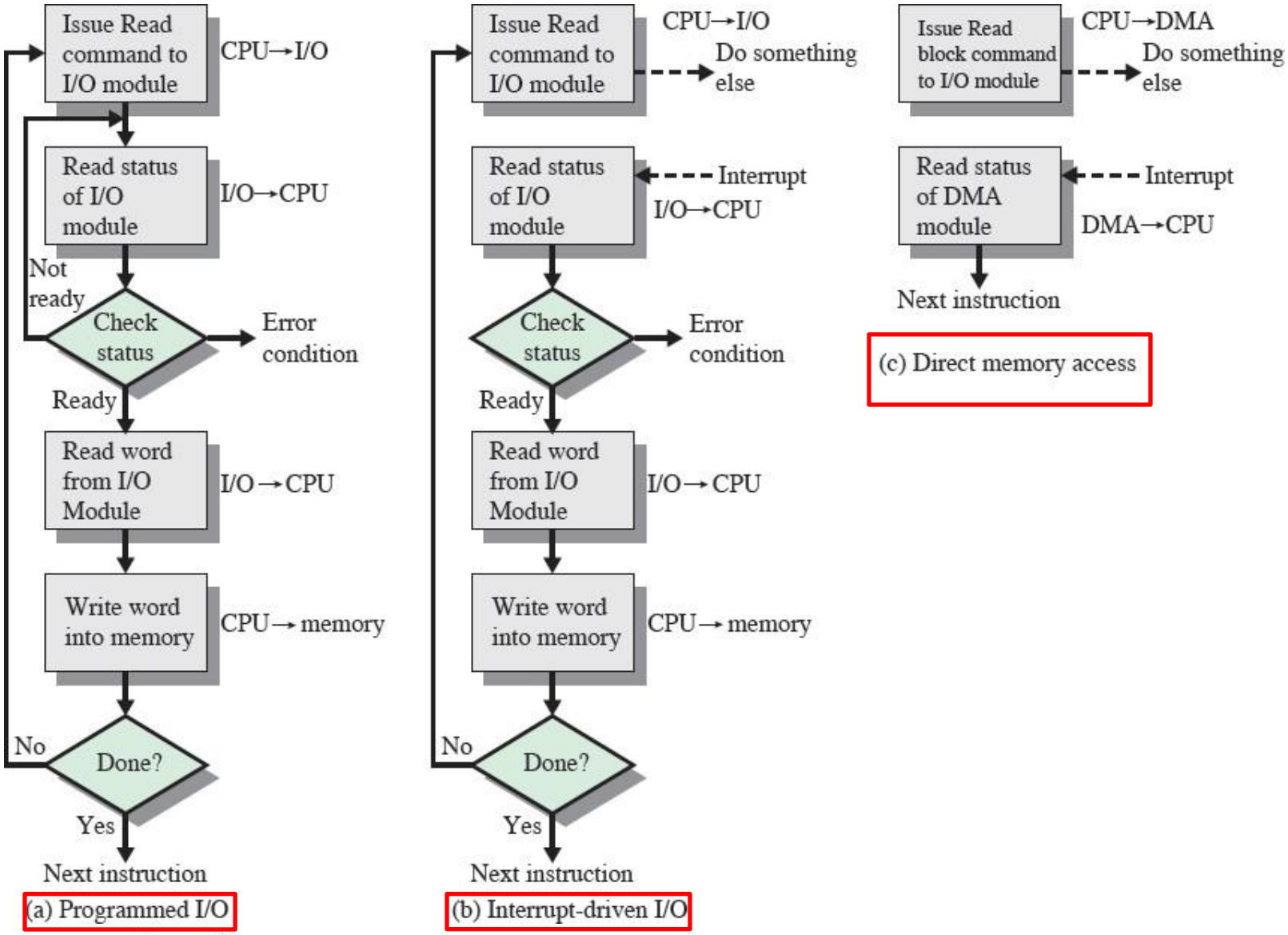
Entrada / Salida II: Técnicas de E/S I

Tres técnicas para la entrada de un bloque de datos



E/S programada con espera de respuesta

1. El procesador espera en un bucle hasta que el periférico esté listo para una transferencia.

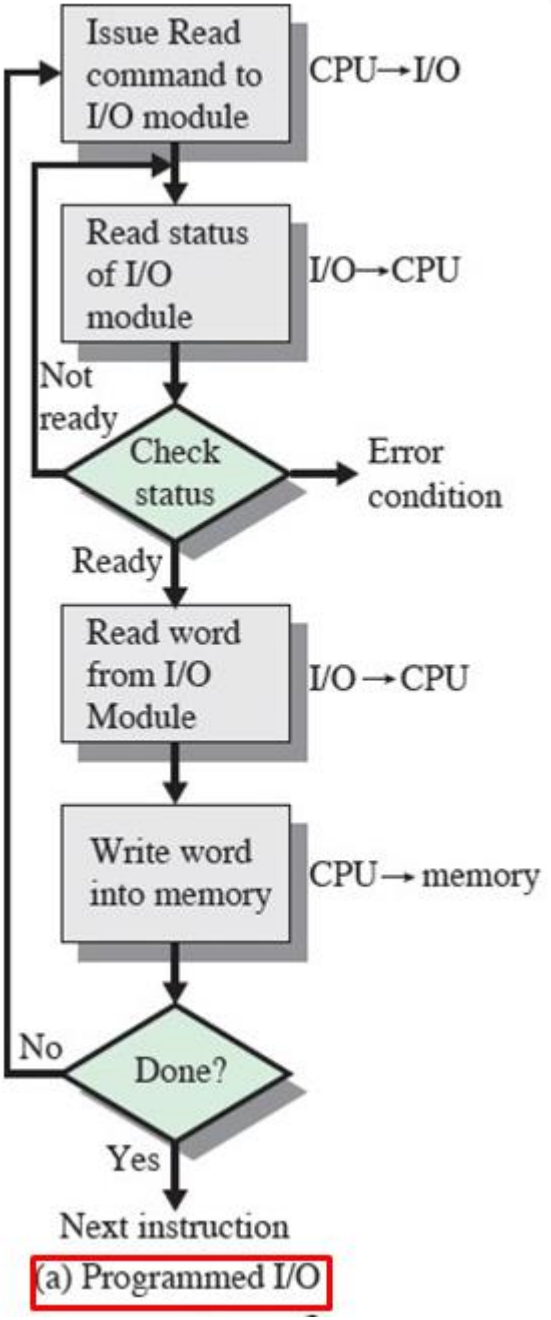
2. El periférico indica su disponibilidad en un registro de estado.

3. El procesador ejecuta un conjunto de instrucciones que mueven un dato entre el dispositivo de E/S y memoria.

Adaptación de la velocidad del procesador a la velocidad del periférico.

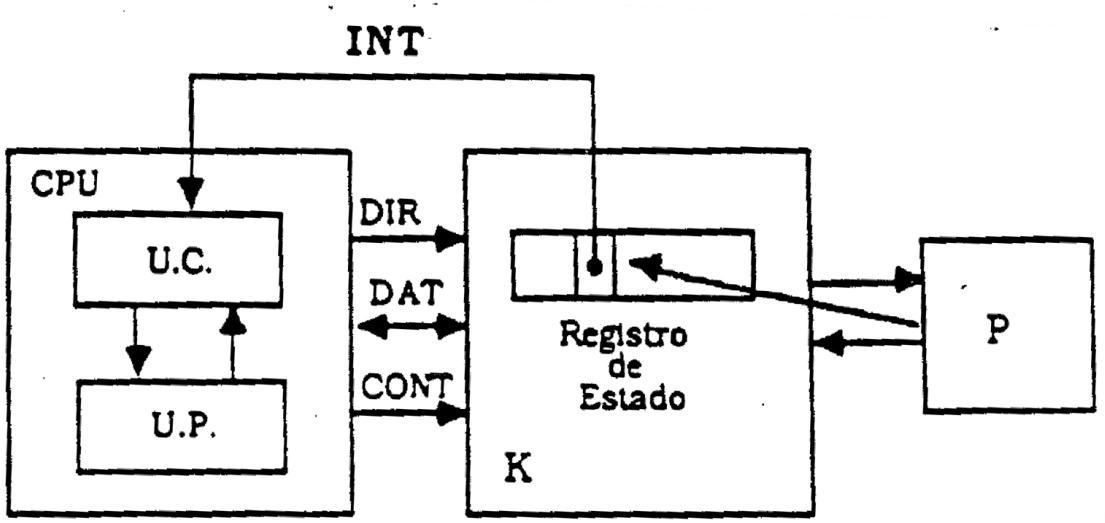
<u>Limitaciones de la E/S programada</u>

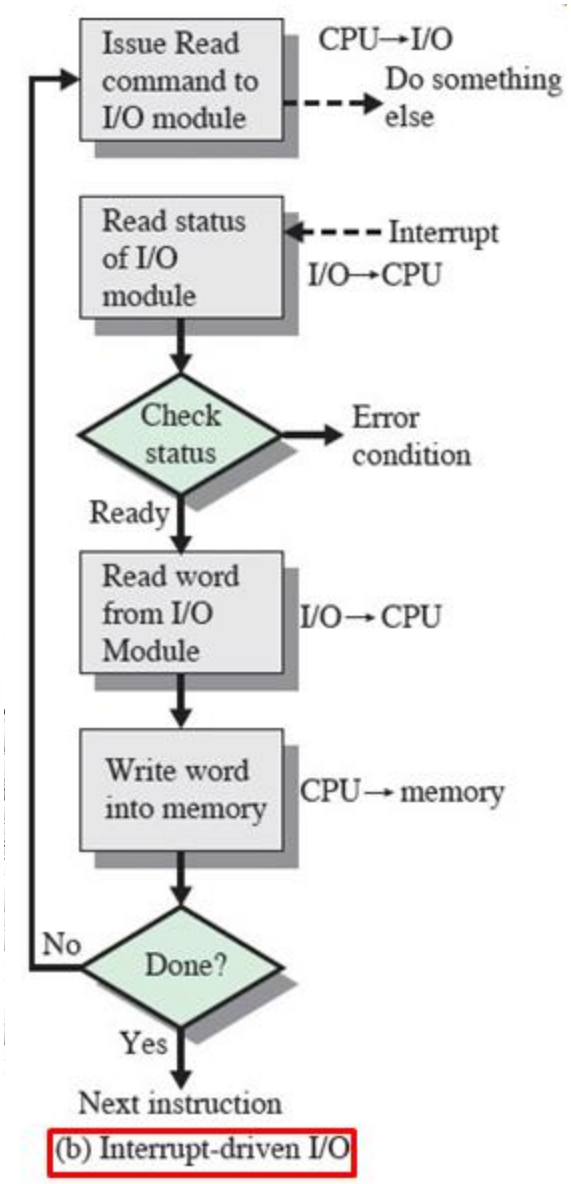
- Pérdida de tiempo
- Dificultad para atender a diferentes periféricos



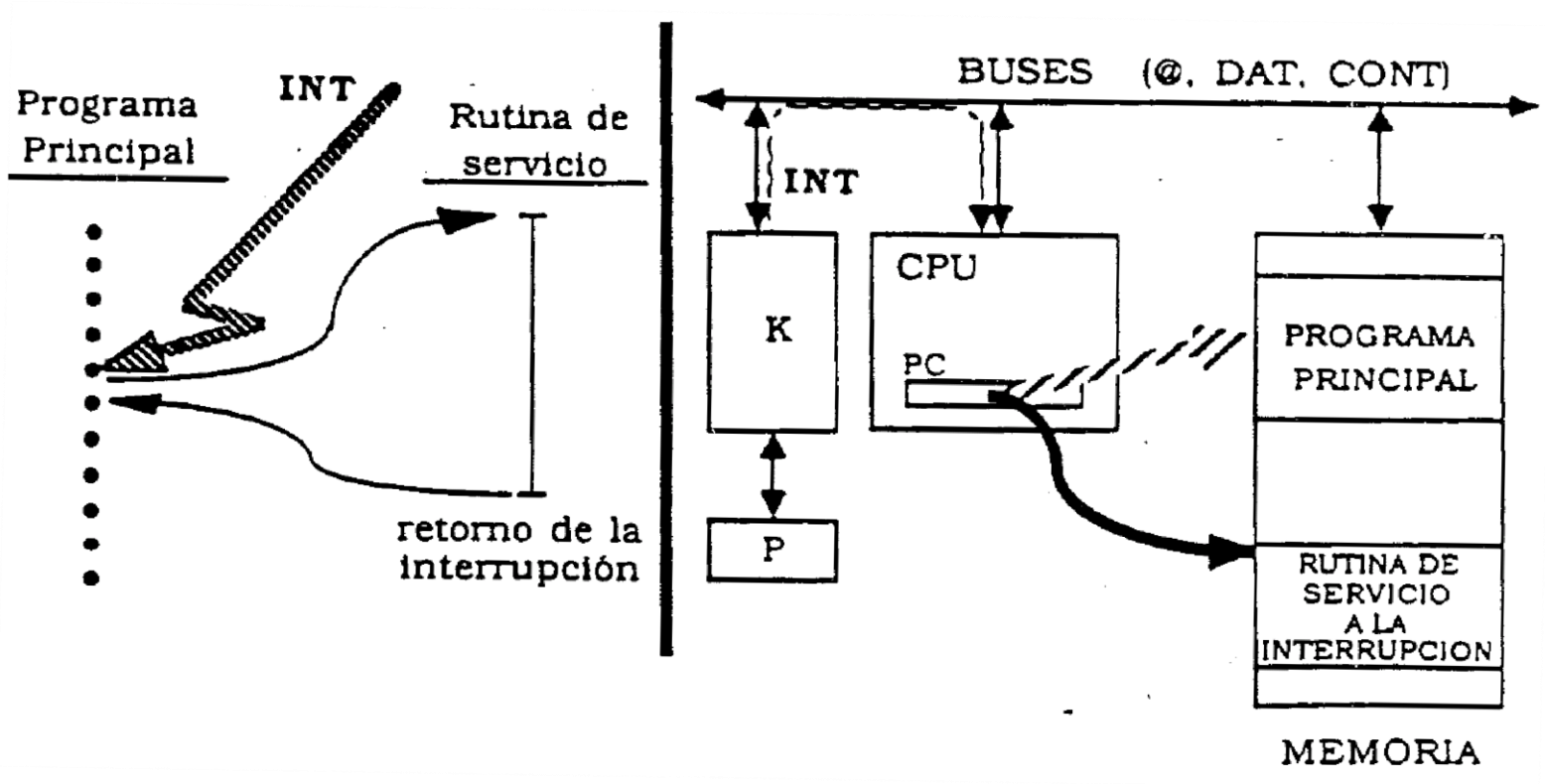
E/S dirigida por interrupción

Cuando el periférico está listo para una transferencia, activa un bit en el registro de estado (precisamente el que hay que analizar en la **sincronización por encuesta).**
Éste, a su vez, activa la señal de salida **INT** que está conectada a una de las entradas de la CPU.





Cuando la CPU detecta que la interrupción será servida, abandona la ejecución del programa que estaba ejecutándose y se pasa el control a un rutina que atenderá al dispositivo externo que provocó la interrupción.



La secuencia de acontecimientos que tiene lugar durante la **atención de una petición de interrupción** es, a grandes rasgos, la siguiente:

1. Detección de la petición
2. Salvar el estado del programa interrumpido
3. Identificación de la rutina a ejecutar (dependiente del dispositivo que interrumpe)
4. Ejecución de la rutina de atención a la interrupción
5. Retorno al programa interrumpido

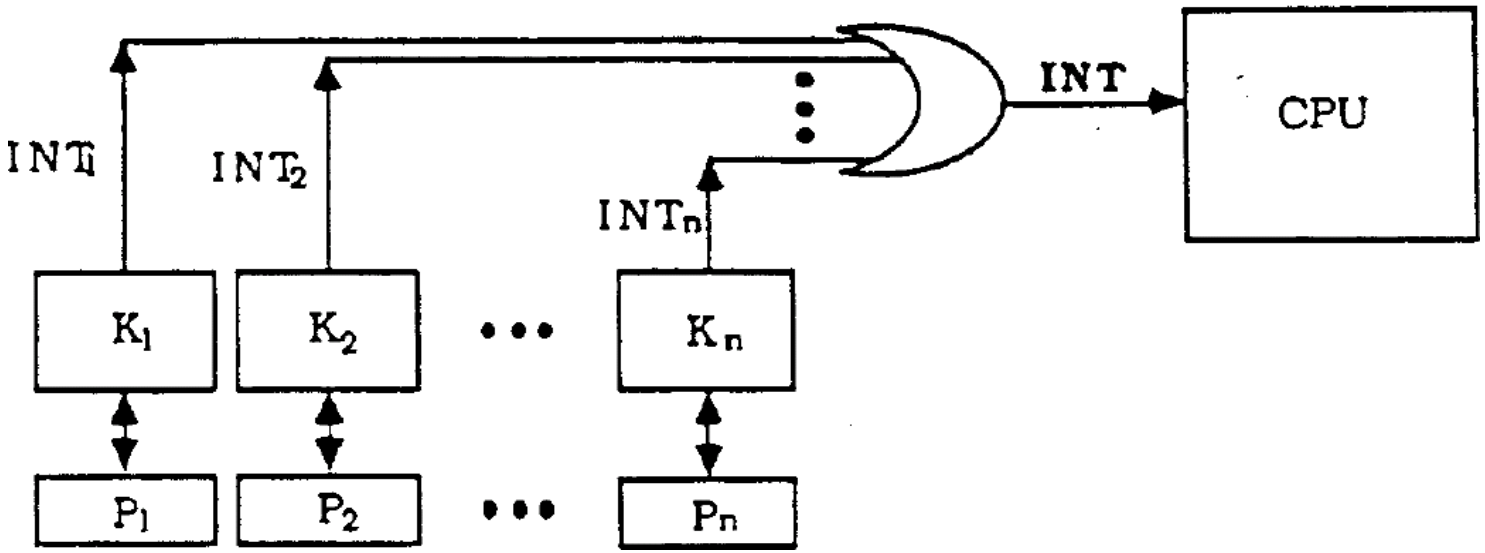
<u>Cuestiones que hay que resolver:</u>

- ¿Qué hacer cuando varios periféricos hacen peticiones de interrupción simultaneas?
- ¿Se debe permitir que algún periférico interrumpa a la CPU cuando la CPU está ejecutando la rutina de servicio de otro periférico?

1. Detección de la petición

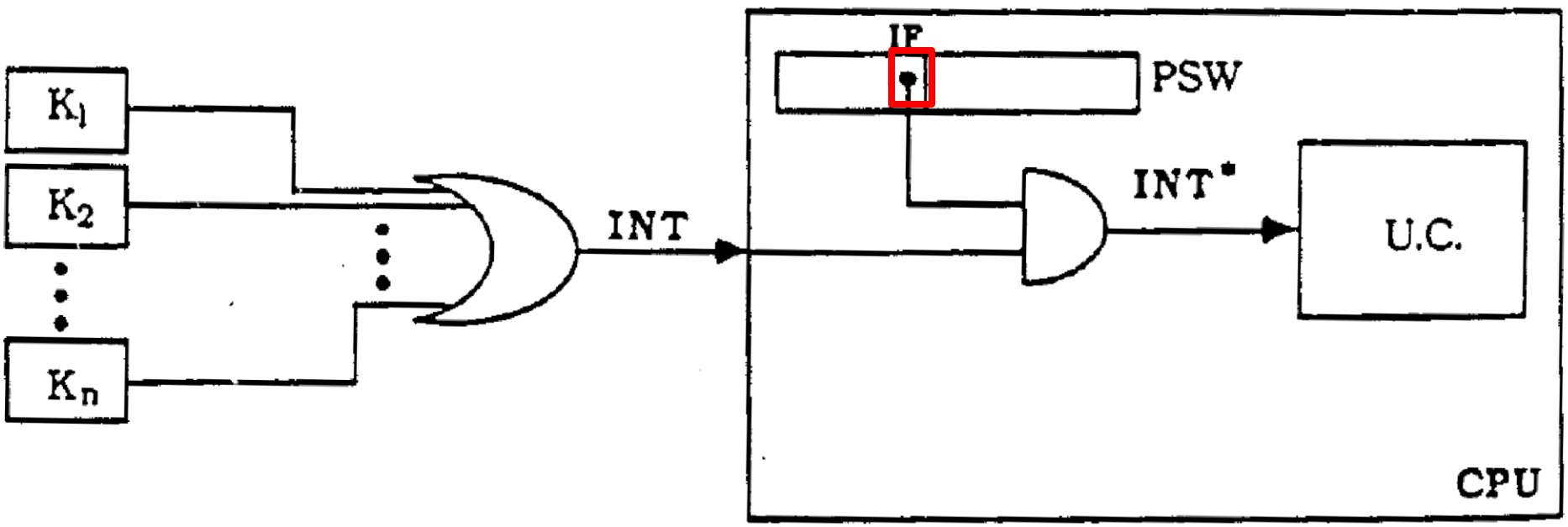
Una petición de interrupción consiste en la activación de alguna de las líneas de petición que llegan a la CPU.
Los dispositivos externos comparten estas líneas de petición de interrupción.

¿Cuándo detecta la CPU la petición de interrupción?
Cuando acaba la ejecución de una instrucción y antes de iniciar la búsqueda de la siguiente.



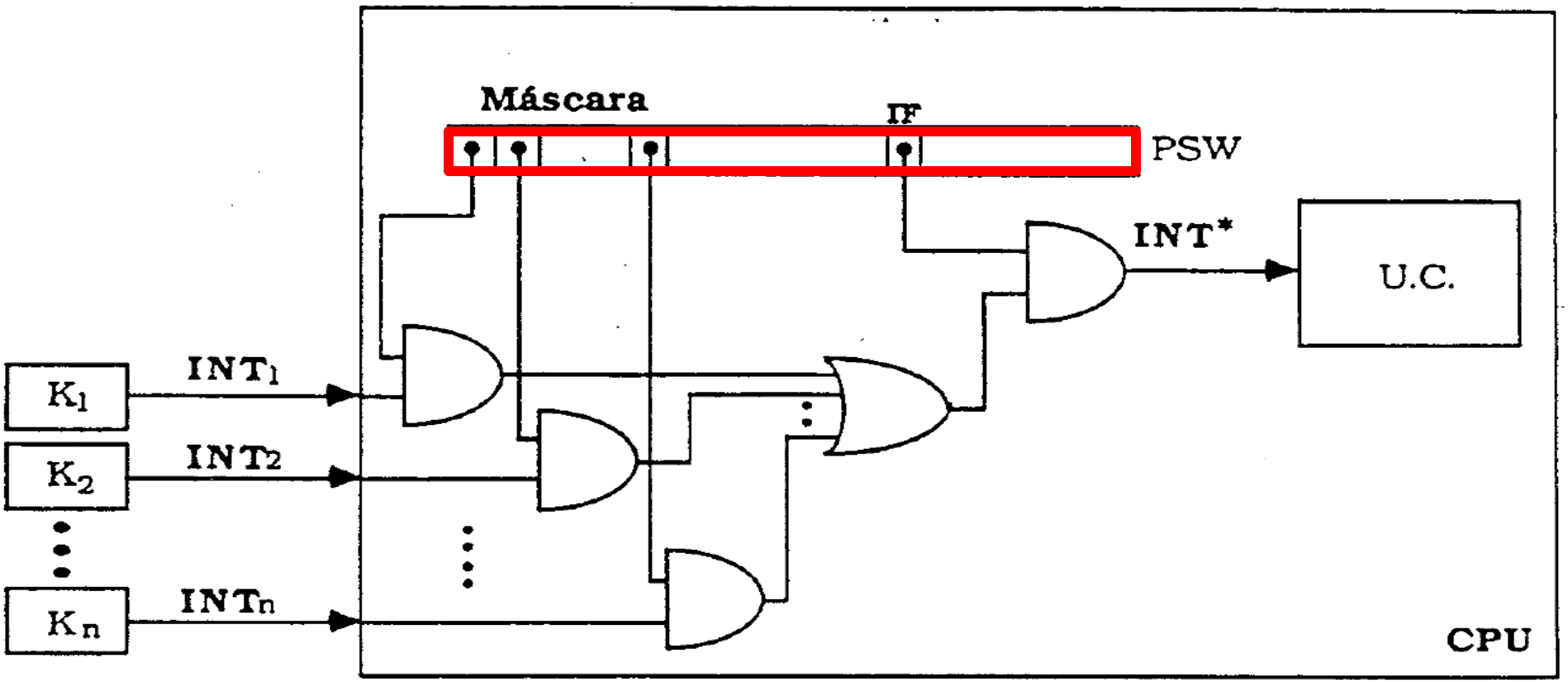
¿Las instrucciones deben estar siempre capacitadas?

Depende del **flag IF** de interrupción almacenado en el **PSW** (palabra de estado del procesador).
Si **IF = 0** las interrupciones están **inhibidas**; si **IF = 1** las interrupciones están **permitidas**.



Si existen varias señales de entrada de petición de interrupción, existe la posibilidad de inhibirlas todas (mediante IF) o inhibir un subconjunto de ellas mediante una máscara.

Existen instrucciones de L.M. para manipular la máscara y el bit IF.



2. Salvar el contexto del programa interrumpido

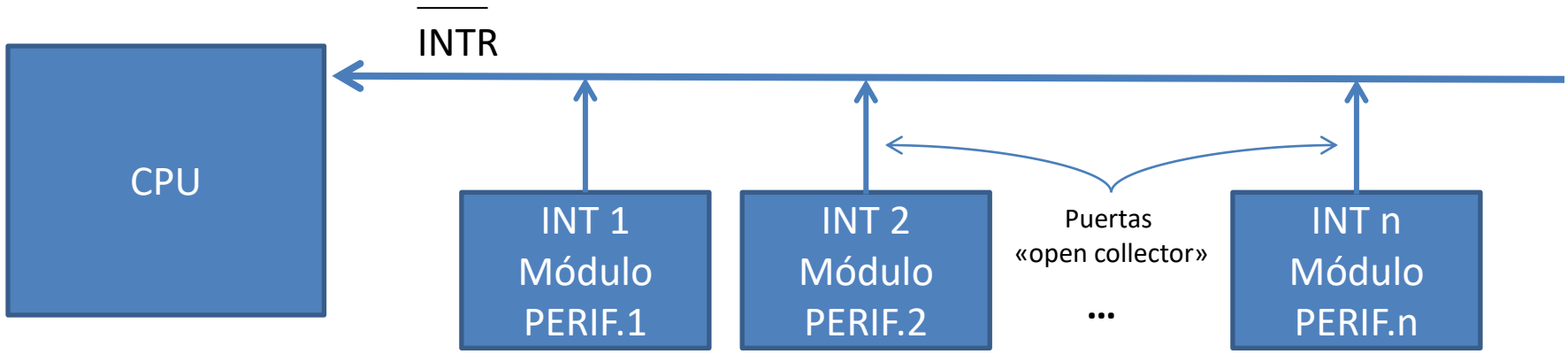
La mínima información que se ha de salvar automáticamente es el estado del programa que es: el **PC** y los **bits de condición de la palabra de estado del procesador**.
La rutina de atención debe salvar y recuperar adicionalmente el contenido de todos los registros que utiliza.
Si se permite el anidamiento de interrupciones, al igual que el anidamiento de subrutinas, se salva el estado del programa en la pila.

3. Identificación de la rutina a ejecutar

La identificación puede ser:
- Por **software:** por **encuesta** (*polling*)
- Por **hardware**: **por vector** (*interrupciones vectoradas*).

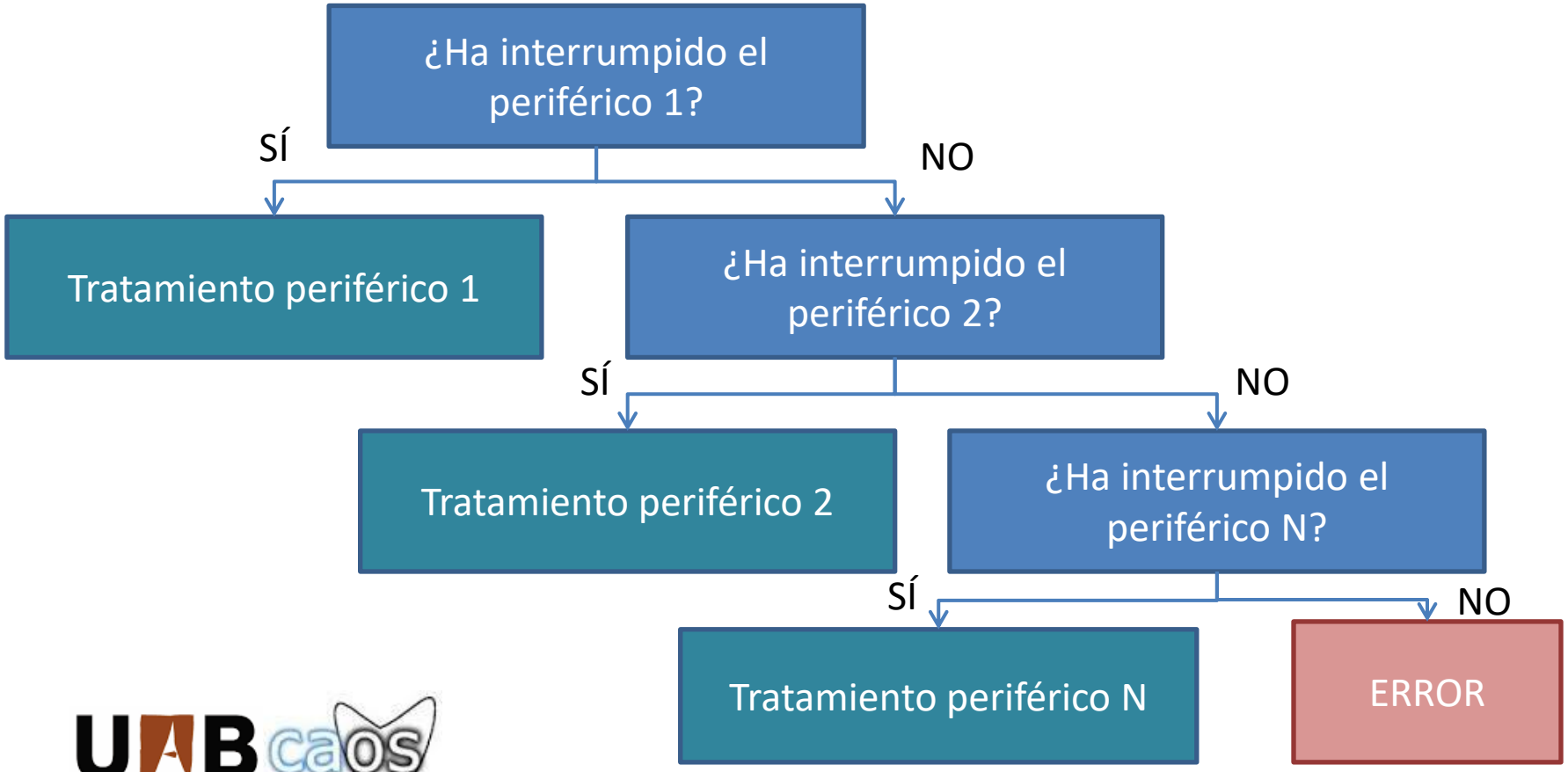
Identificación por software → Línea única o nivel único

Todos los periféricos comparten una sola línea de petición de interrupción ($\overline{\text{INTR}}$).
- Cuando un periférico hace una petición de interrupción, pone su línea de petición a 0.
- El valor de la línea $\overline{\text{INTR}}$ es la Y_lógica (Y cableada) de las salidas de petición de interrupción de todos los periféricos



Identificación por encuesta (*polling*)

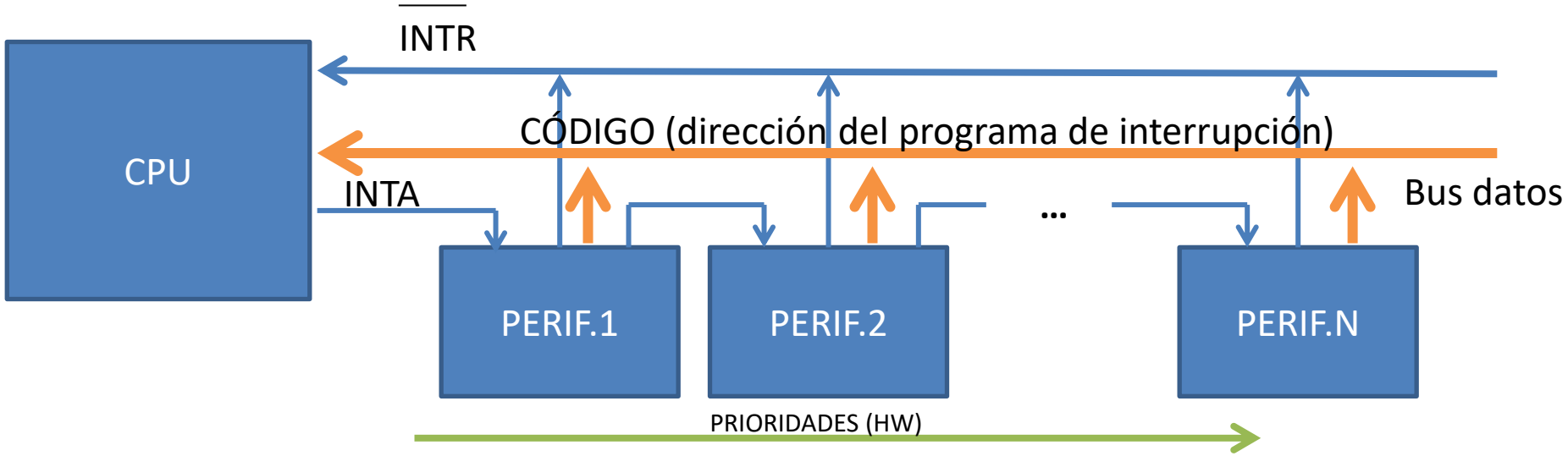
Se examina el registro de estado de cada periférico hasta hallar al que ha activado su línea de petición de interrupción.
La prioridad depende del orden del primer periférico por el que se pregunte.



UAB caos

Hardware: por vector (*interrupciones vectoradas*)

Mecanismo hardware por el cual el periférico envía a la CPU un código que, de forma directa o indirecta, determina el comienzo de su rutina de tratamiento de interrupción, es decir, el dispositivo puede identificarse directamente a la CPU



Secuencia de eventos:

1. El periférico que hace la petición provoca $\overline{\text{INTR}} \rightarrow 0$
2. La CPU activa la señal INTA (Interrupt Acknowledge)
3. Los periféricos que reciben INTA y no han interrumpido, propagan INTA al siguiente periférico
4. El periférico que ha interrumpido, cuando recibe INTA no la propaga y envía el código a la CPU (mecanismo DAISY-CHAIN).

Ese código (vector que se envía por el bus de datos) es un identificador usado por la CPU para acceder a una tabla donde se encuentra la dirección de la rutina a ejecutar (interrupción vectorada). El vector se interpreta como un puntero.

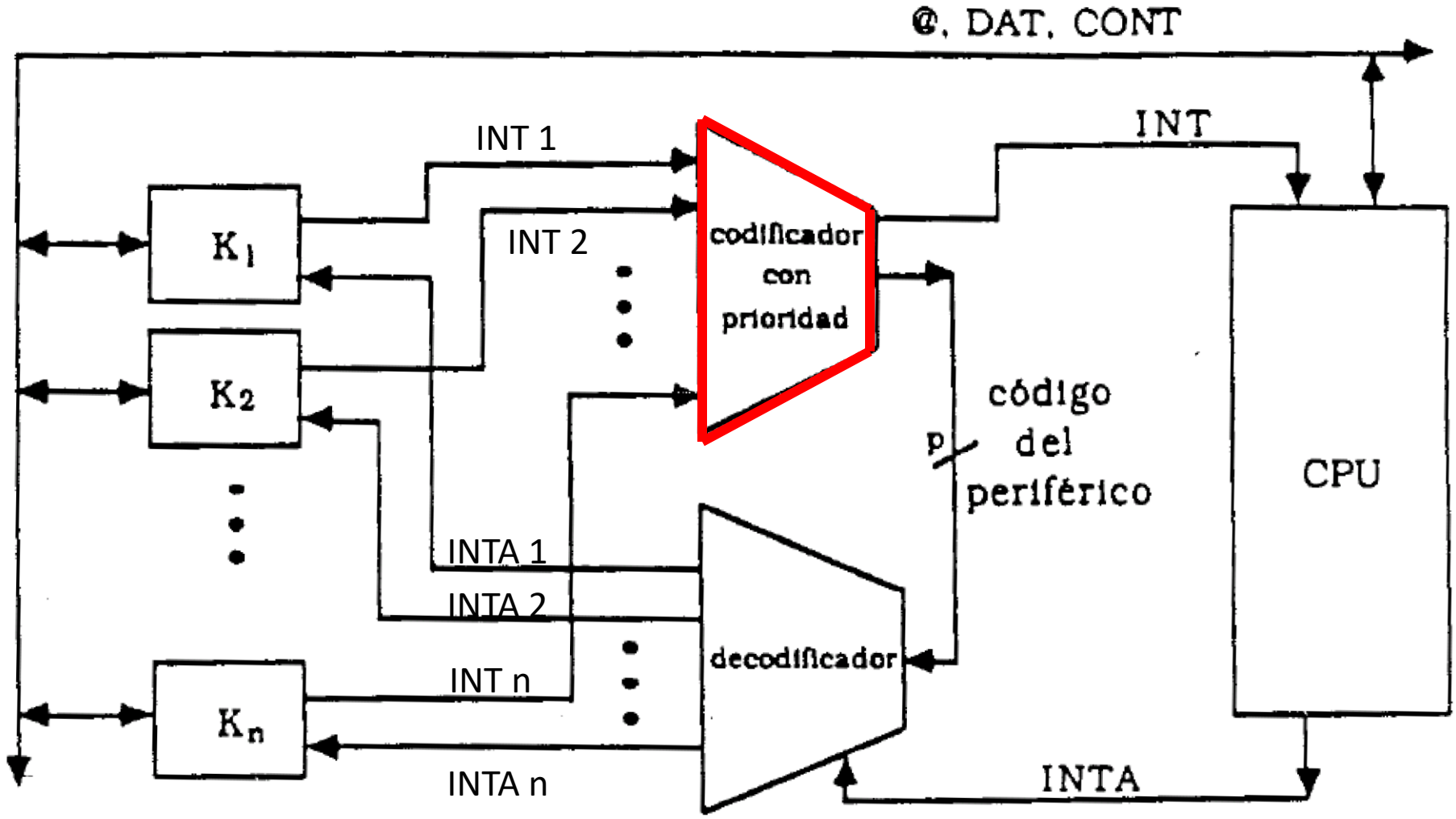
Conflictos de peticiones simultaneas por varia líneas

¿Qué pasa si varios dispositivos piden atención simultánea por varias líneas?

Habría que atender al dispositivo más prioritario.

Incluir en el hardware un codificador de prioridad, que cuando detecta que alguna de las entradas está activa, proporciona a su salida el código correspondiente a la entrada activa de mayor prioridad. Ese código es tomado por el decodificador que se encarga de enviar la señal de respuesta del procesador (INTA) a aquel periférico que tiene más prioridad entre todos los que solicitaron la interrupción.

* Codificador con prioridad:



4. Ejecución de la rutina de atención de la interrupción

5. Retorno al programa interrumpido

La última instrucción de la rutina es una instrucción especial **RTI** (retorno de interrupción) que restaura el estado del programa interrumpido (PC y PSW) de forma que pueda reanudarse su ejecución.

Interrupciones multinivel (anidamiento de interrupciones)

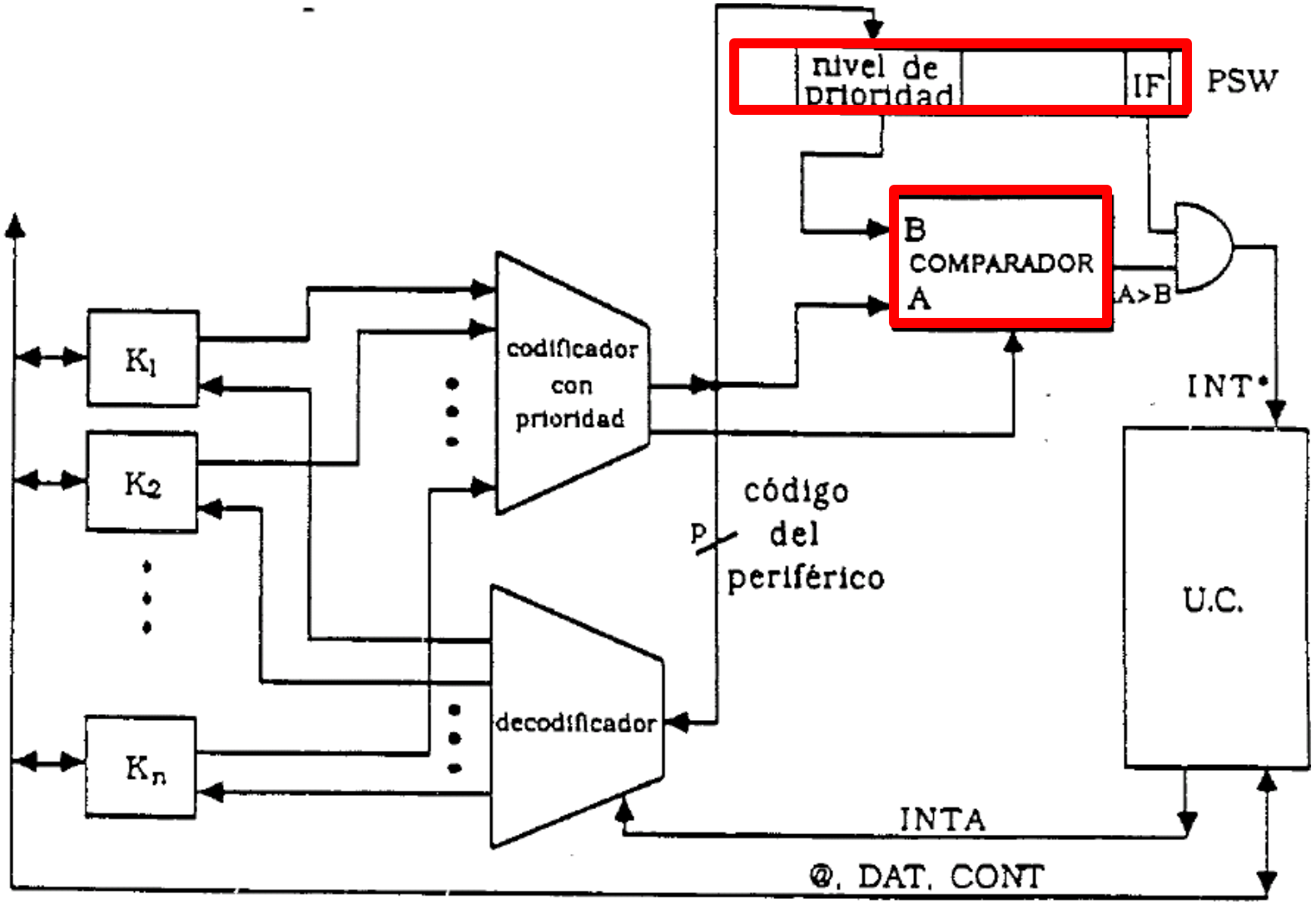
En la mayoría de los computadores, la CPU puede ser interrumpida cuando está ejecutando la rutina de atención a una interrupción previa. En este caso, se abandona la ejecución de la rutina y pasa a ejecutarse la rutina de atención a la nueva interrupción.

En estas máquinas se ordenan las posibles peticiones de interrupción según prioridades decrecientes.

Cuando se produce una interrupción de prioridad N, ésta será atendida si las interrupciones están permitidas y si en ese momento se está ejecutando una rutina de atención a una interrupción menos prioritaria.

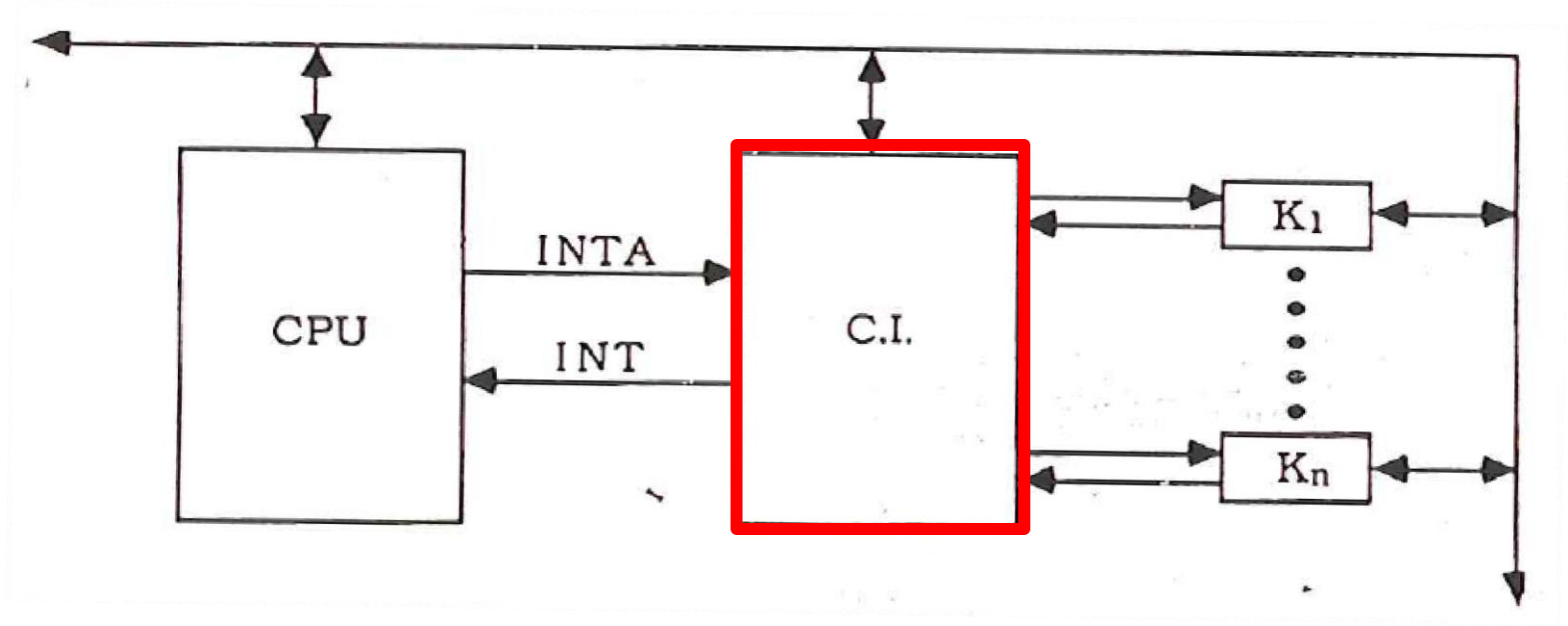
Es necesario ampliar la palabra de estado con información que codifique el nivel de prioridad de la rutina que se está ejecutando.

UAB caos

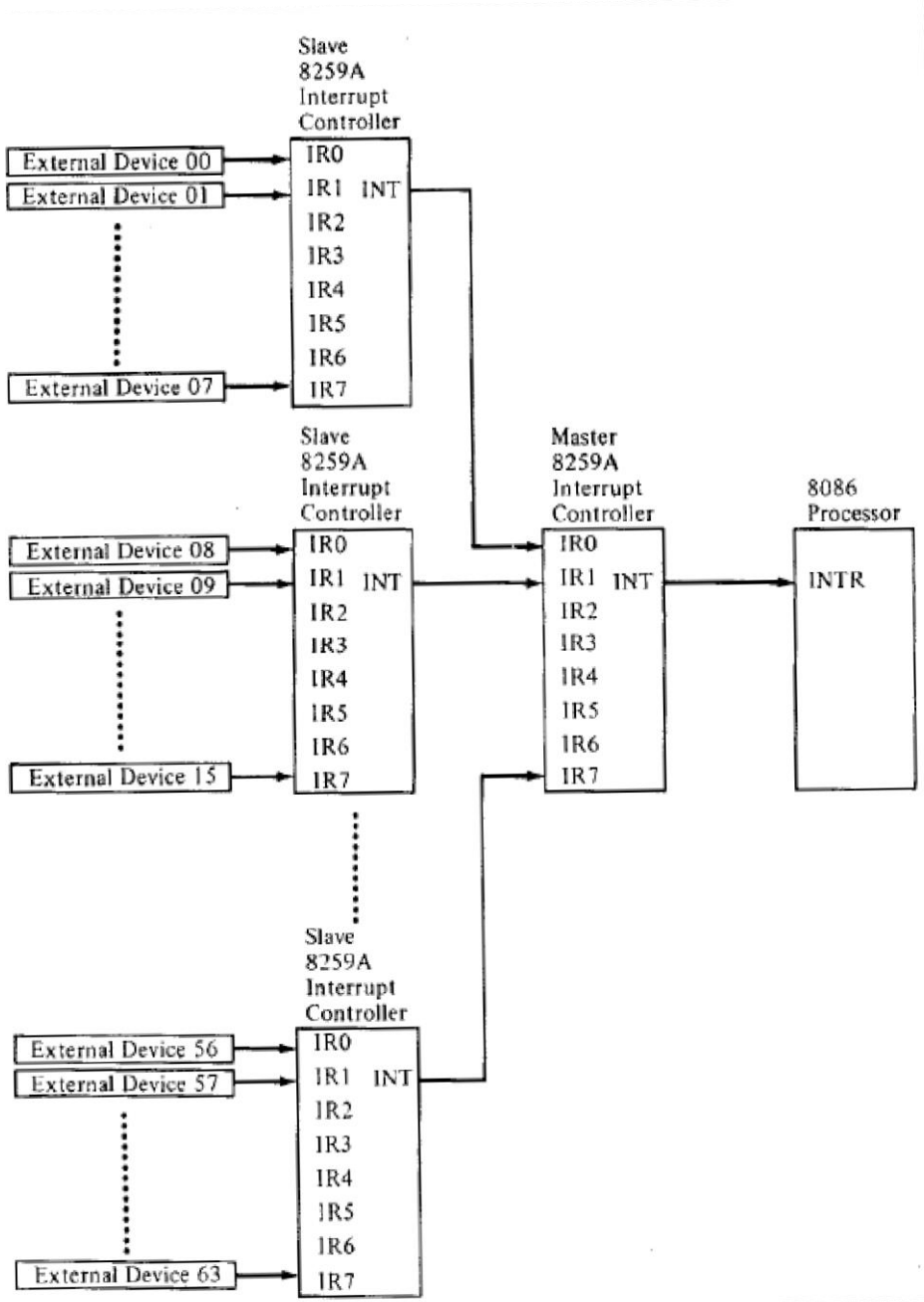

nivel de prioridad
IF
PSW
B
COMPARADOR
A
A>B
INT*
K1
K2
Kn
codificador con prioridad
código del periférico
P
decodificador
U.C.
INTA
@. DAT. CONT

Controlador de interrupciones

Se encarga de recibir cualquier petición de interrupción y de determinar, en función de las prioridades y de las peticiones pendientes y en curso, si la nueva petición debe ser servida



Encadenamiento de los controladores de interrupción



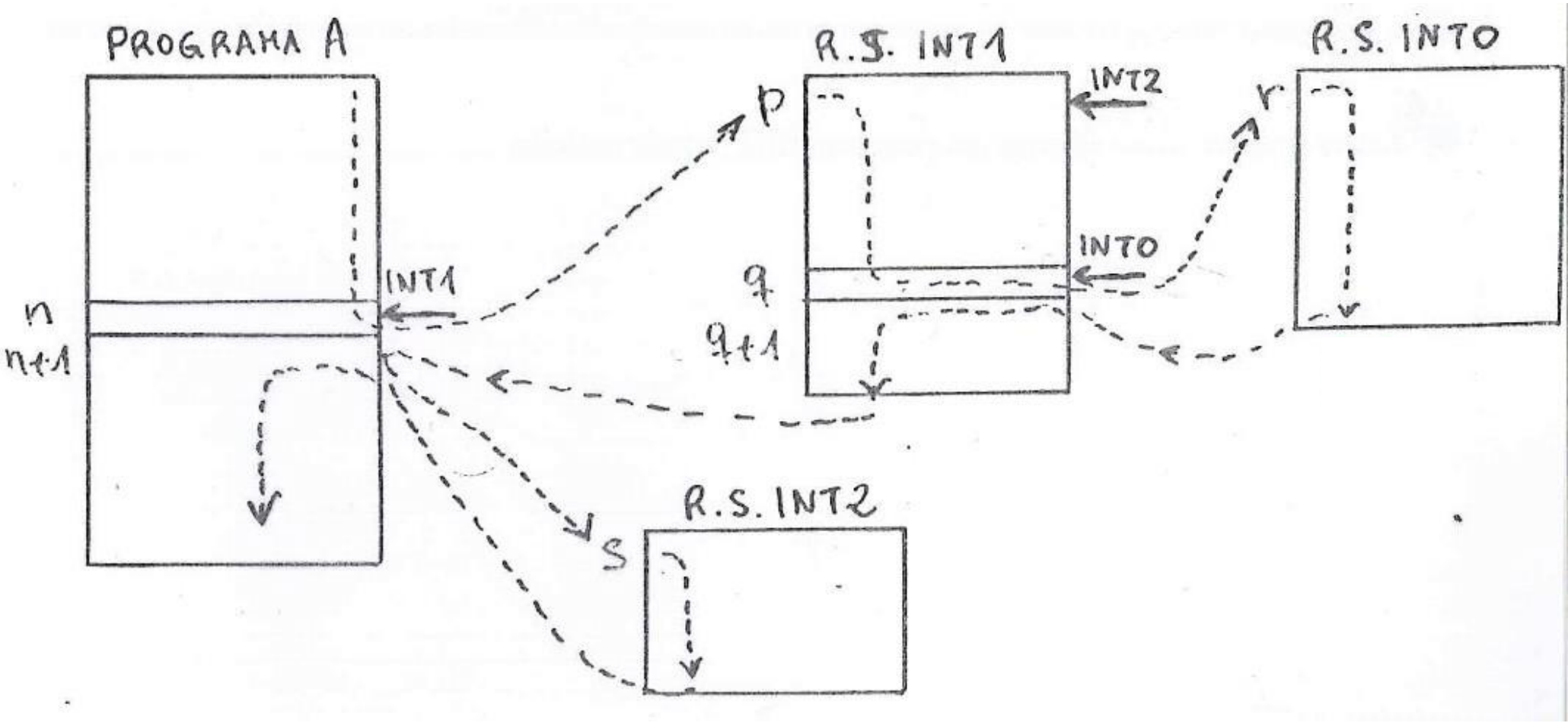
Ejercicio

Tenemos un sistema con 3 líneas de petición de interrupción: INT0, INT1 y INT2 con prioridad creciente. (INT0 más prioritaria).
Son enmascarables mediante 3 bits del registro de estado.
Si es 0 está activa y si es 1 está enmascarada(deshabilitada).
Una petición de interrupción enmascara a las demás líneas menos prioritarias(si se activa la INT0 las otras dos quedan inhabilitadas).

Se producen 3 peticiones de interrupción en el siguiente orden:
INT1, INT2, INT0 teniendo en cuenta que las dos últimas se producen cuando se está ejecutando la rutina de atención de INT1.

Interrupciones de anidamiento

Ejemplo: tratamiento y servicio de interrupciones



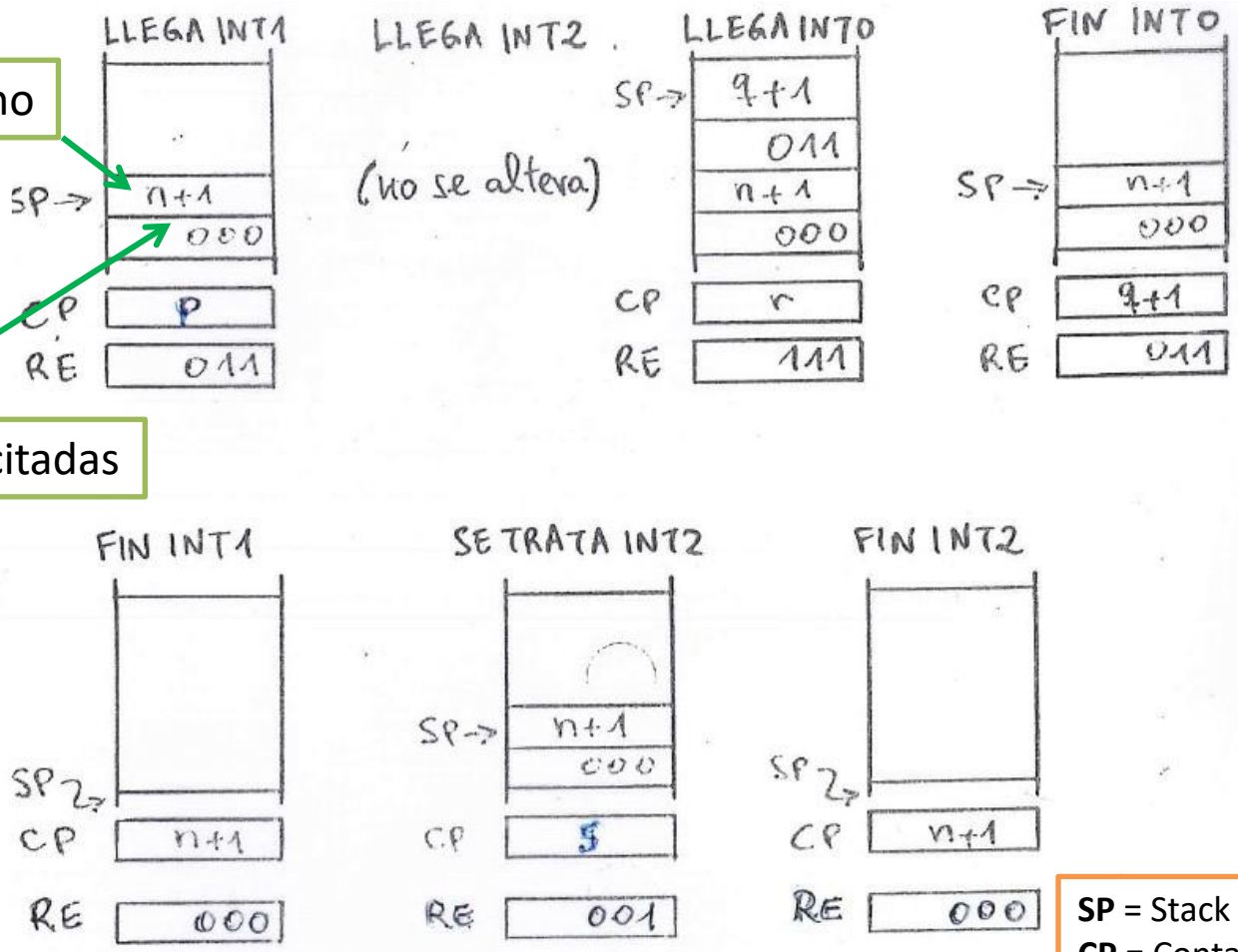
Interrupciones de anidamiento

Ejemplo: tratamiento y servicio de interrupciones

Evolución de stack y registros

Dirección de retorno

Interrupciones capacitadas



SP = Stack pointer
CP = Contador de programa
RE = Registro de estado