SUBSISTEMA DE ENTRADA-SALIDA

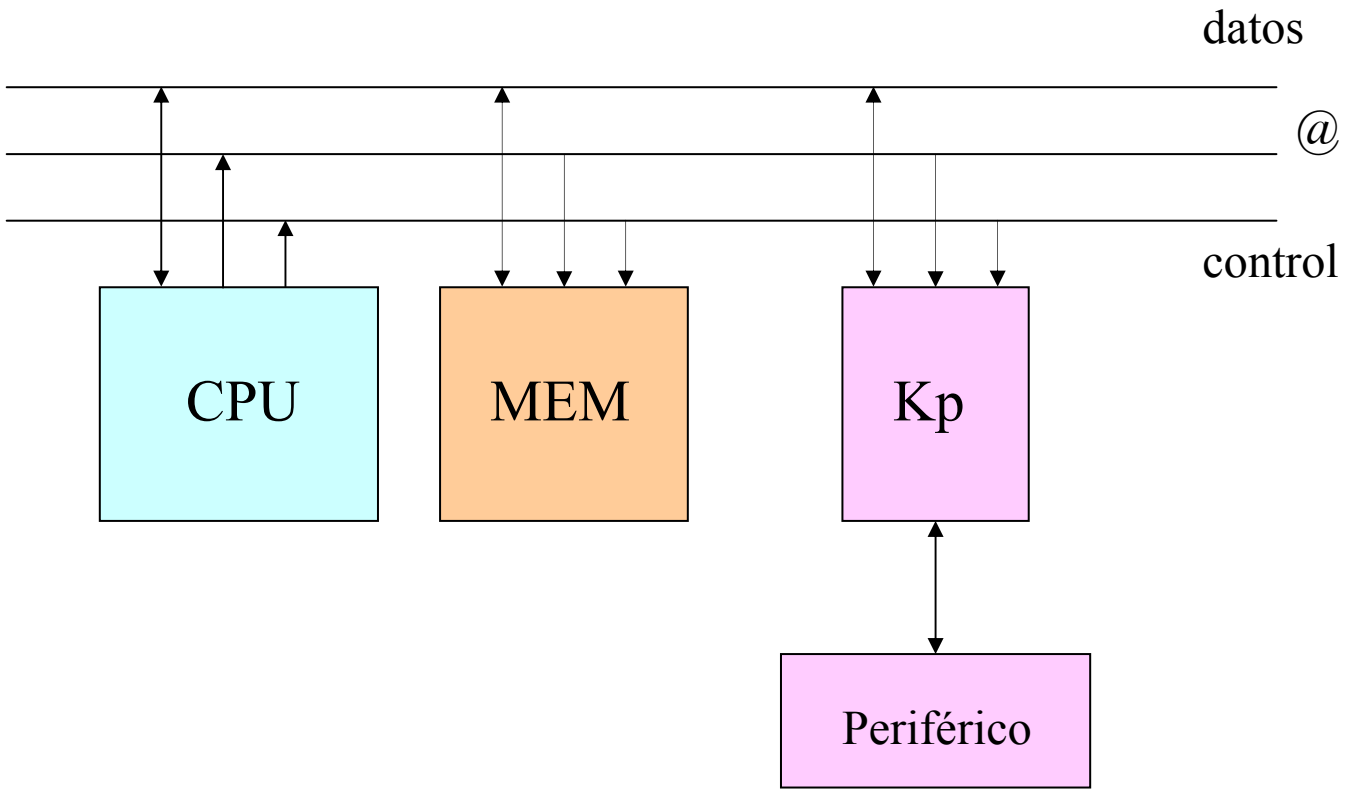
Arquitectura de Computadores I

4º tema

Descripción de la interfaz de E/S

- Objetivo
  - Cómo se comunica la CPU con el exterior (periféricos de E/S)
  - Cómo se controla esta comunicación.
- Periféricos
  - Presentación de datos (pantalla, impresora,...)
  - Adquisición de datos (teclado, sensores,...)
  - Soportes de información (discos, cintas,...)
  - Otros (motores, válvulas,...)
- Conexión (bus único, dos buses)

Esquema de la interfaz de E/S



Controlador

- Justificación

  gran variedad de periféricos con diferentes conexiones físicas a los buses y controlados de forma muy diferente electrónicamente.

- Controlador

  Dispositivo electrónico que se interpone entre los buses del sistema y el periférico y que es visible a nivel de LM como un conjunto de registros. El controlador descarga a la CPU del control directo del dispositivo

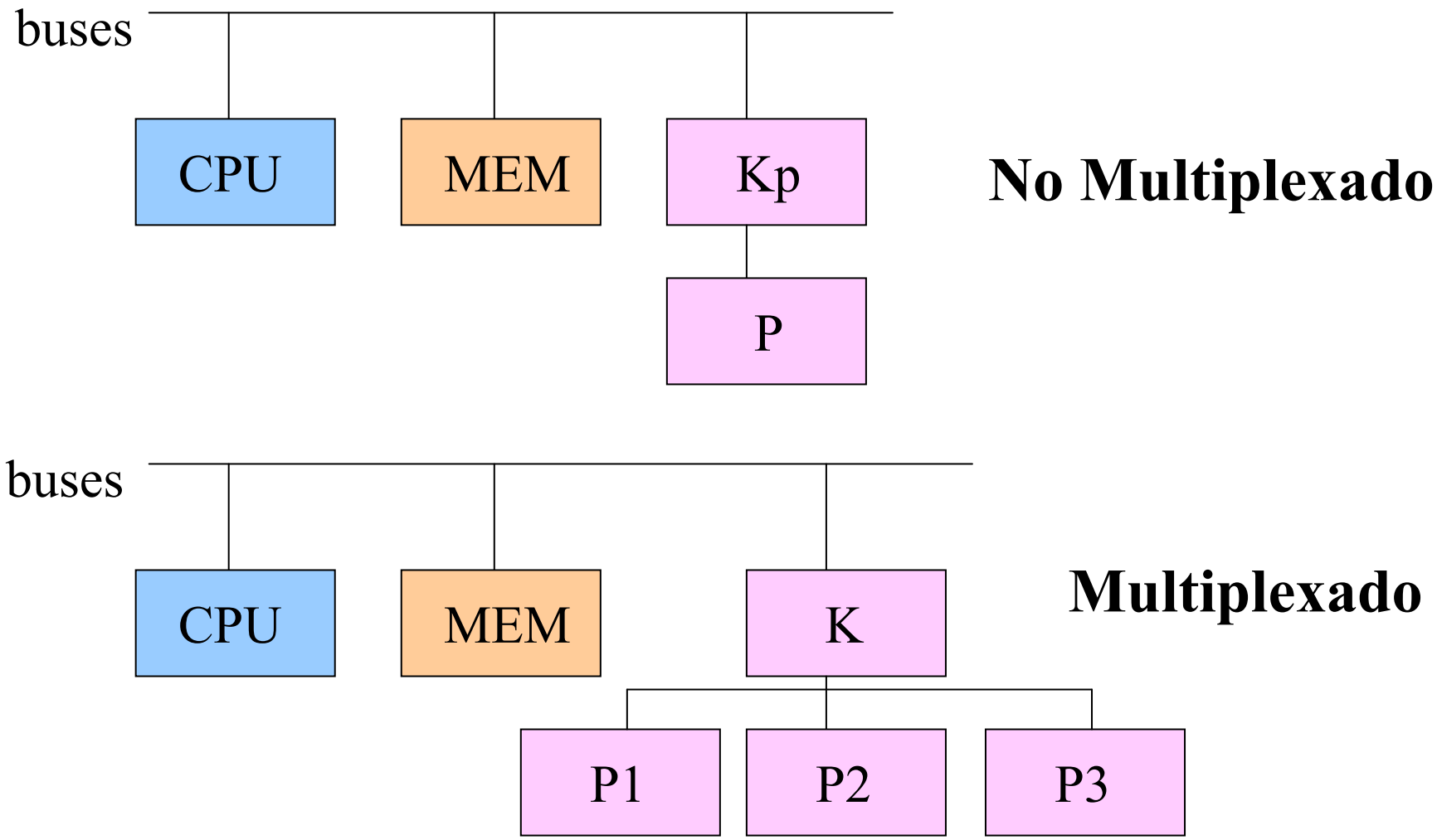
  “La CPU se comunica con los periféricos a través de los registros del controlador”.

Clasificación de los controladores de E/S

- No multiplexado
  - Sólo controla 1 periférico: periféricos rápidos

- Multiplexado
  - Controla varios periféricos
  - Multiplexado por caracteres: periféricos lentos
  - Multiplexado por bloques: periféricos de velocidad intermedia

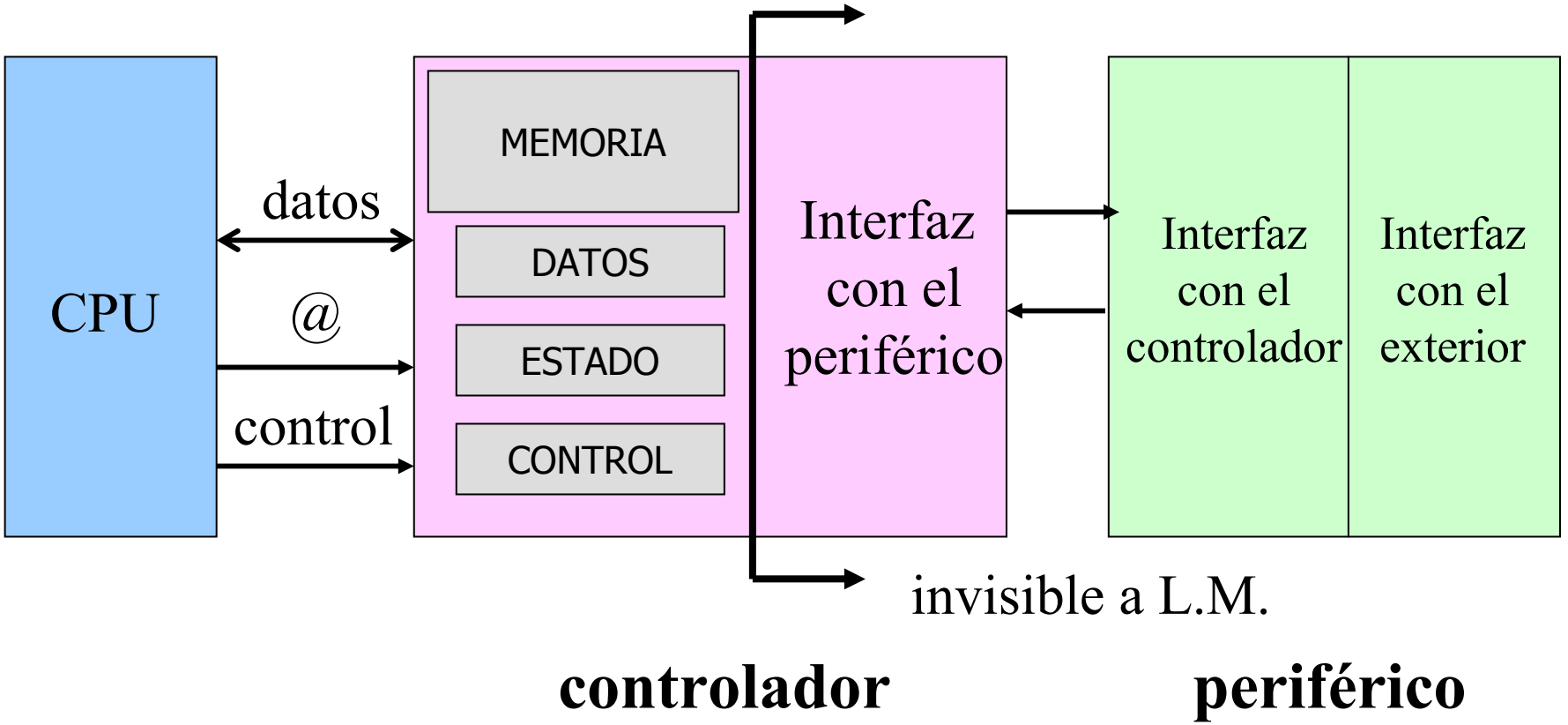
Clasificación de los controladores de E/S



Funciones del controlador de E/S

- Diálogo con la CPU
  - Recibe peticiones para realizar operaciones de E/S
  - Avisa acerca del estado de los periféricos
- Control del periférico
  - Controla que el periférico realice la operación indicada por la CPU
- Facilitar la transferencia con el periférico

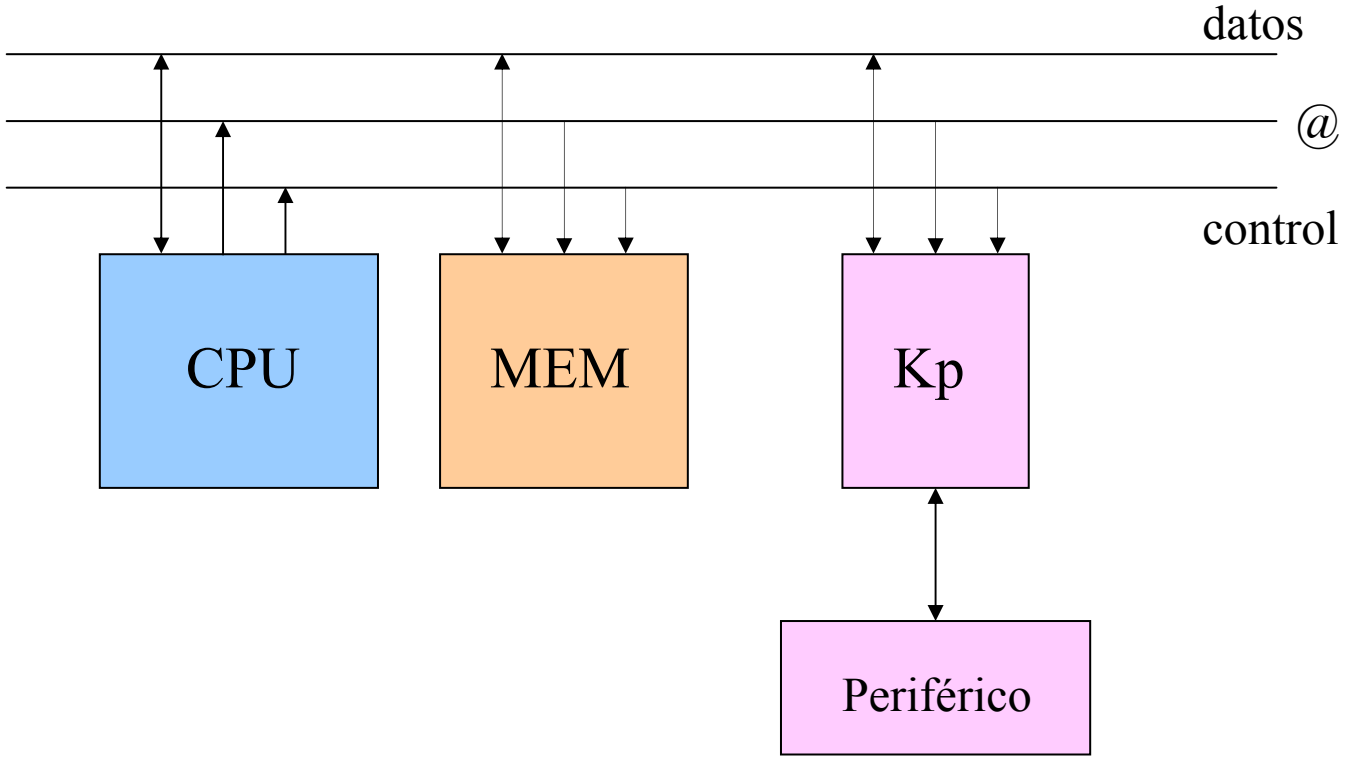
Visión funcional del controlador de E/S



Registros del controlador de E/S

- Registro de Estado
  - Información de estado: se ha pulsado una tecla, se ha terminado la escritura en disco, etc.
  - Información leída por la CPU
- Registro de control
  - Información de control: la operación a realizar (r/w), la forma de trabajar del periférico, etc.
  - Información escrita por la CPU
- Registro de datos
  - Información a transferir entre el periférico y la CPU

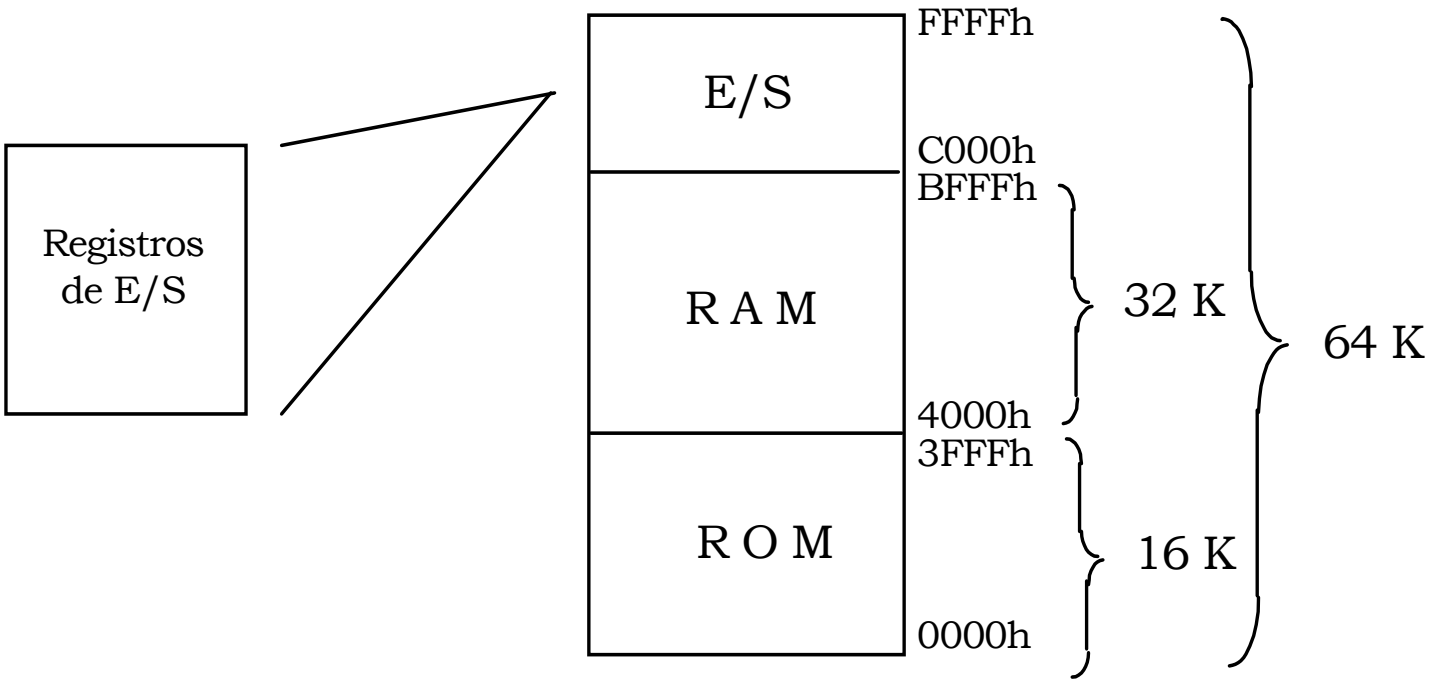
Esquema de la interfaz de E/S



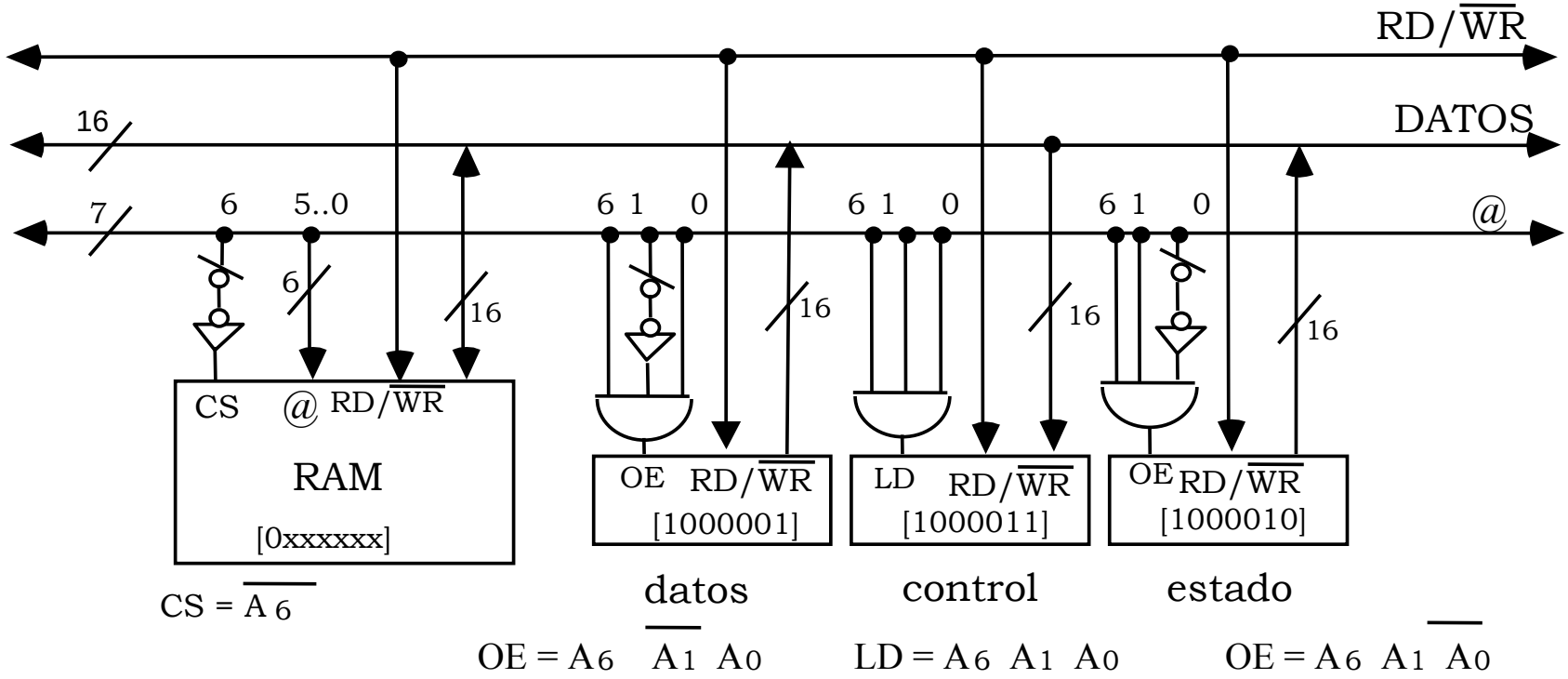
→¿Cómo se accede a los registros del controlador?

- E/S mapeada en memoria
- E/S no mapeada en memoria

Mapa de memoria
E/S mapeada en memoria



Ejemplo de E/S MAPEADA en MEMORIA



RAM → 0000000

Reg. datos → 1000001
Reg. control → 1000011
Reg. estado → 1000010

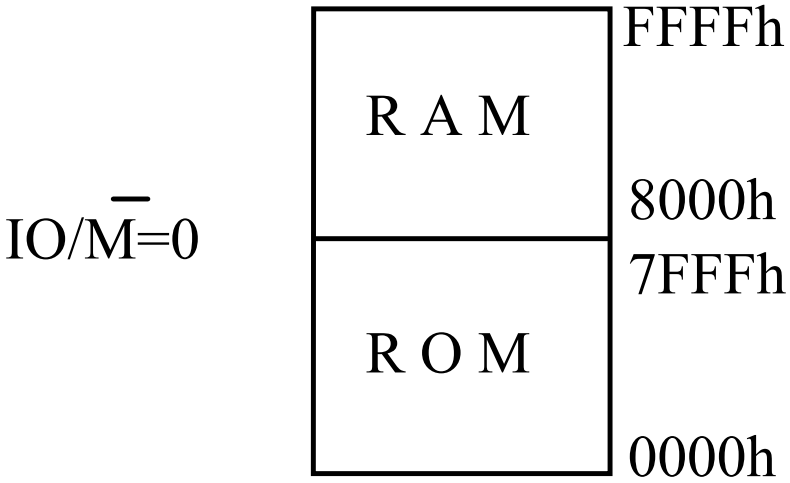
E/S mapeada en memoria

- Ventajas
  - Acceso a los registros: mismas instrucciones y modos de direccionamiento que para el acceso a memoria
  - CPU no diferencia el acceso a memoria y a los reg. E/S
- Desventajas
  - Menor espacio direccionable de memoria
  - Dificultad para controlar el acceso a los reg. E/S por parte de los programas de usuario

    → Modo privilegiado (SO) para accesos a E/S

Mapa de memoria ↔ mapa de E/S

MAPA DE MEMORIA



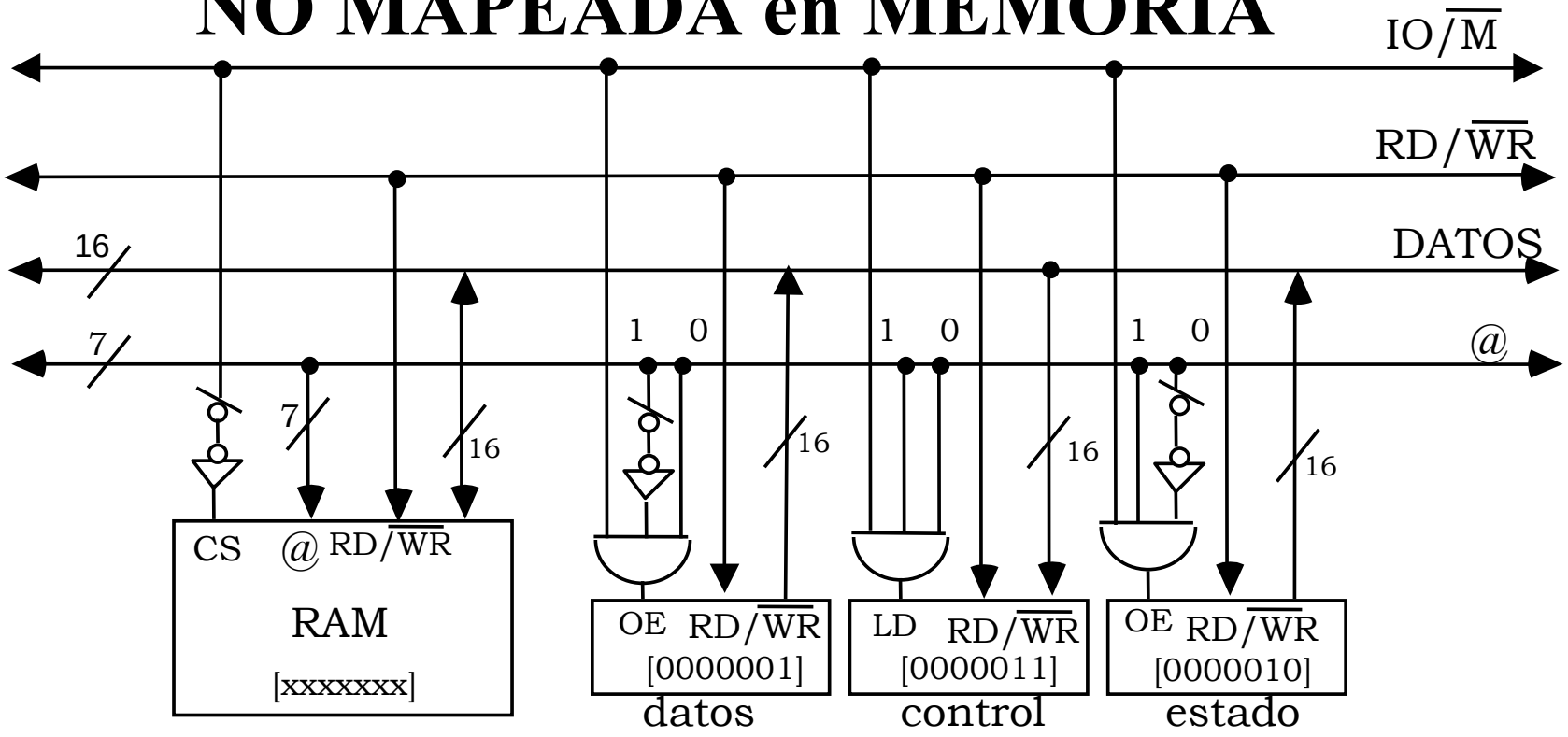
MAPA DE E/S

RAM → $\overline{IO/\overline{M}} * A_{15}$

ROM → $\overline{IO/\overline{M}} * \overline{A_{15}}$

Reg. E/S → $IO/\overline{M}$ (=1)

Ejemplo de E/S INDEPENDIENTE o NO MAPEADA en MEMORIA



$\overline{CS} = IO/\overline{M}$ $\quad OE = IO/\overline{M}\ \overline{A_1}\ A_0$ $\quad LD = IO/\overline{M}\ A_1\ A_0$ $\quad OE = IO/\overline{M}\ A_1\ \overline{A_0}$

RAM → 0000000

Instrucciones in/out

Reg. datos → 0000001
Reg. control → 0000011
Reg. estado → 0000010

E/S no mapeada en memoria

- Ventajas
  - No se pierde espacio direccionable de memoria
  - Control del acceso a los registros de E/S por parte de los programas de usuario

    → *in* y *out* sólo en modo privilegiado (SO)

- Desventajas
  - Necesidad de nuevas instrucciones: *in* y *out*
  - Necesidad de una señal de control más en el bus

Sincronización en las operaciones de E/S

- ¿cómo se sincroniza la CPU con los dispositivos de E/S? ¿cómo se detecta que el dispositivo está listo para comenzar la transferencia?

- 2 alternativas
  - Sincronización por **encuesta**: la CPU consulta constantemente del registro de estado del controlador
  - Sincronización por **interrupción**: el propio controlador avisa de su disponibilidad a la CPU mediante una interrupción

E/S por **encuesta**

- Consulta continua del registro de estado del controlador → encuesta por estado

- 2 alternativas
  - **Encuesta continua**:

    ```
    while (leer_registro_estado==NO_PREPARADO) NOP;
    realizar_operaciones_E/S;
    ```

  - **Encuesta periódica**:

    ```
    while (leer_registro_estado==NO_PREPARADO)
                { otras_acciones; }
    realizar_operaciones_E/S;
    ```

E/S por **interrupciones**

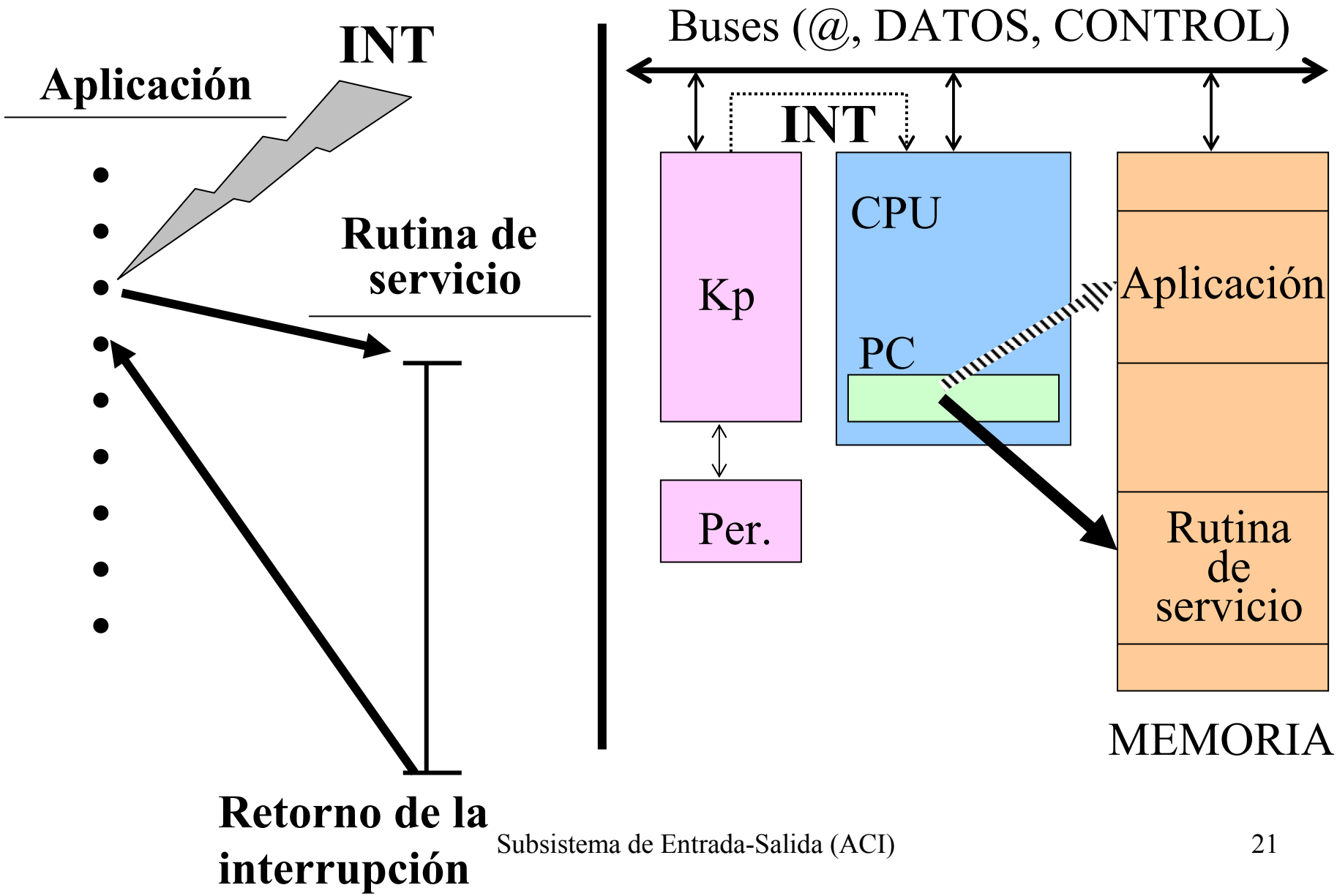
- Interrupción: suceso extraordinario que requiere una atención especial por parte del procesador
- Son sucesos asíncronos: el controlador interrumpe a la CPU cuando está listo para iniciar la transferencia
- Ventaja:
  - La CPU no pierde tiempo realizando la encuesta y puede realizar otras tareas
- Desventajas:
  - Se complica el hardware (mecanismo de ejecución de las instrucciones)

E/S por **interrupciones: fases**

- Detección de la petición de interrupción
- **Salvar el estado** del programa interrumpido
- Identificar la **rutina de atención a la interrupción** que hay que ejecutar (dependiente del dispositivo)
- Ejecutar la rutina de atención correspondiente

  → **Rutina asíncrona**: no activada por el programa principal
- **Recuperar el estado** (la ejecución) del programa interrumpido

E/S por **interrupciones**

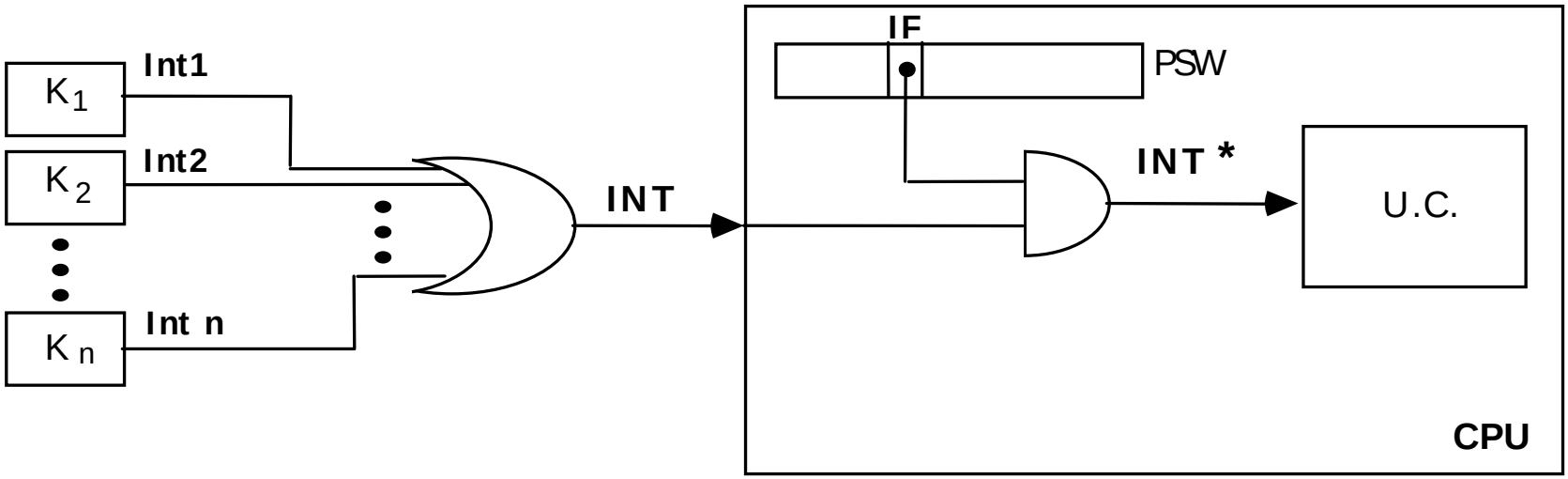


Detección de la interrupción

- La CPU dispone de una o varias líneas a través de las que los dispositivos pueden interrumpir a la CPU

  → INT: señal de entrada a la unidad de control
- La CPU muestrea el valor de la señal de petición de interrupción
  - En alguna de las fases de ejecución de la instrucción
- Flag de interrupción (*Interrupt Flag, IF*):
  - IF = 0 → interrupciones inhibidas/enmascaradas (**CLI** en 80x86)
  - IF = 1 → interrupciones permitidas (**STI** en 80x86)
- Interrupciones no enmascarables **(***Non Maskable Interrupt, NMI***)**

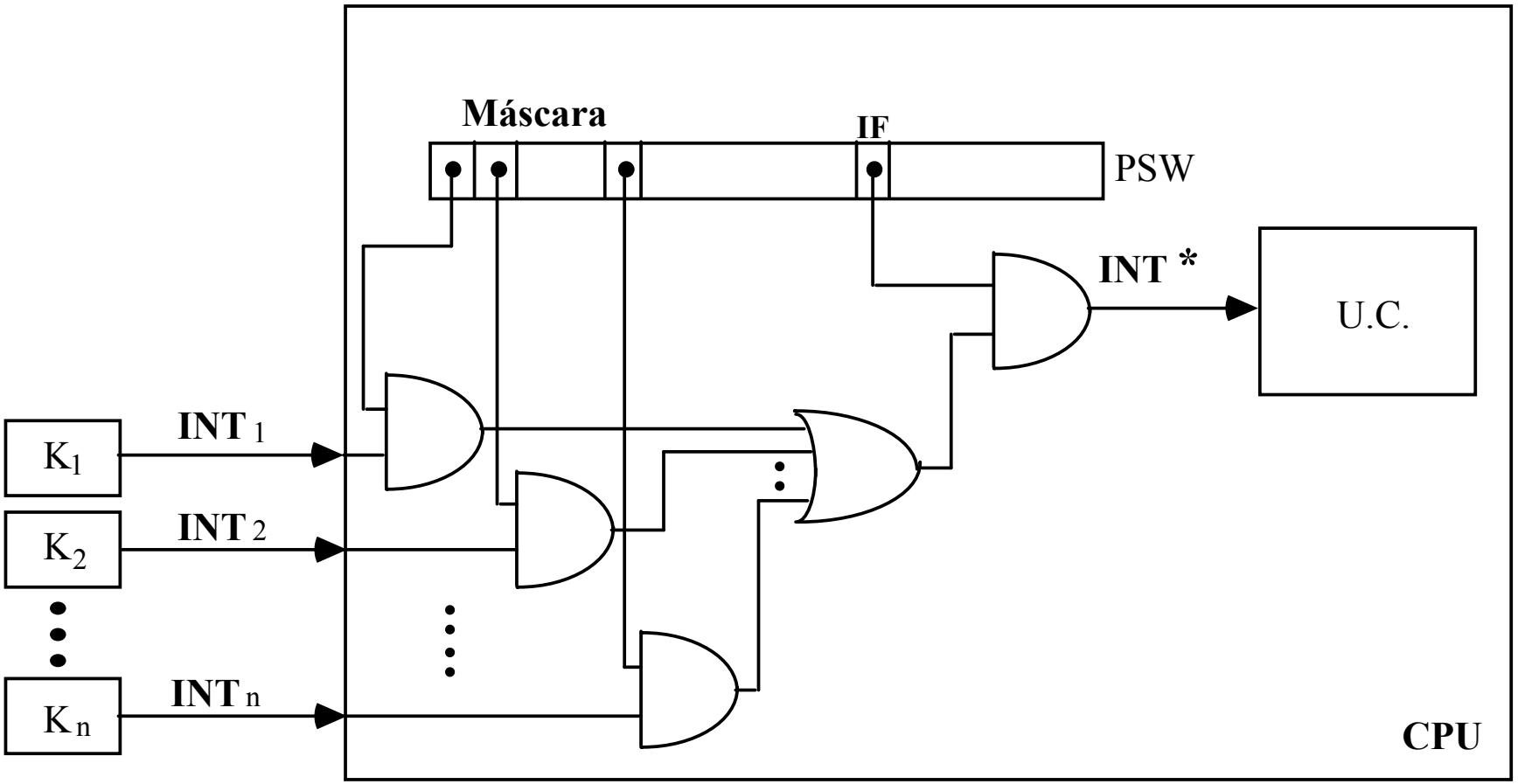
Detección de interrupciones



- Flag de interrupción (*IF*):
  - Cuando la CPU atiende una interrupción, pone IF = 0
  - Cuando finaliza el tratamiento de la interrupción pone IF = 1

    → interrupciones uninivel

Detección y enmascaramiento de interrupciones



Salvar el estado del programa

- La CPU interrumpe la ejecución del programa en el momento de tratar la interrupción y reanuda su ejecución tras el tratamiento de la interrupción
- Hay que salvar información de contexto del programa interrumpido
  - El PC de retorno → PC de la siguiente instrucción en ejecución
  - La palabra de estado, PSW → flags, etc.
- La rutina de interrupción salva los registros que utilice
- ¿Dónde se salva esta información? → **PILA**

Identificación de la interrupción

- problemas
  - ¿Cómo determinar quién ha interrumpido?
  - Si varios han interrumpido por la misma línea INT, ¿a quién se atiende?
- alternativas
  - Identificación **software** (por ejemplo, MC68000)
  - Identificación **hardware**

Identificación software

- Tras la detección de la señal INT y si IF=1, la CPU ejecuta una **rutina de atención general** que determina mediante **encuesta** qué dispositivo ha interrumpido

```
for (i=0;i<n;i++)
        if (leer_reg_estado (K_i) == PREPARADO)
                { rut_atencion_R_i(); break; }
```

- Una vez determinado el periférico, se ejecuta la rutina de atención correspondiente a ese periférico
- Interrupciones simultáneas?

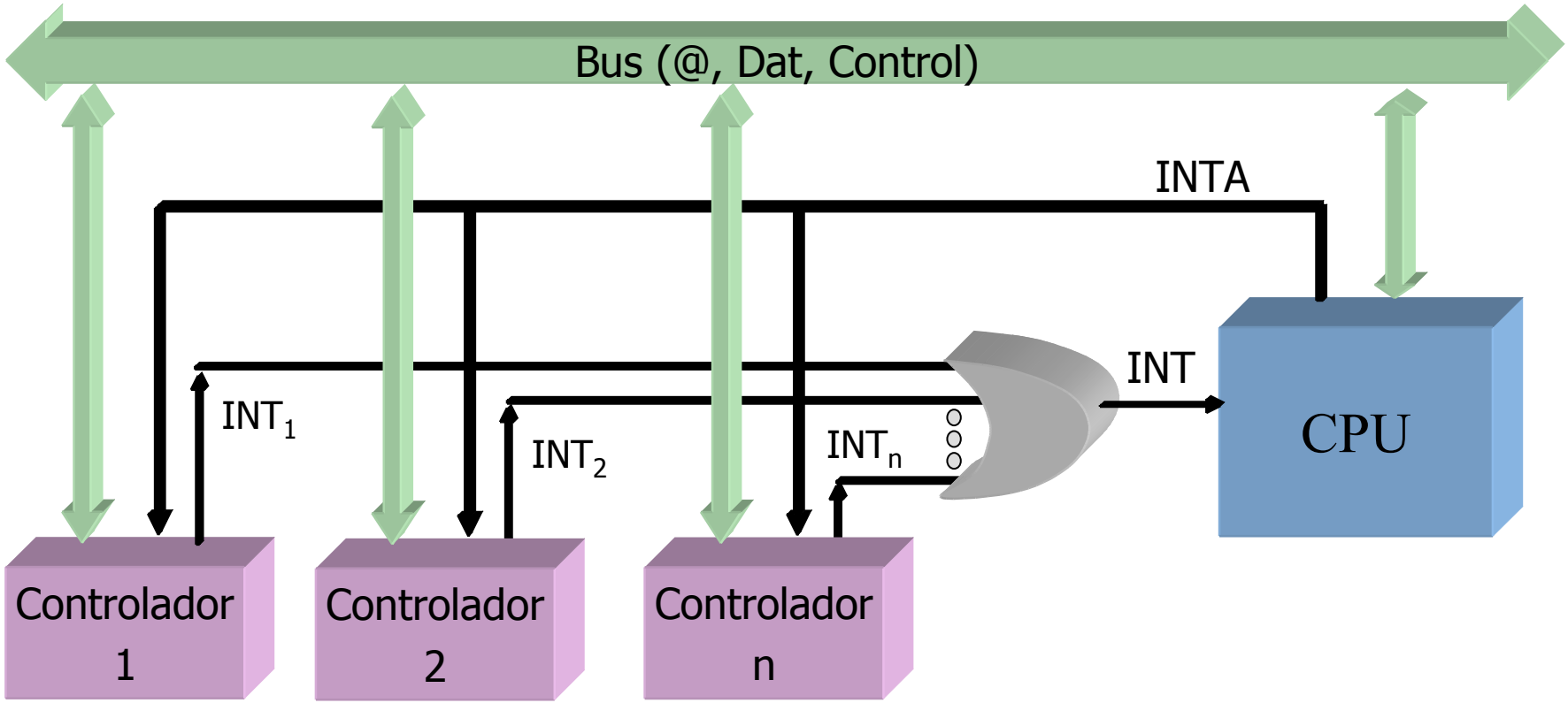
  → El orden en que se realiza la encuesta determina la prioridad entre los dispositivos

Identificación hardware

- El propio dispositivo es quien se identifica
- El controlador envía a la CPU (tres opciones):
  - la propia dirección de la rutina de atención, o bien
  - el código de una instrucción (salto) que será ejecutada por la CPU para lanzar la ejecución de la rutina de atención, o bien
  - un identificador que es usado por la CPU para acceder a una tabla (**vector de interrupción**) donde se encuentra la dirección de la rutina a ejecutar

    → interrupción vectorizada (i80x86)
- La CPU acepta la petición de interrupción por medio de una señal INTA hacia el controlador del periférico

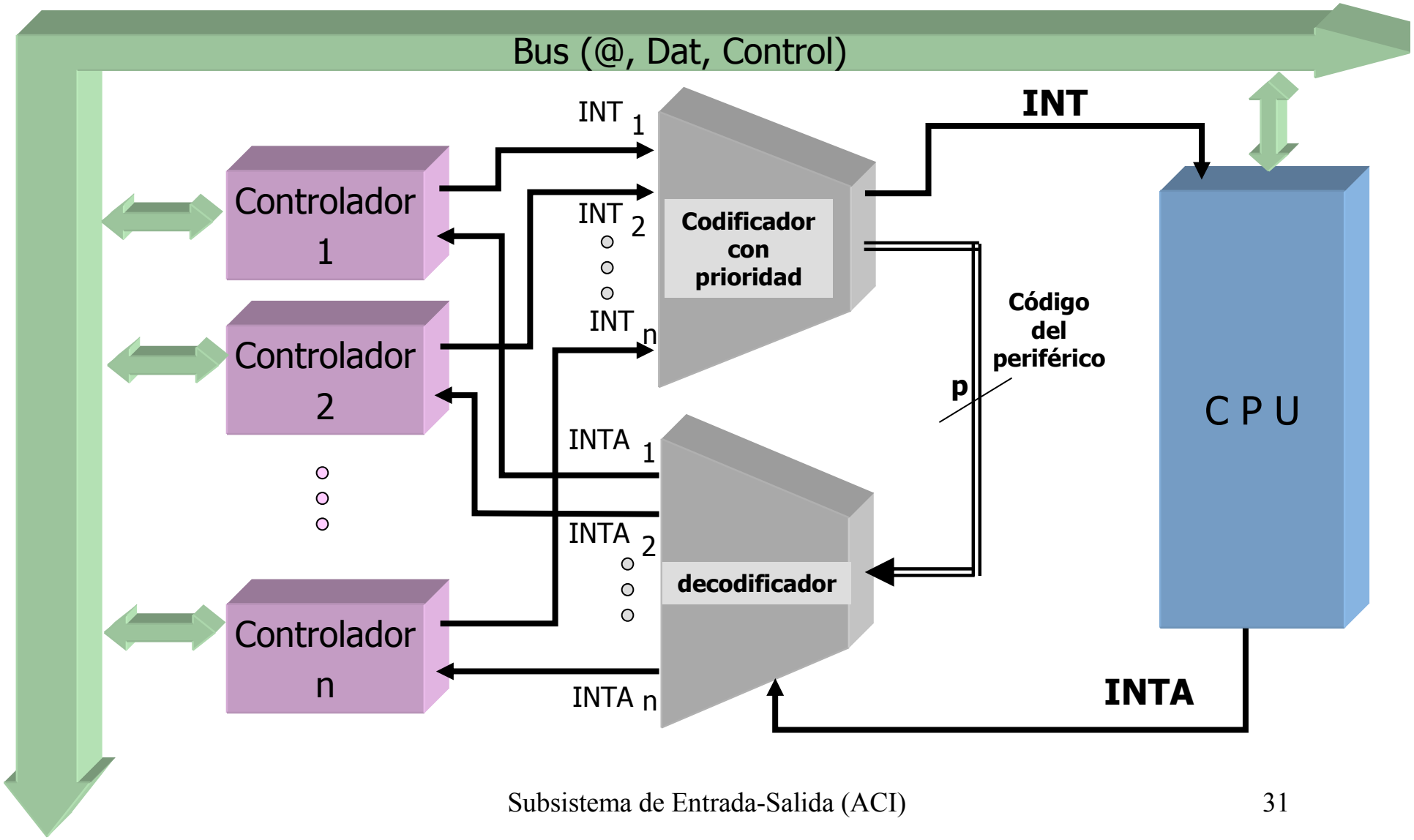
Identificación hardware



Interrupciones simultáneas

- Si varios periféricos han interrumpido por la misma línea INT, ¿a quién se atiende?
- Alternativas
  1. **Codificador con prioridad**:
     + un codificador proporciona, además de la señal INT, el código del periférico con mayor prioridad
     + un decodificador utiliza este código para llevar la señal INTA de la CPU al controlador correspondiente

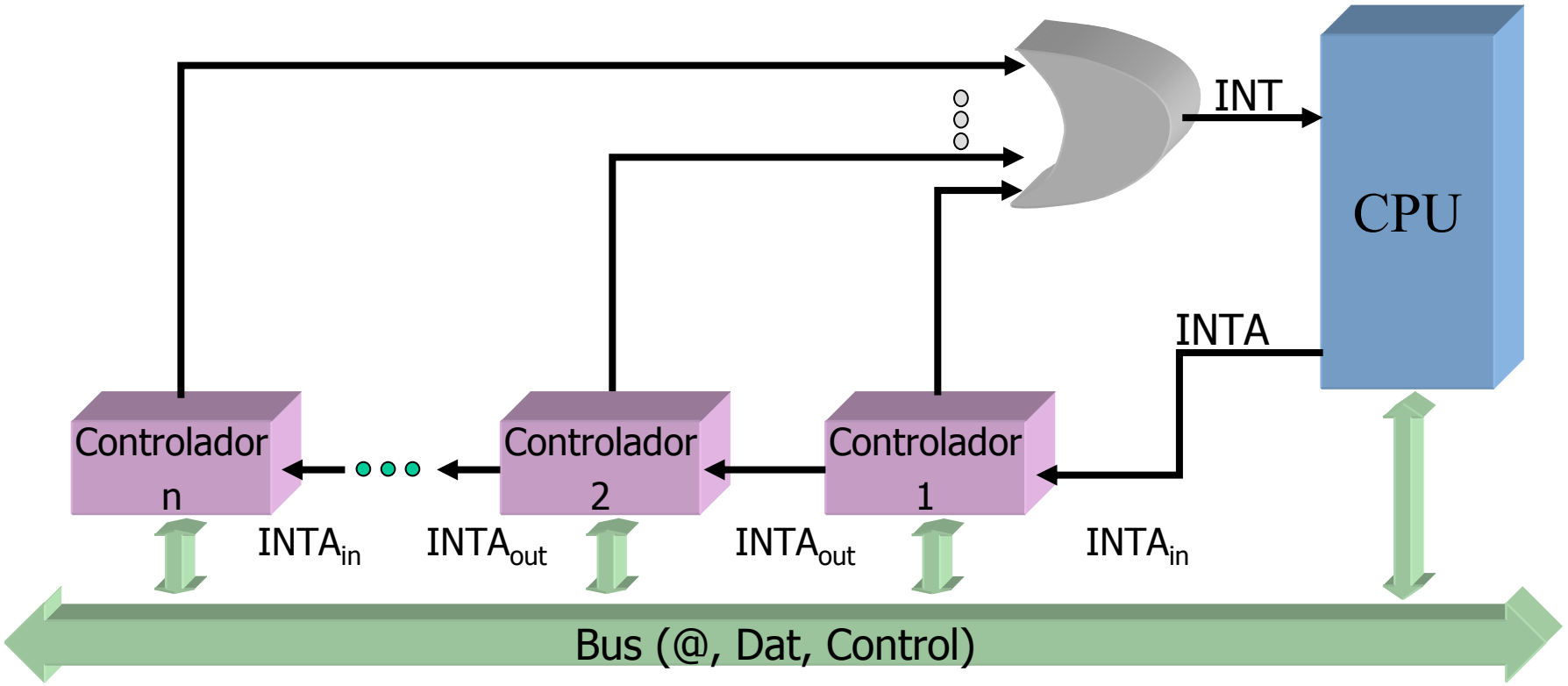
Identificación de **interrupciones simultáneas: Codificador con prioridad**



Interrupciones simultáneas

- alternativas
  2. ***Daisy-chain* o cadena de margaritas**:
     + la CPU activa la señal INTA
     + cuando el controlador detecta la señal INTA
       + si no ha pedido la interrupción, activa la señal de salida INTA hacia el siguiente controlador (así hasta que la señal llega al controlador que ha pedido la interrupción)
       + si ha pedido la interrupción, no pasa la señal INTA al siguiente controlador y envía a la CPU la información necesaria para el tratamiento de la interrupción
     + los controladores más cercanos a la CPU son los más prioritarios

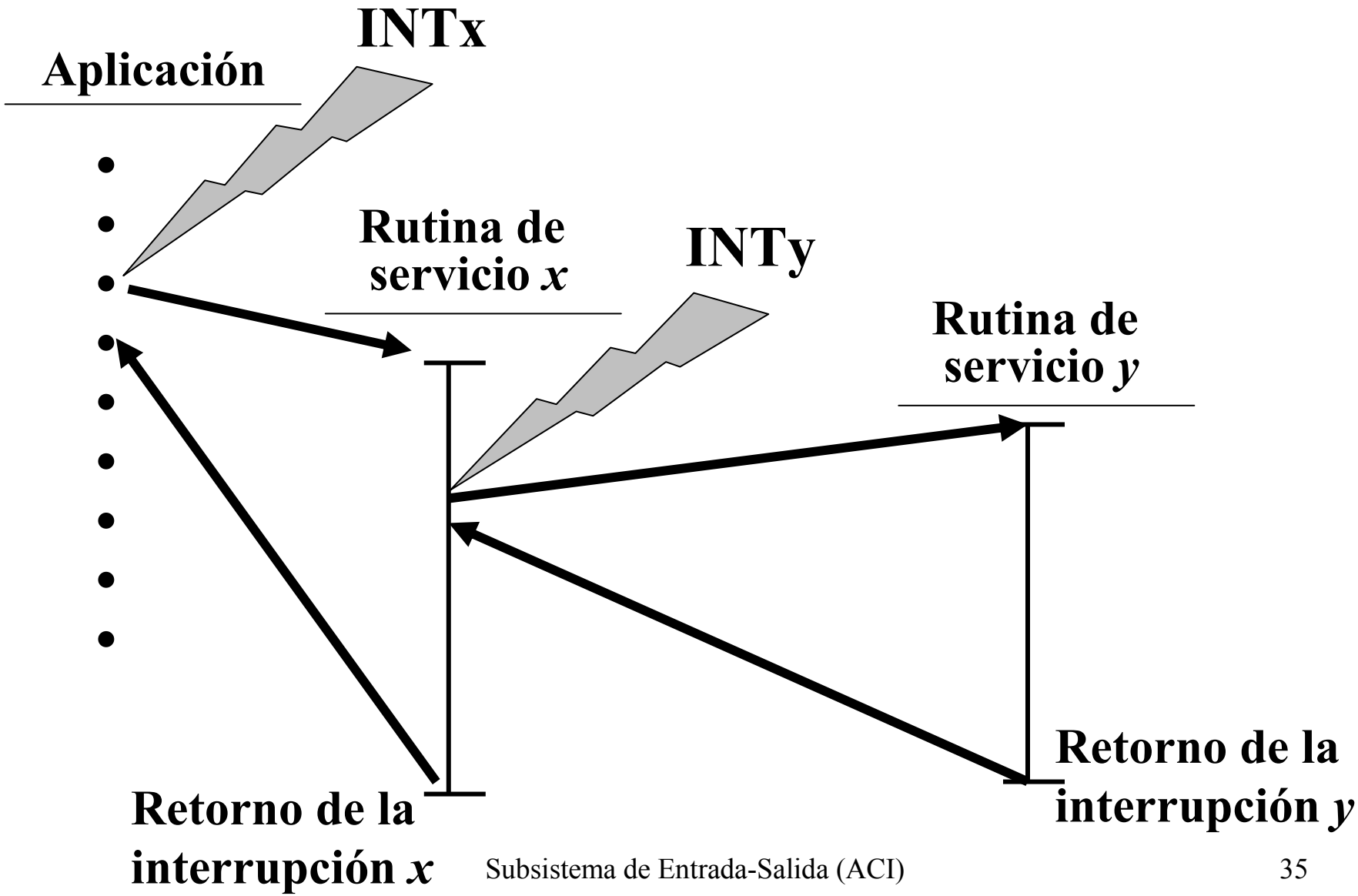
Identificación de **interrupciones simultáneas:** ***Daisy-Chain*** **o cadena de margaritas**



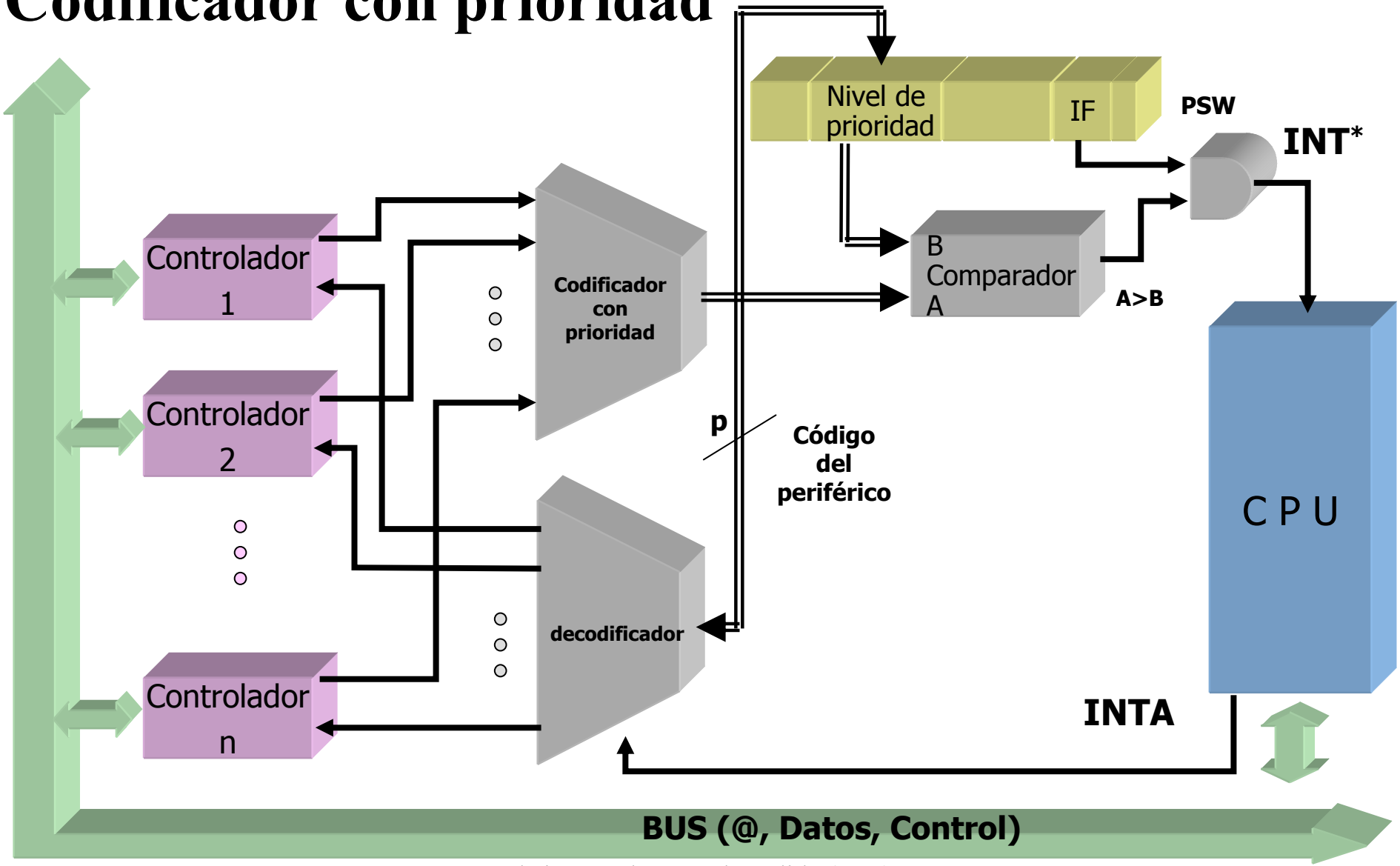
Interrupciones multinivel

- La CPU puede ser interrumpida mientras está ejecutando otra rutina de atención a una interrupción previa
  - + se pospone la ejecución de la rutina de interrupción actual y se atiende la nueva interrupción
  - + IF no se desactiva por defecto al tratar una interrupción
- Prioridades decrecientes:
  - + se atiende la interrupción de un periférico de prioridad *n* si están permitidas y si en ese momento se está ejecutando una rutina de atención a una interrupción menos prioritaria

Interrupciones multinivel

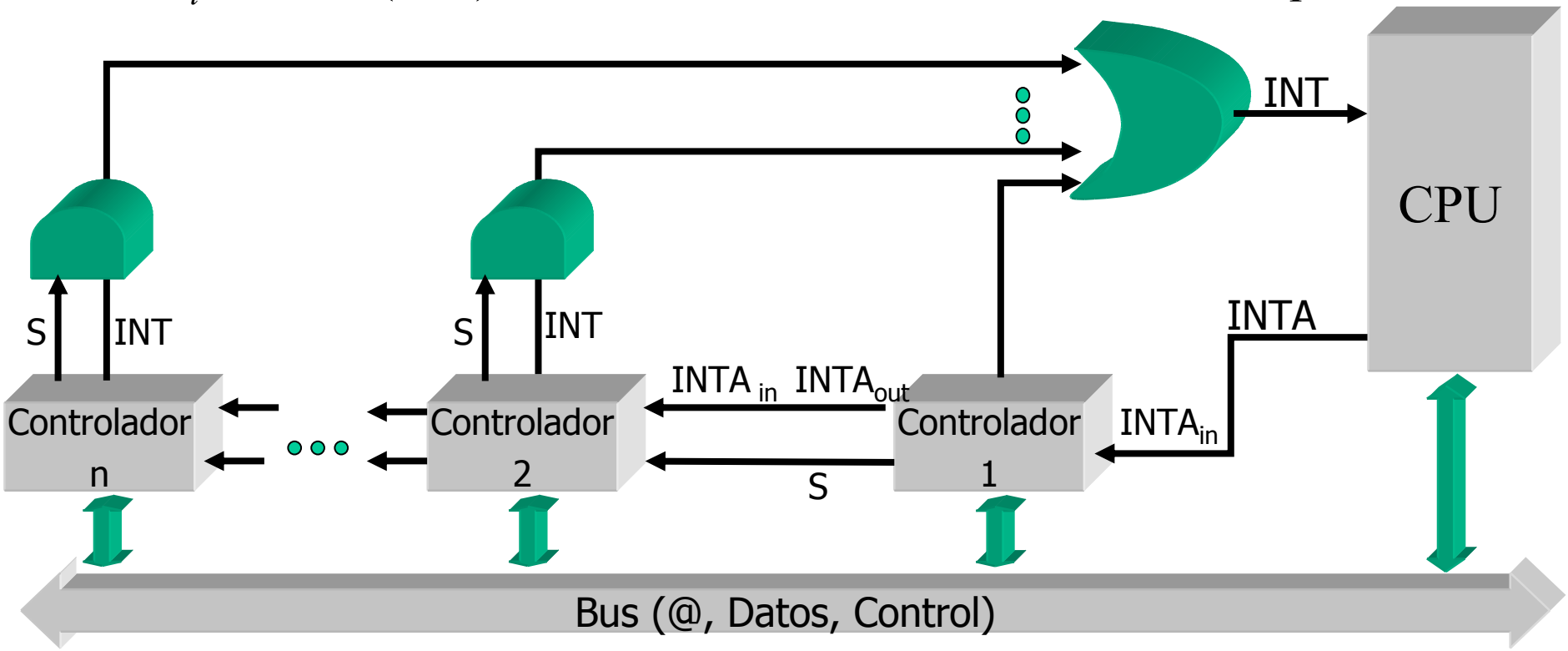


Interrupciones multinivel: Codificador con prioridad



Interrupciones multinivel: *Daisy-Chain* o cadena de margaritas

+ K_i desactiva la señal S ($S=0$) para no permitir interrupciones multinivel a los periféricos con menor prioridad a K_i
+ K_i activa S ($S=1$) tras finalizar el tratamiento de su interrupción



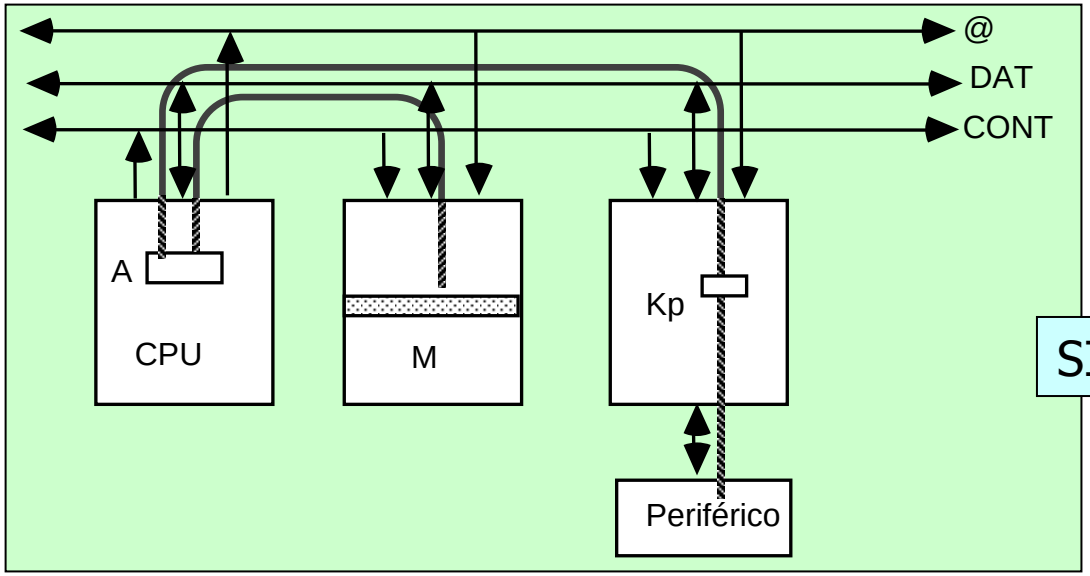
Controlador de interrupciones

- Circuito intermediario entre la CPU y los controladores de los periféricos de E/S
- Circuito que resuelve todos los aspectos comentados:
  - + gestiona la comunicación entre la CPU y el controlador del periférico
  - + determina el periférico a atender en función de prioridades
  - + gestiona las interrupciones multinivel, ...
- Por ejemplo, controlador de interrupciones i8259 del i8086

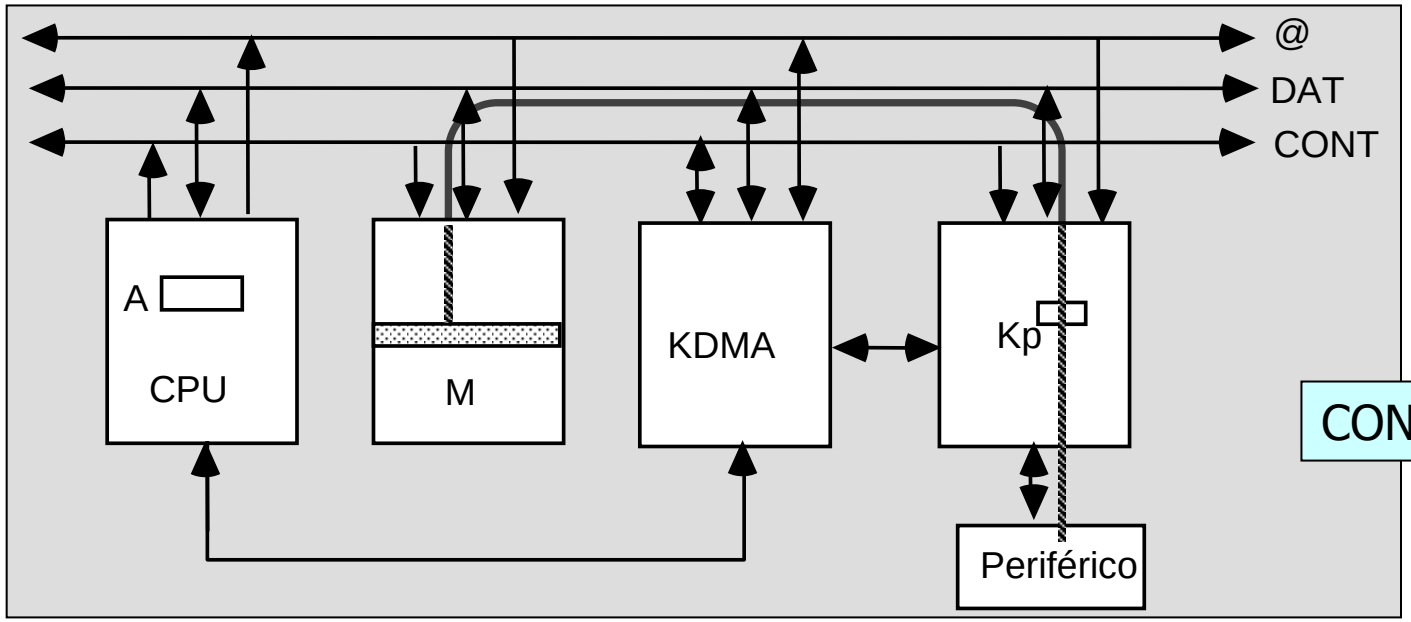
Entrada/salida por DMA: *Direct Memory Access* / Acceso Directo a Memoria

- Motivación:
  - + transferencia de gran volumen de datos desde un dispositivo (por ejemplo, disquete) a posiciones consecutivas de memoria
  - + sin DMA, la CPU se vería constantemente interrumpida (por ejemplo, por cada carácter a transferir desde el disco)
- Solución: controlador de DMA
  - + Circuito especializado que descarga a la CPU de un trabajo simple, repetitivo y frecuente
  - + Circuito capaz de acceder directamente a memoria sin usar los registros generales del procesador

Entrada/salida por DMA

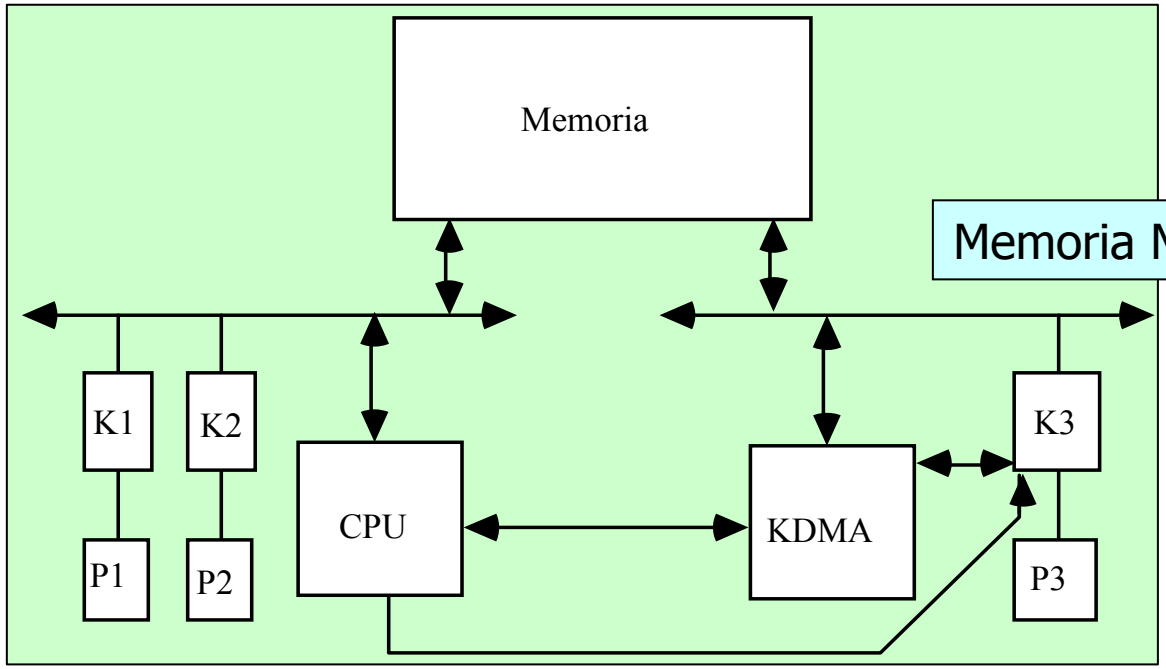


SIN DMA



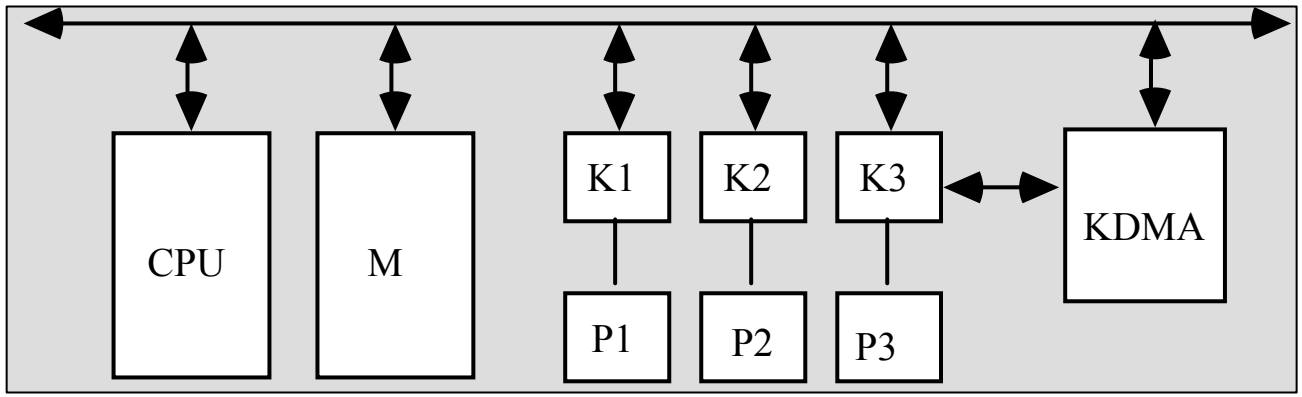
CON DMA

Entrada/salida por DMA



Memoria Multipuerto

Sincronización entre la CPU y el KDMA en el uso del bus

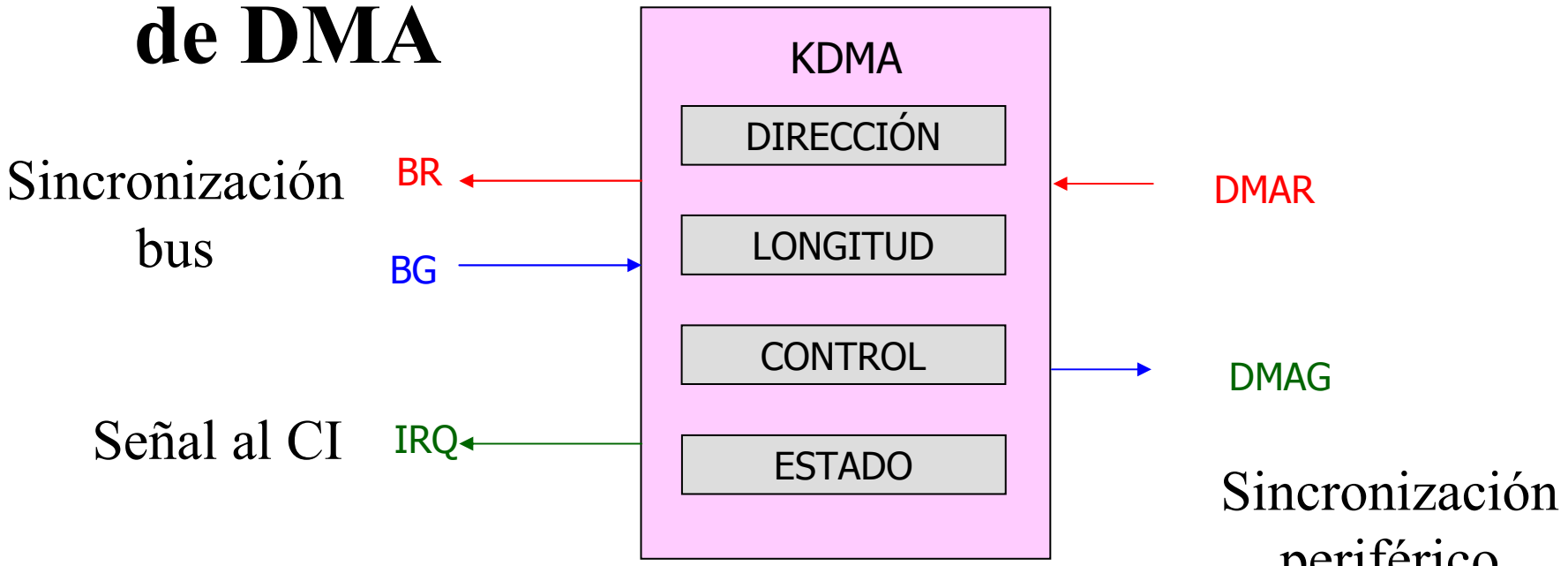


Memoria con un solo puerto

Controlador de DMA

- **Registro de dirección**: dirección de lectura/escritura de la información a transferir
- **Registro de longitud**: contador con el número de datos que faltan por transferir
- **Registro de control**: información de control (lectura/escritura, modo de sincronización al finalizar, etc.)
- **Registro de estado**: función de un registro de estado de cualquier otro controlador (por ejemplo, control de errores)

Controlador de DMA



BR *Bus Request*	Petición de utilización del BUS
BG *Bus Grant*	Concesión del BUS
IRQ *Interruption ReQuest*	Petición de Interrupción

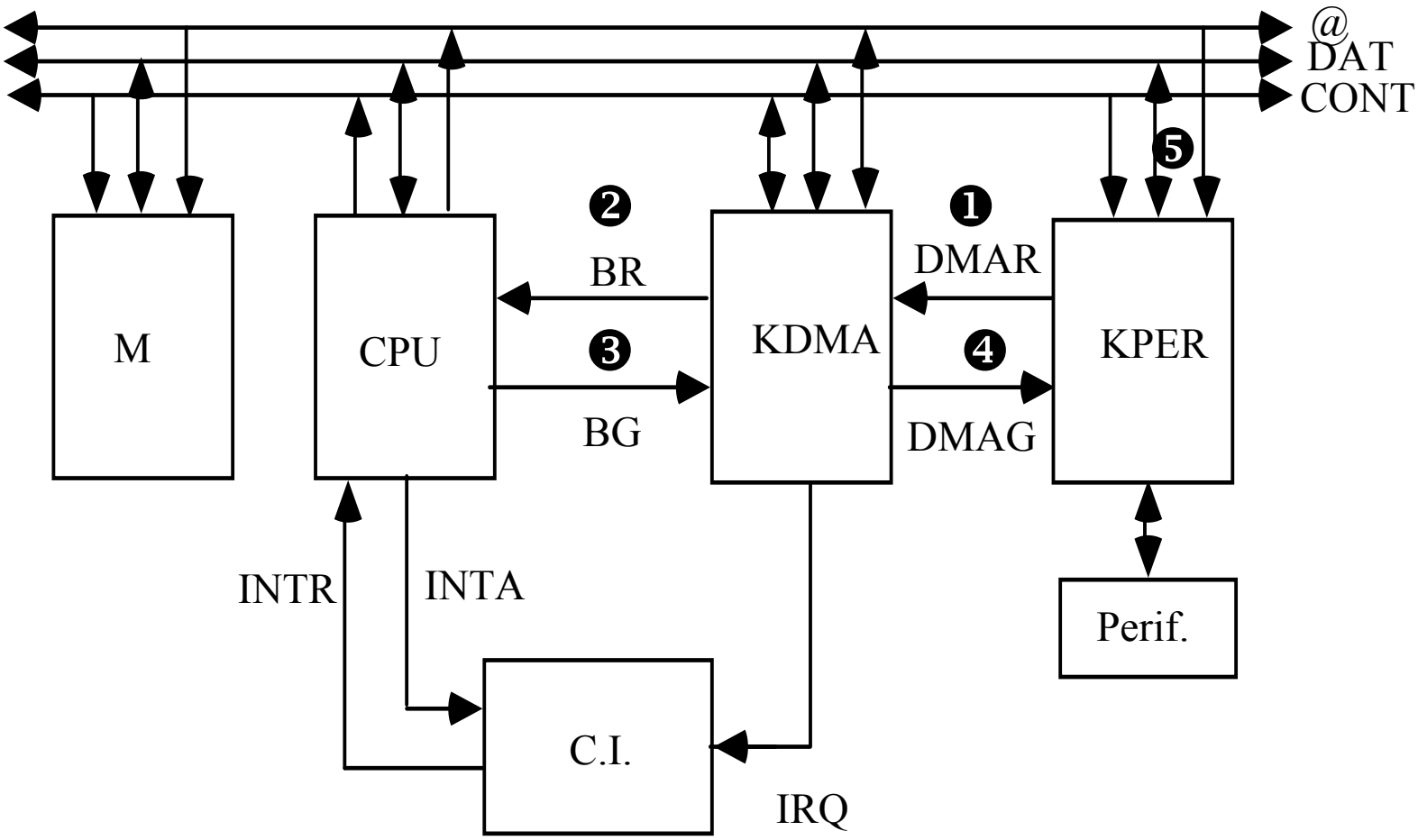
DMAR *Direct Memory Access Request*	Petición de Acceso directo a memoria.
DMAG *Direct Memory AccessGrant*	Concesión de Acceso directo a memoria

Entrada/salida por DMA: funcionamiento

- **Aplicación (CPU):** programa periférico y su controlador de DMA (@, número de datos, lectura/escritura)
- **Periférico**: cada vez que está listo para enviar/recibir un dato, avisa al KDMA
- **KDMA**: sincroniza con la CPU el uso del bus y realiza la transferencia del dato
- Al finalizar la transferencia de todos los datos, **KDMA** se **sincroniza** con la **CPU**: encuesta o **interrupción**

  → **sólo se interrumpe a la CPU al final de la E/S**

Entrada/salida por DMA



Entrada/salida por DMA: fases

1.-Iniciar la transferencia (CPU)

- Programar el KDMA
  - Dirección
  - Longitud
  - Sentido de la operación: Periférico → Memoria
    Memoria → Periférico
  - Interrumpir o no al finalizar
  - Otras señales (incrementar/decrementar)
- Programar el Kper (controlador del periférico)

Entrada/salida por DMA: fases

2.-Transferencia propia (KDMA) [CPU→otras tareas]

- Kper →DMAR (Petición de DMA) ❶
- KDMA →BR (Petición del uso de los buses) ❷
- CPU →BG (Concesión del uso de los buses) ❸
- KDMA realiza la transferencia del dato/datos
  - Dirección en el bus de direcciones
  - Activa la señal (R/W)
  - Activa DMAG ❹ → Kper actúa sobre el bus de datos ❺
- Kper desactiva DMAR
- DMA desactiva BR

Entrada/salida por DMA

Transferencia por **robo de ciclo**

- KDMA roba de vez en cuando un ciclo a la CPU para acceder a memoria
- se degrada el rendimiento de la CPU si la velocidad del periférico es alta

Transferencia por **ráfagas o bloques**

- cuando KDMA recibe la señal BG, no abandona el control del bus hasta que se complete la transferencia de todo el bloque
- modo adecuado para dispositivos rápidos

Entrada/salida por DMA

3.-Finalizar la transferencia

- Sincronización KDMA→CPU para indicar el fin de la E/S
- Por **encuesta continua**: la CPU consulta el registro de estado del KDMA → poco interesante!!!!
- Por **interrupciones**: KDMA interrumpe a la CPU

```
Leer registro de estado del KDMA
Si (no ha ido bien)
    programar de nuevo el KDMA
    programar de nuevo el Kper
sino
    ..............
```