocupado en una transferencia DMA, la CPU no puede acceder a memoria para leer instrucciones/datos
- Este problema se reduce con el uso de memoria cache
  - La mayor parte del tiempo, la CPU lee instrucciones de la cache, por lo que no necesita usar el bus de memoria
  - El DMAC puede aprovechar estos intervalos en los que la CPU está leyendo instrucciones de la cache (y por tanto no usa el bus de memoria) para realizar las transferencias
- En caso de computadores sin cache
  - El procesador no utiliza el bus en todas las fases de la ejecución de una instrucción
  - El DMAC puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus para realizar sus transferencias

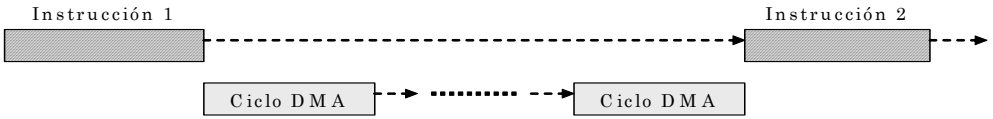
Problema que puede presentar el DMA

- Si el DMAC sólo toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo ⇒ *el rendimiento del sistema no sufrirá degradación alguna*



- Se distinguen, por tanto, tres tipos de transferencias:
  - Ráfaga
  - Robo de ciclo
  - Transparente

- Control del bus en operaciones de DMA
  - **Modo ráfaga**
    - El *DMA* toma control del bus durante la transmisión de un bloque de datos completo.
    - Consigue alta velocidad de transferencia, dejando inactiva la *CPU* durante la operación si esta necesita acceder al bus.

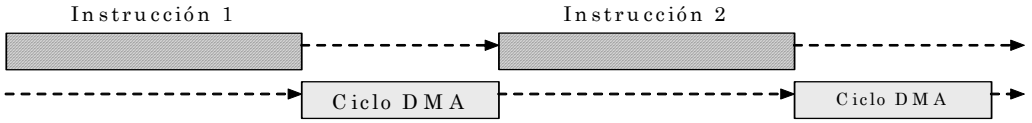


ACCESO DIRECTO A MEMORIA (DMA)

- Control del bus en operaciones de DMA

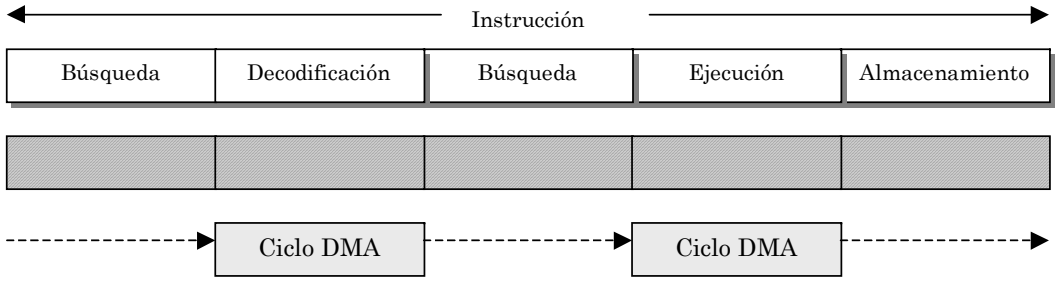
  - **Robo de ciclo**

    - El DMA toma control del bus y lo retiene durante un solo ciclo para transmitir una palabra
    - Es el modo más usual de transferencia. Se dice que el DMA roba ciclos a la CPU.



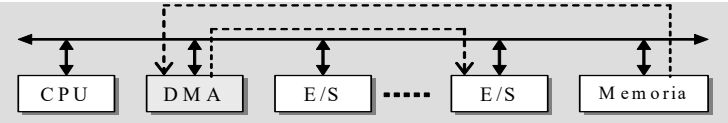
Control del bus en operaciones de DMA

- **Modo transparente**
  - El *DMA* accede al bus sólo en los ciclos en los que la *CPU* no lo utiliza.
  - Esto ocurre en diferentes fases de ejecución de las instrucciones, p.e. decodificación
  - El programa no se ve afectado en su velocidad de ejecución



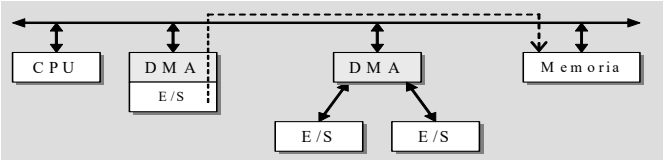
Configuraciones de DMA

- ***DMA* independiente**
  - Todos los módulos comparten el bus del sistema.
  - El *DMA* hace de intermediario entre la memoria y el periférico.
  - Esta configuración es poco eficaz, ya que cada transferencia consume 2 ciclos del bus.



Configuraciones de DMA

- **Integración *DMA-E/S***
  - Proporciona un camino entre el *DMA* y uno o más controladores de E/S
  - Reduce a 1 el número de ciclos de utilización del bus.



ACCESO DIRECTO A MEMORIA (DMA)

- DMA y sistema de memoria
  - Sin el DMA el único acceso a la jerarquía de memoria es a través del procesador.
    - Hay datos que están en alguna de las memorias cache y que pueden no estar actualizados en memoria principal.
  - El controlador de DMA accede directamente a memoria.
  - Problema con la coherencia de cache.