Tema 3. Entrada/salida programada e interrupciones

Índice

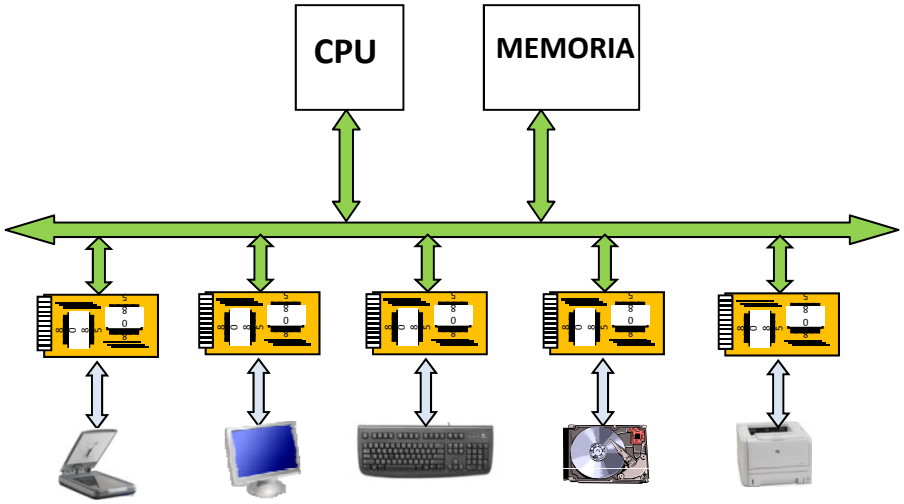
1. Introducción a la E/S

Necesidad de la E/S

➢ Para que un computador pueda ejecutar un programa debe ser ubicado previamente en la memoria, junto con los datos sobre los que opera, y para ello debe existir una unidad funcional de entrada de información capaz de escribir en la memoria desde el exterior.

➢ Para conocer los resultados de la ejecución de los programas, los usuarios deberán poder leer el contenido de la memoria a través de otra unidad de salida de datos.

➢ La unidad de Entrada/Salida (E/S) soporta estas funciones, realizando las comunicaciones del computador con el mundo exterior a través de una variedad de periféricos



1. Introducción a la E/S

Ejemplos de periféricos













1. Introducción a la E/S

Tipos de periféricos

- Dispositivos de **presentación de datos**.
  - Interaccionan con los usuarios, transportando datos entre éstos 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…

1. Introducción a la E/S

Parámetros principales de los dispositivos periféricos

- **Ancho de banda:** cantidad de datos que se pueden transferir por unidad de tiempo (medido en Mbit/s, MB/s…)
  - Los dispositivos de almacenamiento (discos SATA) y red (Gigabit Ethernet) proporcionan un gran ancho de banda

- **Latencia:** tiempo requerido para obtener el primer dato (medido en segundos)
  - La latencia de la mayoría de los dispositivos de E/S es muy alta en comparación con la velocidad del procesador

1. Introducción a la E/S

Ancho de banda y latencia de algunos dispositivos periféricos

➢ Los dispositivos tradicionales de transporte y presentación de datos representan una carga relativamente baja de trabajo para el procesador :

Dispositivos	Ancho de banda
Sensores	1 Bps – 1 KBps
Teclado	10 Bps
Línea de comunicaciones	30 Bps – 200 KBps
Pantalla (CRT)	2 KBps
Impresora de línea	1 – 5 KBps
Cinta (cartridge)	0.5 – 2 MBps
Disco	4.5 MBps
Cinta	3-6 MBps

➢ Los dispositivos de E/S multimedia demandan mayores exigencias de velocidad:

Medio	Ancho de banda	Latencia max. permitida
Gráficos	1 MBps	1 - 5 segundos
Vídeo	100 MBps	20 milisegundos
Voz	64 KBps	50 - 300 milisegundos

2. Estructura y funcionamiento del sistema de E/S

Características generales de los dispositivos periféricos

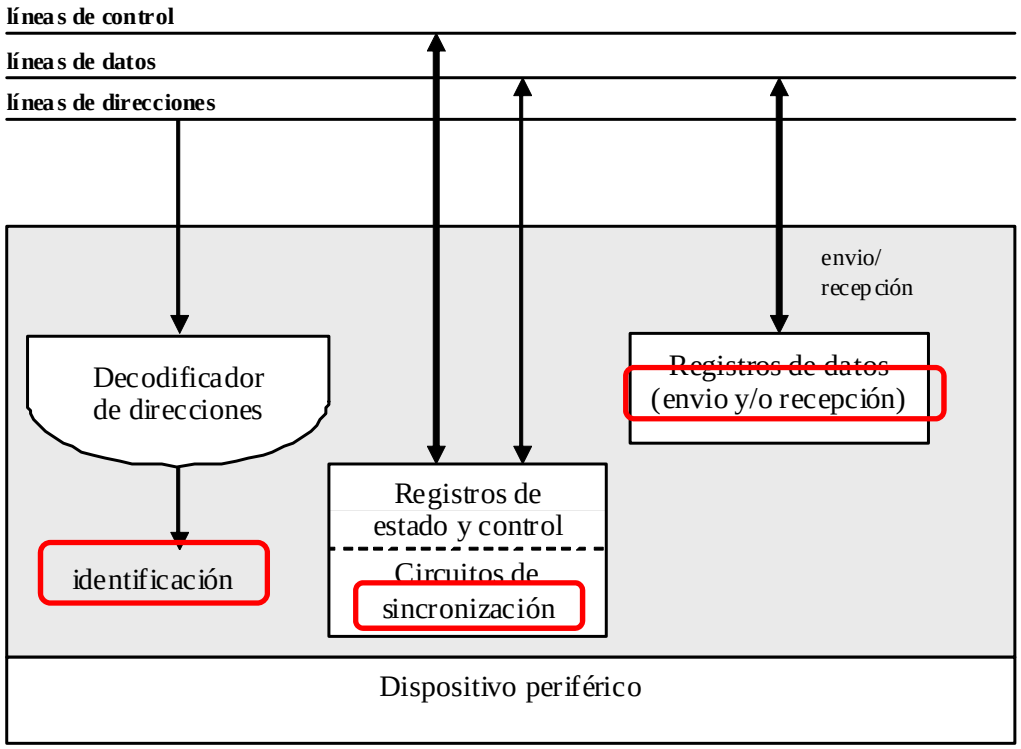
- Los dispositivos periféricos presentan las siguientes características:
  - Tienen formas de funcionamiento muy diferentes entre sí, debido a las diferentes funciones que realizan y a los principios físicos en los que se basan
  - La velocidad de transferencia de datos es también diferente entre sí y diferente de la presentada por la CPU y la memoria
  - Suelen utilizar datos con formatos y longitudes de palabra diferentes

- A pesar de sus diferencias, los dispositivos periféricos presentan las **funciones comunes**:
  - Necesidad de una **identificación** única del dispositivo por parte de la CPU
  - Capacidad de **envío y/o recepción de datos**
  - **Sincronización** de la transmisión, exigida por la diferencia de velocidad de los dispositivos de E/S con la CPU

2. Estructura y funcionamiento del sistema de E/S

Funciones básicas de los dispositivos periféricos

- Las 3 funciones comunes de los dispositivos periféricos determinan su estructura básica:



- Estas funciones básicas se pueden realizar a través del bus del sistema que conecta la memoria y la CPU, o bien se puede utilizar un bus específico para las operaciones de E/S
- Estas alternativas se traducen en dos formas de organización de los espacios de direcciones

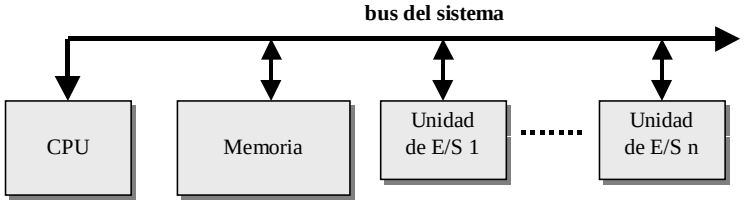
2. Estructura y funcionamiento del sistema de E/S

Espacio de direcciones de E/S (1)

Dos alternativas:

1) Espacio de direcciones unificado con el espacio de memoria

- Existe un único bus al que se conectan los dispositivos de E/S y la memoria
- La memoria y los dispositivos de E/S comparten el espacio de direcciones



- En estos procesadores no existen instrucciones específicas de E/S
- Se utilizan las instrucciones de referencia a memoria tipo load/store (lw/sw)

LOAD Ri, dir_E/S (CPU ← Periférico)
STORE Ri, dir_E/S (Periférico ← CPU)

- Ejemplo: ARM

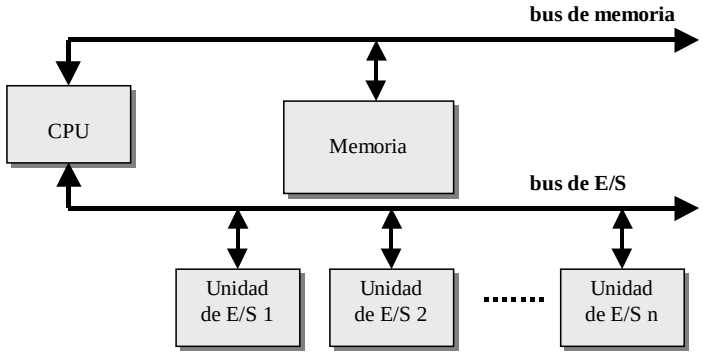
  Cada registro de un módulo de E/S es una dirección de memoria dentro de un rango reservado

2. Estructura y funcionamiento del sistema de E/S

Espacio de direcciones de E/S (2)

2) **Espacio de direcciones independiente del de memoria (buses independientes)**

- Existen buses independientes para la memoria y los dispositivos de E/S



- En estos procesadores sí existe un grupo de instrucciones específico para la E/S

IN dir_E/S, Ri (CPU ← Periférico)
OUT Ri, dir_E/S (Periférico ← CPU)

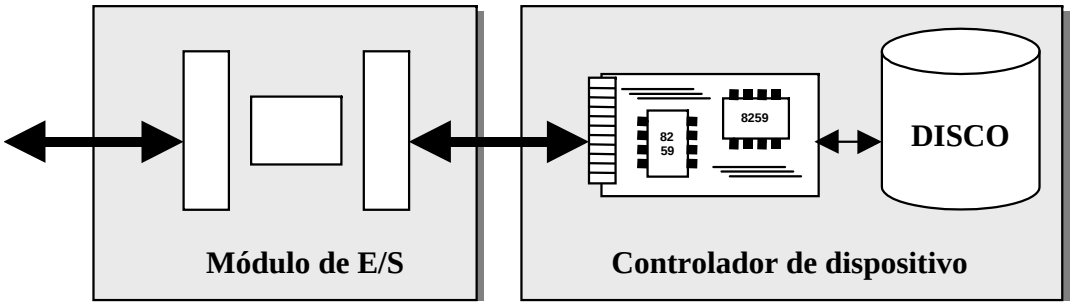
- Ejemplo: Intel x86
  Cada registro de un módulo de E/S es un puerto

2. Estructura y funcionamiento del sistema de E/S

Módulos de E/S y controladores

➢ Para solventar la variedad de dispositivos periféricos, la unidad de E/S de un computador se organiza en torno a dos tipos de elementos:

- Módulos de E/S: soportan las características comunes a muchos dispositivos
- Controladores de dispositivo: son específicos para cada periférico



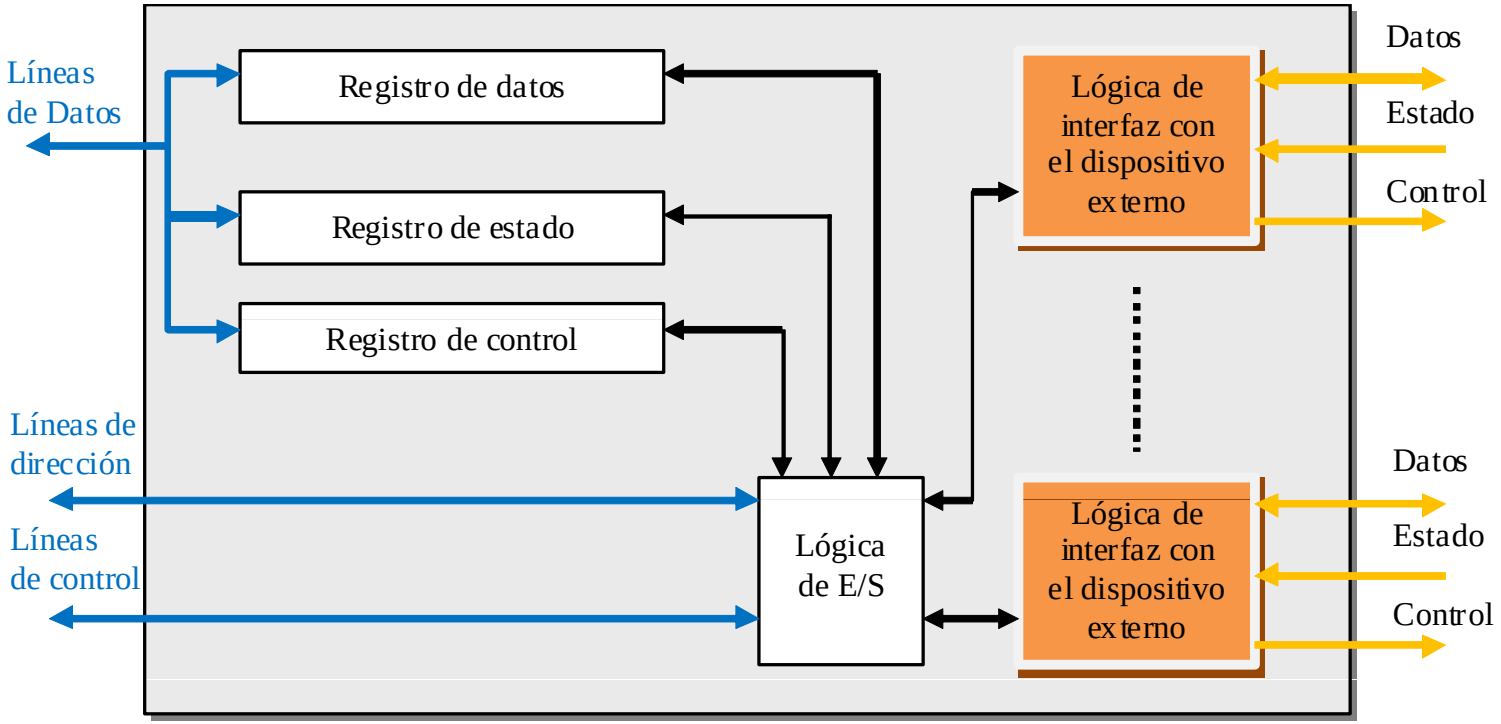
➢ Los **módulos de E/S** permiten que el procesador gestione una amplia gama de dispositivos periféricos

➢ El **controlador** se adapta a las peculiaridades específicas del periférico, actuando sobre elementos electromecánicos (impresoras de línea), sobre elementos ópticos (CD-ROM), o magnéticos (discos), etc.

2. Estructura y funcionamiento del sistema de E/S

Estructura de un módulo de E/S

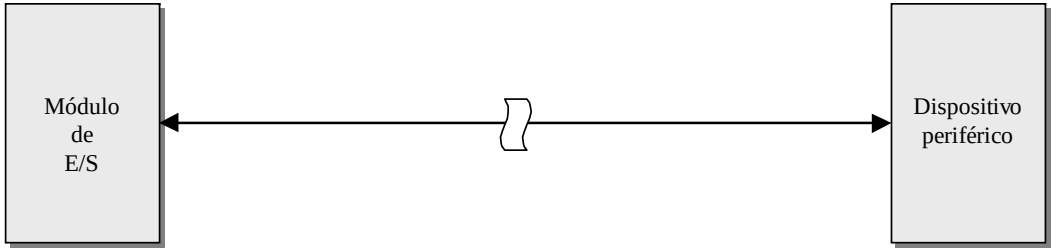
- Se conecta al procesador a través de un conjunto de líneas de *datos*, *dirección* y *control*.
- Los datos que se transfieren se almacenan temporalmente en un *registro de datos*.
- El estado del módulo se refleja en los bits de un *registro de estado*.
- El *registro de control* permite programar diferentes funciones en el módulo.



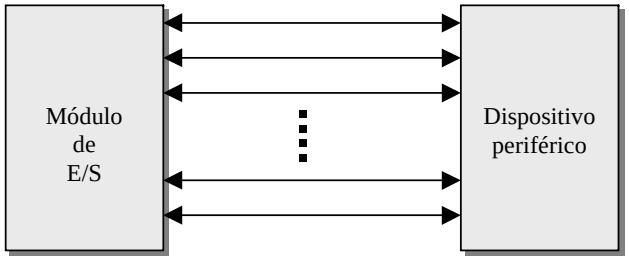
2. Estructura y funcionamiento del sistema de E/S

Módulos de E/S serie y paralelo

- La conexión entre módulo y dispositivo se puede realizar en forma serie o paralela
- **La E/S serie**
  - Se emplea cuando módulo y dispositivo están a una distancia media o larga y el coste del medio de transmisión resulta importante
  - Utiliza una única línea de transmisión
  - La transmisión tiene lugar haciendo que la línea adquiera sucesivamente a lo largo del tiempo el estado de cada uno de los bits constitutivos del mensaje
  - El tiempo asignado a cada bit determina la velocidad de transmisión en bits/segundo



- **La E/S paralela**
  - Se utiliza para conectar módulos de E/S que se encuentran relativamente cerca del dispositivo periférico
  - Utiliza un conjunto de líneas por las que se transmiten en paralelo los bits del mensaje



3. Sincronización de la E/S

Sincronización: mecanismos básicos

- Las diferencias de velocidad entre la CPU y los periféricos de E/S, y la no previsibilidad del tiempo de respuesta de estos últimos, hace necesario un mecanismo de sincronismo que permita coordinar adecuadamente las transferencias de datos entre ambas unidades

- Antes de enviar/recibir datos a/desde un periférico hay que asegurarse de que el dispositivo está preparado para realizar la transferencia, es decir, hay que sincronizarlo

- Existen dos mecanismos básicos para sincronizar las operaciones de E/S con las de la CPU: sincronización por programa (E/S programada) y sincronización por interrupción

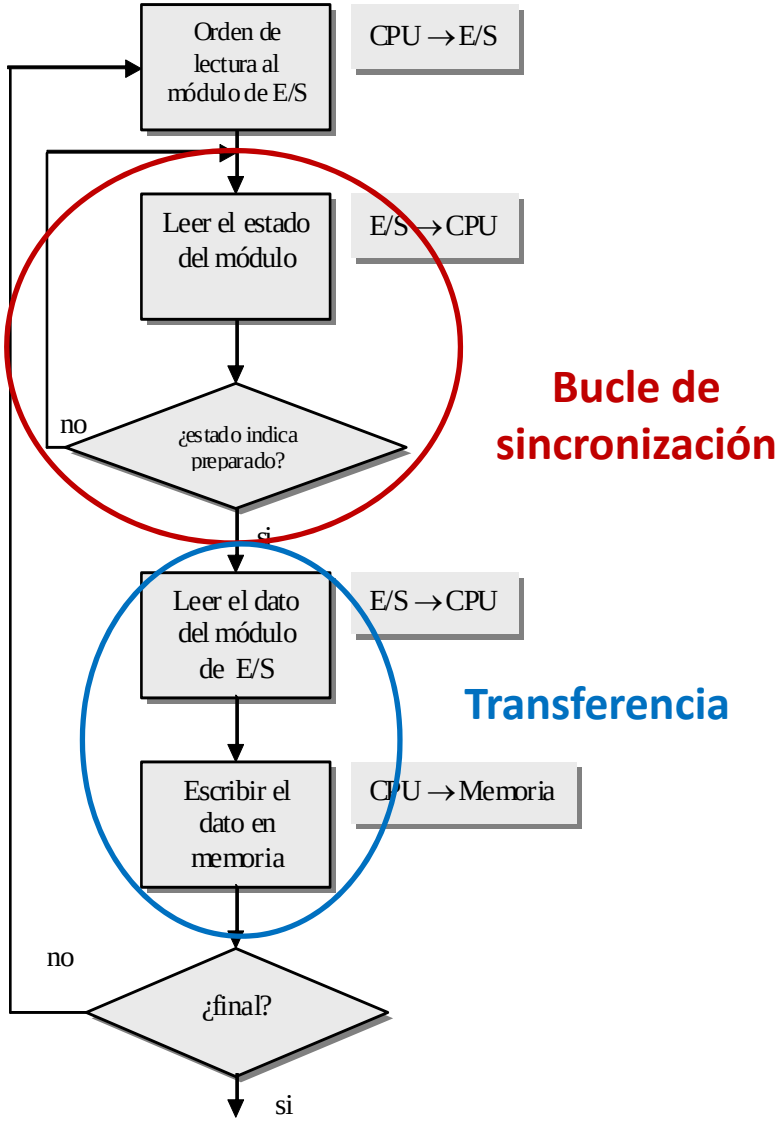
- **La E/S programada** es la más sencilla de implementar, pero presenta el inconveniente de la pérdida de tiempo: el computador no realiza trabajo útil en el bucle de espera

- **La E/S por interrupción** aprovecha mejor el tiempo de CPU, permitiendo la ejecución concurrente de un programa principal y la operación de E/S

4. E/S programada

Operación básica

- Cada vez que la CPU quiere realizar una transferencia:
  1. Entra en un **bucle** en el que lee una y otra vez el **registro de estado** del periférico **(encuesta o "polling")** hasta que esté preparado para realizar la transferencia
  2. Realiza la **transferencia**



4. E/S programada

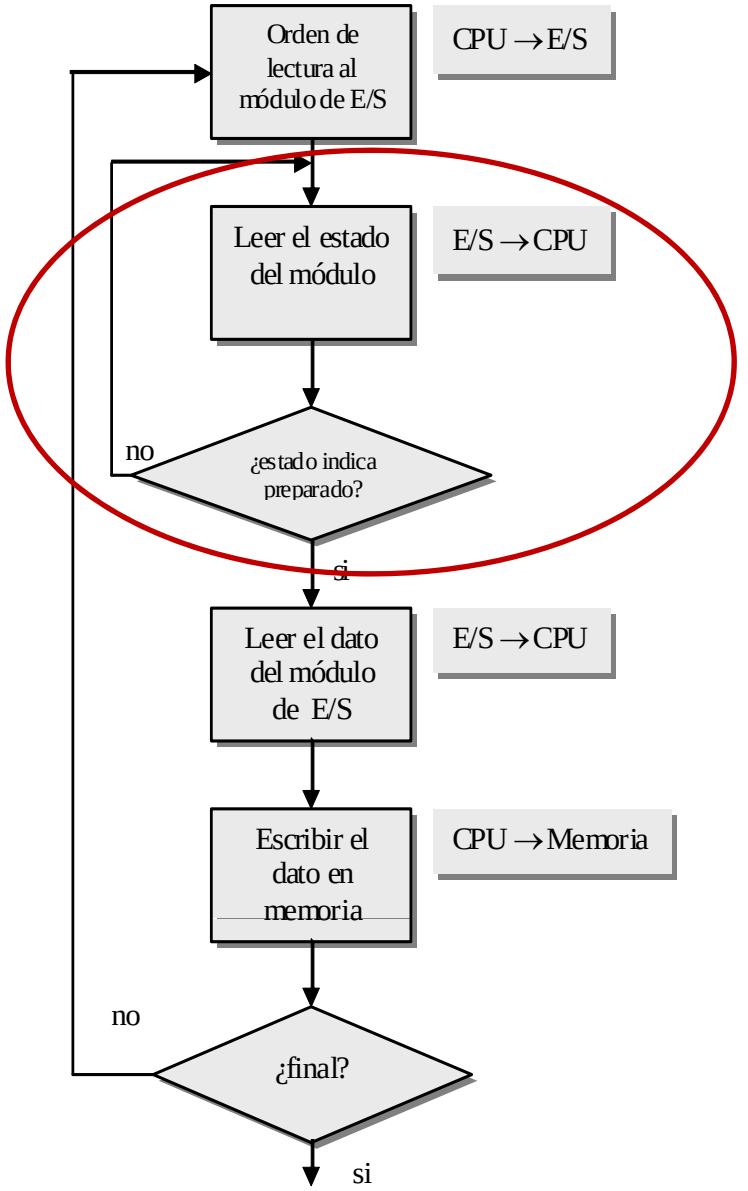
Problemas de la E/S programada

- La CPU no hace trabajo útil durante el bucle de espera
  - Con dispositivos lentos el bucle podría repetirse miles/millones de veces
- La dinámica del programa se detiene durante la operación de E/S
  - Ejemplo: en un vídeo-juego no se puede detener la dinámica del juego a espera 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

4. E/S programada

¿Hay margen de mejora?

- E/S programada:
  - 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**



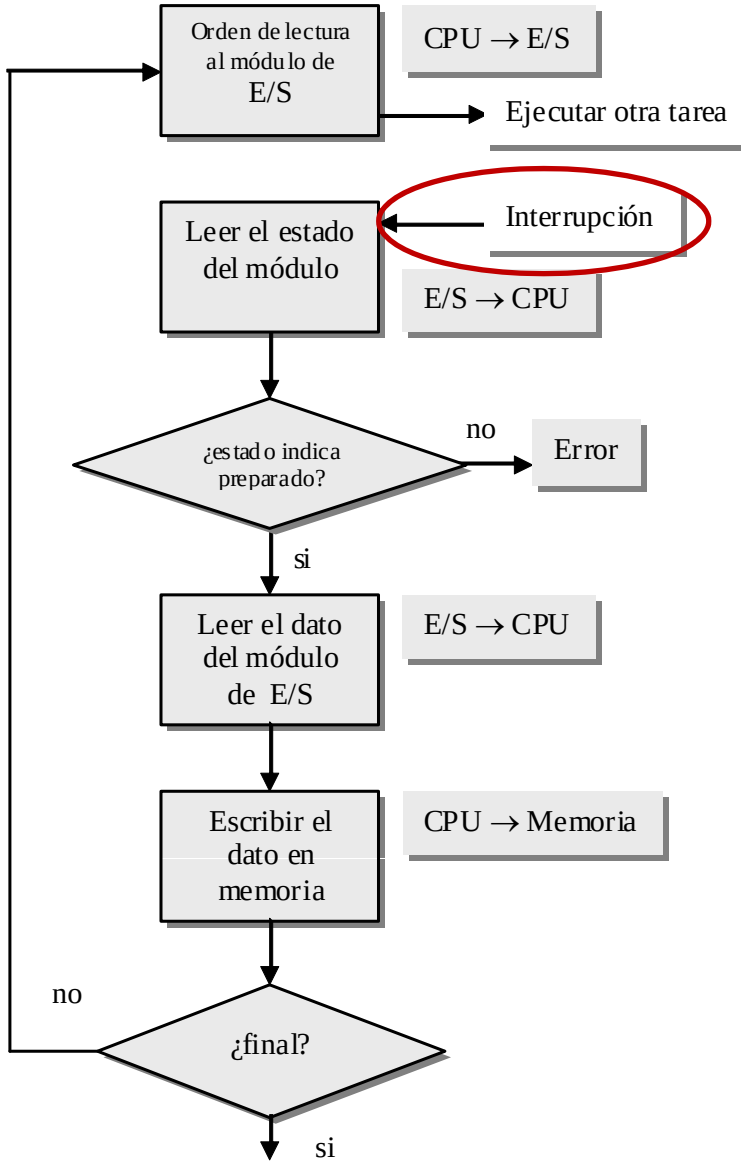
5. E/S por interrupción

E/S mediante interrupción

1. La CPU hace la **petición de operación de E/S** y pasa a ejecutar otros programas
   - ➡ No existe bucle de espera

2. Cuando un periférico está listo para transmitir se lo indica a la CPU activando una **LÍNEA DE PETICIÓN INTERRUPCIÓN**

3. 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**

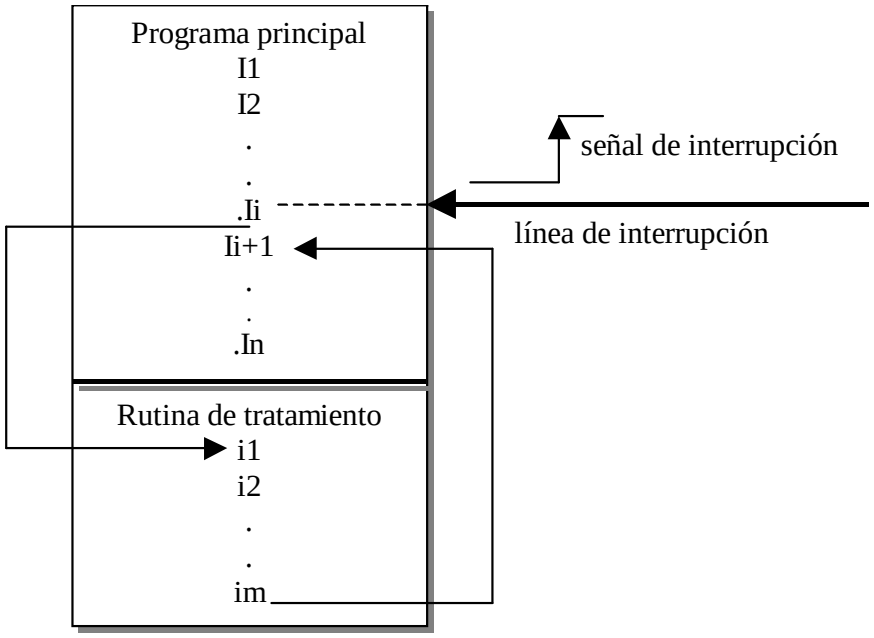


5. E/S por interrupción

Concepto de interrupción (1)

- Una interrupción viene determinada por la ocurrencia de una señal externa que provoca la bifurcación a una dirección especifica de memoria, interrumpiendo temporalmente la ejecución del programa principal

- A partir de esa dirección se encuentra la *rutina de tratamiento* que se encarga de realizar la operación de *E/S* propiamente dicha, devolviendo después el control al punto interrumpido del programa principal



5. E/S por interrupción

Concepto de interrupción (2)

- Una interrupción **es como un salto a subrutina** (*rutina de tratamiento*) ocasionado por una señal externa, y no por una instrucción del programa.
- Las interrupciones **eliminan los tiempos muertos de consulta** de la *E/S* programada
- La implementación de un sistema de interrupciones **implica introducir una fase de consulta de las líneas de interrupción al final de la ejecución de cada instrucción**
- En un procesador sin sistema de interrupciones, se podría conseguir un efecto similar **introduciendo una instrucción de consulta y la correspondiente de salto sobre el valor de la consulta, detrás de cada instrucción útil del programa**
  - Esta solución garantizaría la respuesta al dispositivo de *E/S* en el momento que pasa a estado disponible, al tiempo que la *CPU* ejecuta instrucciones útiles del programa
  - El precio a pagar sería el recargo introducido por la ejecución de las parejas de instrucciones (consulta y salto) detrás de cada instrucción útil del programa
- Un **sistema de interrupciones** podemos verlo como la **integración en hardware del supuesto software anterior**, es decir, la integración de la consulta y posible salto dentro de la ejecución de cada instrucción del repertorio

5. E/S por interrupción

Generalización de las interrupciones: excepciones

- **Interrupción de E/S:**
  - Señal **externa** al procesador que provoca la detención del programa en curso para que la CPU realice otra actividad. El mecanismo de interrupciones fue motivado por la E/S aunque posteriormente se generalizó a eventos producidos internamente en el procesador
- **Extensión de una interrupción: Excepción:**
  - Evento inesperado que provoca un **cambio en el flujo de control normal** del programa
- **Tipos de excepciones:**
  - 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 de E/S)
  - Excepciones **software** (trap): producidas por la ejecución de instrucciones de la CPU

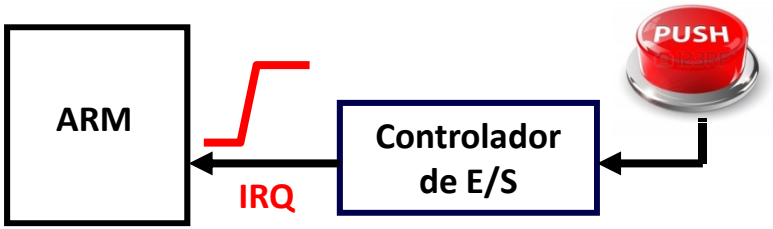
5. E/S por interrupción

Diferencias entre excepciones e interrupciones de E/S

- 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

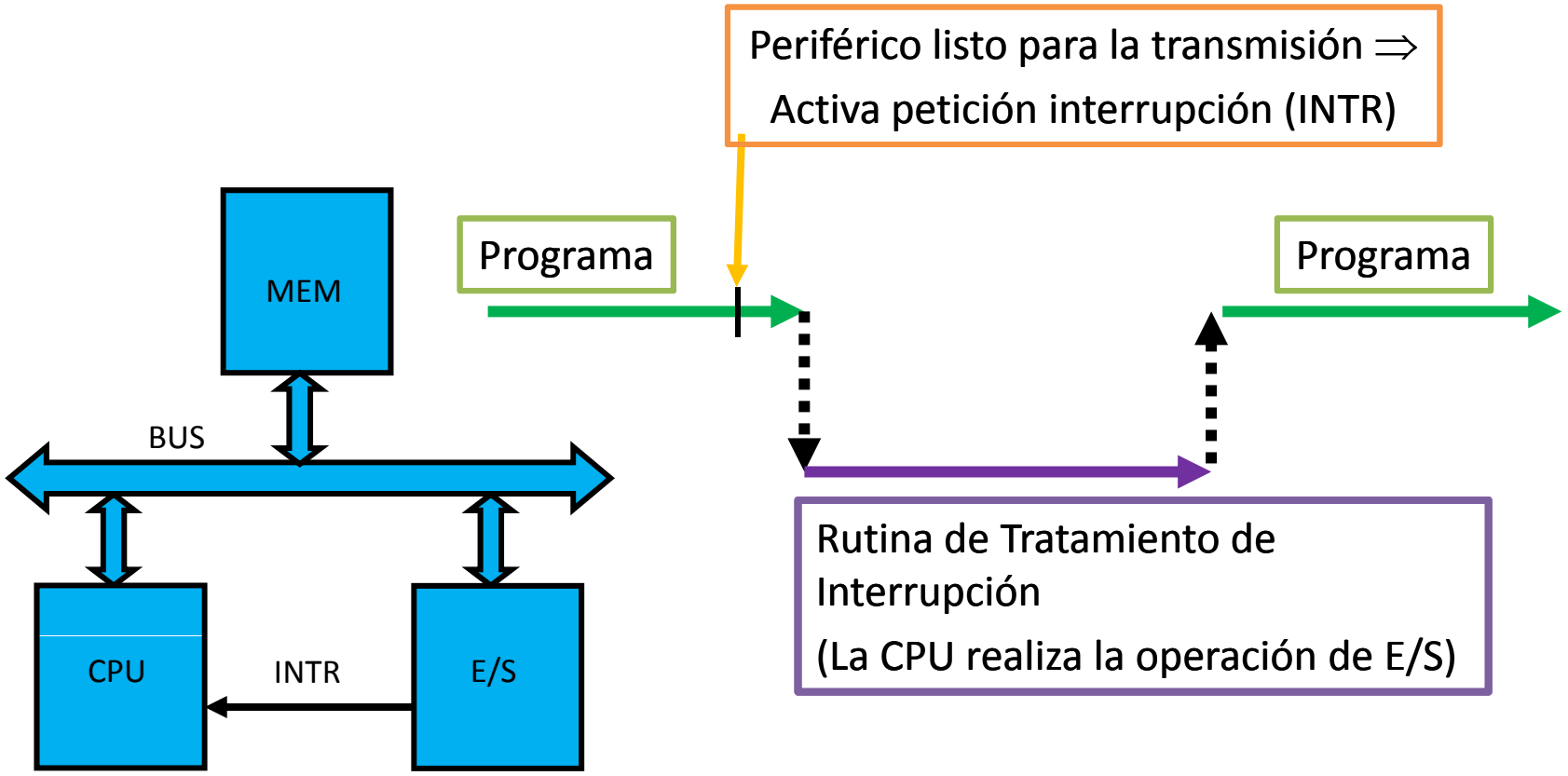
```
ldr r0,=0x0a333333
str r1,[r0]  @ genera Data Abort
```

- 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 de interrupción debida a una operación de entrada/salida:
    - En el ejemplo de lectura de pulsadores se podría programar el controlador para que cada vez que se pulse el botón genere una interrupción por IRQ



5. E/S por interrupción

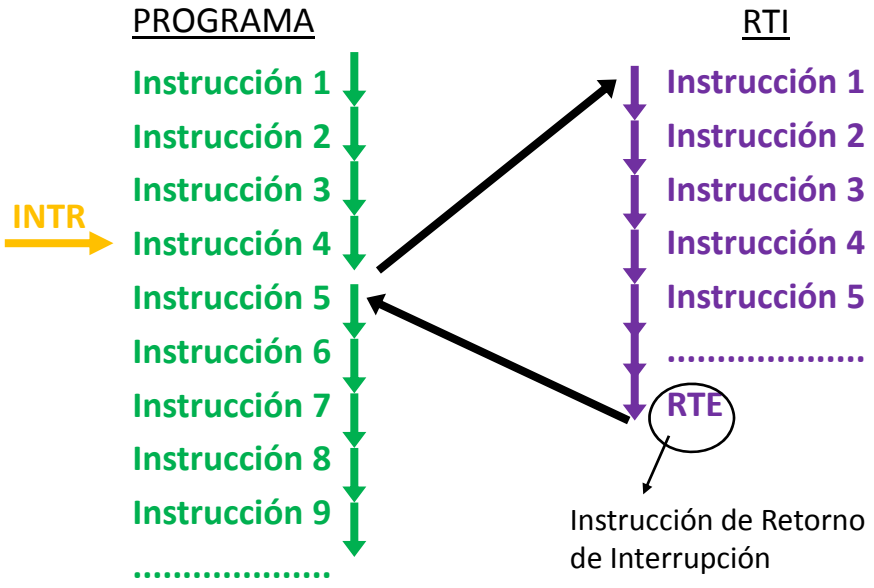
Flujo de ejecución de instrucciones cuando se produce una interrupción



5. E/S por interrupción

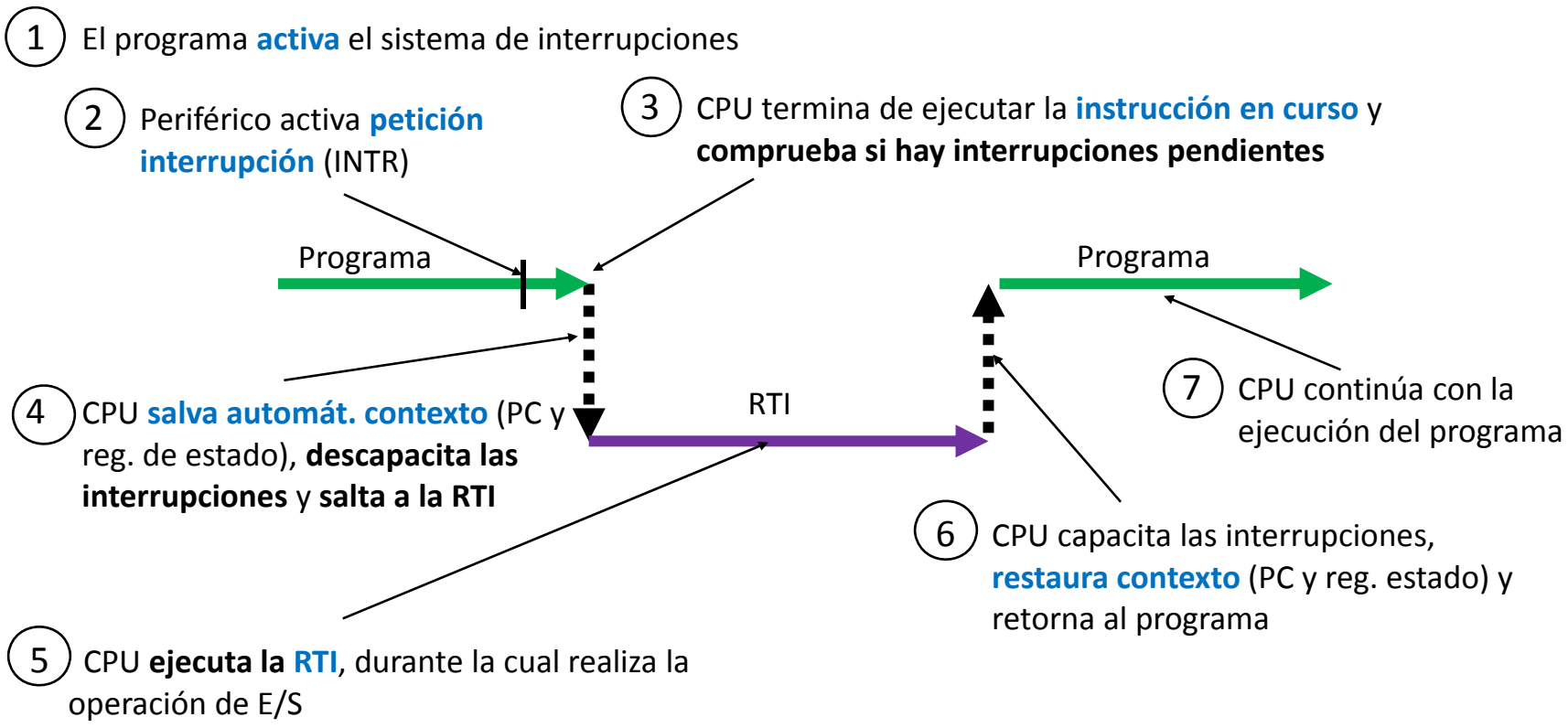
Rutina de tratamiento de interrupción

- **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
    - Debemos guardar 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



5. E/S por interrupción

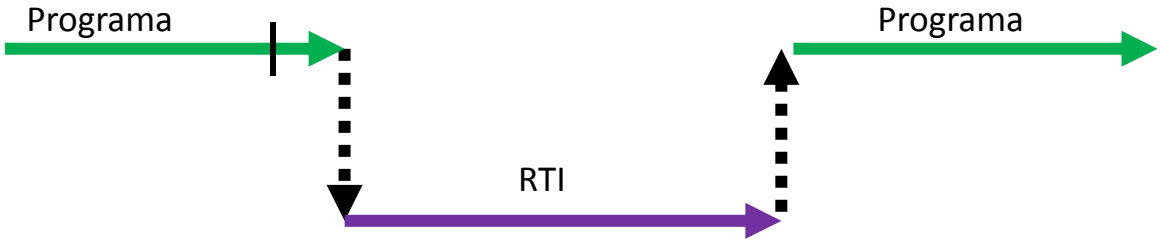
Eventos en el tratamiento de una interrupción



5. E/S por interrupción

Eventos en el tratamiento de una interrupción

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

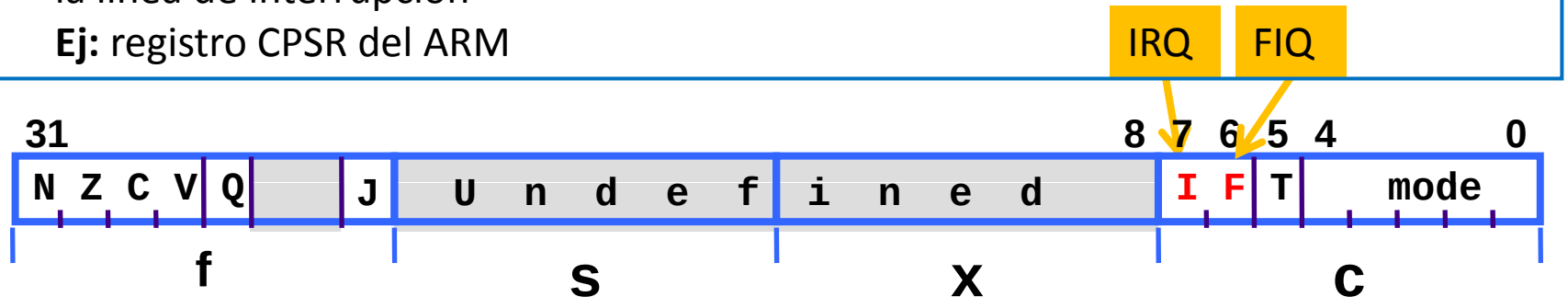


Para que pueda producirse una interrupción la CPU debe permitir que un dispositivo le interrumpa, **capacitando las interrupciones**:

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

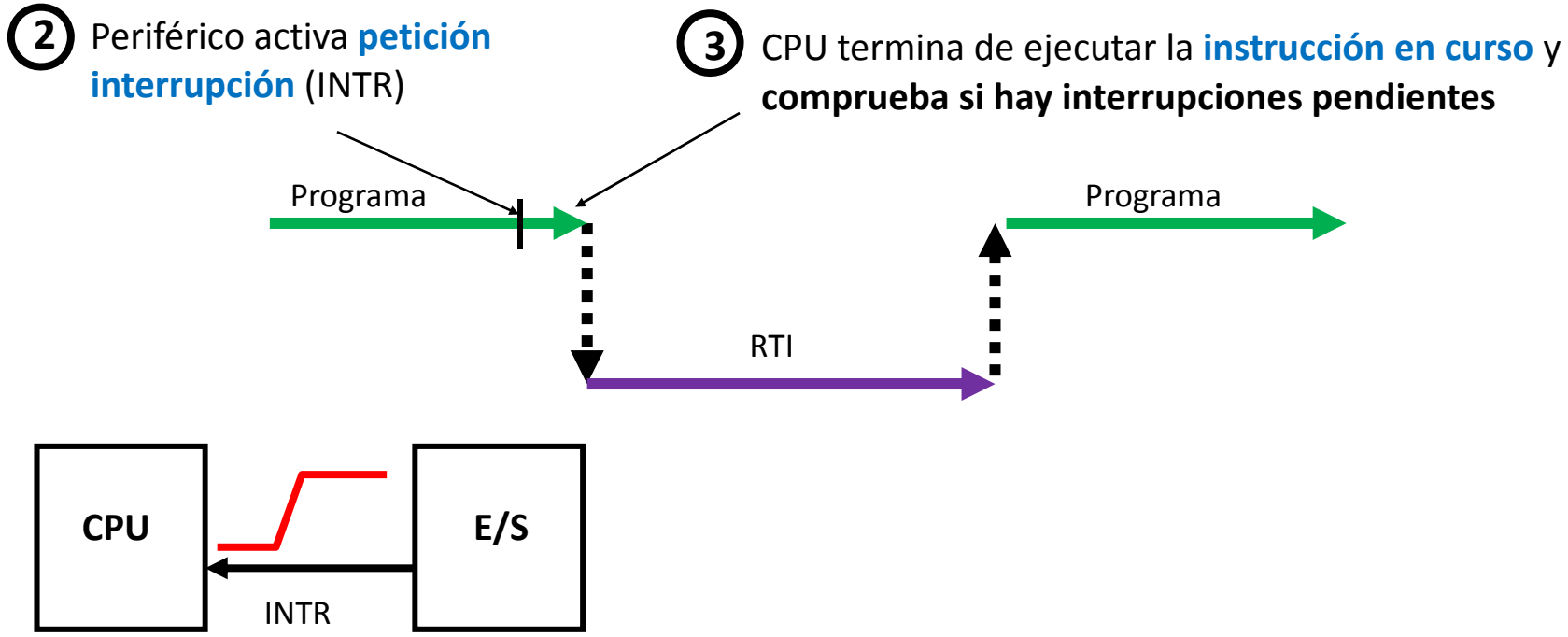
Globalmente: en el registro de estado de la CPU se activa el bit correspondiente a la línea de interrupción

Ej: registro CPSR del ARM



5. E/S por interrupción

Eventos en el tratamiento de una interrupción



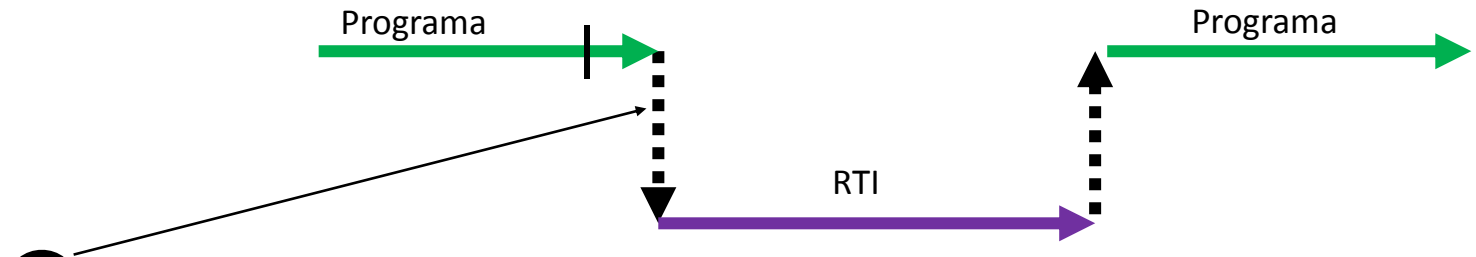
5. E/S por interrupción

Comprobación de peticiones de interrupción pendientes

- 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 reg. 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 de la CPU
      - Reg. 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
    - Interrupciones muy prioritarias
      - Por ejemplo, una interrupción por fallo de página, en la que hay que acceder a disco para traer los operandos en memoria de la instrucción

5. E/S por interrupción

Eventos en el tratamiento de una interrupción



4 CPU salva automát. contexto (PC y reg. de estado), **descapacita las interrupciones** y **salta a la RTI**

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

Descapacitar las interrupciones:
En el registro de estado se inhabilitan **automáticamente** las interrupciones por la línea INTR

¿Por qué?

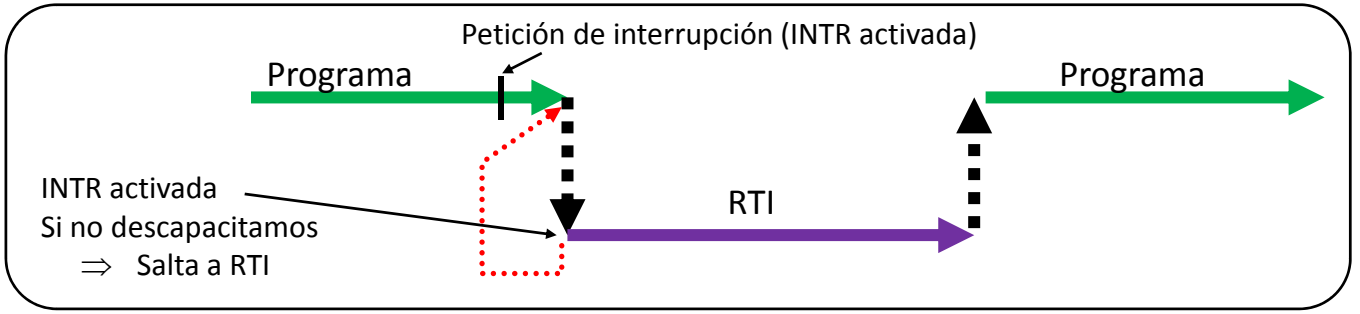
Saltar a RTI:
Se obtiene la dirección de la RTI que corresponda al periférico que ha interrumpido

¿Cómo se identifica al periférico?

5. E/S por interrupción

Inhibición o descapacitación de las interrupciones

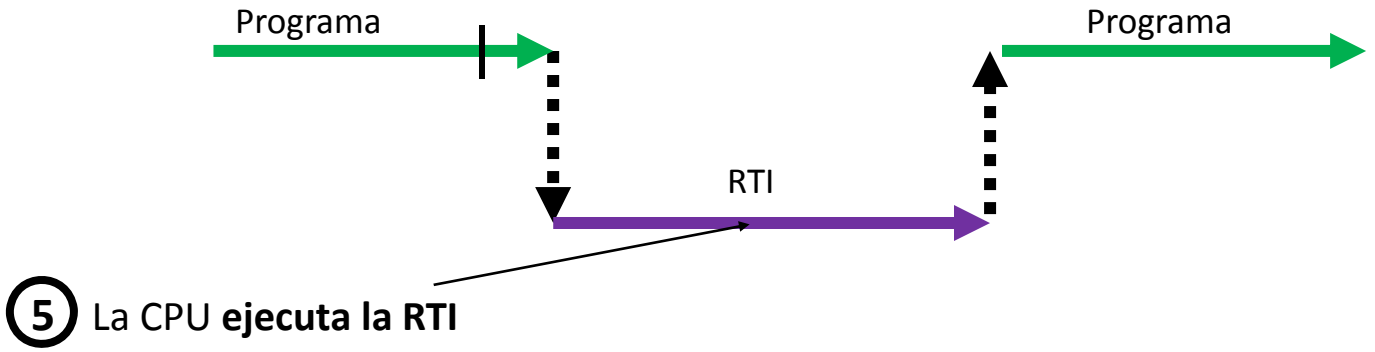
- **Antes de saltar a la RTI es necesario inhibir o descapacitar las interrupciones**
  - **Motivo: Si no se inhiben la CPU puede entrar en un bucle infinito**
    - Cuando se entra en la RTI el periférico todavía no ha desactivado su petición
    - Si las interrupciones están capacitadas ⇒ la CPU detecta una interrupción pendiente y vuelve 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**
    - **Descapacitación global**
      - Se inhiben todas las interrupciones ⇒ ningún otro periférico podrá interrumpir durante la ejecución de la RTI
    - **Descapacitación o enmascaramiento selectivo**
      - Cuando hay varios niveles de interrupción se pueden descapacitar las interrupciones por el nivel que interrumpe, pero no necesariamente por el resto de niveles
      - Véase interrupciones multinivel y anidamiento de interrupciones

5. E/S por interrupción

Eventos en el tratamiento de una interrupción

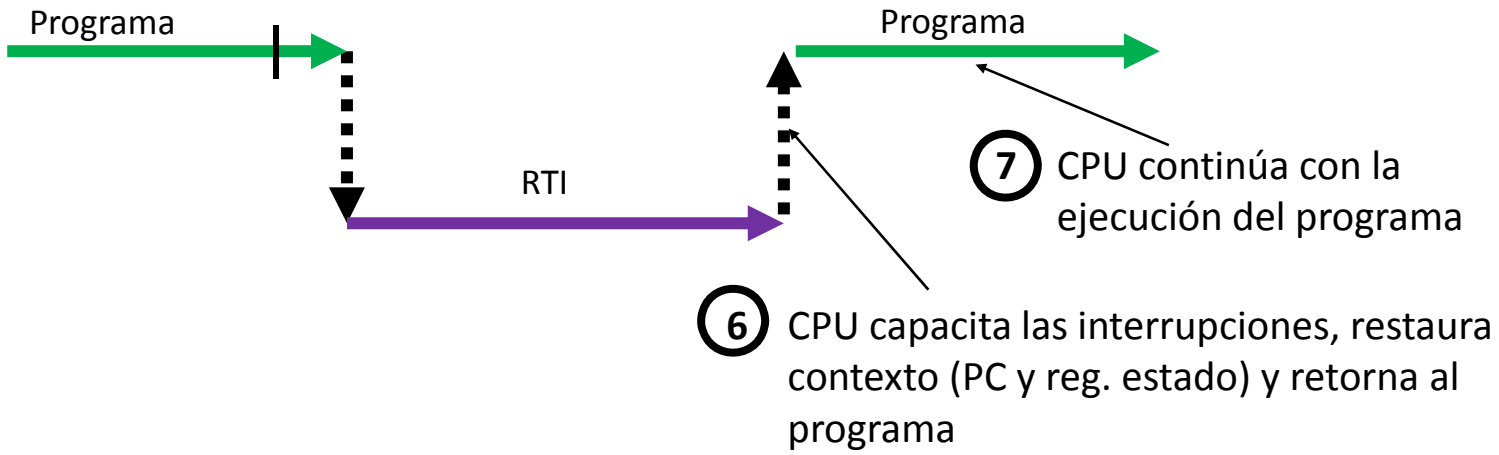


la RTI:

- Salva en pila todos los registros que utiliza (manual)
- Informa al periférico que se ha reconocido su interrupción
  - **¿Cómo?**
    - ⇒Por **software**: accediendo al registro de estado o de datos del controlador
    - ⇒Por **hardware:** activando una señal de reconocimiento de interrupción (INTA)
  - **Una vez informado el periférico desactiva INTR**
- Realiza la operación de E/S con el periférico
- Restaura los registros de datos/direcciones
- Ejecuta la instrucción de retorno de interrupción (RTE)

5. E/S por interrupción

Eventos en el tratamiento de una interrupción



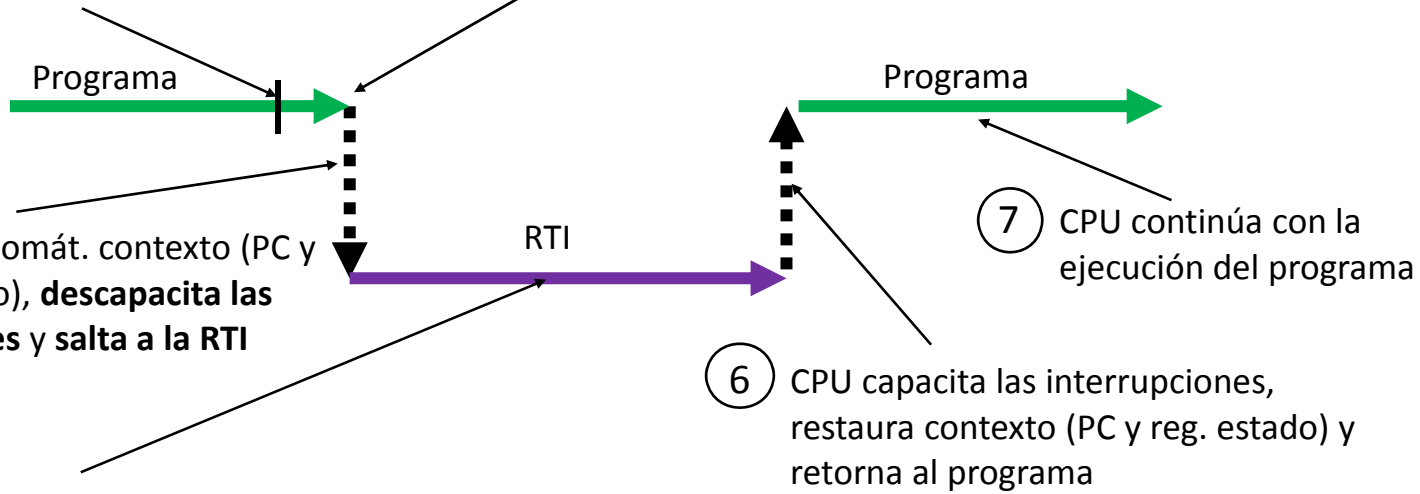
Cuestiones planteadas

- ¿Cómo **se informa al periférico** que se ha reconocido su interrupción?
- ¿Cómo **se identifica la fuente de interrupción** y, por tanto, la RTI que hay que ejecutar?
- ¿Qué ocurre si se produce una segunda interrupción durante la ejecución de la RTI?
  ⇒ ***Interrupciones multinivel y anidamiento de interrupciones***

5. E/S por interrupción

Eventos en el tratamiento de una interrupción

1. El programa activa el sistema de interrupciones
2. Periférico activa petición interrupción (INTR)
3. CPU termina de ejecutar la instrucción en curso y **comprueba si hay interrupciones pendientes**



4. CPU salva automát. contexto (PC y reg. de estado), **descapacita las interrupciones** y **salta a la RTI**
5. CPU ejecuta la RTI, durante la cual:
   - 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 los registros de datos/direcciones
   - Ejecuta la instrucción de retorno de interrupción (RTE)
6. CPU capacita las interrupciones, restaura contexto (PC y reg. estado) y retorna al programa
7. CPU continúa con la ejecución del programa

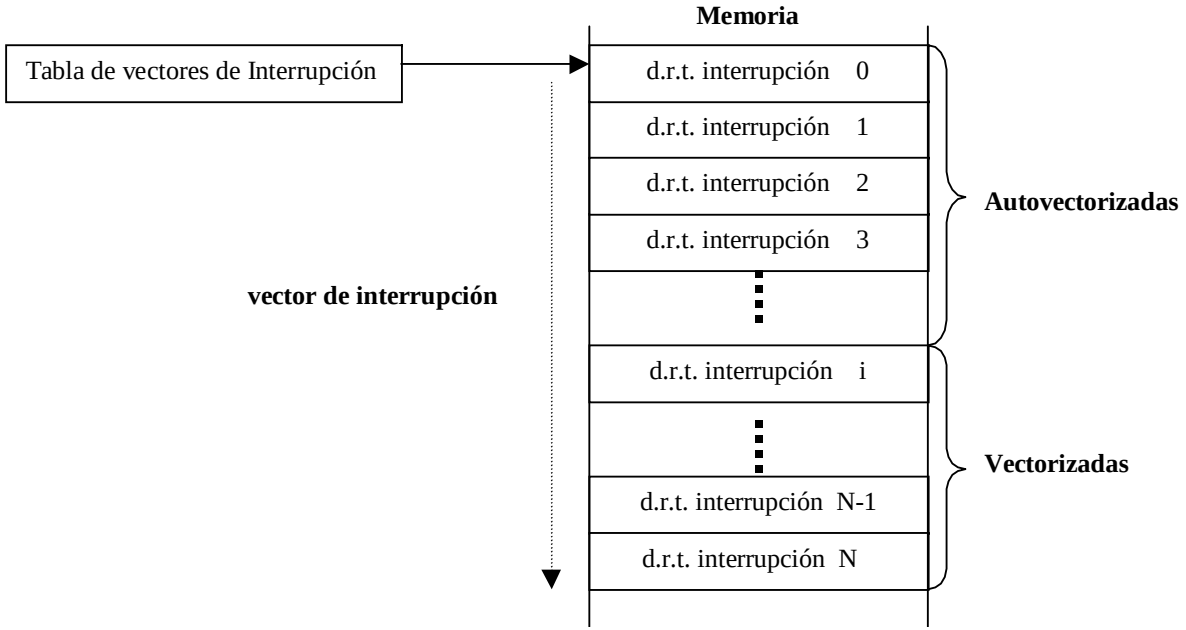
Cuestiones planteadas

- **¿Cómo se informa al periférico que se ha reconocido su interrupción?**
- **¿Cómo se identifica la fuente de interrupción y, por tanto, la RTI que hay que ejecutar?**
- **¿Qué ocurre si se produce una segunda interrupción durante la ejecución de la RTI?**
  ⇒ *Interrupciones multinivel y anidamiento de interrupciones*

5. E/S por interrupción

Dirección de la rutina de tratamiento: vectores de interrupción

➢ Cada computador dispone de una tabla en memoria con las direcciones de las rutinas de tratamiento asociadas a cada interrupción, denominada **tabla de vectores de interrupción**



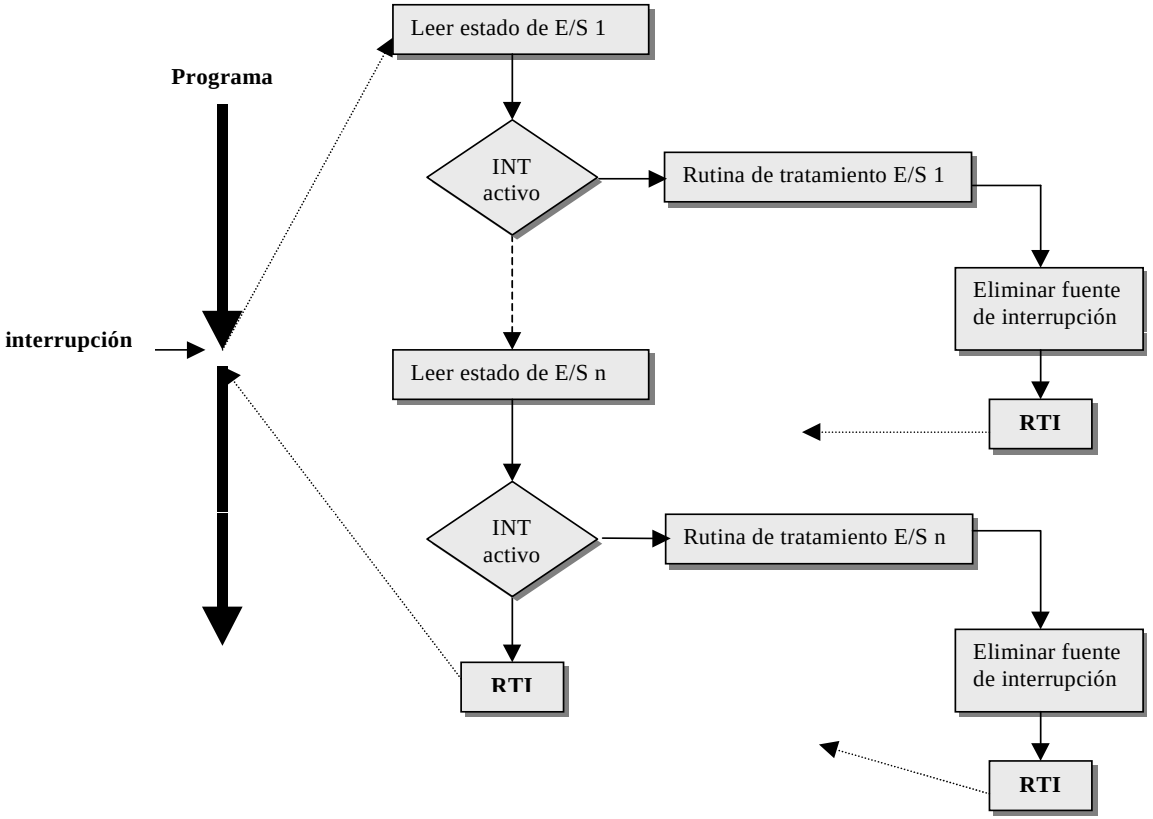
➢ En función de la forma de obtener el vector de interrupción existen dos tipos de líneas de interrupción:

- **Interrupciones autovectorizadas (no vectorizadas)**: el vector de interrupción es fijo, una posición de memoria asociada a la línea de interrupción.
- **Interrupciones vectorizadas**: el vector de interrupción o parte de él lo suministra el propio periférico cuando se le reconoce la interrupción.

5. E/S por interrupción

Interrupciones autovectorizadas

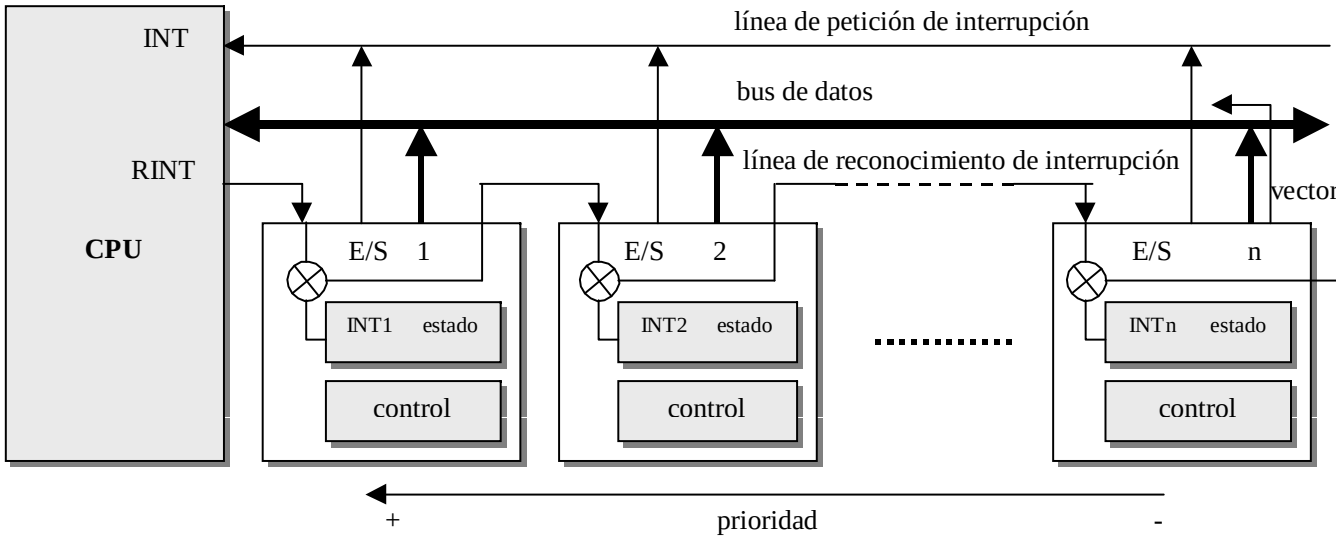
- Cuando se conectan a la misma línea de interrupción autovectorizada (no vectoriazadas) más de un dispositivo periféricos**, la *CPU* debe identificar por software el dispositivo** que produce la interrupción, y establecer el orden de prioridad en caso de que sean más de uno
- La identificación se realiza consultando los registros de estado de cada módulo de *E/S*
- **El orden de la consulta determina el orden de prioridad**



5. E/S por interrupción

Interrupciones vectorizadas

- Disponen de dos líneas, una de petición (INT) y otra de reconocimiento (RINT).
- El vector de interrupción es generado por el dispositivo que produce la interrupción.
- Utiliza un mecanismo de ***daisy chaining*** para transmitir la señal de reconocimiento.
  - El dispositivo de *E/S* genera la petición de interrupción activando *INT*.
  - La *CPU* la reconoce activando *RINT*.
  - Los módulos que reciben *RINT* y no han interrumpido, la transmiten al siguiente módulo.
  - El módulo que recibe *RINT* y realizó petición pone el vector de interrupción *n* en el bus de datos.
  - A partir del vector *n* la *CPU* bifurca a la rutina de tratamiento correspondiente al dispositivo



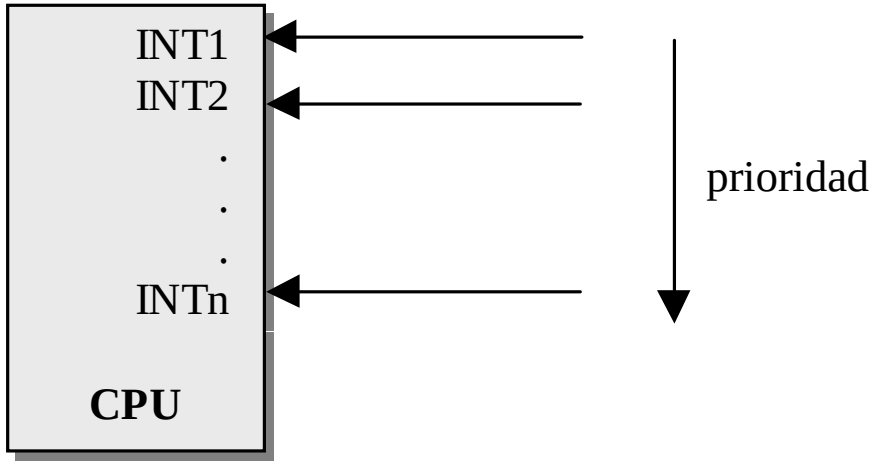
5. E/S por interrupción

Prioridades (1)

- Interrupciones **no vectorizadas**: la prioridad la establece el software de consulta con el orden de recorrido de todos los dispositivos de E/S salida conectados a una línea.

- Interrupciones **vectorizadas**: la prioridad la determina el orden de proximidad a la *CPU* establecido por la línea de retorno de interrupción Es decir, se establece por hardware.

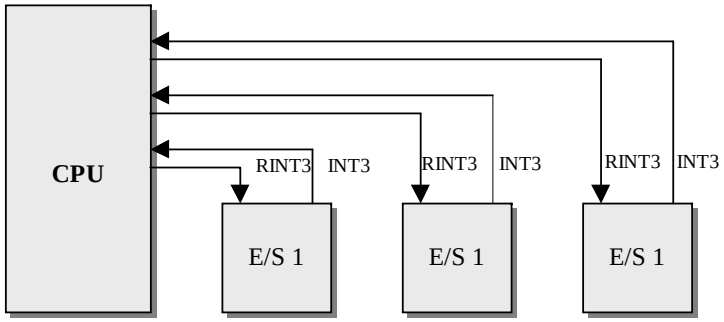
- Cuando existen más de una línea de interrupción, la prioridad viene establecida por hardware en el diseño del procesador:



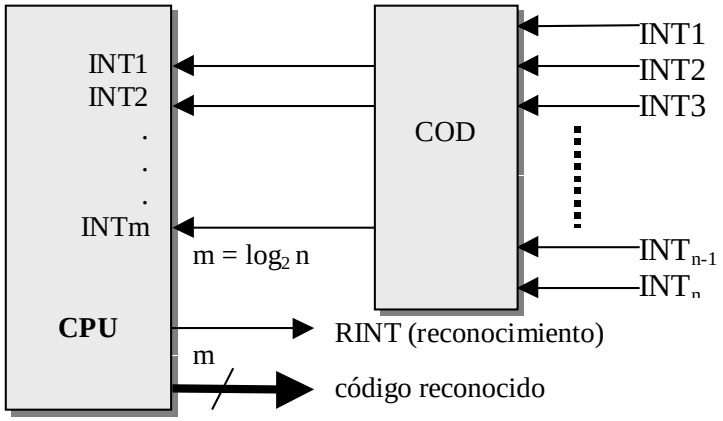
5. E/S por interrupción

Prioridades (2)

- En ocasiones, cada línea de interrupción tiene su propia señal de reconocimiento



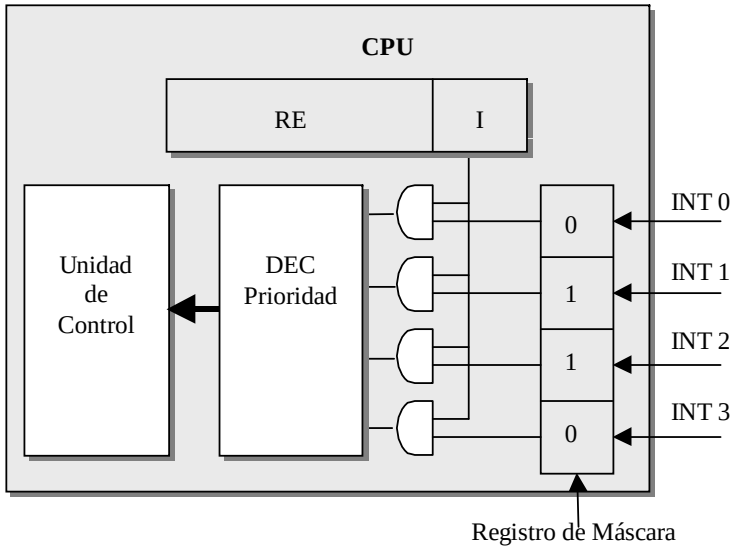
- Para minimizar el número de pines del chip dedicados a las interrupciones, las diferentes líneas pueden entrar codificadas a la *CPU*, siendo necesario el uso de un codificador externo:



5. E/S por interrupción

Enmascaramiento de interrupciones

- El sistema de interrupciones de un procesador dispone de la posibilidad de ser inhibido.
- Esta posibilidad hay que utilizarla en determinadas ocasiones en las que por ningún concepto se puede interrumpir el programa en ejecución.
- Existe también la posibilidad de enmascarar individualmente algunas de las líneas de interrupción utilizando un registro de máscara:



- La descapacitación también se puede realizar en el módulo de E/S.
- También se pueden inhibir las interrupciones en la rutina de tratamiento.

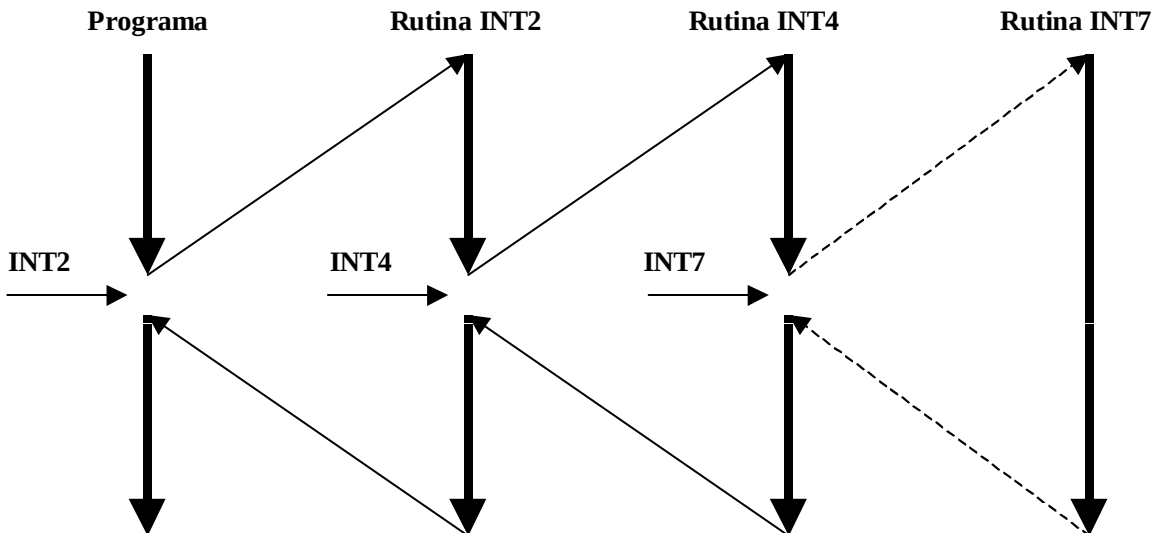
5. E/S por interrupción

Anidamiento de interrupciones

➢ Lo determina el sistema de prioridades:

***Un dispositivo sólo puede interrumpir a otro cuando su nivel de prioridad es mayor que el que se está atendiendo*.**

➢ Cuando las prioridades lo permiten, el anidamiento es análogo al que se produce con las subrutinas:



5. E/S por interrupción

Ejemplo de anidamiento de interrupciones (1)

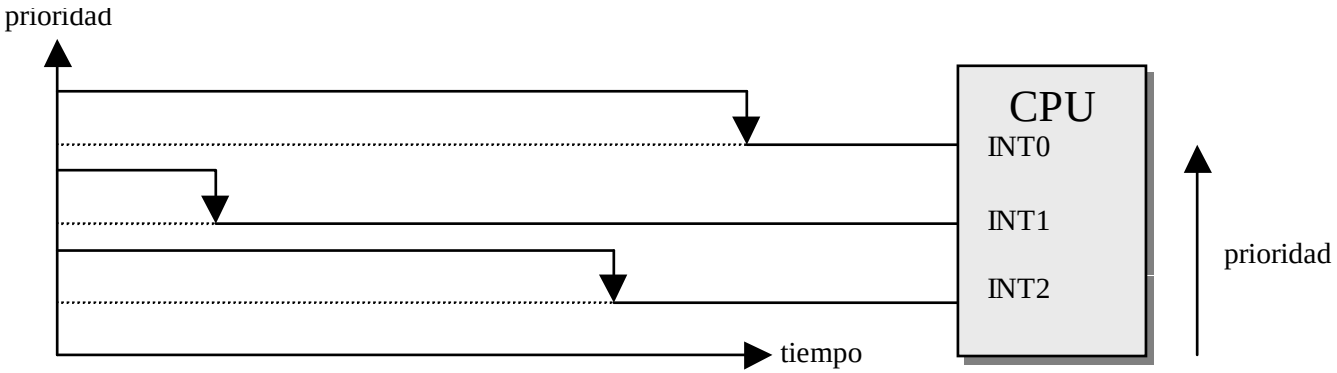
➢ *CPU* con tres líneas de interrupción: *INT0, INT1, INT2*, siendo la primera la más prioritaria, y la última la menos prioritaria.

➢ La *CPU* dispone de tres bits en su registro de estado para la inhibición de cada línea:

Si INTi = 0 nivel capacitado

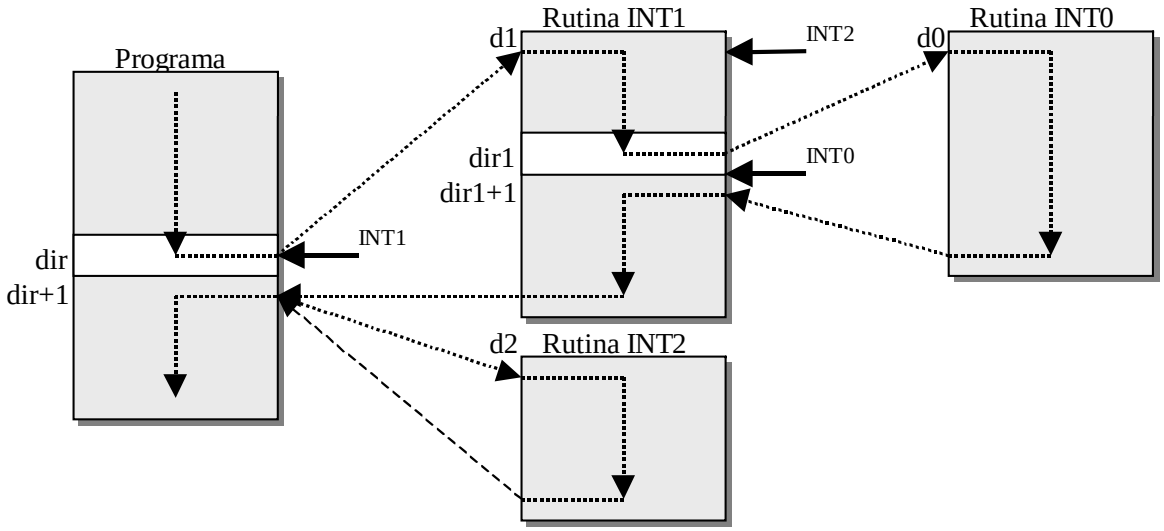
Si INTi = 1 nivel descapacitado

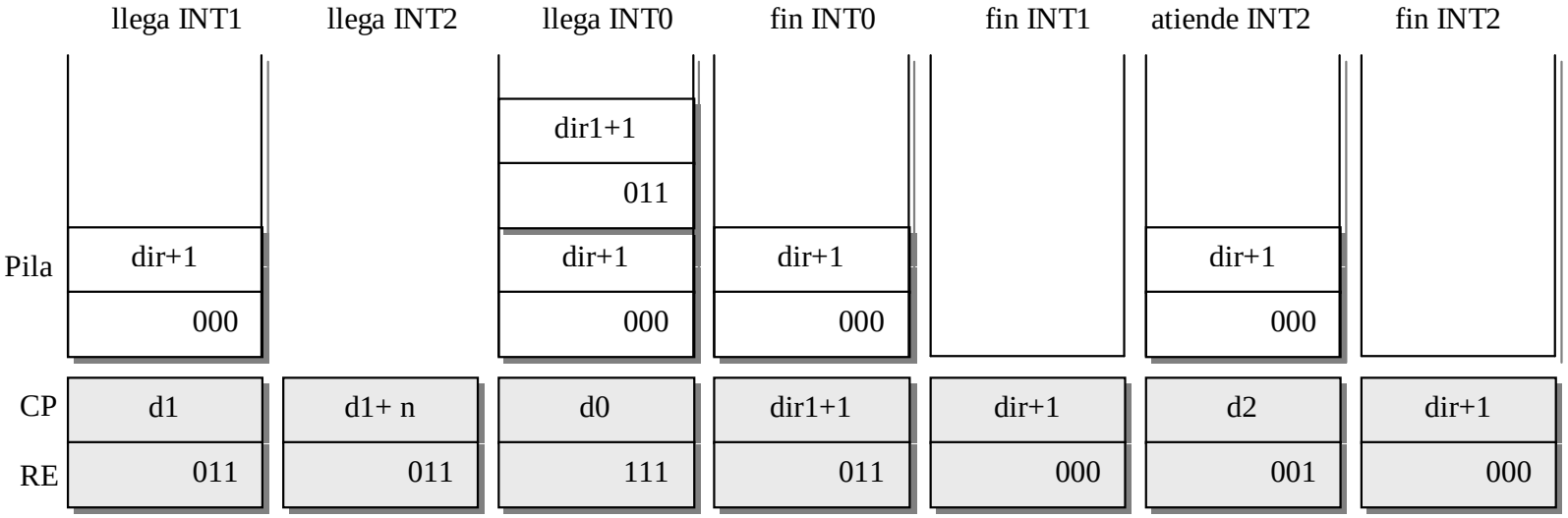
➢ Supongamos que llegan tres peticiones de interrupción en el orden temporal siguiente:



5. E/S por interrupción

Ejemplo de anidamiento de interrupciones (2)





5. E/S por interrupción

Soporte de las interrupciones a niveles superiores

- El sistema de interrupciones se puede utilizar para ejecutar concurrentemente varios programas residentes simultáneamente en memoria
- Para ello la rutina de tratamiento se convierte en un gestor encargado de pasar el control de ejecución a un programa diferente cada vez que se produce una interrupción.
- Las interrupciones las genera un temporizador a intervalos regulares o programables por el gestor.
- Esta posibilidad del sistema de interrupciones hace posible:
  - Los procesos del S.O.
  - Los *threads* en los lenguajes de alto nivel